BACHELOROPPGAVE:

BioDemo - Biometric Demostrator

FORFATTERE:
Anna Kaardal
Joachim Hansen
Synne Gran Østern

DATO:
15.05.2015

Sammendrag av Bacheloroppgaven

Tittel:	**BioDemo - Biometric Demostrator**
Dato:	15.05.2015
Deltakere:	Anna Kaardal Joachim Hansen Synne Gran Østern
Veiledere:	Tom Røise, Høgskolen i Gjøvik
Oppdragsgiver:	Patrick Bours /NisLab & NBL
Kontaktperson:	Prof. Patrick Bours, patrick.bours@hig.no
Nøkkelord:	Biometri, Fingeravtrykk og Tastaturgjenkjenning, Interaksjonsdesign, Programutvikling, Personvern, Java, Databaser
Antall sider:	152
Antall vedlegg:	14
Tilgjengelighet:	Åpen

Sammendrag:

BioDemo er et program som demostrerer hvordan man skanner inn biometriske data, hvordan disse dataene blir brukt til å sammenligne med andre tilsvarende data, og hvilke kvaliteter disse sammenligningene er basert på. Programmet har i denne omgang blitt produsert for Norwegian Biometric Laboratory og på en bestemt maskin på Høgskolen i Gjøvik. Hensikten med programmet er å kunne vise og forklare enkelt hvordan biometriske data kan brukes for forskjellige personer, både for de som har kjentskap til biometri men også for utenforstående.

Vi har i dette prosjektet produsert to underliggende moduler tilhørende BioDemo; fingeravtrykkskannings og tastaturgjenkjenningsmodulene. Vi har også under hele utviklingen også utviklet med tanke på at det skal være mulig å legge til flere ytterligere moduler ved senere anledning. En av de viktigste oppgavene i dette prosjektet var å implementere de eksisterende og eksterne programmene og funksjonene, med prosjektgruppens programkode. De eksterne programmene bestod i hovedsak av C# og MATLAB.

Summary of Graduate Project

Title:	**BioDemo - Biometric Demostrator**
Date:	15.05.2015
Participants:	Anna Kaardal Joachim Hansen Synne Gran Østern
Supervisor:	Tom Røise, Høgskolen i Gjøvik
Employer:	Patrick Bours /NisLab & NBL
Contact Person:	Prof. Patrick Bours, patrick.bours@hig.no
Keywords:	Biometrics, Fingerprint and keystroke recognition, Interaction Design, Software development, Privacy, Java, Databases
Pages:	152
Attachments:	14
Availability:	Open

Abstract:

BioDemo is a program which demonstrates how biometric data is being scanned, and compared with corresponding data, and which qualities thoose data are based on. The program has been developed for Norwegian Biometric Laboratory, for one specific computer at Gjøvik University College. The main purpose with this program is to be able to easily show biometric data can be used, both for people with biometric knowledge, and for those who have no such knowledge.

We have in this project produced two underlying modules to BioDemo; the fingerprint scan - and the keystroke module. We have developed with further development and adding of modules on a later stage in mind. One of the main tasks in this project has been to implement existing and external software and functions, with the project group's program code. The external programs were mainly C# and MATLAB.

Forord

BioDemo har blitt utviklet av tre studenter fra informasjonssikkerhetsstudiet ved Høgskolen i Gjøvik. Prosjektperioden gikk fra begynnelsen av januar til medio mai 2015.

I løpet av denne intensive perioden har vi fått god hjelp fra flere personer, både internt i HiG og utenfra.

Først og fremst vil vi takke oppdragsgiveren vår, Professor Patrick Bours fra Norwegian Biometric Lab/NisLab ved HiG for at han gav oss denne utfordringen, var veldig imøtekommende, kom med gode innspill og var positiv til våre løsninger. Videre vil vi gjerne takke vår veileder Tom Røise som loset oss gjennom dette prosjektet og kom med gode innspill på både arbeidsmetode og rapportskriving.

Underveis i prosjektperioden fikk vi god bistand fra ekspertise innenfor bestemte felt. Vi vil takke Ole Wattne for hans ekspertise og bistand i utviklingen av det grafiske grensesnittet. Vi vil også takke Guoqiang Li ved NBL for hans ekspertise innenfor fingeravtrykk og for at han bisto oss med verdifull informasjon i forbindelse med utviklingen av fingeravtrykkmodulen.

Gjennom våre brukertester fikk vi god hjelp fra testpersonene våre, og vil gjerne takke dem for at de stilte opp og kom med gode innspill på de grafiske skissene våre.

I tillegg vil vi takke de villige personene som stilte opp på brukertester, og bachelorgruppen Norkart (Per Christian Kofstad, Alf Hammerseth og Ida Granholt) som stilte opp som korrekturlesere og prøve-"sensore".

Anna Kaardal

Synne Gran Østern

Joachim Hansen

Innhold

Forord . . . iii
Innhold . . . iv
Figurer . . . vii
Kodeutdrag . . . ix
1 Introduksjon . . . 1
1.1 Problemområde, avgrensning og oppgavedefinisjon . . . 1
1.1.1 Demostrasjoner før og etter BioDemo . . . 2
1.2 Målgruppe . . . 2
1.2.1 Rapporten . . . 2
1.2.2 Oppgaven . . . 2
1.3 Formål . . . 3
1.3.1 Effektmål . . . 3
1.3.2 Resultatmål . . . 3
1.4 Egen bakgrunn og kompetanse . . . 3
1.5 Rammer . . . 4
1.5.1 Utviklingsprosessen . . . 4
1.6 Øvrige roller . . . 6
1.7 Selve rapporten . . . 6
1.7.1 Terminologi . . . 6
2 Kravspesifikasjon . . . 8
2.1 Funksjonelle krav . . . 8
2.1.1 Felles for alle moduler: . . . 8
2.1.2 Fingeravtrykksmodulen: . . . 8
2.1.3 Fingeråregjenkjenningsmodulen: . . . 9
2.1.4 Tastaturgjenkjenningsmodulen: . . . 9
2.2 Ikke-funksjonelle krav: . . . 10
2.2.1 Organisatoriske krav: . . . 10
2.2.2 Andre krav: . . . 11
2.2.3 Ikke fastsatte krav: . . . 11
2.2.4 Use Case-diagram . . . 11
3 Design og arkitektur . . . 14
3.1 Oppdeling av BioDemo . . . 14
3.1.1 Begrensninger for fingerårermodulen . . . 15
3.2 Andre lignede programmer . . . 15
3.3 Biometri . . . 16
3.3.1 Fingeravtrykk . . . 16
3.3.2 Tastaturgjenkjenning . . . 18
3.4 Arkitektur . . . 20
3.4.1 Model view controller . . . 20
3.5 Database . . . 21

3.5.1 UML for databasene . . . 25
3.6 GUI . . . 27
3.6.1 Utviklingsprosess . . . 27
3.6.2 Designprinsipper . . . 29
4 Programutvikling . . . 37
4.1 BioDemo . . . 37
4.1.1 Programmeringsmønstre . . . 37
4.1.2 Det store bildet . . . 37
4.1.3 HomePanel . . . 44
4.1.4 Registrering av bruker . . . 49
4.1.5 Sletting av midlertidig database . . . 51
4.1.6 Sikring av databasespørringer . . . 52
4.2 Fingeravtrykk . . . 53
4.2.1 Fingeravtrykkmodul . . . 53
4.2.2 Eksterne programmer . . . 58
4.2.3 Filhåndtering knyttet til fingeravtrykkmodulen . . . 61
4.3 Tastaturgjenkjenning . . . 64
4.3.1 Tastaturgjenkjenningsmodul . . . 64
4.3.2 Eksterne programmer og funksjoner . . . 69
4.3.3 Utforfordringer med MATLAB . . . 72
4.3.4 Databasetilpassninger . . . 72
4.3.5 Filhåndtering knyttet til tastaturgjenkjenning . . . 74
5 Testing og kvalitetssikring . . . 76
5.1 Testing av BioDemo . . . 76
5.1.1 Automatiserte regresjonstester . . . 76
5.1.2 Gray Box-testing . . . 76
5.1.3 Review . . . 78
5.1.4 Statisk kodeanalyse . . . 78
5.2 Testing av GUI . . . 79
5.2.1 Brukertesting av skisser . . . 79
5.3 Risikovurdering . . . 80
5.3.1 Første runde . . . 80
5.3.2 Andre runde . . . 81
6 Realisering og implementasjon . . . 83
6.1 Utviklingsmiljø . . . 83
6.1.1 NBL-PC . . . 83
6.1.2 Programmeringsspråk . . . 84
6.1.3 Diverse utfordringer med flerspråks programmer (våre erfaringer) 85
6.1.4 MATLAB-integrasjon . . . 86
6.2 Vurdering av sikkerhetsmekanismer . . . 88
6.3 Feilsituasjoner . . . 89
6.3.1 Håndterte feilsituasjoner . . . 89
6.3.2 Ikke håndterte feilsituasjoner . . . 92
7 Avslutning . . . 94
7.1 Drøftinger . . . 94
7.1.1 Resultater . . . 94

7.1.2 Alternativer . . . 95
7.2 Videre arbeid . . . 95
7.3 Evaluering av gruppearbeidet . . . 96
7.3.1 Disponering av tid . . . 97
7.4 Konklusjon . . . 97
Bibliografi . . . **99**
Vedlegg . . . **101**
A Ganttskjema . . . **102**
B Backlog . . . **103**
C Møtereferater . . . **104**
C.1 Oppdragsgivermøter . . . 104
C.2 Veiledningsmøter . . . 109
C.3 Ekspertmøter . . . 114
D Brukertester . . . **116**
E Risikotabell . . . **119**
F UML for fingeravtrykkmodul . . . **120**
G UML for tastaturgjenkjenningsmodul . . . **121**
H UML for C# programmene . . . **122**
I Gray box testing . . . **123**
J Statisk kode analyse resultat . . . **126**
K GUI-skisser . . . **127**
L Timeliste . . . **135**
M Prosjektkontrakt . . . **136**
N Prosjektplan - BioDemo . . . **139**

Figurer

1 Screenshot av Backloggen i sprint 3 (03 mars - 16 mars 2015) 5
2 Use Case diagram for alle moduler. 13
3 Enkel oversikt over moduler 15
4 Opprettelse av en besøkende gjest. [1, s. 32] 16
5 Fremvisning av ansiktsgjenkjenning [1, s.59] 16
6 Fire ulike eksempler på det grafiske grenesnittet i POLYBIO [2, s. 9] 17
7 Forskjellig kvalitet på samme skannet finger. Bildet er tatt fra [3]. 17
8 Numrering av fingre 18
9 Viser ned og opp knapp og latency mellom knappene 19
10 Hvordan prosesseringen skjer fra inntasting til resultat. Figuren er tatt fra [4] 20
11 MVC implementasjon 21
12 UML modell av testdatabasene. Løsning A øverst, Løsning B nederst. . . . 23
13 Graf over resultatene av testingen 24
14 UML av den endelige konseptuelle designet på databasene 26
15 Et tidlig designutkast på ark 28
16 Nyeste versjon av compare-del 29
17 Nyeste versjon av hovedmeny 30
18 Endelige versjon av tastaturgjenkjenning - Vinduet der brukeren kan legge inn sine tastetrykk. 30
19 Endelige versjon av tastaturgjenkjenning - Viser match etter sammenligning med database. 31
20 Eksempel der enhet og sekvensialitetsprinsippene følges og ikke følges . . 32
21 Viser hva som skjer når man ikke har valgt bruker fra databasen. 33
22 Viser hva som skjer - når ugyldig brukernavn (unknown) ved registrering. 33
23 Viser hvordan knapper er deaktivert før fingeravtrykk har bitt skannet inn. 35
24 Viser at knappene aktiveres når fingeravtrykk har blitt skannet inn. 36
25 Delprogrammene som inngår i demostratoren 38
26 Oppdelingen av HomePanel vist med tabell 44
27 To moduler fordelt på en kolonne 46
28 Tre moduler fordelt på en kolonne 46
29 Fem moduler fordelt på tre kolonner 47
30 Viser hvordan en bruker registreres i databasen. 49
31 Prosessen for å skanne inn en finger for en bruker i databasen 54
32 Prosessen for å sammenligne ett fingeravtrykk med databasen 56
33 HiG Fingerprint Tool 58
34 Viser hvordan analyseverktøyet benyttes. 59
35 UML av databaseutseende for Fingeravtrykk-modulen 60
36 Sekvensdiagram for inntasting av ord 66
37 Viser hvordan XML-filen til brukeren vises. 67
38 Viser grafen som genereres ut fra brukerens inntasting 67

