Bacheloroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for IKT og realfag

Fivelstad, Joakim
Kleppe, Henrik Aarnes
Løvlid, Adrian Hjelmeland
Torp, Hans Jørgen

Ansiktsgjenkjenning for Låsesytem

Facial Recognition for Access Control System

Bacheloroppgave i Automatiseringsteknikk
Veileder: Strazdins, Girts
Mai 2019

NTNU
Kunnskap for en bedre verden

Fivelstad, Joakim
Kleppe, Henrik Aarnes
Løvlid, Adrian Hjelmeland
Torp, Hans Jørgen

Ansiktsgjenkjenning for Låsesytem

Facial Recognition for Access Control System

Bacheloroppgave i Automatiseringsteknikk
Veileder: Strazdins, Girts
Mai 2019

Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for IKT og realfag

NTNU
Kunnskap for en bedre verden

Fullmakt til elektronisk publisering av oppgaven

Forfatter(ne) har opphavsrett til oppgaven. Det betyr blant annet enerett til å gjøre verket tilgjengelig for allmennheten (Åndsverkloven §2).
Alle oppgaver som fyller kriteriene vil bli registrert og publisert i Brage HiM med forfatter(ne)s godkjennelse.
Oppgaver som er unntatt offentlighet eller båndlagt vil ikke bli publisert.

Jeg/vi gir herved NTNU i Ålesund en vederlagsfri rett til å gjøre oppgaven tilgjengelig for elektronisk publisering:	☒ **ja**	☐ **nei**
Er oppgaven båndlagt (konfidensiell)? (Båndleggingsavtale må fylles ut) - Hvis ja: **Kan oppgaven publiseres når båndleggingsperioden er over?**	☐ **ja** ☐ **ja**	☒ **nei** ☐ **nei**
Er oppgaven unntatt offentlighet? (inneholder taushetsbelagt informasjon. Jfr. Offl. §13/Fvl. §13)	☐ **ja**	☒ **nei**

Dato: 20.05.2019

Obligatorisk egenerklæring/gruppeerklæring

Den enkelte student er selv ansvarlig for å sette seg inn i hva som er lovlige hjelpemidler, retningslinjer for bruk av disse og regler om kildebruk. Erklæringen skal bevisstgjøre studentene på deres ansvar og hvilke konsekvenser fusk kan medføre. Manglende erklæring fritar ikke studentene fra sitt ansvar.

Du/dere fyller ut erklæringen ved å klikke i ruten til høyre for den enkelte del 1-6:		
1.	**Jeg/vi erklærer herved at min/vår besvarelse er mitt/vårt eget arbeid, og at jeg/vi ikke har brukt andre kilder eller har mottatt annen hjelp enn det som er nevnt i besvarelsen.**	☒
2.	**Jeg/vi erklærer videre at denne besvarelsen:** • ikke har vært brukt til annen eksamen ved annen avdeling/universitet/høgskole innenlands eller utenlands. • ikke refererer til andres arbeid uten at det er oppgitt. • ikke refererer til eget tidligere arbeid uten at det er oppgitt. • har alle referansene oppgitt i litteraturlisten. • ikke er en kopi, duplikat eller avskrift av andres arbeid eller besvarelse.	☒
3.	**Jeg/vi er kjent med at brudd på ovennevnte er å betrakte som fusk og kan medføre annullering av eksamen og utestengelse fra universiteter og høgskoler i Norge, jf. Universitets- og høgskoleloven §§4-7 og 4-8 og Forskrift om eksamen §§14 og 15.**	☒
4.	**Jeg/vi er kjent med at alle innleverte oppgaver kan bli plagiatkontrollert i Ephorus, se Retningslinjer for elektronisk innlevering og publisering av studiepoenggivende studentoppgaver**	☒
5.	**Jeg/vi er kjent med at høgskolen vil behandle alle saker hvor det forligger mistanke om fusk etter høgskolens studieforskrift §31**	☒
6.	**Jeg/vi har satt oss inn i regler og retningslinjer i bruk av kilder og referanser på biblioteket sine nettsider**	☒

Forord

Denne bacheloroppgaven er gitt av Avento AS og utført av fire studenter fra Automatiseringsteknikk ved NTNU i Ålesund. Vi vil rette en spesiell takk til vår veileder Girts Strazdins for meget god hjelp og veiledning gjennom hele prosjektperioden. Vi ønsker også å takke Avento for en svært interessant oppgave, samt alle ansatte ved NTNU i Ålesund som har hjulpet oss med små og store problemer underveis i prosjektet. Til slutt vil vi takke hverandre for et meget godt og konstruktivt samarbeid gjennom hele semesteret. Det har vært en veldig lærerik og givende prosess.

Sammendrag

Ansiktsgjenkjenning blir stadig mer aktuelt i en moderne verden. Teknologien er i stor utvikling hos selskaper som Apple og Microsoft, som tilbyr en veldig pålitelig ansiktsgjenkjenning i smarttelefoner og nettbrett. Disse løsningene er ofte begrenset til én person per enhet.

Dette prosjektet går ut på å lage et program med ansiktsgjenkjenning for et låsesystem, som er tilpasset store brukergrupper. Oppgaven er gitt av Avento som selv har utviklet en simpel løsning, og koblet denne rett på et relé som åpner en dør i deres lokaler. Denne løsningen er enkel å lure og de ønsket derfor å se hvordan en gruppe studenter ville ta fatt på problemstillingen. De la derfor dette frem som en bacheloroppgave for NTNU i Ålesund.

Vi har tatt i bruk skytjenester som Microsoft Azure og BioID for å oppnå høy sikkerhet og stabilitet. Dette gjøres ved at det tas to bilder hvor vi ser etter en naturlig bevegelse i ansiktet før vi foretar selve ansiktsgjenkjenningen. Det ble utviklet en visuell veileder som skal gi brukeren informasjon og veiledning gjennom prosessen.

Løsningen har blitt et raskt, autonomt og modulbasert system som sjekker for liveness, foretar ansiktsgjenkjenning og gir mulighet for tofaktorautentisering. I tillegg er det utviklet et administratorprogram som gjør det mulig å administrere systemet og justere sikkerhetsnivået.

Innholdsliste

Figurer

Terminologi

Begreper

Wrapper Forenkler tilgangen på andre programvarebibliotek eller funksjoner i ett operativsystem.

Spoofing Noe man gjør når man prøver å lure et system ved å forfalske sin egen identitet eller framstå som å være noen andre enn seg selv.

Point cloud En punktsky, brukt når det snakkes om kartlegge noe i 3D.

Liveness Brukes i denne sammenhengen for å sjekke om en person er ekte.

Paradigme En problemløsning som blir akseptert som en "standard" for løsning innen samme vitnskap.

Notasjon

σ sigma, vanlig å bruke for standardavvik

π pi

V Volt

Forkortelser

IEEE Institute of Electronical and Electronic Engineers

OS Operating System

SDK Software Development Kit

IR Infrared

AI Artificial Intelligence

VR Virtual Reality

IDE Integrated Development Environment

GUI Graphical User Interface

API Application Programming Interface

TCP Transmission Control Protocol.

UDP User Datagram Protocol.

IP Internet Protocol

GPIO General-Purpose Input/Output

UART Universal Asyncronous Receiver/Transmitter

USB Universal Serial Bus

PIR Passive Infrared

AIR Active Infrared

REST Representational State Transfer

ROI Region of interest

TOTP Time-based One-time Password

QR Quick Response

JSON JavaScript Object Notation

ASCII American Standard Code for Information Interchange

ACK Acknowledge Message

GIF Graphics Interchange Format

HMAC Hash-based Message Authentication Code

RAM Random Access Memory

SSD Solid State Drive

Kapittel 1 - Introduksjon

1.1 Bakgrunn for oppgaven

Bakgrunnen for prosjektet er at Avento ønsker seg et system hvor de ansatte kan låse seg inn i lokalene ved bruk av ansiktsgjenkjenning. De ønsker å stå fram som en nytenkende bedrift, som er først på markedet med nye idéer. Dette vil være med å gi et imponerende inntrykk for kunder som er på besøk. De har selv utviklet en enkel løsning som er koblet direkte på reléet til døren. Denne løsningen er sårbar mot spoofing og de ønsket derfor å se hvordan en gruppe studenter ville løse dette problemet.

1.1.1 Hvorfor valgte vi denne oppgaven?

Før vi skulle velge oppgave for bachelorprosjektet hadde gruppen en klar formening om hvordan en aktuell oppgave burde fremstå. Vi ønsket også å velge en oppgave som er gitt av en lokal bedrift. Valget falt til slutt på denne oppgaven fordi den trigget interessen til alle i gruppen, den er fremtidsrettet og innebærer arbeid med aktuell teknologi. I tillegg forventet vi å få bruk for flere fag vi har vært innom gjennom tre år på NTNU i Ålesund.

1.1.2 Om Avento

Avento er et konsulentselskap med spisskompetanse innen IT-rådgivning, systemutvikling, Business Intelligence, og digital markedsføring. De har lang erfaring med design og utvikling av forretningskritiske datasystemer.[1] Avento jobber med systemutvikling og samarbeider med mange store aktører for å utvikle teknologiske løsninger.

1.2 Prosjektbeskrivelse

Oppgaven vår er å lage et program med ansiktsgjenkjenning for et låsesystem. Et viktig krav fra Avento var at det ikke skulle være mulig å lure systemet. I Avento sin versjon av låsesystemet var det mulig å komme seg inn ved å bruke bilder av ansatte. Avento ønsket at vi eliminerte denne svakheten. Det var ønskelig med et brukergrensesnitt hvor man kunne legge til og administrere brukere av systemet. Det ble også foreslått at vi brukte en skjerm som kan ønske brukere velkommen når de låser seg inn. Utover dette så ble det ikke satt noen videre krav til hvordan vi skulle løse oppgaven. Avento ga oss relativt frie tøyler og ønsket å se hvordan vi ville løse oppgaven.

1.2.1 Mål for oppgaven

Utifra den definerte prosjektbeskrivelsen satt vi oss noen mål og krav til utførelsen av prosjektet. Det er viktig at prosessen med ansiktgjenkjenningen går så hurtig som mulig. Vi ønsker at systemet skal være så raskt og responsivt at brukere av systemet skal foretrekke ansiktsgjenkjenning over andre løsninger som nøkkelkort og kode. Løsningen må være av høy sikkerhet, og det burde ikke være mulig å lure systemet. Vi vil tilby tofaktorautentisering som kan brukes når det er ønskelig, for eksempel utenom arbeidstid, eller som en offline-løsning. Vi ønsker også å lage en løsning som er modellerbar, med potensiale til utvidelse.

1.2.1.1 Krav til funksjonalitet

Vi satt oss følgende krav til funksjonalitet:

- Legge til og slette personer.
- Legge til og slette persongrupper.
- Liste personer i systemet.
- Mulighet for å bruke systemet når internett tilgang er nede.
- Mulighet for flere sikkerhetsnivåer.

- Visuell veileding for bruker i form av illustrasjonsskjerm.

1.2.1.2 Krav til sikkerhet

Vi satt oss følgende krav til sikkerhet:

- Eliminere mulighet for spoofing.
- Mulighet for tofaktorautentisering.
- Passordbeskyttet administrasjon av systemet.
- Kryptert kommunikasjon med skytjenester.

1.2.2 Utfordringer og mulige problemer

Vi anser sikkerheten som vår største utfordring med oppgaven. Det er mange elementer knyttet til sikkerhet, og vi ønsker et system som baserer seg mest mulig på ren ansiktsgjenkjenning. Det kan bli utfordrende å utvikle et system som er så raskt som vi ønsker fordi det er mange elementer som påvirker tiden. Derfor vil oppbyggingen av programmet være en viktig faktor. Prosesseringskraft til valg av hardware vil også påvirke tiden.

1.3 Brukergrupper

Vi har to brukergrupper i forbindelse med prosjektet. Den ene brukergruppen er knyttet opp mot selve låsessystemet, mens den andre gruppen er knyttet opp mot administrasjon av systemet.

1.3.1 Brukere av ansiktsgjenkjenningen

Den største brukergruppen er de som skal bruke systemet. Dette inkluderer alle som skal inn dørene. I teorien kan hvem som helst bruke låsesystemet, men kun de med tilgang vil kunne åpne døren.

1.3.2 Brukere som administrerer systemet

Brukergruppen som skal administrere systemet har tilgang til et administrasjons program. Programmet er todelt. En del som gjør det mulig å legge til personer, mens den andre administrerer systemet. Hvem som har tilgang reguleres av bedriften selv.

1.4 Tidligere arbeid med ansiktsgjenkjenning

Det finnes få kommersielle låsesystemer som bruker ansiktsgjenkjenning. Per dags dato er de fleste løsningene begrenset til én-til-én. Det vil si at det er få løsninger som tar for seg problemet med å holde på flere personer som skal ha tilgang til samme system.

Ansiktsgjenkjenning har vært i stor utvikling de siste årene, og det er mange selskaper som tilbyr tjenester med denne teknologien. Apple har på noen av sine telefoner gjort det mulig å kunne låse opp, gjøre kjøp og bekrefte betalinger ved bruk av ansiktsgjenkjenning (Face ID). Telefonen bruker et dybdekamera til å kartlegge 3D-punkter i ansiktet.[2]

Facebook DeepFace er facebook's system for ansiktsgjenkjenning. Facebook lager en ansiktsmodell av alle sine brukere. Når et bilde blir lastet opp på facebook, blir det sammenlignet mot en database med disse modellene til de finner en som er lik. De hevder å ha en nøyaktighet på 97,35%. Facebook jobber med å automatisk tagge personer i bilder som blir lastet opp.[3]

1.5 Gruppemedlemmer

Gruppen vår består av fire fulltidsstudenter som studerer automatiseringsteknikk ved NTNU i Ålesund. Vi har alle erfaring med både Java og Arduino C som programmeringsspråk. Vi har hatt fag som bildebehandling, datakommunikasjon, kunstig intelligens og sanntidsprogrammering som vil være relevante i utførelsen av oppgaven. Vi er også vant med å jobbe med elektronikk og mikrokontrollere.

Vi kommer til å bevege oss inn på felt som er nye for oss. Dette inkluderer blant annet nye programmeringspråk og kommunikasjonsprotokoller. Vi mangler en del datarelaterte fag som kunne gjort prosessen enklere med tanke på utviklingsmetodikk og fremstilling av diverse software-diagrammer.

1.6 Roller

Vi ønsker å jobbe ganske tett sammen på gruppen og føler derfor ikke et behov for å ha en utpekt gruppeleder til dette prosjektet. Vi vil fordele arbeidet slik at alle får jobbe litt med alt. På denne måten vil hele gruppa sitte igjen med god kunnskap om hvert aspekt rundt oppgaven og stiller derfor også sterkere når det gjelder å hjelpe hverandre med en gitt problemstilling.

1.7 Lesere av rapporten

Målgruppen for rapporten er alle som ønsker innsyn i hvordan vi har utført prosjektet sett fra et utviklingsperspektiv. Først og fremst sensor og veiledere, men også oppdragsgivere, potensielle arbeidsgivere og eventuelle utviklere som vil ta løsningen videre. Lesere av rapporten bør kjenne til ulike ord og uttrykk relatert til software utvikling.

1.8 Om rapporten

Rapporten vil i hovedsak ta for seg nødvendige teoretiske begreper og metoder, samt gi en utfyllende forklaring på resultatene vi oppnår. Til slutt drøfter vi resultatene og trekker en konklusjon basert på hele prosjektet.

1.8.1 Rapportens struktur

Resten av rapporten har følgende struktur:
Kapittel 2 - Teori: Gir en introduksjon til det teoretiske grunnlaget for oppgaven, samt hva som er gjort på dette feltet tidligere. Her tar vi for oss viktige konsepter vi har tatt i bruk.

Kapittel 3 - Materialer og metoder: Inneholder en oversikt over hvordan vi har jobbet gjennom prosjektet og hvilke valg vi har gjort underveis. Her står det også om hvilke komponenter, tjenester og metoder vi har vurdert og hvilke vi har tatt i bruk.

Kapittel 4 - Resultat: Her blir resultatene presentert og vi forklarer hvordan løsningen er utformet. Vi begrunner også hvorfor vi endte opp med nettopp denne løsningen.

Kapittel 5 - Drøfting: Inneholder en drøfting av prosjektarbeidet. Vi tar for oss sterke og svake sider med løsningen, og hva som kunne vært gjort annerledes.

Kapittel 6 - Konklusjon: I konklusjonen forklarer vi kort om resultatene og de drøftingene vi har gjort. Her kommer vi også med tanker om hvordan fremtidige løsninger kan utvikles.

Kapittel 2 - Teori

I dette kapittelet vil vi gjøre rede for grunnleggende teori som er nødvendig for å forstå resten av rapporten. Her snakker vi blant annet om ansiktsgjenkjenning, bildebehandling og kommunikasjonsprotokoller vi har brukt.

2.1 Ansiktsgjenkjenning

Ansiktsgjenkjenning er en form for biometrisk gjenkjenning. "Biometrisk gjenkjenning, gjenkjenning av et menneske etter ansikt, fingeravtrykk, iris, blodkarmønstre, stemme osv. ved hjelp av metoder (algoritmer) som regner på digitalt registrerte biologiske egenskaper."[4]. Disse algoritmene blir ofte brukt for å samle informasjon til databaser, som igjen blir brukt i systemer for identitetskontroll. Denne teknologien har kommet så langt at det blir brukt i blant annet passkontroller.

2.2 Kunstig intelligens

Kunstig intelligens er en teknikk man bruker for å gi datasystemer mest mulig intelligent atferd. Datasystemene skal kunne løse problemer og lære av sine erfaringer. De skal kunne observere diverse miljøer og ta valg som løser dens oppgave på best mulig måte. Felles for slike systemer er at de "trenes opp" til å ta rette valg. Med det så menes det at et dataprogram blir utsatt for store mengder data. Programmene analysere dette og kan i ettertid komme fram til egne valg/avgjørelser som er bedre enn tidligere.[5]

2.3 Bildebehandling

Viola–Jones object detection framework

Viola-Jones object detection framework er et rammeverk som er beskrevet i artikkelen "Rapid Object Detection using a Boosted Cascade of Simple Features" [6]. Rammeverket inneholder metoder for å kunne klassifisere forskjellige objekter, men er mest utbredt på deteksjon av ansikter. Det har en høy treffsikkerhet, fungerer i sanntid, og er laget for å skille objekter fra andre. Rammeverket baserer seg på enkle matematiske utregninger basert på pikselintensitet i en bestemt region av bildet. Ved å se etter forskjellige "egenskaper" i et bilde kan man, ved å kombinere disse, konkludere med at det finnes et ansikt i bildet. Disse "egenskapene" blir kalt Haar Features, og baserer seg på et par likheter man kan finne hos de fleste mennesker. Eksempel på "haar features" finner du i figur 2.1. En modell som trenes på slike "haar features" kalles en "haar-cascade". Under finner man noen eksempel på "haar features" hos mennesker.

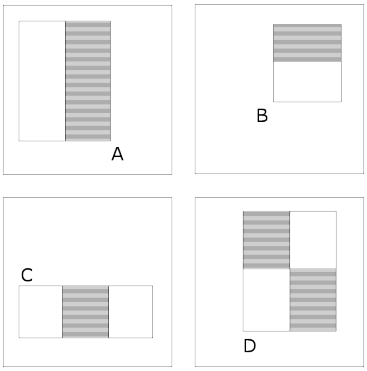


Figure 2.1: Haar Features

- Øyeregionen er lysere en øverste kjeve-parti.
- Neseroten, og frem til nesen, er lysere enn øynene.
- Øyebryn og panne.
- Munn er mørkere enn hake.
- Munn er mørkere enn tenner.

Det som skiller Viola-Jones Object detection fra tidligere arbeid på området er implementasjonen av integralmatriser. Dette baserer seg på at man evaluerer rektangulære funksjoner i sanntid. Dette gjør at man kan gå igjennom bilde-matrisen flere ganger uten å måtte regne

på hver enkel pikselverdi hver gang. Ønsket område av pikselverdier kan ved bruk av integralmatrisen kalkuleres mye hurtigere, ved å trekke fra de uønskede delene av matrisen siden disse allerede er regnet ut. Rammeverket kommer inkludert i OpenCV biblioteket.[7]

Dilasjon

Dilasjon (eng. dilate) er en form for bildebehandling for å utvide bestemte figurer som finnes i inngangsbildet. Utvidelseoperasjonen bruker vanligvis et strukturelement for å undersøke og utvide.[8]

Gaussian blur

Gaussian blur er en metode for å skape uklarhet i bilde, basert på en gaussisk funksjon. Metoden er mye brukt for å redusere støy i bilde og redusere detaljer, og er ofte det første steget i en rekke av bildebehandlinger. Matematisk er bruken av Gaussian blur det samme som å utføre folding av bildematrisen med en gaussisk funksjon. Ved bruk av Fourier-transformasjon når man skal bruke Gaussian-metoden gir dette effekten av å redusere de høye frekvensen i bildet, og fungerer som et lavpassfilter.
Den matematiske fremstillingen av Gaussian blur for hver enkelt piksel i et bilde kan skrives som:

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}$$

hvor x er distansen fra opprinnelsen på den horisontale aksen og σ er standardavviket for Gaussian fordelingen.[9]

Bilderskling

Bilderskling er en bildebehandlingsmetode som endrer verdien til en piksel basert på en gitt terskel. Den enkleste formen for terskling er å erstatte hver piksel med en sort piksel dersom verdien er mindre enn en bestem konstant T, eller erstatte pikselen med en hvit piksel dersom verdien er over konstanten.[10]

Konturer

En kontur er en lukket fremstilling av en figur laget ved bruk av punkter eller linje-segmenter. Dette representer omrisset til et objekt i et bilde. Hvis det finnes synlige konturer på innsiden av omrisset vil det produsere flere omriss rundt de aktuelle konturene. Når vi har funnet konturene til objektene i et bilde kan man detektere verdier som antall, formen til objektet, eller måle størrelsen.[11]

2.4 Objektorientert programmering

Objektorientert programmering er et paradigme innen programmering av datamaskiner.[12] De følgende punktene er hovedprinsippene bak OOP:

- Objekt - legge data og funksjonalitet i et objekt for å gi modularitet til programmet
- Abstraksjon - gjør det mulig for programmereren å ignorere noen detaljer under implementasjon
- Innkapsling - skjule tilstanden til et objekt fra andre deler av koden. På denne måten forhindrer man uønsket utvendig påvirkning på tilstanden til objektet.
- Polymorfi - et objekt kan oppføre seg som et annet objekt så lenge grensesnittet tillater dette
- Arv - forenkle arbeidet med polymorfi og innkapsling ved å muliggjøre opprettelse av mer spesialiserte utgaver av andre objekter. For eksempel "Gutt" er en mer spesialisert utgave av objektet "Menneske".

2.4.1 Class

I objektorientert programmering er en klasse en oppskrift for å opprette et objekt. I klassen definerer man tilstander, oppretter metoder objektet kan utføre og velger hva slags variabler objektet skal holde på.

2.4.2 Interface

Interface forteller noe om hva et objekt kan gjøre. En klasse kan implementere et *interface* og på denne måten få mulighet til å utføre metoder som *interfacet* inneholder.

2.4.3 Coupling

Coupling er prinsippet som omhandler hvordan et objekt direkte påvirker tilstanden eller oppførselen til et annet objekt. God grad av Coupling gjør koden mer lesbar og modulerbar, noe som gjør koden enklere å forandre.

2.4.4 Cohesion

God cohesion blir gitt utifra om metodene i en klasse representerer det samme. I en kode med god cohesion er lesbar kode og gjenbruk økt, mens kompleksiteten blir senket.

2.4.5 Multithreading

Multithreading handler om at flere oppgaver kan kjøres samtidig. I Java blir dette omtalt som flere tråder. En tråd er definert som en egen oppgave på innsiden av et program, og kan kjøres samtidig som andre tråder.[13]

2.5 Kommunikasjonsprotokoller

I vårt system har vi tatt i bruk et par forskjellige kommunikasjonsprotokoller, blant annet HTTP Request, TCP og seriell kommunikasjon. Vi går nærmere inn på hver enkelt her.

2.5.1 REST API (HTTP request)

REST, eller Representational State Transfer, er en arkitektonisk stil for å standardisere kommunikasjon mellom datasystemer på nett. Dette gjør det enklere for systemene å kommunisere med hverandre. REST-systemer skilles ut ved at de er tilstandsløse. Dette vil si at klienten ikke trenger å vite hvilken tilstand serveren er i og omvendt. Det betyr at klientkoden kan forandres

når som helst uten at serveren påvirkes av det eller at serverkoden kan forandres uten at klienten påvirkes. I tillegg betyr dette at både server og klient kan forstå alle beskjeder som mottas, uavhengig av om de har sett tidligere beskjeder fra samme kilde. Denne tilstandsløsheten håndheves gjennom bruk av *resources*, ressurser, istedenfor kommandoer. Ressursene kan sammenlignes med substantiv: de kan være et hvilket som helst objekt, dokument eller ting man ønsker å lagre eller sende til andre tjenester. Siden REST-systemer samhandler gjennom standardoperasjoner på ressurser er de heller ikke avhengig av å måtte implementere grensesnitt.

Så lenge begge sider vet hvilket format beskjedene skal sendes i kan de holdes modulære og separate. Dette gjør at REST-systemer er svært fleksible og enkle å ta i bruk. Når man bruker REST-grensesnittet vil ulike klienter treffe de samme REST-endepunktene, utføre de samme handlingene og dermed få den samme responsen.

Alt dette er med på å gjøre REST-systemer eller REST-applikasjoner veldig pålitelige, raske, skalerbare og modulære. Kommunikasjonen mellom klient og server skjer ved at en klient sender en *request*, forespørsel, til en server for å hente ut eller modifisere data på serveren. En typisk forespørsel består av et HTTP-verb (GET, POST, PUT, DELETE) som forteller hvilken handling som skal utføres, en *header* som holder på informasjon fra klienten om forespørselen, en *path*, sti, til en ressurs og en eventuell *message body* som inneholder data. Dette vil vi gå mer inn på under avsnittet om material og metode.[14]

2.5.2 TCP/IP

TCP/IP er en forkortelse for Transmission Control Protocol/Internet Protocol. Denne nettverksprotokollen gir ende-til-ende datakommunikasjon som angir hvordan data skal pakkes, addresseres, overføres, sendes og mottas.

2.5.2.1 TCP

Denne nettverksprotokollen forsikrer om en forbindelsesorientert, pålitelig overføring av data. Dette gjøres ved en "threeway-handshake". Dette er en godkjenning av tilkoblingen som videre gjør at protokollen garanterer at data blir sendt og mottatt uten tap. I tillegg sjekkes all data for feil, noe som gjør at TCP er litt tregere enn andre protokoller. TCP er mest egnet når overføringshastighet kommer i andre rekke, og prioriteten er at all data kommer frem i riktig rekkefølge.[15]

2.5.2.2 IP

Internet Protocol er den viktigste kommunikasjonsprotokollen vi har for viderekobling av datagram på tvers av nettverk. Hovedoppgaven til IP er å levere *packets* fra kilde til mottaker basert på IP-adresse. IP i seg selv er derfor en upålitelig måte å sende datagram, ettersom det er trådløst uten noen form for autentisering av om datagram eller *packets* er kommet fram eller i hvilken rekkefølge. Derfor kombineres den ofte med TCP for å oppnå sikker transport av data.[16]

2.5.3 Seriell kommunikasjon (UART)

Seriell kommunikasjon er en kommunikasjonsprotokoll der det blir sendt bytes (8bit), høy og lav. Dette er en gammel og relativt enkel kommunikasjonsprotokoll. Det trengs to ledninger for å sende data over UART, en RX (reciever) og en TX (transmitter). På Arduino Uno er det en brikke som oversetter mellom UART og USB Dette gjør det mulig å kommunisere med datamaskiner gjennom USB.[17]

2.5.3.1 USB3.0

USB er en universell seriebus som har blitt en standard for kommunikasjon mellom datamaskiner og eksterne enheter. USB erstatter flere tidligere grensesnitt som serielle porter, parallelle porter og strømforsyning. USB3.0 har en hastighet på 5 Gigabit per sekund og har mulighet til å sende og motta data samtidig.[18]

2.5.4 JavaScript Object Notation

JSON er et format for utveksling av data. Det er enkelt for mennesker å lese og skrive, og det er enkelt for datamaskiner å analysere og generere. JSON støttes av Java og er et ideelt språk for datautveksling. En JSON-utveksling kan inneholde tall, tekst, boolske verdier, tabeller, objekter og null.[19]

2.5.5 ASCII

American Standard Code for Information Interchange er en standard for utveksling av tekst mellom datamaskiner. ASCII kan gjøre om 7 bit med data til kode. Det vil si at man kan kode 128 mulige verdier. ASCII har et tegnsett som består av store og små bokstaver, tall og en del andre vanlige tegn, som for eksempel "!, ?, , ()".[20]

2.5.6 QR

En Quick Response kode er en matriseformet strekkode, vist i figur 2.2. En QR-kode bruker fire kodingsmoduser: numerisk, alfanumerisk, byte/binær og kanji. Det er en lesbar etikett som inneholder informasjon som kan leses av fra en bildebehandlingsenhet.[21]

Figure 2.2: Slik ser en QR-kode ut

2.6 Informasjonssikkerhet

Informasjonssikkerhet omhandler hovedsaklig tre begreper: konfidensialitet, integritet og tilgjengelighet. Dette betyr henholdsvis at informasjonen ikke blir kjent for uvedkommende, at informasjonen ikke blir endret utilsiktet eller av uvedkommende, og at informasjonen er tilgjengelig ved behov.[22]

2.6.1 HOTP

HOTP er en algoritme som som brukes for å generere et engangspassord, fra en "hemmelig" nøkkel og et gitt tidspunkt på formen *UNIX EPOCH TIME*. Dette er antall sekunder som har gått etter 00:00:00 Torsdag, 1 januar 1970.[23][24]

2.6.2 TOTP

TOTP er en algoritme for et tidsbasert engangspassord. Algoritmen genererer et engangspassord som må brukes innen en viss tidsperiode før passordet er ugyldig. TOTP bruker HOTP algoritmen som er kort beskrevet i 2.6.1. TOTP erstatter telleren i HOTP, med en ikke-avtagende verdi basert på den nåværende tiden. $TOTP_{value}$(K) = $HOTP_{value}$(K, CT), der tidstelleren, CT, er et tall som teller antall runder, TX, basert på forskjellen mellom *UNIX EPOCH TIME*, T, og et tidsintervall T0. Alle verdier er oppgitt på samme form som *UNIX EPOCH TIME*.[25] CT kan da skrives som:

$$C_T = \frac{T - T_0}{T_X}$$

2.6.2.1 Base32

Base32 er en notasjon hovedsaklig for koding av binære tall, men kan også brukes til å kode binær tekst til ASCII. Base32 bruker et tegnsett av bokstavene A-Z og tallene 2-7. Base32-format kan brukes til å lage hemmeligheter i QR-koder.[26]

2.6.3 HTTPS

HTTPS er en sikret utgave av HTTP. HTTPS tilbyr autentisering og kryptering av kommunikasjon i forbindelse med kommunikasjon på nett. En HTTPS-sesjon blir kryptert enten ved bruk av SSL-protokollen eller TLS protokollen, og tilbyr beskyttelse mot lytting, samt uønsket forandring på de sendte dataene.

Graden av beskyttelse avhenger av hvor korrekt implementasjonen av HTTPS er i nettleseren, tjeneren og hvilken krypteringsalgoritme som er brukt.

Kapittel 3 - Materialer og metode

Dette kapittelet forteller hvordan vi har jobbet gjennom prosjektet og hvilke valg vi har gjort underveis. Her står det også om hvilke komponenter, tjenester og metoder vi har vurdert og hvilke vi har tatt i bruk.

3.1 Utviklingsprosess

I denne seksjonen skal vi ta for oss hvordan vi gikk frem da vi designet og utviklet alle deler av systemet. Det finnes mange forskjellige systemutviklingsmodeller, men felles for dem alle er at de forsøker å gi utviklingen en systematisk tilnærming. Ved starten av prosjektet opprettet vi en fremgangsplan i Asana og synkroniserte den med en tidslinje i Instagantt. Se vedlegg for komplett fremgangsplan.

3.2 Utviklingsmetodikk

Valg av metode baserte seg på flere faktorer. Vi hadde få rammer og begrensinger fra Avento, men de følgende punktene var med på å forme valgene rundt metode.

- Mye frihet rundt prosjektet.
- Mange ideer å vurdere.
- Funksjonalitet ble til underveis.

Ved punktene over til grunne, krevde ikke denne oppgaven en spesiell type metodikk. Oppgaven ble definert som fleksibel på både funksjonalitet og fremgangsmåte. Løsninger ble til

underveis, og det krevdes mye prøving og feiling i løpet av prosjektet. Vi lagde tidlig en fremgangsplan som i grove trekk beskrev prosjektprosessen. Dette gjorde det enklere å nå delmål, og prioritere tiden riktig.

Vi brukte Asana og Instagantt som program for å organisere arbeidsoppgaver. Vi la inn oppgavene i Asana og fremstilte disse i en tidslinje ved hjelp av Instagantt. Ved å bruke disse programmene fikk vi en god oversikt over hva som skulle gjøres til enhver tid.

3.2.1 Gjennomføring av prosjektet

Vi bestemte oss for å arbeide i intervaller på to uker for å få en god flyt, samtidig som vi beholdt en viss struktur i arbeidet. En intervall varte fra mandag til mandag to uker senere. På grunn av forelesninger i faget Industri 4.0 ble disse intervallene av og til kortere eller lengre enn to uker, men dette klarte vi fint å tilpasse sammen med veileder og justerte møtetider deretter.

3.2.1.1 Møter med oppdragsgiver

Vi startet prosjektarbeidet med et møte med Anders Beite og Stian Bang i Avento der vi gikk gjennom prosjektet og deres forventninger til oss. Avento ønsket å se hvordan vi ville løse denne oppgaven og ville derfor ikke påvirke oss i form av valg av løsninger, men heller bistå dersom vi trengte hjelp. Derfor ble det kun planlagt to møter med Avento etter dette. Et møte med Låsservice og Avento tidlig i prosjektet, og et med Avento mot slutten av prosjektperioden.

I møtet med Låsservice og Avento ble vi enige om at Låsservice skulle sende oss dokumentasjon og prøve å skaffe oss et låsemiljø i form av en sentral og en dør vi kunne bruke i utviklingsprosessen. 30.04.2019 møtte vi Anders Beite hos Avento igjen og gikk gjennom det som var gjort så langt. Det ble også planlagt et møte en gang etter presentasjonen av bacheloroppgaven som er 24.05.2019.

3.2.1.2 Interne møter

Hver mandag hadde vi et gruppemøte der vi gikk gjennom hva hver enkelt skulle gjøre den kommende uken, og oppdaterte Asana med den informasjonen. Vi hadde også et kort møte hver fredag der vi gikk gjennom hva som ble gjort i løpet av uken og hvor langt vi hadde kommet i

forhold til planen. Annenhver fredag klokken 09:00 hadde vi et utvidet møte der vi gikk gjennom alt som var gjort i den foregående intervallen, og hva som skulle gjøres neste intervall. Dette ble nedskrevet og sendt til oppdragsgiver i Avento og veilederne våre.

3.2.1.3 Møte med veileder

Det ble planlagt å ha et statusmøte med veileder Girts Strazdins annenhver fredag klokken 12:00, samme dag som vi hadde vårt utvidete interne møte. På dette møtet var hovedfokuset på hva vi hadde gjort og hva som skulle gjøres i neste intervall, samt vise nyeste versjon av løsningen vår. Siden vi sendte en skriftlig oversikt over hva som var gjort og planlagt fikk veileder muligheten til å forberede seg før møtet. Dette resulterte i en god dialog mellom veileder og gruppe som gjorde at vi fikk mulighet til å se løsningen fra et annet perspektiv enn vårt eget. Dette hjalp oss mye i utviklingen av prosjektet.

3.2.1.4 Asana og Instagantt

Vi brukte gratisversjonen av Asana som prosjektstyringsverktøy og delte først prosjektet inn i seksjoner. Deretter la vi inn arbeidsoppgavene som *tasks* eller oppgaver under sin respektive seksjon. Vi ga også oppgavene en tidsfrist til de skulle være ferdig. Hver oppgave kunne også ha flere underoppgaver, noe vi ofte følte var nødvendig for å visualisere fremgang. Se figur 3.1 for oppgaver i Asana.

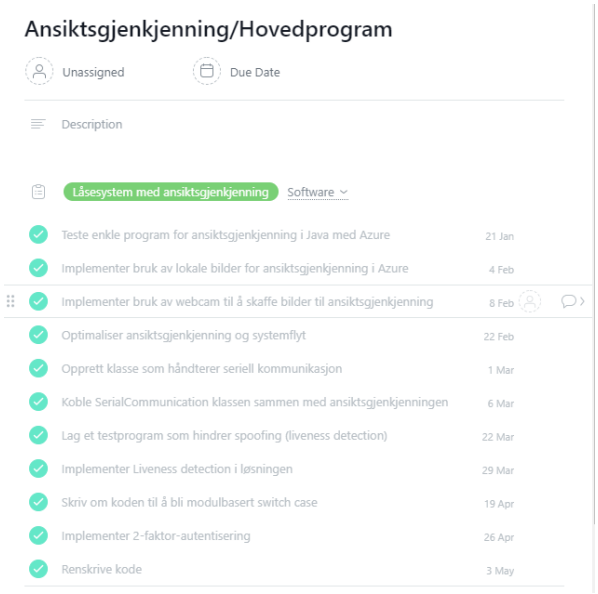


Figure 3.1: Asana task med subtask

Oppgaver i Asana har som oftest to tilstander: Uferdig og ferdig. Når de markeres som ferdig får de en grønn hake og skjules frem til man velger at disse også skal vises. Når dette var gjort synkroniserte vi Asana med Instagantt og fikk muligheten til å sette startdatoer på oppgavene og fremstille disse i en oversiktlig tidslinje. Vi bruker Instagantt til dette ettersom dette ikke er en del av gratisversjonen til Asana.

Noe vi merket iløpet av prosjektperioden er at det er veldig vanskelig å treffe godt med antatt tidsbruk per oppgave, ofte på grunn av uventede feilmeldinger og lignende i kodeprosessen. På grunn av dette måtte vi justere ganske mye på tidsfrister og lignende på interne møter. Dette var veldig enkelt å gjøre både i Asana og Instagantt og lot oss i tillegg enkelt legge til nye oppgaver samt fjerne oppgaver vi ikke lenger har bruk for. I figur 3.2 kan man se et eksempel på hvordan prosjektet ble delt inn i egne oppgaver og tidsfrister.

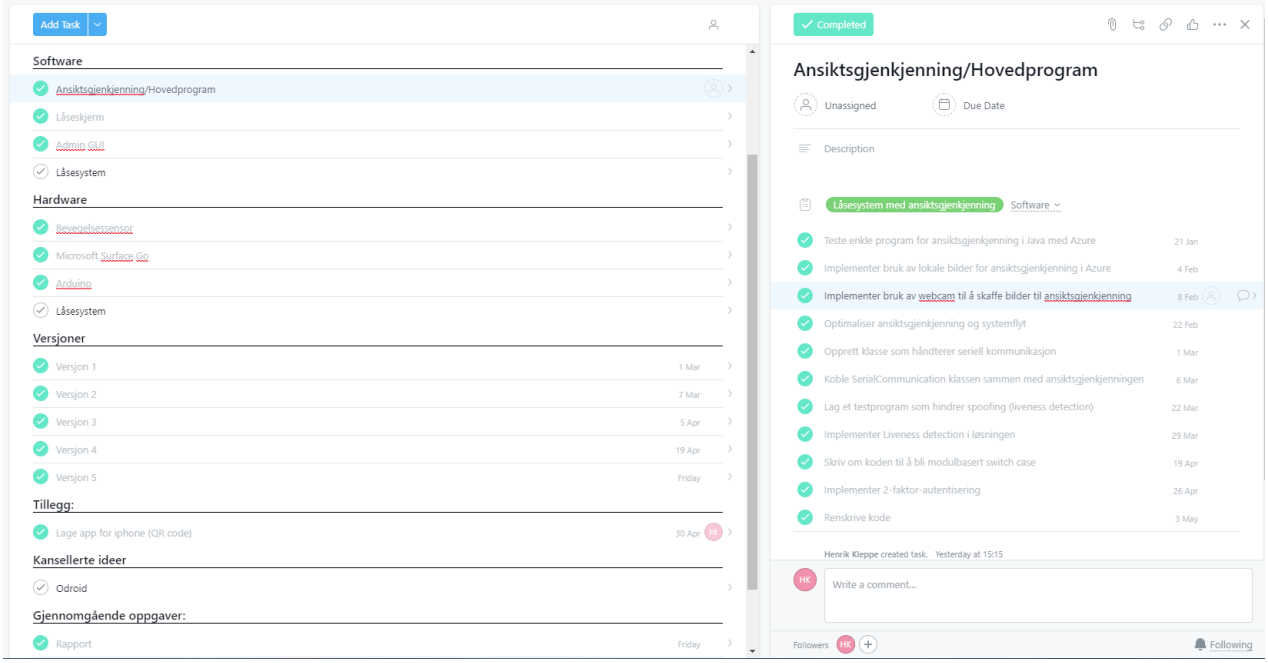


Figure 3.2: Oppgaver og tidsfrister i Asana

Etterhvert som vi gikk vekk fra Odroid la vi alle relaterte oppgaver inn i en egen seksjon som vi kalte "Skrinlagte ideer".

3.3 Programmeringsspråk

I denne seksjonen går vi igjennom diverse programmeringsspråk vi har vært innom i løpet av prosjektet.

3.3.1 Java

Java er et objektorientert programmeringspråk. Dette er blant de mest brukte språkene innen utvikling av programvare og kan kjøres på de fleste operativsystemer, som blant annet Windows, macOS og Linux. Java SDK er et verktøy for å utvikle software. Det inneholder biblioteker til å skrive og kjøre programmer.[27]

3.3.2 Swift

Apple har et egetutviklet programmeringsspråk kalt Swift. Dette språket er spesialisert for å lage applikasjoner og programmer til macOS, iOS, watchOS osv.[28]

3.3.3 Arduino C

Arduino C er laget for at brukeren skal forstå språket. Når man skriver Arduino C blir funksjonene oversatt automatisk av en C/C++ kompilerer. Dette språket er organisert i funksjoner, variabler og konstanter.[29]

3.3.4 C++

C++ er et av de vanligste høynivå programmeringsspråkene og er foretrukket hos selskap som Microsoft og Intel. C++ er en videreutvikling av C, og ble i starten omtalt som "C med klasser". Dette språket er et multiparadigme språk og har støtte for både objektorientert programmering, funksjonell programmering, dataabstraksjon og generisk programmering.[30]

3.3.5 Python

Python er et programmeringsspråk som er utviklet for å skrive kode raskere og enklere for brukere å lese. Python støtter også flere programmeringsparadigmer som f.eks. prosedyremessig, objekt orientert og funksjonell programmering. Python har også et omfattende standardbibliotek som er mye større enn andre språk.[31]

3.4 Kodestil

Vi har valgt å implementere en kodestil som følger retningslinjene i boken Objects First with Java. Denne boken baserer seg på objektorientert programmering i Java. Kodestilen angår bare programmene som er skrevet i Java.

3.4.1 Navn

Navn skal skrives på følgende måte:

- Alle navn skal skrives på engelsk.
- Klasse navn skal starte med stor bokstav, fulgt av små. Vi bruker stor bokstav til å skille mellom ord. For eksempel *MyFirstClass*.
- Metoder og variable skal starte med liten bokstav. Stor bokstav brukes til å skille mellom ordene. For eksempel *myFirstMethod()*.
- Variabler som er konstante skal navngis med kun store bokstaver, og bruker understrek for å skille mellom ord. For eksempel *FINAL_VARIABLE*.

3.4.2 Struktur

Fire mellomrom skal være standarden for identasion. Eksakt bruk av mellomrom over *tab* er uspesifisert.

Linje-lengde

Unngå linjer som er lengre en 80 karakterer. Dette kan by på problemer hos mange terminaler og verktøy. I dokumentasjon burde linjene være kortere en 70 karakterer.

Tekstbryting

Når et utrykk ikke passer på en linje, skal teksten bryte på følgende måte:

- Bryte etter komma.

- Bryte før en operator.
- Den nye linjen skal starte samme sted som begynnelsen av uttrykket på linjen over.
- Hvis regelen over fører til forvirring eller kode som ser rotete ut, brukes en standard på 8 mellomrom isteden.

Et eksempel på hvordan det kan gjøres er vist i figur 3.3.

```
function(longExpression1, longExpression2, longExpression3,
        longExpression4, longExpression5);

var = function1(longExpression1,
                function2(longExpression2,
                          longExpression3));
```

Figure 3.3: Eksempel på bryting

Curly brackets

Curly brackets skal starte på samme linje som metoden/klassenavnet, se eksempel i figur 3.4.

```
if (condition) {
 /* Handle the condition. */
 ...
}
```

Figure 3.4: Eksempel på bruk av curly brackets

Vertikale mellomrom

To blanke linjer skal brukes under følgene omstendigheter:

- Mellom seksjoner med kildekode.
- Mellom klasser og *interface* definisjoner.

En blank linje skal brukes under følgende omstendigheter:

- Mellom metoder.
- Mellom den lokale variabelen i en metode og dens første deklarasjon.

3.4.3 Kommentarer

Vi bruker *block* kommentarer over klasser og metoder, mens vi bruker *single line* kommentarer der det er nødvendig. Vi prøver i størst mulig grad å la metodene og variablene ta seg av det meste av kommenteringen. Dette gjør at vi unngår unødvendig "støy" i koden, som kommentarer ofte fører til.

3.4.4 Språk

All programmering og kommentering foregår på engelsk, siden engelsk er det dominante språket når det kommer til kodestil.

3.4.5 Implementasjon

Vi har prøvd å følge kodestilen til punkt å prikke. Det var i stor grad opp til hver enkelt medlem i gruppen å følge de retningslinjene som gjaldt. Vi gikk ofte igjennom koden sammen og rettet feil der vi følte at kodestilen ble brutt. Vi prøvde å utelukke bruk av *header*, siden programmet er skrevet av en gruppe og ikke en enkeltperson.

3.5 Eksterne Bibliotek

I løpet av prosjektperioden har vi brukt flere eksterne bibliotek for å oppnå ønsket resultat eller for å forenkle noen prosesser.

3.5.1 Java-bibliotker

Her beskriver vi noen av bibliotekene vi har brukt i Java-miljøene.

jSerialComm

jSerialComm er et javabibliotek som er designet for å gi tilgang til standard serielle porter (for eksempel USB-porter), uavhengig av plattform. Biblioteket er ment som et alternativ til RxTx og den utgåtte Java Communcations API'en. Biblioteket er veldig enkelt å bruke, og ved en enkel kodesnutt som dette: *SerialPort.getCommPorts()* genereres en liste over tilgjengelige serielle porter som kan itereres gjennom. Vi brukte dette biblioteket til å sette opp den serielle kommunikasjonen mellom Java og Arduino. Hovedårsaken til at nettopp dette biblioteket ble benyttet er at vi har brukt det ved en tidligere anledning og hadde derfor litt kjennskap til det fra før. Vi har brukt versjon 2.3.0.[32]

JSON in Java

Dette biblioteket implementerer JSON enkodere og dekodere i Java (se avsnitt om JSON her: 2.5.4). Det inkluderer også muligheten til å konvertere mellom blant annet JSON og XML eller JSON og HTTP headers. Kort fortalt hjelper dette biblioteket oss med å enkode og dekode data i

JSON format. Vi trengte et JSON bibliotek for å kunne tolke REST responsen fra Azure og BioID på en enkel og oversiktlig måte. Vi har brukt versjon json-20180813.[33]

Sarxos webcam capture

Denne pakken brukes for å enkelt ta i bruk innebygde eller eksterne webkamera i Java. Vi bruker denne pakken for å åpne et webkamera i Admin GUI og vise streamen i GUIen. Pakken består av tre bibliotek:

- bridj-0.7.0 - Link/overgang for å kunne bruke C++ metoder i koden og opprettholde hastighet og brukervennlighet. Må inkluderes for at webcam-capture skal virke.[34]
- slf4j-api-1.7.2 - Simple Logging Facade for Java. Kun brukt fordi den er en del av Sarxos Webcam Capture.[35]
- webcam-capture-0.3.12 - selve webkamera håndteringen, samt mulighet for å leite etter tilkoblede kamera på maskinen.[36]

httpclient-4.5.6

Selv om Java.net-pakken inneholder grunnleggende funksjonalitet for HTTP, mangler den mye for å kunne tilby den komplette funksjonaliteten og fleksibiliteten mange applikasjoner trenger. Derfor tar vi i bruk httpclient fra Apache, ettersom dette biblioteket tilbyr en effektiv, oppdatert og funksjonsrik pakke som implementerer klientsiden av de nyeste HTTP-standardene. Vi bruker blant annet dette biblioteket i kommunikasjonen med Azure.[37]

httpcore-4.4.10

Dette biblioteket implementerer de mest grunnleggende aspektene ved HTTP protokollen, men samtidig nok funksjonalitet til å kunne utvikle fullverdige HTTP-tjenester, både på server- og klientsiden. Vi bruker dette biblioteket sammen med httpclient i kommunikasjon med Azure. Noe av funksjonaliteten vi bruker i dette biblioteket kommer i form av Stringentity, HTTPEntity, HTTPResponse, FileEntity og EntityUtils.[38]

Kotlin

For å kommunisere med BioID trenger vi to Kotlin bibliotek: stdlib og stdlib-common. Disse brukes ikke direkte i koden, men er nødvendige for at informasjonen som sendes og mottas er korrekt. Dersom disse ikke er med, vil vi få feilmeldinger nesten umiddelbart etter programmet startes. Vi har brukt versjon 1.3.21.

OkHttp

Kommunikasjonen med BioID er avhengig av et par andre bibliotek også.[39] Disse er:

- MockWebServer - En klasse under okhttp3. Trengs i forbindelse med BioID.
- OkHttp - HTTP-klient, dersom nettverket sliter, vil dette biblioteket i stillhet hente seg inn igjen fra vanlige tilkoblingsproblemer. Dersom en tjeneste har flere IP addresser vil OkHttp prøve alternative addresser dersom det første tilkoblingsforsøket feiler.
- OKIO - Bibliotek som implementerer java.io og java.nio for å forenkle tilgang, lagring og prosesseringsdata.[40]

commons-codec-1.9

På grunn av bruk av QR-kode og tofaktorautentisering trengte vi et bibliotek for kryptering og dekryptering av informasjon. Dette biblioteket gjør dette, sammen med ZXing, og støtter for eksempel Base64, Hex, Fonetisk og URL.[41]

ZXing

Bibliotek for kryptering og dekryptering av barkode/QR kode. Vi har brukt to biblioteker fra ZXing:

- Core - Kjernefunksjonalitet
- JavaSE - Java SE-spesifikke utvidelser til ZXing biblioteket.

OpenCV wrapper for Java

Det viktigste biblioteket som er brukt er et C++ bibliotek med wrapper for Java. Mer om dette biblioteket under avsnitt 3.5.2.

3.5.2 C++ biblioteker

Da vi eksperimenterte med point cloud i C++ ble det hovedsakelig brukt to biblioteker: OpenCV og Librealsense2.