39 Et eksempel på hvordan en graf ser ut. . . . 69
40 Viser C# -programmet DemoKD som startes fra BioDemo . . . 71
41 Den endelige løsningen for implementasjon av MATLAB . . . 72
42 UML av databasen for tastaturgjenkjenningsmodulen . . . 73
43 Utdrag fra oversikt over gjennomførte tester . . . 77
44 Utdrag fra oversikt over resultatene av testene . . . 77
45 Utdrag av de høyeste risikoene i runde 1 . . . 80
46 Utdrag av de høyeste risikoene i runde 2 . . . 81
47 Måling av lese/skrivehastighet på 2 forskjellige PCer . . . 84
48 Viser de tre alternativene gruppen så på for integrasjon av Matlab . . . 87
49 Screenshot fra skannerprogrammet . . . 90
50 En pop-up som forteller brukerne bilder er slettet . . . 91
51 To fingre skannet i samme bilde . . . 93
52 Antall arbeidstime pr gruppemedlem . . . 97
A.1 Gantt-skjema for prosjektet . . . 102
B.2 Backlog i form av user stories for BioDemo . . . 103
E.3 Siste versjon av risikotabellen . . . 119
F.4 Fullstendig UML av klasser, funksjoner og variabler knyttet til fingeravtrykkmodulen . . . 120
G.5 Fullstendig UML av klasser, funksjoner og variabler knyttet til tastaturgjenkjenningsmodulen . . . 121
H.6 Det som står som konfidensielt har ikke vi utviklet. Diagram over alle C# programmene og deres klassediagram . . . 122
I.7 Testdokumentasjon over aktuelle tester . . . 124
I.8 Testdokumentasjon over aktuelle tester . . . 125
J.9 Resultat av oppdagede ”troubling” bugs før fiks . . . 126
J.10 Resultat av oppdagede ”concern” bugs før fiks . . . 126
K.11 GUI-Velkomstskjerm . . . 127
K.12 GUI-Sammenlign før skanning . . . 127
K.13 GUI-Livescan av fingeravtrykk . . . 128
K.14 GUI-Analyse av fingeravtrykk . . . 128
K.15 GUI-Resultat av sammenligningen mot databasen . . . 129
K.16 GUI-Forstørrelse av element i resultatlisten . . . 129
K.17 GUI-Registrer bruker . . . 130
K.18 GUI-Velg finger - ikke implementert . . . 130
K.19 GUI-Valgt finger- ikke implementert . . . 130
K.20 GUI-Skannet finger - ikke implementert . . . 131
K.21 GUI-Analyser skannet finger- ikke implementert . . . 131
K.22 GUI-Compare-del for keystroke . . . 131
K.23 GUI-Brukerinntasting for keystroke . . . 132
K.24 GUI-Viser grafen av brukerens inntasting . . . 132
K.25 GUI-Viser taste-match etter sammenligning med database . . . 133
K.26 GUI-Viser innzoomet match-graf . . . 133
K.27 GUI-Viser lagring av tastetrykk på bestemt bruker . . . 134
L.28 Timeliste for arbeid i prosjekttiden . . . 135

Kodeutdrag

1 Kode som viser hvordan vi slettet og tok tiden etter løsning A 24
2 addComponent: Funksjonen som setter en gridBagLayout for et gitt panel. 45
3 Hvordan et panel i HomePanel legges til 48
4 Utdrag fra sjekk om brukernavn allerde finnes 50
5 Utdrag fra spørringen som sjekker om brukernavnet eksisterer, og i såfall ignorerer forespørselen 50
6 Sletting av tabeller i database 51
7 Sletting av fingeravtrykk-filer tilhørende midleretidige brukere eller ”live-scan” 52
8 Utdrag fra kode med preparedStatements som brukes i forbindelse med brukerinput. 53
9 Starter fingeravtrykkskannerprogrammet og venter til den er avsluttet . . 55
10 SQL-spørring for å hente ut alle fingeravtrykk i databaser 57
11 Funksjon som endrer navn på fingeravtrykksfilene 62
12 Funksjon som sletter fingeravtrykk med dårlig kvalitet 63
13 Viser arrayen i Utilities-klassen som angir hvilke ord som kan tastes inn av brukeren. 64
14 Viser hvordan et ord legges til på brukeren i databasen. 65
15 Matlab-funksjonen oneLineGraph 68
16 Matlab-funksjonen threeLineGraph 70
17 Utdrag som viser hvordan DemoKD kalles, og xml- dokument opprettes . . 71
18 Eksempel på en XML-fil til en bruker i databasen 74
19 Kode for å sikre at det ikke kommer en null pointer exception når en databaseforbindelse stenges 78
20 String håndtering av locale dependent verdier 86
21 Håndtering av feilsituasjonen i DemoKD.exe 92

Ordliste og forkortelser

API Application Programming Interface - grensesnittet mellom komponentene
DLL Dynamic-link library. En sammensetning av kode og bibliotek som kan utnyttes av forskjellige ressuerser samtidig. I BioDemo blir dll brukt til å kommunisere mellom funksjoner av forskjellige språk (eks Java og C#)
Duration Benyttes i tastaturgjenkjenning. Tastevarighet, altså tiden brukeren holder en knapp nede.
Gray Box Testing Kombinasjon av White Box og Black Box testing.
GUI Graphic User Interface. Grafisk brukergrensesnitt.
HIG Høgskolen i Gjøvik.
Jar-fil Java Archive. En kjørbar java-fil.
JRE Java Runtime Environment. Nødvendig for å kunne kjøre Javaprogrammer.
Latency - Benyttes i tastaturgjenkjenning. Tiden (evt. ventetiden) det tar fra en knapp slippes til neste knapp trykkes ned.
Live Scan eller livescan; Betegnelsen for skanning av fingeravtrykk med hjelp av digitale verktøy. Brukes i denne rapporten også om tastatugjenkjenning og når en bruker skriver ordet en gang.
MBJA MATLAB Builder JA - En ”brobygger” mellom MATLAB-objekter og Java. Pakker om MATLAB-objekter til kjørbare jar-filer, og kan brukes i Javaprogrammer.
MBNE MATLAB Builder NE - En ”brobygger” mellom MATLAB-objektet og .NET.
Minutia Betegnelsen for hovedegenskaper ett fingeravtrykk har. Flertall: Minutiae
NBL The Norwegian Biometric Lab, underlagt NisLab
NISLab Norwegian Information Security Laboratory. Forskningslab knyttet til HIG.
Signed byte Representerer både positive og negative tall.
TFU Test før utvikling. En modell der man skriver testen før funksjonen. Fra engelsk: TDD - ”Test Driven Development”
Unsigned byte Representerer kun positive tall.
Wrapper I programvare, underrutiner som har som eneste hensikt å kalle andre underrutiner.
XML Extensible Markup Language. Et markeringspråk som strukturerer data hierarkisk og er både lesbart for mennesker og maskiner.

1 Introduksjon

Biometri brukes i informasjonssikkerhet som identifisering og verifisering av personer basert på målbare fysiske egenskaper. I de siste årene har denne måten å identifisere og verifisere personer blitt mer og mer brukt innenfor vidt forskjellige områder. Dagens smarttelefoner har skjermlås knyttet til fingeravtrykklesere, og flere og flere flyplasser begynner å implementere irisgjenkjenning for reisende.

Oppgaven vår består av å utvikle et program for Norwegian Biometric Laboratory (NBL) som skal demostrere flere biometriske sammenligninger og analyse, og benyttes når ansatte ved NBL ønsker å vise hvordan dette foregår.

BioDemo vil fungere som en felles portal for forskjellige biometriske områder. Det er et mål å gjøre biometri til et mer kjent begrep blant befolkningen, og derfor er det viktig at vi utvikler et grafisk brukergrensesnitt som er forståelig og brukervennlig, også for personer utenfor NBL. Programmet som utvikles skal benytte seg av flere eksisterende programmer, både utviklet av NBL selv og andre linsensierte programmer. En viktig del av vår oppgave vil derfor også være å sikre kommunikasjon mellom BioDemo og disse eksterne programmene, på tvers av forskjellige programmeringsspråk, samt tilpasse dem til BioDemo, både med tanke på det grafiske og funksjonalitet.

1.1 Problemområde, avgrensning og oppgavedefinisjon

Det er en utfordring for ansatte ved NBL å demostrere for andre enn forskerne selv hva biometriske data faktisk er, hva det arbeides med hos NBL, og hvordan biometriske data blir behandlet. NBL og NisLab er en av de mest anerkjente miljøene innenfor informasjonssikkerhet og får mange besøkende hvert år. I desember 2014 var Statsminister Erna Solberg på besøk, og fikk da en enkel demostrasjon hos NBL. [5] Video av hvordan vår oppdragsgiver demostrerte hvordan fingerårer ble skannet for Erna Solberg finnes på denne lenken:[1] Det er i slike situasjoner vårt program skal benyttes, og en av målene er å gjøre denne demostrasjonen enkel og forståelig.

NBL arbeider og forsker på mange områder innenfor biometri, som for eksempel; fingeravtrykk, fingerårer, tastaturgjennkjenning, tanngjennkjenning og irisgjennkjenning. Med den tiden gruppen har hatt til dispsosisjon var det ikke tid til å ta for seg alle disse områdene. Derfor ba gruppen oppdragsgiver om å velge ut noen ”må-ha” moduler. Det resulterte i at gruppen fikk tre områder som fikk første prioritet, for deretter å se om det ble tid til flere områder. de tre ”må-ha” - modulene ble som følger:

- Fingeravtrykk
- Fingerårer
- Tastaturgjenkjenning

[1] http://www.oa.no/Erna_p__Gj_vik_visitt-5-35-9806.html

1.1.1 Demostrasjoner før og etter BioDemo

Som nevnt i kapittel 1.1, var Statsminister Erna Solberg på besøk hos NBL. I denne delen vil vi bruke hennes besøk og fingeravtrykkskanning som eksempel for å illustrere hvordan BioDemo forenkler oppgaven med å demonstrere hvordan innsamling og sammenligning av biometriske kjennetegn foregår hos NBL.

Før BioDemo ble tatt i bruk:
Da Erna kom på besøk til NBL, ble hun introdusert for de forskjellige sensorene, og fikk teste ut disse. Da turen kom til fingeravtrykkskanning, og hun skulle registrere fingeravtrykk, ble hun presentert for et program som skanner inn fingeravtrykk. Fingeravtrykk ble skannet inn, og lagret i en mappe. For å finne tilbake til hennes fingeravtrykk, måtte en NBL-ansatt manuelt gå gjennom mappen på jakt etter det riktige fingeravtrykksbildet. Det ble generert noen uforståelige tall, og det var vanskelig for Erna å forstå hva som akkurat skjedde, og hva resultatet ble. En representant fra NBL måtte hele tiden forklare hva som foregikk, hva som skulle gjøres, og tegne opp dette for å forsøke å demonstrere og gjøre denne prosessen forståelig for Erna. Erna satt antagelig igjen med en følelse av lett forvirring, og visste ikke med sikkerhet hva som skjedde videre med hennes biometriske kjennetegn.

Etter at BioDemo har blitt tatt i bruk:
Erna blir nå presentert for et intuitivt og oversiktelig brukergrensesnitt som fører henne gjennom hele fingeravtrykkssprosessen, fra fingeravtrykket skannes inn, til det sammenlignes opp mot databasen og hun får opp en liste med de fingeravtrykkene som matcher hennes. Hun kan også registrere seg som bruker og sammenligne fingeravtrykket hun skanner inn med disse, og få se alle fingeravtrykkene hun har skannet inn. Hun har selv kontroll over prosessen, og velger selv hvilke oppgaver som skal utføres, og når disse skal utføres ved å klikke på bestemte knapper. Erna sitter igjen med en følelse av å selv ha tatt del i prosessen. I tillegg kan hun være trygg på at hennes biometriske kjennetegn blir slettet fra databasen etter at demonstrasjonen er ferdig, ettersom hun selv valgte å registrere seg midlertidig.

1.2 Målgruppe

Vi har i dette prosjektet to forskjellige målgrupper; en for rapporten og en for programmet.

1.2.1 Rapporten

Målgruppen for rapporten er personer med IT-bakgrunn, med hovedfokus på andre IT-studenter og ansatte ved HiG. Derfor vil vi skrive denne rapporten basert på at de som leser har grunnleggende IT-kunnskaper. Det er et mål å skrive rapporten på en slik måte at det er forståelig og interessant for denne målgruppen. I tillegg ønsker vi at rapporten kan benyttes ved et eventuelt videreutvikling av vårt program. Denne rapporten er også et vuderingsgrunnlag for vår bacheloroppgave, og vi har av den grunn vært optatt av å argumente og diskutere våre valg og løsninger for å gi grunnlag for denne vurderingen.

1.2.2 Oppgaven

Målgruppen for programmet er først og fremst oppdragsgiver. I og med at deler av kildekoden (noen av de eksterne programmene) ikke kan inkluderes i rapporten eller deles

åpent, er det ikke aktuelt i første omgang å distribuere programmet videre utover NBL. Det kan derimot være aktuelt at andre skal videreutvikle eller endre de eksisterende modulene som gruppen utvikler. I den forbindelse vil det være aktuelt for gruppen å gjøre det enklest mulig for utenforstående å sette seg inn i koden slik at den kan videreutvikles.

En annen undermålgruppe for BioDemo er de eksterne personene som kanskje ikke kjenner til biometri fra før og skal kunne få en demonstrasjon av dette sammen med en ansatt fra NBL eller en som kjenner programmet.

1.3 Formål

1.3.1 Effektmål

Hovedformålet for dette prosjektet er at oppdragsgiver skal sitte igjen med et program som kan demonstrere biometri enkelt og forståelig. Dette gjelder både for besøkende, gjester og andre personer utenfor NBL og NisLab som ikke kjenner til biometri og hva NBL jobber med. Forskningen som skjer hos NBL fokuserer mest på å lage så nøyaktige og korrekte biometriske analysealgoritmer som mulig, og ikke på den grafiske visualiseringen. Vårt program skal øke forståelsen og kunnskapen rundt biometrifeltet til en større gruppe mennesker ved å vise hva biometri er og hvordan det fungerer. Dette skal skje ved å gjenskape hvordan biomtetriske data blir innsamlet, vise hvordan disse dataene blir brukt til identifisering og autentisering og hvilke egenskaper som blir brukt til dette.

1.3.2 Resultatmål

Ved prosjektets slutt den 15. mai 2015, skal programmet minst inneholde de tre moduelen nevnt i kapittel 1.1. Innad i disse modulene skal programmet være i stand til å innhente biometriske data, lagre disse, og kunne sammenligne og analysere med andre lignende data. Videre skal programmet utvikles på en slik måte at det skal være lett for videreutvikle, utvide og legge til ny funksjonalitet.

1.4 Egen bakgrunn og kompetanse

Alle de tre gruppemedlemmene studerer bachelor i informasjonssikkerhet og har dermed mye av det samme kunnskapsgrunnlaget. Gruppen har gjennom utdannelsen gått gjennom blant annet fagområdene programmering, operativsystemer, databaser, programutvikling og systemutvikling, i tillegg til fagområder innenfor informasjonssikkerhet. Mye av kunnskapen er ganske teoretisk, men vi vil få bruk for mye av det vi har lært hittil. I tillegg har en av gruppemedlemmene erfaringer med programmering i C#. Og en annen har erfaring med større prosjektarbeid, leveranser og gjennomføring i forbindelse med deltidsjobb.

Av biometrikunnskaper var det ingen av gruppemedlemmene som hadde noen kunnskaper eller erfaringer utover veldig grunnleggende kunnskaper om hva biomteri er. Alle kjente til NBL og litt om hva slags forskning de arbeider med gjennom noen av deres presentasjoner ved HiG, men satt ikke på noen detaljkunnskaper.

Dette prosjektet var et av de største prosjektene gruppen har vært borti hittil, og er det største programmet vi har utviklet. Vi var også ganske ferske på å ha en oppdragsgiver som satte kravene vi skulle rette oss etter. For å kunne produsere et godt produkt, måtte

vi tilegne oss mye ny kunnskap, i tillegg til å styrke vår eksisterende kunnskap.

1.5 Rammer

Tidsperioden for prosjektet løp fra 7. januar 2015 til 3. juni 2015. I januar ble planleggingen og informasjonsinnhentingen gjennomført, og de siste ukene i mai ble brukt til å sluttføre rapporten og produktet før endelige innlevering 15. mai 2015. 3. februar til 5. mai ble satt av til utvikling. I denne perioden foregikk programutvikling parallelt med rapportskriving. Tiden etter 15. mai gikk i hovedsak med til planlegging og forberedelser til presentasjonen i begynnelsen av juni.

Gruppen fikk tildelt en PC fra NBL (NBL-PCen), som BioDemo skulle kjøres på. PCen ble plassert i et adgangskontrollert rom ved siden av NBL sine lokaler på HiG, og dette kunne vi benytte som grupperom i prosjekttiden. Programutviklingen foregikk både på NBL-PCen, og på våre private PC'er. For utvikling på våre private PC'er brukte vi Bitbucket som versjonskontroll og samarbeid, men også som lagringsplass for arbeidsdokumenter og rapporten. NBL-PCen kunne ikke kobles til internett, ei heller flyttes på grunn av sikkerheten: Sensitiv data ble lagret på den, og ingen andre sikringstiltak enn påloggingspassord var tilstede, og samtidig kjørte den Windows XP. Gruppen måtte derfor benytte minnepenner og flytte over programfiler manuelt.

Vedlegg A viser Ganttskjema for hele prosjektperioden. Her vises at rapportskriving foregår parallellt med programutviklingen.

1.5.1 Utviklingsprosessen

Både oppdragsgiver og gruppen var enige at en modell som støtter endringer underveis i prosjektetarbeidet var det beste for prosjektet. Dette fordi vi så for oss at det ville bli behov for å gjøre endringer i kravspesifiksjon underveis, og at det derfor ville være en fordel med fleksibilitet rundt dette. Det ville også være viktig for oss at forskerne ved NBL kunne ta del i prosjektet underveis, slik at vi kunne utvikle et program som oppfyller deres behov, og samtidig unngå større misforståelser. I tillegg måtte utviklingsmodellen også ha rom for at det allerede fantes programmer og kode som BioDemo skulle implementere.

Prosjektet var som nevnt tidligere, vårt største utviklingsprosjekt, og det var første gang vi jobbet etter oppdragsmodell med en faktisk kunde. Dette i tillegg til at vi ikke hadde så mye erfaring med programutvikling, medførte at det hadde vært utfordrende å detaljbestemme på et tidlig stadium hva vi skulle gjøre og levere på bestemte tider i løpet av prosjektet.

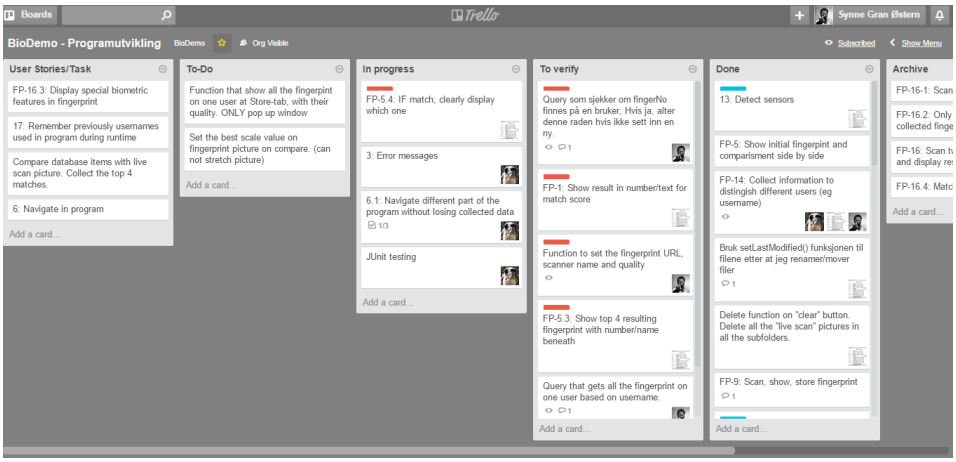
På bakgrunn av dette så vi at de plandrevne modellene ikke var aktuelle for oss, og vi trengte en agil modell. Agile modeller gir rom for fleksibilitet og oppdragsgiver kan ta en mer aktiv del i prosjektet i utviklingen og komme med endringer og innspill underveis. I tillegg gir agile modeller muligheter for delleveranser i inkrementer, noe som var hensiktsmessig for vårt program da det fra et tidlig stadium ble delt i moduler, for å sikre at vi var i stand til å levere noen moduler som fungerte enkeltstående, før vi leverte den siste stabile versjonen av BioDemo i mai 2015. På grunn av alle de eksisterende program-

mene og kodefunksjonene var det naturlig å implementere en form for gjenbruksmodell. Her var det viktig å kunne tilpasse BioDemo etter de eksterne programmene ved å finne løsninger som passet, modifisere om det skulle bli nødvendig, integrere og lage funksjonaliteter rundt.

Ut fra de agile metodene vi kjente til, vurderte vi XP, Scrum og RUP. Valget falt på Scrum, som ble tilpasset gruppens behov.

Scrum er en utviklingsmodell som består av blant annet av sprinter og product backlog. Viktige roller i Scrum er Product Owner, som hos oss var oppdragsgiveren- ved Patrick Bours og Scrum Team som består av gruppens tre medlemmer. Scrum benytter seg av en product backlog der Product Owner velger ut de elementene som skal jobbes med i neste sprint.

Gruppen valgte å løse denne med en grafisk presentasjon - og utvelgelse basert på dette, i stedet for å gå gjennom backlogen. Dette fungerte på den måten at vi på et Sprint update- møte, enten på papir eller på PC viste det grafiske av programmet og oppdragsgiver fortalte og påpekte ting som måtte gjøres. Utifra dette justerte vi backlogen slik at den tilsvarte de arbeidsoppgavene som var diskutert på møtet med oppdragiver. I fig. 1 ser vi et utklipp av backloggen slik den så ut i Sprint 3. Vi brukte verktøyet Trello [6] for å holde oversikt over backloggen og status. I tillegg valgte vi å ta med prinsipper fra XP, som parprogrammering i noen av arbeidsoppgavene og felles eierskap til kode. I tillegg valgte gruppen å utvikle moduler noenlunde inkrementelt, i den forstand at vi hadde hovedfokus på utviklingen av en og en modul av gangen, med noe overlapping (mer om dette i kapittel 3.1). Ellers foregikk utviklingen i henhold til Scrum, og gruppen hadde underveis i utviklingen deleveranser av programmet, med demostrasjoner for oppdragsgiver.



Figur 1: Screenshot av Backloggen i sprint 3 (03 mars - 16 mars 2015)

Når det gjaldt sprintlengde ble sprinter på en, to og tre uker vurdert. Gruppen trengte en sprintlengde som var kort nok til at vi kunne få jevnlige tilbakemedlinger fra oppdragsgiver, men ikke for korte, slik at gruppen ikke ”rakk” å gjøre noe før neste Sprint

Update-møte. Med en ukes sprinter fikk vi ofte møter med oppdragsgiver, men kunne risikere å ha noen uker der vi var mer eller mindre på samme stadium som forrige sprint og ikke kunne ”gå videre” med backloggen. Med tre ukers sprinter ville vi ha rukket å få gjort mye mer, men vi ville ha sjeldere møter med oppdrasgiver, noe som kunne medføre at og hans innspill kom for sent, og at vi måtte gjøre om mye av vårt arbeid. Derfor falt valget på to ukers sprinter.

Hver sprint var på to uker (ti arbeidsdager), med møte med oppdragsgiver (og evt. andre aktuelle personer fra NBL) ved begynnelse/slutt av hver sprint.

Første sprint startet i begynnelsen av februar og siste ble ferdig i begynnelsen av mai. Se vedlegg A for å se hvordan de seks sprintene ble fordelt utover prosjekttiden.

1.6 Øvrige roller

Veileder for dette prosjektet var Tom Røise ved IMT/HiG, som underviser blant annet innenfor systemutvikling og er studieprogramansvarlig for Bachelor i Programvareutvkling.
Oppdragsgiver for prosjektet var Patrick Bours, leder for NBL og underviser blant annet i flere biometrirelaterte fag.
På fingeravtrykk-modulen fikk vi også hjelp fra Guoqiang Li, PhD kandidat hos NBL.

1.7 Selve rapporten

Vi valgte å strukturere rapporten i sju kapitler, og disse er som følger:

1. Introduksjon - som presenterer oppgaven, mål og målgrupper, hvilke rammer vi måtte forholde oss til, roller og valget av utviklingsmodell.
2. Kravspesifikasjonen, som presenterer funksjonelle- og ikke-funksjonelle krav for programmet.
3. Design og arkitektur - som presenterer fingeravtrykksskanning og tastegjennkjenning og forklarer hvordan vi valgte å organisere og designe programmet.
4. Programutvikling - Som forklarer hvordan vi valgte å inplementere løsningene i kapittel 3.
5. Testing og kvalitetssikring - som beskriver hvordan vi valgte å teste programmets funksjonalitet og sikre at det oppfylte kravene i kapittel 2, samt risikovurdering.
6. Realisering og implementasjon - som beskriver utviklingsmiljøet, vurdering av sikkerhetsmekanismer, integrasjon mellom eksterne programmer, feilsituasjoner - og hvordan vi valgte å forholde oss til disse.
7. Avslutning - der vi oppsummerer prosjektet, evaluerer arbeidet vårt, presenterer mulig videreutvikling av programmet og konkluderer.
8. Vedlegg

1.7.1 Terminologi

- *BioDemo:* Navnet ”BioDemo” om programmet som ble utviklet. Dette programmet består av mange moduler og eksterne programmer, der noen av disse er underlagt lisenser.
- *Bruker:* ”Bruker” benyttes når det er snakk om begge grupper av brukere (NBL og gjester) av BioDemo.

- *NBL-bruker:* Dette er betegnelesen for den som skal holde demonstrasjoner ved hjelp av BioDemo. Dette er i et typisk tilfelle en NBL-ansatt, og derav navnet.
- *Gjeste-bruker:* Gjestene som demostrasjonen foregår for.
- *NBL-PCen:* PCen som vi fikk tildelt av NBL for å utvikle og kjøre programmet på. Den er innelåst i et rom på høgskolen.

2 Kravspesifikasjon

I denne delen tar vi for oss de aktuelle funksjonelle og ikke-funksjonelle kravene for vårt prosjekt. Tallene i parentes er en referanse til nummeret på aktuelt backlog-element. Oversikt over hele backlogen er å finne i vedlegg B. I likhet med resten av prosjektet, ble kravspesifikasjonen utviklet i henhold til Scrum. Det var også et ønske fra oppdragsgiver at han skulle kunne komme med endrede krav underveis i utviklingsprosessen. Selv om en stor andel av kravene har stått fast i prosjektet, ble det også behov for noen endringer og justeringer underveis for å sikre at endrede ønsker ble fulgt, men også utfordringer i forbindelse med driftsmiljø. Dette kan det leses mer om i selve kravspesifikasjonen.

2.1 Funksjonelle krav

I denne delen presenteres programmets funksjonelle krav.

2.1.1 Felles for alle moduler:

- Brukeren skal som første punkt i programmet presenteres for en startskjerm der det kan velges hvilken modul å starte med (fingeravtrykksskanning, fingeråreskanning eller tastegjenkjenning.) (1,4).
- Brukeren skal når som helst i programmet ha mulighet til å registrere seg midlertidig eller permanent. (14). Her skal brukeren kunne registrere brukernavn, fornavn, etternavn, kjønn, fødselsdato og oppgi om vedkommende ønsker å lagres midlertidig (slettes når programmet avsluttes), eller lagres permanent (blir værende selv om programmet lukkes og startes på ny). De obligatoriske feltene som skal fylles ut er: brukernavn og fornavn, samt å velge om man vil registrere seg midlertidig eller permanent. Øvrige felter skal være valgfrie å fylle ut. (9)
- Brukeren skal kunne bevege seg mellom forskjellige deler av programmet uten at data tapes.(6).
- BioDemo skal opprette databasetilkobling ved oppstart (7). Den midlertidige databasen skal også opprettes ved hver programstart.
- Programmet skal holde på innkommende data så lenge programmet kjøres, eller overskrives av nye brukerdata. Det vil si at brukeren skal kunne forflytte seg mellom de forskjellige de forskjellige delene av programmet, uten at data tapes.
- Det skal sikres at data for midlertidige brukere skal slettes for hver gang programmet lukkes. Ved hver oppstart skal det også utføres en sjekk på om det fremdeles finnes midlertidige data. Hvis dette er tilfelle, skal disse slettes. Permanente data skal bli værende, også ved programslutt og oppstart.