OpenCV

OpenCV står for Open Source Computer Vision Library, og er et bibliotek med åpen kildekode for datasyn- og maskinlæringsprogramvare. OpenCV ble bygd for å gi en felles infrastruktur for applikasjoner med datasyn og for å akselerere bruken av maskinoppfatning i kommersielle produkter.[42] Biblioteket består av over 2500 optimaliserte algoritmer, inkludert algoritmer for ansiktsgjenkjenning, objektidentifisering og maskinlæring. Vi har hovedsakelig benyttet oss av ansiktsgjenkjenningen til OpenCV for å forsikre oss om at bildene som brukes i løsningen vår er brukbare. Selv om OpenCV er et C++ bibliotek, har de *wrappere* for andre språk, som Java og Python. Vi brukte både versjon 4.0.1 og 4.1.0.

Librealsense2

Librealsense er Intel sitt offisielle bibliotek for RealSense kameraene. Vi brukte versjon 2.0 fordi denne versjonen er optimalisert for D400-serien. Dette biblioteket inneholder mye eksempelkode for å teste kameraets funksjonalitet, i tillegg til OpenCV wrapper, point cloud wrapper og mange flere. Det å bruke dette biblioteket var ganske enkelt på grunn av svært god dokumentasjon og gode eksempler.

3.5.3 Swift-biblioteker

Her er noen av bibliotekene vi har brukt i Swift. Disse ble brukt til å lage iOS-applikasjonen for QR-passordene.

Wallet

Wallet er et bibliotek kan man bruke i swift. Dette gir deg mulighet til å oprette en replika av "Wallet" *interfacet* til Apple. Biblioteket gir deg muligheten til å oprette, slette og vise kort. Biblioteket er open source og under MIT-lisensen.[43]

SwiftOTP

SwiftOTP er et bibliotek som gjør det enkelt å generere engangspassord. SwiftOTP støtter både HMAC-baserte engangspassord (HOTP) og Tids-baserte engangspassord(TOTP).[44]

3.6 Skytjenester

I denne seksjonen går vi igjennom hvilke skytjenester vi har benyttet oss av.

3.6.1 BioID og Liveness detection

BioID er en autentiseringstjeneste som sjekker om en person er ekte eller ikke ved hjelp av biometri.

Liveness detection er en av tjenestene til BioID. Den kan finne ut om et ansikt tilhører en ekte person eller et bilde. Ved å sammenligne to bilder kan man se endringer og naturlig bevegelse i et par nøkkelpunkt i ansiktet. Et tredimensjonalt ansikt (ekte) og et bilde av et ansikt ("falskt") beveger seg forskjellig, og algoritmene til BioID kan oppdage forskjellen.[45]

3.6.2 Microsoft Azure kognitive tjenester

Microsoft kognitive tjenester er en samling med maskinlæringsalgoritmer. Dette er en tjeneste som gir enkel tilgang for utviklere til å løse oppgaver med kunstig intelligens. I kognitive tjenester kan man bruke diverse pakker til å løse AI-problemer. Man kan blant annet analysere bilder og videoer, og gjenkjenne stemmer. Azure har også en pakke for ansiktsgjenkjenning (Face API), og det er denne vi benytter oss av i løsningen.[46]

Når et bilde blir sendt opp til Azure for å gjøre ansiktsgjenkjenningen, så vil en algoritme prosessere bildet. Man får så tilbake et resultat i form av en likhetskonfidens mellom 0 til 1,

som forteller hvor likt personen på bildet er med bildet i databasen. Dette er et samlet resultat basert på mange faktorer som hudfarge, hår, øyne, karakteristikker i ansiktet og mye mer. Det er også verdt å nevne at Azure ikke lagrer bilder i skyen, men en *FaceID* som genereres fra disse faktorene. *FaceID* vil kun lagres i skyen i 24 timer og er en midlertidig ID. Dersom vi kobler denne IDen opp mot en person i databasen, blir den omgjort til en *PersistedFaceID* som følger personen til enten ID eller person blir slettet.

3.7 Utviklingsverktøy

I prosjektet vårt har vi utnyttet flere utviklingsverktøy. Vi vil gå kort gjennom disse i denne seksjonen.

3.7.1 Netbeans

NetBeans er en utviklingsplattform for å utvikle programvare med Java, C++, PHP og andre språk. NetBeans har åpen kildekode som betyr at det er tilgjengelig for alle. NetBeans er selv kodet i Java og kjører på de fleste operativsystemer.[47]

3.7.2 Arduino IDE

Arduino IDE har åpen kildekode som betyr at det er tilgjengelig for alle. Dette utviklingsverktøyet gjør det enkelt å skrive kode og laste det opp på Arduino brettet. Dette utviklingsverktøyet kan brukes med alle Arduino brettene.[48]

3.7.3 Xcode

Xcode er en utviklingsplattform for designing av applikasjoner for Apple produkter. Man kan velge mellom programmeringsspråkene Swift og Objective-C. Denne plattformen er designet for å være brukervennlig. Underveis i programmeringen kan man simulere og teste programmet på en virtuell iPhone eller iPad for å se hvordan applikasjonen vil se ut og fungere. Har man en Iphone eller Ipad tilgjengelig, kan også programmet kjøres direkte på den fysiske enheten.[49]

3.7.4 SourceTree

SourceTree er et grafisk brukergrensesnitt som brukes for å jobbe med Git. I SourceTree slipper man å jobbe i kommandolinjen. Man kan jobbe med grener, noe som gjør det enklere å sette sammen arbeid fra forskjellige versjoner. Dette gjør det enkelt og oversiktlig å jobbe flere på et prosjekt samtidig.[50]

3.7.5 Git og GitHub

GitHub er en nettside og en tjeneste, som bruker Git. Git er et linuxprogram for versjonskontroll av diverse prosjekter som applikasjoner, programmer og dokumenter. Det brukes til lagring av utviklingsprosjekter der produktet stadig er i endring. Git gjør det enklere for utviklere å jobbe sammen, da man kan laste ned, gjøre endringer, og laste opp igjen en versjon av et prosjekt. Git er åpen kildekode og er det mest brukte versjonskontrollsystemet blant utviklere.[51]

3.7.6 Asana og Instagantt

Asana er et prosjektstyringsverktøy designet for å hjelpe grupper å organisere, følge og håndtere prosjektarbeid og tilbyr svært mange tjenester i en applikasjon. Grupper kan opprette prosjekt, sette tidsfrister, fordele arbeid på gruppemedlemmene, og kommunisere med hverandre direkte i applikasjonen. Man kan også få tilgang til kalender, tidslinje visning, Android/iOS app med mer. Siden Asana er et kommersielt produkt vil det koste litt å få tilgang til alle tjenestene de tilbyr, men de gjør opp for dette ved å tilby integrasjon med Gmail, Dropbox, Microsoft Outlook, Google Drive, Instagantt og mange flere lignende tjenester.[52]

Instagantt er en tjeneste som tilbyr noe av det samme som Asana, men med hovedfokus på gantt diagram og tidslinjer. Integrasjonen mellom disse to tjenestene gjøres svært enkelt og synkroniseres automatisk etter det er satt opp.

3.8 Test av tidsbruk på prosessen

Vi testet tiden det tar fra man kommer frem til døren og til døren blir låst opp for en person som har tilgang. Tiden er tatt med stoppeklokke. Vi har gjort 100 forsøk og regnet ut gjennomsnittet

av disse. Samme prøveperson er brukt for hele testen og prøvene er gjort i omgivelser med god belysning. Dette er simpel test for å få et inntrykk av hvor lang tid prosessen tar.

3.9 Materialer

I denne delen ønsker vi å gi en kort introduksjon til de ulike hardware-komponentene som ble vurdert som aktuelle for prosjektet.

Generell utstyrsliste

- Hovedenhet
- Kamera
- Bevegelsessensor
- Tilleggsutstyr

3.9.1 Hovedenhet

Som hovedenhet for prosjektet vårt ønsket vi noe lite og mobilt, som samtidig innehar den prosessorkraften vi trenger for å drive med bildebehandling og for å få et raskt og responsivt system. Dette var et av de aller første valgene vi gjorde da vi skulle starte på bacheloroppgaven. Våre to alternativ på dette tidspunktet var i all hovedsak Raspberry Pi 3 B+ og Odroid XU4, ettersom skolen har flere av disse liggende.

Vi endte til slutt opp med å bruke en Microsoft Surface Go. Den svarte til alle kriteriene våres for hovedenheten. I tillegg hadde den en skjerm vi kunne bruke til visuell veileding.

Odroid-XU4

Odroid XU4 er en SBC, en Single Board Computer, fra Sør-Koreanske Hardkernel og en av de fremste konkurrentene til Raspberry Pi. Odroid-XU4 er kjent for å ha god prosesseringskraft og rask oppstart på grunn av sin åtte-kjerners prosessor, eMMC flash storage modul og 2GB LPDDR3 RAM. Av innganger finner man to USB 3.0 porter og én USB 2.0 port, Ethernet port,

HDMI port og MicroSD for ekstra lagringsplass. Den kan kjøre flere operativsystemer som blant annet Linux/Ubuntu, Android og Nougat. En negativ side med Odroid er at den ikke har innebygd WiFi og Bluetooth, men dette er enkelt å løse ved hjelp av WiFi eller Bluetooth USB-adapter.[53] Figur 3.5 viser et blokkdiagram for en Odroid-XU4.

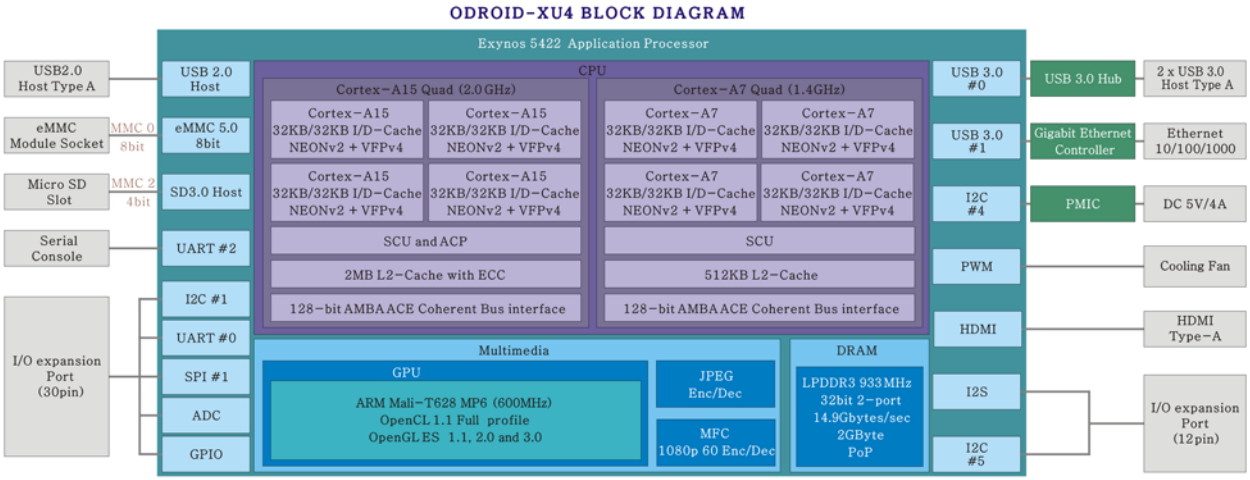


Figure 3.5: Odroid-XU4 blokkdiagram

Raspberry Pi 3 B+

Raspberry Pi 3 B+ er en SBC fra Britiske Raspberry Pi Foundation, og er sannsynligvis den mest kjente SBC'en. Pi 3 B+ har en fire-kjerners prosessor, 1GB DDR2 SDRAM, fire USB 2.0 porter, CSI kamera port (for Pi-kamera), DSI display port (for touchskjerm) og 40-pin GPIO header. I tillegg har den HDMI utgang, PoE (Power-over-Ethernet, en funksjon som tillater å tilføre strøm til enheter gjennom Ethernet kabel) og innebygde WiFi og Bluetooth moduler, samt mulighet for MicroSD for ekstra lagringsplass. Raspberry Pi støtter også flere operativsystemer som Raspbian, ChromeOS, Android og diverse linux distribusjoner.[53] Figur 3.6 viser en raspberry pi.

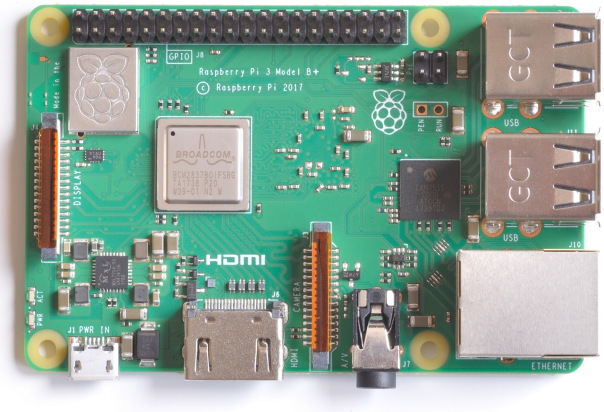


Figure 3.6: Raspberry Pi 3 B+

Microsoft Surface GO

Microsoft Surface Go er en hybrid av nettbrett og datamaskin, med 10" touchskjerm og windows 10 operativsystem. Enheten vi har brukt har 128GB SSD intern lagringsplass, Intel Pentium firekjerners prosessor, 8GB RAM, innebygd WiFi og bluetooth, innebygd kamera, høyttalere, mikrofon og en rekke andre ting vi ikke trenger å gå inn på her. Enheten har én USB-C port, AUX utgang, én Surface Connect-port og microSDXC kortleser.[54] Figur 3.7 viser en Surface Go.

Figure 3.7: Microsoft Surface GO

Odroid XU4 vs Raspberry Pi3 B+

Vi brukte litt tid på å se på sammenligninger av disse to og hver av disse har sine fordeler og ulemper. Odroid er kjent for å være en del kraftigere enn Raspberry Pi, og med sin åtte-kjerners prosessor og dobbel mengde RAM er den totalt overlegen Raspberry Pi på dette aspektet. På

en annen side så mangler Odroid innebygd WiFi og Bluetooth, selv om dette kan løses med WiFi eller Bluetooth USB-adapter. Raspberry Pi har begge disse innebygd og støtter blant annet Gigabit Ethernet 2.4GHz og 5GHz trådløs LAN, samt Bluetooth 4.2 og PoE. Begge to har stort sett den samme støtten for operativsystemer (Linux distribusjoner, Android, med flere) så dette var forholdsvis irrelevant for vårt bruk. Når det gjelder lagringsplass har begge port for MicroSD kort, men eMMC modulen til Odroid gjør at oppstart er betraktelig raskere. Når det kommer til tilleggsutstyr og støtte fra samfunn og forum er Raspberry Pi klart å foretrekke ettersom denne er mer populær og mest vanlig å bruke av de to.[53]

Valget vårt falt til slutt på Odroid, vist i figur 3.8, hovedsakelig av to grunner: den første er at Odroid er betraktelig mye kraftigere, noe vi så på som en nøkkelegenskap for å kunne gi oss et raskt og responsivt system. Den andre grunnen er at vi har brukt en Raspberry Pi i et tidligere prosjekt hvor vi møtte på mange problemer, spesielt ved installasjon av OpenCV for Java. Dette viste seg i ettertid å ikke være et problem på Odroid, selv om vi møtte på en del utfordringer der også. Pris hadde ingen innvirkning på vår avgjørelse ettersom skolen har begge to liggende med tilhørende utstyr.

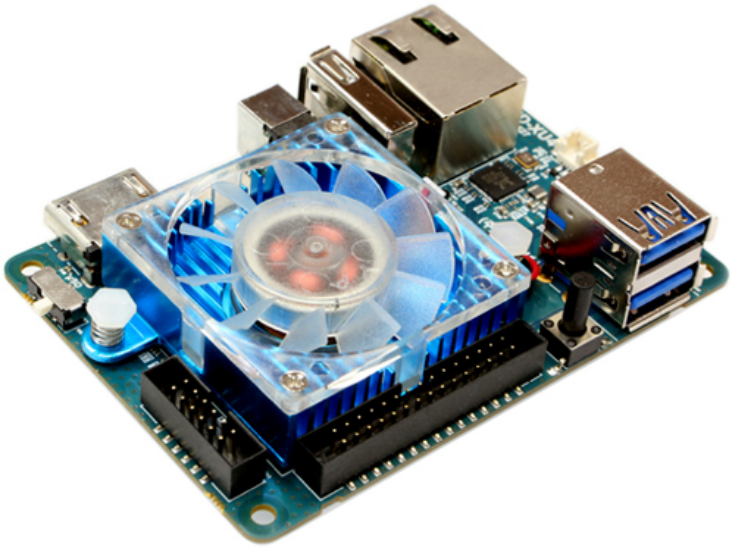


Figure 3.8: Odroid XU4

Overgang fra Odroid til Surface Go

Mot slutten av prosjektet fant vi ut at vi trengte en liten skjerm for visualisering av opplåsingsprosessen. Vi ønsket derfor å bestille en 7" skjerm til Odroid, men på grunn av lang lev-

eringstid fra Hardkernel ble vi introdusert for et alternativ: en Microsoft Surface Go. Vi fikk utdelt et slikt nettbrett fra skolen for å vurdere om vi kunne bruke denne istedenfor Odroid. Det viste seg at denne var kraftigere enn Odroiden og hadde i tillegg Windows 10 som operativsystem, noe som forenklet en del ting. I tillegg hadde denne innebygd kamera og touch-skjerm, noe som eliminerte behovet for et ekstra kamera og ekstra skjerm. Hovedproblemet var mangelen på "vanlige" USB-porter som USB2.0 og USB3.0, men dette løste seg enkelt ved bruk av en USB-dongel. Ved å ta i bruk denne fikk vi et raskere system som i tillegg ble enklere å jobbe med på grunn av operativsystemet.

3.9.2 Kamera

For å kunne drive med ansiktsgjenkjenning er vi avhengig av å ha et godt kamera. Siden vi ønsket å gjøre løsningen så sikker som mulig var bruk av dybdekamera et naturlig alternativ. Vi var innom flere forskjellige, men valget stod til slutt mellom Intel Realsense D415/D435, Orbecc Persee og Asus Xtion PRO. I tillegg ble det brukt ulike webkamera for å teste andre løsninger som ikke var avhengige av dybde.

Intel RealSense

Intel dybdekameraene virket som de mest seriøse av de vi kikket på, og det var også enkelt å finne informasjon om disse. Av D415 og D435 var D435 den mest aktuelle for oss på grunn av noen nøkkelegenskaper: den har bredere FOV (Field of View), global shutter istedenfor rolling shutter og kortere minimum dybdedistanse (0.11m vs 0.16m hos D415). Vi ønsket global shutter kontra rolling shutter fordi kameraet skal filme eller ta bilde av personer som beveger seg mot en dør. Med rolling shutter vil dette kunne forvrenge deler av bildet, noe som igjen kan påvirke ansiktsgjenkjenningen.

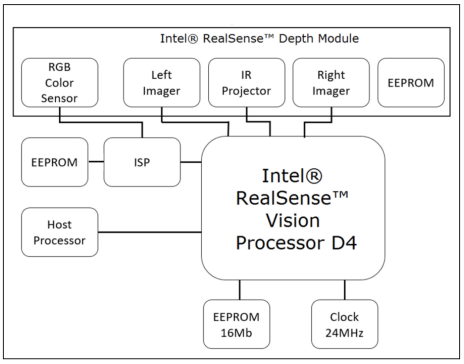


Figure 3.9: Blokkdiagram

Intel Realsense D435 er et kamera med to linser. Det bruker stereosyn som gjør det mulig å måle dybde. Kameraet kan også gjøre målinger med infrarød stråling. Dette gjør det mulig å

forstå og lære av det miljøet som blir tatt bilde av. Kameraet kan brukes både inne og ute, og kan måle dybde opp til 10 meter.

En av tingene som talte imot å bruke Intel RealSense er at de ikke har offisiell støtte eller SDK for Java. I tillegg er det verdt å merke seg at disse dybdekameraene krever USB 3.0 på grunn av at det kreves mye båndbredde for å sende både dybdedata og fargedata fra enheten. Figur 3.9 viser hvordan kameraet er bygget opp.

Orbecc Persee

Orbecc Persee skiller seg litt ut fra de andre, med at denne har en innebygd datamaskin med ARM prosessor. Dette virket i utgangspunktet som et spennende produkt, men på grunn av lite tilgjengelig dokumentasjon så vi raskt bort fra denne.

Asus Xtion PRO

Asus Xtion PRO har vært på markedet lenge og virket veldig aktuell for vårt bruk, og tilbyr mye av det samme som Intel sine. Hovedproblemet med denne er at den tilsynelatende kun støtter x86 og x64 arkitektur. Odroid XU4 har ARMv7 arkitektur, så det var usikkert om denne ville fungere sammen med Odroid.

D435

Valget endte til slutt på Intel RealSense D435 etter gode samtaler med veileder Girts Strazdins angående hvilke alternativ vi hadde. Det var også usikkert om dette kameraet ville fungere med Odroid, på grunn av at Intel ikke har offisiell støtte til ARM arkitektur. Vi fant likevel eksempel på forum der det har fungert på noen ARM-baserte produkter. Derfor lastet vi ned SDK til Odroid og kompilerte eksempelkode for å gi oss en indikasjon på om det ville fungere. Kompileringen gikk fint og vi valgte da å bestille et kamera for å prøve å ta det i bruk i C++ med OpenCV. Kameraet kan du se i figur 3.10.

Figure 3.10: Intel Realsense D435

Webkamera

Som nevnt tidligere har vi kikket på flere løsninger enn bare 3D og point cloud. Til disse løsningene trenger vi absolutt ikke å bruke et dybdekamera da et webkamera gir minst like bra resultat, og i noen tilfeller, bedre resultat. Webkameraene vi brukte var enten innebygde i PCene våre (Macbook, Asus) eller et eksternt webkamera i form av Odroid 720p kamera og Logitech C930e 1080p.

3.9.3 Bevegelsessensor

Når vi skulle velge hvilken type bevegelsessensor vi ville bruke, falt valget på en passiv infrarød sensor. Denne er gunstig å bruke til deteksjon av mennesker på grunn av sin lange rekkevidde og stor synsvinkel. PIR-sensor venter på å registrere bevegelse i motsetning til en aktiv infrarød sensor som både sender og mottar signal, ofte i form av laser eller LED.

Panasonic 10m PIR-sensor

Panasonic 10m PIR Motion Detector er en bevegelsessensor som registrerer bevegelse ved å detektere infrarød stråling. Sensoren har en rekkevidde på opp til 10 meter og har en synsvinkel på 110° horisontalt og 93° vertikalt. Sensoren krever 5 volt.[55] Etter bevegelse er detektert så bruker sensoren gjennomsnittlig syv sekunder for å stabilisere seg. Figur 3.11 viser et blokkdiagram for bevegelsessensoren.

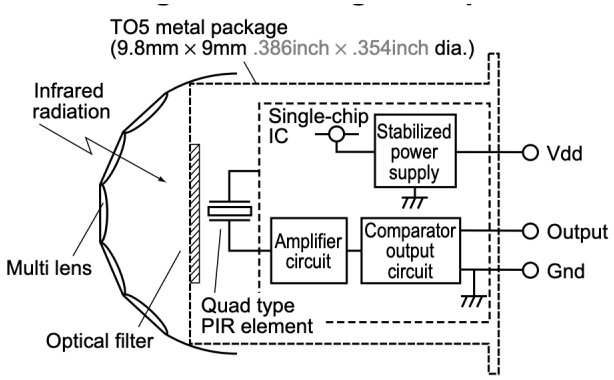


Figure 3.11: Blokkdiagram av bevegelsessensoren

3.9.4 Tilleggsutstyr

I tillegg til hovedkomponentene ble det brukt et par andre ting, både til testing og som del av den ferdige løsningen.

XBOX Kinect

Xbox Kinect var laget som spill-tilleggsutstyr for Xbox og Microsoft Windows PCer. Kinect gjorde at brukere kunne kontrollere og samhandle med enheten sin uten å bruke spillkontrollen, bare ved å bruke bevegelser og talekommandoer. Sensorene som gir Face ID-funksjonene i IPhone X er i praksis en kinect i miniatyr.[56] Vi ville bruke Kinect for å lære oss å manipulere point cloud ettersom det finnes mye dokumentasjon på forskjellige forum relatert til Kinect. I tillegg hadde vi da to enheter som støtter point cloud, som betydde at to på gruppen kunne jobbe med point cloud på hver sin enhet og prøve å lære seg det. Kinecten ble ikke brukt mer etter vi gikk over til liveness detection.

Arduino Uno

Arduino Uno, vist i figur 3.12, er en mikrokontroller som egner seg bra til styring av motorer, servoer og liknende komponenter. Den har blant annet 14 digitale pins hvor seks av dem kan benyttes som PWM tilkoblinger, og i tilegg har den seks analoge tilkoblinger. Dette gjør den veldig gunstig til å styre IO. Det finnes også en rekke *shields* man kan koble til for å trygt kunne

kjøre motorer og liknende fra Arduino. Arduino har sitt eget programmeringsverktøy kalt Arduino IDE som er laget i Java. Mer om Arduino C i avsnitt 3.3.3.

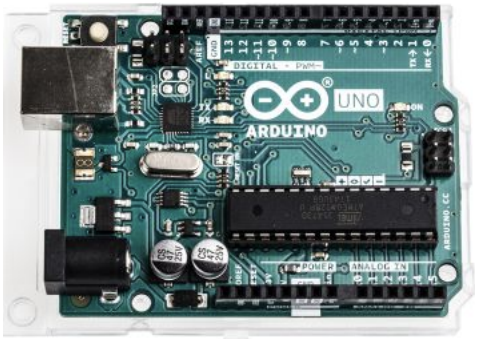


Figure 3.12: Arduino Uno

Skjerm

Som nevnt tidligere, oppdaget vi iløpet av prosjektet at det ville vært fint å ha en liten skjerm for å grafisk hjelpe brukeren gjennom opplåsingsprosessen. I utgangspunktet ønsket vi å bestille en Odroid VU7 skjerm. Dette er en 7" multi-touch skjerm for Odroid. Et annet alternativ var en 7" skjerm fra beetronics. Denne har ikke touch, men kan brukes som vanlig skjerm gjennom bruk av HDMI kabel. Da vi spurte skolen om å bestille en av disse, fikk vi beskjed om å heller bruke en av de nyinnkjøpte Surface Go.

Kapittel 4 - Resultat

I dette kapittelet vil vi presentere resultatene og forklarer hvordan løsningen er utformet. Vi vil også begrunne hvorfor vi endte opp med denne løsningen.

4.1 Systemarkitektur

Vi vil begynne dette kapittelet med en overordnet forklaring på hvilke komponenter systemet er bygget opp av. Figuren 4.1 viser arkitekturen i løsningen, der de stiplede linjene skiller de ulike kommunikasjonsprotokollene.

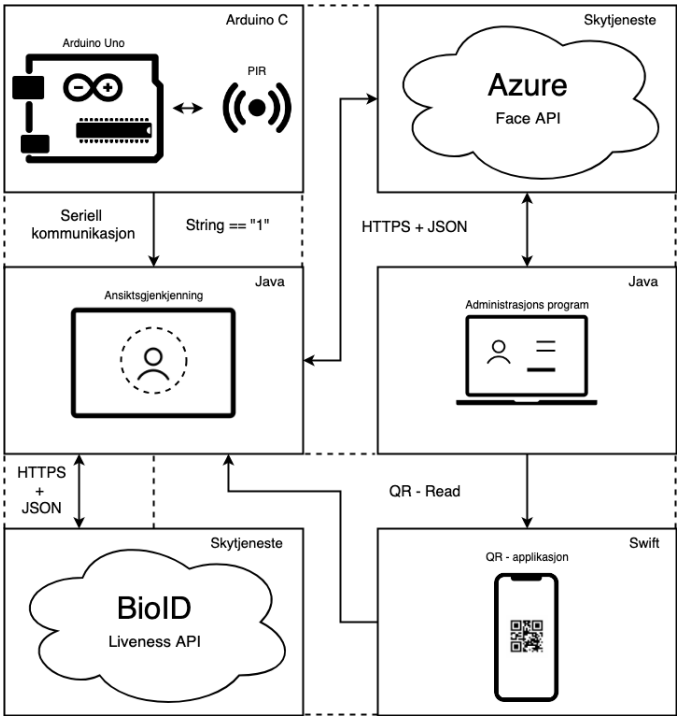


Figure 4.1: Systemarkitektur

Undersystemer

Systemet vårt er bygget opp av i hovedsak seks undersystemer som i sin helhet utgjør løsningen vår. Under finner du en kort beskrivelse av de seks undersystemene.

- Arduino-miljø med bevegelsessensor.
- Java-miljø som har ansvar for ansiktsgjenkjenningen, visuell veileder og QR-passord avleser.
- Swift-miljø som kjører på mobile enheter. Dette har ansvar for å holde på tofaktorautentiseringens passord i form av QR-koder.
- Skytjenesten Microsoft Azure Cognitive Services som inneholder persondatabase og tjenester for å ansiktsgjenkjenning.
- Skytjenesten BioID som vi bruker for å sjekke om det er en ekte person ved bruk av liveness.
- Java-miljø i form av et grafisk brukergrensesnitt som gjør det mulig administrere systemet.

4.2 Use cases

Vi har laget et use case diagram for å lettere illustrere hvordan interaksjonen skal foregå mellom brukere og de forskjellige systemene våre, vist i figur 4.2. Vi har i hovedsak delt brukere inn i 3 forskjellige bolker: besøkende, ansatte og administratorer. Besøkende kan kun bruke systemet og er avhengig av at de har en avtale med bedriften de skal besøke for å kunne slippe inn. De ansatte kan bruke systemet, og har også tilgang til å skru av alarm og komme utenom offisielle åpningstider. Admin kan også bruke systemet, men kan i tillegg administrere hele systemet. Dette innebærer blant annet å slette og liste personer.

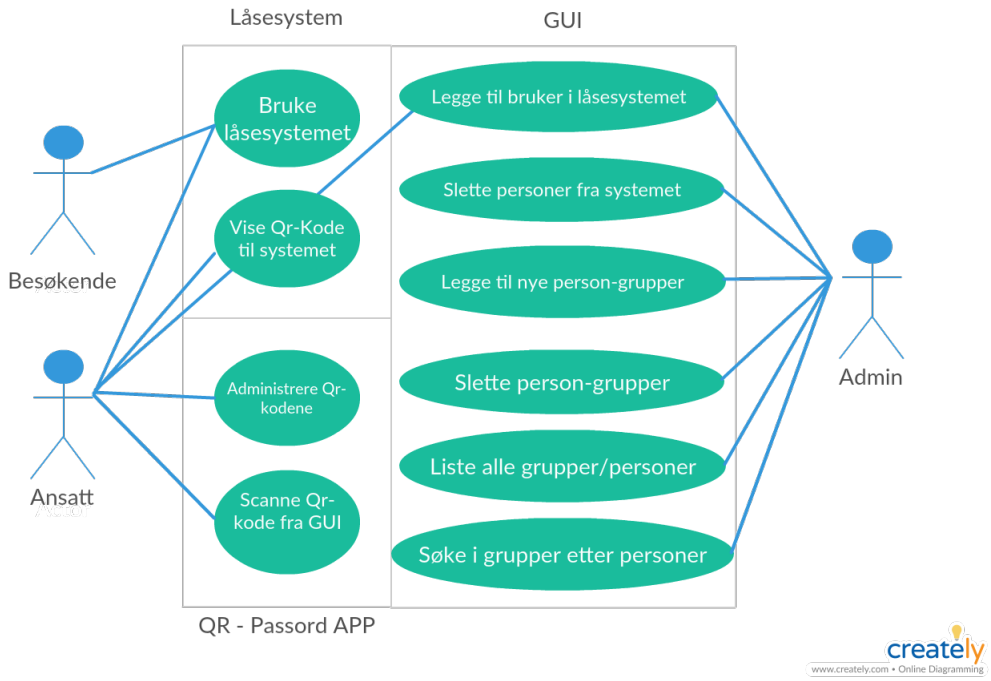


Figure 4.2: Use case diagram for systemet

4.3 Azure Database Design

Med tanke på at vi ikke har laget databasen for systemet vårt er det vanskelig å si nøyaktig hvordan den er bygd opp. Vi ønsker likevel å gå litt inn på hvordan vi bruker databasen og hva databasen "vår" består av.

Når vi gjør kall til Azure gjøres disse til en spesifikk URL som varierer fra metode til metode. Det er denne URL-en som skiller objektene (persongrupper, store persongrupper, personer osv.) fra hverandre. Denne URL-en inneholder f.eks. en gruppeID, personID eller ansiktsID for å spesifisere hvor i databasen vi ønsker å gjøre forandringer eller hvor vi ønsker å hente informasjon fra. I kapittel 4.3.1 forklarer vi bedre hvordan kallene fungerer, ved å ta utgangspunkt i en enkelt metode.

4.3.1 Bruk av REST-API

Både Azure og BioID er REST-systemer og benytter seg av HTTP Request. REST-API er foklart i 2.5.1. For å forklare generelt hvordan vi har implementert metoder som benytter HTTP Request

har vi valgt å ta utgangspunkt i metoden *createPersonToPg(String userData, String groupId)* i klassen *PersonGroupPerson*. Dette er en metode som oppretter en person i en spesifisert persongruppe og REST-forespørselen sendes til Microsoft Azure. Metoden krever to *Strings* som parametere:

- String userData - en String som optimalt sett skal bestå av navn, nummer, epostadresse og en unik "hemmelighet" i forbindelse med tofaktorautentisering.
- String groupId - ID til gruppen man ønsker å opprette personen i.

Det første som gjøres er at vi må opprette og bygge en *HTTPClient*. Deretter trenger vi en *URIBuilder* som består av en Request URL og eventuelle parametre. I dette tilfellet brukes *groupId* som eneste parameter. Når parametrene er satt må URIen bygges før man oppretter en *request*, eller en forespørsel. For denne metoden brukes POST forespørsel. Når forespørselen er opprettet må vi definere noen *headers* for forespørselen. I Azure består *headeren* som regel av to elementer:

- Content-Type - Hvilken type innhold blir sendt? I våre metoder brukes typen "application/json"
- Ocp-Apim-Subscription-Key - En gyldig abonnementsnøkkel for å få tilgang til Azure sine skytjenester.

Deretter definerer vi en *request body* som inneholder dataen vi ønsker å sende. Her vil dette være *userData*, siden denne variabelen består av all informasjon vi har om personen. Forespørsel er så klar for å sendes, og vi kaller da på *httpclient.execute(request)* og avventer respons. Responsen gjøres om til en *HttpEntity*, og det første vi må gjøre med denne er å sjekke at den ikke er null. Dersom den ikke er null kan vi gjøre om *HttpEntity* til *String* og begynne å tolke JSON responsen ved bruk av ulike JSON metoder.

4.3.2 Entity Relationship / Enhetsforhold

Kort oppsummering av hvilket forhold de ulike objektene har til hverandre i databasen:

- Vi kan ha flere *PersonGroups*

- Hver *PersonGroup* kan ha null eller flere *Persons*
- En *Person* tilhører kun én *PersonGroup*
- Hver *Person* kan ha null eller flere *PersistedFaceIDs*
- En *PersistedFaceID* kan kun tilhøre én *Person*
- En *Person* kan ha *UserData* som kan bestå av epostadresse, telefonnummer og autentiseringshemmelighet for tofaktorautentisering.

På figur 4.3 ser vi at alle variablene er av typen *varchar*, bedre kjent som *String*. Grunnen til at felt som *Number* også er *String*, er fordi dette aldri brukes som en *Integer*. Det vil si at det aldri gjøres noen numeriske operasjoner på nummeret. I tillegg hentes verdien fra et *JtextField* som er *String* som standard. Når *Number* skal lagres for videre bruk, lagres den i *UserData* som også er en *String*. Derfor vil det å konvertere *Number* til en *Integer* være bortkastet tid. Vi går litt imot ønsket kodestil når vi lagrer tre verdier som *Number*, *Email* og *AuthenticationSecret* i en *String* som kalles *UserData*, men dette gjøres fordi Azure kun godtar tre felt i *Create Person-metoden*. Disse er *Name*, *personGroupID* og *UserData*, hvor *UserData* inneholder all tilleggsinformasjon vi ønsker å ha om en person.

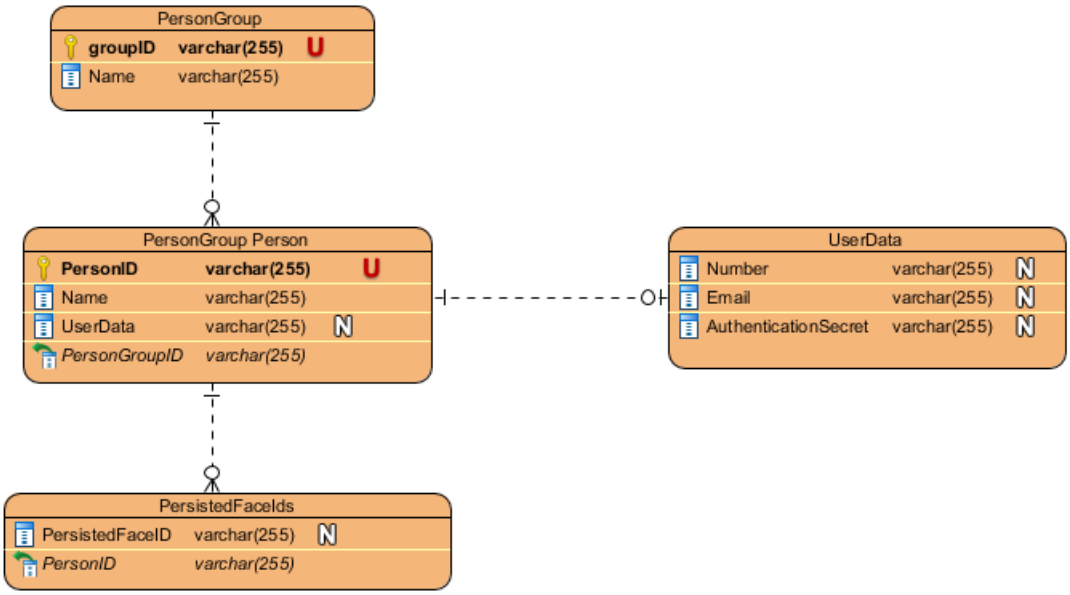


Figure 4.3: Entity Relationship Diagram

4.3.3 Oversikt over egenskaper til enheter

Figur 4.4, 4.5, 4.6 og 4.7 gir en litt mer detaljert oversikt over egenskapene til de forskjellige enhetene i databasen, med utgangspunkt i diagrammet som vist i figur 4.3.

PersonGroup

Egenskap/navn	Datatype	Kommentar
groupId	String	En ID bestående av små bokstaver og tall. Maks 64 tegn
name	String	Navnet på gruppen
(optional) userdata	String	Valgfri tilleggsinfo man ønsker å sende med gruppen. Ikke i bruk.

Figure 4.4: PersonGroup tabell

PersonGroup Person

Egenskap/navn	Datatype	Kommentar
personID	String	Unik ID som identifiserer denne personen
Name	String	Navnet til personen
(optional)Userdata	String	Ekstra informasjon om personen
personGroupID	String	ID til gruppen personen tilhører.

Figure 4.5: PersonGroup Person tabell

UserData

Egenskap/navn	Datatype	Kommentar
(optional)number	String	Lagres som string fordi vi ikke trenger å utføre numeriske operasjoner på nummer
(optional)email	String	Epostadresse til personen
(optional)authenticationsecret	String	En unik hemmelighet som legges ved når personen opprettes. Brukes i 2FA.

Figure 4.6: UserData tabell

persistedFaceIds

Egenskap/navn	Datatype	Kommentar
persistedFaceID	String	En permanent ansiktsID
personID	String	ID til personen relatert til ansiktsID'en

Figure 4.7: PersistedFaceId tabell

4.3.4 Brukte metoder fra Face API - Azure

Azure sin Face API tilbyr en rekke metoder fordelt på ulike klasser, men vi hadde ikke bruk for alle. De tre klassene vi har tatt i bruk er *Face*, *PersonGroup* og *PersonGroup Person* med utvalgte metoder. Disse metodene er beskrevet i "Cognitive services API documentation". Der finnes blant annet helt grunnleggende eksempelkode for å sende REST requests til Azure. Løsningen vår er bygd videre på disse eksemplene. Et par av metodene gir oss funksjonalitet som vi ikke følte vi trengte så disse vil ikke nevnes her. Fullstendig dokumentasjon for disse klassene kan finnes på Microsoft Azure sine hjemmesider.[57]

Face

Denne klassen utfører ulike operasjoner forbundet med ansiktsgjenkjenning. Metodene vi har brukt er:

- Detect - Gjenkjenner ansikt i et bilde og finner nøkkelpunkter, egenskaper og en ansiktsID.
- Identify - Leter gjennom en *PersonGroup* etter et ansikt som matcher det innsendte bildet. Kan kun brukes på en trent persongruppe.

PersonGroup

Denne klassen representerer en gruppe med personer. Metodene vi har brukt er:

- Create - Opprette en ny *PersonGroup* med ID og navn.
- Delete - Slette en *PersonGroup* ved hjelp av ID til gruppen.
- Get - Hente informasjon (ID, navn, tilleggsinfo) om en *PersonGroup*
- List - Liste alle *PersonGroups* med navn, ID, tilleggsinfo. Gruppene lagres i alfabetisk rekkefølge etter gruppeID.
- Train - Trener en *PersonGroup* slik at den kan brukes i forbindelse med Face - Identify. Gjennom testing oppdaget vi at en *PersonGroup* må ha minimum to personer for å kunne trenes.

PersonGroup Person

Denne klassen representerer en *Person* som tilhører en *PersonGroup*. Metodene vi har brukt er:

- Add face - Legger til et ansikt (*persistedFaceID*) til en *Person*. Dette gjøres både ved oppretting av en *Person* og dersom man prøver å opprette en ny *Person* med et ansikt som allerede ligger i en *PersonGroup*.
- Create - Oppretter en ny *Person* i en gitt *PersonGroup*
- Delete - Slett *Person* fra *PersonGroup*
- Delete Face - Slett ansikt (*persistedFaceID*) fra en *Person*. Vi gjør dette hver gang vi vil slette en *Person* for å forhindre at det ligger igjen ansikt uten personer i databasen. Dersom dette ikke gjøres kan det gi problemer hvis man ønsker å legge den samme personen inn i den samme gruppen på et senere tidspunkt.
- Get - Henter ut *name* og *userdata*, samt *persistedFaceIDs* relatert til den valgte personen.
- List - Liste alle personer med *name, personID, userdata* og *persistedFaceIDs*.

4.4 Ansiktsgjenkjenning

Ansiktsgjenkjenningen er skrevet i Java. Programmet har blitt bygd opp med en hovedklasse som står for logikken. Denne er implementert som en tilstandsmaskin, med seks tilstander som interakterer med andre klasser. Programmet kjører også en grafisk veiledning for brukeren på skjermen. Figur 4.8 viser de forskjellige klassene i ansiktsgjenkjenningen.

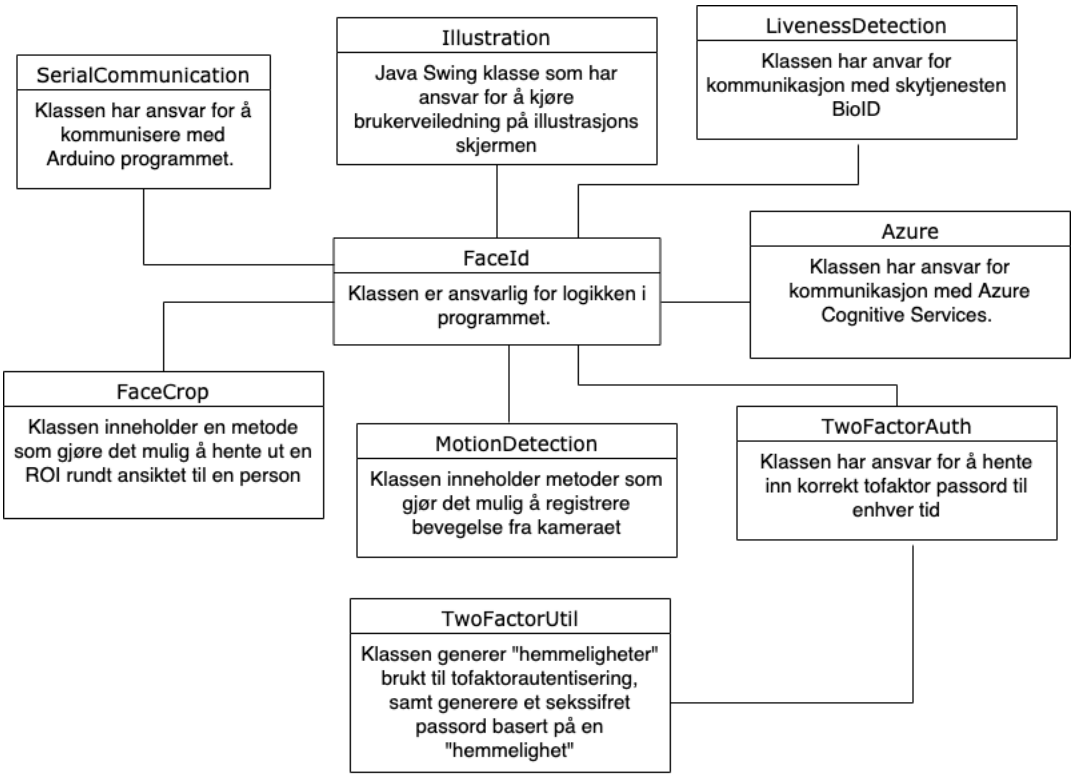


Figure 4.8: Klassediagram av ansiktsgjenkjenningen

4.4.1 Klasser

Løsningen vår består av flere klasser som har egne oppgaver. Klassene har enten ansvar for kommunikasjon mot andre tjenester og systemer, eller laget for å tilføre programmet vårt nødvendige metoder. I denne seksjon tar vi for oss hvilke oppgaver hver klasse har.

MotionDetection

Denne klassen sammenligner to bilder tatt relativt kort tid etter hverandre. Hensikten med denne klassen er å forsikre oss om at det har skjedd en stor nok forandring i ansiktet til brukeren som står foran kameraet før vi tar det andre bildet. Når det har skjedd en stor nok forandring i pikselverdi på bildene, returnerer den true.

Klassen tar en bildematrise og gjør den om til svart/hvitt, for så å legge på en Gausisk blur. Vi bruker en 21 x 21 pikselmatrise, med sigma lik null, for å skape uklarhet. Denne lagres som første bildematrise.

Videre repeterer vi dette og får bildematrise nummer to. Ved hjelp av OpenCV kan vi regne

ut nøyaktig hvor stor forandring som har skjedd fra det første til det andre bilde ble tatt. Dette gjøres ved å utføre en rekke bildebehandlingsmetoder. Først starter vi med å gjøre en enkel terskling. Så utfører en dilasjon og tilslutt utføres en metode for å finne konturene i bildet. Vi gjør denne bildebehandling for å enklere kunne finne konturene i bildene. Alle bildebehandlingsmetodene står beskrevet i seksjonen 2.3. Ved å finne konturene kan vi sammenligne den totale størrelsen i konturendring fra det første bilde til det neste. Figuren 4.9 viser tre eksempler på registrert bevegelse.

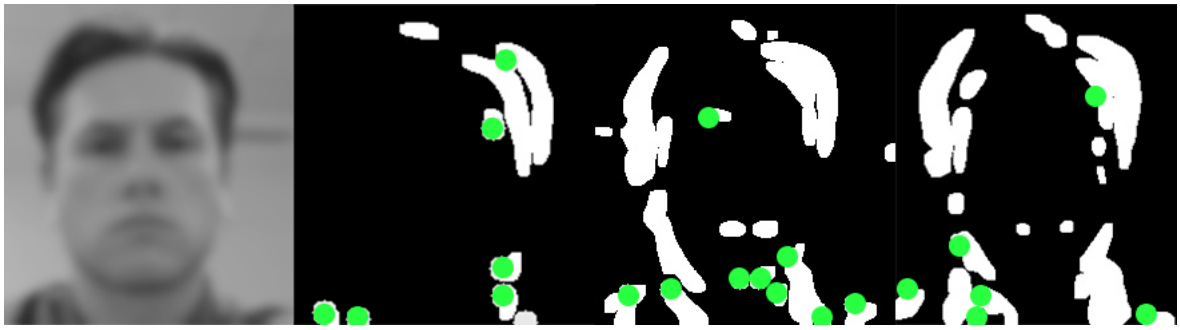


Figure 4.9: Fremstilling av endring i konturer

LivenessDetection

Denne klassen tar for seg selve API-kallet til BioID. Tjenesten BioID står beskrevet under avsnitt 3.6.1. Klassen tar inn to bilder før de blir sendt til tjenesten. Bildene blir sendt via et REST API kall. Hvordan dette gjøres står beskrevet i avsnitt 4.3.1. Responsen vil være true eller false, og representere om personen er ekte eller ikke.

Azure

Denne klassen tar seg av kommunikasjonen med Azure Cognitive Services. Hvordan denne tjenesten fungerer står beskrevet i avsnitt 3.6.2. Vi bruker denne klassen til å detektere om personen finnes i databasen og om personen skal ha tilgang. Kommunikasjonen skjer via et REST API kall. Klassen har også ansvar for å innhente navnet til personen fra Azure, dersom personen som låser seg inn finnes i databasen.

FaceCrop

Denne klassen har ansvar for å finne ansiktet til personen som står foran kameraet. Er det flere personer foran kameraet vil vi sortere disse og kun benytte oss av det største ansiktet. Dette vil sørge for at det ikke blir sendt bilde av to forskjellige personer til BioId. En annen fordel med dette er at vi eliminerer former i bakgrunnen som ofte kan bli oppfattet som et ansikt. Figur 4.10 viser alle ansikter som blir detektert, samt resultatet etter vi har utført beskjæringen.

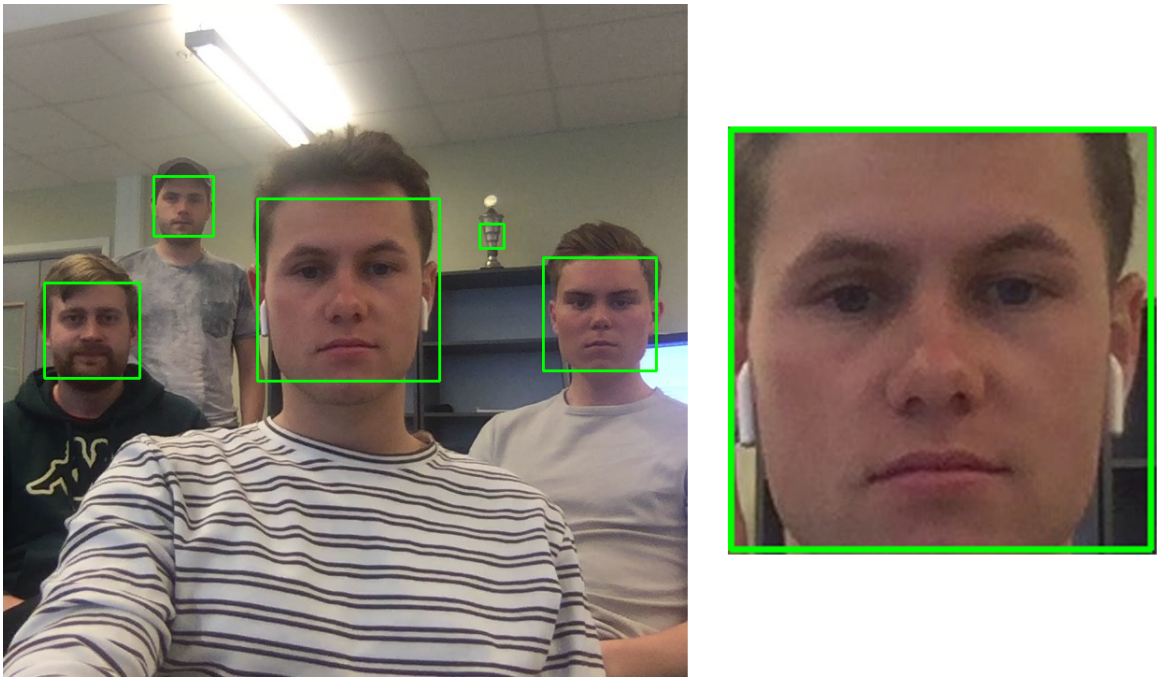


Figure 4.10: Resultatet av beskjæringen i FaceCrop klassen

SerialCommunication

Denne klassen har ansvar for å kommunisere med Arduino C programmet som kjører på mikrokontrolleren. Vi leser av enheten via seriell kommunikasjon.