2.1.2 Fingeravtrykksmodulen:

- Brukeren skal kunne registrere ti fingeravtrykk på sin bruker (en for hver finger).
- Brukeren skal ha mulighet til å gå direkte til livescanning uten å registrere seg først.
- Brukeren skal kunne velge å analysere sitt fingeravtrykk og få opp minutiae-punkter og informasjon om avtrykkets kvalitet. (10)
- Brukeren skal kunne sammenligne sitt fingeravtrykk med databasen, og få opp topp

4 match over de fingeravtrykkene med høyest match. Hver match skal vises sammen med prosent match, brukerinfo, hvilken finger det er snakk om, samt hvilken skanner som ble brukt. Virkelig match skal markeres med grønn ramme, mens øvrige markeres med rød ramme. (15)

For at match skal markeres med grønn, må matchtallet overstige 48.

- Brukeren skal kunne klikke på hver match og få opp et mer detaljert (og forstørret) bilde som viser målepunktene på fingeravtrykket.
- Brukeren skal ha muligheten til å begynne forfra med blank skjerm og skanne inne et nytt avtrykk.

2.1.3 Fingeråregjenkjenningsmodulen:

- NBL-bruker skal ha muligheten til å registrere sine fingerårer i den permanente databasen. (23)
- Brukeren skal kunne gå rett til en sammenligning, og starte fingeråreskanningen uten å trenge å registrere seg. (19)
- Brukeren skal kunne få se livescan av sine fingerårer. (26)
- Etter live scan skal brukeren kunne velge å sammenligne sitt bilde med de andre lagrede fingeråbildene, og deretter få opp en liste med topp fire match. (21, 25)
- Brukeren skal kunne begynne forfra og få muligheten til å starte fingeråreskanningen på ny. (20)

2.1.4 Tastaturgjenkjenningsmodulen:

- Brukeren skal kunne velge et ord fra databasen som skal brukes til tastegjenkjenningsdemonstrasjonen. Disse ordene er bestemt av opdragsgiver. (29, 36)
  Ordene som bruker skal kunne velge mellom er:
  welcome42
  password
  Password
  Aidv50Xv
  Longer sentences are also possible
  123456
- Bruker skal kunne registrere sine tastetrykk i den permanente databasen. Dette skal foregå ved at brukeren velger det ordet fra databasen som vedkommende vil bruke, og deretter taster inn dette ordet ti ganger, slik at et tastegjennomsnitt kan genereres. Dette fordi det for å kunne lage gjennomsnitt være mange nok tallverdier å regne ut fra. (37)
- Brukeren skal, slik som i fingeravtrykkmodulen kunne gå rett til sammenligning og starte tastaturgjenkjenning uten å måtte registrere seg. Her skal det komme opp en beskjed når det klart for å registrere ord, og hvilket ord som skal registreres. (38)
- Etter inntasting skal brukeren få opp en graf som viser sine inntastede verdier.
- Når inntastingen er ferdig, skal brukeren få muligheten til å sjekke sitt tastemønster opp mot de andre lagrede tastemønstrene. (30)
- Sammenligningen skal baseres på ventetid mellom knappene og tastevarighet.
- Når brukeren har valgt å sammenligne tastetrykk, vil det komme opp en graf som viser brukerens egen tasting (blå strek) og to røde streker som representerer maximums- og minimumsmåling for den inntastingen som ligner mest.(32)
- Brukeren skal ha muligheten til å starte forfra med inntastingsprosedyren. (28)

2.2 Ikke-funksjonelle krav:

I denne delen presenteres programmets ikke-funksjonelle krav.

2.2.1 Organisatoriske krav:

Krav til driftsmiljø:

BioDemo skal kjøres på en PC som er plassert i NBLs lokaler. Dette er en PC som kjører Windows XP og har ytelsesutfordringer, og på bakgrunn av dette må vi ta flere hensyn, både med tanke på sikkerhet, kompitablitet og ytelse:

- BioDemo skal kunne kjøres på Windows XP og senere Windows OS.
- BioDemo skal være kompitabel med 32-bits system.
- Microsoft tilbyr ikke lenger support av Windows XP, og dette medfører at Windows XP blir fem ganger så sårbar for sikkerhetstrusler, et tall som bare vil fortsette å stige [7]. Et krav fra oppdragsgiver er derfor at denne maskinen ikke skal kobles til nettet.
- Det ble utført målinger av lese- og skrivehastighet på harddisken. Hvordan disse målingene ble utført kan det leses om i kapittel 47a. Under ser man et utdrag av resultatene for disken i NBL-PCen:

	Lesehastighet (MB/s)	**Skrivehestighet (MB/s)**
Sekvensiell	63,17	62,05
Random	0,363	0,966

  På bakgrunn av dette måtte det tas hensyn til ytelsesproblematikk, og tyngre sikkerhetsmekanismer og andre mekaniser som kan medføre uakseptabelt treg responstid må derfor utelates. Her skal gruppen i samråd med oppdragsgiver ta en vurdering på hvilke sikkerhetsmekanismer det er behov for, i henhold til risikobildet.
- På grunn av ingen tilkobling til internett og sikkerhetshensyn skal databasene (gjestedatabase og permanent database) lagres lokalt på NBL-PCen.
- BioDemo skal skrives i Java, men også benytte seg av eksterne programmer som er skrevet i MATLAB og C#. BioDemo skal derfor kunne kommunisere melom Java og C# og MATLAB. For å kunne kommunisere med MATLAB må Java SE-1.7 benyttes.
- Ved overlatelse av BioDemo skal det også lages en kjørbar fil, slik at brukere kan klikke å for å starte programmet og slipper å måtte gå inn i koden og kjøre den.

Eksterne krav:

Lovverk:

Personopplysningsloven [8] setter begresninger for lagring og behandling av personopplysninger. Biometriske kjenentegn kommer inn under kategorien «andre entydige identifikasjonsmidler», og derfor må vi forholde oss til aktuelle paragrafer i forbindelse med lagring og behandling av biometriske data:

- §8 – «Personopplysninger kan bare behandles dersom den registrerte har samtykket, eller det er fastsatt i lov at det er adgang til slik behandling».
- §9- «Sensitive personopplysninger kan bare behandles dersom behandlingen oppfyller ett av vilkårene i §8 og a) Den registrerte samtykker i behandlingen»
- §12 - «Fødselsnummer og andre entydige identifikasjonsmidler kan bare nyttes i

> behandlingen når det er saklig behov for sikker identifisering og metoden er nødvendig for å oppnå slik identifisering. Datatilsynet kan pålegge en behandlingsansvarlig å bruke identifikasjonsmidler som nevnt i første ledd for å sikre at personopplysningene har tilstrekkelig kvalitet.»

Biometriske kjennetegn skal kun registreres på frivillig basis. Personene som skal være med på demonstrasjonen må samtykke i at deres biometriske kjennetegn lagres midlertidig i databasen under demonstrasjonen og at denne informasjonen slettes i forbindelse med at programmet avsluttes. I tilfeller der opplysninger skal lagres permanent i databasen, må det gis skriftlig tillatelse av eieren. Det vil være NBL sin oppgave å sørge for og oppbevare slike skjemaer for samtykke.

Lisensering:
BioDemo vil bestå av flere eksterne programmer. Noen av disse er tilknyttet lisenser, og kan på bakgrunn av dette ikke vises åpent. Bruk av flere programmer vil også begrenses til NBL-PCen. Et annet punkt er at det legges begrensninger på hvilke endringer og tilpassninger vi kan gjøre på programmene, noe som medfører at programmene må brukes som de er. Dette er noe gruppen må forholde seg til.

GUI:

Brukeren skal presenteres for et intuitivt og oversiktelig brukergrensesnitt (18). Vi skal benytte oss av grunnleggende designprinsipper i utviklingen av BioDemo. Disse har vi forklart i kapittel 3.6.2.

2.2.2 Andre krav:

Et krav fra oppdragsgiver var at det skulle lages brukermanual for videreutvikling av BioDemo.

2.2.3 Ikke fastsatte krav:

Backup var ikke et krav fra oppdragsgiver. Det blir derfor opp til gruppen å vurdere om det er behov for dette tiltaket, og i såfall hvordan vi velger å løse dette.

Oppdragsgiver satte heller ingen krav til universell utforming - slik som tilpassing for fargeblinde eller synshemmede.

2.2.4 Use Case-diagram

Use case-diagrammet under (fig.2) viser hvilke funksjoner i kravspesifikasjonen som tilhører hvilke moduler i BioDemo, samt hvilke av disse funksjonene som utføres av eksterne programmer. Her ser man at modulene har forskjellige funksjonaliteter, og at det benyttes flere eksterne programmer for å utføre bestemte oppgaver. I BioDemo anses de eksterne programmene som de analyse- og skanneprogrammene som BioDemo skal kommunisere med, og som gruppen ikke har utviklet.

Fingeravtrykksmodulen:
Use casene som tilhører denne delen av programmet har fargen blå.
For fingeravtrykksmodulen ser man at det skal være mulig å skanne inn et fingeravtrykk. Selve skanningen av fingeravtrykket utføres av det eksterne programmet HiG Finger-

print Tool. Etter innskanningen, kan brukeren analysere fingeravtrykket sitt (kvalitet og minutiae-punkter), og her benyttes et eksternt analyseprogram. Etter dette, skal brukeren få opp resultatet av sammenligningen. Brukeren skal også kunne registrere ti fingeravtrykk (ett pr. finger) på sin bruker, samt fjerne data som befinner seg på skjermen, slik at vedkommende kan starte skanningen forfra.

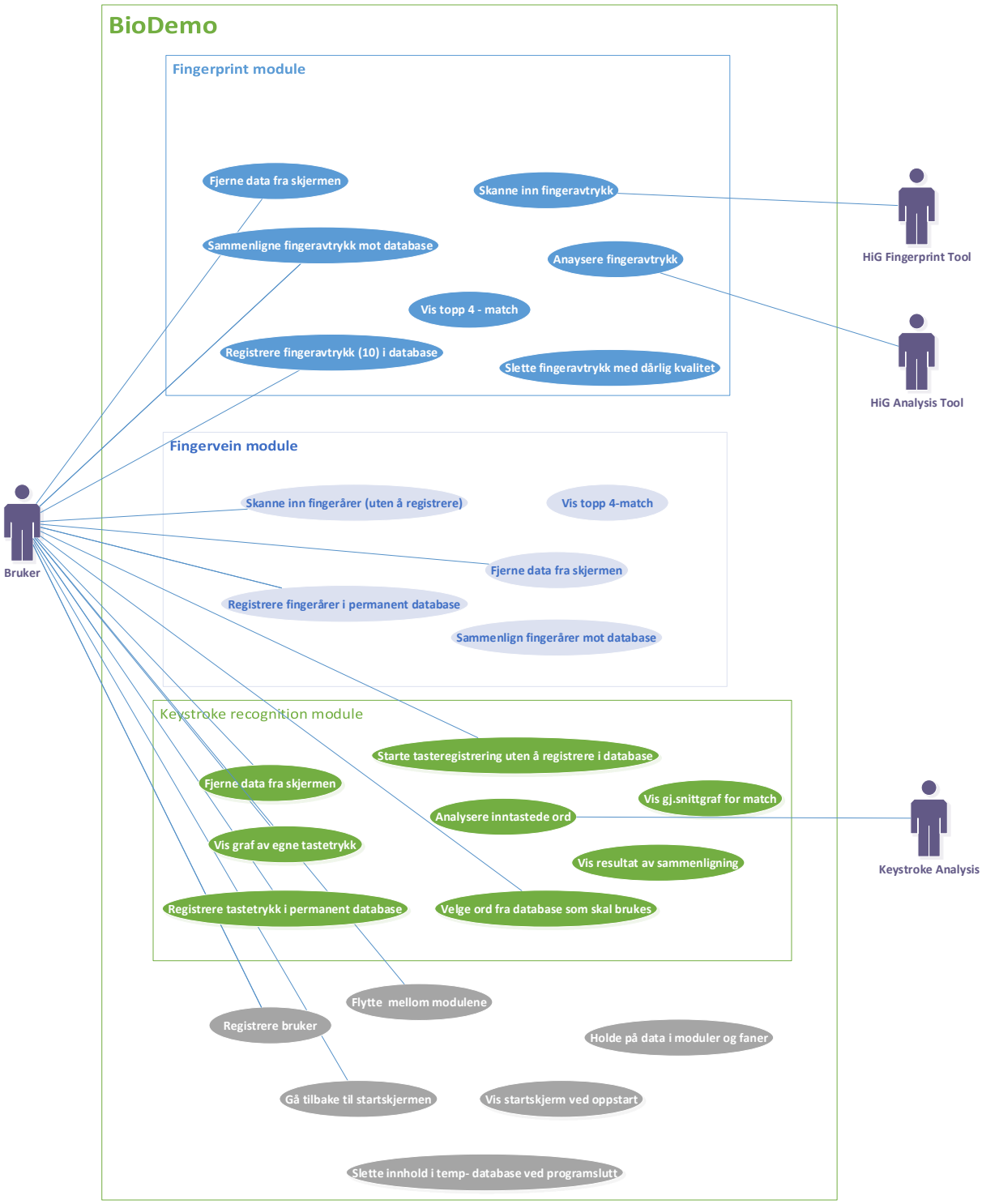
Fingeråremodulen:
Use casene for denne delen av programmet har fått fargen lys blå/grå i figuren under.
For fingeråremodulen skal det være mulig for brukeren å skanne inn sine fingerårer, og deretter få sammenlignet dette mot andre lagrede brkere. Til slutt får brukeren opp resulatet fra sammenligningen som en toppliste. Brukeren skal også kunne registrere sine fingerårer på sin bruker, og fjerne data som befinner seg på skjermen og dermed kunne starte forfra med skanneprosessen.