Illustration

Denne klassen er en Java Swing klasse som innholder forskjellige illustrasjoner laget for å guide brukeren igjennom autentiseringsprosessen på en god og responsiv måte. Vi har designet noen illustrative GIF'er som kjører på illustrasjonsskjermen. Figure 4.11 viser de viktigste illustrasjonene som kjører på skjermen.

Figure 4.11: Illustrasjoner som kjører på illustrasjonsskjermen

TwoFactorAuth

Denne klassen blir kjørt i en egen tråd. Dette gjør at den kan kjøres simultant med resten av programmet. Vi er avhengig av dette fordi denne klassen genererer den 6-sifra koden som brukes i tofaktorautentiseringen. Koden er bassert på TOTP algoritmen, som er beskrevet i avsnitt 2.6.2. Koden bruker en hjelpe-klasse som heter *TwoFactorUtil*. Kildekoden til denne klassen er hentet fra Github.[58]

4.4.2 Tilstandsmaskin

I denne seksjonen vil vi gå igjennom de forskjellige tilstandene og forklare hvordan de fungerer. Figur 4.12 viser en illustrasjon av tilstandsmaskinen.

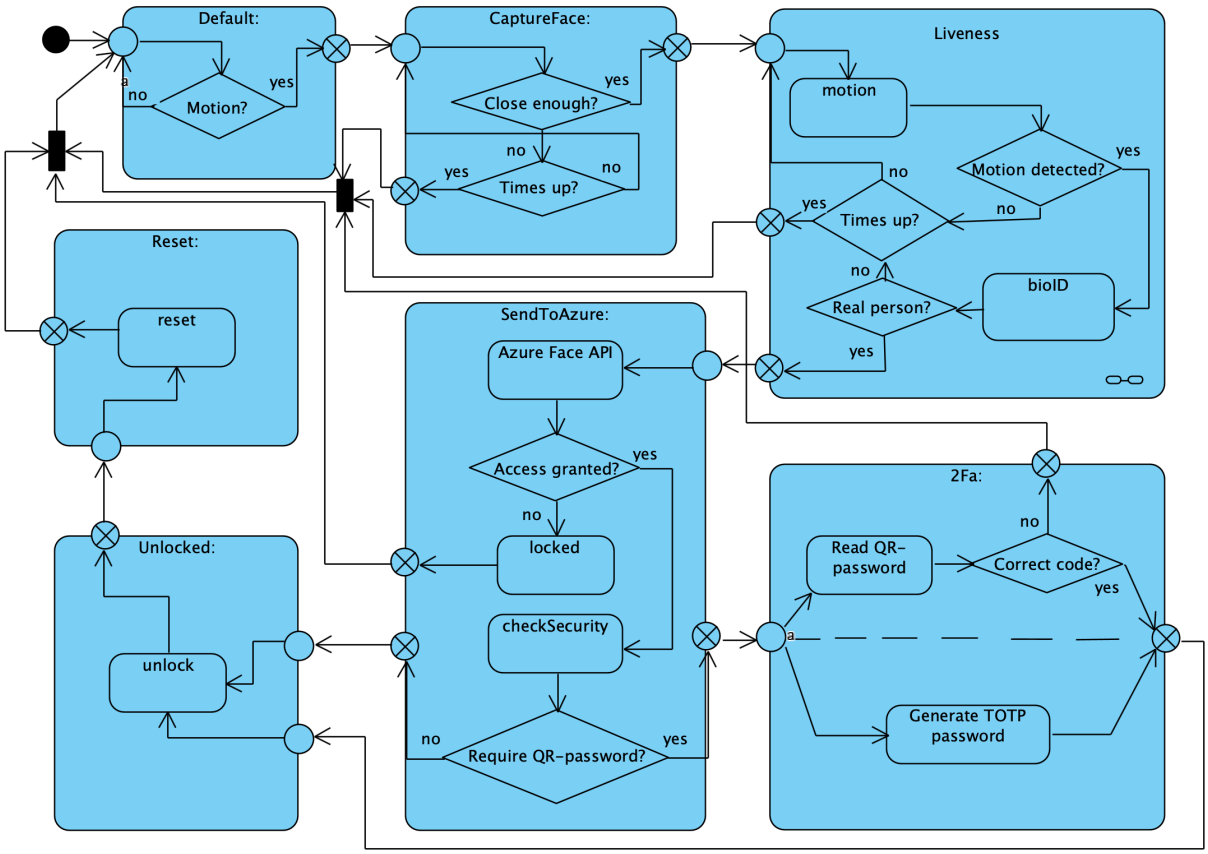


Figure 4.12: State machine UML diagram

"Default" :

Denne tilstanden er *idle*. Det er her programmet står å venter på input fra Arduino C programmet. Programmet blir stående her til det blir registrert bevegelse. Når bevegelse blir registrert bytter programmet tilstand, og vi starter en *timer* for å holde kontroll på tiden. Vi bruker en *timer* slik at programmet ikke kjører seg fast i noen av tilstandene.

"CaptureFace" :

Denne tilstanden finner ansikter, og sjekker om personen står nærme nok. Vi prioriterer ansiktene etter størrelse og guider personen via illustrasjonsskjermen til å trekke nærmere om personen er for langt unna. Avstanden fra kameraet blir definert ut ifra størrelsen på ansiktet. Se figur 4.13 for eksempel på dette. Bruker personen lengre en 30 sekunder på å komme nærme nok, definerer vi dette som at personen har forlatt området og programmet returnerer til tilstanden *Default*. Er personen nærme nok går programmet inn i neste tilstand.

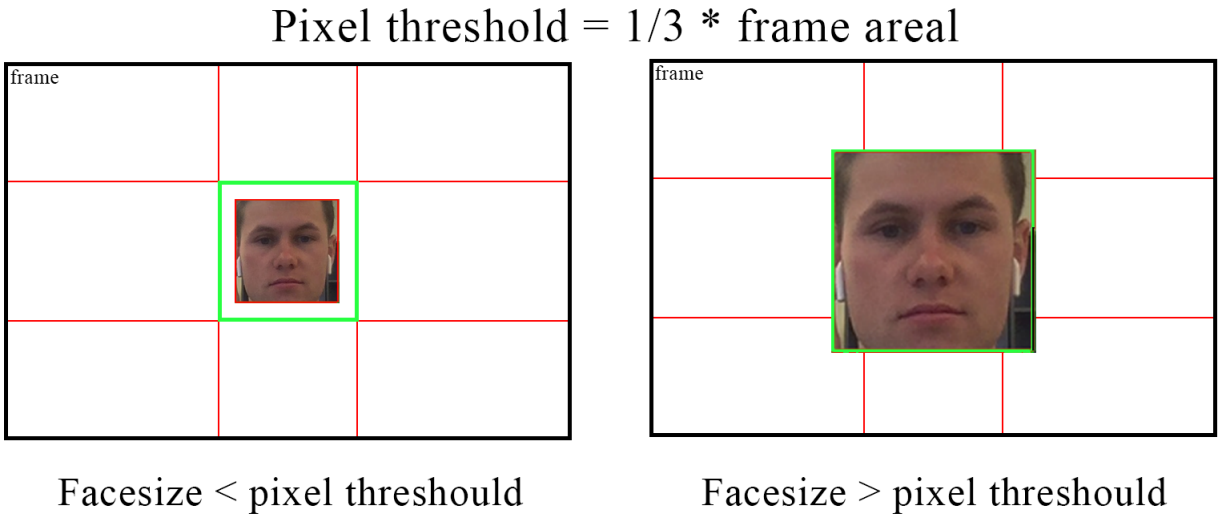


Figure 4.13: Pixel threshold

"Liveness" :

Denne tilstanden tar seg av alt som har med liveness deteksjon å gjøre. Det første som skjer i denne tilstanden er at det blir tatt et bilde av brukeren. Så vil programmet vente til det skjer en viss forandring fra det første bilde ble tatt. Hvordan dette foregår står beskrevet i avsnitt 4.4.1.

Figur 4.9 viser tre eksempler på bevegelse. Når det er registrert bevegelse, blir det tatt et nytt bilde. Disse to bildene blir redusert i størrelse og videresendt til tjenesten BioID. Tjenesten blir beskrevet i avsnitt 3.6.1.

Videre tolkes resultatet vi får fra BioID. Om resultatet er negativt, vil vi først sjekke *timeren*. Har denne gått ut returnerer vi til tilstanden *Default*. Om den ikke har det, prøver vi prosessen på nytt. Om tjenesten gir oss svar om at personen er ekte vil programmet gå til neste tilstand.

"SendToAzure" :

I denne tilstanden foregår alt som har med Azure Cognitve Services. Azure er forklart i avsnitt 3.6.2. Her får vi svar på om personen skal ha tilgang eller ikke. Har ikke personen tilgang vil programmet låse seg i 10 sekunder, før det returnerer til tilstanden *Default*. Har personen tilgang vil vi først sjekke om det kreves et QR-passord. Dette er et ekstra sikkerhetstiltak som ble implementert og står beskrevet i avsnitt 4.9.4. Om det kreves ekstra passord, går programmet til tilstanden *2Fa*. Er ikke dette påkrevd bytter programmet til tilstanden *Unlocked*.

"2Fa":

I denne tilstanden kjører to tråder samtidig. Den ene bruker kameraet til å lese av QR-koder som blir vist til programmet. Den andre tråden genererer TOTP passord som blir sammenlignet oppmot det passordet programmet får via QR-koden. Tofaktorautentisering er forklart i 4.9.4. Blir feil passord vist blir personen avist og programmet går tilbake til tilstanden *Default*. Blir derimot riktig passord vist, vil programmet skifte til tilstanden *Unlocked*.

"Unlocked" :

Denne tilstand er implementert for å kunne håndtere kommunikasjon mot diverse låsesytemer, men av ulike årsaker hadde vi ikke et låsesystem å kommunisere med. Mer om dette i drøftingen. Når programmet er ferdig i denne tilstanden går det videre til *Reset*.

"Reset" :

Den siste tilstanden brukes for å tilbakestille systemet. Her tilbakestiller vi alle nødvendige variabler, og sørger for at programmet er klart for neste person. Programmet vil også gi brukeren

10 sekunder til å gå igjennom døren. Dette gjør vi så ikke programmet registrerer bevegelse fra personen som skal gå inn døren. Når tilbakestillingen er fullført returnerer programmet til tilstanden *Default* og venter på en ny person.

4.5 Administrasjons applikasjon

Administrasjons applikasjonen er skrevet i Java, ved bruk av Java Swing. Applikasjonen er utviklet slik at vi kan administrere systemet på en effektiv og trygg måte. Figur 4.14 viser et klassediagrammet over av programmet.

4.5.1 Klasser

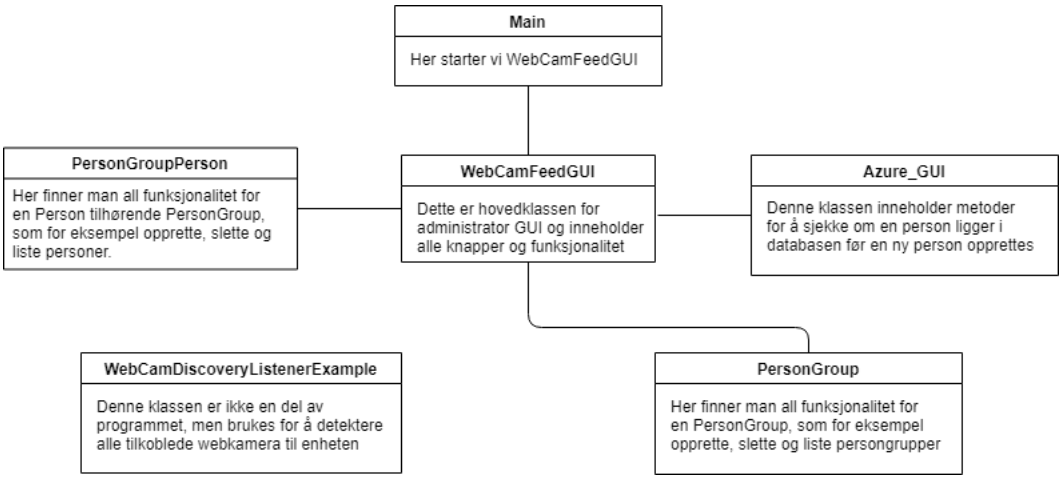


Figure 4.14: Klassediagram over administrasjonsapplikasjonen

I admin GUI har vi implementert funksjonalitet som gjør det mulig å administrere systemet. Dette innebærer å legge til og fjerne personer som har adgang, opprette og slette grupper, og liste både personer og grupper. Personer kan deles inn i grupper som: ansatte, besøkende, admin etc. Når en person legges til i systemet må man fylle ut navn, telefonnummer og epost. Et bilde av ansiktet til personen må også legges til. Dette kan tas direkte når brukeren legges til, eller lastes opp fra fil. Brukeren blir så lagt til i den gruppen som er ønsket, og dette lagres i Azure sin database.

I all hovedsak består administratordelen av fire klasser: *PersonGroupPerson*, *PersonGroup*, *Azure_GUI* og *WebCamFeedGUI*.

PersonGroup

Denne klassen representerer en *PersonGroup* i Azure. Metodene i denne klassen er hovedsaklig kall til Azure, men vi har også laget noen variabler som kan holde på informasjon fra Azure lokalt. Disse variablene gir oss mulighet til å vise alle tilgjengelige grupper i rullegardinlister, og i listefeltet i GUI. Metodene i denne klassen inkluderer funksjonalitet som å opprette en ny gruppe, slette en gruppe, hente detaljer om en spesifikk gruppe, liste alle gruppene i databasen og "trene" en persongruppe.

PersonGroupPerson

En *PersonGroupPerson* representerer en person som tilhører en *PersonGroup*. Akkurat som i *PersonGroup* klassen, er de fleste metodene i denne klassen kall til Azure som gir oss mulighet til å holde på informasjonen fra Azure lokalt og fremstille dette i det grafiske brukergrensesnittet. De viktigste metodene i denne klassen inkluderer mulighet til å opprette en person, slette en person, hente info om en spesifikk person og liste personer i en gitt gruppe.

Azure_GUI

Denne klassen er omtrent identisk til klassen *Azure* i ansiktsgjenkjenningen. Denne metoden er implementert her også fordi vi ønsker å sjekke om en person ligger i databasen når vi skal opprette nye personer. Dette gjøres ved å kalle på *detectFace()* og *identify()*. På denne måten sikrer vi oss mot å opprette flere "identiske personer".

WebCamFeedGUI

WebCamFeedGUI-klassen er hovedklassen for admin GUI. Her opprettes og oppdateres alle GUI-elementene i tillegg til å tilføre funksjonalitet til elementene. Denne klassen kaller på metoder i de ovennevnte klassene og har det overordnede ansvaret for all funksjonalitet i administrator GUI. Nøyaktig hva som gjøres når knapper trykkes vil nevnes i seksjon 4.5.2.

WebCamDiscoveryListenerExample

Denne klassen er ikke en del av programmet, men kjekk å ha for å detektere tilkoblede kameraer til enheten. Vi har ikke laget denne selv, men den er hentet fra Github og er en hjelpeklasse fra Sarxos.[59]

4.5.2 Funksjonalitet

Administratorapplikasjonen består i hovedsak av fire forskjellige vinduer. Disse er startskjerm, passordskjerm, administratorskjerm, og en legg-til-bruker skjerm.

Startskjerm

Startskjermen som er vist i figur 4.15 er et enkelt vindu med to knapper og en Gif. De to knappene er "Add User" og "Admin". Disse knappene sender deg videre til nye vindu. Om man trykker på "Add User" knappen får man opp vinduet hvor man legger til nye brukere. Trykker man på "Admin" knappen kommer det opp et passord-vindu.

Figure 4.15: GUI startskjerm

Legg til bruker-vinduet

Legg til bruker-vinduet inneholder det som trengs for å legge til en person. Det er tre tekstfelter som blir brukt til å skrive inn navn, epostadresse og nummer. En rullegardinliste brukes for å enkelt velge hvilken gruppe man ønsker å legge personen inn i. "Capture" knappen blir brukt til å ta bilde, og dette bilde blir vist i et panel. Om man ikke er fornøyd med bildet kan man trykke på "Retake" knappen som dukker opp oppå "Capture" knappen når et bilde er tatt. Bildet som blir tatt blir lagret lokalt før det blir sendt videre. Dette blir erstattet når en ny bruker legger seg til for at den lokale disken ikke blir fylt opp. Alle feltene må være fylt ut og bildet må være tatt for at man skal kunne sende bildet og informasjonen videre til Azure ved å trykke på "Send" knappen. Vinduet kan du se i figur 4.16.

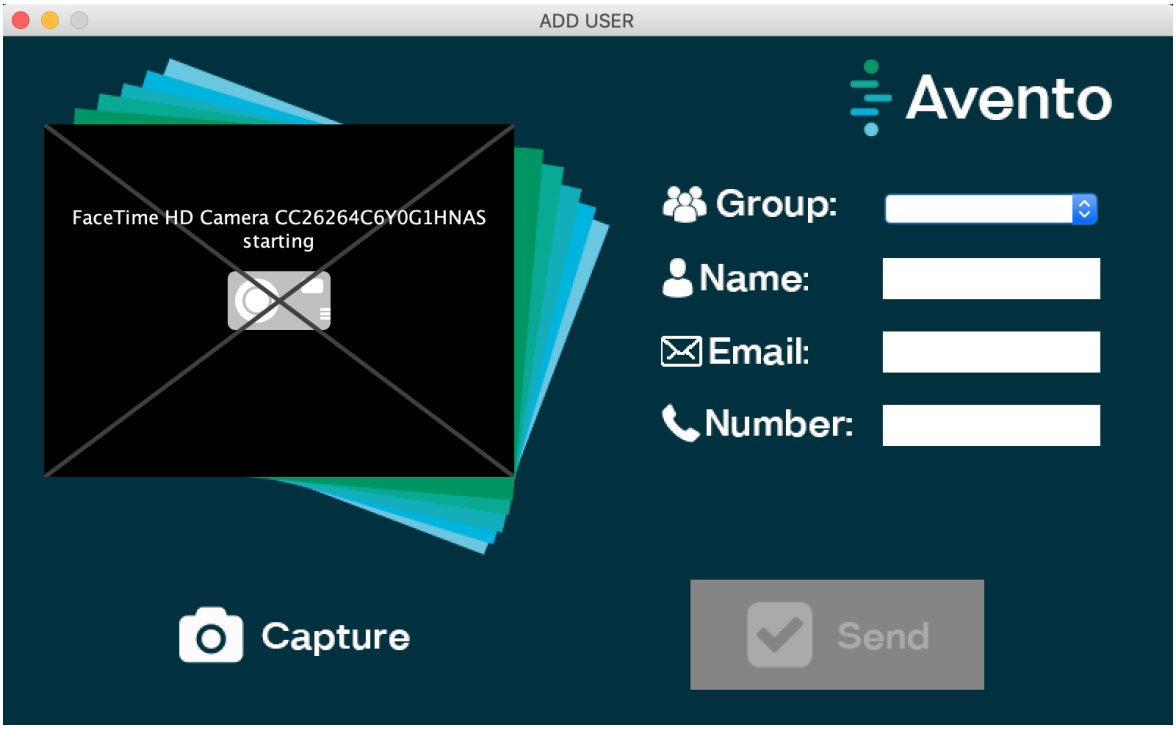


Figure 4.16: Legge til bruker

Restriksjonene til feltene er at alle feltene må være fylt ut korrekt. Dette innebærer at navnefeltet må være minst tre bokstaver, e-mailen må inneholde både "@" og ".", og nummerfeltet må bestå av akkurat åtte tall for å kunne gå videre med å sende. Når "Send" knappen blir trykt ser den først etter hvilken gruppe som er valgt slik at programmet kan legge til bilde av personen i denne gruppen. Det blir også sjekket om personen allerede er i registeret ved å kalle

på *detectFace()* og *identify()* i Azure_GUI. Dersom personen er der fra før vil den nye *FaceID'en* legges til den eksisterende personen som en ny *persistedFaceID*. Om ansiktet til personen ikke er i databasen vil den legge til ny bruker ved å kalle på funksjonene *createPersonToPg()* etterfulgt av *addFaceToPerson()*. Disse funksjonene finner man i klassen *PersonGroupPerson*. Deretter trener vi persongruppen med funksjonen *trainPersonGroup()* i klassen *PersonGroup*. Når personen er lagt til vil det komme opp en informasjonsboks med en QR-kode i. Denne QR-koden kan skannes med QR-applikasjonen som blir beskrevet i avsnitt 4.6.

Passord-vinduet

I dette vinduet må man skrive inn riktig passord for å komme videre. Passordet vi har valgt er "admin". Deretter er det bare å trykke på "Login" knappen eller "enter"-tasten for å komme til administrator vinduet. Om man skriver feil passord vil man ikke komme gjennom og en infoboks vil dukke opp med teksten "Wrong password. Please try again". Figur 4.17 viser passordvinduet.

Figure 4.17: Passord for admin innlogging

Administrator-vinduet

Administrator-vinduet kan man se i figur 4.18. Administratordelen er passordbeskyttet og er kun ment for et fåtall personer i en bedrift. Det er her man finner de administrative metodene. Hovedkomponentene i dette vinduet er to knapper. Når en av disse trykkes dukker det opp flere underkomponenter, avhengig av hvilken knapp som trykkes. Disse to knappene er "Users" og "List Groups".

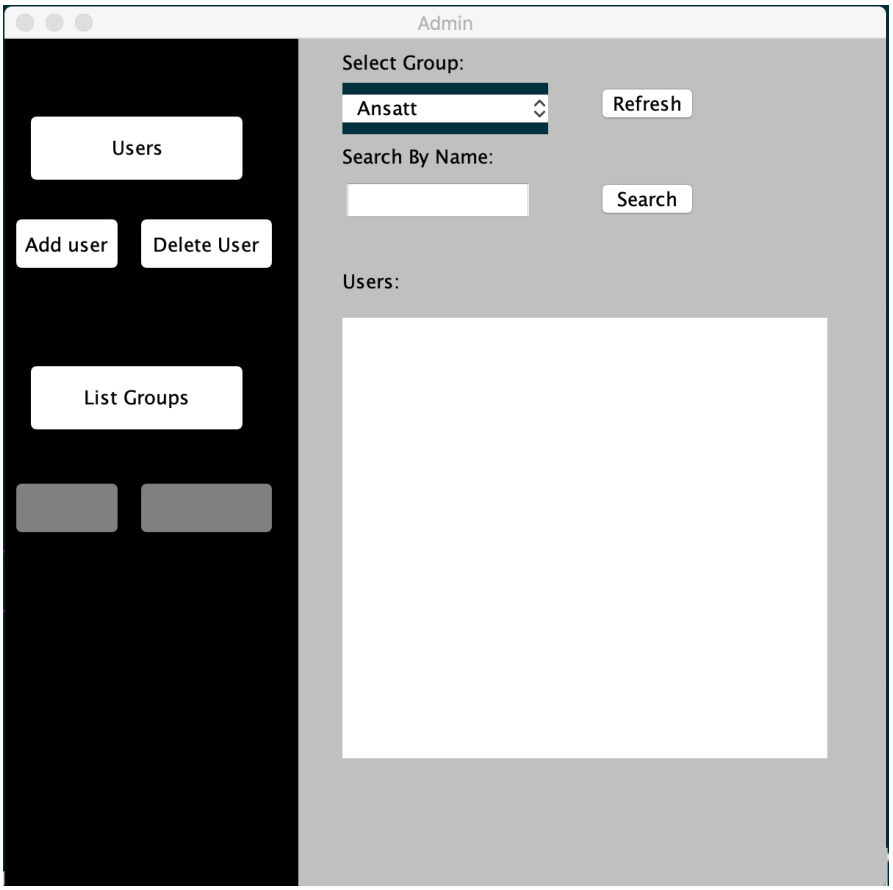


Figure 4.18: Liste/slette brukere

Når "Users" knappen trykkes dukker det opp syv underkomponenter: en rullegardinliste, en tom liste, fire knapper og et tekstfelt. Samtidig vil alt som tilhører "List Groups" skjules. I tillegg gjøres det et kall til Azure som oppdaterer en *ArrayList* som holder på gruppenavnene fra Azure, samt en *HashMap* som holder på gruppenavn med tilhørende gruppeID. Navnene fra ArrayListen legges så inn i rullegardinlisten.

I rullegardinlisten velger man hvilken gruppe man ønsker å søke etter og liste brukere i.

Når "Refresh" knappen trykkes brukes det valgte gruppenavnet til å finne gruppeID. GruppeID brukes deretter i et kall til Azure som oppdaterer en *ArrayList* med navn på personer og en *HashMap* med navn og tilhørende personID. ArrayListen brukes så til å fylle inn det tomme listefeltet. I tillegg tømmes også tekstfeltet, som er et søkefelt. Her kan man søke opp personer i den valgte gruppen.

Når "Search" knappen trykkes skjer det et par ting. Først tømmes den eksisterende listen og en *ArrayList* med navn som matcher søk, for å gjøre plass til søkeresultater. Deretter hentes gruppeID fra gruppenavnet for å oppdatere listen med personer fra Azure. Når dette er gjort har vi tomme lister som er klare for å holde på søkeresultater som matcher søket, og oppdaterte lister over hvilke personer som ligger i systemet. Deretter sjekker vi at søkefeltet ikke er tomt. Dersom det er tomt printes en feilmelding. Hvis feltet ikke er tomt bruker vi en *Iterator* til å iterere over alle navn (i den valgte gruppen) og lete etter navn som inneholder det man søker etter. Dersom man finner navn som matcher, legges disse til i en egen *ArrayList*. Når iterasjonen er ferdig, brukes denne ArrayListen til å fylle inn en DefaultListModel, som igjen legges i listen og vises. Da får man opp alle navn som inneholder det man søkte på.

For å slette en person velger man et navn på listen over personer, og trykker "Delete User". Da hentes først gruppeID fra rullegardinlisten, og deretter kalles metoden *deletePerson()*. I denne metoden blir det gjort et par kall til Azure. Først og fremst vil vi hente ut all info om personen. Deretter går vi gjennom alle *persistedFaceIds* og sletter disse. Når dette er gjort kan personen slettes, og GUI-listen samt ArrayLists og HashMaps oppdateres. En infoboks vil fortelle om slettingen gikk fint.

Når knappen "Add User" trykkes, vil man se en ny skjerm med en knapp som heter "Choose file" som åpner den lokale disken på datamaskinen ved hjelp av en "file chooser". Vinduet er vist i figur 4.19. Her kan man velge et bilde av en person og laste det opp. Det valgte bildet blir vist i et panel under. Dette er nesten helt likt Add User-vinduet, i motsetning til at det er administratoren som må legge til brukeren.

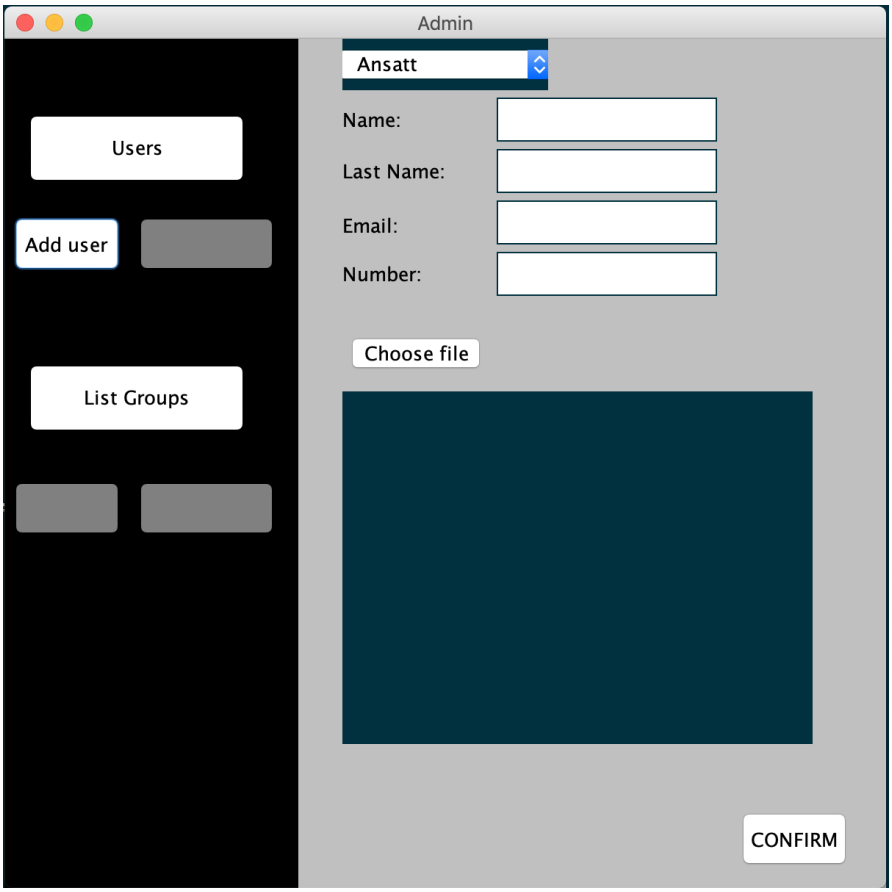


Figure 4.19: Legge til bruker fra fil

Når den andre hovedknappen "List Groups" trykkes, vil en ny skjerm dukke opp. I figur 4.20 ser vi fem underkomponenter: tre knapper, et tekstfelt og en liste, samtidig som alt fra "Users" skjules. Søkefeltet og knappen "Search" fungerer på samme måte som i "Users" delen, bare at ArrayLists og HashMaps, samt listene holder på gruppenavn og gruppeID istedenfor personnavn og personID. "Delete" knappen henter ut gruppeID ut fra valgt gruppenavn og sender denne ID'en i et kall til Azure for å slette gruppen.

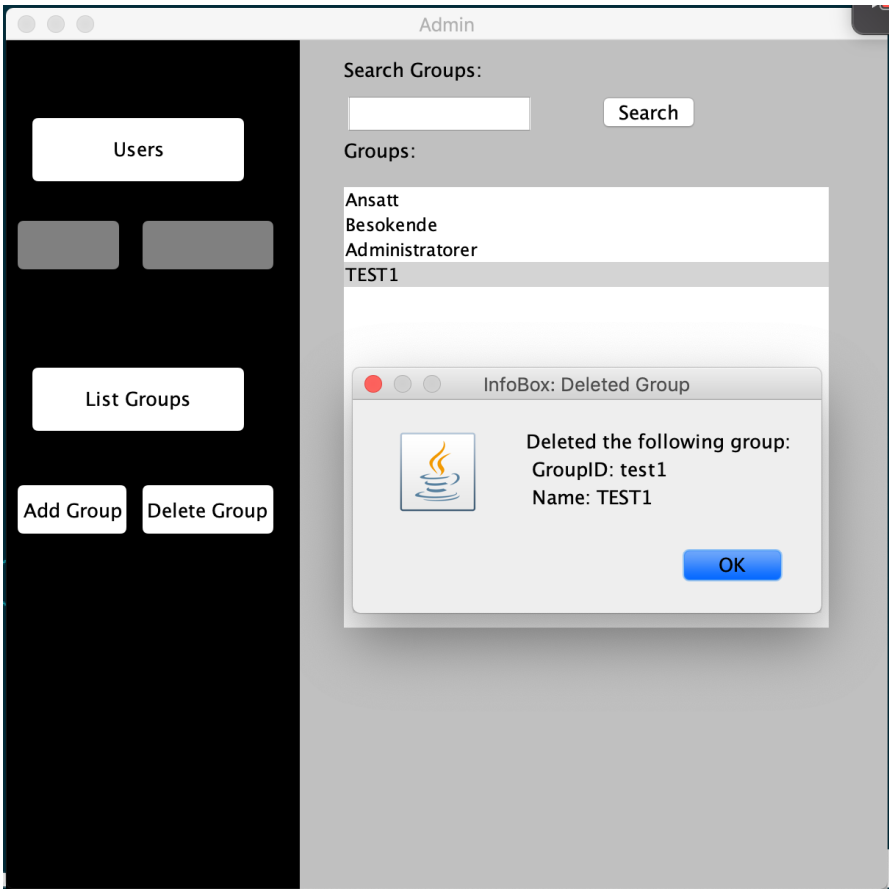


Figure 4.20: Liste/slette grupper

Dersom man ønsker å legge til en ny gruppe gjøres dette ved å trykke på "Add group". Da får man opp en ny skjerm, vist i figur 4.21, med to nye tekstfelt og to nye knapper.

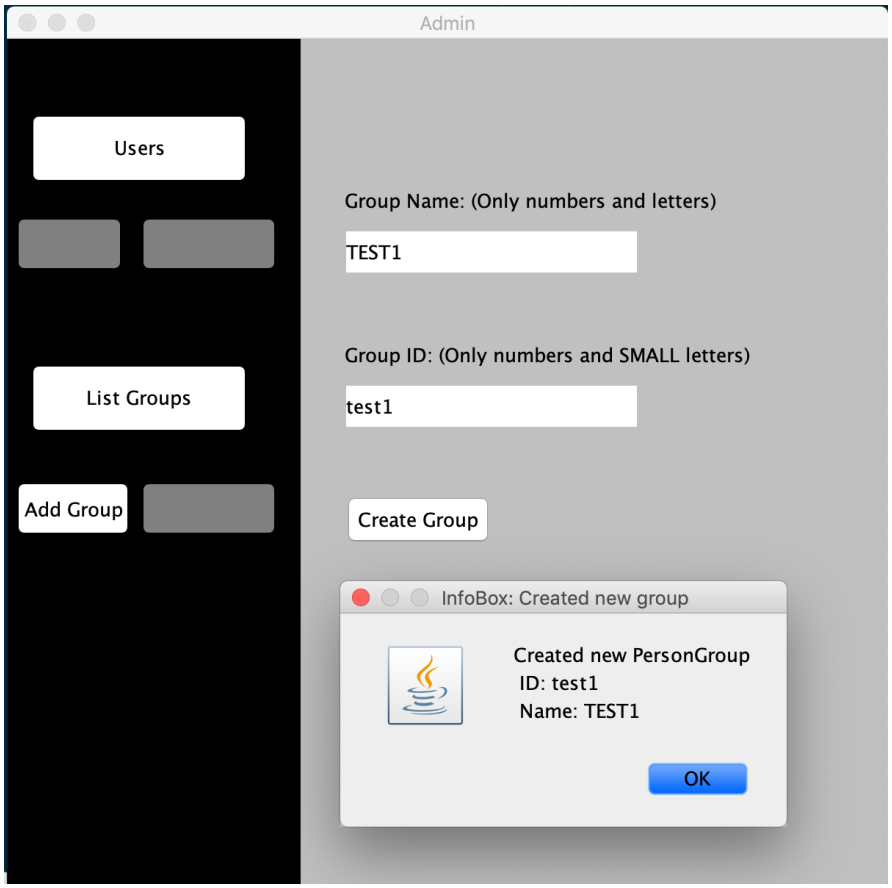


Figure 4.21: Opprette nye grupper

Her kan man fylle ut med ønsket gruppenavn (A-Z,a-z,0-9) og gruppeID (a-z,0-9). Deretter trykker man på "Create Group" og det sendes et kall til Azure med ønsket gruppenavn og gruppeID. En popup-box vil fortelle om gruppen ble opprettet på korrekt måte.

4.6 QR-applikasjon

Vi utviklet en applikasjon i programmet XCode skrevet i språket swift. Denne kan bare kjøres på iOS. Appen scanner en QR-kode fra administrasjons programmet når personen legger seg til i systemet. Denne QR-koden blir brukt i tofaktorautentiserings prosess som står beskrevet i avsnitt 4.9.4.

Vi implementerte først en sekssifret kode som brukeren måtte taste inn på skjermen. Vi syntes dette var en dårlig løsning og gikk fort over til et QR-passord generert i appen, som man kan vise til systemet. Med denne løsningen ville vi gjøre prosessen mye raskere, og man elim-

inerer faren for brukerfeil.

4.6.1 Klasser

Programmet består av fem klasser. Mye av det grafiske er hentet fra tredje-parts bibliotek, som blir nevnt i avsnitt 3.5.3. Figur 4.22 viser alle klassene som blir brukt i programmet.

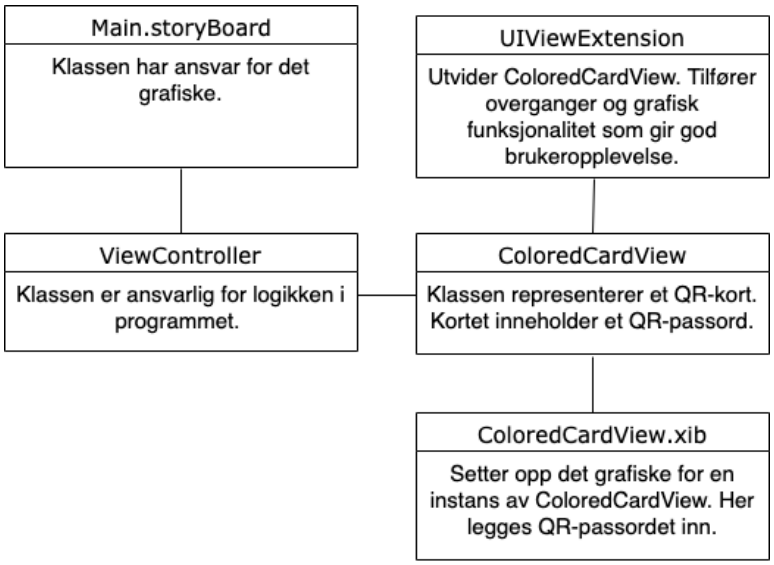


Figure 4.22: Class diagram for QR-applikasjonen

Main.storyBoard

Her setter vi opp den biten av det grafiske som alltid vil være tilstede i applikasjonen. Vi har et tekstfelt øverst hvor det står: *PASSWORDS* og under dette er det en beholder hvor QR-passord kortene kommer opp.

ColoredCardView

Denne klassen representerer et QR-Passord kort. Klassen inneholder alle spesifikasjonene som trengs for å kunne opprette et kort. Den definerer farge, navn på kortet og setter inn QR-passordet på kortet. Her finner vi også logikken til knappen *delete card*, som sletter et kort fra applikasjonen.

ColoredCardView.xib

Denne klassen er en grafiske fremstilling av ColoredCardView klassen, og her settes grafikken til kortet opp. Det består av et tekstfelt helt øverst som holder på navnet til QR-passordet, en bildebeholder som skal holde på QR-passordet, og en *delete card* knapp helt nederst for å slette kortet.

UIViewExtension

Denne klassen er en utvidelse av *UIView* og holder på logikken til å vise kortene og utføre dette med en overgang. Dette gir inntrykk av en mer gjennomført og profesjonell applikasjon.

ViewControler

Denne klassen står for selve logikken. Det er denne klassen som sørger for å lese av QR-koder, og sørge for at det alltid blir generert et nytt engangs QR-passord hvert 30 sekund.

4.6.2 Funksjonalitet

Applikasjonen er laget for å holde på passord for brukeren. Applikasjonen er brukervennlig og responsiv. Når man skal legge til en bruker i systemet via administrasjons applikasjonen får brukeren opp en QR-kode som han skannes med QR-applikasjonen. QR-passordet vil da ligge inne i applikasjonen som sørger for å alltid generere et gyldig passord for brukeren. Se Figur 4.23 for eksempel hentet fra appen. Kortene kan enkelt slettes. Om kortet blir slettet må brukeren tilbake til administrasjonen for å hente ny QR-kode. Applikasjonen kan holde på flere passord tilhørende flere systemer, men vil kun ha et passord for hvert system.

Figure 4.23: Grafisk fremstilling av et QR-passord i applikasjonen

4.7 Bevegelsesdeteksjon

Til løsningen vår trengte vi en enhet til å detektere bevegelses slik at vi kunne starte systemet. Vi laget et program skrevet i Arduino C for til å håndtere deteksjonen ved hjelp av en bevegelsessensor.

4.7.1 IO-Diagram

Figur 4.24 viser et IO-diagram av Arduinobrettet med bevegelsessensoren.

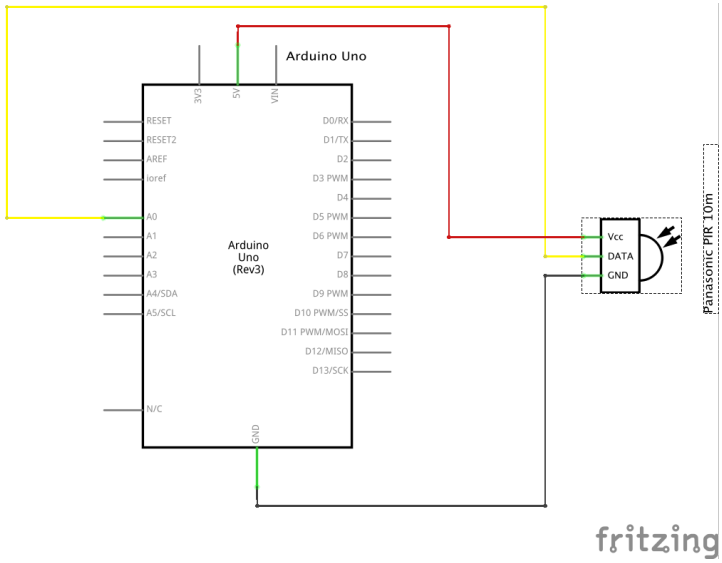


Figure 4.24: IO-diagram av Arduino med bevegelsessensor

4.7.2 Hardware

Mikrokontrolleren Arduino Uno krever lite strøm og er laget for å kjøre repeterende oppgaver. Dette passet fint for en komponent som skal kunne detektere bevegelse hele døgnet. Bevegelsessensoren vi bruker er pålitelig, har bra rekkevidde og ga oss tydelig utslag når den detekterte bevegelse.

4.7.3 Beskrivelse av Arduino-miljøet

Vi valgte å bruke seriell kommunikasjon mellom Java-miljøet som kjører ansiktgjenkjenningen og Arduino-miljøet. Når sensoren gir utslag, vil programmet på mikrokontrolleren gi beskjed til Java-miljøet om å sette i gang ansiktgjenkjenningen.

Sensoren gir en verdi fra 0 til 1023 på bevegelse, der 1023 er "maks" bevegelse. Etter mye testing så kom vi fram til at 1000 var en god terskel å ha for at ansiktgjenkjenningen skulle kjøres. Vi valgte en relativt høy terskel for å eliminere mest mulig støy og redusere belastningen på resten av systemet.

Metoden *sendString()* sender en *String* når verdien er over 1000. I java-miljøet leses kun verdien når programmet står i default. Når Java-programmet blir trigget av bevegelsessensoren ignoreres verdiene som blir sendt mens prosessen pågår. *sendString()* sender en *String* "1". Figur

4.25 viser Arduino C koden til programmet.

```
void loop(){

    readSensor();

    if (sensorValue > 1000){
        sendString("1");
    }
}

void sendString(String msg){
    Serial.print(msg);
    Serial.flush();
}
```

Figure 4.25: Kodesnutt fra bevegelsesprogrammet

4.8 Design

I denne seksjonen går vi inn på hvordan vi valgte å designe de forskjellige grafiske elementene i systemet.

4.8.1 Illustrasjonsskjerm

For å ta brukeren igjennom prosessen for ansiktsgjenkjenningen satt vi opp illustrasjoner på skjermen som brukeren må følge. Om en bruker ikke er tilstede vil skjermen stå i ventemodus. Denne skjermen er satt opp til å kjøre en gif med logoen til Avento i fokus. Når en bruker blir detektert av bevegelsessensoren vil skjermen gå over til å guide brukeren nærme nok. På skjermen vil det stå "Come closer..". Når brukeren er nærme nok, går skjermen over til å illustrere at den jobber med prosessen. Om brukeren ikke finnes i systemet vil skjermen vise en lås som ikke åpner seg. Om brukeren finnes i systemet vil skjermen enten gå over til å vise at den trenger et QR-passord, eller gå rett i skjermen som låser opp for brukeren. Denne skjermen viser en lås som åpner seg. Når låsen har åpnet seg bytter den til en skjerm som sier "Welcome" pluss navnet til brukeren som er godkjent. Skjermen går tilbake til venteskjermen når prosessen er

over, eller hvis prosessen blir avbrutt underveis på grunn av tidsbruk.

Vi har valgt å ha fokus på Avento sine farger og enkle illustrasjoner. Vi ville at prosessen skulle være enkel å forstå og stilren i form av farger og ikoner. Brukeren får på denne måten et godt inntrykk av hvor man er i prosessen, og om man trenger å gjøre noen endringer i form av avstand.

4.8.2 Administrasjons program

Administrasjons programmet er designet i Avento sine farger. Vi har valgt å gjøre programmet brukervennlig i form av tekst og ikoner. Man skal kunne bruke programmet uten å måtte trenge noen form for opplæring. Alle knapper og felter er illustrert med et ikon som skal illustrere hva brukeren trenger å oppgi/gjøre. Knappene og feltene er satt opp på en logisk måte, med brukervennlige instrukser.

4.8.3 QR-passord applikasjon

QR-passord applikasjonen er designet med nøytrale farger. Applikasjonen er ikke designet for Avento, men laget mer som et generelt verktøy som kan brukes i mange systemer. Instruksene er tydelige og brukervennligheten er høy. QR-passordene vil legge seg kategorisert etter når de ble lagt inn, og det er enkelt å manøvrere mellom de forskjellige passordene. Se figur 4.26 for illustrasjoner hentet fra applikasjonen.

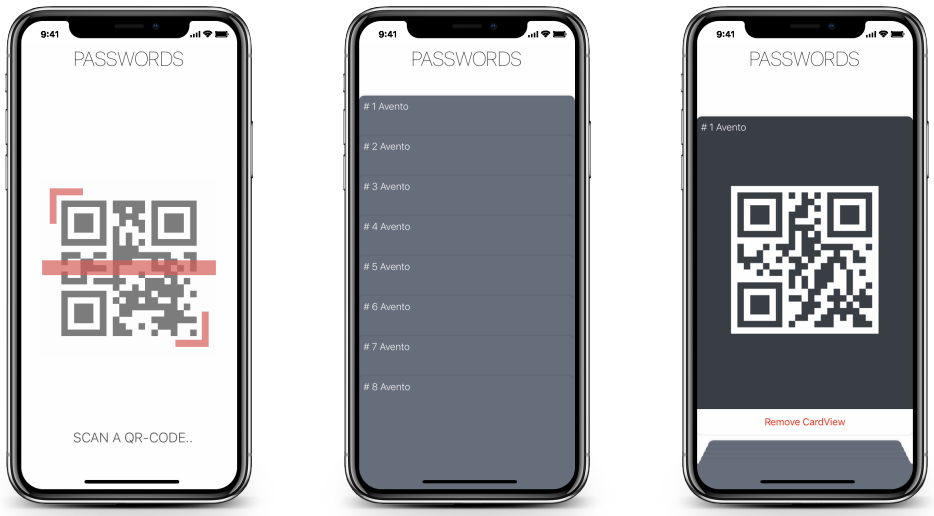


Figure 4.26: Grafisk fremstilling av Qr-passord applikasjonen

4.9 Sikkerhet

Høy sikkerhet må ligge til grunn for et hvert låsesystem. Vi har sørget for at systemet holdt et forsvarlig nivå med tanke på sikkerhet. Spesielt viktig var det at systemet vårt ikke slapp inn personer som ikke skulle ha tilgang. I dette avsnittet går vi igjennom de viktigste elementene som sørger for at sikkerheten blir opprettholdt.

4.9.1 HTTPS

Både BioID og Azure bruker HTTPS protokoll. HTTPS blir omtalt i teori under avsnittet 2.6.3. Dette gjorde at vi fint kunne sende av gårde person data til Azure og BioID uten at vi trengte å bekymre oss for tyvlytting eller endring av dataen som blir sendt.

4.9.2 Liveness

Vi valgte BioID for å opprettholde sikkerhet til systemet. Tjenesten står beskrevet i avsnitt 3.6.1. BioID ga oss muligheten til å kunne skille ut bilder og videoer av personer. På denne måten kunne vi med høy sikkerhet si at personen som sto foran kameraet var en ekte person. BioID har

en liten sårbarhet hvis det blir vist en video av en person i høy oppløsning med god belysning.

4.9.3 Azure

Azure regner ut en likhetskonfidens når bilder blir sammenlignet i databasen. Azure har i sine eksempler satt denne til 0,5 som utgangspunkt for hva som er sikkert nok. Er denne konfidensen for lav vil personer som ikke ligner bli oppfattet som samme person. Om konfidensen er for høy så kan det hende at personer ikke vil bli oppfattet som lik seg selv, og dermed ikke få tilgang. Vi valgte derfor å gjøre det samme som Azure.

4.9.4 Tofaktorautentisering

Vi implementerte en ekstra sikkerhetsløsning til systemet vårt. Dette baserer seg på TOTP algoritmen beskrevet i avsnitt 2.6.2. Når brukeren oppretter seg selv, vil man få ut en QR-kode som inneholder en "hemmelighet". QR-koden scannes med QR-appen. Systemet og appen vil da dele den samme hemmeligheten og kan generere riktig passord på hver sin side uten noen form for kommunikasjon. Figur 4.27 viser en fremstilling av hvordan prosesssen fungerer.

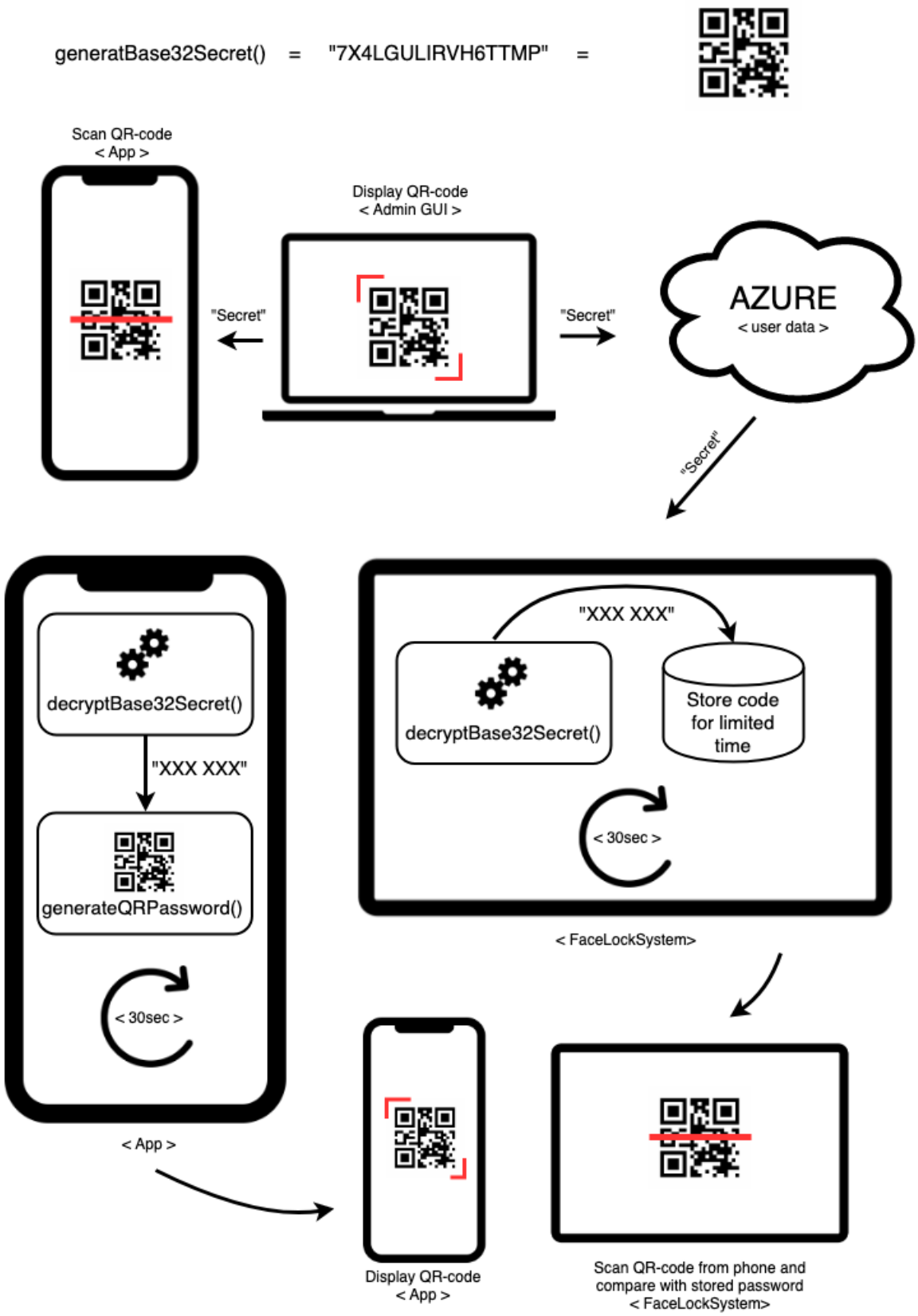


Figure 4.27: Grafisk fremstilling av prosessen for å generere QR-passord

4.10 Tidsbruk

Tiden det tar for å låse seg gjennom døren er avhengig av flere faktorer. Først og fremst er det avhengig av hvordan programmet er bygd opp. Elementer som også tar mye tid er diverse kall til skytjenester. Men tidsbruken blir også preget av plassering av brukeren fremfor kameraet.

Vi har testet hvor lang tid prosessen fremfor kameraet tar for personer som ligger i systemet. Hvordan testen er utført forklares i avsnitt 3.8.

Gjennomsnittlig tid på prosessen var 4,29 sekunder. Resultatene varierer fra 3,3 til 5,5 sekunder. Under testen ble alle kall til BioID godkjent på første forsøk. Det er viktig at omgivelsene har god belysning, og at brukeren står med riktig avstand fra kameraet slik at bildene som sendes til BioID er gode. I tilfellet hvor bildene ikke er gode og man må sende nytt kall til BioID, vil det ta cirka 2 sekunder ekstra for hvert kall til BioID. Med "gode" bilder så menes det at ansiktet er i fokus og dekker store deler av bildematrisen.

4.10.1 API-kall

Løsningen vår har flere API-kall. Tiden slike kall tar er preget av internettforbindelsen.

Det er som sagt gunstig at kallet til BioID lykkes på første forsøk for at prosessen skal gå fort. Vi har derfor satt begrensinger i koden, for bildene som skal sendes til BioID. Vi har blant annet lagt inn at ansiktet må dekke 1/3 av pikslene i bildematrisen. Er man for langt i fra så vil den visuelle veilederen gi beskjed om å komme nærmere.

Vi gjør også om størrelsen på bildene som blir tatt. De blir gjort mindre slik at det går raskerer å sende og prosessere bildene.

Kapittel 5 - Drøfting

I dette kapittelet skal vi diskutere resultatene vi har oppnådd i prosjektet, og selve utførelsen. Vi vil også snakke om andre potensielle måter å løse oppgaven på.

5.1 Oppkobling mot låsesystem

Etter møtet med Avento og Låseservice var planen at Låseservice skulle stille med et låsemiljø vi kunne koble programmet opp imot for å teste løsningen. Dette ble det aldri noe av, fordi Låsservice nedprioriterte dette. Vi fullførte likevel prosjektet og implementerte de løsningene som vi trengte. Produktet vi endte opp med er klart til å kobles opp mot et låsemiljø. Vi fikk heller ikke lov til å koble oss opp mot NTNUs låsesystem av sikkerhetsmessige årsaker. Da valgte vi å bruke skjermen til å visualisere om døren låses opp eller ikke.

5.2 Tekniske resultater

I denne seksjonen skal vi diskutere de tekniske resultatene vi har oppnådd.

5.2.1 Bevegelsessensor

Vi bestemte tidlig at vi trengte en bevegelsessensor og denne ble bestilt fortløpende. Tanken var å koble denne opp mot en Arduino som tok seg av avlesning fra sensoren og videresendte resultatet til systemet. På et tidspunkt ville vi kutte ned antall komponenter i systemet og ønsket å koble bevegelsessensoren direkte på Odroiden som vi på et stadie skulle bruke til å kjøre systemet på. Dette lot seg ikke gjøre da vi manglet et shifter shield for Odroid. Dette hadde svært

lang leveringstid og vi valgte da å beholde Arduinoen.

På et tidspunkt vurderte vi å utføre bevegelsesdeteksjon i software. Vi ville da tatt *frames* hentet fra kamera og sammenlignet det første mot det neste. Hadde vi sett endringer i konturene i bilde kunne man konkludert med bevegelse. Vi valgte å ikke gå for denne løsningen da det ville vært krevende for systemet å la kameraet kjøre hele tiden, samt utføre de nødvendige prosessene på bildet for å kunne se etter konturendringer. Vi bestemte oss da for å la Arduinoen ta seg av denne oppgaven, da slike mikrokontrollere er laget for repeterende oppgaver. Systemet vårt kunne da ligge i en *default* tilstand frem til den ble "vekket" av mikrokontrolleren. Denne løsningen ble da implementert og fungerer godt.

5.2.2 Ansiktsgjenkjenning

For å kunne utvikle et godt system for ansiktsgjenkjenning trengte vi en god og trygg måte å detektere og analysere personene som sto foran systemet vårt på. Vi måtte utvikle eller implementere en løsning som kunne ta et bilde av personen for så å sjekke om denne personen skulle ha tilgang eller ikke. I vårt første møte med Avento, tipset de oss om Azure og en tjeneste kalt Face API. Vi tilegnet oss fort kunnskap om bruk, implementasjon og avgrensinger for denne tjenesten.

Vi så også på andre alternativer for å utføre ansiktsgjenkjenning. Dette innebar bruk av kunstig intelligens, eller andre tjenester. På et tidspunkt i prosessen vurderte vi å bygge en egen kunstig intelligens lokalt, for å kunne trene personer og holde på de personene som skulle ha tilgang. Vi valgte å gå bort fra denne løsningen da det ville bli tidkrevende å innhente data, samt trene modellen. Når det kom til andre tjenester, så var det ingen som kunne matche det Azure kunne tilby.

Vi valgte da å gå for Azure. Tjenesten var rask og ga oss trygghet på at den ville gi oss rett svar når vi sendte av gårde et bilde av en person. I løpet av testingen og utviklingen har vi aldri opplevd et feil svar fra Azure. Når systemet fikk se en person som ikke skulle ha tilgang fikk vi returnert null, og det var tydelig at systemet ikke var i nærheten av å slippe inn personer som ikke skulle ha tilgang.

Azure ga oss også muligheten til å ha store deler av systemet i skyen. Dette gjør systemet mer skalerbart og vi trenger ingen form for lokal lagring av personene i databasen.

5.2.3 Liveness deteksjon

En stor utfordring med oppgaven var å detektere om person som stod foran systemet vårt var en ekte person eller ikke. Azure hadde ingen måter å skille en person fra bilde eller video. Dette førte til et stort sikkerhetshull i systemet, om man ikke implementerte en form for liveness deteksjon.

Vi brukte mye tid på å lete etter løsninger i forbindelse med liveness. Tidlig i prosessen begynte vi å se på hvilke løsninger Apple har brukt i sin ansiktsgjenkjenning, samt Windows i deres *Windows Hello* implementasjon. Det viste seg at begge selskapene bruker 3D-skanning i form av IR-sensorer og utformer en 3D modell av ansiktet til brukeren. De gjør trolig dette ved bruk av point cloud. Vi ønsket å ta i bruk samme løsning, men kun for deteksjon på om personen er ekte eller ikke. Systemet vårt er designet for å holde på personer hvor antallet kan strekke seg oppmot 10 000, og det å sammenligne 3D modeller på 10 000 personer vil være ekstremt tidskrevende og tungt å prosessere. Vi ønsket derfor kun å detektere at personen foran kameraet var en ekte person ved å sjekke for dybde i ansiktet. Hadde vi fått bekreftet at dette var et ekte ansikt ville vi hentet ut et bilde fra samme ROI og sendt dette til Azure.

Løsningen med point cloud viste seg å være mer tidkrevende enn vi hadde sett for oss, og kameraet vi kjøpte var avhengig av et bibliotek som kun var støttet i C++. Dette førte til at vi måtte begynne å lære oss et nytt språk, med ny syntaks. Planen var å skrive point cloud koden i C++ og opprette kommunikasjon med Java som håndterte alt annet. Alternativet var å skrive hele programmet i C++, men dette ville blitt enda mer tidkrevende.

Vi var også innom andre løsninger for å håndtere liveness problematikken. Vi begynte å trene en kunstig intelligens på å detektere forskjellen på et bilde/video av en person og en ekte person. Modellen bygde på to datasett, hvor det ene inneholdt bilder tatt direkte av personen, mens det neste var bilde av en mobilskjerm som viste det samme bildet. Ideen var å trene modellen på å kunne skille mellom de bildene som kom fra mobilen og de som var ekte. Vi utviklet en modell som gjorde nettopp dette. Programmet var skrevet i Python, og kunne i sanntid forutsi om personen var ekte eller ikke. Programmet var dessverre ganske ustabilt, og opprettholdt ikke de sikkerhetskravene vi ønsket i en liveness deteksjon. Dette er i hovedsak grunnet dårlig datasett. Vi kunne nok også til fordel endret strukturen til det nevrale nettverket og optimalis-

ert dette for vår løsning. Etter litt testing av denne løsningen kom vi over en tjeneste som heter BioID.

Vi implementerte denne løsningen i systemet og etter mye testing kunne vi konkludere med at dette var den foretrukne løsningen å bruke i vårt system. Vi kunne med dette også opprettholde ønske om en modulbasert løsning, da hele tjenesten kunne bygges som en modul, og det eneste vi trengte å sende den var to bilder tatt omtrent rett etter hverandre. Tidsrommet mellom det første og andre bildet ble definert utifra en metode som sjekket for bevegelse i bilde. Tjenesten var også skybasert noe som gjorde at vi minsket mengde software vi trengte å ha lokalt.

5.2.4 Administrasjon

I løsningen vår var det viktig for oss å kunne administrere systemet på en god måte. Administrasjons løsningen ble mer eller mindre utviklet underveis, men vi visste at vi trengte et eget program som kunne ta seg av det administrative. Vi ønsket et program med begrenset tilgang, da ikke alle brukere skulle ha tilgang til å administrere systemet. Vi vurderte flere muligheter når det kom til å utvikle programmet.

Vi vurderte blant annet en web-løsning, som kunne kjøre på det lokale nettverket. Fordelene med dette var at de som hadde tilgang kunne administrere systemet fra hvilken som helst enhet på det lokale nettverket. Web-løsningen ville vært tilnærmet lik program-løsningen vi har idag. Vi valgte å gå bort ifra web-løsningen da ingen på gruppen hadde erfaring fra Javascript eller web-utvikling. En liten web-løsning ble etablert, men manglet mye av de essensielle funksjonene administrasjons programmet vårt trengte. Løsningen vi utviklet ferdig ble en *Java Swing GUI*. For videre arbeid vil vi anbefale å implementere en slik web-løsning, da det ville gjort administrering av systemet mye enklere.

5.2.5 Programmeringsspråk og struktur

I denne seksjonen skal vi legge frem resultater basert på programmeringsspråk og struktur i prosjektet.

5.2.5.1 Java

Vi ønsket på et tidlig stadie og skrive programmet vårt i Java. Dette er i hovedgrunn fordi Java er språket majoriteten på gruppen kunne best. Vi ønsket å utvikle en løsning med objektorientert programmering som ryggrad. Iløpet prosessen var vi innom C++ og Python, men vi valgte å ikke implementere noen løsninger i noen av disse språkene, da løsningenene vi valgte kunne skrives i Java. I starten var det først og fremst funksjonalitet som sto i høysetet. En av de første løsningenene var kun en *while-loop* hvor mye av logikken og metodene var implementer direkte i loopen. Dette var en løsning som var tungt preget av mye *if-statements*. Denne løsningen fulgte heller ikke prinsippene for *cohesion* og *coupling*. Når den overordnede strukturen var mer definert og vi visste hvilke tjenester og metoder vi trengte, utviklet vi klasser og metoder som moduler. Disse modulene ble brukt i en hovedklasse som ble bygget opp som en tilstandsmaskin. Tilstandsmaskinen ble satt inn i en while(true) loop, med en *default* tilstand som sjekker etter en verdi fra bevegelsessensoren. På denne måten bruker programmet vårt minimalt med ressurser før den vet at det er en person tilstede. Ved å implementere en tilstandsmaskin hadde vi også bedre kontroll på hvor i prosessen vi var og det var enklere å repetere steg i prosessen. Vi gjorde også programmet betraktelig mer modulerbart da vi kunne opprette en tilstand for hver modul. Ved å omstrukturere koden oppnådde vi høyere grad av *cohesion* og *coupling*, samt en mye bedre kodestil.

Vi valgte også å implementere *concurrency* som gjorde at vi kunne kjøre TOTP-algoritmen og generere passord, samtidig som vi leste av QR-passordene som ble vist av brukeren når dette ble påkrevd. Tidligere løsninger av systemet hadde ikke mulighet til å gjøre dette.

5.2.5.2 Arduino C

Vi laget en løsning i Arduino C hvor vi hadde to-veis seriell kommunikasjon. Dette var fordi vi ville hindre Arduino fra å sende verdien flere ganger til systemet vårt. Vi gikk etterhvert bort fra løsningen med to-veis kommunikasjon, da vi konkluderte med at systemet ikke tok noen form for belastning ved at vi sendte verdien flere ganger. Vi vekter derfor pålitelig en-veis kommunikasjon, og eliminerte to-veis kommunikasjonen. Vi testet også forskjellige terskeler for når verdien fra bevegelsessensoren skulle tolkes, og valget falt på "1000". Bevegelsessensoren gir

maks utslag "1023" så fort den oppfatter moderat bevegelse.

5.2.6 Sikkerhet

Sikkerheten i et låsesystem er kritisk. Systemet vårt skal regulere adgangen til et område hvor mange mennesker har tilgang.

5.2.6.1 Liveness

Liveness-deteksjon var en avgjørende faktor for at systemet vårt skulle opprettholde en viss standard for sikkerhet. I starten hadde vi tenkt til å implementer point cloud. Hadde vi kunnet implementere point cloud hadde systemet sannsynligvis hatt en enda høyere form for sikkerhet enn vi klarer å oppnå med Liveness-deteksjon via BioID. Vi oppdaget et lite sikkerhetshull som gjør det mulig, under perfekte omstendigheter å lure systemet sin liveness deteksjon. Dette kan gjøres ved å ta et filmopptak av en bruker som finnes i systemet og holde denne filmen på en nøyaktig avstand fra kameraet. Mobilen må også beveges litt for å trigge bevegelses deteksjonen, og kvaliteten må være meget høy. Spoofing på dette nivået anslår vi som meget usannsynlig da denne type materiale er vanskelig å få tak i, med mindre man er veldig godt beslektet eller har et godt bekjentskap til personen som har tilgang i systemet. Dette anslår vi som den største svakheten til systemet. Ved bruk av point cloud kunne vi minimert risikoen ved spoofing betraktelig. Men basert på hva vi har erfart er løsningen vi har valgt å definere som meget sikker.

5.2.6.2 HTTPS

Vi håndterer mye brukerdata, og det er viktig at vi håndterer dette på en god måte. Siden vi i hovedsak bruker skytjenester til å oppbevare vår brukerdata, trengte vi som utviklere å gjøre relativt lite for å sørge for at dette blir opprettholdt. Vi kommuniserer med skytjenestene våre over REST med HTTPS, noe som gjør at det er *end-to-end* kryptert. Vi kan på denne måten både laste opp og hente ned brukerdata på en sikker måte uten å være redd for at noen av brukerdataene kommer på avveie. Ved bruk av skytjenester sitter systemet på svært lite brukerdata.

5.2.6.3 Tofaktorautentisering

På bakgrunn av det som er tidligere nevnt i oppgaven om spoofing valgte vi å implementere en ekstra sikkerhet. Vi hadde lenge en idé om at systemet kun skulle bruke ansisktsgjenkjenning, men på grunn av problemet med spoofing under liveness deteksjonen, bestemte vi oss for å implementere ekstra sikkerhet. Vi ønsket da å implementere en løsning som økte den generelle sikkerheten til systemet, men samtidig kunne fungere som en offline løsning. Etter mange alternativer kom vi frem til tofaktorautentisering, basert på TOTP-algoritmen. Denne er forklart under avsnittet 2.6.2. På denne måten kunne brukeren bruke mobiltelefonen sin til å autentisere at personen var ekte, samt at vi fikk en offline-løsning. Implementasjonen ble vellykket og fungerer som forventet opp mot systemet. Det ble kun utviklet en iPhone-applikasjon. Vi valgte kun IOS, fordi det opprinnelig ikke var en del av løsningen, men kunne implementeres raskt pga kjennskap til Swift og Xcode.

5.2.7 Design

Vi har designet flere løsninger igjennom oppgaven. I hovedsak har det vært utforming av administrasjons programmet, men vi har også måttet utvikle en illustrasjonsskjerm for veiledning ved ansiktsgjenkjenningen. Mot slutten tok vi også for oss å designe en iOS-applikasjon.

Alle design er laget for å kunne gi god brukervennlighet. Vi har hatt fokus på at alle våre applikasjoner og illustrasjoner skal være selvforklarende og gi brukeren en god opplevelse. Vi har diskutert flere løsninger for hvordan vi ønsket å designe applikasjonene, men er meget fornøyd med hvordan designet har blitt utformet i alle de forskjellige applikasjonene og illustrasjonene.

5.2.8 Tidsbruk

I et låsesystem er tiden veldig viktig. Brukeren av systemet vil ha en god opplevelse og ikke føle at prosessen er treg. Ofte er brukeren vant til nøkkelkort eller lignende, og dette er prosesser som går veldig raskt. Et system med ansiktsgjenkjenning må da opprettholde noen krav i forhold til tidsbruk for å kunne konkurrere med et slikt system. Vi hadde flere aspekter ved prosjektet som kunne påvirke tidsbruken til systemet. Det som påvirket tidbruken mest var REST API kall til BioID. Tjenesten brukte mellom 1-2 sekunder fra API kallet ble sendt til vi fikk respons. Vi

hadde et overordnet mål på under 6 sekunder når det kom til respons. Dette klarte vi å oppnå kun når BioID godkjente de første bildene vi sendte. Vi kunne oppnådd et raskere system om vi hadde brukt en mer pålitelig tjeneste enn BioID, eller en annen type liveness deteksjon som for eksempel point cloud.

5.2.9 Manuell testing

Vi brukte mye tid på testing. Dette gikk mest på optimalisering og sikkerhet. Vi bruker en god del bildebehandling for å optimalisere systemet vårt. Dette førte til at vi brukte mye tid i koden på å finjustere parametere slik at bildene vi tok ble perfekte hver gang. Vi bruker funksjoner som *cropping* og *resize*. Disse funksjonen i kombinasjon med *haar-cascade* førte til mye *fake* materiale. Vi løste dette problemet ved å kun prioritere det ansiktet som programmet vårt estimerte som størst. Videre ble disse bildene sendt til BioID. Vi testet ut en mengde forskjellige måter å sende bilder på, inkludert følgende:

- *Resize* bildene
- *Crop* rundt ansiktet
- Diverse filter
- Endre bilde til *grayscale*
- Diverse kombinasjoner av punktene over

Etter mye testing på området endte vi opp med at en enkel *resize* hvor vi halverer størrelsen på bilde ga oss best resultat.

Vi utførte også tester med tanke på spoofing blant annet med diverse bilder og videoer. Som oftest ville den ikke godkjenne video, men den ga oss godkjent i spesielle tilfeller. Dette drøftet vi i avsnitt 5.2.6.

5.2.10 Eksisterende løsninger

Som nevnt tidligere i oppgaven finnes det ikke mange lignende kommersielle løsninger på området, og vi velger derfor å se bort ifra disse. Apple og Microsoft derimot har ansiktsgjenkjenning

på noen av sine produkter. Det som skiller disse fra vår løsning er at disse løsningene baserer seg på 1:1, mens vår løsning kan holde på opptil 10 000 personer, og flere dersom vi hadde byttet ut *PersonGroup* med *LargePersonGroup*. Både Apple og Microsoft har tatt i bruk IR-skanning til å genere en 3D model av brukeren. Dette minner om løsningen vi har diskutert med point cloud, men kan ikke si med sikkerhet at det er slik disse fungerer.

5.3 Prosjektgjennomføring

I løpet av prosjektet har gruppen lagt ned mange timer og utrolig mye arbeid for å etablere et godt system. I denne seksjonen vil vi drøfte selve gjennomføringen av prosjektet.

5.3.1 Metodikk

Vi startet tidlig med å utvikle en fremdriftsplan for prosjektet. Hver uke startet med et møte hvor vi gikk gjennom hva som var blitt gjort og ble enige om arbeidsfordelingen videre. Noe av det vanskeligste var å beregne hvor mye tid hver oppgave ville ta. Når det kommer til utvikling av software er det vanskelig å estimere tidsbruken, og vi havnet ofte i situasjoner hvor enkelte oppgaver tok mye lengre tid en først antatt.

Siden vi var en gruppe med fire automasjonsstudenter hadde vi lite kunnskaper om hvordan man jobber profesjonelt i et software team. Vi fikk tidlig inntrykk av dette når veileder Girts introduserte oss for en del software metodikk som vi ikke hadde brukt tidligere.

5.3.2 Organisering

Vi valgte å ikke ha noen gruppeleder, men heller hyppige møter. Dette gjorde at vi sammen hadde full kontroll hvor hvilke deler av prosjektet som ble jobbet med, og hvilke person som hadde ansvar for denne delen. Dette ble oppdatert fortløpene i Asana. Organiseringen funket greit, og vi hadde mye selvstendig arbeid. Vi jobbet alltid på samme sted, noe som førte til at vi hurtig kunne løse problemer som oppstod sammen, dersom det trengtes.

5.3.3 Versjonkontroll

Vi valgte å bruke Git som versjonskontroll når det kom til kode. Vi hadde brukt Git lite og dette førte til at endringer ikke ble pushet så ofte som de skulle. Derimot funket det bra når vi var flinke til å *pushe*. Git gjorde det mulig for oss å endre deler av samme kode samtidig, og gjorde arbeidsflyten bedre. Om vi hadde satt oss mer inn i bruk av forskjellige *branches* kunne vi oppnådd et enda høyere nivå av struktur i versjonskontrollen.

5.3.4 Kildebruk

I prosjektet har vi hatt fokus på å bruke anerkjente kilder. Vi har for det meste forholdt oss til offisiell dokumentasjon fra de forskjellige tjenestene og bibliotekene.

I de tilfellene vi trengte informasjon vi ikke kunne hente lokalt fra skolen eller i offisielle biblioteker, har vi brukt diverse nettsider. Vi har brukt disse kritisk, kryssjekket informasjonen, og laget løsninger basert på informasjonen.

5.4 Videre arbeid

Vi har utviklet et velfungerende system, men det er store muligheter for forbedring. Et alternativ for liveness deteksjon vi har brukt kunne vært å implementere point cloud. Dette vil eliminere de største problemene vi møtte i forhold til spoofing, og muligens også gjøre systemet raskere. Det vil også være naturlig å få administratorapplikasjonen over på en webbasert løsning, slik at alle som er administrator kan ha tilgang til systemet uavhengig av hvor de befinner seg.

Det er mange muligheter hvor vi kobler systemet opp mot applikasjoner. En idé til en applikasjon er om de administratoren på Avento har oversikt over hvem som er inne i lokalene, og til hvilken tid de kom. En utfordring her kan være om flere personer kommer gjennom døren samtidig, noe som vil værer sannsynlig på begynnelsen av en arbeidsdag hos en bedrift.

En mulighet for utvidele av systemet kan være funksjoner som gjør at brukere blir lagt til på tidsbegrensning, eventuelt at noen brukere bare har tilgang innen et visst tidsrom.

5.4.1 Videre arbeid på QR-applikasjon

En funksjon som ville tatt applikasjonen videre er å koble den opp mot stedstjenester til de telefonene som blir brukt. Tanken er at når du nærmer deg den adressen eller område som du skal låse deg inn på, så vil det komme en varsel på telefonen som gjør at du kan åpne QR-koden rett fra varselen. Dette er for at prosessen fremfor døren skal gå så fort som mulig.

Kapittel 6 - Konklusjon

Avento ønsket seg et låsesystem hvor de ansatte kunne låse seg inn til lokalene på jobb ved hjelp av ansiktsgjenkjenning. Vi har utviklet et program for akkurat dette. De har gitt oss relativt frie tøyler for hvordan vi vil løse oppgaven, men ønsket et system som ikke er mulig å spoofe. I dette prosjektet har vi valgt å vektlegge sikkerhet, tidsbruk, og mulighet for utvidelse og videreutvikling.

Brukere kan legge seg inn i systemet med administratorens grensesnitt. Grensesnittet gjør det mulig å administrere systemet, som å legge til og fjerne personer, samt ha en god oversikt over brukere. Vi har også designet en visuell fremstilling som veileder brukeren gjennom prosessen i å låse seg gjennom systemet.

Løsningen vår er basert på skytjenester hvor BioID og Azure cognitive services er brukt. Disse tjenestene sørger for at personen som låser seg gjennom systemet er ekte, og at de ligger i databasen for folk med tilgang. Vi fikk dessverre ikke muligheten til å teste systemet opp mot et reelt låsemiljø.

Vi valgte å utvikle en applikasjon for tofaktorautentisering. Dette gir høyere sikkerhet og mulighet til å bruke låsesystemet selv om man skulle miste internettilgangen. Som et resultat av en modulerbar løsning så kan denne velges å bruke eller ikke. Applikasjonen er bare utviklet for iOS, og burde videre også implementeres for Android.

Vi har utviklet et program med mål om høy kodekvalitet. Det har vært fokus på en løs og lett kodestil, basert på prinsipper innenfor objektorientert programmering, med lav coupling og høy cohesion. Det har vert ønskelig med et resultat som er enkelt for nye brukere å sette seg inn i og videreutvikle. Resultatet ble et modulerbart system som kan delest opp basert på ønsket sikkerhet og tidsbruk.

I dette prosjektet har vi vært innom mange felter som er nye for oss. Vi har jobbet med nye programmeringspråk og kommunikasjonsprotokoller. Vi har også utviklet en applikasjon, samt fått innsyn i diverse krypteringsmetoder.

Vi har fått en smakebit på hvordan det er å jobbe profesjonelt i team. Det å bruke Git og SourceTree, der flere har arbeidet sammen på forskjellige maskiner, har bydd på utfordringer når det arbeidet skal settes sammen. Vi har vært avhengig av god kommunikasjon og samarbeid. Hver og en av oss har måttet tatt ansvar for å komme i mål med prosjektet.

For videre utvikling av prosjektet mener vi det neste steget er å implementere funksjoner som sørger for at bildene som blir sendt opp til BioID, i større grad vil komme tilbake som godkjent. Videre vil nok den beste veien å gå være å erstatte liveness med point cloud. Vi har vært innom point cloud i løpet av prosjektet, men konkluderte med at det var for tidskrevende å implementere dette.

Løsningen har blitt et raskt, autonomt og modulbasert system som sjekker for liveness, foretar ansiktsgjenkjenning og gir mulighet for tofaktorautentisering. I tillegg er det utviklet et administratorprogram som gjør det mulig å administrere systemet og justere sikkerhetsnivået.

Kildeliste

[1] Avento, "Hva vi gjør." https://www.avento.no/hva-vi-gjor/, 2019. (Funnet Mar 28, 2019).

[2] Apple Support, "Om avansert face id-teknologi." https://support.apple.com/no-no/HT208108, 2018. (Funnet Mar 28, 2019).

[3] Ståle Lindblad, "Ansiktsgjenkjenning på facebook." http://www.stalelindblad.no/2018/05/ansiktsgjenkjenning-pa-facebook/, 2018. (Funnet Mar 28, 2019).

[4] Eirik Rossen, "Biometrisk gjenkjenning." https://snl.no/biometrisk_gjenkjenning, 2017. (Funnet Mar 28, 2019).

[5] Ivar Liseter, "Kunstig intelligens." https://snl.no/kunstig_intelligens, 2018. (Funnet Mar 28, 2019).

[6] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *Comput. Vis. Pattern Recog*, vol. 1, 01 2001.

[7] Wikipedia contributors, "Viola–jones object detection framework." https://en.wikipedia.org/wiki/Viola%E2%80%93Jones_object_detection_framework, n.d. (Funnet Mai 5, 2019).

[8] A. W. S. Perkins and E. Wolfart., "Dilation." https://homepages.inf.ed.ac.uk/rbf/HIPR2/dilate.htm, 2003. (Funnet Mai 19 2019).

[9] Wikipedia contributors, "Gaussian blur." https://en.wikipedia.org/wiki/Gaussian_blur, n.d. (Funnet Mai 6, 2019).

[10] Wikipedia contributors, "Thresholding (image processing)." https://en.wikipedia.org/wiki/Thresholding_(image_processing), n.d. (Funnet Mai 7, 2019).

[11] GitHub contributors, "Image processing: Contours." https://mmeysenburg.github.io/image-processing/09-contours/, n.d. (Funnet Mai 7, 2019).

[12] Eirik Rossen, "objektorientert programmering – it – store norske leksikon." https://snl.no/objektorientert_programmering_-_IT, 2017. (Funnet Mai 18 2019).

[13] Wikipedia contributors, "Java concurrency." https://en.wikipedia.org/wiki/Java_concurrency, n.d. (Accessed on May 19 2019).

[14] Codecademy, "What is rest?." https://www.codecademy.com/articles/what-is-rest, n.d. (Funnet Apr 01, 2019).

[15] Wikipedia contributors, "Tcp." https://no.wikipedia.org/wiki/TCP, n.d. (Funnet May 03, 2019).

[16] Wikipedia contributors, "Internet protocol." https://en.wikipedia.org/wiki/Internet_Protocol, n.d. (Funnet May 03, 2019).

[17] HiST elektro, "Seriell kommunikasjon (uart)." https://histelektro.weebly.com/seriell-kommunikasjon-uart.html, n.d. (Funnet Mar 28, 2019).

[18] Vegard Jansen, "Nå er usb 3.0 her." https://www.tek.no/artikler/naa_er_usb_3_0_her/87110, 2009. (Funnet Mar 28, 2019).

[19] Wikipedia contributors, "Json." https://www.json.org/, n.d. (Funnet Apr 30, 2019).

[20] Wikipedia contributors, "Ascii." https://no.wikipedia.org/wiki/ASCII, n.d. (Funnet May 06, 2019).

[21] Wikipedia contributors, "Qr-kode." https://no.wikipedia.org/wiki/QR-kode, n.d. (Funnet Apr 29, 2019).

[22] Difi, "Begrepsliste: Informasjonssikkerhet." https://internkontroll-infosikkerhet.difi.no/begrepsliste-informasjonssikkerhet, n.d. (Funnet Mai 18 2019).

[23] Wikipedia contributors, "Hmac-based one-time password algorithm." https://en.wikipedia.org/wiki/HMAC-based_One-time_Password_algorithm#HOTP_value, n.d. (Funnet Mai 7, 2019).

[24] Wikipedia contributors, "Unix time." https://en.wikipedia.org/wiki/Unix_time, n.d. (Funnet Mai 19 2019).

[25] Wikipedia contributors, "Time-based one-time password algorithm." https://en.wikipedia.org/wiki/Time-based_One-time_Password_algorithm, n.d. (Funnet Apr 29, 2019).

[26] Wikipedia contributors, "Base32." https://en.wikipedia.org/wiki/Base32, n.d. (Funnet May 06, 2019).

[27] Technopedia, "What is java?." https://www.techopedia.com/definition/3927/java, n.d. (Funnet Mar 28, 2019).

[28] Apple Inc, "Swift. En kraftig og åpen kildekode som gjør at alle kan lage imponerende apper.." https://www.apple.com/no/swift/, 2019. (Funnet Mai 14, 2019).

[29] Wikipedia contributors, "Arduino." https://en.wikipedia.org/wiki/Arduino, n.d. (Funnet Mai 14, 2019).

[30] Wikipedia contributors, "C++." https://no.wikipedia.org/wiki/C%2B%2B, n.d. (Accessed on May 19 2019).

[31] Wikipedia contributors, "Python (programming language)." https://en.wikipedia.org/wiki/Python_(programming_language), n.d. (Funnet Mai 14, 2019).

[32] Will Hedgecock, "Github - what is jserialcomm?." http://fazecast.github.io/jSerialComm/, 2018. (Funnet Mai 14, 2019).

[33] GitHub contributors, "Github - a reference implementation of a json package in java.." https://github.com/stleary/JSON-java, 2015. (Funnet Mai 14, 2019).

[34] GitHub contributors, "Github - bridj." https://github.com/nativelibs4java/BridJ, n.d. (Funnet Mai 14, 2019).

[35] Quality Open Software, "Simple logging facade for java." https://www.slf4j.org/, n.d. (Funnet Mai 14, 2019).

[36] GitHub contributors, "Github - sarxos/webcam-capture." https://github.com/sarxos/webcam-capture, n.d. (Funnet Mai 14, 2019).

[37] The Apache software Foundation, "Preface." https://hc.apache.org/httpcomponents-client-4.5.x/tutorial/html/preface.html, n.d. (Funnet Mai 14, 2019).

[38] Oleg Kalnichevski, "Httpcore tutorial." http://hc.apache.org/httpcomponents-core-ga/tutorial/pdf/httpcore-tutorial.pdf, n.d. (Funnet Mai 13, 2019).

[39] Square Inc, "Okhttp." https://square.github.io/okhttp/, 2016. (Funnet Mai 13, 2019).

[40] GitHub contributors, "Github - a modern i/o api for java." https://github.com/square/okio, 2013. (Funnet Mai 13, 2019).

[41] The Apache software Foundation, "Codec." http://commons.apache.org/proper/commons-codec/, 2019. (Funnet Mai 15, 2019).

[42] OpenCV Team, "About opencv." https://opencv.org/about/, n.d. (Funnet Mai 13, 2019).

[43] Ruslan Shevchuk, "Wallet is a library to manage cards and passes.." https://github.com/rshevchuk/Wallet, n.d. (Funnet Mai 8, 2019).

[44] GitHub contributors, "A swift library for generating one time passwords (otp)." https://github.com/lachlanbell/SwiftOTP, n.d. (Funnet Mai 8, 2019).

[45] ®BioID, "Face recognition liveness detection software." https://www.bioid.com/, 2019. (Funnet Mar 28, 2019).

[46] Microsoft, "Cognitive services." https://azure.microsoft.com/nb-no/services/cognitive-services/, 2019. (Funnet Mai 15, 2019).

[47] Technopedia, "What is netbeans?." https://www.techopedia.com/definition/24735/netbeans. (Funnet Mar 28, 2019).

[48] ®Arduino, "Arduino - software." https://www.arduino.cc/en/Main/Software, 2019. (Funnet Mai 15, 2019).

[49] Apple Developer, "Xcode." https://developer.apple.com/xcode/, 2019. (Funnet Mai 15, 2019).

[50] Atlassian, "Sourcetree | free git gui for mac and windows." https://www.sourcetreeapp.com/, 2019. (Funnet Mar 28, 2019).

[51] Korbin Brown, "What is github, and what is it used for?." https://www.howtogeek.com/180167/htg-explains-what-is-github-and-what-do-geeks-use-it-for/, 2017. (Funnet Mar 28, 2019).

[52] Wikipedia contributors, "Asana (software)." https://en.wikipedia.org/wiki/Asana_(software), n.d.

[53] Moe Long, "Odroid xu4 vs raspberry pi 3 b+." https://www.electromaker.io/blog/article/odroid-xu4-vs-raspberry-pi-3-b, 2018. (Funnet Mai 14, 2019).

[54] Microsoft, "Vi presenterer nye surface go." https://www.microsoft.com/nb-no/p/surface-go/8V9DP4LNKNSZ/JJTQ?activetab=pivot:techspecstab, 2019. (Funnet Mai 14, 2019).

[55] "Mp motion sensor (amn1)." https://docs-emea.rs-online.com/webdocs/0030/0900766b80030a9d.pdf, 2001. (Funnet Mar 28, 2019).

[56] Stein Jarle Olsen, "Det er offisielt: Microsofts kinect er død." https://www.tek.no/artikler/microsoft-slutter-a-produsere-kinect/410763, 2017. (Funnet Mai 14, 2019).

[57] Microsoft, "Cognitive services apis reference." https://westus.dev.cognitive.microsoft.com/docs/services/563879b61984550e40cbbe8d/operations/563879b61984550f3039523c, 2017. (Funnet Mai 18 2019).

[58] GitHub contributors, "Two factor authentication java code." https://github.com/j256/two-factor-auth, n.d. (Funnet Mai 18, 2019).

[59] GitHub contributors, "webcam-capture · github." https://github.com/sarxos/webcam-capture/blob/master/webcam-capture/src/example/java/WebcamDiscoveryListenerExample.java, n.d. (Funnet Mai 18 2019).

Vedlegg

A Forprosjekt rapport

FORPROSJEKT - RAPPORT

FOR BACHELOROPPGAVE

NTNU
Kunnskap for en bedre verden

TITTEL:
Ansiktsgjenkjenning for låsesystemer

KANDIDATNUMMER(E):
476123 476124 476128 997307

DATO:	EMNEKODE:	EMNE:	DOKUMENT TILGANG:
	IE303612	**Bacheloroppgave**	- Åpen

STUDIUM:	ANT SIDER/VEDLEGG:	BIBL. NR:
AUTOMATISERINGSTEKNIKK, INGENIØR	/	- Ikke i bruk -

OPPDRAGSGIVER(E)/VEILEDER(E):
Oppdragsgiver: Avento AS Veileder: Girts Strazdins og Saleh Abdel-Afou Alaliyat

OPPGAVE/SAMMENDRAG:
Denne bacheloroppgaven er gitt av Avento, i samarbeid med Låseservice. Oppgaven går ut på å lage en tjeneste for ansiktsgjenkjenning koblet mot et låssystem. Dette skal kunne erstatte behovet for nøkkel/kodebrikke. Et kamera skal kunne ta bilde og gjenkjenne ansikt. Det skal lages et program som kan sammenligne bildene opp mot en database. Vi skal bruke Azure Cognitive services. Blir det gjenkjent en person som har tillatelse til bygget, så skal døren låses opp. **Avento har gitt ganske fritt spillerom til hvordan oppgaven skal implementeres. Det er mange måter å gjøre dette på. Vi skal også lage et brukergrensesnitt, muligens i form av en applikasjon.**

Postadresse
Høgskolen i Ålesund
N-6025 Ålesund
Norway

Besøksadresse
Larsgårdsvegen 2
Internett
www.hials.no

Telefon
70 16 12 00
Epostadresse
postmottak@hials.no

Telefax
70 16 13 00

Bankkonto
7694 05 00636
Foretaksregisteret
NO 971 572 140

Denne oppgaven er en eksamensbesvarelse utført av student(er) ved NTNU i Ålesund.

INNHOLD

1 INNLEDNING

Vi valgte denne oppgaven fordi vi syntes prosjektet er spennende og fremtidsrettet. Det var en prioritet for oss å kunne jobbe med en oppgave for en privat bedrift. Dette er en mulighet for oss til å vise oss frem og skape kontakter på det private markedet. Avento er en fremtidsrettet bedrift i stor vekst, og var derfor veldig attraktiv for oss. Vi skal lage en tjeneste for ansiktsgjenkjennelse koblet mot et låssystem. Målet er å lage en løsning som kan brukes både i bedrifter, husstander, barnehager osv.

2 BEGREPER

- API – Application Programming Interface
- GUI – Graphical User Interface
- SQL – Structured Query Language
- UML – Unified Modeling Language

3 PROSJEKTORGANISASJON

3.1 Prosjektgruppe

Studentnummer(e)
476123
476124
476128
997307

3.2 Styringsgruppe (veileder og kontaktperson oppdragsgiver)

Girts Strazdins og Saleh Abdel-Afou Alaliyat er våre veildere. Avento AS er vår oppdragsgiver med Anders beite som kontaktperson. Låseservice AS skal også vere med på prosjektet.

4 AVTALER

4.1 Avtale med oppdragsgiver

- Levere oppdatering hver andre uke.
- Et møte halvveis i prosjektet med gjennomgang av fremgang.
- Levere et videreutviklet prosjekt av et låsesystem som bruker ansiktsgjenkjenning som nøkkel.

4.2 Arbeidssted og ressurser

Anders Beite er vår kontaktperson ved Avento, de vil bistå med rådgiving om det er noe vi lurer på. Girts Strazdins blir vår veileder fra NTNU og skal hjelpe oss gjennom prosjektet.
Vi har avtalt å oppdatere Avento med framgangen vår annenhver uke.

Møte Onsdag 24.01.19 med Låseservice vil gi oss en oversikt over tilgjengelige ressurser

4.3 Gruppenormer – samarbeidsregler – holdninger

Vi har som utgangspunkt å møte på skolen 08.00 til 16.00 mandag til fredag som om at dette skulle være vanlige arbeidsdager. Dette trengs for å komme i mål med oppgaven. Vi har en oppgave som vi kan utvide og utvikle langt utover det som var satt som krav, og dette ønsker vi å gjøre.

Vi skal alle delta på felles avgjørelser og obligatoriske møter. Hvert møte finner vi ut sammen hvilke oppgaver hver enkelt skal jobbe med fram til neste møte.

Vi skal respektere hverandre. Jobbe profesjonelt med et mål om en god karakter. Vi skal ha en åpen kommunikasjon som gir gode arbeidsforhold og sunne diskusjoner.

5 PROSJEKTBESKRIVELSE

5.1 Problemstilling - målsetting - hensikt

Vår oppgave er å designe og implementere et låssystem med ansiktsgjenkjenning.

Vårt hovedmål er å lage et låssystem der en person kan gå opp til en dør, se på et kamera og låse opp denne døren.

Vi har også et mål om å implementere et brukergrensesnitt. Vi ønsker å lage en applikasjon, der en administrator har mulighet til å legge til og slette brukere som skal ha tilgang til å låse opp døren.

5.2 Krav til løsning eller prosjektresultat – spesifikasjon

Kravet er at vi skal kunne låse opp en dør ved hjelp av ansiktsgjenkjenning. Personer som ikke skal ha tilgang, skal ikke ha mulighet til å lure systemet. Vi har fått lite spesifikasjoner på hvordan dette skal gjøres. Vi stiller store krav til oss selv rundt sikkerheten til låssystemet. Vi ønsker ett systemet som skal være så trygt som mulig.

Avento har allerede laget en enkel løsning av systemet. De er interesserte i å se hvordan vi tenker å løse problemstillingen og har derfor gitt oss ganske fritt spillerom. Vår innstilling er å ta dette prosjektet så langt som mulig.

5.3 Planlagt framgangsmåte(r) for utviklingsarbeidet – metode(r)

Vi har laget en fremdriftsplan som vi forholder oss til.
Her legger vi inn ukes mål samt måneds mål og vi prøver å holde oss innenfor tidsrammene. Dette gjør at alle kan se hva som blir gjort og vi kan optimalisere prosessen. Dette gjør det også lettere å jobbe på tvers av oppgaver og gir en bedre helhetsløsning.

5.4 Informasjonsinnsamling – utført og planlagt

Vi har vært i kontakt med Avento som har informert oss om den foreløpige løsningen de har laget. Vi hadde lite kunnskap om problemstillingen før vi startet.

Vi bruker Avento, Internett og veiledere for å skaffe oss tilstrekkelig informasjon til å gjennomføre prosjektet på en god og sikker måte.

Vi er i kontakt med Låseservice og håper de har mulighet til å bidra med et låsemiljø, og eventuelle kildekoder som hører med.

5.5 Vurdering – analyse av risiko

Vi tror det er god mulighet for å klare å lage en enkel løsning av låssystemet. Det er mange steg på veien vi kan sette oss fast. Det er ennå ikke avklart hvilket låssystem vi får til disposisjon. Det er uvisst hvordan vi skal koble vårt program opp mot låssystemet. Det er også mange feller å gå i med tanke på sikkerheten. Vi ønsker å jobbe med. Vi har ikke mye erfaring med å lage applikasjon.

Dette er en tidskrevende oppgave og vi er avhengig av å jobbe mye. For at vi skal lykkes så er vi nok også avhengig av veiledning og hjelp fra veileder. Avento har også tilbudt seg til å hjelpe oss når vi står fast.

Mulige utfordringer som hindrer oss å lykkes med prosjektet:
-Hardware
- Komponentene fungerer som de skal.
- Odroiden er kraftig nok til å kjøre programmet raskt nok.

-Software
- Hvor lang tid det tar å godkjenne en person.

- Om vi får til å lese av dybdeverdiene og bruke pointcloud

5.6 Hovedaktiviteter i videre arbeid

Se vedlegg med fremdriftsplan.

5.7 Framdriftsplan – styring av prosjektet

Se vedlegg med fremdriftsplan.

5.7.1 Intern kontroll – evaluering

Vi jobber for det meste samtidig, og sammen. Vi har som utgangspunkt at vi møtes 0800 på morgenen og jobber til 1600 som en vanlig arbeidsdag. På starten av hver dag har vi en oppsummering på hva vi gjorde dagen før og legger en plan for hvilke mål vi har for kommende dag. Vi oppdaterer hverandre på hva vi har gjort slik at vi kan hjelpe hverandre når vi står fast, og at det skal bli lettere å bytte oppgaver underveis om det skulle være nødvendig.

5.8 Beslutninger – beslutningsprosess

Avgrensing og presisering av oppgaven kommer vi fram til sammen i gruppen. Vi brainstormer mye, og får nok ikke gjennomført alle idèene vi har.

6 DOKUMENTASJON

6.1 Rapporter og tekniske dokumenter

I vår sluttrapport så vil vi ha med teoretisk begrunnelse for alt vi har gjort.

7 PLANLAGTE MØTER OG RAPPORTER

7.1 Møter

7.1.1 Møter med styringsgruppen

7.1.2 Prosjektmøter

Annenhver uke kommer vi til å ha møte med veiledere.
Det vil også bli noen møter med Avento gjennom semesteret.

7.2 Periodiske rapporter

7.2.1 Framdriftsrapporter (inkl. milepæl)

Det blir rapportert tilbake til Avento annenhver uke om hvordan vi ligger an med prosjektet.

8 PLANLAGT AVVIKSBEHANDLING

Visst vi ikke skulle få til å lage låssystemet, så må vi snakke med Avento om de kan hjelpe oss. Det er mulig at vi har for høye ambisjoner til hva vi skal få til med denne oppgaven, og at vi kanskje må gjøre noen løsninger enklere enn hva vi har sett for oss.