Vi rakk ikke å begynne på fingeravtrykksmodulen, og derfor hadde vi ingen info om hvilke eksterne programmer som skulle benyttes, noe som medfører at use case-diagrammet mangler disse aktørene. Programflyten i fingeråremodulen ville ellers ha foregått mye på samme måte som i fingeravtrykksmodulen.

Tastaturgjenkjenningsmodul:
Use casene for denne delen av programmet har i figuren under fått fargen grønn.
For tastaturgjenkjenningsmodulen ser man at det skal være mulig for brukeren å starte tastaturgjenkjenning. Derfra benyttes et eksternt anlyseprogram for å analysere inntastet data. Det genereres så en graf for brukerens inntastede ord og en gjennomsnittsgraf. Brukeren skal også her kunne registrere tastetrykk på sin bruker, fjerne data fra skjermen, så vedkommende kan starte inntasting på ny. I tillegg skal brukeren kunne velge fra en forhåndsdefinert liste med ord, hvilket ord vedkommende ønsker å registrere.

De delene av programmet som er felles for alle moduler:
Use casene i figuren med grå farge er felles for hele programmet. Brukeren skal når som helst i programmet kunne registrere seg som bruker. Vedkommende skal også kunne forflytte seg ulike deler av programmet og gå tilbake til startskjermen som vedkommende ønsker.

Programmet inneholder også deler som ikke trigges av brukerinteraksjon, men som skjer automatisk. Disse use casene representerer følgende: at brukeren får opp startskjermen ved oppstart, at data blir værende ved forflytting, samt sletting av data i den midlertidige databasen ved oppstart og programslutt. I tillegg skal det i fingeravtrykksmodulen automatisk slettes innskannede fingeravtrykk med for dårlig kvalitet i henhold til analyseprogrammet.



Figur 2: Use Case diagram for alle moduler.

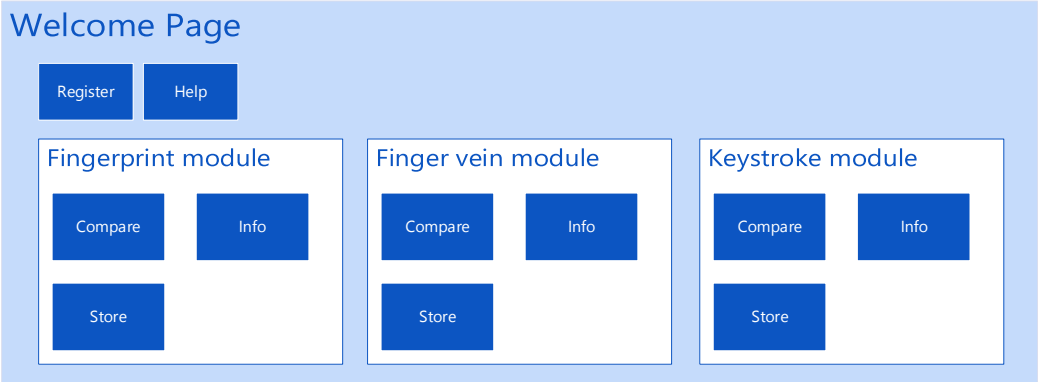
3 Design og arkitektur

3.1 Oppdeling av BioDemo

Gruppen valgte å dele opp BioDemo modulbasert, både i design-fasen og i utviklingsfasen, det betyr at fingeravtrykk, fingerårer og tastaturgjenkjenning ble utviklet hver for seg, i hver sin modul. Årsaken til at vi valgte å gjøre det slik, var for å kunne ha delleveranser underveis som modulene ble ferdige. Dette var for å sikre at vi leverte noe ved prosjektets slutt og ikke risikertee en situasjon der vi hadde tre halvferdige moduler og ingenting som var klare til bruk og kunne demostreres. Selve modulene i dette prosjektet ble derfor utviklet i inkrementer (dog med noe overlapp), men innad i hver modul foregikk utviklingen i henhold til Scrum (altså med sprinter, delleveringer og mulighet for å gjøre endringer underveis).

For hver modul var det forskjellige ressurspersoner som vi skulle forholde oss til, noe som også medførte at moduler var en fordel. I tillegg var en av de største utfordringene for prosjektet de forskjellige eksterne programmene og utstyret som BioDemo skulle implementere og kommunisere med. For å møte disse utfordringene vurderte vi flere muligheter; en var å dele opp hele utviklingsfasen i faser: en fase for innsamling av informasjon(om biometriområdene, hvordan oppdragsgiver ønsket at programmet skulle se ut og gjennomføres, hvordan de eksterne programmene fungerte), en fase for planlegging (design, GUI, gjennomføing), en fase for koding, og en fase for testing og kvalitetssikring. Disse fasene går på tvers av modulene, og ville kreve at vi fra begynnelsen så hvordan resultatet for BioDemo ville bli. Denne oppdelingen passer også godt for fossefallmodellen og andre plandrevne utviklingsmodeller, men for vårt prosjekt var det behov å kunne tilpasse underveis i utviklingen. Derfor falt denne løsningen bort, og vi valgte å dele opp i seperate moduler.

Fig. 3 viser en enkel organisering av moduelene. Alle modulene har sine egene undersider med ”Compare”, ”Store” og ”Info”. Innad i disse undersider finner vi de fleste funksjonene som er knyttet til en modul. Vi vil gå i mer detalj om disse funksjonalitetene i kapittel 4. Under ”Welcome Page” finner vi undersiden ”Register” og ”Help”. ”Register”-funksjonaliteten er felles for alle moduler og her skal bruker registere seg. Dette må gjøres før brukeren ønsker å samle inn biometriske data om seg selv. I tillegg har vi en hjelpefunksjon hvor brukerene blir vist en hjelpeside med mer informasjon om hvordan de kan bruke programmet. Denne siden er tilgjengelig uansett hvor brukeren befinner seg i programmet.



Figur 3: Enkel oversikt over moduler

3.1.1 Begrensninger for fingeråremodulen

Ettersom prosjektet ble utviklet med en agil utviklingsmetode, var det ikke nødvendig å ha alt av eksternt utstyr og programvare tilgjengelig fra begynnelsen. Gruppen fikk tildelt utstyr og eksternt programvare av oppdragsgiver etter behov, for eksempel når en modul nærmet seg slutten og det var tid til å starte på neste. Når fingeravtrykkmodulen nærmet seg slutten forespurte gruppen etter utstyr knyttet til fingeråremodulen og fikk beskjed om at utstyret var ikke ledig da en annen student brukte det. Gruppen valgte da å ikke vente på dette utstyret, men begynne på tastaturgjenkjenningsmodulen istedenfor. Ved prosjektets slutt var utstyret som trengtes for å skanne fingeråre forsatt ikke ledig. Dette medførte at gruppen ikke har levert denne modulen som planlagt. Se møtereferat i vedlegg C.1 der gruppen ble orientert om dette.

3.2 Andre lignede programmer

Gruppen investerte en del ressurser i å finne andre programmer som lignet BioDemo. Dette var for å hente inspirasjon om funksjonalitet, design og kunnskaper om muligheter og utfordringer utviklerne stod ovenfor. Av de som ble undersøkt, var det to eksempler med liknende krav som BioDemo og vi fokuserte på hvordan disse kravene ble løst.

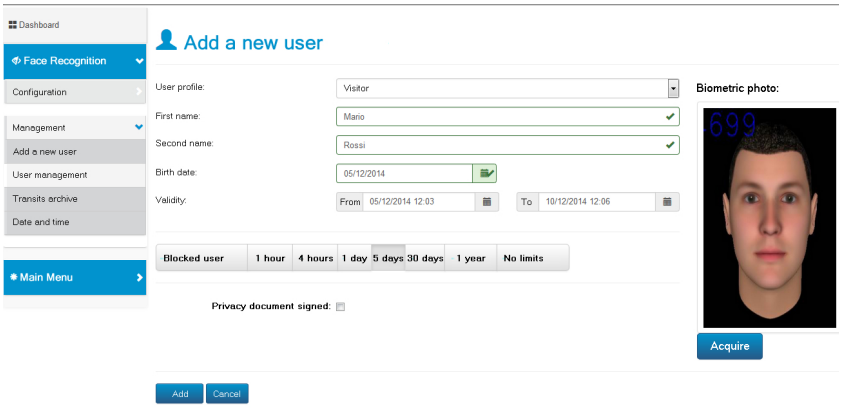
Prosjektet nSHIELD [1] har utviklet et biometrisk program med liknende personvernshensyn som i BioDemo. I dokumentasjonen til denne applikasjonen ser man eksempelvis dette ikke-funksjonelle kravet:

> “Protection, confidentiality and privacy during the acquisition, processing and storage of data and metadata concerning the people identity and related biometric profile.
>
> *- sitat fra nSHILD rapport kode D7.10 [1, p. 13]* ”

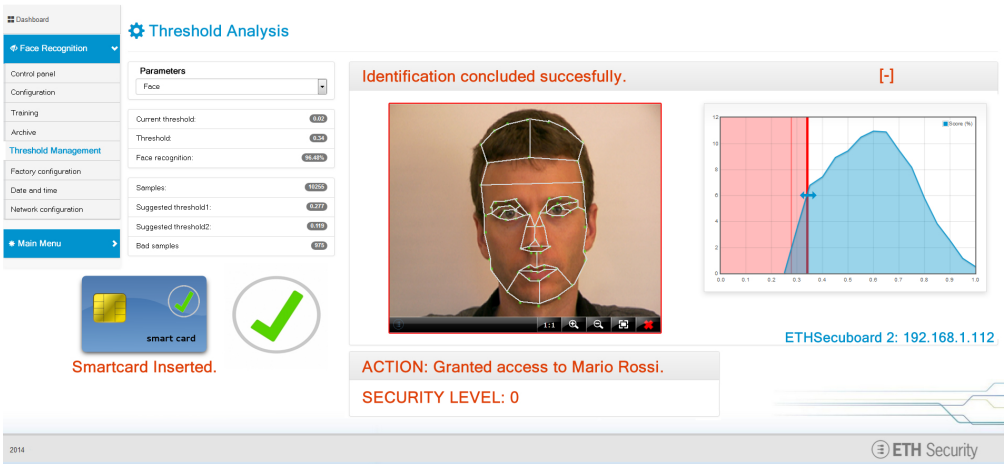
De tilfredsstiller dette kravet blant annet ved å kombinere permanent sletting av biometridata og kryptering.

I fig. 4 vises opprettelsen av en biometriprofil der informasjon om en gitt bruker (eksempelvis gjestebruker) blir lagret og i fig. 5 vises en grafisk ansiktsgjenkjenning.

[1] nSHILD forsker på sikkerhet, personvern og avhengighet i embedded systems (for mer informasjon se http://www.newshield.eu/what-is-nshield/)



Figur 4: Opprettelse av en besøkende gjest. [1, s. 32]



Figur 5: Fremvisning av ansiktsgjenkjenning [1, s.59]

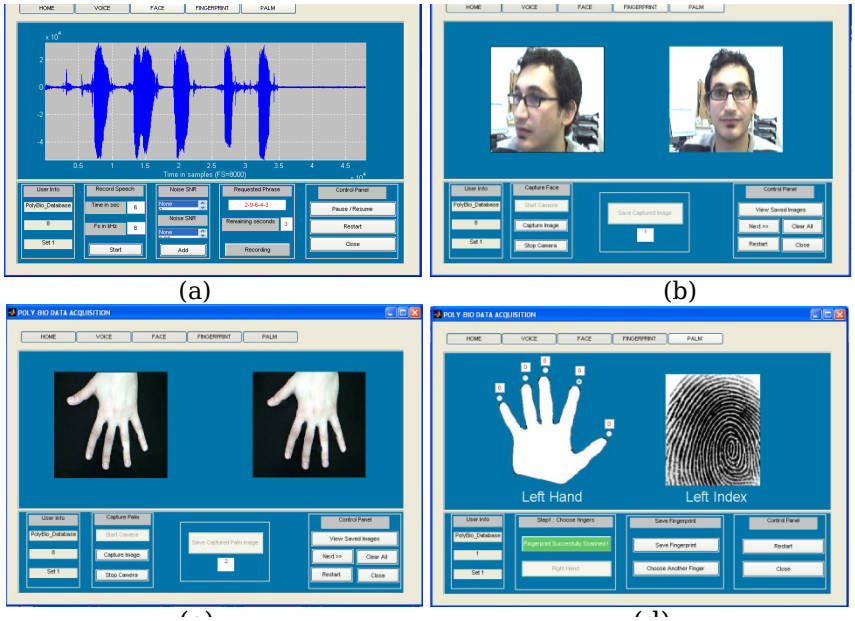
En av de andre kravene i kravspesifikasjonen, er å samle de ulike kodesnuttene til NBL til et helhetlig grensesnitt. POLYBIO er et eksempel på et program som benytter flere ulike typer biometrisensorere i et og samme program og fremstiller resultatet på en grafisk måte. Se fig. 6 for en oversikt over hvordan POLYBIO fremstiller sine biometriske data.

3.3 Biometri

En av hovedmålene med BioDemo var å demostrere hvordan biometriske data blir innsamlet, behandlet og analysert for personer som ikke kjenner til biometri og NBL. For at gruppen skulle være i stand til å utvikle et slikt program, måtte vi sette oss inn i grunnleggende teori om de områdene som vårt program dekker, og dette presenterer vi i underkapitlene som følger.