9 UTSTYRSBEHOV/FORUTSETNINGER FOR GJENNOMFØRING

Nødvendig utstyr til oppgave:

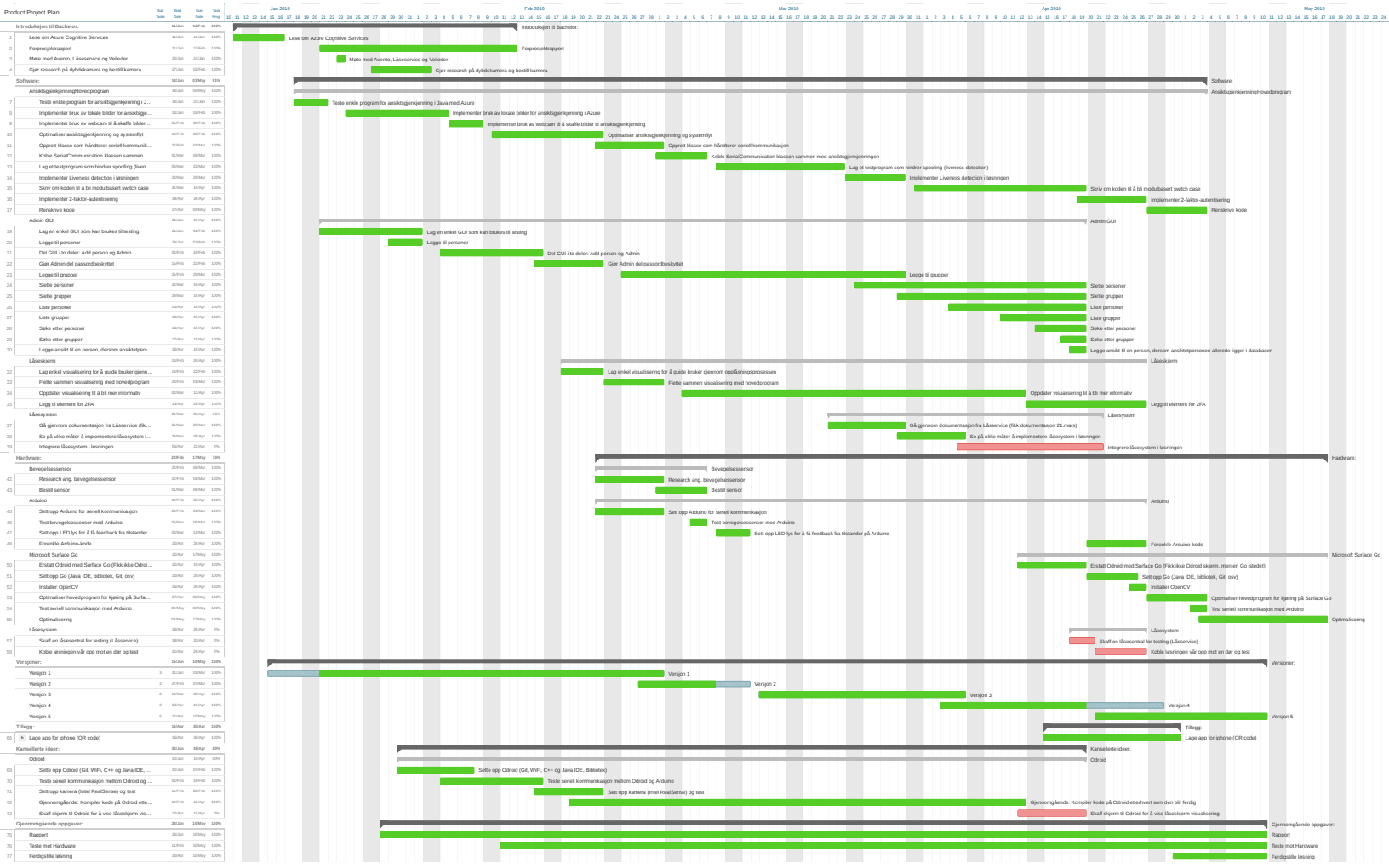
- Mikrokontroller
- Bevegelsessensor
- Kamera med mulighet til å måle dybde.
- Låsesystem og ett låsemiljø

10 REFERANSER

VEDLEGG

Vedlegg 1: Framgangsplan

B Gantt diagram - fremgangsplan


Product Project Plan
Jan 2019
Feb 2019
Mar 2019
Apr 2019
May 2019
Introduksjon til Bachelor:
Lese om Azure Cognitive Services
Forprosjektrapport
Møte med Aventio, Låseservice og Veileder
Gjør research på dybdekamera og bestill kamera
Software:
AnsiktsgjenkjenningHovedprogram
Teste enkle program for ansiktsgjenkjenning i Java med Azure
Implementer bruk av lokale bilder for ansiktsgjenkjenning i Azure
Implementer bruk av webcam til å skaffe bilder til ansiktsgjenkjenning
Optimaliser ansiktsgjenkjenning og systemflyt
Opprett klasse som håndterer seriell kommunikasjon
Koble SerialCommunication klassen sammen med ansiktsgjenkjenningen
Lag et testprogram som hindrer spoofing (liveness detection)
Implementer Liveness detection i løsningen
Skriv om koden til å bli modulbasert switch case
Implementer 2-faktor-autentisering
Rensk/rive kode
Admin GUI
Lag en enkel GUI som kan brukes til testing
Legge til personer
Del GUI i to deler: Add person og Admin
Gjør Admin del passordbeskyttet
Legge til grupper
Slette personer
Slette grupper
Liste personer
Liste grupper
Søke etter personer
Søke etter grupper
Legge ansikt til en person, dersom ansiktspersonen allerede ligger i databasen
Låseskjerm
Lag enkel visualisering for å guide bruker gjennom oppläsningsprosessen
Flette sammen visualisering med hovedprogram
Oppdater visualisering til å bli mer informativ
Legg til element for 2FA
Låsesystem
Gå gjennom dokumentasjon fra Låservice (fikk dokumentasjon 21.mars)
Se på ulike måter å implementere låsesystem i løsningen
Integrere låsesystem i løsningen
Hardware:
Bevegelsessensor
Research ang. bevegelsessensor
Bestill sensor
Arduino
Sett opp Arduino for seriell kommunikasjon
Test bevegelsessensor med Arduino
Sett opp LED lys for å få feedback fra tilstander på Arduino
Forenkle Arduino-kode
Microsoft Surface Go
Erstatt Odroid med Surface Go (Fikk ikke Odroid skjerm, men en Go istedet)
Sett opp Go (Java IDE, bibliotek, Git, osv)
Installer OpenCV
Optimaliser hovedprogram for kjøring på Surface Go
Test seriell kommunikasjon med Arduino
Optimalisering
Låsesystem
Skaff en låsesentral for testing (Låsservice)
Koble løsningen vår opp mot en dør og test
Versjoner:
Versjon 1
Versjon 2
Versjon 3
Versjon 4
Versjon 5
Tillegg:
Lage app for iphone (QR code)
Kansellerte ideer:
Odroid
Sette opp Odroid (Git, WiFi, C++ og Java IDE, Bibliotek)
Teste seriell kommunikasjon mellom Odroid og Arduino
Sett opp kamera (Intel RealSense) og test
Gjennomgående: Kompiler kode på Odroid etterhvert som den blir ferdig
Skaff skjerm til Odroid for å vise låseskjerm visualisering
Gjennomgående oppgaver:
Rapport
Teste mot Hardware
Ferdigstille løsning

C Java kildekode

Ansiktsgjenkjenning

```
 1 package FaceID;
 2
 3 import java.awt.FontFormatException;
 4 import java.awt.image.BufferedImage;
 5 import java.awt.image.DataBufferByte;
 6 import java.awt.image.WritableRaster;
 7 import java.io.IOException;
 8 import static java.lang.Thread.sleep;
 9 import java.text.ParseException;
10 import java.util.Scanner;
11 import java.util.logging.Level;
12 import java.util.logging.Logger;
13 import org.json.JSONException;
14 import static org.opencv.core.CvType.CV_8UC3;
15 import org.opencv.core.Mat;
16 import org.opencv.core.Rect;
17 import org.opencv.core.Scalar;
18 import static org.opencv.imgcodecs.Imgcodecs.imwrite;
19 import org.opencv.videoio.VideoCapture;
20 import com.google.zxing.*;
21 import com.google.zxing.client.j2se.BufferedImageLuminanceSource;
22 import com.google.zxing.common.HybridBinarizer;
23 import org.opencv.core.Size;
24 import org.opencv.imgproc.Imgproc;
25
26 /**
27  * Class that handles the logic in conjuction with the face recognition.
28  *
29  * @author Hans JÃ¸rgen Torp, Henrik Aarnes Kleppe, Adrian LÃ¸vlid, Joakim
30  * Fivelstad
31  */
32 public class FaceID {
33
34     String state = "";
35     VideoCapture camera;
36     FaceCrop faceCrop;
37     BufferedImage firstLiveImage;
38     BufferedImage secondLiveImage;
39     MotionDetection motion;
40     LivenessDetection livenessDetection;
41     Azure azure;
42     SerialCommunication sc;
43     GUI gui;
44     TwoFactorAuth twofa;
```

```
45      Thread t1;
46
47      private long motionSensor;
48      private long startTime;
49      private final long waitTime = 30000;
50      private long timeIsUp;
51      private boolean firstBioTry;
52      Scanner s;
53      String code;
54
55      Mat frame;
56      Mat threshMat;
57      Mat threshMatSub;
58      Mat secondImage;
59
60      /**
61       * Contructor that initializes the necessary fields.
62       *
63       * @throws FontFormatException
64       * @throws IOException
65       * @throws ParseException
66       */
67      public FaceID() throws FontFormatException, IOException, ParseException {
68          gui = new GUI();
69          gui.run();
70          twofa = new TwoFactorAuth();
71          t1 = new Thread(twofa);
72      }
73
74      /**
75       * Method that starts the run loop. This method contains the state-machine
76       * responsible for the logic in the program.
77       *
78       * @throws IOException
79       * @throws InterruptedException
80       */
81      public void run() throws IOException, InterruptedException {
82
83
System.load("C:\\incubating-netbeans-10.0-bin\\netbeans\\java\\opencv\\build\\java\\
x64\\opencv_java401.dll");
84          camera = new VideoCapture(0);
85
86          t1.start();
```

```
 87
 88         frame = new Mat();
 89         threshMat = new Mat();
 90         threshMatSub = new Mat();
 91         motion = new MotionDetection();
 92         azure = new Azure();
 93         faceCrop = new FaceCrop();
 94         livenessDetection = new LivenessDetection();
 95         firstBioTry = true;
 96         s = new Scanner(System.in);
 97         sc = new SerialCommunication("COM3");
 98
 99         Mat startMat1 = new Mat(2, 2, CV_8UC3, new Scalar(0, 0, 255));
100         firstLiveImage = MatToBufferedImage(startMat1);
101         secondLiveImage = firstLiveImage;
102         camera.read(frame);
103
104         // Starts up the system by sending a API call to BioID.
105         livenessDetection.sendToBioId(firstLiveImage, secondLiveImage);
106         secondLiveImage.flush();
107         firstLiveImage.flush();
108
109         while (true) {
110
111             switch (state) {
112
113                 /**
114                  * State that captures a frame, finds the faces and check if the
115                  * person is close enought for the state to be switched.
116                  */
117                 case "CaptureFace":
118
119                     double pixelThreshould;
120                     camera.read(frame);
121                     threshMat = frame.clone();
122
123                     Rect threshMatRect = faceCrop.getFaceCrop(threshMat, this);
124                     Rect threshImageSub = new Rect(threshMatRect.x, 0,
threshMatRect.width, threshMat.height());
125                     try {
126                         threshMatSub = threshMat.submat(threshImageSub);
127                     } catch (Exception e) {
128                         System.out.println(e.getMessage());
129                         break;
```

```
130                     }
131
132                     Size sz = new Size(threshMat.width() / 2, threshMat.height()
/ 2);
133                     try {
134                         Imgproc.resize(threshMat, threshMat, sz);
135                     } catch (Exception e) {
136                         System.out.println(e.getMessage());
137                         break;
138                     }
139
140                     int threshMatArea = threshMatSub.width() *
threshMatSub.height();
141
142                     pixelThreshould = frame.height() * frame.width() / 5;
143                     if (threshMatArea > pixelThreshould) {
144
145                         if (firstBioTry == true) {
146                             firstBioTry();
147                             camera.read(frame);
148                             threshMat = frame;
149                             firstLiveImage = MatToBufferedImage(threshMat);
150                             imwrite("bilde.jpg", threshMat);
151                         } else {
152
153                             firstLiveImage = MatToBufferedImage(threshMat);
154                             state = "liveness";
155                             imwrite("bilde.jpg", threshMat);
156                         }
157
158                     } else {
159                         state = "CaptureFace";
160                         timeToReset();
161                         break;
162                     }
163
164                 /**
165                  * State that checks if the person standing in front of the
166                  * system is a real person, using BioID.
167                  */
168                 case "liveness":
169
170                     boolean liveness = motion.checkForMotion(this);
171                     boolean faceLiveNess = false;
```

```
172
173                     if (liveness == true) {
174
175                         camera.read(frame);
176
177                         Mat newMat = frame.clone();
178                         secondImage = newMat;
179
180                         sz = new Size(secondImage.width() / 2,
secondImage.height() / 2);
181
182                         try {
183                             Imgproc.resize(secondImage, secondImage, sz);
184                         } catch (Exception e) {
185                             break;
186                         }
187
188                         imwrite("bilde1.jpg", frame);
189                         secondLiveImage = MatToBufferedImage(frame);
190
191                         try {
192                             faceLiveNess =
livenessDetection.sendToBioId(firstLiveImage, secondLiveImage);
193                             firstLiveImage.flush();
194                             secondLiveImage.flush();
195                         } catch (JSONException | IOException ex) {
196
Logger.getLogger(FaceID.class.getName()).log(Level.SEVERE, null, ex);
197                         }
198
199                         if (faceLiveNess == true) {
200                             state = "SendToAzure";
201                             break;
202
203                         } else {
204                             state = "CaptureFace";
205                             timeToReset();
206                             break;
207                         }
208                     }
209                     state = "CaptureFace";
210                     timeToReset();
211                     break;
212
```

```
213                 /**
214                  * State that send a picture to Azure Face API.
215                  * Checks if the person in the pictures should have access.
216                  */
217                 case "SendToAzure":
218 
219                     azure.detectFace();
220                     boolean identify = azure.identify();
221                     secondImage = null;
222 
223                     if (identify != true) {
224                         gui.card2();
225                         locked();
226                         state = "";
227                         break;
228                     } else {
229                         azure.getName();
230                         state = "2fa";
231                         break;
232                     }
233 
234                 /**
235                  * State that checks the validation of the QRF-password that
236                  * is shown to the system.
237                  */
238                 case "2fa":
239                     gui.smile();
240                     code = "0000";
241                     camera.read(frame);
242                     BufferedImage image = MatToBufferedImage(frame);
243                     LuminanceSource source = new
BufferedImageLuminanceSource(image);
244 
245                     BinaryBitmap bitmap = new BinaryBitmap(new
HybridBinarizer(source));
246 
247                     try {
248                         Result result = new MultiFormatReader().decode(bitmap);
249                         code = result.getText();
250 
251                     } catch (Exception e) {
252                         System.out.println("No qr code found..");
253                         System.out.println(e.getMessage());
254                     }
```

```
255
256                     if (code.equals(twofa.code)) {
257
258                         state = "Unlocked";
259                         gui.card4();
260                         azure.getName();
261                         gui.name = azure.personName;
262                         reset();
263                         break;
264                     }
265
266                     timeToReset();
267                     state = "2fa";
268                     break;
269
270                 /**
271                  * State representing a unlocked system.
272                  */
273                 case "Unlocked":
274                     state = "Reset";
275                     break;
276
277                 /**
278                  * State that takes care of reseting the system.
279                  */
280                 case "Reset":
281
282                     state = "";
283                     break;
284
285                 /**
286                  * The default state, system reads the in signal from the
arduino.
287                  * When the motion sensor is triggered, the systemn start to
run.
288                  */
289                 default:
290                     // Read signal from motionSensor.
291                     motionSensor = sc.readSensor();
292                     if (motionSensor == 1) {
293                         state = "CaptureFace";
294                         firstBioTry = true;
295                         setTimer();
296                         gui.comeCloser();
```

```
297                         break;
298                     } else {
299                         gui.card1();
300                         break;
301                     }
302             }
303 
304         }
305 
306     }
307 
308     /**
309      * Method that make a BufferedImage from a OpenCV Mat.
310      *
311      * @param frame, OpenCV Mat representing an image.
312      * @return a BuffereImage
313      */
314     public BufferedImage MatToBufferedImage(Mat frame) {
315         //Mat() to BufferedImage
316         int type = 0;
317         if (frame.channels() == 1) {
318             type = BufferedImage.TYPE_BYTE_GRAY;
319         } else if (frame.channels() == 3) {
320             type = BufferedImage.TYPE_3BYTE_BGR;
321         }
322         BufferedImage image0 = new BufferedImage(frame.width(), frame.height(),
type);
323         WritableRaster raster = image0.getRaster();
324         DataBufferByte dataBuffer = (DataBufferByte) raster.getDataBuffer();
325         byte[] data = dataBuffer.getData();
326         frame.get(0, 0, data);
327 
328         return image0;
329     }
330 
331     /**
332      * Method for resetting a timer.
333      */
334     public void setTimer() {
335         startTime = System.currentTimeMillis();
336         timeIsUp = startTime + waitTime;
337     }
338 
339     /**
```

```
340      * Method that send the system in locked mode for 6 seconds.
341      *
342      * @throws InterruptedException
343      */
344     public void locked() throws InterruptedException {
345         sleep(4000);
346         gui.card6();
347         sleep(2000);
348     }
349 
350     /**
351      * Method that checks if the passed time is bigger than the time that is set
352      * for the timer.
353      *
354      * @return
355      */
356     public boolean timeToReset() {
357         if (System.currentTimeMillis() > timeIsUp) {
358             state = "";
359             motionSensor = 0;
360             return true;
361         } else {
362             return false;
363         }
364     }
365 
366     /**
367      * Method that grabs a frame from the camera.
368      *
369      * @return the frame from the camera.
370      */
371     public Mat getOneFrame() {
372         camera.read(frame);
373         return frame;
374     }
375 
376     /**
377      * Method that sleeps for 800 milliseconds.
378      *
379      * @throws InterruptedException
380      */
381     public void firstBioTry() throws InterruptedException {
382         sleep(800);
383     }
```

```
384
385     /**
386      * Method that resets the system, changes Illustrasion cards and sleeps for
387      * 8,5 seconds.
388      *
389      * @return
390      * @throws InterruptedException
391      */
392     public boolean reset() throws InterruptedException {
393         sleep(2500);
394         gui.card7();
395         sleep(1000);
396         gui.card3();
397         sleep(5000);
398         return true;
399     }
400
401 }
```

```
  1 package FaceID;
  2
  3 import java.io.File;
  4 import java.io.IOException;
  5 import java.net.URI;
  6 import java.net.URISyntaxException;
  7 import java.util.ArrayList;
  8 import java.util.HashMap;
  9 import java.util.Iterator;
 10 import org.apache.http.HttpEntity;
 11 import org.apache.http.HttpResponse;
 12 import org.apache.http.ParseException;
 13 import org.apache.http.client.HttpClient;
 14 import org.apache.http.client.methods.HttpGet;
 15 import org.apache.http.client.methods.HttpPost;
 16 import org.apache.http.client.utils.URIBuilder;
 17 import org.apache.http.impl.client.HttpClientBuilder;
 18 import org.apache.http.util.EntityUtils;
 19
 20 import org.json.JSONArray;
 21 import org.json.JSONObject;
 22 import org.apache.http.entity.FileEntity;
 23 import org.apache.http.entity.StringEntity;
 24 import org.apache.http.impl.client.HttpClients;
 25 import org.json.JSONException;
 26
 27 /**
 28  *
 29  * This class contains methods to communicate with Azure Cognitive Services.
 30  * @author Hans JÃ¸rgen Torp, Henrik Aarnes Kleppe, Adrian LÃ¸vlid, Joakim
Fivelstad
 31  */
 32 public class Azure {
 33
 34     // Final variables that holds information we need to communicate and use the
service.
 35     private static final String subscriptionKey =
"51bd92792c08426a9c08448790006061";
 36     private static final String uriBase =
 37
"https://northeurope.api.cognitive.microsoft.com/face/v1.0/detect";
 38     private static final String faceAttributes
 39                                 = "headPose";
 40
```

```
41      //Variables that stores different types of information used in the methods.
42      String faceId;
43      String compareString;
44      double confidence;
45      String groupId;
46      public String personName;
47      String userData;
48      String id;
49      public volatile ArrayList<String> groupArray = new ArrayList<>();
50      public volatile HashMap groups = new HashMap<>();
51
52      /**
53       * Method that takes a picture from a saved file, and sends it to the
FaceAPi.
54       */
55      public void detectFace() {
56
57          HttpClient httpclient = HttpClientBuilder.create().build();
58
59          try {
60              URIBuilder builder = new URIBuilder(uriBase);
61
62              // Request parameters. All of them are optional.
63              builder.setParameter("returnFaceId", "true");
64              builder.setParameter("returnFaceLandmarks", "false");
65              builder.setParameter("returnFaceAttributes", faceAttributes);
66
67              // Prepare the URI for the REST API call.
68              URI uri = builder.build();
69              HttpPost request = new HttpPost(uri);
70
71              // Request headers.
72              request.setHeader("Content-Type", "application/octet-stream");
73              request.setHeader("Ocp-Apim-Subscription-Key", subscriptionKey);
74              File file = new
File("C:\\Users\\NTNT_LÃ…SESYSTEM\\Documents\\NetBeansProjects\\Bachelor-2019\\FaceA
PI_TEST\\bilde1.jpg");
75              HttpEntity face1 = new FileEntity(file);
76              request.setEntity(face1);
77
78              // Execute the REST API call and get the response entity.
79              HttpResponse response = httpclient.execute(request);
80              HttpEntity entity = response.getEntity();
81              if (entity != null) {
```

```
 82
 83                 String str = EntityUtils.toString(entity).trim();
 84                 //System.out.println(str);
 85
 86                 JSONArray jsonArray = new JSONArray(str);
 87                 try {
 88                     System.out.println("The detectFace array has " +
jsonArray.length() + " items.");
 89
 90                     for (int i = 0; i < jsonArray.length(); ++i) {
 91                         JSONObject faces = jsonArray.getJSONObject(i);
 92                         faceId = faces.getString("faceId");
 93                         System.out.println("FaceID: " + faceId);
 94                     }
 95                 } catch (JSONException e) {
 96                     System.out.println(e.getMessage());
 97                 }
 98
 99                 //compareString = "{\"personGroupId\":\"" + group + "\",
\"faceIds\":[\"" + faceId + "\"]}";
100             }
101         } catch (IOException | URISyntaxException | ParseException |
JSONException e) {
102             // Display error message.
103             System.out.println(e.getMessage());
104         }
105     }
106
107     /**
108      * Method that checks if the person that recently have been detected
109      * in the method detectface() is valid, and can be found in the database.
110      * @return a boolean value representing the response from Azure.
111      * True represent that a person is valid. False represent the opposite.
112      */
113     public boolean identify() {
114         confidence = 0;
115         boolean result = false;
116
117         JSONObject objectToSend = new JSONObject();
118         JSONArray faceIds = new JSONArray();
119         faceIds.put(faceId);
120         objectToSend.put("faceIds", faceIds);
121         listPersonGroup();
122         Iterator it = groupArray.iterator();
```

```
123         while (it.hasNext()) {
124             String groupName = it.next().toString();
125             System.out.println("GROUPNAME: " + groupName);
126             groupId = groups.get(groupName).toString();
127             objectToSend.put("personGroupId", groupId);
128             compareString = objectToSend.toString();
129 
130             HttpClient httpclient = HttpClientBuilder.create().build();
131 
132             try {
133                 URIBuilder builder = new URIBuilder(
134 
"https://northeurope.api.cognitive.microsoft.com/face/v1.0/identify");
135 
136                 URI uri = builder.build();
137                 HttpPost request = new HttpPost(uri);
138 
139                 request.setHeader("Content-Type", "application/json");
140                 request.setHeader("Ocp-Apim-Subscription-Key", subscriptionKey);
141 
142                 StringEntity reqEntity = new StringEntity(compareString);
143 
144                 request.setEntity(reqEntity);
145 
146                 // Execute the REST API call and get the response entity.
147                 HttpResponse response = httpclient.execute(request);
148                 HttpEntity entity = response.getEntity();
149 
150                 if (entity != null) {
151 
152                     String str = EntityUtils.toString(entity).trim();
153                     JSONArray jsonArray = new JSONArray(str);
154 
155                     try {
156                         System.out.println("The identify array has " +
jsonArray.length() + " items.");
157                         for (int i = 0; i < jsonArray.length(); ++i) {
158                             JSONObject faces = jsonArray.getJSONObject(i);
159                             String faceid = faces.getString("faceId");
160                             System.out.println("FaceID: " + faceid);
161 
162                             JSONArray candidates =
faces.getJSONArray("candidates");
163                             System.out.println("Number of candidates found: " +
```

```
candidates.length());
                            for (int j = 0; j < candidates.length(); ++j) {
                                JSONObject candidateObjects =
candidates.getJSONObject(j);
                                id = candidateObjects.getString("personId");
                                confidence =
candidateObjects.getDouble("confidence");
                                System.out.println("PersonID: " + id);
                                System.out.println("Confidence: " + confidence);
                            }

                        }

                        if (confidence > 0.5) {
                            result = true;
                            break;
                        } else {
                            result = false;

                        }
                    } catch (JSONException e) {
                        System.out.println(e.getMessage());
                    }
                }
            } catch (Exception e) {

            }
        }
        return result;
    }

    /**
     * Method to retrive a name of a person stored in Azure.
     */
    public void getName() {
        personName = "";

        HttpClient httpclient = HttpClients.createDefault();

        try {
            URIBuilder builder = new URIBuilder(
                    "https://northeurope.api.cognitive.microsoft.com/face/v1.0/"
                            + "persongroups/" + groupId + "/persons/" + id);

```

```
205             builder.setParameter(groupId, id);
206 
207             URI uri = builder.build();
208             HttpGet request = new HttpGet(uri);
209 
210             request.setHeader("Ocp-Apim-Subscription-Key", subscriptionKey);
211 
212 
213             HttpResponse response = httpclient.execute(request);
214             HttpEntity entity = response.getEntity();
215 
216             if (entity != null) {
217 
218                 String str = EntityUtils.toString(entity);
219                 if(str.contains("error")){
220                     JSONObject jsonObject = new JSONObject(str);
221                     JSONObject obj = jsonObject.getJSONObject("error");
222                     String errorCode = obj.getString("code");
223                     String errorMessage = obj.getString("message");
224                     System.out.println("Something went wrong..." +
225                             "\n ERRORCODE: " + errorCode + "\n Message: "
226                             + errorMessage);
227                 }
228                 try {
229                     JSONObject person = new JSONObject(str);
230                     if (person.has("name")) {
231                         personName = person.getString("name");
232                         System.out.println("PersonName: " + personName);
233                     }
234                     if (person.has("userData")) {
235                         Object userdataObject = person.get("userData");
236                         userData = userdataObject.toString();
237                         System.out.println("User Data: " + userData);
238                     }
239                 } catch (JSONException e) {
240                     System.out.println(e.getMessage());
241                 }
242             }
243         } catch (Exception e) {
244             System.out.println(e.getMessage());
245         }
246     }
247 
248     /**
```

```
249      * This method populates an ArrayList of groupNames and a HashMap with
250      * groupNames and their groupId. If the Map or ArrayList is not empty, we
251      * make sure to clear them before moving on.
252      */
253     public void listPersonGroup() {
254         if (!groupArray.isEmpty()) {
255             groupArray.clear();
256         }
257         if (!groups.isEmpty()) {
258             groups.clear();
259         }
260 
261         HttpClient httpclient = HttpClientBuilder.create().build();
262 
263         try {
264             URIBuilder builder = new
URIBuilder("https://northeurope.api.cognitive.microsoft.com/face/v1.0/persongroups")
;
265 
266             builder.setParameter("start", "0");
267             //builder.setParameter("top", "1000");
268 
269             URI uri = builder.build();
270             HttpGet request = new HttpGet(uri);
271 
272             // Request headers.
273             request.setHeader("Ocp-Apim-Subscription-Key", subscriptionKey);
274 
275             // Execute the REST API call and get the response entity.
276             HttpResponse response = httpclient.execute(request);
277             HttpEntity entity = response.getEntity();
278             if (entity != null) {
279                 String jsonString = EntityUtils.toString(entity).trim();
280                 //System.out.println("REST RESPONSE: "+ jsonString);
281 
282                 try {
283                     JSONArray jsonArray = new JSONArray(jsonString);
284                     System.out.println("Person Group array has " +
jsonArray.length() + " items");
285                     // Parse the JSON response and retrieve the information we
286                     // we need. I.e. persongroupname and id.
287                     for (int i = 0; i < jsonArray.length(); ++i) {
288                         JSONObject group = jsonArray.getJSONObject(i);
289                         String personGroupId = group.getString("personGroupId");
```

```
290                         String personGroupName = group.getString("name");
291                         // Add name and id to the HashMap
292                         groups.put(personGroupName, personGroupId);
293                         //String arrayString = "PersonGroupId: " + personGroupId
+ ", name: " + personGroupName;
294                         //System.out.println("ArrayString: " + arrayString);
295                         // Add names to the ArrayList.
296                         groupArray.add(personGroupName);
297                     }
298                 } catch (JSONException e) {
299                     System.out.println(e.getMessage());
300                 }
301             }
302         } catch (Exception e) {
303             // Display error message.
304             System.out.println(e.getMessage());
305         }
306     }
307 }
```

```
 1 package FaceID;
 2
 3 import org.opencv.core.Mat;
 4 import org.opencv.core.MatOfRect;
 5 import org.opencv.core.Rect;
 6 import org.opencv.objdetect.CascadeClassifier;
 7
 8 /**
 9  * Class that contains a method for cropping the image based on where the face
10  * is.
11  *
12  * @author @author Hans JÃ¸rgen Torp, Henrik Aarnes Kleppe, Adrian LÃ¸vlid,
Joakim
13  * Fivelstad
14  */
15 public class FaceCrop {
16
17     CascadeClassifier faceDetector;
18     MatOfRect faceDetections;
19     Rect faceCrop;
20     Rect biggestRect;
21     Rect newFaceCrop;
22     Mat image_roi;
23
24     /**
25      * Method to retrive a rectangle around the biggest face found in the frame.
26      *
27      * @param frameFromCamera, frame captured from the camera.
28      * @param faceId, an instance of the Class FaceId.
29      * @return a rectangle thet represnt where the biggest face in the picture
30      * is located.
31      */
32     public Rect getFaceCrop(Mat frameFromCamera, FaceID faceId) {
33
34         faceDetector = new
CascadeClassifier("C:\\incubating-netbeans-10.0-bin\\netbeans\\java\\opencv\\build\\
etc\\haarcascades\\haarcascade_frontalface_alt2.xml");
35         faceDetections = new MatOfRect();
36         biggestRect = new Rect(0, 0, 0, 0);
37
38         faceDetector.detectMultiScale(frameFromCamera, faceDetections);
39
40         if (faceDetections.toArray().length > 0) {
41
```

```
42              // Iterates over every pixel in the array and finds the rectangle for
the face detection.
43              for (Rect rect : faceDetections.toArray()) {
44                  faceCrop = new Rect(rect.x, rect.y, rect.width, rect.height);
45                  if (faceCrop.area() > biggestRect.area()) {
46                      biggestRect = faceCrop;
47                  }
48              }
49
50              newFaceCrop = rotateRect(biggestRect, 160, 150);
51
52              if (0 >= newFaceCrop.y) {
53                  newFaceCrop.y = 0;
54              } else if (frameFromCamera.height() <= newFaceCrop.y +
newFaceCrop.height) {
55                  newFaceCrop.y = frameFromCamera.height() - newFaceCrop.height;
56              }
57
58              if (0 >= newFaceCrop.x) {
59                  newFaceCrop.x = 0;
60              } else if (frameFromCamera.width() <= newFaceCrop.x +
newFaceCrop.width) {
61                  newFaceCrop.x = frameFromCamera.width() - newFaceCrop.width;
62              }
63
64              return newFaceCrop;
65          }
66
67          faceId.timeToReset();
68          return biggestRect;
69
70      }
71
72      /**
73       * Method to crop the image, it takes in the rect, and ads height and width
74       * to the rectangle.
75       *
76       * @param rect, rectangle that represent the rectangle around the face.
77       * @param heightPercentage, the size we want to expand the rectangle in
78       * height.
79       * @param widthPercetange, the size we want to expand the rectangle in
80       * width.
81       * @return a rectangle that is resized with the incoming parameters.
82       */
```

```
83     private static Rect rotateRect(Rect rect, int heightPercentage, int
widthPercetange) {
84         int rwidth = rect.width;
85         int rheight = rect.height;
86 
87         rect.width = Math.round((rect.width * widthPercetange) / 100.0f);
88         rect.height = Math.round((rect.height * heightPercentage) / 100.0f);
89         rect.x += (rwidth - rect.width) / 2;
90         rect.y += (rheight - rect.height) / 2;
91 
92         return rect;
93     }
94 
95 }
```

```
 1 package FaceID;
 2
 3 import ODROID.FaceDetect;
 4 import java.awt.*;
 5 import java.awt.image.BufferedImage;
 6 import java.io.IOException;
 7 import java.io.InputStream;
 8 import java.text.ParseException;
 9 import javax.swing.*;
10
11 /**
12  * Class that represnt the illustrasion screen that takes the user thrue the
13  * process.
14  *
15  * @author @author Hans JÃ¸rgen Torp, Henrik Aarnes Kleppe, Adrian LÃ¸vlid,
Joakim
16  * Fivelstad
17  */
18 public class GUI {
19
20     final CardLayout cardLayout = new CardLayout();
21     final JPanel cardPanel = new JPanel(cardLayout);
22     FaceDetect face;
23     JLabel welcomeText;
24     JLabel welcomeText2;
25     JLabel comeCloserText;
26     JLabel smileText;
27     JFrame passwordBox;
28     String numberString = "";
29
30     TwoFactorAuth tf;
31
32     InputStream is = GUI.class.getResourceAsStream("Gobold-Bold.ttf");
33     Font font;
34     public volatile String name;
35
36     /**
37      *
38      * @throws FontFormatException
39      * @throws IOException
40      */
41     public GUI() throws FontFormatException, IOException {
42         this.font = Font.createFont(Font.TRUETYPE_FONT, is);
43     }
```

```
 44
 45     /**
 46      * Starts the GUI.
 47      * @throws ParseException
 48      */
 49     public void run() throws ParseException {
 50
 51         BufferedImage cursorImg = new BufferedImage(16, 16,
BufferedImage.TYPE_INT_ARGB);
 52         Cursor blankCursor =
Toolkit.getDefaultToolkit().createCustomCursor(cursorImg, new Point(0, 0),
"blankcursor");
 53
 54         JFrame window = new JFrame();
 55         window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
 56         window.setResizable(false);
 57         window.getContentPane().setCursor(blankCursor);
 58
 59         passwordBox = new JFrame();
 60         passwordBox.setSize(500, 650);
 61         passwordBox.setResizable(false);
 62         passwordBox.setUndecorated(true);
 63         passwordBox.setLayout(new BorderLayout());
 64         passwordBox.getContentPane().setCursor(blankCursor);
 65
 66         Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();
 67         passwordBox.setLocation(dim.width / 2 - passwordBox.getSize().width / 2,
dim.height / 2 - passwordBox.getSize().height / 2);
 68
 69         window.setExtendedState(JFrame.MAXIMIZED_BOTH);
 70         window.setUndecorated(true);
 71         window.setVisible(true);
 72
 73         ImageIcon image1 = new
ImageIcon("C:\\Users\\NTNT_LÃ…SESYSTEM\\Documents\\NetBeansProjects\\Bachelor-2019\\
FACEIDGUI\\avento999.gif");
 74         JLabel imagelabel1 = new JLabel(image1);
 75
 76         ImageIcon image2 = new
ImageIcon("C:\\Users\\NTNT_LÃ…SESYSTEM\\Documents\\NetBeansProjects\\Bachelor-2019\\
FACEIDGUI\\lockeds_2.gif");
 77         JLabel imagelabel2 = new JLabel(image2);
 78
 79         ImageIcon image3 = new
```

```
ImageIcon("C:\\Users\\NTNT_LÃ…SESYSTEM\\Documents\\NetBeansProjects\\Bachelor-2019\\
FACEIDGUI\\brukdenne1.png");
 80         JLabel imagelabel3 = new JLabel(image3);
 81
 82         ImageIcon image4 = new
ImageIcon("C:\\Users\\NTNT_LÃ…SESYSTEM\\Documents\\NetBeansProjects\\Bachelor-2019\\
FACEIDGUI\\unlocked99.gif");
 83         JLabel imagelabel4 = new JLabel(image4);
 84
 85         ImageIcon image5 = new
ImageIcon("C:\\Users\\NTNT_LÃ…SESYSTEM\\Documents\\NetBeansProjects\\Bachelor-2019\\
FACEIDGUI\\lockeds.png");
 86         JLabel imagelabel5 = new JLabel(image5);
 87
 88         ImageIcon image6 = new ImageIcon("");
 89         JLabel robot = new JLabel(image6);
 90
 91         ImageIcon image7 = new
ImageIcon("C:\\Users\\NTNT_LÃ…SESYSTEM\\Documents\\NetBeansProjects\\Bachelor-2019\\
FACEIDGUI\\unlocked2_2.gif");
 92         JLabel unlockedName = new JLabel(image7);
 93
 94         ImageIcon image8 = new
ImageIcon("C:\\Users\\NTNT_LÃ…SESYSTEM\\Documents\\NetBeansProjects\\Bachelor-2019\\
FACEIDGUI\\unlockedName.png");
 95         JLabel unlockedName2 = new JLabel();
 96
 97         ImageIcon comeCloser = new
ImageIcon("C:\\Users\\NTNT_LÃ…SESYSTEM\\Documents\\NetBeansProjects\\Bachelor-2019\\
FACEIDGUI\\liveness2.gif");
 98         JLabel comeCloserLabel = new JLabel(comeCloser);
 99
100         ImageIcon smile = new
ImageIcon("C:\\Users\\NTNT_LÃ…SESYSTEM\\Documents\\NetBeansProjects\\Bachelor-2019\\
FACEIDGUI\\qrUnlock.gif");
101         JLabel smileLabel = new JLabel(smile);
102
103         Font sizedFont = font.deriveFont(25f);
104
105         JPanel panel1 = new JPanel(new GridLayout(4, 3));
106         panel1.setBackground(new Color(1, 49, 63));
107
108
109
```

```
110         JPanel welcomeCard = new JPanel();
111         welcomeCard.setLocation(0, 0);
112         welcomeCard.setBackground(new Color(1, 49, 63));
113         welcomeCard.add(imagelabel3);
114         welcomeText = new JLabel();
115         welcomeText.setBackground(new Color(1, 49, 63));
116 
117         // create two dummy panels (the "cards") to show
118         JPanel card1 = new JPanel();
119         card1.setBackground(new Color(1, 49, 63));
120         card1.add(imagelabel1);
121 
122         //card1.add(jlabel1);
123         JPanel card2 = new JPanel();
124         card2.setBackground(new Color(1, 49, 63));
125         card2.add(imagelabel2);
126 
127         JPanel card3 = new JPanel();
128         card3.setBackground(new Color(1, 49, 63));
129         card3.add(welcomeCard, BorderLayout.SOUTH);
130         card3.add(welcomeText, BorderLayout.SOUTH);
131 
132         JPanel card6 = new JPanel();
133         card6.setBackground(new Color(1, 49, 63));
134         card6.add(imagelabel5);
135 
136         JPanel card4 = new JPanel();
137         card4.setBackground(new Color(1, 49, 63));
138         card4.add(imagelabel4);
139 
140         JPanel robotPanel = new JPanel();
141         robotPanel.setBackground(Color.white);
142         robotPanel.add(robot);
143         robotPanel.setPreferredSize(new Dimension(640, 480));
144 
145         JPanel card5 = new JPanel();
146         card5.setBackground(Color.white);
147 
148         card5.add(robotPanel);
149 
150         JPanel card7 = new JPanel();
151         card7.setBackground(new Color(1, 49, 63));
152         card7.add(unlockedName);
153 
```

```
154         comeCloserText = new JLabel();
155         comeCloserText.setText("                Kom nÃ¦rmere!
");
156         Font sizedFont1 = font.deriveFont(50f);
157         comeCloserText.setFont(sizedFont1);
158         comeCloserText.setForeground(new Color(1, 49, 63));
159         JPanel comeCloserCard = new JPanel();
160         comeCloserCard.setBackground(new Color(1, 49, 63));
161         comeCloserCard.add(comeCloserLabel, BorderLayout.NORTH);
162         comeCloserCard.add(comeCloserText, BorderLayout.CENTER);
163 
164         smileText = new JLabel();
165         smileText.setText("                        Smil!
");
166         sizedFont1 = font.deriveFont(50f);
167         smileText.setFont(sizedFont1);
168         smileText.setForeground(new Color(1, 49, 63));
169 
170         JPanel smileCard = new JPanel();
171         smileCard.add(smileLabel, BorderLayout.NORTH);
172         smileCard.add(smileText, BorderLayout.SOUTH);
173         smileCard.setBackground(new Color(1, 49, 63));
174 
175         cardPanel.add(card1, "RedCard");
176         cardPanel.add(card2, "BlueCard");
177         cardPanel.add(card3, "OrangeCard");
178         cardPanel.add(card4, "CyanCard");
179         cardPanel.add(card5, "webcam");
180         cardPanel.add(card6, "lockeds");
181         cardPanel.add(card7, "yes");
182         cardPanel.add(comeCloserCard, "comeCloser");
183         cardPanel.add(smileCard, "smile");
184         cardPanel.setBackground(Color.white);
185 
186         // add card & button panels to the main window
187         window.add(cardPanel, BorderLayout.CENTER);
188         window.setVisible(true);
189 
190     }
191 
192     /**
193      * Method for showming a given card from the cardPanel.
194      */
195     public void card1() {
```

```
196         cardLayout.show(cardPanel, "RedCard");
197     }
198 
199     /**
200      * Method for showming a given card from the cardPanel.
201      */
202     public void card2() {
203         cardLayout.show(cardPanel, "BlueCard");
204     }
205 
206     /**
207      * Method for showming a given card from the cardPanel.
208      */
209     public void card3() {
210         welcomeText.setText("          Welcome  " + name + "             ");
211         //welcomeText.setFont(new Font("Courier New", Font.ITALIC, 50));
212         Font sizedFont = font.deriveFont(50f);
213         welcomeText.setFont(sizedFont);
214         welcomeText.setForeground(Color.white);
215         welcomeText.setLocation(0, 0);
216         welcomeText.setSize(800, 250);
217 
218         cardLayout.show(cardPanel, "OrangeCard");
219     }
220 
221     /**
222      * Method for showming a given card from the cardPanel.
223      */
224     public void card4() {
225         cardLayout.show(cardPanel, "CyanCard");
226     }
227 
228     /**
229      * Method for showming a given card from the cardPanel.
230      */
231     public void card5() {
232         cardLayout.show(cardPanel, "webcam");
233     }
234 
235     /**
236      * Method for showming a given card from the cardPanel.
237      */
238     public void card6() {
239         cardLayout.show(cardPanel, "lockeds");
```

```
240     }
241
242     /**
243      * Method for showming a given card from the cardPanel.
244      */
245     public void card7() {
246         cardLayout.show(cardPanel, "yes");
247     }
248
249     /**
250      * Method for showming a given card from the cardPanel.
251      */
252     public void comeCloser() {
253         cardLayout.show(cardPanel, "comeCloser");
254     }
255
256     /**
257      * Method for showming a given card from the cardPanel.
258      */
259     public void smile() {
260         cardLayout.show(cardPanel, "smile");
261     }
262
263 }
```

```
 1 package FaceID;
 2
 3 import java.awt.image.BufferedImage;
 4 import java.io.ByteArrayOutputStream;
 5 import java.io.IOException;
 6 import java.util.Base64;
 7 import javax.imageio.ImageIO;
 8 import okhttp3.Credentials;
 9 import okhttp3.MediaType;
10 import okhttp3.OkHttpClient;
11 import okhttp3.Request;
12 import okhttp3.RequestBody;
13 import okhttp3.Response;
14 import org.json.JSONException;
15 import org.json.JSONObject;
16
17 /**
18  * Class that takes care of all the communication with the BioID service.
19  *
20  * @author Hans JÃ¸rgen Torp, Henrik Aarnes Kleppe, Adrian LÃ¸vlid, Joakim
21  * Fivelstad
22  */
23 public class LivenessDetection {
24
25     private final String appSecret = "qTgaiHyxqyEr9HkdJMMJYHHo";
26     private final String appId = "43a5e1f9-5ea3-4c4d-a212-d7e2913c0756";
27
28     /**
29      * Method that send two pictures to BioID service.
30      * The service returns a true og false based on
31      * if the image is from a real person or not.
32      * @param image1, first image taken from the system.
33      * @param image2, second image taken from the system.
34      * @return a boolean value that represnt the reault from the service.
35      * @throws IOException
36      * @throws JSONException
37      */
38     public boolean sendToBioId(BufferedImage image1, BufferedImage image2) throws
IOException, JSONException {
39
40         ByteArrayOutputStream outputStream1 = new ByteArrayOutputStream();
41         ByteArrayOutputStream outputStream2 = new ByteArrayOutputStream();
42
43         ImageIO.write(image1, "png", outputStream1);
```

```
44
45          ImageIO.write(image2, "png", outputStream2);
46
47          // using org.json.JSONObject from JSON-java library
48          JSONObject requestBody = new JSONObject();
49          byte[] png1AsByteArray = null;
50
51          requestBody.put(
52                  "liveimage1", "" +
Base64.getEncoder().encodeToString(outputStream1.toByteArray()));
53          byte[] png2AsByteArray = null;
54
55          requestBody.put(
56                  "liveimage2", "" +
Base64.getEncoder().encodeToString(outputStream2.toByteArray()));
57
58          // using OkHttpClient from the OkHttp library
59          Request request = new Request.Builder()
60                  .url("https://bws.bioid.com/extension/livedetection")
61                  .addHeader("Authorization", Credentials.basic(appId, appSecret))
62                  .post(RequestBody.create(MediaType.parse("application/json"),
requestBody.toString()))
63                  .build();
64          OkHttpClient client = new OkHttpClient();
65          Response response = client.newCall(request).execute();
66
67          System.out.println(response.toString());
68
69          if (response.body()
70                  .string().equals("true")) {
71              System.out.println("recorded from a live person");
72              return true;
73
74          } else {
75              System.out.println("not recorded from a live person");
76              return false;
77
78          }
79
80      }
81
82  }
```

```
 1 package FaceID;
 2
 3 import java.awt.FontFormatException;
 4 import java.io.IOException;
 5 import java.text.ParseException;
 6
 7 /**
 8  *
 9  * @author Hans JÃ¸rgen Torp, Henrik Aarnes Kleppe, Adrian LÃ¸vlid, Joakim
Fivelstad
10  */
11 public class Main {
12
13     /**
14      * Main class, represents the start class in our program.
15      * @param args the command line arguments
16      * @throws java.io.IOException
17      * @throws java.lang.InterruptedException
18      * @throws java.awt.FontFormatException
19      * @throws java.text.ParseException
20      */
21     public static void main(String[] args) throws IOException,
InterruptedException, FontFormatException, ParseException {
22
23         FaceID faceId = new FaceID();
24         faceId.run();
25     }
26
27 }
```

```
 1 package FaceID;
 2
 3 import java.util.ArrayList;
 4 import java.util.List;
 5 import org.opencv.core.Core;
 6 import org.opencv.core.Mat;
 7 import org.opencv.core.MatOfPoint;
 8 import org.opencv.core.Point;
 9 import org.opencv.core.Rect;
10 import org.opencv.core.Size;
11 import org.opencv.imgproc.Imgproc;
12
13 /**
14  * Class that holds methods to detect motion in the frame.
15  *
16  * @author Hans JÃ¸rgen Torp, Henrik Aarnes Kleppe, Adrian LÃ¸vlid, Joakim
17  * Fivelstad
18  */
19 public class MotionDetection {
20
21     Mat frame;
22     Mat comparedFrame;
23     Mat frameDelta;
24     Mat thresh;
25     FaceCrop faceCrop;
26     List<MatOfPoint> cnts;
27     Mat firstLiveMotionImage;
28     boolean confirmed;
29
30     /**
31      * Method that uses the camera to detect motion based on calculated changes
32      * in contours based on two images.
33      *
34      * @param camera, Instance of FaceID.
35      * @return a boolean value. True represnt that motion is registred. False
36      * represent the opposite.
37      */
38     public boolean checkForMotion(FaceID camera) {
39
40         faceCrop = new FaceCrop();
41         cnts = new ArrayList<>();
42         frameDelta = new Mat();
43         thresh = new Mat();
44         confirmed = false;
```

```
45
46          while (true) {
47
48              frame = camera.getOneFrame();
49              Rect frameRect = faceCrop.getFaceCrop(frame, camera);
50
51              try {
52                  frame = frame.submat(frameRect);
53              } catch (Exception e) {
54                  System.out.println("No frame..");
55                  return false;
56
57              }
58
59              if (frame.width() != 0 && frame.height() != 0) {
60
61                  Imgproc.cvtColor(frame, frame, Imgproc.COLOR_BGR2GRAY);
62                  Imgproc.GaussianBlur(frame, frame, new Size(21, 21), 0);
63
64                  while (true) {
65
66                      comparedFrame = camera.getOneFrame();
67                      Rect comparedFrameRect = faceCrop.getFaceCrop(comparedFrame,
camera);
68
69                      Rect comparedFrameRect2 = new Rect(comparedFrameRect.x,
comparedFrameRect.y, frameRect.width, frameRect.height);
70
71                      try {
72                          comparedFrame =
comparedFrame.submat(comparedFrameRect2);
73                      } catch (Exception e) {
74                          System.out.println("Frame out of bounds..");
75                          return false;
76                      }
77
78                      Imgproc.cvtColor(comparedFrame, comparedFrame,
Imgproc.COLOR_BGR2GRAY);
79                      Imgproc.GaussianBlur(comparedFrame, comparedFrame, new
Size(21, 21), 0);
80
81                      Core.absdiff(frame, comparedFrame, frameDelta);
82                      Imgproc.threshold(frameDelta, thresh, 25, 255,
Imgproc.THRESH_BINARY);
```

```
 83
 84                     Imgproc.dilate(thresh, thresh, new Mat(), new Point(-1, -1),
2);
 85                     Imgproc.findContours(thresh, cnts, new Mat(),
Imgproc.RETR_EXTERNAL, Imgproc.CHAIN_APPROX_SIMPLE);
 86
 87                     for (int i = 0; i < cnts.size(); i++) {
 88
 89                         if (Imgproc.contourArea(cnts.get(i)) > 800) {
 90
 91                             return true;
 92                         } else {
 93                             camera.timeToReset();
 94
 95                         }
 96
 97                     }
 98                     return false;
 99
100                 }
101             }
102
103         }
104
105     }
106
107 }
```

```
 1 /*
 2  * To change this license header, choose License Headers in Project Properties.
 3  * To change this template file, choose Tools | Templates
 4  * and open the template in the editor.
 5  */
 6 package FaceID;
 7
 8 import com.fazecast.jSerialComm.*;
 9 import java.io.BufferedReader;
10 import java.io.IOException;
11 import java.io.InputStreamReader;
12 import java.io.PrintWriter;
13
14 /**
15  * This class is responsible for the two-way serial communication with arduino.
16  * it can both send commands to the arduino and read strings from the arduino.
17  *
18  * @author Hans JÃ¸rgen Torp, Henrik Aarnes Kleppe, Adrian LÃ¸vlid, Joakim
19  * Fivelstad
20  */
21 public class SerialCommunication {
22
23     private PrintWriter output;
24     private SerialPort chosenPort;
25     private final String wantedPort;
26     private BufferedReader input;
27     private boolean updated;
28     private long sensorValue;
29
30     /**
31      * Constructor for SerialCommunication class
32      *
33      * @param portName The port which we want the serial communication to
34      * happen, usually "ttyUSB0" when using a raspberry Pi
35      * @throws IOException
36      */
37     public SerialCommunication(String portName) throws IOException {
38         this.wantedPort = portName; //"ttyUSB0", "ttyUSB1" etc.
39         setWantedPort();
40         connectPort();
41         //Set the updated variable to false
42         this.updated = false;
43     }
44
```

```
45      /**
46       * setWantedPort. Set the port we want to use, and also finds the available
47       * Serial ports. Arduino need to be connected to the USB port if this code
48       * should detect it
49       */
50      private void setWantedPort() {
51          SerialPort[] portNames = SerialPort.getCommPorts();
52          for (SerialPort portName : portNames) {
53              System.out.println("Ports available: " +
portName.getSystemPortName());
54              if (portName.getSystemPortName().equals(wantedPort)) {
55                  chosenPort = portName;
56              }
57          }
58         // System.out.println("chosen port is " +
chosenPort.getSystemPortName());
59      }
60
61      /**
62       * Get all connected ports
63       *
64       * @return the ports
65       */
66      public String getConnectedPorts() {
67          String ports = chosenPort.getSystemPortName();
68          return ports;
69      }
70
71      /**
72       * connectPort. Connects to the port we set as the port we wanted.
73       */
74      private void connectPort() {
75          chosenPort.setComPortTimeouts(SerialPort.TIMEOUT_SCANNER, 0, 0);
76
77          if (chosenPort.openPort()) {
78              System.out.println("Connected to port: " +
chosenPort.getSystemPortName());
79              output = new PrintWriter(chosenPort.getOutputStream());
80          }
81      }
82
83      /**
84       * sendString. Send a string to the serial port.
85       *
```

```
 86      * @param toSend string to send
 87      */
 88     public synchronized void sendString(String toSend) {
 89         output.write(toSend);
 90         output.flush();
 91 
 92         // May include line below for testing.
 93         // System.out.println("Sent: '" + toSend + "' to the Arduino.");
 94     }
 95 
 96     /**
 97      * Read a String from the arduino. Most likely a number
 98      *
 99      * @return the number that was read
100      */
101     public synchronized long readSensor() {
102         try {
103             this.input = new BufferedReader(new
InputStreamReader(chosenPort.getInputStream()));
104             long val = input.read();
105             this.sensorValue = val;
106         } catch (Exception e) {
107         }
108         return this.sensorValue;
109     }
110 
111 }
```

```
  1 package FaceID;
  2
  3 import java.security.GeneralSecurityException;
  4 import java.security.SecureRandom;
  5 import java.util.Arrays;
  6 import java.util.Random;
  7
  8 import javax.crypto.Mac;
  9 import javax.crypto.spec.SecretKeySpec;
 10
 11 /**
 12  * Java implementation for the Time-based One-Time Password (TOTP) two factor
authentication algorithm. To get this to
 13  * work you:
 14  *
 15  * <ol>
 16  * <li>Use generateBase32Secret() to generate a secret key for a user.</li>
 17  * <li>Store the secret key in the database associated with the user
account.</li>
 18  * <li>Display the QR image URL returned by qrImageUrl(...) to the user.</li>
 19  * <li>User uses the image to load the secret key into his authenticator
application.</li>
 20  * </ol>
 21  *
 22  * <p>
 23  * Whenever the user logs in:
 24  * </p>
 25  *
 26  * <ol>
 27  * <li>The user enters the number from the authenticator application into the
login form.</li>
 28  * <li>Read the secret associated with the user account from the database.</li>
 29  * <li>The server compares the user input with the output from
generateCurrentNumber(...).</li>
 30  * <li>If they are equal then the user is allowed to log in.</li>
 31  * </ol>
 32  *
 33  * <p>
 34  * See: https://github.com/j256/two-factor-auth
 35  * </p>
 36  *
 37  * <p>
 38  * For more details about this magic algorithm, see:
http://en.wikipedia.org/wiki/Time-based_One-time_Password_Algorithm
```

```
 39  * </p>
 40  *
 41  * @author graywatson
 42  */
 43 public class TimeBasedOneTimePasswordUtil {
 44
 45 /** default time-step which is part of the spec, 30 seconds is default */
 46 public static final int DEFAULT_TIME_STEP_SECONDS = 30;
 47 /** set to the number of digits to control 0 prefix, set to 0 for no prefix */
 48 private static int NUM_DIGITS_OUTPUT = 6;
 49
 50 private static final String blockOfZeros;
 51
 52 static {
 53 char[] chars = new char[NUM_DIGITS_OUTPUT];
 54 for (int i = 0; i < chars.length; i++) {
 55 chars[i] = '0';
 56 }
 57 blockOfZeros = new String(chars);
 58 }
 59
 60 /**
 61  * Generate and return a 16-character secret key in base32 format (A-Z2-7) using
{@link SecureRandom}.Could be used
 62  to generate the QR image to be shared with the user. Other lengths should use
{@link #generateBase32Secret(int)}.
 63      * @return
 64  */
 65 public static String generateBase32Secret() {
 66 return generateBase32Secret(16);
 67 }
 68
 69 /**
 70  * Similar to {@link #generateBase32Secret()} but specifies a character length.
 71      * @param length
 72      * @return
 73  */
 74 public static String generateBase32Secret(int length) {
 75 StringBuilder sb = new StringBuilder(length);
 76 Random random = new SecureRandom();
 77 for (int i = 0; i < length; i++) {
 78 int val = random.nextInt(32);
 79 if (val < 26) {
 80 sb.append((char) ('A' + val));
```

```
 81 } else {
 82 sb.append((char) ('2' + (val - 26)));
 83 }
 84 }
 85 return sb.toString();
 86 }
 87
 88 /**
 89  * Validate a given secret-number using the secret base-32 string.This allows
you to set a window in milliseconds to
 90  account for people being close to the end of the time-step. For example, if
windowMillis is 10000 then this method
 91  will check the authNumber against the generated number from 10 seconds before
now through 10 seconds after now.
 92
 93  <p>
 94  * WARNING: This requires a system clock that is in sync with the world.
 95  * </p>
 96  *
 97  * @param base32Secret
 98  *            Secret string encoded using base-32 that was used to generate the
QR code or shared with the user.
 99  * @param authNumber
100  *            Time based number provided by the user from their authenticator
application.
101  * @param windowMillis
102  *            Number of milliseconds that they are allowed to be off and still
match. This checks before and after
103  *            the current time to account for clock variance. Set to 0 for no
window.
104  * @return True if the authNumber matched the calculated number within the
specified window.
105      * @throws java.security.GeneralSecurityException
106  */
107 public static boolean validateCurrentNumber(String base32Secret, int authNumber,
int windowMillis)
108 throws GeneralSecurityException {
109 return validateCurrentNumber(base32Secret, authNumber, windowMillis,
System.currentTimeMillis(),
110 DEFAULT_TIME_STEP_SECONDS);
111 }
112
113 /**
114  * Similar to {@link #validateCurrentNumber(String, int, int)} except exposes
```

```
other parameters.Mostly for testing.
115  *
116  * @param base32Secret
117  *            Secret string encoded using base-32 that was used to generate the
QR code or shared with the user.
118  * @param authNumber
119  *            Time based number provided by the user from their authenticator
application.
120  * @param windowMillis
121  *            Number of milliseconds that they are allowed to be off and still
match. This checks before and after
122  *            the current time to account for clock variance. Set to 0 for no
window.
123  * @param timeMillis
124  *            Time in milliseconds.
125  * @param timeStepSeconds
126  *            Time step in seconds. The default value is 30 seconds here. See
{@link #DEFAULT_TIME_STEP_SECONDS}.
127  * @return True if the authNumber matched the calculated number within the
specified window.
128      * @throws java.security.GeneralSecurityException
129  */
130 public static boolean validateCurrentNumber(String base32Secret, int authNumber,
int windowMillis, long timeMillis,
131 int timeStepSeconds) throws GeneralSecurityException {
132 long from = timeMillis;
133 long to = timeMillis;
134 if (windowMillis > 0) {
135 from -= windowMillis;
136 to += windowMillis;
137 }
138 long timeStepMillis = timeStepSeconds * 1000;
139 for (long millis = from; millis <= to; millis += timeStepMillis) {
140 long compare = generateNumber(base32Secret, millis, timeStepSeconds);
141 if (compare == authNumber) {
142 return true;
143 }
144 }
145 return false;
146 }
147
148 /**
149  * Return the current number to be checked.This can be compared against user
input.<p>
```

```
150  * WARNING: This requires a system clock that is in sync with the world.
151  * </p>
152  *
153  * @param base32Secret
154  *            Secret string encoded using base-32 that was used to generate the
QR code or shared with the user.
155  * @return A number as a string with possible leading zeros which should match
the user's authenticator application
156  *         output.
157      * @throws java.security.GeneralSecurityException
158  */
159 public static String generateCurrentNumberString(String base32Secret) throws
GeneralSecurityException {
160 return generateNumberString(base32Secret, System.currentTimeMillis(),
DEFAULT_TIME_STEP_SECONDS);
161 }
162
163 /**
164  * Similar to {@link #generateCurrentNumberString(String)} except exposes other
parameters.Mostly for testing.
165  *
166  * @param base32Secret
167  *            Secret string encoded using base-32 that was used to generate the
QR code or shared with the user.
168  * @param timeMillis
169  *            Time in milliseconds.
170  * @param timeStepSeconds
171  *            Time step in seconds. The default value is 30 seconds here. See
{@link #DEFAULT_TIME_STEP_SECONDS}.
172  * @return A number as a string with possible leading zeros which should match
the user's authenticator application
173  *         output.
174      * @throws java.security.GeneralSecurityException
175  */
176 public static String generateNumberString(String base32Secret, long timeMillis,
int timeStepSeconds)
177 throws GeneralSecurityException {
178 long number = generateNumber(base32Secret, timeMillis, timeStepSeconds);
179 return zeroPrepend(number, NUM_DIGITS_OUTPUT);
180 }
181
182 /**
183  * Similar to {@link #generateCurrentNumberString(String)} but this returns a
long instead of a string.
```

```
184  *
185      * @param base32Secret
186  * @return A number which should match the user's authenticator application
output.
187      * @throws java.security.GeneralSecurityException
188  */
189 public static long generateCurrentNumber(String base32Secret) throws
GeneralSecurityException {
190 return generateNumber(base32Secret, System.currentTimeMillis(),
DEFAULT_TIME_STEP_SECONDS);
191 }
192
193 /**
194  * Similar to {@link #generateNumberString(String, long, int)} but this returns
a long instead of a string.
195  *
196      * @param base32Secret
197      * @param timeMillis
198      * @param timeStepSeconds
199  * @return A number which should match the user's authenticator application
output.
200      * @throws java.security.GeneralSecurityException
201  */
202 public static long generateNumber(String base32Secret, long timeMillis, int
timeStepSeconds)
203 throws GeneralSecurityException {
204
205 byte[] key = decodeBase32(base32Secret);
206
207 byte[] data = new byte[8];
208 long value = timeMillis / 1000 / timeStepSeconds;
209 for (int i = 7; value > 0; i--) {
210 data[i] = (byte) (value & 0xFF);
211 value >>= 8;
212 }
213
214 // encrypt the data with the key and return the SHA1 of it in hex
215 SecretKeySpec signKey = new SecretKeySpec(key, "HmacSHA1");
216 // if this is expensive, could put in a thread-local
217 Mac mac = Mac.getInstance("HmacSHA1");
218 mac.init(signKey);
219 byte[] hash = mac.doFinal(data);
220
221 // take the 4 least significant bits from the encrypted string as an offset
```

```
222 int offset = hash[hash.length - 1] & 0xF;
223
224 // We're using a long because Java hasn't got unsigned int.
225 long truncatedHash = 0;
226 for (int i = offset; i < offset + 4; ++i) {
227 truncatedHash <<= 8;
228 // get the 4 bytes at the offset
229 truncatedHash |= (hash[i] & 0xFF);
230 }
231 // cut off the top bit
232 truncatedHash &= 0x7FFFFFFF;
233
234 // the token is then the last 6 digits in the number
235 truncatedHash %= 1000000;
236
237 return truncatedHash;
238 }
239
240 /**
241  * Return the QR image url thanks to Google.This can be shown to the user and
scanned by the authenticator program
242  as an easy way to enter the secret.
243  *
244  * @param keyId
245  *            Name of the key that you want to show up in the users
authentication application. Should already be
246  *            URL encoded.
247  * @param secret
248  *            Secret string that will be used when generating the current
number.
249      * @return
250  */
251 public static String qrImageUrl(String keyId, String secret) {
252 StringBuilder sb = new StringBuilder(128);
253
sb.append("https://chart.googleapis.com/chart?chs=200x200&cht=qr&chl=200x200&chld=M|
0&cht=qr&chl=");
254 addOtpAuthPart(keyId, secret, sb);
255 return sb.toString();
256 }
257
258 /**
259  * Return the otp-auth part of the QR image which is suitable to be injected
into other QR generators (e.g.JS
```

```
260  generator).
261  *
262  * @param keyId
263  *            Name of the key that you want to show up in the users
authentication application. Should already be
264  *            URL encoded.
265  * @param secret
266  *            Secret string that will be used when generating the current
number.
267      * @return
268  */
269 public static String generateOtpAuthUrl(String keyId, String secret) {
270 StringBuilder sb = new StringBuilder(64);
271 addOtpAuthPart(keyId, secret, sb);
272 return sb.toString();
273 }
274
275 private static void addOtpAuthPart(String keyId, String secret, StringBuilder
sb) {
276
sb.append("otpauth://totp/").append(keyId).append("%3Fsecret%3D").append(secret);
277 }
278
279 /**
280  * Return the string prepended with 0s. Tested as 10x faster than
String.format("%06d", ...); Exposed for testing.
281  */
282 static String zeroPrepend(long num, int digits) {
283 String numStr = Long.toString(num);
284 if (numStr.length() >= digits) {
285 return numStr;
286 } else {
287 StringBuilder sb = new StringBuilder(digits);
288 int zeroCount = digits - numStr.length();
289 sb.append(blockOfZeros, 0, zeroCount);
290 sb.append(numStr);
291 return sb.toString();
292 }
293 }
294
295 /**
296  * Decode base-32 method. I didn't want to add a dependency to Apache Codec just
for this decode method. Exposed for
297  * testing.
```

```
298  */
299 static byte[] decodeBase32(String str) {
300 // each base-32 character encodes 5 bits
301 int numBytes = ((str.length() * 5) + 7) / 8;
302 byte[] result = new byte[numBytes];
303 int resultIndex = 0;
304 int which = 0;
305 int working = 0;
306 for (int i = 0; i < str.length(); i++) {
307 char ch = str.charAt(i);
308 int val;
309 if (ch >= 'a' && ch <= 'z') {
310 val = ch - 'a';
311 } else if (ch >= 'A' && ch <= 'Z') {
312 val = ch - 'A';
313 } else if (ch >= '2' && ch <= '7') {
314 val = 26 + (ch - '2');
315 } else if (ch == '=') {
316 // special case
317 which = 0;
318 break;
319 } else {
320 throw new IllegalArgumentException("Invalid base-32 character: " + ch);
321 }
322 /*
323  * There are probably better ways to do this but this seemed the most
straightforward.
324  */
325 switch (which) {
326 case 0:
327 // all 5 bits is top 5 bits
328 working = (val & 0x1F) << 3;
329 which = 1;
330 break;
331 case 1:
332 // top 3 bits is lower 3 bits
333 working |= (val & 0x1C) >> 2;
334 result[resultIndex++] = (byte) working;
335 // lower 2 bits is upper 2 bits
336 working = (val & 0x03) << 6;
337 which = 2;
338 break;
339 case 2:
340 // all 5 bits is mid 5 bits
```

```
341 working |= (val & 0x1F) << 1;
342 which = 3;
343 break;
344 case 3:
345 // top 1 bit is lowest 1 bit
346 working |= (val & 0x10) >> 4;
347 result[resultIndex++] = (byte) working;
348 // lower 4 bits is top 4 bits
349 working = (val & 0x0F) << 4;
350 which = 4;
351 break;
352 case 4:
353 // top 4 bits is lowest 4 bits
354 working |= (val & 0x1E) >> 1;
355 result[resultIndex++] = (byte) working;
356 // lower 1 bit is top 1 bit
357 working = (val & 0x01) << 7;
358 which = 5;
359 break;
360 case 5:
361 // all 5 bits is mid 5 bits
362 working |= (val & 0x1F) << 2;
363 which = 6;
364 break;
365 case 6:
366 // top 2 bits is lowest 2 bits
367 working |= (val & 0x18) >> 3;
368 result[resultIndex++] = (byte) working;
369 // lower 3 bits of byte 6 is top 3 bits
370 working = (val & 0x07) << 5;
371 which = 7;
372 break;
373 case 7:
374 // all 5 bits is lower 5 bits
375 working |= (val & 0x1F);
376 result[resultIndex++] = (byte) working;
377 which = 0;
378 break;
379 }
380 }
381 if (which != 0) {
382 result[resultIndex++] = (byte) working;
383 }
384 if (resultIndex != result.length) {
```

```
385 result = Arrays.copyOf(result, resultIndex);
386 }
387 return result;
388 }
389 }
```

```
 1 package FaceID;
 2
 3 import java.awt.image.BufferedImage;
 4 import java.awt.image.DataBufferByte;
 5 import java.io.IOException;
 6 import java.net.MalformedURLException;
 7 import java.net.URL;
 8 import java.security.GeneralSecurityException;
 9 import java.security.SecureRandom;
10 import java.util.logging.Level;
11 import java.util.logging.Logger;
12 import javax.imageio.ImageIO;
13 import org.apache.commons.codec.binary.Base32;
14 import org.opencv.core.CvType;
15 import org.opencv.core.Mat;
16 import static org.opencv.imgcodecs.Imgcodecs.imwrite;
17
18 /**
19  * Class that handles the calculation of a new code based on the TOTP algorithm.
20  *
21  * @author Hans JÃ¸rgen Torp, Henrik Aarnes Kleppe, Adrian LÃ¸vlid, Joakim
22  * Fivelstad
23  */
24 public class TwoFactorAuth implements Runnable {
25
26     /**
27      * Public varable that holds the twoFactor code.
28      */
29     public volatile String code;
30     BufferedImage img;
31
32     @Override
33     public void run() {
34
35
System.load("C:\\incubating-netbeans-10.0-bin\\netbeans\\java\\opencv\\build\\java\\
x64\\opencv_java401.dll");
36
37         //TimeBasedOneTimePasswordUtil base = new
TimeBasedOneTimePasswordUtil();
38         String test = "7X4LGULIRVH6TTMP";
39
40         //String base32Secret = base.generateBase32Secret();
41         System.out.println(
```

```
 42                 "secret = " + test);
 43
 44         // this is the name of the key which can be displayed by the
authenticator program
 45         String keyId = "Avento";
 46         // generate the QR code
 47
 48         System.out.println(
 49                 "Image url = " + TimeBasedOneTimePasswordUtil.qrImageUrl(keyId,
test));
 50         try {
 51             img = ImageIO.read(new
URL(TimeBasedOneTimePasswordUtil.qrImageUrl(keyId, test)));
 52         } catch (MalformedURLException ex) {
 53             Logger.getLogger(TwoFactorAuth.class.getName()).log(Level.SEVERE,
null, ex);
 54         } catch (IOException ex) {
 55             Logger.getLogger(TwoFactorAuth.class.getName()).log(Level.SEVERE,
null, ex);
 56         }
 57
 58         imwrite(
 59                 "URL.jpg", bufferedImageToMat(img));
 60         // we can display this image to the user to let them load it into their
auth program
 61         try {
 62             // we can use the code here and compare it against user input
 63             //String code =
TimeBasedOneTimePasswordUtil.generateCurrentNumberString(base32Secret);
 64             code =
TimeBasedOneTimePasswordUtil.generateCurrentNumberString(test);
 65             /*
 66             * this loop shows how the number changes over time
 67              */
 68         } catch (GeneralSecurityException ex) {
 69             Logger.getLogger(TwoFactorAuth.class.getName()).log(Level.SEVERE,
null, ex);
 70         }
 71
 72         while (true) {
 73             long diff = TimeBasedOneTimePasswordUtil.DEFAULT_TIME_STEP_SECONDS
 74                     - ((System.currentTimeMillis() / 1000) %
TimeBasedOneTimePasswordUtil.DEFAULT_TIME_STEP_SECONDS);
 75             try {
```

```
 76                 code =
TimeBasedOneTimePasswordUtil.generateCurrentNumberString(test);
 77             } catch (GeneralSecurityException ex) {
 78
Logger.getLogger(TwoFactorAuth.class.getName()).log(Level.SEVERE, null, ex);
 79             }
 80             try {
 81                 Thread.sleep(1000);
 82             } catch (InterruptedException ex) {
 83
Logger.getLogger(TwoFactorAuth.class.getName()).log(Level.SEVERE, null, ex);
 84             }
 85
 86         }
 87
 88     }
 89
 90     /**
 91      * Mehod that generates a new random key
 92      *
 93      * @return a String containing a random key.
 94      */
 95     public static String getRandomSecretKey() {
 96         SecureRandom random = new SecureRandom();
 97         byte[] bytes = new byte[8];
 98         random.nextBytes(bytes);
 99         Base32 base32 = new Base32();
100         String secretKey = base32.encodeToString(bytes);
101         String secretKey1 = base32.encodeAsString(bytes);
102         System.out.println(secretKey1);
103         // make the secret key more human-readable by lower-casing and
104         // inserting spaces between each group of 4 characters
105         return secretKey1.toUpperCase().replaceAll("(.{4})(?=.{4})", "$1 ");
106     }
107
108     /**
109      * Method that takes in a BufferedImage and returns a OpenCV Mat.
110      *
111      * @param bi, BufferedImage
112      * @return a OpenCV Mat.
113      */
114     public static Mat bufferedImageToMat(BufferedImage bi) {
115         Mat mat = new Mat(bi.getHeight(), bi.getWidth(), CvType.CV_8UC3);
116         byte[] data = ((DataBufferByte)
```

```
bi.getRaster().getDataBuffer()).getData();
117         mat.put(0, 0, data);
118         return mat;
119     }
120 }
```

Admin GUI

```
 1
 2 package FaceAPI;
 3
 4
 5 import java.net.URI;
 6 import org.apache.http.HttpEntity;
 7 import org.apache.http.HttpResponse;
 8 import org.apache.http.client.HttpClient;
 9 import org.apache.http.client.methods.HttpPost;
10 import org.apache.http.entity.StringEntity;
11 import org.apache.http.client.utils.URIBuilder;
12 import org.apache.http.impl.client.HttpClientBuilder;
13 import org.apache.http.util.EntityUtils;
14
15 import org.json.JSONArray;
16 import org.json.JSONObject;
17 import java.io.File;
18 import org.apache.http.entity.FileEntity;
19 import org.json.JSONException;
20
21 /**
22  * This class is responsible for checking if a person is already in the system
23  * when we try to add a new person.
24  *
25  * @author henri
26  */
27 public class Azure_GUI {
28
29     // Replace <Subscription Key> with your valid subscription key.
30     private static final String subscriptionKey =
"dc84d7ce01df4c61b247f43fd4f71f7d";
31
32     private static final String uriBase
33             =
"https://northeurope.api.cognitive.microsoft.com/face/v1.0/detect";
34
35     private static final String imageWithFaces
36             =
"{\"url\":\"https://media.snl.no/system/images/33747/standard_Donald_Trumo.jpg\"}";
37
38     private static final String faceAttributes
39             = "headPose";
40
41     File file = new File("D:\\Skole\\6.
```

```
semester\\Bachelor\\FaceAPI_TEST\\1.png");
 42
 43     private String faceId = "";
 44
 45     private String compareString = "";
 46
 47     public volatile double confidence;
 48     public volatile boolean newResult = false;
 49
 50     public volatile String personName;
 51     public String id;
 52
 53     /**
 54      * Detects a face in a picture. This method gets a FaceID which we use in
 55      * identify()
 56      *
 57      * @param fileName picture to scan for faces
 58      * @param groupId ID of group we want to add a person to
 59      */
 60     public void detectFace(File fileName, String groupId) {
 61
 62         HttpClient httpclient = HttpClientBuilder.create().build();
 63
 64         try {
 65             URIBuilder builder = new URIBuilder(uriBase);
 66
 67             // Request parameters. All of them are optional.
 68             builder.setParameter("returnFaceId", "true");
 69             builder.setParameter("returnFaceLandmarks", "false");
 70             builder.setParameter("returnFaceAttributes", faceAttributes);
 71
 72             // Prepare the URI for the REST API call.
 73             URI uri = builder.build();
 74             HttpPost request = new HttpPost(uri);
 75
 76             // Request headers.
 77             request.setHeader("Content-Type", "application/octet-stream");
 78             request.setHeader("Ocp-Apim-Subscription-Key", subscriptionKey);
 79
 80             HttpEntity face1 = new FileEntity(fileName);
 81             request.setEntity(face1);
 82
 83             // Execute the REST API call and get the response entity.
 84             HttpResponse response = httpclient.execute(request);
```

```
 85             HttpEntity entity = response.getEntity();
 86             if (entity != null) {
 87 
 88                 String str = EntityUtils.toString(entity).trim();
 89                 JSONArray jsonArray = new JSONArray(str);
 90                 try {
 91 
 92                     for (int i = 0; i < jsonArray.length(); ++i) {
 93                         JSONObject faces = jsonArray.getJSONObject(i);
 94                         faceId = faces.getString("faceId");
 95 
 96                     }
 97                 } catch (JSONException e) {
 98                     System.out.println(e.getMessage());
 99                 }
100                 JSONObject obj = new JSONObject();
101                 obj.put("personGroupId", groupId);
102                 JSONArray array = new JSONArray();
103                 array.put(faceId);
104                 obj.put("faceIds", array);
105                 compareString = obj.toString();
106             }
107         } catch (Exception e) {
108             // Display error message.
109             System.out.println(e.getMessage());
110         }
111     }
112 
113     /**
114      * This method checks if a person is already in the database using FaceID
115      * found in detectFace()
116      *
117      * @return boolean true or false based on if the person is identified
118      */
119     public boolean identify() {
120         boolean result = false;
121 
122         HttpClient httpclient = HttpClientBuilder.create().build();
123 
124         try {
125             URIBuilder builder = new URIBuilder(
126 
"https://northeurope.api.cognitive.microsoft.com/face/v1.0/identify");
127 
```

```
128             URI uri = builder.build();
129             HttpPost request = new HttpPost(uri);
130 
131             request.setHeader("Content-Type", "application/json");
132             request.setHeader("Ocp-Apim-Subscription-Key", subscriptionKey);
133 
134             // Request body
135 
136             StringEntity reqEntity = new StringEntity(compareString);
137             request.setEntity(reqEntity);
138 
139             // Execute the REST API call and get the response entity.
140             HttpResponse response = httpclient.execute(request);
141             HttpEntity entity = response.getEntity();
142 
143             if (entity != null) {
144                 // Format and display the JSON response.
145                 String str = EntityUtils.toString(entity).trim();
146                 JSONArray jsonArray = new JSONArray(str);
147                 try {
148                     System.out.println("The identify array has " +
jsonArray.length() + " items.");
149                     for (int i = 0; i < jsonArray.length(); ++i) {
150                         JSONObject faces = jsonArray.getJSONObject(i);
151                         String faceid = faces.getString("faceId");
152 
153                         JSONArray candidates = faces.getJSONArray("candidates");
154                         for (int j = 0; j < candidates.length(); ++j) {
155                             JSONObject candidateObjects =
candidates.getJSONObject(j);
156                             id = candidateObjects.getString("personId");
157                             confidence =
candidateObjects.getDouble("confidence");
158                         }
159                     }
160 
161                     if (confidence > 0.5) {
162                         result = true;
163                     } else {
164                         result = false;
165                     }
166                 } catch (JSONException e) {
167                     System.out.println(e.getMessage());
168                 }
```

```
169             }
170 
171         } catch (Exception e) {
172             // Display error message.
173             System.out.println(e.getMessage());
174         }
175         return result;
176 
177     }
178 
179 }
```

```
 1
 2 package FaceAPI;
 3
 4 import java.io.IOException;
 5
 6 /**
 7  *
 8  * @author henri
 9  */
10 public class Main {
11
12     /**
13      * @param args the command line arguments
14      * @throws java.io.IOException
15      * @throws java.lang.InterruptedException
16      */
17     public static void main(String[] args) throws IOException,
InterruptedException {
18
19         WebCamFeedGUI GUI = new WebCamFeedGUI();
20
21     }
22
23 }
```

```
 1 package FaceAPI;
 2
 3 import java.net.URI;
 4 import java.util.ArrayList;
 5 import java.util.HashMap;
 6 import javax.swing.JOptionPane;
 7 import org.apache.http.HttpEntity;
 8 import org.apache.http.HttpResponse;
 9 import org.apache.http.client.HttpClient;
10 import org.apache.http.client.methods.HttpDelete;
11 import org.apache.http.client.methods.HttpGet;
12 import org.apache.http.client.methods.HttpPost;
13 import org.apache.http.client.methods.HttpPut;
14 import org.apache.http.entity.StringEntity;
15 import org.apache.http.client.utils.URIBuilder;
16 import org.apache.http.impl.client.HttpClientBuilder;
17 import org.apache.http.util.EntityUtils;
18 import org.json.JSONArray;
19 import org.json.JSONException;
20 import org.json.JSONObject;
21
22 /**
23  *
24  * @author henri
25  */
26 public class PersonGroup {
27
28     // Replace <Subscription Key> with your valid subscription key.
29     private static final String subscriptionKey =
"dc84d7ce01df4c61b247f43fd4f71f7d";
30     public volatile ArrayList<String> groupArray = new ArrayList();
31     public volatile HashMap groups = new HashMap();
32
33     // Include this if we want access to more faceAttributes.
34     /*private static final String faceAttributes
35             = "age,gender,headPose,smile,facialHair,glasses,emotion,hair,makeup,
36                 occlusion,accessories,blur,exposure,noise";*/
37     /**
38      * Main method used for testing purposes.
39      *
40      * @param args the command line arguments
41      */
42     public static void main(String[] args) {
43         // The following code lines can be uncommented for testing purposes.
```

```
44
45          //PersonGroup pg = new PersonGroup();
46          //pg.createPersonGroup("200", "test2");
47          //pg.listPersonGroup();
48          //pg.deletePersonGroup("300");
49          //pg.listPersonGroup();
50          //pg.getPersonGroup();
51          //pg.getTrainingStatus();
52          //pg.trainPersonGroup();
53      }
54
55      /**
56       * This method creates a new PersonGroup with given parameters
57       *
58       * @param groupId The ID of the group. String consisting of 64 characters
59       * Only small letters (a-z) or numbers (0-9)
60       * @param groupName the chosen name of the PersonGroup
61       */
62      public void createPersonGroup(String groupId, String groupName) {
63
64          HttpClient httpclient = HttpClientBuilder.create().build();
65
66          try {
67              URIBuilder builder = new
URIBuilder("https://northeurope.api.cognitive.microsoft.com/face/v1.0/persongroups/g
roupId");
68
69              /* Request parameters. All of them are optional. Include if
70              faceAttributes is included (line 39).
71              builder.setParameter("returnFaceId", "true");
72              builder.setParameter("returnFaceLandmarks", "true");
73              builder.setParameter("returnFaceAttributes", faceAttributes);
74               */
75              builder.setParameter("personGroupId", groupId);
76
77              // Prepare the URI for the REST API call.
78              URI uri = builder.build();
79              HttpPut request = new HttpPut(uri);
80
81              // Request headers.
82              request.setHeader("Content-Type", "application/json");
83              request.setHeader("Ocp-Apim-Subscription-Key", subscriptionKey);
84
85              // Request body.
```

```
 86              String groupInfo = "{\"name\":\"" + groupName + "\"}";
 87              StringEntity reqEntity = new StringEntity(groupInfo);
 88              request.setEntity(reqEntity);
 89
 90              // Execute the REST API call and get the response entity.
 91              HttpResponse response = httpclient.execute(request);
 92              HttpEntity entity = response.getEntity();
 93              if (entity != null) {
 94                  // Format and display the JSON response.
 95                  String jsonString = EntityUtils.toString(entity).trim();
 96                  if (jsonString.isEmpty()) {//isBlank()){
 97                      infoBox("Created new PersonGroup \n ID: " + groupId
 98                              + " \n Name: " + groupName, "Created new group");
 99                  }
100                  if (jsonString.contains("error")) {
101                      // If JSON Response is blank, we assume the creation of the
102                      // group has been successful and display a message to the
user
103                      if (jsonString.isEmpty())/*isBlank())*/ {
104                          infoBox("Created new PersonGroup \n ID: " + groupId
105                                  + " \n Name: " + groupName, "Created new
group");
106                      }
107
108                      // Check if the jsonString contains an error message
109                      if (jsonString.contains("error")) {
110                          JSONObject jsonObject = new JSONObject(jsonString);
111                          JSONObject obj = jsonObject.getJSONObject("error");
112                          String errorCode = obj.getString("code");
113                          String errorMessage = obj.getString("message");
114
115                          // Display the error message in a popup box
116                          infoBox("ERROR: " + errorCode + "\n"
117                                  + errorMessage, "An error has occurred");
118                      }
119                  }
120              }
121          } catch (Exception e) {
122              // Display error message.
123              System.out.println(e.getMessage());
124          }
125      }
126
127      /**
```

```
128      * This method deletes an existing persongroup by groupName.
129      *
130      * @param groupName Name of group to delete. String.
131      */
132     public void deletePersonGroup(String groupName) {
133         // Get groupId from groups HashMap. We use the groupId when deleting a
134         // group.
135         Object groupIdObject = groups.get(groupName);
136         String groupIdToDelete = groupIdObject.toString();
137         HttpClient httpclient = HttpClientBuilder.create().build();
138 
139         try {
140             URIBuilder builder = new URIBuilder(
141                     "https://northeurope.api.cognitive.microsoft.com/face/v1.0/"
142                     + "persongroups/" + groupIdToDelete);
143 
144             builder.setParameter("personGroupId", groupIdToDelete);
145 
146             URI uri = builder.build();
147             HttpDelete request = new HttpDelete(uri);
148 
149             // Request headers.
150             request.setHeader("Ocp-Apim-Subscription-Key", subscriptionKey);
151 
152             // Execute the REST API call and get the response entity.
153             HttpResponse response = httpclient.execute(request);
154             HttpEntity entity = response.getEntity();
155             if (entity != null) {
156                 // Format and display the JSON response.
157                 String jsonString = EntityUtils.toString(entity).trim();
158 
159                 // Check if the jsonString contains an error.
160                 if (jsonString.contains("error")) {
161                     JSONObject jsonObject = new JSONObject(jsonString);
162                     JSONObject obj = jsonObject.getJSONObject("error");
163                     String errorCode = obj.getString("code");
164                     String errorMessage = obj.getString("message");
165 
166                     // Display the error message in a popup box
167                     infoBox("ERROR: " + errorCode + "\n"
168                             + errorMessage, "An error has occurred");
169                 } else {
170                     // If there is no error, we assume the operation was
171                     // successful and display a pop up box to the user
```

```
172                     infoBox("Deleted the following group:\n " + "GroupID: "
173                             + groupIdToDelete + "\n Name: " + groupName,
174                             "Deleted Group");
175                 }
176             }
177         } catch (Exception e) {
178             // Display error message.
179             System.out.println(e.getMessage());
180         }
181     }
182 
183     /**
184      * This method populates an ArrayList of groupNames and a HashMap with
185      * groupNames and their groupId. If the Map or ArrayList is not empty, we
186      * make sure to clear them before moving on.
187      */
188     public void listPersonGroup() {
189         if (!groupArray.isEmpty()) {
190             groupArray.clear();
191         }
192         if (!groups.isEmpty()) {
193             groups.clear();
194         }
195 
196         HttpClient httpclient = HttpClientBuilder.create().build();
197 
198         try {
199             URIBuilder builder = new
URIBuilder("https://northeurope.api.cognitive.microsoft.com/face/v1.0/persongroups")
;
200 
201             builder.setParameter("start", "0");
202             //builder.setParameter("top", "1000");
203 
204             URI uri = builder.build();
205             HttpGet request = new HttpGet(uri);
206 
207             // Request headers.
208             request.setHeader("Ocp-Apim-Subscription-Key", subscriptionKey);
209 
210             // Execute the REST API call and get the response entity.
211             HttpResponse response = httpclient.execute(request);
212             HttpEntity entity = response.getEntity();
213             if (entity != null) {
```

```
214                 String jsonString = EntityUtils.toString(entity).trim();
215
216                 try {
217                     JSONArray jsonArray = new JSONArray(jsonString);
218                     System.out.println("Person Group array has " +
jsonArray.length() + " items");
219                     // Parse the JSON response and retrieve the information we
220                     // we need. I.e. persongroupname and id.
221                     for (int i = 0; i < jsonArray.length(); ++i) {
222                         JSONObject group = jsonArray.getJSONObject(i);
223                         String personGroupId = group.getString("personGroupId");
224                         String personGroupName = group.getString("name");
225                         // Add name and id to the HashMap
226                         groups.put(personGroupName, personGroupId);
227                         // Add names to the ArrayList.
228                         groupArray.add(personGroupName);
229                     }
230                 } catch (JSONException e) {
231                     System.out.println(e.getMessage());
232                 }
233             }
234         } catch (Exception e) {
235             // Display error message.
236             System.out.println(e.getMessage());
237         }
238     }
239
240     /**
241      * This method retrieves information about a persongroup. Currently not in
242      * use.
243      *
244      * @param groupId String
245      */
246     public void getPersonGroup(String groupId) {
247
248         HttpClient httpclient = HttpClientBuilder.create().build();
249
250         try {
251             URIBuilder builder = new URIBuilder(
252                     "https://northeurope.api.cognitive.microsoft.com/face/v1.0/"
253                     + "persongroups/" + groupId);
254
255             builder.setParameter("personGroupId", groupId);
256
```

```
257             URI uri = builder.build();
258             HttpGet request = new HttpGet(uri);
259 
260             // Request headers.
261             request.setHeader("Ocp-Apim-Subscription-Key", subscriptionKey);
262 
263             // Execute the REST API call and get the response entity.
264             HttpResponse response = httpclient.execute(request);
265             HttpEntity entity = response.getEntity();
266             if (entity != null) {
267                 // Format and display the JSON response.
268                 String jsonString = EntityUtils.toString(entity).trim();
269             }
270         } catch (Exception e) {
271             // Display error message.
272             System.out.println(e.getMessage());
273         }
274     }
275 
276     /**
277      * This method gets the current Training status of a persongroup
278      *
279      * @param groupId
280      */
281     public void getTrainingStatus(String groupId) {
282 
283         HttpClient httpclient = HttpClientBuilder.create().build();
284 
285         try {
286             URIBuilder builder = new URIBuilder(
287                     "https://northeurope.api.cognitive.microsoft.com/face/v1.0/"
288                     + "persongroups/" + groupId + "/training");
289 
290             builder.setParameter("personGroupId", groupId);
291 
292             URI uri = builder.build();
293             HttpGet request = new HttpGet(uri);
294 
295             // Request headers.
296             request.setHeader("Ocp-Apim-Subscription-Key", subscriptionKey);
297 
298             // Execute the REST API call and get the response entity.
299             HttpResponse response = httpclient.execute(request);
300             HttpEntity entity = response.getEntity();
```

```
301             if (entity != null) {
302                 // Format and display the JSON response.
303                 String jsonString = EntityUtils.toString(entity).trim();
304             }
305         } catch (Exception e) {
306             // Display error message.
307             System.out.println(e.getMessage());
308         }
309     }
310 
311     /**
312      * This method trains the persongroup based on new additions to the group.
313      *
314      * @param groupId
315      */
316     public void trainPersonGroup(String groupId) {
317 
318         HttpClient httpclient = HttpClientBuilder.create().build();
319 
320         try {
321             URIBuilder builder = new URIBuilder(
322                     "https://northeurope.api.cognitive.microsoft.com/face/v1.0/"
323                     + "persongroups/" + groupId + "/train");
324 
325             builder.setParameter("personGroupId", groupId);
326 
327             URI uri = builder.build();
328             HttpPost request = new HttpPost(uri);
329 
330             // Request headers.
331             request.setHeader("Ocp-Apim-Subscription-Key", subscriptionKey);
332 
333             // Execute the REST API call and get the response entity.
334             HttpResponse response = httpclient.execute(request);
335             HttpEntity entity = response.getEntity();
336             if (entity != null) {
337                 // Format and display the JSON response.
338                 String jsonString = EntityUtils.toString(entity).trim();
339             }
340         } catch (Exception e) {
341             // Display error message.
342             System.out.println(e.getMessage());
343         }
344 
```

```
345     }
346
347     /**
348      * Popup box for showing messages to the user. This method was found on
349      * StackOverflow at the following link:
350      * https://stackoverflow.com/questions/7080205/popup-message-boxes
351      *
352      *
353      * @param message the message to show.
354      * @param title the title of the messagebox
355      */
356     private static void infoBox(String message, String title) {
357         // null puts it in the center of the screen. Size is adjusted according
358         // to the size of the message.
359         JOptionPane.showMessageDialog(null, message, "InfoBox: " + title,
JOptionPane.INFORMATION_MESSAGE);
360     }
361
362 }
```

```
 1 package FaceAPI;
 2
 3 import java.io.File;
 4 import java.util.ArrayList;
 5 import java.util.HashMap;
 6 import org.apache.http.client.methods.HttpDelete;
 7 import org.apache.http.client.methods.HttpPost;
 8 import org.apache.http.entity.StringEntity;
 9 import org.apache.http.entity.FileEntity;
10 import org.apache.http.impl.client.HttpClientBuilder;
11 import java.net.URI;
12 import java.util.Iterator;
13 import javax.swing.JOptionPane;
14 import org.apache.http.HttpEntity;
15 import org.apache.http.HttpResponse;
16 import org.apache.http.client.HttpClient;
17 import org.apache.http.client.methods.HttpGet;
18 import org.apache.http.client.utils.URIBuilder;
19 import org.apache.http.impl.client.HttpClients;
20 import org.apache.http.util.EntityUtils;
21 import org.json.JSONArray;
22 import org.json.JSONException;
23 import org.json.JSONObject;
24
25 /**
26  *
27  * @author hansjtorp
28  */
29 public class PersonGroupPerson {
30
31     // Replace <Subscription Key> with your valid subscription key.
32     private static final String subscriptionKey =
"dc84d7ce01df4c61b247f43fd4f71f7d";
33     public String faceId = "";
34     public String personIdString = "";
35     public volatile ArrayList<String> userName = new ArrayList();
36     public volatile HashMap users = new HashMap();
37     private volatile ArrayList<String> pfIdArray = new ArrayList();
38
39     /**
40      * Main. This is used solely for testing the different methods in the
41      * PersonGroupPerson class
42      *
43      * @param args the command line arguments
```

```
 44      */
 45     public static void main(String[] args) {
 46         // The following code lines can be uncommented for testing purposes.
 47
 48         //PersonGroupPerson pgp = new PersonGroupPerson();
 49         //File file = new
File("/Users/hansjorgentorp/Documents/BACHELOR/bilde/bilde1.jpg");
 50         //pgp.addFaceToPerson(file, personId);
 51         //pgp.createPersonToPg("Henrik", "100");
 52         //pgp.deletePerson();
 53         //pgp.deletePersonFace();
 54         //pgp.listPersons();
 55     }
 56
 57     /**
 58      * This method adds a detected face to an existing Person. When adding a new
 59      * Person to a persongroup this method will be called after the Person has
 60      * been created. This associates the captured and detected FaceID from the
 61      * File with the personID of the created Person.
 62      *
 63      * @param file A photo with a face
 64      * @param personId Add face and faceId to this personId
 65      * @param personGroupId id of the persongroup in which the Person is
 66      */
 67     public void addFaceToPerson(File file, String personId, String
personGroupId) {
 68
 69         HttpClient httpclient = HttpClientBuilder.create().build();
 70
 71         try {
 72             URIBuilder builder = new URIBuilder(
 73                     "https://northeurope.api.cognitive.microsoft.com/"
 74                     + "face/v1.0/persongroups/" + personGroupId
 75                     + "/persons/" + personId + "/persistedFaces");
 76
 77             builder.setParameter("personGroupId", personGroupId);
 78             builder.setParameter("personId", personId);
 79
 80             URI uri = builder.build();
 81             HttpPost request = new HttpPost(uri);
 82             request.setHeader("Content-Type", "application/octet-stream");
 83
 84             // Request headers.
 85             request.setHeader("Ocp-Apim-Subscription-Key", subscriptionKey);
```

```
 86
 87             // Request body.
 88             HttpEntity face1 = new FileEntity(file);
 89             request.setEntity(face1);
 90
 91             // Execute the REST API call and get the response entity.
 92             HttpResponse response = httpclient.execute(request);
 93             HttpEntity entity = response.getEntity();
 94             if (entity != null) {
 95
 96                 // Format and display the JSON response.
 97                 String jsonString = EntityUtils.toString(entity).trim();
 98
 99                 // Check if we have gotten an error from the JSON Response..
100                 if (jsonString.contains("error")) {
101                     JSONObject jsonObject = new JSONObject(jsonString);
102                     JSONObject obj = jsonObject.getJSONObject("error");
103                     String errorCode = obj.getString("code");
104                     String errorMessage = obj.getString("message");
105
106                     // Display the error message in the infoBox (A popup Box)
107                     infoBox("ERROR: " + errorCode + "\n"
108                             + errorMessage, "An error has occurred");
109                 }
110                 // If the jsonString does not contain "error" we assume
111                 // everything went fine and move on.
112             }
113         } catch (Exception e) {
114             // Display error message.
115             System.out.println(e.getMessage());
116         }
117     }
118
119     /**
120      * This method creates a new Person and adds that person to a personGroup
121      *
122      * @param userData A string made from a JSONObject. Can contain whatever is
123      * preferred. In our case, this contains name, phone number and email
124      * address. To extract this info, use JSON parsing like we have done further
125      * down.
126      * @param groupId The groupID to add the person to.
127      */
128     public void createPersonToPg(String userData, String groupId) {
129         // Make everything upperCase for easier use of the information.
```

```
130         userData = userData.toUpperCase();
131         HttpClient httpclient = HttpClientBuilder.create().build();
132         try {
133             URIBuilder builder = new URIBuilder(
134                     "https://northeurope.api.cognitive.microsoft.com/face/v1.0/"
135                     + "persongroups/" + groupId + "/persons");
136 
137             builder.setParameter("personGroupId", groupId);
138 
139             URI uri = builder.build();
140             HttpPost request = new HttpPost(uri);
141             request.setHeader("Content-Type", "application/json");
142 
143             // Request headers.
144             request.setHeader("Ocp-Apim-Subscription-Key", subscriptionKey);
145 
146             // Request body
147             StringEntity reqEntity = new StringEntity(userData);
148             request.setEntity(reqEntity);
149 
150             // Execute the REST API call and get the response entity.
151             HttpResponse response = httpclient.execute(request);
152             HttpEntity entity = response.getEntity();
153 
154             if (entity != null) {
155 
156                 // Format and display the JSON response.
157                 String jsonString = EntityUtils.toString(entity).trim();
158                 JSONObject jsonObject = new JSONObject(jsonString);
159 
160                 // Check if we have gotten an error from the JSON Response..
161                 if (jsonString.contains("error")) {
162                     JSONObject obj = jsonObject.getJSONObject("error");
163                     String errorCode = obj.getString("code");
164                     String errorMessage = obj.getString("message");
165 
166                     // Display the error message in a pop-up infoBox.
167                     infoBox("ERROR: " + errorCode + "\n"
168                             + errorMessage, "An error has occurred");
169                 }
170                 // Retrieve personID from the JSON response
171                 String id = jsonObject.getString("personId");
172 
173                 // Check if the personId has a value
```

```
174                 if (!id.isEmpty())/*isBlank())*/ {
175                     personIdString = id;
176                     JSONObject personObject = new JSONObject(userData);
177 
178                     // Get person name from the jsonObject
179                     String name = personObject.getString("NAME");
180 
181                     // Get userdata from the jsonObject
182                     String data = personObject.getString("USERDATA");
183 
184                     // Show info of the created person in a pop-up box
185                     infoBox("Created a new Person: \n" + "Name: " + name
186                             + "\n Userdata: " + data + "\n Person ID: " + id
187                             + "\n Group: " + groupId, "Created a new Person");
188                 }
189             }
190         } catch (Exception e) {
191             // Display error message.
192             System.out.println(e.getMessage());
193         }
194     }
195 
196     /**
197      * Delete a person from a persongroup. Before deleting a person, also
198      * deletes every associated persistedFaceID to that person.
199      *
200      * @param nameToDelete name of the person to delete
201      * @param groupId id of group we want to delete the person from
202      */
203     public void deletePerson(String nameToDelete, String groupId) {
204 
205         // Get the personId related to the person name from a HashMap.
206         Object personIdObject = users.get(nameToDelete);
207 
208         String personIdToDelete = personIdObject.toString();
209 
210         // Check for persistedFaceIds and put them in an ArrayList (pfIdArray)
211         getPerson(groupId, nameToDelete);
212 
213         //Check if the persistedFaceIdArray is empty
214         if (!pfIdArray.isEmpty()) {
215 
216             // Create an Iterator to iterate over the pfidArray if it is not
empty
```

```
217             Iterator it = pfIdArray.iterator();
218             while (it.hasNext()) {
219                 String persistedFaceId = it.next().toString();
220 
221                 // Delete all found persistedFaceIds
222                 deletePersonFace(groupId, personIdToDelete, persistedFaceId);
223             }
224         } else {
225             // Display a message if pfidArray is empty
226             infoBox("Could not find any PersistedFaceIds to delete..",
227                     "An error has occurred");
228         }
229 
230         HttpClient httpclient = HttpClientBuilder.create().build();
231 
232         try {
233             URIBuilder builder = new URIBuilder(
234                     "https://northeurope.api.cognitive.microsoft.com/face/v1.0/"
235                     + "persongroups/" + groupId + "/persons/" +
personIdToDelete);
236 
237             builder.setParameter("personGroupId", groupId);
238             builder.setParameter("personId", personIdToDelete);
239 
240             URI uri = builder.build();
241             HttpDelete request = new HttpDelete(uri);
242 
243             // Request headers.
244             request.setHeader("Ocp-Apim-Subscription-Key", subscriptionKey);
245 
246             // Execute the REST API call and get the response entity.
247             HttpResponse response = httpclient.execute(request);
248             HttpEntity entity = response.getEntity();
249             if (entity != null) {
250 
251                 // Format and display the JSON response.
252                 String jsonString = EntityUtils.toString(entity).trim();
253 
254                 if (jsonString.isEmpty()) { //isBlank()) {
255 
256                     // Check if the JSON response is empty or contains
whitespaces
257                     if (jsonString.isEmpty())/*isBlank())*/ {
258                         // If jsonString is empty, display info about the
```

```
deleted
259                         // person in the pop-up box
260
261                         infoBox("Deleted Person: \n From group: " + groupId
262                                 + " \n Name: " + nameToDelete + "\n All "
263                                 + "associated PersistedFaceIDs are also
deleted",
264                                 "Deleted person");
265                     }
266                     // Check if the jsonString contains an error
267                     if (jsonString.contains("error")) {
268                         JSONObject jsonObject = new JSONObject(jsonString);
269                         JSONObject obj = jsonObject.getJSONObject("error");
270                         String errorCode = obj.getString("code");
271                         String errorMessage = obj.getString("message");
272
273                         // Display error message in a pop-up box
274                         infoBox("ERROR: " + errorCode + "\n"
275                                 + errorMessage, "An error has occurred");
276                     }
277                 }
278             }
279         } catch (Exception e) {
280             // Display error message.
281             System.out.println(e.getMessage());
282         }
283     }
284
285     /**
286      * Delete a persons "face". This method deletes a persistedFaceId associated
287      * to a persons personId.
288      *
289      * @param personGroupId the persongroupid from which the person, whose face
290      * we want to delete, belongs to.
291      * @param personId The persons personId whose face we want to delete
292      * @param persistedFaceId The "face" we want to delete.
293      */
294     public void deletePersonFace(String personGroupId, String personId,
295             String persistedFaceId) {
296
297         HttpClient httpclient = HttpClientBuilder.create().build();
298
299         try {
300             URIBuilder builder = new URIBuilder(
```

```
301                     "https://northeurope.api.cognitive.microsoft.com/face/v1.0/"
302                     + "persongroups/" + personGroupId + "/persons/"
303                     + personId + "/persistedFaces/" + persistedFaceId);
304
305             builder.setParameter("personGroupId", personGroupId);
306             builder.setParameter("personId", personId);
307             builder.setParameter("persistedFaceId", persistedFaceId);
308
309             URI uri = builder.build();
310             HttpDelete request = new HttpDelete(uri);
311
312             // Request headers.
313             request.setHeader("Ocp-Apim-Subscription-Key", subscriptionKey);
314
315             // Execute the REST API call and get the response entity.
316             HttpResponse response = httpclient.execute(request);
317             HttpEntity entity = response.getEntity();
318             if (entity != null) {
319
320                 // Format and display the JSON response.
321                 String jsonString = EntityUtils.toString(entity).trim();
322
323                 // Check if the jsonString contains an error message
324                 if (jsonString.contains("error")) {
325                     JSONObject jsonObject = new JSONObject(jsonString);
326                     JSONObject obj = jsonObject.getJSONObject("error");
327                     String errorCode = obj.getString("code");
328                     String errorMessage = obj.getString("message");
329
330                     // Display the error message in a pop-up box
331                     infoBox("ERROR: " + errorCode + "\n"
332                             + errorMessage, "An error has occurred");
333                 }
334             }
335         } catch (Exception e) {
336             // Display error message.
337             System.out.println(e.getMessage());
338         }
339     }
340
341     /**
342      * This method call sends a request to Microsoft Azure to get and list all
343      * persons in a personGroup. We save the response from azure in two ways.
344      * One is a HashMap with Name and PersonID. The other is an ArrayList with
```

```
345      * only names. This ArrayList is used to populate a JList.
346      *
347      * @param personGroupId the id of the person group we want to get persons
348      * from
349      */
350     public void listPersons(String personGroupId) {
351         // Check if the userName ArrayList is empty. If empty, move on.
352         // If not, clear it so it can be refilled with updated information.
353         if (!userName.isEmpty()) {
354             userName.clear();
355         }
356         // Check if the users HashMap is empty. If empty, move on.
357         // If not, clear it so it can be refilled with updated information.
358         if (!users.isEmpty()) {
359             users.clear();
360         }
361 
362         HttpClient httpclient = HttpClientBuilder.create().build();
363         try {
364             URIBuilder builder = new URIBuilder(
365                     "https://northeurope.api.cognitive.microsoft.com/face/v1.0/"
366                     + "persongroups/" + personGroupId + "/persons");
367 
368             URI uri = builder.build();
369             HttpGet request = new HttpGet(uri);
370             HttpPost request1 = new HttpPost(uri);
371 
372             // Request headers.
373             request.setHeader("Ocp-Apim-Subscription-Key", subscriptionKey);
374 
375             StringEntity reqEntity = new StringEntity(personGroupId);
376             request1.setEntity(reqEntity);
377 
378             // Execute the REST API call and get the response entity.
379             HttpResponse response = httpclient.execute(request);
380             HttpEntity entity = response.getEntity();
381             if (entity != null) {
382                 // Format and display the JSON response.
383                 String jsonString = EntityUtils.toString(entity).trim();
384 
385                 try {
386                     JSONArray jsonArray = new JSONArray(jsonString);
387 
388                     // Parse the jsonArray and retrieve information from the
```

```
389                     // JSON response, and add information to HashMap and
ArrayList
390                     for (int i = 0; i < jsonArray.length(); ++i) {
391                         JSONObject person = jsonArray.getJSONObject(i);
392                         String personId = person.getString("personId");
393                         String name = person.getString("name").toUpperCase();
394                         Object userdataObject = person.get("userData");
395                         String userData = userdataObject.toString();
396 
397                         // ArrayList with names
398                         userName.add(name);
399 
400                         // HashMap with names and associated personIds.
401                         users.put(name, personId);
402                     }
403                 } catch (JSONException e) {
404                     System.out.println(e.getMessage());
405                 }
406                 // Check if the jsonString contains an error message
407                 if (jsonString.contains("error")) {
408                     JSONObject jsonObject = new JSONObject(jsonString);
409                     JSONObject obj = jsonObject.getJSONObject("error");
410                     String errorCode = obj.getString("code");
411                     String errorMessage = obj.getString("message");
412 
413                     // Display the error message in a pop-up box
414                     infoBox("ERROR: " + errorCode + "\n"
415                             + errorMessage, "An error has occurred");
416                 }
417             }
418         } catch (Exception e) {
419             // Display error message.
420             System.out.println(e.getMessage());
421         }
422     }
423 
424     /**
425      * This method populates an ArrayList with persistedFaceIds belonging to a
426      * specific person. To avoid mixups with other persons and faceIds, make
427      * sure the ArrayList is empty before populating it again.
428      *
429      * @param personGroupId the id of the personGroup in which the person is
430      * @param personName name of the person we want to get persistedFaceIds from
431      */
```

```
432     public void getPerson(String personGroupId, String personName) {
433         String personId = " ";
434         // Check if the pfIdArray is empty, if not, clear it.
435         if (!pfIdArray.isEmpty()) {
436             pfIdArray.clear();
437         }
438         if (!personName.isEmpty()) {//isBlank()) {
439 
440             // Check that personName is not blank (empty or contain whitespace).
441             // If personName is not blank, get personId belonging to the
personName.
442             // Else, display an error message in a popupbox
443             if (!personName.isEmpty())/*isBlank())*/ {
444 
445                 personId = users.get(personName).toString();
446             } else {
447                 infoBox("The person you searched for could not be found...",
"Error: "
448                         + "Person not found");
449             }
450 
451             HttpClient httpClient = HttpClientBuilder.create().build();
452             try {
453                 URIBuilder builder = new URIBuilder(
454 
"https://northeurope.api.cognitive.microsoft.com/face/v1.0/"
455                         + "persongroups/" + personGroupId + "/persons/"
456                         + personId);
457                 URI uri = builder.build();
458                 HttpGet request = new HttpGet(uri);
459                 HttpPost request1 = new HttpPost(uri);
460 
461                 // Request headers.
462                 request.setHeader("Ocp-Apim-Subscription-Key", subscriptionKey);
463 
464                 StringEntity reqEntity = new StringEntity(personGroupId);
465                 request1.setEntity(reqEntity);
466 
467                 // Execute the REST API call and get the response entity.
468                 HttpResponse response = httpClient.execute(request);
469                 HttpEntity entity = response.getEntity();
470 
471                 if (entity != null) {
472                     // Format and display the JSON response.
```

```
473                     String jsonString = EntityUtils.toString(entity).trim();
474                     try {
475                         JSONObject jsonObject = new JSONObject(jsonString);
476                         String name = jsonObject.getString("name");
477                         JSONArray persistedFaceIds
478                                 = jsonObject.getJSONArray("persistedFaceIds");
479                         // Check if the JSON array is empty. If not, fill
pfIdArray
480                         // with persistedFaceIds from JSONArray
481                         if (!persistedFaceIds.isEmpty()) {
482                             for (int j = 0; j < persistedFaceIds.length(); ++j)
{
483                                 String pfId = persistedFaceIds.getString(j);
484                                 pfIdArray.add(pfId);
485                             }
486                         }
487                         if (name.isEmpty()) {//isBlank()) {
488                             // If the name is blank, print a short message.
489                             infoBox("Person was not found...", "Error");
490                         }
491                     } catch (JSONException e) {
492                         System.out.println(e.getMessage());
493                     }
494                 }
495             } catch (Exception e) {
496                 System.out.println(e.getMessage());
497             }
498         }
499     }
500
501     /**
502      * Popup box for showing messages to the user. This method was found on
503      * StackOverflow at the following link:
504      * https://stackoverflow.com/questions/7080205/popup-message-boxes
505      *
506      *
507      * @param message the message to show.
508      * @param title the title of the message and box
509      */
510     private static void infoBox(String message, String title) {
511         // null puts it in the center of the screen. Size is adjusted according
512         // to the size of the message.
513         JOptionPane.showMessageDialog(null, message, "InfoBox: " + title,
514                 JOptionPane.INFORMATION_MESSAGE);
```

```
515     }
516 }
```

```
 1
 2 package FaceAPI;
 3
 4
 5 import com.github.sarxos.webcam.Webcam;
 6 import com.github.sarxos.webcam.WebcamDiscoveryEvent;
 7 import com.github.sarxos.webcam.WebcamDiscoveryListener;
 8
 9
10 public class WebcamDiscoveryListenerExample implements WebcamDiscoveryListener {
11
12 public WebcamDiscoveryListenerExample() {
13 for (Webcam webcam : Webcam.getWebcams()) {
14 System.out.println("Webcam detected: " + webcam.getName());
15 }
16 Webcam.addDiscoveryListener(this);
17 System.out.println("\n\nPlease connect additional webcams, or disconnect already
connected ones. Listening for events...");
18 }
19
20 @Override
21 public void webcamFound(WebcamDiscoveryEvent event) {
22 System.out.format("Webcam connected: %s \n", event.getWebcam().getName());
23 }
24
25 @Override
26 public void webcamGone(WebcamDiscoveryEvent event) {
27 System.out.format("Webcam disconnected: %s \n", event.getWebcam().getName());
28 }
29
30 public static void main(String[] args) throws Throwable {
31 new WebcamDiscoveryListenerExample();
32 Thread.sleep(120000);
33 System.out.println("Bye!");
34 }
35 }
```

```
 1 package FaceAPI;
 2
 3 import com.github.sarxos.webcam.Webcam;
 4 import com.github.sarxos.webcam.WebcamPanel;
 5 import java.awt.BorderLayout;
 6 import java.awt.Color;
 7 import java.awt.Dimension;
 8 import java.awt.Image;
 9 import java.awt.event.ActionEvent;
10 import java.awt.event.ActionListener;
11 import javax.imageio.ImageIO;
12 import java.io.File;
13 import java.io.IOException;
14 import java.util.ArrayList;
15 import java.util.Arrays;
16 import java.util.Iterator;
17 import java.util.logging.Level;
18 import java.util.logging.Logger;
19 import javax.swing.BorderFactory;
20 import javax.swing.DefaultComboBoxModel;
21 import javax.swing.DefaultListModel;
22 import javax.swing.ImageIcon;
23 import javax.swing.JButton;
24 import javax.swing.JComboBox;
25 import javax.swing.JFileChooser;
26 import javax.swing.JFrame;
27 import javax.swing.JLabel;
28 import javax.swing.JList;
29 import javax.swing.JOptionPane;
30 import javax.swing.JPanel;
31 import javax.swing.JPasswordField;
32 import javax.swing.JTextField;
33 import javax.swing.border.LineBorder;
34 import org.json.JSONObject;
35
36
37 /**
38  * This class is responsible for creating and updating GUI elements and
39  * implement functionality behind all elements.
40  *
41  * Libraries from github https://github.com/sarxos/webcam-capture
42  *
43  * @author Adrian
44  */
```

```
45 public class WebCamFeedGUI extends JFrame {
46
47     private Webcam webcam;
48     private WebcamPanel panel;
49     private JButton captureButton;
50     private JFrame captureFrame;
51     private JLabel imageholder;
52     private JTextField nameField;
53     private JLabel nameLabel;
54     private JTextField emailField;
55     private JLabel emailLabel;
56     private JTextField numberField;
57     private JLabel numberLabel;
58     private JLabel errorLabel;
59     private JButton sendButton;
60     private JComboBox groupComboBox;
61     private JPanel wrapper;
62     private JFrame adminFrame;
63     private JButton adminButton;
64     private JFrame homeScreen;
65     private JButton addUserButton;
66     private JButton deleteUserButton;
67     private JButton deleteGroupButton;
68     private JButton listAllUsersButton;
69     private JButton listAllGroupsButton;
70     private JPanel adminPanel1;
71     private JPanel adminPanel2;
72     private JList allUsersList;
73     private JList allGroupsList;
74     private JLabel userLabel;
75     private JLabel groupLabel;
76     private JPanel buttonPanel;
77     private JPanel rightButtonPanel;
78     private JPanel leftButtonPanel;
79     private JFrame passwordWindow;
80     private JPanel passwordPanel;
81     private JButton password;
82     private JLabel passwordLabel;
83     private JPasswordField passwordField;
84     private JPanel groupComboBoxPanel;
85     private JButton retakeButton;
86     private JButton addUserAdminBtn;
87     private JButton addGroupAdminBtn;
88     private JTextField groupIdField;
```

```
 89     private JTextField groupNameField;
 90     private JLabel groupIdLabel;
 91     private JLabel groupNameLabel;
 92     private JButton creatGroupBtn;
 93     private JComboBox listComboBox;
 94     private JPanel listComboBoxPanel;
 95     private JButton refreshUsersBtn;
 96     private JLabel listCboxLabel;
 97     private JTextField searchUserField;
 98     private JTextField searchGroupField;
 99     private JButton searchUserBtn;
100     private JButton searchGroupBtn;
101     private JLabel searchNameLabel;
102     private JLabel searchGroupLabel;
103     private JLabel errorGroupName;
104     private JLabel errorGroupId;
105     private JButton chooseFileBtn;
106     private JPanel uploadedFileHolder;
107     private JTextField adminNameField;
108     private JTextField adminLastNameField;
109     private JTextField adminEmailField;
110     private JTextField adminNumberField;
111     private JLabel adminNameLabel;
112     private JLabel adminLastNameLabel;
113     private JLabel adminEmailLabel;
114     private JLabel adminNumberLabel;
115     private JButton confirmButton;
116     private JComboBox addUserComboBox;
117     private JPanel addUserCboxPanel;
118
119     private PersonGroupPerson pgp = new PersonGroupPerson();
120     private PersonGroup pg = new PersonGroup();
121     private DefaultComboBoxModel comboBoxModel = new DefaultComboBoxModel();
122     private DefaultComboBoxModel listComboBoxModel = new
DefaultComboBoxModel();
123     private DefaultListModel groupListModel = new DefaultListModel();
124     private DefaultListModel listModel = new DefaultListModel();
125     private volatile ArrayList<String> matchedNames = new ArrayList<>();
126     private volatile ArrayList<String> matchedGroups = new ArrayList<>();
127
128     String filePath;
129
130     /**
131      * The constructor of this class
```

```
132      * @throws IOException
133      */
134     public WebCamFeedGUI() throws IOException {
135         initComponents();
136     }
137 
138     private void initComponents() {
139         pg.listPersonGroup();
140 
141         /* ----------------------- Design images ----------------------- */
142         ImageIcon adminImg = new
ImageIcon("/Users/Adrian/Bachelor-2019/GUIAdrian/admin-kopi.png");
143 
144         ImageIcon animatImage = new
ImageIcon("/Users/Adrian/Bachelor-2019/GUI/avento2.gif");
145 
146         ImageIcon captureeImage = new
ImageIcon("/Users/Adrian/Bachelor-2019/GUIAdrian/capture-kopi.png");
147 
148         ImageIcon sendImage = new
ImageIcon("/Users/Adrian/Bachelor-2019/GUIAdrian/send-kopi.png");
149 
150         ImageIcon backgroundImage = new
ImageIcon("/Users/Adrian/Bachelor-2019/GUIAdrian/aventob-kopi.png");
151 
152         ImageIcon nameImage = new
ImageIcon("/Users/Adrian/Bachelor-2019/GUIAdrian/name-kopi.png");
153 
154         ImageIcon emailImage = new
ImageIcon("/Users/Adrian/Bachelor-2019/GUIAdrian/email-kopi.png");
155 
156         ImageIcon numberImage = new
ImageIcon("/Users/Adrian/Bachelor-2019/GUIAdrian/number-kopi.png");
157 
158         ImageIcon addUserImg = new
ImageIcon("/Users/Adrian/Bachelor-2019/GUIAdrian/ADDUSER-kopi.png");
159 
160         ImageIcon retakeImg = new
ImageIcon("/Users/Adrian/Bachelor-2019/GUIAdrian/retake.png");
161 
162         ImageIcon groupImg = new
ImageIcon("/Users/Adrian/Bachelor-2019/GUIAdrian/group.png");
163 
164         ImageIcon loginImage = new
```

```
ImageIcon("/Users/Adrian/Bachelor-2019/GUIAdrian/login.png");
 165
 166         ImageIcon passwordImage = new
ImageIcon("/Users/Adrian/Bachelor-2019/GUIAdrian/password.png");
 167
 168         // Setting up the camera and the size
 169         //webcam = Webcam.getWebcamByName("Logitech Webcam C930e
0x142000000046d0843");
 170         //webcam = Webcam.getWebcamByName("HD 720P Webcam 1");
 171         webcam = Webcam.getDefault();
 172         webcam.setViewSize(new Dimension(320, 240));
 173
 174         // Initialises the panel for the cameraFeed
 175         panel = new WebcamPanel(webcam);
 176         panel.setFPSDisplayed(false);
 177         panel.setDisplayDebugInfo(false);
 178         panel.setImageSizeDisplayed(false);
 179         panel.setMirrored(false);
 180         panel.setSize(320, 240);
 181         panel.setLocation(30, 60);
 182
 183         /* ---------------------- HOMESCREEN ---------------------- */
 184         // Initializing the buttons, panels and label for the homeScreen
 185         adminButton = new JButton(adminImg);
 186
 187         addUserButton = new JButton(addUserImg);
 188
 189         buttonPanel = new JPanel();
 190         buttonPanel.setBackground(new Color(1, 49, 63));
 191         buttonPanel.setForeground(new Color(1, 49, 63));
 192         buttonPanel.setPreferredSize(new Dimension(800, 100));
 193         buttonPanel.setLayout(new BorderLayout());
 194
 195         rightButtonPanel = new JPanel();
 196         rightButtonPanel.setBackground(new Color(1, 49, 63));
 197         rightButtonPanel.setForeground(new Color(1, 49, 63));
 198         rightButtonPanel.setLayout(new BorderLayout());
 199         rightButtonPanel.setPreferredSize(new Dimension(400, 100));
 200         rightButtonPanel.setLocation(0, 600);
 201
 202         leftButtonPanel = new JPanel();
 203         leftButtonPanel.setBackground(new Color(1, 49, 63));
 204         leftButtonPanel.setForeground(new Color(1, 49, 63));
 205         leftButtonPanel.setLayout(new BorderLayout());
```

```
206         leftButtonPanel.setPreferredSize(new Dimension(400, 100));
207         leftButtonPanel.setLocation(0, 600);
208 
209         rightButtonPanel.add(adminButton, BorderLayout.CENTER);
210         leftButtonPanel.add(addUserButton, BorderLayout.CENTER);
211 
212         buttonPanel.add(rightButtonPanel, BorderLayout.EAST);
213         buttonPanel.add(leftButtonPanel, BorderLayout.WEST);
214 