3.3.1 Fingeravtrykk

Fingeravtrykk er kanskje det mest kjente og brukte området innenfor biometri. Fingeravtrykk består av spesielle linjemønstre som har en såpass stor kompleksitet at de er unike hos hver person. De er mer eller mindre den samme hele livet og vanskelige å endre



Figur 6: Fire ulike eksempler på det grafiske grenesnittet i POLYBIO [2, s. 9]

på. Disse egenskapene har ført til at fingeravtrykk idag er den mest brukte biometriske metoden for å identifisere mennesker.
Sir Edward Richard Henry klassifiserte fire hovedmøstre av fingeravtrykk; ”arch”, ”loop’, ”whorl” og kompositt"i hans bok fra 1900 ” Classifiaction and use of finger prints”. [9] Til tross for at det over 100 år siden denne boken ble utgitt, blir denne klassifiseringen brukt den dag i dag, med modifiseringer. I tillegg blir minutiae også brukt til å identifisere fingeravtrykk. Minutiae er hovedegenskaper et fingeravtrykk har, og de typiske minutia er:
- *Rigde ending* - der en linje i mønsteret slutter
- *Bifurcations* - punktet der en enkel linje splittes til to linjer
- *Short ridge eller dot* - en linje som er betraktelig kortere en de andre. [10]

Figur 7: Forskjellig kvalitet på samme skannet finger. Bildet er tatt fra [3].

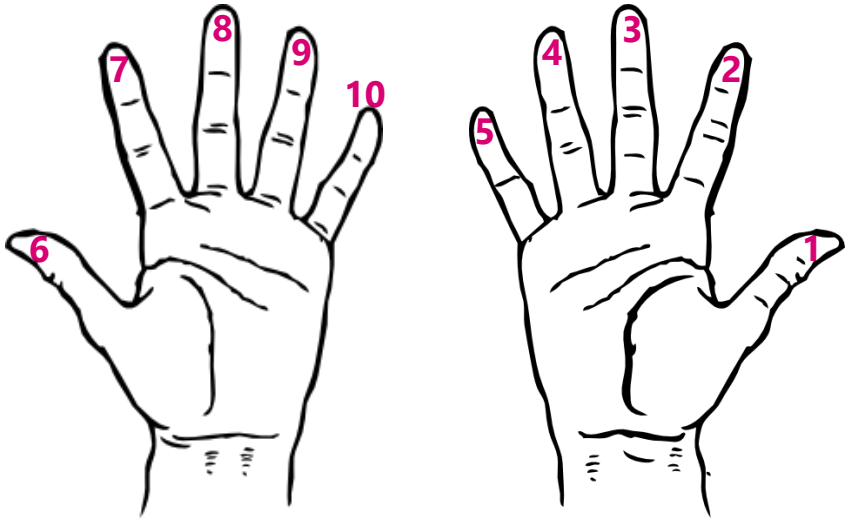
Skanning av fingeravtrykk

Ved skanning av fingeravtrykk er det flere faktorer som er med på å påvirke kvaliteten på bildet av det innskannede fingeravtrykket. Det er noen ganger svært krevende å få et godt skannet fingeravtrykk, og hvis dette ikke gjøres riktig, vil det påvirke kvaliteten på analysen og identifiseringen av fingeravtrykket. Teknikken og utstyret har kommet en lang vei fra da man måtte identifisere fingeravtrykk manuelt, men som fig. 7 viser, er det forsatt mulig å få store forskjeller på skannede fingeravtrykk av samme finger hos samme person. a) viser et bilde av god kvalitet, b) viser en veldig tørr finger, c) viser en våt finger og d) er en av dårlig kvalitet på grunn av vridning av finger. [3, s 74]

Fingeravtrykk i BioDemo

BioDemo vil etter innskanning av fingeravtrykk, vise selve avtrykket og avtrykkets minutiae-punkter, slik at brukeren kan se sitt mønster og de forskjellige egenskapene. Det legges opp til at NBL-brukeren som er med på demostrasjonen er i stand til å forklare hva minutiae er og hva slags mønster gjeste-brukerens finger har. Vi benytter oss av livescan, som betyr at vi benytter oss av digitale verktøy til å skanne fingeravtrykkene, kontra metoder som innebærer å sverte fingrene med blekk og rulle dem på papirlapper. [11].

NBLs måte å numrere fingrene følges i forbindelse med skanning av fingeravtrykk. Her telles fingrene fra en til ti, og starter på høyre tommel (Se fig. 8).



Figur 8: Numrering av fingre

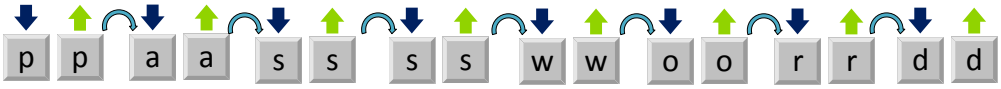
3.3.2 Tastaturgjenkjenning

Tastaturgjenkjenning er en automatisert metode for å gjenkjenne en person basert på hvordan vedkommende taster på tastaturet. Dette er en form for adferdsbiometri, eller ”noe man gjør” for å identifisere seg. En av de største fordelene med tastaturgjenkjenning er at maskinvare allerede er på plass (de fleste PCer har et tastatur), og man trenger derfor ikke noe ekstra utstyr for at dette skal fungere. Men dette er også en av utfordringene med denne typen autentisering fordi det finnes forskjellige tastatur for forskjellige språk. [12] Ut fra brukerens inntasting, genereres en egen ”template” for brukeren. Det

er flere tasteegenskaper som kan brukes for å gjenkjenne en bruker;
- Duration - Tastevarighet, tiden brukeren holder en knapp nede.
- Latency - Ventetid, tiden mellom to knapper
- Trykket når en knapp ble trykt ned.
- Brukerens plassering av finger på knappen.
- Hvilken finger som benyttes. [4]

For BioDemo skal det kun benyttes duration og latency mellom knappene.

For å demostrere hvordan denne utregningen foregår, har gruppen lagt ved noen figurer basert på ordet ”password”. Første figur (fig.9) viser de forskjellige verdiene som man regner duration og latency fra. Pilene ned og opp demostrerer når en bruker trykker en knapp ned og når vedkommende slipper knappen. Pilen som ligger mellom opp og neste knapp ned er latency.



Figur 9: Viser ned og opp knapp og latency mellom knappene

Videre blir noen enkle matematiske formler brukt for å regne ut duration og latency, For tastevarighet:

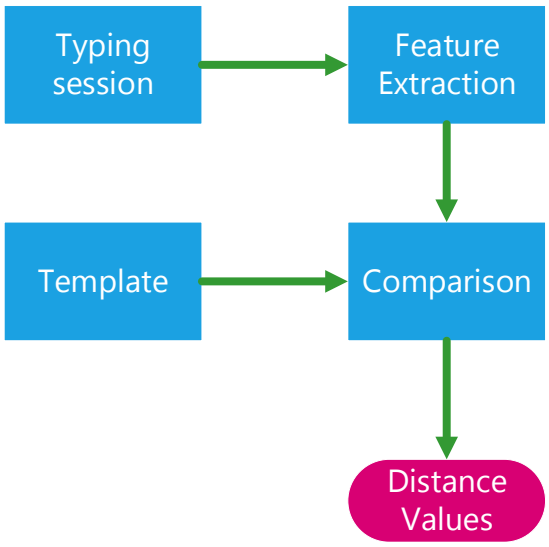
$p^{opp} - p^{ned} = p(duration)$
$a^{opp} - a^{ned} = a(duration)$
$s^{opp} - s^{ned} = s(duration)$
$s^{opp} - s^{ned} = s(duration)$
$w^{opp} - w^{ned} = w(duration)$
$o^{opp} - o^{ned} = o(duration)$
$r^{opp} - r^{ned} = r(duration)$
$d^{opp} - d^{ned} = d(duration)$

For latency:

$a^{ned} - p^{opp} = pa(latency)$
$s^{ned} - a^{opp} = as(latency)$
$s^{ned} - s^{opp} = ss(latency)$
$w^{ned} - s^{opp} = sw(latency)$
$o^{ned} - w^{opp} = wo(latency)$
$r^{ned} - o^{opp} = or(latency)$
$d^{ned} - r^{opp} = rd(latency)$

Merk at for latency er det mulig å få negative tallverdier. Dette skjer i forbindelse med at en bruker slipper første knapp etter at andre knapp er trykket ned. Dette er ganske vanlig ved for eksemplel ”Password” med stor bokstav foran. Her benyttes ofte shift-knappen for å lage en stor bokstav, og shift slippes etter at p trykkes ned. For duration er dette ikke mulig.

For å lage en template kreves det av brukeren taster samme ord flere ganger. For hvert ord som tastes inn, blir det produsert duration og latency, og utifra disse blir gjennomsnitt (mu - μ) og standardavvik (sigma - σ) generert. Det er disse verdiene som ligger i template og benyttes for å sjekke om det nylig inntastede ordet kommer fra den samme brukeren i databasen. Jo større distanse det er fra templaten, jo mindre sannsynlighet er det for at det er snakk om samme person. Fig. 10 viser denne prosessen.



Figur 10: Hvordan prosesseringen skjer fra inntasting til resultat. Figuren er tatt fra [4]

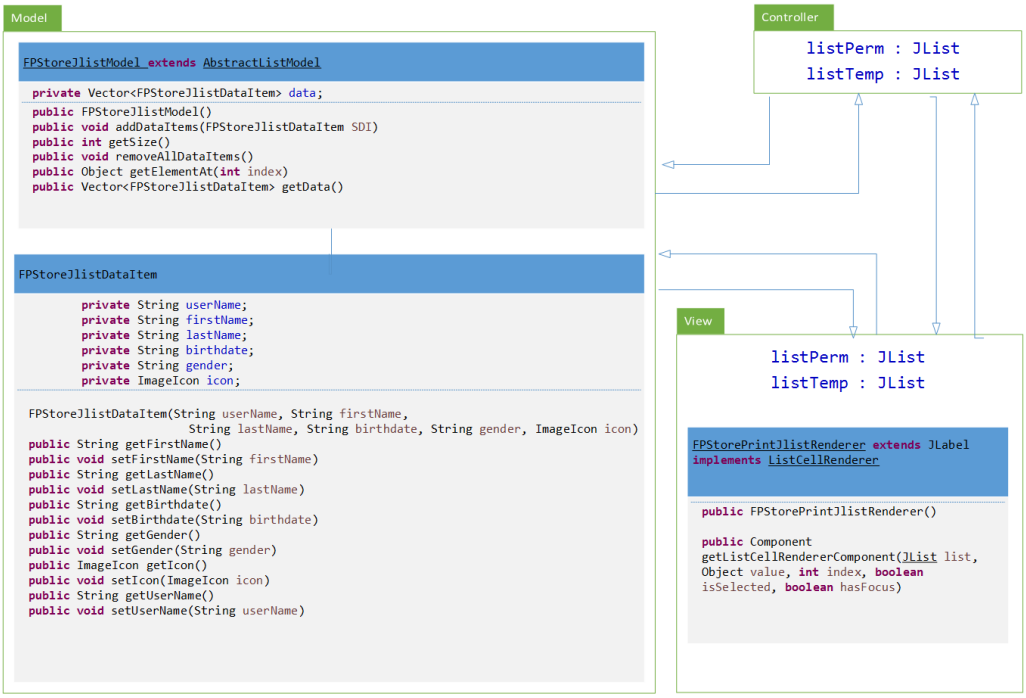
3.4 Arkitektur

3.4.1 Model view controller

Java Swing-komponenter fungerer bra med et Model View Controller (MVC)-mønster. Dette mønsteret deler logikken og operasjonene mellom delene model, view og controller. Controller er ansvarlig for å holde rede på brukerinteraksjon (f.eks. hvilket element i lista som er valgt.) View har ansvar for å gjengi modelldataene og dataene for visning. Modellen har ansvaret for å holde på dataene og oppdatere ved endret tilstand. [13, side 153-157]

For å vise hvordan vi har implementert MVC i BioDemo, viser vi i fig. 11 et konkret eksempel, som gjengir hvordan brukerlisten under storebenytter MVC. I modellen kan man legge til ”FPStoreJlistDataItem”, og den vil automatisk oppdatere view. Man kan fjerne alle elementer med ”removeAllDataItems”, noe som er nyttig når brukeren trykker på ”clear”-knappen i det grafiske grensesnittet. I view har Jlistene ansvaret for å vise dataene og render-klassen styrer logikken for hvordan disse datene skal fremvises. Renderen bruker modellen sin ”getElementAt(...)” for å hente det ”FPStoreJlistDataItem-objektet” som skal gjengis. I controlleren har Jlistene ansvaret for å holde oversikten over hvil-

ken indeks som er valgt i listen av brukeren og når vedkommende endrer endrer valgt element.



Figur 11: MVC implementasjon

3.5 Database

Gruppen vurderte det som mest naturlig å benytte en database for lagring av brukerdata. Vi så på flere databaseløsninger, men valgte å benytte en Derby-database på bakgrunn av at det er en open source-løsning som er godt tilpasset Java.