215         JLabel animationLabel = new JLabel(animatImage);
216 
217         // Initialise the homeScreen frame menu
218         homeScreen = new JFrame("Home");
219         homeScreen.setSize(800, 700);
220 
homeScreen.setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
221         homeScreen.setResizable(false);
222         homeScreen.setLocationRelativeTo(null);
223 
224         //Adding components to the homeScreen.
225         homeScreen.add(animationLabel, BorderLayout.CENTER);
226         homeScreen.add(buttonPanel, BorderLayout.SOUTH);
227         homeScreen.setVisible(true);
228 
229         /* ---------------------- ADD USER SCREEN ---------------------- */
230         //Initializing the Buttons for the captureFrame
231         captureButton = new JButton(captureeImage);
232         captureButton.setSize(200, 75);
233         captureButton.setLocation(100, 370);
234 
235         sendButton = new JButton(sendImage);
236         sendButton.setSize(200, 75);
237         sendButton.setLocation(470, 370);
238         sendButton.setEnabled(false);
239 
240         retakeButton = new JButton(retakeImg);
241         retakeButton.setSize(200, 75);
242         retakeButton.setLocation(100, 370);
243         retakeButton.setVisible(false);
244 
245         // Initialise the dropdown choice box for the groups and an label
246         groupComboBoxPanel = new JPanel();
247         groupComboBoxPanel.setSize(150, 50);
248         groupComboBoxPanel.setLocation(600, 100);
```

```
249         groupComboBoxPanel.setBackground(new Color(1, 49, 63));
250
251         groupComboBox = new JComboBox();
252         groupComboBox.setModel(comboBoxModel);
253         groupComboBox.setSelectedItem(null);
254         groupComboBox.setVisible(true);
255         groupComboBox.setSize(130, 50);
256         groupComboBox.setLocation(600, 100);
257         groupComboBox.addActionListener(groupComboBox);
258
259         groupLabel = new JLabel(groupImg);
260         groupLabel.setSize(125, 35);
261         groupLabel.setLocation(450, 97);
262
263         groupComboBoxPanel.add(groupComboBox);
264
265         JLabel backgroundLabel = new JLabel(backgroundImage);
266         wrapper = new JPanel();
267         wrapper.setBackground(new Color(1, 49, 63));
268         wrapper.setLocation(700, 100);
269         wrapper.setSize(100, 30);
270         wrapper.add(backgroundLabel, BorderLayout.CENTER);
271
272         // Initialise the nameField and nameLabel
273         nameField = new JTextField();
274         nameField.setSize(150, 30);
275         nameField.setLocation(600, 150);
276         nameField.setBorder(BorderFactory.createLineBorder(new Color(1, 49,
63)));
277
278         nameLabel = new JLabel(nameImage);
279         nameLabel.setSize(100, 35);
280         nameLabel.setLocation(450, 147);
281
282         // Initialise the emailField and emailLabel
283         emailField = new JTextField();
284         emailField.setSize(150, 30);
285         emailField.setLocation(600, 200);
286         emailField.setBorder(BorderFactory.createLineBorder(new Color(1, 49,
63)));
287
288         emailLabel = new JLabel(emailImage);
289         emailLabel.setSize(100, 35);
290         emailLabel.setLocation(450, 197);
```

```
291
292         // Initialise the numberField and numberLabel
293         numberField = new JTextField();
294         numberField.setSize(150, 30);
295         numberField.setLocation(600, 250);
296         numberField.setBorder(BorderFactory.createLineBorder(new Color(1, 49,
63)));
297
298         numberLabel = new JLabel(numberImage);
299         numberLabel.setSize(140, 35);
300         numberLabel.setLocation(450, 247);
301
302         // The imageHolder holds the image that is taken
303         imageholder = new JLabel();
304         imageholder.setSize(320, 240);
305         imageholder.setLocation(30, 60);
306         imageholder.setVisible(false);
307
308         // Initialise the errorLabel, will popup if the fields are empty
309         errorLabel = new JLabel();
310         errorLabel.setVisible(false);
311
312         // Initialise the frame
313         captureFrame = new JFrame("ADD USER");
314         captureFrame.setSize(800, 500);
315         captureFrame.setResizable(false);
316         captureFrame.setLocationRelativeTo(null);
317
318         // Adding components to the frame/window
319         captureFrame.add(captureButton);
320         captureFrame.add(panel);
321         captureFrame.add(nameField);
322         captureFrame.add(nameLabel);
323         captureFrame.add(emailField);
324         captureFrame.add(emailLabel);
325         captureFrame.add(numberField);
326         captureFrame.add(numberLabel);
327         captureFrame.add(errorLabel);
328         captureFrame.add(sendButton);
329         captureFrame.add(imageholder);
330         captureFrame.add(retakeButton);
331         captureFrame.add(groupLabel);
332         captureFrame.add(groupComboBoxPanel);
333         captureFrame.add(wrapper);
```

```
334
335         /* ---------------------- Password Screen ---------------------- */
336         // Initialize password screen
337         passwordWindow = new JFrame("Password");
338         passwordWindow.setSize(300, 300);
339         passwordWindow.setResizable(false);
340         passwordWindow.setLocationRelativeTo(null);
341
342         // Initialize panel for password frame
343         passwordPanel = new JPanel();
344         passwordPanel.setPreferredSize(new Dimension(300, 300));
345         passwordPanel.setBackground(new Color(1, 49, 63));
346
347         // Initialize button fro password frame
348         password = new JButton(loginImage);
349         password.setBackground(new Color(1, 49, 63));
350         password.setSize(300, 75);
351         password.setBorder(BorderFactory.createLineBorder(new Color(1, 49,
63)));
352
353         // Initialize password label
354         passwordLabel = new JLabel(passwordImage);
355
356         // Initalize password field
357         passwordField = new JPasswordField(10);
358
359         //Adding components to frame and panel
360         passwordWindow.add(passwordPanel);
361         passwordPanel.add(passwordLabel);
362         passwordPanel.add(passwordField);
363         passwordPanel.add(password);
364
365         //Sets the passwordbutton to defaultButton --> can use enter on
keyboard
366         passwordWindow.getRootPane().setDefaultButton(password);
367
368         /* ---------------------- Admin Screen ---------------------- */
369         //Initializing the buttons for the AdminFrame
370         deleteUserButton = new JButton();
371         deleteUserButton.setText("Delete User");
372         deleteUserButton.setSize(95, 40);
373         deleteUserButton.setLocation(90, 120);
374         deleteUserButton.setEnabled(false);
375         addUserAdminBtn = new JButton();
```

```
376         addUserAdminBtn.setText("Add user");
377         addUserAdminBtn.setSize(75, 40);
378         addUserAdminBtn.setLocation(5, 120);
379         addUserAdminBtn.setEnabled(false);
380 
381         deleteGroupButton = new JButton();
382         deleteGroupButton.setText("Delete Group");
383         deleteGroupButton.setSize(95, 40);
384         deleteGroupButton.setLocation(90, 300);
385         deleteGroupButton.setEnabled(false);
386         addGroupAdminBtn = new JButton();
387         addGroupAdminBtn.setText("Add Group");
388         addGroupAdminBtn.setSize(80, 40);
389         addGroupAdminBtn.setLocation(5, 300);
390         addGroupAdminBtn.setEnabled(false);
391 
392         listAllGroupsButton = new JButton();
393         listAllGroupsButton.setText("List Groups");
394         listAllGroupsButton.setSize(150, 50);
395         listAllGroupsButton.setLocation(15, 220);
396 
397         listAllUsersButton = new JButton();
398         listAllUsersButton.setText("Users");
399         listAllUsersButton.setSize(150, 50);
400         listAllUsersButton.setLocation(15, 50);
401 
402         // LISTS AND LABELS
403         allUsersList = new JList();
404         allUsersList.setVisible(false);
405         allUsersList.setBackground(Color.WHITE);
406         allUsersList.setSize(330, 300);
407         allUsersList.setLocation(230, 190);
408 
409         userLabel = new JLabel();
410         userLabel.setText("Users:");
411         userLabel.setSize(75, 30);
412         userLabel.setLocation(230, 150);
413         userLabel.setVisible(false);
414 
415         groupLabel = new JLabel();
416         groupLabel.setText("Groups:");
417         groupLabel.setSize(75, 50);
418         groupLabel.setLocation(230, 50);
419         groupLabel.setVisible(false);
```

```
420         allGroupsList = new JList();
421         allGroupsList.setVisible(false);
422         allGroupsList.setBackground(Color.WHITE);
423         allGroupsList.setSize(330, 300);
424         allGroupsList.setLocation(230, 100);
425
426         // Initializing and creating the components for the create group frame
427         groupNameField = new JTextField();
428         groupNameField.setSize(200, 30);
429         groupNameField.setLocation(230, 130);
430         groupNameField.setVisible(false);
431
432         groupNameLabel = new JLabel();
433         groupNameLabel.setText("Group Name: (Only numbers and letters)");
434         groupNameLabel.setSize(350, 30);
435         groupNameLabel.setLocation(230, 95);
436         groupNameLabel.setVisible(false);
437
438         groupIdField = new JTextField();
439         groupIdField.setSize(200, 30);
440         groupIdField.setLocation(230, 235);
441         groupIdField.setVisible(false);
442
443         groupIdLabel = new JLabel();
444         groupIdLabel.setText("Group ID: (Only numbers and SMALL letters)");
445         groupIdLabel.setSize(350, 30);
446         groupIdLabel.setLocation(230, 200);
447         groupIdLabel.setVisible(false);
448
449         creatGroupBtn = new JButton();
450         creatGroupBtn.setText("Create Group");
451         creatGroupBtn.setSize(100, 35);
452         creatGroupBtn.setLocation(230, 310);
453         creatGroupBtn.setVisible(false);
454
455         errorGroupName = new JLabel();
456         errorGroupName.setSize(250, 30);
457         errorGroupName.setLocation(230, 160);
458         errorGroupName.setText("You can only use numbers and letters");
459         errorGroupName.setForeground(Color.red);
460         errorGroupName.setVisible(false);
461
462         errorGroupId = new JLabel();
463         errorGroupId.setSize(350, 30);
```

```
464         errorGroupId.setLocation(230, 270);
465         errorGroupId.setText("You can only use numbers and small letters");
466         errorGroupId.setForeground(Color.red);
467         errorGroupId.setVisible(false);
468
469         // Initialise the dropdown choice box for the select group, label and
button
470         // for the Users frame
471         listComboBoxPanel = new JPanel();
472         listComboBoxPanel.setSize(140, 35);
473         listComboBoxPanel.setLocation(230, 30);
474         listComboBoxPanel.setBackground(new Color(1, 49, 63));
475         listComboBoxPanel.setVisible(false);
476
477         listComboBox = new JComboBox();
478         listComboBox.setModel(listComboBoxModel);
479
480         listComboBox.setSelectedItem(null);
481         listComboBox.setVisible(true);
482         listComboBox.setSize(130, 35);
483         listComboBox.setLocation(230, 30);
484         listComboBox.addActionListener(listComboBox);
485
486         listComboBoxPanel.add(listComboBox);
487
488         listCboxLabel = new JLabel();
489         listCboxLabel.setText("Select Group: ");
490         listCboxLabel.setSize(100, 30);
491         listCboxLabel.setLocation(230, 1);
492         listCboxLabel.setVisible(false);
493
494         refreshUsersBtn = new JButton();
495         refreshUsersBtn.setText("Refresh");
496         refreshUsersBtn.setSize(75, 30);
497         refreshUsersBtn.setLocation(400, 30);
498         refreshUsersBtn.setVisible(false);
499
500         // Creating searchfield and buttons for both the users and groups
501         searchUserField = new JTextField();
502         searchUserField.setSize(130, 30);
503         searchUserField.setLocation(230, 95);
504         searchUserField.setVisible(false);
505         searchUserBtn = new JButton();
506         searchUserBtn.setText("Search");
```

```
507         searchUserBtn.setSize(75, 30);
508         searchUserBtn.setLocation(400, 95);
509         searchUserBtn.setVisible(false);
510         searchNameLabel = new JLabel();
511         searchNameLabel.setText("Search By Name:");
512         searchNameLabel.setSize(150, 30);
513         searchNameLabel.setLocation(230, 65);
514         searchNameLabel.setVisible(false);
515
516         searchGroupField = new JTextField();
517         searchGroupField.setSize(130, 30);
518         searchGroupField.setLocation(230, 35);
519         searchGroupField.setVisible(false);
520         searchGroupBtn = new JButton();
521         searchGroupBtn.setText("Search");
522         searchGroupBtn.setSize(75, 30);
523         searchGroupBtn.setLocation(400, 35);
524         searchGroupBtn.setVisible(false);
525         searchGroupLabel = new JLabel();
526         searchGroupLabel.setText("Search Groups:");
527         searchGroupLabel.setSize(100, 30);
528         searchGroupLabel.setLocation(230, 5);
529         searchGroupLabel.setVisible(false);
530
531         // Creating components to the Add user screen in the admin window
532         chooseFileBtn = new JButton();
533         chooseFileBtn.setText("Choose file");
534         chooseFileBtn.setSize(100, 30);
535         chooseFileBtn.setLocation(230, 200);
536         chooseFileBtn.setVisible(false);
537
538         uploadedFileHolder = new JPanel();
539         uploadedFileHolder.setSize(320, 240);
540         uploadedFileHolder.setLocation(230, 240);
541         uploadedFileHolder.setBackground(new Color(1, 49, 63));
542         uploadedFileHolder.setVisible(false);
543
544         adminNameField = new JTextField();
545         adminNameField.setSize(150, 30);
546         adminNameField.setLocation(335, 40);
547         adminNameField.setBorder(BorderFactory.createLineBorder(new Color(1,
49, 63)));
548         adminNameField.setVisible(false);
549
```

```
 550         adminNameLabel = new JLabel();
 551         adminNameLabel.setText("Name:");
 552         adminNameLabel.setSize(100, 30);
 553         adminNameLabel.setLocation(230, 40);
 554         adminNameLabel.setVisible(false);
 555 
 556         adminLastNameField = new JTextField();
 557         adminLastNameField.setSize(150, 30);
 558         adminLastNameField.setLocation(335, 75);
 559         adminLastNameField.setBorder(BorderFactory.createLineBorder(new
Color(1, 49, 63)));
 560         adminLastNameField.setVisible(false);
 561 
 562         adminLastNameLabel = new JLabel();
 563         adminLastNameLabel.setText("Last Name:");
 564         adminLastNameLabel.setSize(100, 30);
 565         adminLastNameLabel.setLocation(230, 75);
 566         adminLastNameLabel.setVisible(false);
 567 
 568         adminEmailField = new JTextField();
 569         adminEmailField.setSize(150, 30);
 570         adminEmailField.setLocation(335, 110);
 571         adminEmailField.setBorder(BorderFactory.createLineBorder(new Color(1,
49, 63)));
 572         adminEmailField.setVisible(false);
 573 
 574         adminEmailLabel = new JLabel();
 575         adminEmailLabel.setText("Email:");
 576         adminEmailLabel.setSize(100, 35);
 577         adminEmailLabel.setLocation(230, 110);
 578         adminEmailLabel.setVisible(false);
 579 
 580         adminNumberField = new JTextField();
 581         adminNumberField.setSize(150, 30);
 582         adminNumberField.setLocation(335, 145);
 583         adminNumberField.setBorder(BorderFactory.createLineBorder(new Color(1,
49, 63)));
 584         adminNumberField.setVisible(false);
 585 
 586         adminNumberLabel = new JLabel();
 587         adminNumberLabel.setText("Number:");
 588         adminNumberLabel.setSize(100, 30);
 589         adminNumberLabel.setLocation(230, 145);
 590         adminNumberLabel.setVisible(false);
```

```
591
592         confirmButton = new JButton();
593         confirmButton.setText("CONFIRM");
594         confirmButton.setSize(75, 40);
595         confirmButton.setLocation(500, 525);
596         confirmButton.setVisible(false);
597
598         addUserComboBox = new JComboBox();
599         addUserComboBox.setModel(listComboBoxModel);
600         addUserComboBox.setSelectedItem(null);
601         addUserComboBox.setVisible(true);
602         addUserComboBox.setSize(130, 35);
603         addUserComboBox.setLocation(230, 0);
604         addUserComboBox.addActionListener(addUserComboBox);
605
606         addUserCboxPanel = new JPanel();
607         addUserCboxPanel.setSize(140, 35);
608         addUserCboxPanel.setLocation(230, 0);
609         addUserCboxPanel.setBackground(new Color(1, 49, 63));
610         addUserCboxPanel.setVisible(false);
611         addUserCboxPanel.add(addUserComboBox);
612
613         //Adding colorPanels to the adminpanel
614         adminPanel1 = new JPanel();
615         adminPanel1.setBackground(Color.black);
616         adminPanel1.setPreferredSize(new Dimension(200, 400));
617         adminPanel2 = new JPanel();
618         adminPanel2.setBackground(Color.lightGray);
619         adminPanel2.setPreferredSize(new Dimension(400, 400));
620         adminPanel2.setLayout(new BorderLayout());
621
622         // A single frame for the Admin-part. The admin can delete
persons/users,
623         // make groups and see whos in the different groups in this frame.
624         adminFrame = new JFrame("Admin");
625         adminFrame.setSize(600, 600);
626         adminFrame.setResizable(true);
627         adminFrame.setLocationRelativeTo(null);
628
629         // Add components to the adminFrame
630         adminFrame.add(deleteUserButton);
631         adminFrame.add(deleteGroupButton);
632         adminFrame.add(listAllGroupsButton);
633         adminFrame.add(listAllUsersButton);
```

```
634         adminFrame.add(allUsersList);
635         adminFrame.add(allGroupsList);
636         adminFrame.add(userLabel);
637         adminFrame.add(groupLabel);
638         adminFrame.add(addUserAdminBtn);
639         adminFrame.add(addGroupAdminBtn);
640         adminFrame.add(groupNameField);
641         adminFrame.add(groupNameLabel);
642         adminFrame.add(groupIdField);
643         adminFrame.add(groupIdLabel);
644         adminFrame.add(creatGroupBtn);
645         adminFrame.add(listComboBoxPanel);
646         adminFrame.add(listCboxLabel);
647         adminFrame.add(refreshUsersBtn);
648         adminFrame.add(searchUserBtn);
649         adminFrame.add(searchUserField);
650         adminFrame.add(searchGroupBtn);
651         adminFrame.add(searchGroupField);
652         adminFrame.add(searchNameLabel);
653         adminFrame.add(searchGroupLabel);
654         adminFrame.add(errorGroupId);
655         adminFrame.add(errorGroupName);
656         adminFrame.add(chooseFileBtn);
657         adminFrame.add(uploadedFileHolder);
658         adminFrame.add(adminNameField);
659         adminFrame.add(adminLastNameField);
660         adminFrame.add(adminEmailField);
661         adminFrame.add(adminNumberField);
662         adminFrame.add(adminNameLabel);
663         adminFrame.add(adminLastNameLabel);
664         adminFrame.add(adminEmailLabel);
665         adminFrame.add(adminNumberLabel);
666         adminFrame.add(confirmButton);
667         adminFrame.add(addUserCboxPanel);
668         adminFrame.getContentPane().add(adminPanel1, BorderLayout.CENTER);
669         adminFrame.getContentPane().add(adminPanel2, BorderLayout.LINE_END);
670
671
672
673         /*// Finding the supported dimensions for the camera
674         for (Dimension supportedSize : webcam.getViewSizes()) {
675             System.out.println(supportedSize.toString());
676         }*/
677
```

```
 678         // Adding actionListeners for the buttons, waiting for the buttons to
be
 679         // pressed .
 680         // When buttons pressed, the actionPerformed event sends them to their
own
 681         // method who gives them tasks to do.
 682         captureButton.addActionListener(new java.awt.event.ActionListener() {
 683             public void actionPerformed(ActionEvent e) {
 684                 sendButton.setEnabled(true);
 685                 captureBtnActionPerformed(e);
 686 
 687             }
 688         });
 689         // Checks if the Textfields in the add user screen is filled out
correctly.
 690         // If the textFields are correctly filled out, it will go to its own
method with tasks.
 691         // If not an errorLabel will show
 692         sendButton.addActionListener(new java.awt.event.ActionListener() {
 693             public void actionPerformed(ActionEvent e) {
 694                 if (!addUserTextValid()) {
 695                     errorLabel.setSize(250, 30);
 696                     errorLabel.setLocation(490, 300);
 697                     errorLabel.setText("You must fill out all the fields
correctly");
 698                     errorLabel.setForeground(Color.red);
 699                     errorLabel.setVisible(true);
 700                     sendButton.setSize(200, 75);
 701                     sendButton.setLocation(470, 370);
 702                 } else {
 703                     errorLabel.setVisible(false);
 704                     sendBtnActionPerformed(e);
 705                 }
 706             }
 707         });
 708         adminButton.addActionListener(new java.awt.event.ActionListener() {
 709             public void actionPerformed(ActionEvent e) {
 710                 adminBtnActionPerformed(e);
 711             }
 712         });
 713         addUserButton.addActionListener(new java.awt.event.ActionListener() {
 714             public void actionPerformed(ActionEvent e) {
 715                 addUserBtnActionPerformed(e);
 716             }
```

```
 717         });
 718         deleteUserButton.addActionListener(new java.awt.event.ActionListener()
{
 719             public void actionPerformed(ActionEvent e) {
 720                 deleteUserBtnActionPerformed(e);
 721             }
 722         });
 723         deleteGroupButton.addActionListener(new java.awt.event.ActionListener()
{
 724             public void actionPerformed(ActionEvent e) {
 725                 deleteGroupBtnActionPerformed(e);
 726             }
 727         });
 728         listAllGroupsButton.addActionListener(new
java.awt.event.ActionListener() {
 729             public void actionPerformed(ActionEvent e) {
 730                 listAllGroupsBtnActionPerformed(e);
 731             }
 732         });
 733         listAllUsersButton.addActionListener(new
java.awt.event.ActionListener() {
 734             public void actionPerformed(ActionEvent e) {
 735                 listAllUsersBtnActionPerformed(e);
 736             }
 737         });
 738         password.addActionListener(new java.awt.event.ActionListener() {
 739             public void actionPerformed(ActionEvent e) {
 740                 passwordBtnActionPerformed(e);
 741             }
 742         });
 743         retakeButton.addActionListener(new java.awt.event.ActionListener() {
 744             public void actionPerformed(ActionEvent e) {
 745                 retakeBtnActionPerformed(e);
 746             }
 747         });
 748         addUserAdminBtn.addActionListener(new java.awt.event.ActionListener() {
 749             public void actionPerformed(ActionEvent e) {
 750                 addUserAdminBtnActionPerformed(e);
 751             }
 752         });
 753         addGroupAdminBtn.addActionListener(new java.awt.event.ActionListener()
{
 754             public void actionPerformed(ActionEvent e) {
 755                 addGroupBtnActionPerformed(e);
```

```
 756             }
 757         });
 758         // Checks if the Textfields in the admin Add-group screen is filled out
correctly.
 759         // If the textFields are correctly filled out, it will go to its own
method with tasks.
 760         // If not one or two errorLabels will show
 761         creatGroupBtn.addActionListener(new java.awt.event.ActionListener() {
 762             public void actionPerformed(ActionEvent e) {
 763 
 764                 if (!creatGroupTextValid()) {
 765                     errorGroupId.setVisible(true);
 766                     errorGroupName.setVisible(true);
 767 
 768                 } else {
 769                     errorGroupId.setVisible(false);
 770                     errorGroupName.setVisible(false);
 771                     createGroupBtnActionPerformed(e);
 772 
 773                 }
 774             }
 775         });
 776         refreshUsersBtn.addActionListener(new java.awt.event.ActionListener() {
 777 
 778             public void actionPerformed(ActionEvent e) {
 779                 refreshUsersBtnActionPerformed(e);
 780             }
 781         });
 782         searchUserBtn.addActionListener(new java.awt.event.ActionListener() {
 783 
 784             public void actionPerformed(ActionEvent e) {
 785                 searchUserBtnActionPerformed(e);
 786             }
 787         });
 788         searchGroupBtn.addActionListener(new java.awt.event.ActionListener() {
 789 
 790             public void actionPerformed(ActionEvent e) {
 791                 searchGroupBtnActionPerformed(e);
 792             }
 793         });
 794         chooseFileBtn.addActionListener(new ActionListener() {
 795             public void actionPerformed(ActionEvent e) {
 796                 try {
 797                     chooseFileBtnActionPerformed(e);
```

```
 798                 } catch (IOException ex) {
 799 
Logger.getLogger(WebCamFeedGUI.class.getName()).log(Level.SEVERE, null, ex);
 800                 }
 801             }
 802         });
 803         confirmButton.addActionListener(new java.awt.event.ActionListener() {
 804 
 805             public void actionPerformed(ActionEvent e) {
 806                 confirmBtnActionPerformed(e);
 807             }
 808         });
 809     }
 810 
 811     /**
 812      * When the confirm button is pressed the imagefile that is choosen will be
the face to the user
 813      * that is getting added. If the face is equal to a face that already exist
in the system, the face
 814      * will be added to the existing person. If not a new person will be added
in a specified group.
 815      * And then it will train the person group.
 816      * @param e this is the ActionEvent
 817      */
 818     public void confirmBtnActionPerformed(java.awt.event.ActionEvent e) {
 819         // file ist the filepath to the picture from filechooser
 820         File file = new File(filePath);
 821 
 822         Azure_GUI fdGui = new Azure_GUI();
 823         String groupIdString = getGroupId(listComboBox);
 824         fdGui.detectFace(file, groupIdString);
 825         boolean identify = fdGui.identify();
 826         if (identify == true) {
 827             pgp.addFaceToPerson(file, fdGui.id, groupIdString);
 828         } else {
 829             String cPerson = "{\"name\":" + "\"" + adminNameField.getText() +
"\",\"userdata\":" + "\"" + adminNumberField.getText() + "\"" + "}";
 830 
 831             pgp.createPersonToPg(cPerson, groupIdString);
 832             pgp.addFaceToPerson(file, pgp.personIdString, groupIdString);
 833             pg.trainPersonGroup(groupIdString);
 834         }
 835     }
 836 
```

```
 837     /**
 838      * When the chooseFile button is pressed it will remove everything that is
in the file holder before
 839      * a new file is choosen. The file that is choosen will be an ImageIcon,
this ImageIcon get scaled and
 840      * displayed in the file holder. The path to the file is used in the
confirm button Action performed.
 841      * @param e this is the ActionEvent
 842      * @throws IOException
 843      */
 844     public void chooseFileBtnActionPerformed(java.awt.event.ActionEvent e)
throws IOException {
 845         int imageWidth = 240;
 846         int imageHeigth = 320;
 847         uploadedFileHolder.removeAll();
 848         JFileChooser fc = new JFileChooser();
 849         int result = fc.showOpenDialog(null);
 850         if (result == JFileChooser.APPROVE_OPTION) {
 851             File file = fc.getSelectedFile();
 852             filePath = file.getAbsolutePath();
 853             System.out.println(filePath);
 854             ImageIcon icon = new ImageIcon(filePath);
 855 
 856             Dimension scaledImageDimension;
 857             Image scaledImage;
 858             if (icon.getIconHeight() > icon.getIconWidth()) {
 859                 scaledImageDimension = getScaledDimension(new
Dimension(icon.getIconHeight(), icon.getIconWidth()), new Dimension(imageHeigth,
imageWidth));
 860                 scaledImage =
icon.getImage().getScaledInstance(scaledImageDimension.height,
scaledImageDimension.width, Image.SCALE_DEFAULT);
 861             } else {
 862                 scaledImageDimension = getScaledDimension(new
Dimension(icon.getIconWidth(), icon.getIconHeight()), new Dimension(imageWidth,
imageHeigth));
 863                 scaledImage =
icon.getImage().getScaledInstance(scaledImageDimension.width,
scaledImageDimension.height, Image.SCALE_DEFAULT);
 864             }
 865 
 866             JLabel image = new JLabel("", new ImageIcon(scaledImage),
JLabel.CENTER);
 867             uploadedFileHolder.add(image, BorderLayout.CENTER);
```

```
 868             uploadedFileHolder.revalidate();
 869             uploadedFileHolder.repaint();
 870 
 871         }
 872     }
 873 
 874     /**
 875      * This method is scaling the inputImage from the fileChooser, this is used
so
 876      * that the users can see if it is the correct picture that they have
choosen.
 877      * This method is from
https://stackoverflow.com/questions/10245220/java-image-resize-maintain-aspect-ratio
 878      * @param imgSize this is the size of the inputimage
 879      * @param boundary this is the boundary, max width and height.
 880      * @return new Dimension This returns the new dimension of the inputimage
from the file chooser
 881     */
 882     public static Dimension getScaledDimension(Dimension imgSize, Dimension
boundary) {
 883 
 884         int original_width = imgSize.width;
 885         int original_height = imgSize.height;
 886         int bound_width = boundary.width;
 887         int bound_height = boundary.height;
 888         int new_width = original_width;
 889         int new_height = original_height;
 890 
 891         // first check if we need to scale width
 892         if (original_width > bound_width) {
 893             //scale width to fit
 894             new_width = bound_width;
 895             //scale height to maintain aspect ratio
 896             new_height = (new_width * original_height) / original_width;
 897         }
 898         // then check if we need to scale even with the new height
 899         if (new_height > bound_height) {
 900             //scale height to fit instead
 901             new_height = bound_height;
 902             //scale width to maintain aspect ratio
 903             new_width = (new_height * original_width) / original_height;
 904         }
 905         return new Dimension(new_width, new_height);
 906     }
```

```
907
908     /**
909      * A button that is used to gain access to addUser and delete users
methods.
910      * Sets components enabled or and visible with a boolean.
911      * @param e this is the ActionEvent
912      */
913     public void addUserAdminBtnActionPerformed(java.awt.event.ActionEvent e) {
914         deleteUserButton.setEnabled(false);
915         refreshUsersBtn.setVisible(false);
916         searchNameLabel.setVisible(false);
917         searchUserBtn.setVisible(false);
918         searchUserField.setVisible(false);
919         allUsersList.setVisible(false);
920         userLabel.setVisible(false);
921         listComboBoxPanel.setVisible(false);
922         listCboxLabel.setVisible(false);
923         creatGroupBtn.setVisible(false);
924         chooseFileBtn.setVisible(true);
925         uploadedFileHolder.setVisible(true);
926         adminNameField.setVisible(true);
927         adminLastNameField.setVisible(true);
928         adminNumberField.setVisible(true);
929         adminEmailField.setVisible(true);
930         adminNameLabel.setVisible(true);
931         adminLastNameLabel.setVisible(true);
932         adminNumberLabel.setVisible(true);
933         adminEmailLabel.setVisible(true);
934         confirmButton.setVisible(true);
935         allGroupsList.setVisible(false);
936         groupIdField.setVisible(false);
937         groupNameField.setVisible(false);
938         groupIdLabel.setVisible(false);
939         groupNameLabel.setVisible(false);
940         creatGroupBtn.setVisible(false);
941         addUserCboxPanel.setVisible(true);
942     }
943
944     /**
945      * Serch for groups by groupName, do not have to write the whole name
946      * @param e this is the ActionEvent
947      */
948     public void searchGroupBtnActionPerformed(java.awt.event.ActionEvent e) {
949         // Remove all elements in the list, to make room for groups that match
```

```
950         // the search.
951         groupListModel.removeAllElements();
952         if (!matchedGroups.isEmpty()) {
953             matchedGroups.clear();
954         }
955         pg.listPersonGroup();
956         String wantedGroup = searchGroupField.getText();
957         if (!wantedGroup.isEmpty()) {//isBlank()) {
958             Iterator it = pg.groupArray.iterator();
959             while (it.hasNext()) {
960                 String groupString = it.next().toString();
961                 if (groupString.contains(wantedGroup)) {
962                     matchedGroups.add(groupString);
963                 }
964             }
965             for (int i = 0; i < matchedGroups.size(); ++i) {
966                 groupListModel.addElement(matchedGroups.get(i));
967             }
968             allGroupsList.setModel(groupListModel);
969         }
970     }
971 
972     /**
973      * Search for users by the users name, do not have to write the whole name
974      * @param e this is the ActionEvent
975      */
976     public void searchUserBtnActionPerformed(java.awt.event.ActionEvent e) {
977         // Remove all elements in the list, to make room for names that match
978         // the search
979         listModel.removeAllElements();
980         if (!matchedNames.isEmpty()) {
981             matchedNames.clear();
982         }
983         Object chosenGroup = listComboBox.getSelectedItem();
984         Object groupIdObject = pg.groups.get(chosenGroup);
985         String groupIdString = groupIdObject.toString();
986         pgp.listPersons(groupIdString);
987         String searchname = searchUserField.getText().toUpperCase();
988 
989         if (!searchname.isEmpty()) { //isBlank()) {
990             Iterator it = pgp.userName.iterator();
991             while (it.hasNext()) {
992                 String nameString = it.next().toString().toUpperCase();
993                 if (nameString.contains(searchname)) {
```

```
 994                     matchedNames.add(nameString);
 995                 }
 996             }
 997             for (int i = 0; i < matchedNames.size(); ++i) {
 998                 listModel.addElement(matchedNames.get(i));
 999             }
1000             allUsersList.setModel(listModel);
1001         }
1002         if (matchedNames.isEmpty()) {
1003             infoBox("The person you searched for could not be found...",
"Error: "
1004                     + "Person not found");
1005 
1006         } else if (searchname.isEmpty()) {//isBlank()){
1007             infoBox("Searchfield is empty...", "Error");
1008         }
1009     }
1010 
1011     /**
1012      * The refreshUsersBtn is used to refresh list of users, if you switch
between
1013      * groups you have to refresh to see whos in the group.
1014      * @param e this is the ActionEvent
1015      */
1016     public void refreshUsersBtnActionPerformed(java.awt.event.ActionEvent e) {
1017 
1018         Object chosenGroup = listComboBox.getSelectedItem();
1019         Object groupIdObject = pg.groups.get(chosenGroup);
1020         String groupIdString = groupIdObject.toString();
1021         pgp.listPersons(groupIdString);
1022         listModel.removeAllElements();
1023 
1024         for (int i = 0; i < pgp.userName.size(); i++) {
1025             listModel.addElement(pgp.userName.get(i));
1026         }
1027         allUsersList.setModel(listModel);
1028         searchUserField.setText("");
1029 
1030     }
1031 
1032     /**
1033      * This is the method for adding groups, it takes the choosen groupId
groupName
1034      * from the textfields and creates a new person group.
```

```
1035      * @param e this is the ActionEvent
1036      */
1037
1038     public void createGroupBtnActionPerformed(java.awt.event.ActionEvent e) {
1039         String groupId = groupIdField.getText();
1040         String groupName = groupNameField.getText();
1041
1042         pg.createPersonGroup(groupId, groupName);
1043
1044     }
1045
1046     /**
1047      * The addGroupButton is used to remove the components that not belong, and
1048      * insert the components that belong in this screen.
1049      * @param e this is the ActionEvent
1050      */
1051     public void addGroupBtnActionPerformed(java.awt.event.ActionEvent e) {
1052         allGroupsList.setVisible(false);
1053         groupIdField.setVisible(true);
1054         groupNameField.setVisible(true);
1055         groupIdLabel.setVisible(true);
1056         groupNameLabel.setVisible(true);
1057         creatGroupBtn.setVisible(true);
1058         searchGroupBtn.setVisible(false);
1059         searchGroupField.setVisible(false);
1060         searchGroupLabel.setVisible(false);
1061         groupLabel.setVisible(false);
1062         deleteGroupButton.setEnabled(false);
1063         chooseFileBtn.setVisible(false);
1064         uploadedFileHolder.setVisible(false);
1065         adminNameField.setVisible(false);
1066         adminLastNameField.setVisible(false);
1067         adminNumberField.setVisible(false);
1068         adminEmailField.setVisible(false);
1069         adminNameLabel.setVisible(false);
1070         adminLastNameLabel.setVisible(false);
1071         adminNumberLabel.setVisible(false);
1072         adminEmailLabel.setVisible(false);
1073         confirmButton.setVisible(false);
1074     }
1075
1076     /**
1077      * This method checks if the password is correct, if it is correct the
adminFrame
```

```
1078      * will get visible. If not the border around the password field will be
red and
1079      * set to empty.
1080      * @param e this is the ActionEvent
1081      */
1082     public void passwordBtnActionPerformed(java.awt.event.ActionEvent e) {
1083         boolean password = actionPerformed1(e);
1084         if (password == true) {
1085             adminFrame.setVisible(true);
1086             passwordWindow.setVisible(false);
1087         } else {
1088             passwordField.setBorder(BorderFactory.createLineBorder(Color.red));
1089             passwordField.setText("");
1090         }
1091     }
1092
1093     /**
1094      * Listing all the users in the choosen group
1095      * @param e this is the ActionEvent
1096      */
1097     public void listAllUsersBtnActionPerformed(java.awt.event.ActionEvent e) {
1098         listComboBoxModel.removeAllElements();
1099         pg.listPersonGroup();
1100
1101         for (int i = 0; i < pg.groupArray.size(); i++) {
1102             listComboBoxModel.addElement(pg.groupArray.get(i));
1103         }
1104         allUsersList.setVisible(true);
1105         allGroupsList.setVisible(false);
1106         userLabel.setVisible(true);
1107         groupLabel.setVisible(false);
1108         deleteGroupButton.setEnabled(false);
1109         addGroupAdminBtn.setEnabled(false);
1110         deleteUserButton.setEnabled(true);
1111         addUserAdminBtn.setEnabled(true);
1112         listComboBoxPanel.setVisible(true);
1113         listCboxLabel.setVisible(true);
1114         refreshUsersBtn.setVisible(true);
1115         searchUserBtn.setVisible(true);
1116         searchUserField.setVisible(true);
1117         searchGroupBtn.setVisible(false);
1118         searchGroupField.setVisible(false);
1119         searchNameLabel.setVisible(true);
1120         searchGroupLabel.setVisible(false);
```

```
1121         groupNameLabel.setVisible(false);
1122         groupIdLabel.setVisible(false);
1123         groupNameField.setVisible(false);
1124         groupIdField.setVisible(false);
1125         chooseFileBtn.setVisible(false);
1126         uploadedFileHolder.setVisible(false);
1127         adminNameField.setVisible(false);
1128         adminLastNameField.setVisible(false);
1129         adminNumberField.setVisible(false);
1130         adminEmailField.setVisible(false);
1131         adminNameLabel.setVisible(false);
1132         adminLastNameLabel.setVisible(false);
1133         adminNumberLabel.setVisible(false);
1134         adminEmailLabel.setVisible(false);
1135         confirmButton.setVisible(false);
1136         errorGroupId.setVisible(false);
1137         errorGroupName.setVisible(false);
1138         addUserCboxPanel.setVisible(false);
1139     }
1140 
1141     /**
1142      * Lists all the groups
1143      * @param e this is the ActionEvent
1144      */
1145     public void listAllGroupsBtnActionPerformed(java.awt.event.ActionEvent e) {
1146         searchGroupField.setText("");
1147         pg.listPersonGroup();
1148         groupListModel.removeAllElements();
1149 
1150         for (int i = 0; i < pg.groupArray.size(); ++i) {
1151             groupListModel.addElement(pg.groupArray.get(i));
1152         }
1153         allGroupsList.setModel(groupListModel);
1154         allUsersList.setVisible(false);
1155         allGroupsList.setVisible(true);
1156         userLabel.setVisible(false);
1157         groupLabel.setVisible(true);
1158         deleteGroupButton.setEnabled(true);
1159         addGroupAdminBtn.setEnabled(true);
1160         deleteUserButton.setEnabled(false);
1161         addUserAdminBtn.setEnabled(false);
1162         listComboBoxPanel.setVisible(false);
1163         listCboxLabel.setVisible(false);
1164         refreshUsersBtn.setVisible(false);
```

```
1165         searchUserBtn.setVisible(false);
1166         searchUserField.setVisible(false);
1167         searchGroupBtn.setVisible(true);
1168         searchGroupField.setVisible(true);
1169         searchNameLabel.setVisible(false);
1170         searchGroupLabel.setVisible(true);
1171         chooseFileBtn.setVisible(false);
1172         uploadedFileHolder.setVisible(false);
1173         adminNameField.setVisible(false);
1174         adminLastNameField.setVisible(false);
1175         adminNumberField.setVisible(false);
1176         adminEmailField.setVisible(false);
1177         adminNameLabel.setVisible(false);
1178         adminLastNameLabel.setVisible(false);
1179         adminNumberLabel.setVisible(false);
1180         adminEmailLabel.setVisible(false);
1181         confirmButton.setVisible(false);
1182         errorGroupId.setVisible(false);
1183         errorGroupName.setVisible(false);
1184         addUserCboxPanel.setVisible(false);
1185     }
1186 
1187     /**
1188      * The two next methods deleteUser and deleteGroup, gets the selected index
1189      * and delete it from the list and from our system with pg.deletePerson and
1190      * pgp.deletePersonGroup
1191      * @param e this is the ActionEvent
1192      */
1193     public void deleteUserBtnActionPerformed(java.awt.event.ActionEvent e) {
1194         int selectedChoice = allUsersList.getSelectedIndex();
1195         if (selectedChoice == -1) {
1196             infoBox("Please select a user...", "Error");
1197         } else {
1198             String selectedName = allUsersList.getSelectedValue().toString();
1199             String groupId = getGroupId(listComboBox);
1200             pgp.deletePerson(selectedName, groupId);
1201             listModel.remove(selectedChoice);
1202             allUsersList.remove(selectedChoice);
1203         }
1204 
1205     }
1206 
1207     public void deleteGroupBtnActionPerformed(java.awt.event.ActionEvent e) {
1208         int selectedChoice = allGroupsList.getSelectedIndex();
```

```
1209         if (selectedChoice == -1) {
1210             infoBox("Please select a group...", "Error");
1211         } else {
1212             String selectedGroupName =
allGroupsList.getSelectedValue().toString();
1213             pg.deletePersonGroup(selectedGroupName);
1214             groupListModel.remove(selectedChoice);
1215         }
1216     }
1217 
1218     /**
1219      * A button that opens the Add User frame, starts with refreshing the
groups.
1220      * @param e this is the ActionEvent
1221      */
1222     public void addUserBtnActionPerformed(java.awt.event.ActionEvent e) {
1223         pg.listPersonGroup();
1224         comboBoxModel.removeAllElements();
1225         for (int i = 0; i < pg.groupArray.size(); i++) {
1226             comboBoxModel.addElement(pg.groupArray.get(i));
1227         }
1228         captureFrame.setVisible(true);
1229         adminFrame.setVisible(false);
1230     }
1231 
1232     /**
1233      * Opens the password frame
1234      * @param e this is the ActionEvent
1235      */
1236     public void adminBtnActionPerformed(java.awt.event.ActionEvent e) {
1237         if (!adminFrame.isVisible()) {
1238             passwordField.setBorder(BorderFactory.createLineBorder(new Color(1,
49, 63)));
1239             passwordField.setText("");
1240             passwordWindow.setVisible(true);
1241             captureFrame.setVisible(false);
1242         } else {
1243             adminFrame.setAlwaysOnTop(true);
1244         }
1245     }
1246 
1247     /**
1248      * When the send button is pressed the taken picture will be the users
"face"
```

```
1249      * that is getting added. If the face is equal to a face that already exist
in the system, the face
1250      * will be added to the existing person. If not a new person will be added
in a specified group.
1251      * To send all the textfields must be correctly filled out.
1252      * And then it will train the person group.
1253      * @param e this is the ActionEvent
1254      */
1255     public void sendBtnActionPerformed(java.awt.event.ActionEvent e) {
1256         File file = new File("/Users/Adrian/Bachelor-2019/FaceAPI_TEST/1.png");
1257 
1258         Azure_GUI fdGui = new Azure_GUI();
1259         String groupIdString = getGroupId(groupComboBox);
1260         fdGui.detectFace(file, groupIdString);
1261         boolean identify = fdGui.identify();
1262         System.out.println("IDENTIFY: " + identify);
1263         if (identify == true) {
1264             pgp.addFaceToPerson(file, fdGui.id, groupIdString);
1265         } else {
1266 
1267             JSONObject obj = new JSONObject();
1268             obj.put("userdata", "TLF: " + numberField.getText() + " EMAIL: "
1269                     + emailField.getText());
1270             obj.put("name", nameField.getText());
1271             pgp.createPersonToPg(obj.toString(), groupIdString);
1272             pgp.addFaceToPerson(file, pgp.personIdString, groupIdString);
1273             pg.trainPersonGroup(groupIdString);
1274         }
1275         imageholder.setVisible(false);
1276         nameField.setText("");
1277         emailField.setText("");
1278         numberField.setText("");
1279         panel.setVisible(true);
1280         retakeButton.setVisible(false);
1281         captureButton.setVisible(true);
1282         nameField.setBorder(BorderFactory.createLineBorder(new Color(1, 49,
63)));
1283         groupComboBox.setSelectedIndex(0);
1284     }
1285 
1286     /**
1287      * This method checks if the text fields are empty and sets their borders
as red.
1288      * Returns a boolean false in case any of the text fields are empty, else
```

```
true.
1289      * @return bolean Returns a boolean false in case any of the text fields
are empty, else true.
1290      */
1291     private boolean addUserTextValid() {
1292         boolean validTextFields = true;
1293         String name = nameField.getText();
1294         String email = emailField.getText();
1295         String numb = numberField.getText();
1296 
1297         // Checks if the nameField is correctly filled in. Checks if the field
1298         // is empty and if there are enough letters
1299         if (name.isEmpty()) {
1300             validTextFields = false;
1301             nameField.setBorder(new LineBorder(Color.red));
1302         }
1303 
1304         // Checks if the emailField is correctly filled in. Checks if the field
1305         // is empty, if theres a @ in the field and if there are enough
letters.
1306         if (email.isEmpty()) {
1307             validTextFields = false;
1308             emailField.setText("tomt");
1309             emailField.setBorder(new LineBorder(Color.red));
1310         }
1311         if (!email.contains("@") && !email.contains(".")) {
1312             validTextFields = false;
1313             emailField.setText("Incorrect adress");
1314             emailField.setBorder(new LineBorder(Color.red));
1315         }
1316 
1317         // Checks if the numberField is correctly filled in, checks if the
field
1318         // is empty, if its just numbers, and exact 8 digits.
1319         if (numb.isEmpty()) {
1320             validTextFields = false;
1321             numberField.setBorder(new LineBorder(Color.red));
1322         }
1323         if (numb != null) {
1324             if (!numb.matches("[0-9]*")) {
1325                 numberField.setText("Try again, digits only");
1326                 validTextFields = false;
1327                 numberField.setBorder(new LineBorder(Color.red));
1328             }
```

```
1329         }
1330         if (numb.length() != 8) {
1331             numberField.setText("Must be 8 digits");
1332             validTextFields = false;
1333             numberField.setBorder(new LineBorder(Color.red));
1334         }
1335         return validTextFields;
1336     }
1337 
1338     /**
1339      * This method checks if the groupName and groupId textfields are empty and
sets
1340      * their borders as red.
1341      * This method also checks if the groupName only uses letters and numbers,
1342      * also if the groupID uses small letters and numbers.
1343      * @return boolean Returns a boolean false in case any of the text fields
are empty, else true.
1344      */
1345     private boolean creatGroupTextValid() {
1346         boolean validTextFields = true;
1347         String groupName = groupNameField.getText();
1348         String groupId = groupIdField.getText();
1349 
1350         // Checks if the text field is correctly filled in. Checks if the field
1351         // is empty, if theres a @ in the field and if there are enough
letters.
1352         if (groupName.isEmpty()) {
1353             validTextFields = false;
1354             groupNameField.setText("");
1355             groupNameField.setBorder(new LineBorder(Color.red));
1356 
1357         }
1358         if (!groupName.matches("[a-zA-Z0-9]*")) {
1359             validTextFields = false;
1360             groupNameField.setText("");
1361             groupNameField.setBorder(new LineBorder(Color.red));
1362         }
1363 
1364         if (groupId.isEmpty()) {
1365             validTextFields = false;
1366             groupIdField.setText("");
1367             groupIdField.setBorder(new LineBorder(Color.red));
1368         }
1369 
```

```
1370         if (!groupId.matches(
1371                 "[a-z0-9]*")) {
1372             validTextFields = false;
1373             groupIdField.setText("");
1374             groupIdField.setBorder(new LineBorder(Color.red));
1375         }
1376 
1377         if (groupId.length() >= 60) {
1378             groupIdField.setText("Must be less than 60 digits");
1379             validTextFields = false;
1380             groupIdField.setBorder(new LineBorder(Color.red));
1381         }
1382 
1383         if (validTextFields == true) {
1384             groupIdField.setBorder(new LineBorder(Color.LIGHT_GRAY));
1385             groupNameField.setBorder(new LineBorder(Color.LIGHT_GRAY));
1386         }
1387         return validTextFields;
1388     }
1389 
1390     /**
1391      * Removes the taken image and makes the panel ready for a new picture.
1392      * @param e this is the ActionEvent
1393      */
1394     public void retakeBtnActionPerformed(java.awt.event.ActionEvent e) {
1395         imageholder.setVisible(false);
1396         panel.setVisible(true);
1397         retakeButton.setVisible(false);
1398         captureButton.setVisible(true);
1399         sendButton.setEnabled(false);
1400     }
1401 
1402     /**
1403      * If the captureButton is pressed, a picture will be taken. The taken
picture will
1404      * be shown in the imageholder. Every picture captured will be saved as
"1.png"
1405      * and will overwrite the last picture.
1406      * @param e this is the ActionEvent
1407      */
1408     public void captureBtnActionPerformed(java.awt.event.ActionEvent e) {
1409         this.webcam.open();
1410         Image image = webcam.getImage();
1411         imageholder.setIcon(new ImageIcon(image));
```

```
1412         imageholder.setVisible(true);
1413         panel.setVisible(false);
1414         captureButton.setVisible(false);
1415         retakeButton.setVisible(true);
1416         sendButton.setEnabled(true);
1417         nameField.setBorder(BorderFactory.createLineBorder(new Color(1, 49,
63)));
1418         try {
1419             ImageIO.write(webcam.getImage(), "PNG",
1420                     new File("1.png"));
1421 
1422         } catch (IOException ex) {
1423             Logger.getLogger(WebCamFeedGUI.class.getName()).log(Level.SEVERE,
null, ex);
1424         }
1425     }
1426 
1427     /**
1428      * This method takes what is written in the passwordfield and checks if its
correct
1429      * and prints messages if its wrong or right.
1430      * @param e this is the ActionEvent
1431      * @return boolean This returns true if the password is correct and false
if wrong.
1432      */
1433     public boolean actionPerformed1(java.awt.event.ActionEvent e) {
1434         String cmd = e.getActionCommand();
1435 
1436         char[] input = passwordField.getPassword();
1437         if (isPasswordCorrect(input)) {
1438             return true;
1439         } else {
1440             infoBox("Wrong Password. Please try again.", "Wrong password");
1441             return false;
1442         }
1443     }
1444 
1445     /**
1446      * This method sets the password and checks if the password is correct
1447      * @param input It is the password input written by the user
1448      * @return boolean This returns if the password is correct or not
1449      */
1450     public static boolean isPasswordCorrect(char[] input) {
1451 
```

```
1452         boolean isCorrect = true;
1453         char[] correctPassword = {'a', 'd', 'm', 'i', 'n'};
1454 
1455         if (input.length != correctPassword.length) {
1456             isCorrect = false;
1457         } else {
1458             isCorrect = Arrays.equals(input, correctPassword);
1459         }
1460 
1461         //Zero out the password.
1462         Arrays.fill(correctPassword, '0');
1463 
1464         return isCorrect;
1465     }
1466 
1467     /**
1468      * Get method to get the groupId string from the combobox.
1469      * @param comboBox the combobox we want to extract an ID from.
1470      * @return String This returns the String of the groupId
1471      */
1472     private String getGroupId(JComboBox comboBox) {
1473         Object chosenGroup = comboBox.getSelectedItem();
1474         Object groupIdObject = pg.groups.get(chosenGroup);
1475         String groupIdString = groupIdObject.toString();
1476         return groupIdString;
1477     }
1478 
1479     /**
1480      * Popup box for showing messages to the user.
1481      * This method was found on StackOverflow at the following link:
1482      * https://stackoverflow.com/questions/7080205/popup-message-boxes
1483      *
1484      * @param message the message to show.
1485      * @param title the title of the message and box
1486      */
1487     private static void infoBox(String message, String title) {
1488         // null puts it in the center of the screen. Size is adjusted according
1489         // to the size of the message.
1490         JOptionPane.showMessageDialog(null, message, "InfoBox: " + title,
JOptionPane.INFORMATION_MESSAGE);
1491     }
1492 
1493 }
```

D Swift kildekode

ColoredCardView

```
import UIKit
import Wallet

import RandomColorSwift
import DynamicColor

class ColoredCardView: CardView {

    //var viewController: ViewController = ViewController()

    @IBOutlet weak var qrTextField: UILabel!
    @IBOutlet weak var contentView: UIView!
    @IBOutlet weak var qrView: UIImageView!

    let presentedCardViewColor = UIColor(red:0.22, green:0.24, blue:
0.27, alpha:1.0)

    lazy var depresentedCardViewColor:   UIColor = { return
self.presentedCardViewColor.lighter() }()

    @IBOutlet weak var indexLabel: UILabel!
    var index: Int = 0 {
        didSet {
            indexLabel.text = "# \(index)"
        }
    }

    func setQrText(value: Bool){
        qrTextField.isHidden = value
    }

    func setQrCode(image: UIImage){
        qrView.image = image
    }

    override func awakeFromNib() {
        super.awakeFromNib()

        contentView.layer.cornerRadius  = 10
        contentView.layer.masksToBounds = true

        presentedDidUpdate()

    }

    override var presented: Bool { didSet { presentedDidUpdate() } }

    func presentedDidUpdate() {

        removeCardViewButton.isHidden = !presented
        contentView.backgroundColor = presented ?
presentedCardViewColor : depresentedCardViewColor
        contentView.addTransitionFade()
```

```
    }

    @IBOutlet weak var removeCardViewButton: UIButton!
    @IBAction func removeCardView(_ sender: Any) {

        ViewController().setIndexOneNegativ()
        walletView?.remove(cardView: self, animated: true)
    }

}
```

ColoredCardViewXib

```
<?xml version="1.0" encoding="UTF-8"?>
<document type="com.apple.InterfaceBuilder3.CocoaTouch.XIB"
version="3.0" toolsVersion="14460.31" targetRuntime="iOS.CocoaTouch"
propertyAccessControl="none" useAutolayout="YES"
useTraitCollections="YES" colorMatched="YES">
    <device id="retina3_5" orientation="portrait">
        <adaptation id="fullscreen"/>
    </device>
    <dependencies>
        <plugIn
identifier="com.apple.InterfaceBuilder.IBCocoaTouchPlugin"
version="14460.20"/>
        <capability name="documents saved in the Xcode 8 format"
minToolsVersion="8.0"/>
    </dependencies>
    <objects>
        <placeholder placeholderIdentifier="IBFilesOwner" id="-1"
userLabel="File's Owner"/>
        <placeholder placeholderIdentifier="IBFirstResponder"
id="-2" customClass="UIResponder"/>
        <view contentMode="scaleToFill" id="iN0-l3-epB"
customClass="ColoredCardView" customModule="Wallet_Example"
customModuleProvider="target">
            <rect key="frame" x="0.0" y="0.0" width="320"
height="480"/>
            <autoresizingMask key="autoresizingMask"
widthSizable="YES" heightSizable="YES"/>
            <subviews>
                <imageView userInteractionEnabled="NO"
contentMode="scaleToFill" horizontalHuggingPriority="251"
verticalHuggingPriority="251" image="Shadow"
translatesAutoresizingMaskIntoConstraints="NO" id="nBp-qR-HbH">
                    <rect key="frame" x="-2" y="-4" width="324"
height="20"/>
                    <constraints>
                        <constraint firstAttribute="height"
constant="20" id="zTo-8q-qjH"/>
                    </constraints>
                </imageView>
                <imageView userInteractionEnabled="NO"
contentMode="scaleToFill" horizontalHuggingPriority="251"
verticalHuggingPriority="251" image="Shadow"
translatesAutoresizingMaskIntoConstraints="NO" id="0O6-Zo-qgT">
                    <rect key="frame" x="-2" y="464" width="324"
height="20"/>
                    <constraints>
                        <constraint firstAttribute="height"
constant="20" id="5tp-Zu-PGm"/>
                    </constraints>
                </imageView>
                <view clipsSubviews="YES" contentMode="scaleToFill"
translatesAutoresizingMaskIntoConstraints="NO" id="OGR-lR-C5v">
                    <rect key="frame" x="0.0" y="0.0" width="320"
height="480"/>
```

```
                    <subviews>
                        <button opaque="NO"
contentMode="scaleToFill" contentHorizontalAlignment="center"
contentVerticalAlignment="center" buttonType="roundedRect"
lineBreakMode="middleTruncation"
translatesAutoresizingMaskIntoConstraints="NO" id="Z4R-si-942">
                            <rect key="frame" x="0.0" y="436"
width="320" height="44"/>
                            <color key="backgroundColor" white="1"
alpha="1" colorSpace="calibratedWhite"/>
                            <constraints>
                                <constraint firstAttribute="height"
constant="44" id="hjV-Zg-aXI"/>
                            </constraints>
                            <fontDescription key="fontDescription"
type="system" pointSize="17"/>
                            <color key="tintColor" red="1"
green="0.074743433489999997" blue="0.0" alpha="1"
colorSpace="custom" customColorSpace="sRGB"/>
                            <state key="normal" title="Remove
CardView"/>
                            <connections>
                                <action selector="removeCardView:"
destination="iN0-l3-epB" eventType="touchUpInside" id="IVD-gZ-hsC"/>
                            </connections>
                        </button>
                        <label opaque="NO"
userInteractionEnabled="NO" contentMode="left"
horizontalHuggingPriority="251" verticalHuggingPriority="251"
text="# 1" textAlignment="natural" lineBreakMode="tailTruncation"
baselineAdjustment="alignBaselines" adjustsFontSizeToFit="NO"
translatesAutoresizingMaskIntoConstraints="NO" id="0ks-p9-g31">
                            <rect key="frame" x="8" y="8"
width="173" height="21"/>
                            <constraints>
                                <constraint firstAttribute="width"
constant="173" id="rOf-cc-kfb"/>
                            </constraints>
                            <fontDescription key="fontDescription"
type="system" weight="light" pointSize="17"/>
                            <color key="textColor" white="1"
alpha="1" colorSpace="custom"
customColorSpace="genericGamma22GrayColorSpace"/>
                            <nil key="highlightedColor"/>
                        </label>
                        <imageView userInteractionEnabled="NO"
contentMode="scaleToFill" horizontalHuggingPriority="251"
verticalHuggingPriority="251"
translatesAutoresizingMaskIntoConstraints="NO" id="UWf-ZA-RJ0">
                            <rect key="frame" x="60" y="120"
width="200" height="200"/>
                            <constraints>
                                <constraint firstAttribute="width"
secondItem="UWf-ZA-RJ0" secondAttribute="height" multiplier="1:1"
```

```
id="BrN-pF-gHa"/>
                                <constraint firstAttribute="height"
constant="200" id="wEn-0m-ne0"/>
                            </constraints>
                        </imageView>
                    </subviews>
                    <color key="backgroundColor" white="1" alpha="1"
colorSpace="calibratedWhite"/>
                    <constraints>
                        <constraint firstItem="UWf-ZA-RJ0"
firstAttribute="centerX" secondItem="Z4R-si-942"
secondAttribute="centerX" id="48F-Um-0QE"/>
                        <constraint firstAttribute="trailing"
secondItem="Z4R-si-942" secondAttribute="trailing" id="8NK-LQ-Tn5"/>
                        <constraint firstItem="UWf-ZA-RJ0"
firstAttribute="leading" secondItem="OGR-lR-C5v"
secondAttribute="leading" constant="60" id="JBg-D3-0gg"/>
                        <constraint firstItem="UWf-ZA-RJ0"
firstAttribute="top" secondItem="OGR-lR-C5v" secondAttribute="top"
constant="120" id="RoN-aA-WF8"/>
                        <constraint firstItem="0ks-p9-g31"
firstAttribute="leading" secondItem="OGR-lR-C5v"
secondAttribute="leadingMargin" id="bka-OB-dTj"/>
                        <constraint firstItem="UWf-ZA-RJ0"
firstAttribute="top" secondItem="0ks-p9-g31"
secondAttribute="bottom" constant="91" id="iXU-X8-fmU"/>
                        <constraint firstAttribute="bottom"
secondItem="Z4R-si-942" secondAttribute="bottom" id="pJW-F2-0vg"/>
                        <constraint firstItem="Z4R-si-942"
firstAttribute="leading" secondItem="OGR-lR-C5v"
secondAttribute="leading" id="rXN-pF-x9F"/>
                    </constraints>
                </view>
            </subviews>
            <color key="backgroundColor" white="0.0" alpha="0.0"
colorSpace="calibratedWhite"/>
            <constraints>
                <constraint firstItem="OGR-lR-C5v"
firstAttribute="centerX" secondItem="iN0-l3-epB"
secondAttribute="centerX" id="6Au-be-gZ8"/>
                <constraint firstItem="OGR-lR-C5v"
firstAttribute="centerY" secondItem="iN0-l3-epB"
secondAttribute="centerY" id="AG8-xl-na4"/>
                <constraint firstAttribute="trailing"
secondItem="nBp-qR-HbH" secondAttribute="trailing" constant="-2"
id="HWI-5Z-tUp"/>
                <constraint firstItem="OGR-lR-C5v"
firstAttribute="width" secondItem="iN0-l3-epB"
secondAttribute="width" id="TEo-HT-6Tw"/>
                <constraint firstItem="nBp-qR-HbH"
firstAttribute="leading" secondItem="iN0-l3-epB"
secondAttribute="leading" constant="-2" id="YfS-a6-yot"/>
                <constraint firstAttribute="bottom" secondItem="0O6-
Zo-qgT" secondAttribute="bottom" constant="-4" id="dyb-5Z-TyC"/>
```

```
                <constraint firstItem="OGR-lR-C5v"
firstAttribute="height" secondItem="iN0-l3-epB"
secondAttribute="height" id="e7s-HL-Ytp"/>
                <constraint firstAttribute="trailing"
secondItem="0O6-Zo-qgT" secondAttribute="trailing" constant="-2"
id="fJY-lv-HYd"/>
                <constraint firstItem="nBp-qR-HbH"
firstAttribute="top" secondItem="iN0-l3-epB" secondAttribute="top"
constant="-4" id="qCQ-Wu-GFd"/>
                <constraint firstItem="0O6-Zo-qgT"
firstAttribute="leading" secondItem="iN0-l3-epB"
secondAttribute="leading" constant="-2" id="xAF-8i-ptM"/>
            </constraints>
            <nil key="simulatedStatusBarMetrics"/>
            <connections>
                <outlet property="contentView" destination="OGR-lR-
C5v" id="fSz-Gj-R5Z"/>
                <outlet property="indexLabel" destination="0ks-p9-
g31" id="MY2-PU-84M"/>
                <outlet property="qrView" destination="UWf-ZA-RJ0"
id="SaJ-3S-Dru"/>
                <outlet property="removeCardViewButton"
destination="Z4R-si-942" id="FgL-Mx-pc7"/>
            </connections>
            <point key="canvasLocation" x="-125.625" y="-55"/>
        </view>
    </objects>
    <resources>
        <image name="Shadow" width="27.5" height="18"/>
    </resources>
</document>
```

UIViewExtension

```
import UIKit

extension UIView {

    static func reuseIdentifier() -> String {
        return
NSStringFromClass(classForCoder()).components(separatedBy:
".").last!
    }

    static func UINibForClass(_ bundle: Bundle? = nil) -> UINib {
        return UINib(nibName: reuseIdentifier(), bundle: bundle)
    }

    static func nibForClass() -> Self {
        return loadNib(self)

    }

    static func loadNib<A>(_ owner: AnyObject, bundle: Bundle =
Bundle.main) -> A {

        let nibName =
NSStringFromClass(classForCoder()).components(separatedBy:
".").last!

        let nib = bundle.loadNibNamed(nibName, owner: owner,
options: nil)!

        for item in nib {
            if let item = item as? A {
                return item
            }
        }

        return nib.last as! A

    }

    func addTransitionFade(_ duration: TimeInterval = 0.5) {
        let animation = CATransition()

        animation.type = CATransitionType.fade
        animation.timingFunction = CAMediaTimingFunction(name:
CAMediaTimingFunctionName.default)
        animation.fillMode = CAMediaTimingFillMode.forwards
        animation.duration = duration

        layer.add(animation, forKey: "kCATransitionFade")

    }

}
```

ViewController

```
import UIKit
import Wallet
import AVFoundation
import SwiftOTP


class ViewController: UIViewController {

    var lastSecret: String = "start"

    var index: Int = 1

    var qrCard: ColoredCardView = ColoredCardView()

    var coloredCardViews = [ColoredCardView]()

    var qrData: Data?

    var hasQr: Bool = false

    var cardIsMade: Bool = false

    var qrTimer: Timer!

    @IBOutlet weak var walletHeaderView: UIView!
    @IBOutlet weak var walletView: WalletView!

    @IBOutlet weak var addCardViewButton: UIButton!

    // QR reader
    var captureSession = AVCaptureSession()

    var videoPreviewLayer: AVCaptureVideoPreviewLayer?
    var qrCodeFrameView: UIView?

    private let supportedCodeTypes =
[AVMetadataObject.ObjectType.upce,

AVMetadataObject.ObjectType.code39,

AVMetadataObject.ObjectType.code39Mod43,

AVMetadataObject.ObjectType.code93,

AVMetadataObject.ObjectType.code128,

AVMetadataObject.ObjectType.ean8,

AVMetadataObject.ObjectType.ean13,

AVMetadataObject.ObjectType.aztec,

AVMetadataObject.ObjectType.pdf417,
```

```
AVMetadataObject.ObjectType.itf14,

AVMetadataObject.ObjectType.dataMatrix,

AVMetadataObject.ObjectType.interleaved2of5,

AVMetadataObject.ObjectType.qr]



    override func viewDidLoad() {
        super.viewDidLoad()

        qrTimer = Timer.scheduledTimer(timeInterval: 1, target:
self, selector:          #selector(updateQrCodes), userInfo: nil,
repeats: true)

        walletView.walletHeader = walletHeaderView

        walletView.contentInset = UIEdgeInsets(top: 20, left: 0,
bottom: 0, right: 0)



        walletView.didUpdatePresentedCardViewBlock = { [weak self]
(_) in
            self?.showAddCardViewButtonIfNeeded()
            self?.addCardViewButton.addTransitionFade()
        }

        // Get the back-facing camera for capturing videos
        let deviceDiscoverySession =
AVCaptureDevice.DiscoverySession(deviceTypes:
[.builtInWideAngleCamera], mediaType: AVMediaType.video,
position: .back)

        guard let captureDevice =
deviceDiscoverySession.devices.first else {
            print("Failed to get the camera device")
            return

        }


        do {
            // Get an instance of the AVCaptureDeviceInput class
using the previous device                  object.
            let input = try AVCaptureDeviceInput(device:
captureDevice)

            // Set the input device on the capture session.
            captureSession.addInput(input)
```

```
            // Initialize a AVCaptureMetadataOutput object and set
it as the output device           to the capture session.
            let captureMetadataOutput = AVCaptureMetadataOutput()
            captureSession.addOutput(captureMetadataOutput)

            // Set delegate and use the default dispatch queue to
execute the call back
            captureMetadataOutput.setMetadataObjectsDelegate(self
as?
AVCaptureMetadataOutputObjectsDelegate, queue: DispatchQueue.main)
            captureMetadataOutput.metadataObjectTypes =
supportedCodeTypes
            //            captureMetadataOutput.metadataObjectTypes
=
[AVMetadataObject.ObjectType.qr]

        } catch {
            // If any error occurs, simply print it out and don't
continue any more.
            print(error)
            return
        }



        // Start video capture.
        captureSession.startRunning()


    }

    @objc func updateQrCodes(){


        var otpString: String?

                for i in coloredCardViews {
                    if let totp = TOTP(secret: qrData!) {
                        otpString = totp.generate(time: Date.init())

                        // Get data from the string
                        let data = otpString?.data(using:
String.Encoding.ascii)

                        // Get a QR CIFilter
                        guard let qrFilter = CIFilter(name:
"CIQRCodeGenerator") else
{ return }
                        // Input the data
                        qrFilter.setValue(data, forKey:
"inputMessage")
                        // Get the output image
                        guard let qrImage = qrFilter.outputImage
else { return }
```

```
                        // Scale the image
                        let transform = CGAffineTransform(scaleX:
10, y: 10)
                        let scaledQrImage = qrImage.transformed(by:
transform)

                        // Replace the black with transparency
                        guard let maskToAlphaFilter = CIFilter(name:
"CIMaskToAlpha") else                       { return }
                        maskToAlphaFilter.setValue(scaledQrImage,
forKey: "inputImage")
                        guard let outputCIImage =
maskToAlphaFilter.outputImage else
{ return }

                        // Do some processing to get the UIImage
                        let context = CIContext()
                        guard let cgImage =
context.createCGImage(outputCIImage, from:
         scaledQrImage.extent) else { return }
                        let processedImage = UIImage(cgImage:
cgImage)
                        i.qrView.image = processedImage
                }



            }

        }




    func showAddCardViewButtonIfNeeded() {
        addCardViewButton.alpha = walletView.presentedCardView ==
nil ||                    walletView.insertedCardViews.count <= 1 ?
1.0 : 0.0
    }

    @IBAction func addCardViewAction(_ sender: Any) {

        walletView.insert(cardView: ColoredCardView.nibForClass(),
animated: true,           presented: true)

    }

    override var prefersStatusBarHidden: Bool {
        return false
    }

    func setIndexOneNegativ(){
         index -= 1
    }
```

```
    func decodeQrCode(secret: String){

        guard let data = base32DecodeToData(secret) else { return }
        qrData = data
        hasQr = true

        //
        var otpString: String?

        print("inside..")

        if (hasQr == true){
            if let totp = TOTP(secret: qrData!) {
                otpString = totp.generate(time: Date.init())

                // Get data from the string
                let data = otpString?.data(using:
String.Encoding.ascii)

                // Get a QR CIFilter
                guard let qrFilter = CIFilter(name:
"CIQRCodeGenerator") else { return }
                // Input the data
                qrFilter.setValue(data, forKey: "inputMessage")
                // Get the output image
                guard let qrImage = qrFilter.outputImage else
{ return }
                // Scale the image
                let transform = CGAffineTransform(scaleX: 10, y: 10)
                let scaledQrImage = qrImage.transformed(by:
transform)

                // Replace the black with transparency
                guard let maskToAlphaFilter = CIFilter(name:
"CIMaskToAlpha") else                    { return }
                maskToAlphaFilter.setValue(scaledQrImage, forKey:
"inputImage")
                guard let outputCIImage =
maskToAlphaFilter.outputImage else { return }

                // Do some processing to get the UIImage
                let context = CIContext()
                guard let cgImage =
context.createCGImage(outputCIImage, from:
        scaledQrImage.extent) else { return }
                let processedImage = UIImage(cgImage: cgImage)
                print(otpString!)

                if (lastSecret != otpString!){
                    lastSecret = otpString!

                    if (cardIsMade == false){
```

```
                        cardIsMade = true

                        //var coloredCardViews = [ColoredCardView]()
                        let cardView = ColoredCardView.nibForClass()
                        cardView.qrView.image = processedImage
                        cardView.indexLabel.text = "Avento"
                        cardView.index = index
                        index = index + 1
                        coloredCardViews.append(cardView)

                        walletView.reload(cardViews:
coloredCardViews)

                    }

                }

            }



        }
    }
}


extension ViewController: AVCaptureMetadataOutputObjectsDelegate {


    func metadataOutput(_ output: AVCaptureMetadataOutput, didOutput
metadataObjects:                    [AVMetadataObject], from
connection: AVCaptureConnection) {
        // Check if the metadataObjects array is not nil and it
contains at least one             object.
        if metadataObjects.count == 0 {
            //qrCodeFrameView?.frame = CGRect.zero
            //messageLabel.text = "No QR code is detected"



            return
        }

        // Get the metadata object.
        let metadataObj = metadataObjects[0] as!
AVMetadataMachineReadableCodeObject

        if supportedCodeTypes.contains(metadataObj.type) {
            // If the found metadata is equal to the QR code
metadata (or barcode) then                update the status label's
text and set the bounds
            //let barCodeObject =
videoPreviewLayer?.transformedMetadataObject(for:
metadataObj)
```

```
            //qrCodeFrameView?.frame = barCodeObject!.bounds

            if metadataObj.stringValue != nil {
                print(metadataObj.stringValue!)
                cardIsMade = false
                decodeQrCode(secret: metadataObj.stringValue!)
                //messageLabel.text = metadataObj.stringValue
            }
        }
    }


}
```

Main.storyBoard

```
<?xml version="1.0" encoding="UTF-8"?>
<document
type="com.apple.InterfaceBuilder3.CocoaTouch.Storyboard.XIB"
version="3.0" toolsVersion="14460.31" targetRuntime="iOS.CocoaTouch"
propertyAccessControl="none" useAutolayout="YES"
useTraitCollections="YES" colorMatched="YES"
initialViewController="vXZ-lx-hvc">
    <device id="retina3_5" orientation="portrait">
        <adaptation id="fullscreen"/>
    </device>
    <dependencies>
        <plugIn
identifier="com.apple.InterfaceBuilder.IBCocoaTouchPlugin"
version="14460.20"/>
        <capability name="documents saved in the Xcode 8 format"
minToolsVersion="8.0"/>
    </dependencies>
    <scenes>
        <!--View Controller-->
        <scene sceneID="ufC-wZ-h7g">
            <objects>
                <viewController id="vXZ-lx-hvc"
customClass="ViewController" customModule="Wallet_Example"
customModuleProvider="target" sceneMemberID="viewController">
                    <layoutGuides>
                        <viewControllerLayoutGuide type="top"
id="jyV-Pf-zRb"/>
                        <viewControllerLayoutGuide type="bottom"
id="2fi-mo-0CV"/>
                    </layoutGuides>
                    <view key="view" contentMode="scaleToFill"
id="kh9-bI-dsS">
                        <rect key="frame" x="0.0" y="0.0"
width="320" height="480"/>
                        <autoresizingMask key="autoresizingMask"
flexibleMaxX="YES" flexibleMaxY="YES"/>
                        <subviews>
                            <view contentMode="scaleToFill"
fixedFrame="YES" translatesAutoresizingMaskIntoConstraints="NO"
id="pwH-rO-BWA" customClass="WalletView" customModule="Wallet">
                                <rect key="frame" x="0.0" y="0.0"
width="320" height="480"/>
                                <autoresizingMask
key="autoresizingMask" widthSizable="YES" heightSizable="YES"/>
                                <color key="backgroundColor"
white="1" alpha="1" colorSpace="custom"
customColorSpace="genericGamma22GrayColorSpace"/>
                            </view>
                            <view contentMode="scaleToFill"
fixedFrame="YES" translatesAutoresizingMaskIntoConstraints="NO"
id="rp7-RM-zY5">
                                <rect key="frame" x="0.0" y="20"
width="320" height="48"/>
                                <autoresizingMask
```

```
key="autoresizingMask" widthSizable="YES" flexibleMaxY="YES"/>
                                <subviews>
                                    <label opaque="NO"
userInteractionEnabled="NO" contentMode="left"
horizontalHuggingPriority="251" verticalHuggingPriority="251"
fixedFrame="YES" text="PASSWORDS" textAlignment="center"
lineBreakMode="tailTruncation" baselineAdjustment="alignBaselines"
adjustsFontSizeToFit="NO"
translatesAutoresizingMaskIntoConstraints="NO" id="W3P-PW-CxD">
                                        <rect key="frame" x="0.0"
y="-2" width="320" height="48"/>
                                        <autoresizingMask
key="autoresizingMask" widthSizable="YES" heightSizable="YES"/>
                                        <fontDescription
key="fontDescription" type="system" weight="ultraLight"
pointSize="25"/>
                                        <nil key="textColor"/>
                                        <nil key="highlightedColor"/
>
                                    </label>
                                    <button opaque="NO"
contentMode="scaleToFill" fixedFrame="YES"
contentHorizontalAlignment="center"
contentVerticalAlignment="center" buttonType="contactAdd"
lineBreakMode="middleTruncation"
translatesAutoresizingMaskIntoConstraints="NO" id="crn-64-bx2">
                                        <rect key="frame" x="290"
y="11" width="22" height="22"/>
                                        <autoresizingMask
key="autoresizingMask" flexibleMinX="YES" flexibleMinY="YES"
flexibleMaxY="YES"/>
                                        <color key="tintColor"
cocoaTouchSystemColor="darkTextColor"/>
                                        <connections>
                                            <action
selector="addCardViewAction:" destination="vXZ-lx-hvc"
eventType="touchUpInside" id="DOI-WQ-Cqs"/>
                                        </connections>
                                    </button>
                                </subviews>
                                <color key="backgroundColor"
white="0.0" alpha="0.0" colorSpace="calibratedWhite"/>
                            </view>
                        </subviews>
                        <color key="backgroundColor" red="1"
green="1" blue="1" alpha="1" colorSpace="custom"
customColorSpace="sRGB"/>
                    </view>
                    <connections>
                        <outlet property="addCardViewButton"
destination="crn-64-bx2" id="i12-t2-OM5"/>
                        <outlet property="walletHeaderView"
destination="rp7-RM-zY5" id="wfi-gh-aea"/>
                        <outlet property="walletView"
```

```
destination="pwH-rO-BWA" id="bzw-8m-mfB"/>
                    </connections>
                </viewController>
                <placeholder
placeholderIdentifier="IBFirstResponder" id="x5A-6p-PRh"
sceneMemberID="firstResponder"/>
            </objects>
            <point key="canvasLocation" x="-384.375" y="-366.25"/>
        </scene>
    </scenes>
</document>
```

E Arduino kildekode

```
const int motionSensorPin = A0; // PIR-sensor pin.
int sensorValue; // Variable for sensor value.



/**
 * Void Setup.
 * Sets motionSensorPin as an input.
 * Sets sensor to 0.
 * Starts Serial communication.
 */
void setup() {
    pinMode(motionSensorPin, INPUT);
    sensorValue = 0;
    Serial.begin(9600);
}

/**
 * The loop reads the value from a PIR-Sensor.
 * Send "1" to java if motion is registred.
 */
void loop() {

  if(readSensor() > 1000){
    sendStringToJava("1");
  }

}

/**
 * Method that read a value from the PIR-Sensor.
 */
int readSensor(){
  sensorValue = analogRead(motionSensorPin);
  return sensorValue;

}

/**
 * Method that send a string to Java.
 */
void sendStringToJava(String msg)
{
  Serial.print(msg);
  Serial.flush();
}
```

F Oppsummering av 2-ukersintervallene

Oversikt over fremgang iløpet av prosjektperioden

Dette er oppsummeringene som er sendt til Avento og veileder iløpet av prosjektperioden.

08.02.2019

- Azure: Her har vi fått satt opp alt slik at vi tar inn et bilde fra webcam, bilde sjekkes så opp mot database i azure og vi får ut om personen finnes i databasen. Ikke møtt på noen store problemer. Vi jobber med å renskrive koden, og det er mulig at den skal skrives om til C++ grunnet bedre kompatibilitet med Intel sitt dybdekamera. Vi må også implementere Azure i javascript til Web appen.
- Web app/GUI: Så og si ferdig med HTML og CSS til web app, Javascript gjenstår. Har laget en fungerende GUI i Java for testing mot Azure, men den er enda ikke koblet opp mot javakoden som snakker med azure.
- Odroid: Har kjøpt oss et dybdekamera (Intel RealSense D435) og satt det opp for Odroid. Kjørt en del testprogram i C++ for å sjekke at alt virker som det skal. Satt opp Odroiden med nødvendige bibliotek og program. Vi har også begynt å skrive kode som kombinerer OpenCV med realsense biblioteket til Intel (SDK 2.0). Fortsatt en del utfordringer med C++ ettersom det er helt nytt for oss, i tillegg til at det stadig dukker opp nye bibliotek som Odroiden mangler. Kameraet fungerer som det skal.
- Låsesystem: Venter fortsatt på mer utfyllende info fra Låsservice. Ringte Per Otto onsdag 06.02.19, han ville ordne det iløpet av dagen men vi har ikke hørt noe mer siden. Ringer igjen på mandag dersom vi fortsatt ikke har fått noe.
- Begynt å skrive rapport.

Ting vi skal gjøre de neste to ukene (hovedfokus):

- Videreutvikle ansiktsverifisering, webløsning, implementere Azure i web og GUI.
- Finne ut hvordan tolke Depth frames fra Intel kamera.
- Utforske andre løsninger enn Azure.
- Forhåpentligvis begynne å kikke på låsesytemdelen.

25.02.2019

Jobbet i hovedsak med å finne gode løsninger på autentisering.

- Installert open3D for python og C++, for å lese av pointcloud data.
- Jobber fortsatt med å finne en god måte å tolke 3D data på.
- Sett på nye løsninger for autentisering.
    - Bruk av head position via azure og 3 kameraer. Får ut posisjon på hodet i 3 akser.
    - Ved bruk av øyne - passord for hver enkelt person. Leser av hvor øynene går over en skjerm. Eget passord for hver enkelt person.

Har også sett på trening av en Haar Cascade modell i OpenCV, der vi trener den på bilder fra depth stream. Har fått til å trene modellen men ikke testet enda pga. problem med Visual Studio og OpenCV på laptop.

Java wrapper for librealsense: kun for Windows 10 og Mac OSX, ikke komplett.
Begynt å kikke på bevegelsessensor. Passiv/aktiv. Vi går sannsynligvis for en passiv.
Gjort ferdig GUI i Java, mangler fortsatt å kobles sammen med Azure. Venter fortsatt på lisens til Adobe for videreutvikling av web app.
Ingen oppdatering angående låsesystem fra låsservice...

Det har ikke blitt gjort så mye siden sist grunnet obligatoriske forelesninger og vinterferie.

Videre arbeid fremover:

Versjon 1: Klar iløpet av uka. Består av enkel GUI koblet opp mot Azure, fullt fungerende med ansiktsgjenkjenning og legge til folk i database.

Versjon 2: Bygge videre på versjon 1 ved å legge til bevegelsessensor i koden og automatisere prosessen. Satser på å bestille bevegelsessensor i dag/morgen. Skal være klar til 8.mars (hvis vi får bestilt sensor i dag).

Versjon 3: Bygge videre på Versjon 1 og 2 ved å implementere dybde. Klart innen slutten av Mars.

Versjon 4: Bytte ut Java GUI med webbasert GUI. Klart 2 uker etter versjon 3

Versjon 5: Optimalisere kode, objektorientere og sanntidsorientere koden. Koble opp mot låsesystem. Ferdig prototype, litt avhengig av Låsservice.

08.03.2019

Vi har nå klar en versjon 2 av prosjektet. Denne versjonen består av tre deler: GUI, autoriseringsmiljø og arduino. GUI kjører på pc mens vi kjører autoriseringsmiljøet på Odroid. Arduino er tilkoblet Odroid.
GUI: Her kan vi nå legge til personer og ansikt til personer i en persongruppe. Når dette legges til trenes automatisk det nye ansiktet til en ny person eller en eksisterende person, dersom navnet eksisterer fra før. Dette kan gjøres samtidig som programmet på Odroiden kjører.

Odroid: autoriseringsmiljøet vil kjøre på odroiden. Den er koblet til en arduino med bevegelsessensor og noen LED lys. Når sensoren ser bevegelse gir den beskjed til Odroid om å starte kamera. Dersom OpenCV registrerer et ansikt starter ansiktsgjenkjenningen. Dersom ansiktet ligger i databasen vil det grønne LED-lyset lyse, som symboliserer at døren åpner seg. Dersom ansiktet ikke er godkjent vil et rødt LED-lys lyse og "døren" vil ikke åpne seg.

Fokuset de neste to ukene blir å prøve å ta hensyn til dybde, og finne ut hvilken metode vi ønsker å bruke. Det mest aktuelle for øyeblikket er Pointcloud i C++ for å kunne bruke

RealSense biblioteket til Intel. Derfor bør også autoriseringsmiljøet skrives om til C++, men vi vil prøve å få til et lite program som sjekker dybde først.

Venter fortsatt på svar og mer informasjon fra Låsservice. Sliter derfor med å kunne koble løsningen opp mot et låsemiljø.

21.03.2019

Vi har nå implementert en løsning for å sjekke etter "spoofing". Dette er gjort ved å implementere Liveness detection ved hjelp av BioID. Dette er flettet sammen med Azure Cognitive Services sin ansiktsgjenkjenning så vi kan ikke lenger komme gjennom eller få godkjent ved hjelp av et bilde på telefonen eller et utskrevet bilde. GUI er nesten ferdig, mangler litt finpuss. Vi jobber fortsatt med å prøve å finne en god løsning med PointCloud men dette virker å være mye vanskeligere enn forventet, spesielt i kombinasjon med C++.

Arbeid fremover:

- Videreutvikle kode og skrive om til modulbasert switch-case.
- Jobbe med PointCloud
- Finpusse og videreutvikle GUI
- Se på implementasjon av låsesystem.

23.04.2019

Vi har nå lagt vekk tanken om PointCloud grunnet tidsbruk. Vi har også byttet ut Odroiden med en Microsoft Surface Go som skolen hadde liggende. På denne vil vi kjøre hele programmet og har der en visuell veileder som brukes som velkomstskjerm. Administrator GUI mangler fortsatt et par funksjoner, men disse håper vi å få implementert iløpet av de neste dagene. Vi har også kikket på låsesystemet og prøvd å leke oss med demoen vi fikk med Låsservice, men vi får ikke gjort noe særlig uten å koble oss til en server og teste metodekallene i xml format. Låsesystem blir derfor ikke lenger prioritert.

Videre arbeid:

- Finpuss av kode
- Rapportskriving
- Implementasjon av de siste funksjonene i Admin GUI

Avslutning av prosjektet

Etter forrige oppsummering var vi ofte i dialog med veileder og hadde møte med Avento. Derfor ble det ikke skrevet flere oppsummeringer for prosjektet, ettersom hovedfokus nå omhandlet rapport og finpuss av kode, og ikke ny funksjonalitet. Hva som er resultatet etter dette er formulert i Rapporten.

G Referat fra oppstartsmøtet

Oppstartsmøte med Avento 14.01.2019

14.01.2019 var vi på besøk hos Avento i Breivika og pratet om våre forventninger samt bedriftens forventninger til det kommende bachelorprosjektet. Prosjektet går ut på å lage et låsesystem med ansiktsgjenkjenning. Avento har allerede lekt seg med dette og laget en enkel løsning som omgår låsesystemet i bygget, så vår utfordring blir å implementere låsesytemet samt videreutvikle og forbedre løsningen. Vi har fått ganske fritt spillerom for hvordan vi ønsker å løse oppgaven. Det ble diskutert å lage en app for å enkelt kunne legge inn bilder i databasen og eventuelt koble systemet opp mot ansattes kalendere. På denne måten kan de ansatte få varsel når den de venter besøk av ankommer bygget, ved at den besøkende kommer til en skjerm, skriver in epost adresse og tar bilde av seg selv (hvis de vil selvsagt).

Mot slutten av møtet vårt ble det planlagt et møte med låseservice angående å få tilgang til låsesystemet deres, samt om de eventuelt kan stille med et lite økosystem (dør med låsesystem) som vi kan bruke i utviklingen av prosjektet.

Fivelstad, Kleppe, Løvlid, Torp

Ansiktsgjenkjenning for Låsesytem

NTNU
Kunnskap for en bedre verden