Som det kommer frem i kravspesifikasjonen under operasjonelle krav i kapittel 2, skal databasen være i stand til å registrere data midlertidig, slik at besøkende som ønsker det, kan matche seg selv med databasen eller med andre besøkende. Av personvernshensyn, skal disse midlertidige dataene slettes så fort BioDemo avsluttes. Det er mange mulige løsninger på den slik utfordring og de løsningene vi diskuterte var blant annet:

Løsning A: En felles database

Ett alternativ var en stor database som inneholdt alle data, både for brukerne som skulle lagres midlertdig og de som skulle lagres permanent. For å kunne slette kun de midlertidige dataene, ville det bli behov for en sjekk som gikk gjennom alle unike brukeres data og sjekket om denne brukeren var midlertdig registrert. Hvis dette var tilfellet, måtte alle tilhørende data om denne brukeren hentes ut fra forskjellige tabeller, og til slutt slettes. På en liten database (mindre en 50 registerte brukere) ville ikke dette ta altfor lang tid. Ved en større database med over 50 brukere, og en stor mengde registerte data som for eksempel ti fingeravtrykk, fire fingerårer og fem registerte ord til bruk for tastegjenkjenning, ville det bli mye data som måtte tas hensyn til. En slik slette-funksjon måtte bli kjørt ved avslutning av BioDemo og ved oppstart og ville ha medført lengre responstid i forbindelse med dette.

I tillegg er det flere potensielle feilsituasjoner som kan oppstå i forbindelse med en slik database. De feilsituasjoene som vi diskutere var blant annet; hva skjer hvis vi sletter feil person eller feil tilhørende data, hva om det ikke blir slettet, og på grunn av en mer intrikat design på databasen med avhenigheter og begrensinger (constraints), ville det bli mer utfordrende å kunne kode riktige SQL-spørringer som utførte disse operasjonene riktig. Dette ville medføre større risiko for at databasen ikke slettet sensitve biometriske data til brukerne. Og med tanke på at dette programmet ble laget for å gi et godt inntrykk av hva NBL og NISlab arbeider med, anså vi det som svært viktig at informasjonssikkerheten ble ivaretatt på best mulig måte.

Løsning B: To seperate databaser

En annen løsning var å skille databasen i to seperate databaser; en for midlertidig lagring og en for permanent laring. Disse to databasene vil ikke ha noe med hverandre å gjøre utover at de i BioDemo vil bli samlet til for eksempel en liste med alle brukere som har lagt inn data. Det som vil være spesielt med den midlertidige er som nevnt tidligere, er at den skal slettes ved avslutning av BioDemo. De to databasene vil ha identisk struktur i de modulene der det er aktuelt å registere data midlertidig (for eksempel fingeravtrykkmodul), i de modulene der det ikke er aktuelt vil det kun være en permanent database (for eksempel tastaturgjenkjenningsmodul). Dette gjør koden litt enklere ved at det kan være felles funksjoner som tar imot URL til den aktuelle databasen, istedenfor unike funksjoner for hver database. Ved sletting av databasen ville dette utføres ved å slette tabeller fremfor å søke frem til de radene som skulle slettes i den felles databasen. Mange av feilsituasjonene som vi nevnte i forrige avsnitt vil kunne dukke opp i en slik løsning også, men i og med hele databasen håndteres ved for eksempel sletting, vil det mest sannsynlig ikke dukke opp situasjoner der f.eks. data fra en permanent bruker slettes ved feil.

Løsning C: Database for hver modul

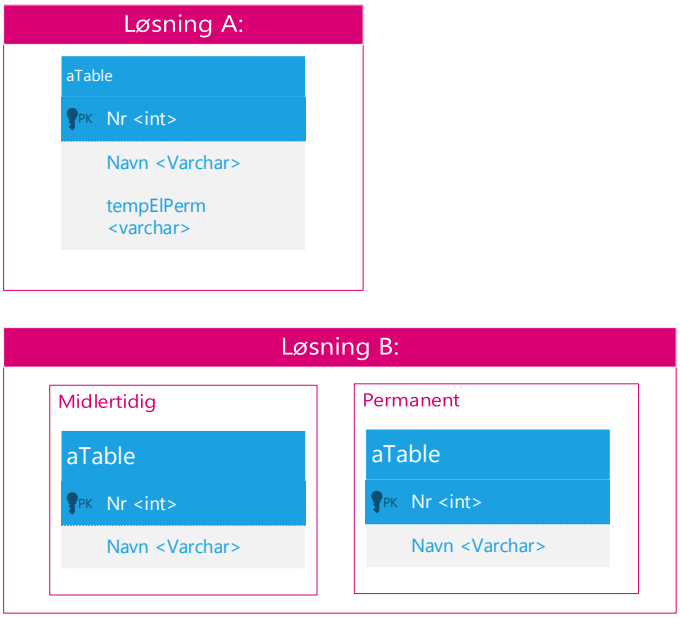
En tredje løsning vi vurderte, var muligheten for å ha en database pr. modul; Dette ville medføre at hver modul hadde sin egen midlertidige og permanente database. En fordel med denne løsningen, ville være at man fikk alle data relatert til modulen, og slapp å skille ut data fra de øvrige modulene. Dette kunne gi raskere responstid i forbindelse med uthenting av data og sammenligning mot databasen, da spørringene ville blir midre omfattende. Men det ville også dukke opp utfordringer i forbindelse med denne løsningen. Det ville bli flere databaser å holde orden på (to databaser pr. modul for tre moduler vil gi seks databaser, seks moduler vil gi tolv databaser etc.) Dette vil innebære mer kompleks sletting og oppretting, med flere adskilte tabeller som måtte opprettes og slettes for hver gang programmet ble avsluttet og startet opp. En slik løsning ville også medføre potensielt store mengder av duplisert data, spesielt med tanke på at hver modul ville ha behov for en tabell med brukerinformasjon pr. database.

Testing og konklusjon

For vår database var det behov for å være mest mulig plassbesparende og effektiv på grunn utfordringer med lese/skrive-hastighet på NBL-PC'en (Se fig. 47a). Vi ønsket også å unngå å ha mye duplisert data i databasene. Videre så vi et behov for å gjøre databasen enkel å forstå og for andre å kunne legge til tabeller i forbindelse med videreutvikling med andre moduler. Og vi ønsket også å gjøre det lettest mulig for brukerene å registere data i databasene.

Testing av løsning A og B

Vi valgte å utføre noen veldig enkle tester på hvor stor forskjell det var mellom løsning A og B. Vi valgte å ikke ta med løsning C ettersom denne ville være lik løsning C, men duplisert på antall moduler. For å ikke gjøre dette til en altfor stor oppgave, valgte vi å lage små og enkle tabeller med tilfeldige verdier i seg. All kode ble skrevet i Java for å være mest mulig realistisk. Fig. 12 viser en UML-modell av databasene som ble opprettet og utført testing på.



Figur 12: UML modell av testdatabasene. Løsning A øverst, Løsning B nederst.

For Løsning A utførte vi en SQL-spørring som plukket ut de radene i tabellen som hadde ordet ”temp” i ”tempElPerm”-kollonnen. For å ta tiden benyttet vi en av Javas System-funksjoner. Kodeutdraget i kodeutdrag 1 viser koden som utførte slettingen og tidtakningen.

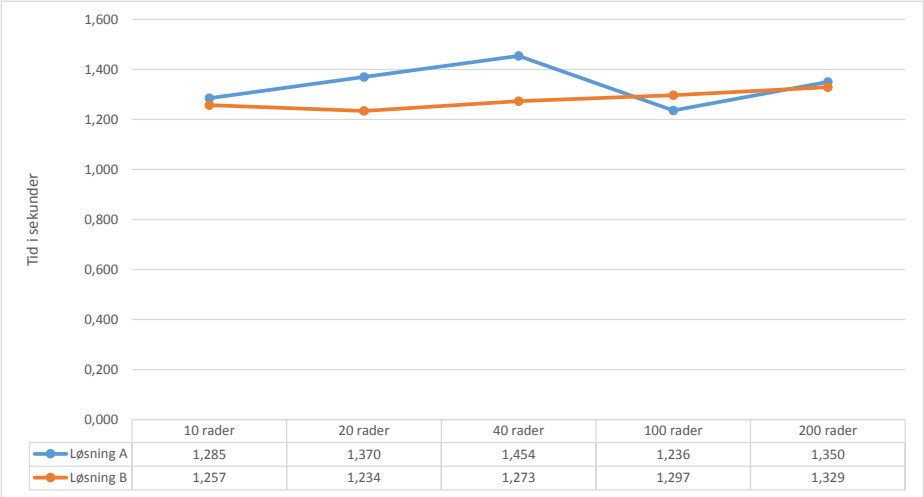
For løsning B utførte vi en spørring som slettet hele tabellen, ved hjelp av ”DROP”-spørringen i SQL. Her går ikke spørringen inn og leter etter hvilken rader den skal hente ut og slette, men heller slette hele tabellen. Databasen blir stående. Koden for denne er ganske lik i kodeutdrag 1, med unntak av selve spørringen.

Vi utførte testen på en av våre private PCer, i tre runder. Resultatet fra testingen vises i fig. 13. Den blå linjen representerer løsning A og den oransje løsning B. I denne grafen kan vi se at det var en ørliten forsjell mellom løsningene, men ikke mye. Det er snakk om hundredeler. Men i og med at denne testen ble utført på en PC med lignede ytelse som i figuren 47b, og ikke på NBL-PCen, kunne vi forvente tregere responstid på denne enheten.

Løsning C ville være den enkleste løsningen med tanke på å tilpasse for videreutvikling av BioDemo og utvidning med flere moduler. I en slik situasjon kunne man lage

```
1
2   //Starter tidstakningen
3   long start = System.currentTimeMillis();
4   //Query som skal utfores
5   String query =
6         "DELETE FROM aTable WHERE temporperm = ?";
7   /**
8   * Kode klippet ut
9   **/
10  //Setter verdien '?' til 'temp' slik at riktige
11    //rader blir slettet.
12  java.sql.PreparedStatement prepS
13    = con.prepareStatement(query);
14  prepS.setString(1, "temp");
15  prepS.execute();
16  System.out.println("Temp rader slettet. ");
17  /**
18  * Kode klippet ut
19  **/
20  //Etter at slettingen er avsluttet "stopper" vi tiden,
21    // og gjor om til sekunder
22  long elapsedTimeMillis = System.currentTimeMillis()
23                           -start;
24  float elapsedTimeSec = elapsedTimeMillis/1000F;
25  System.out.println("Tiden det tok a slette databasen
26    losning A: " + elapsedTimeSec );
```

Kodeutdrag 1: Kode som viser hvordan vi slettet og tok tiden etter løsning A



Figur 13: Graf over resultatene av testingen

"egne" databaser, uten å måtte røre gruppens databaser og trengte heller ikke å forstå dem. Men denne løsningen ville medført unødvendig mye duplisering av data. I en situasjon der en NBL-bruker ønsket å registere både sine fingeravtrykk og tastaturinntastingsverdier, måtte vedkommende først registere seg som bruker i den permanete databasen

i fingeravtrykkmodulen for deretter å gjøre akkurat det samme som en annen bruker i den permanente databasen i tastaturgjenkjenningsmodulen.

Dette så vi som unødvendig, og løsning B og A var bedre alternativer. Både løsning A og C ville medføre lengre responstid ved oppstart og avslutning på grunn av sletting av de midlertidige tabellene i databasen(e). Når det gjelder løsning B, så vi at denne løsningen gjorde det enklere med tanke på sletting av riktig data og bevaring av data som skulle tas vare på, samt unngå unødig duplisering av data og treg responstid i forbindelse med oppstart og avslutning. I tillegg ville denne løsningen gjøre tilpassingene til hver enkelt modul lettere med tanke på at ikke alle moduler vil få behov for en midlertidig database, og skillet mellom brukere som skulle lagres permanent og midlertidig ble klarere.

Vurdering av sikkerhetsmekanismer i databasen

Vi visste at databasen ville inneholde biometriske kjennetegn, som er sensitive opplysninger. Kryptering av databasen var et alternativ som tidlig ble vurdert for å hindre uautorisert tilgang til dataene. Etter samtale med oppdragsgiver (19.02.15), der vi tok opp mulige sikkerhetsmekanismer, ble det klart at NBL-PCen hadde en disk med treg lese- og skrivehastighet, og vi måtte ta hensyn til ytelsesproblematikk. Kryptering av disken eller databasen ville kunne medføre svært treg responstid. I tillegg ville maskinen være låst inne på laben, og det ville kreves passord ved pålogging, noe som også begrenset mulighetene for å få uautorisert tilgang. På bakgrunn av dette ble det avgjort at databasen ikke skulle krypteres.

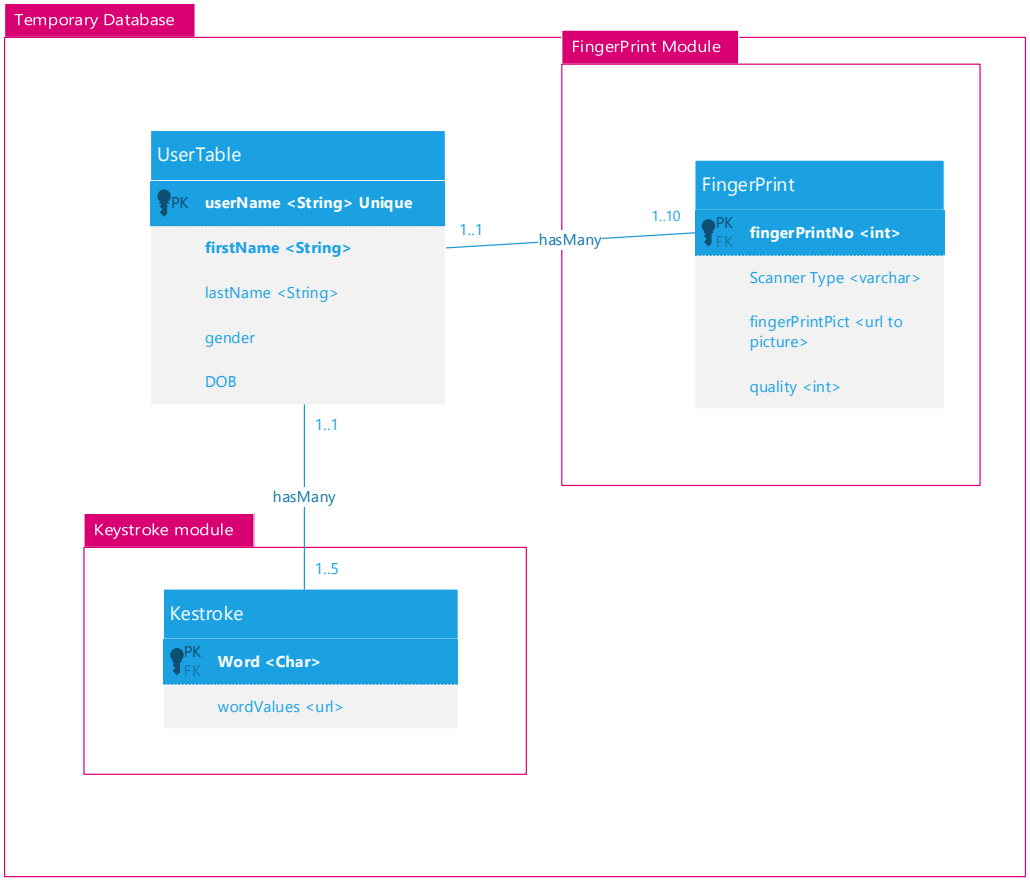
3.5.1 UML for databasene

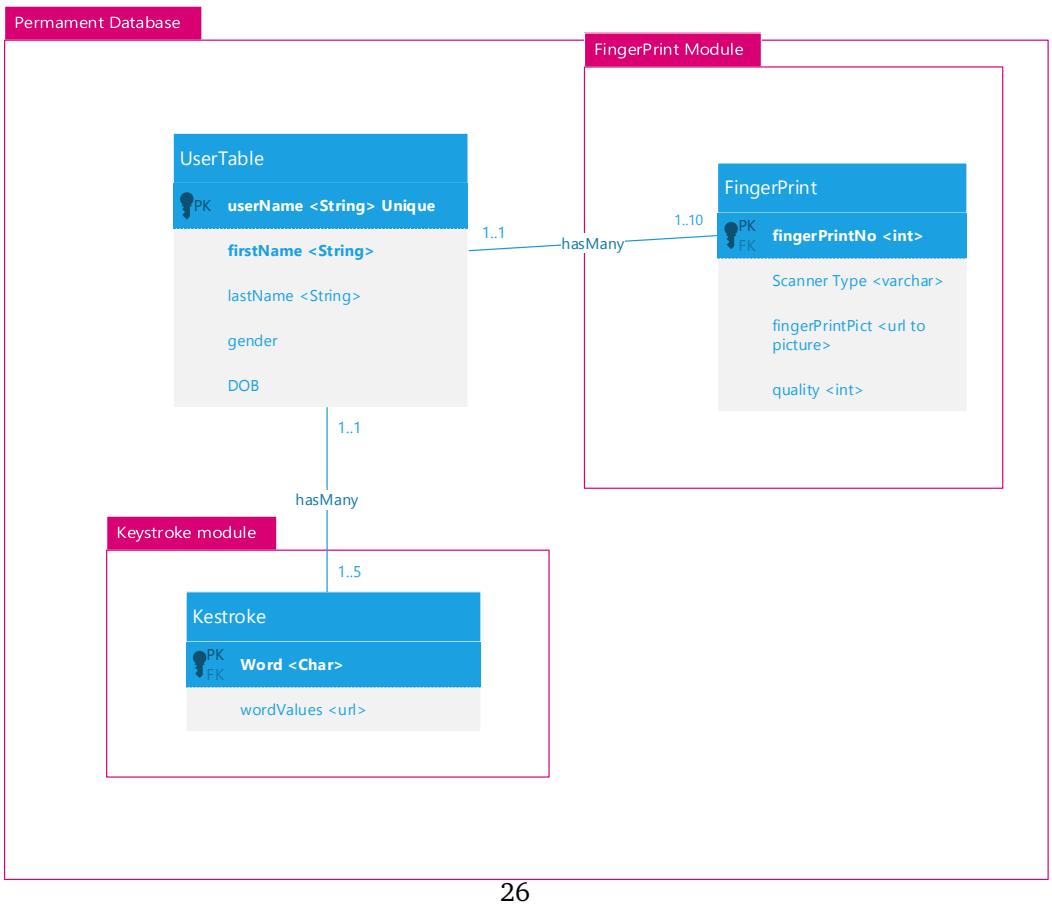
En konseptuell modell av databasen vises under i fig. 14. Her kan vi se de to identiske databasene, og de underliggende modulene i hver database: ”Fingerprint module” og ”Keystroke module”.

Ved design og utvikling av databaser vil man prøve å oppnå normalisering. Normalisering minimerer redundans, og gjør databasen mer robust. I tillegg blir den mer effektiv ved at overflødige dataer, dupliseringer og lignede blir redusert. En normalisert database vil si at ingen av de andre attributtene, som ikke er primærnøkler, er transitivt avhengige av primærnøkkelen. [14] Kort forklart betyr dette at hvis du vet primærnøklen, skal du også kunne vite de resterende dataene i de andre attributtene.

Dette stemmer ikke for ”Fingerprint”-tabellen i vår database; Når vi vet ”Username” og ”FingerprintNo”, vet vi ikke automatisk ”Scannertype”. Før vi implementerte databasen var de normaliserte ved at vi hadde flyttet ut skannertypene i en egen tabell.

Etter innspill fra oppdragsgiver, endret vi dette utkastet til det nåværende. Valget ble tatt bevisst, og gruppen vurderte fordeler og ulemper med å gjøre det slik. En av de største utfordringene var problematikken rundt selve skannerene og at vi ikke hadde fullstendig oversikt over hvor mange av disse som skulle benyttes i BioDemo. Oppdragsgiver ønsket selv å ha muligheten til å benytte flere forskjellige skannere uten å måtte gå inn i koden og endre/tilpasse til det. I tillegg ville skannerprogrammet ta seg av detektering av skannere tilkoblet PCen, og som kunne benyttes ved skanning av fingeravtrykk. Dermed falt argumentet om at vi trengte å ha en ferdig liste over alle skannertyper bort.





Figur 14: UML av den endelige konseptuelle designet på databasene

3.6 GUI

Et viktig krav fra oppdragsgiver var at BioDemo skulle bli et oversiktelig og lettfattelig program for NBLs gjester. Derav var det viktig å få på plass et program med lettfattelig flyt og brukerinteraksjon. Ingen av gruppemedlemmene hadde noen designkunnskaper fra før. For å være i stand til å utvikle et brukervennlig program, ble vi i samråd med veileder enige om å sette av en del tid de første ukene av prosjektet på å skaffe oss kunnskap om grunnleggende designprinsipper og brukerinteraksjon. Kunnskap har vi både skaffet oss gjennom faglitteratur og fagpersonell.

3.6.1 Utviklingsprosess

Som det kommer frem i programutvikling under avsnitt om programoppdeling i kapittel 3.1, valgte vi å utvikle programmet modulbasert. Utviklingen av designet har foregått på samme måte. Vi valgte å benytte samme design i alle modulene, og derfor la vår første modul (fingeravtrykksmodulen) mye av grunnlaget for designet. Med grunndesignet på plass, krevdes mindre jobb for utviklingen av øvrige moduler, da vi på forhåndt hadde utviklet og testet grunndesignet". Gjennom designutviklingen forholdt vi oss til flere designprinsipper. Mer om disse valgene i kapittel 3.6.2.

I løpet av de første møtene med oppdragsgiver ble det avklart at fingeravtrykksmodulen skulle være den første modulen. Funksjonalitet som BioDemo skulle ha, og hvordan dette kunne representeres på en god måte ble diskutert. I forbindelse med dette ble det blant annet ytret ønske om et program med ”fancy” utseende, gjerne noe ”CSI-aktig”, og med en hovedmeny, og menyer for å navigere mellom de forskjellige modulene. (Se referater fra gruppemøter i vedleggene C.1 og C.1).

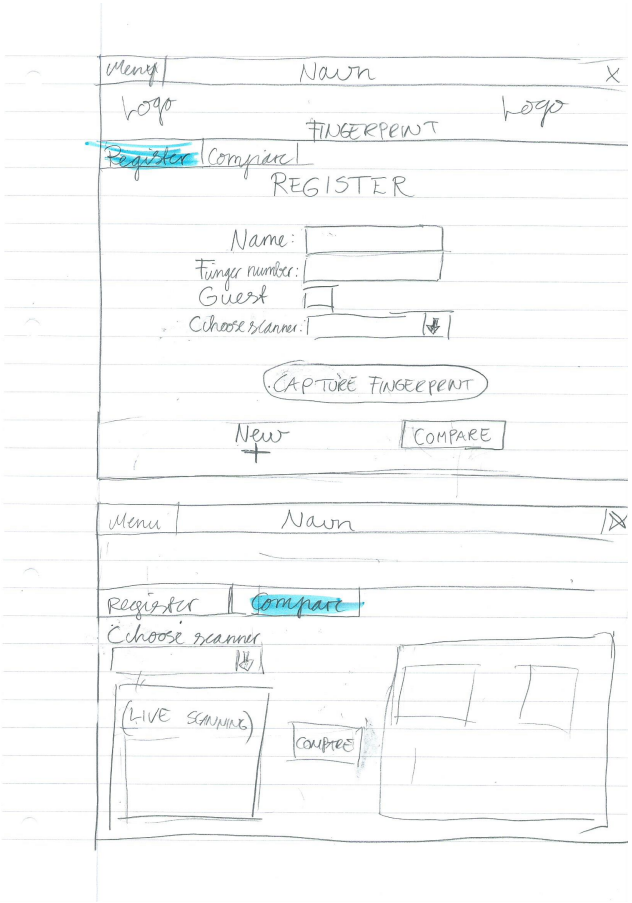
Det som kom ut av disse møtene ble satt opp som use cases av gruppemedlemmene, og gav oss informasjon om hva som trengte å være med i designet, men BioDemo skulle også være brukervennlig og se bra ut.

De første utkastene av fingeravtrykkmodul- og hovedskjerm-design ble skissert på ark. (Tidlige utkast kan ses i fig. 15). Etter å ha fått det grunnleggende designet på plass, ble det behov for en en wireframeløsning som lot oss sette opp mer detaljerte designskisser.

I denne forbindelse var Visio et alternativ som raskt dukket opp. Vi hadde alle erfaring med dette programmet fra før. Vi fant ingen inkluderte stencilspakker som passet til våre behov, men i Visio har man også muligheten til å laste ned stencilspakker, både fra Microsoft og tredjeparter, og valget falt på to stencilspakker fra tredjeparter, som til sammen dekket våre behov til knapper, vinduer, frames etc. Disse to var: Sketch GUI shapes by Niklas Wolkert og Nick Finck wireframe [2].

Etter at første designutkast i Visio ble satt opp, avgjorde gruppa at vi skulle ta kontakt med en ekspert innenfor designfeltet, slik at vi kunne få tips og tilbakemeldinger på skissene og input på designprosessen. I denne forbindelse ble vi anbefalt å ta kontakt med Ole Wattne som er studieprogramleder for BA Mediedesign. Møtereferat kan sees i vedlegg C.3. Vi fikk mye god input, blant annet på hva vi kunne gjøre for å få programmet vårt mer brukervennlig, både gjennom språkbruk (bruke ord som er lettfattelige for

[2] http://www.nickfinck.com/blog/entry/visio_stencils_for_information_architects/

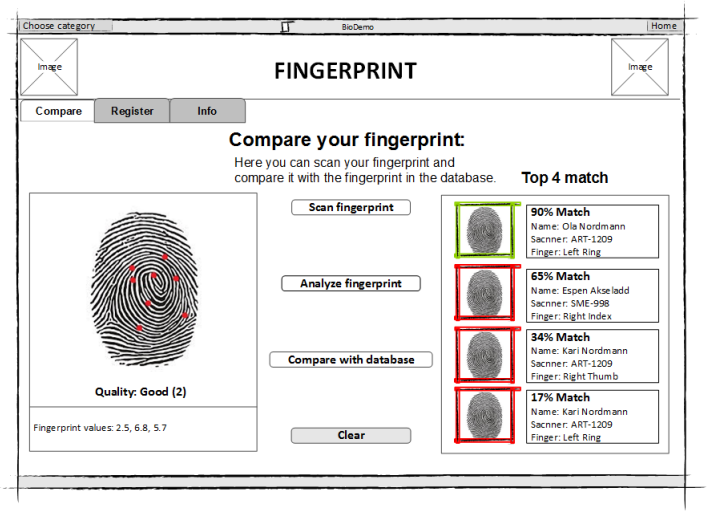


Figur 15: Et tidlig designutkast på ark

utenforstående)og på plassering av objekter. Han satte fokus på at programmet skal ha en god sekvens som er lett for brukeren å følge med på, og understrekte viktigheten av testing, og begynne med dette på et tidlig tidspunkt. Disse testene måtte gjerne tas på ark i starten, altså ved å utføre tester på designskissene våre. Vi ble også tipset om ikke å fokusere så mye på fargevalg på et så tidlig tidspunkt, men heller forholde oss nøytrale til dette og vente til mesteparten av designet var på plass.

Nye designversjoner ble satt opp på bakgrunn av dette møtet, og vi begynte med brukertester på skissene. Denne testingen kan det leses mer om i kapittel 5.

Basert på tilbakemeldingene på vårt sjetteutkast, ble endelige versjon av designskisser satt opp (se fig. 16 og fig. 17), og implementasjonen kunne starte. Denne prosessen kan det leses om i kapittel 4.



Figur 16: Nyeste versjon av compare-del

Tastaturgjenkjenningsmodul

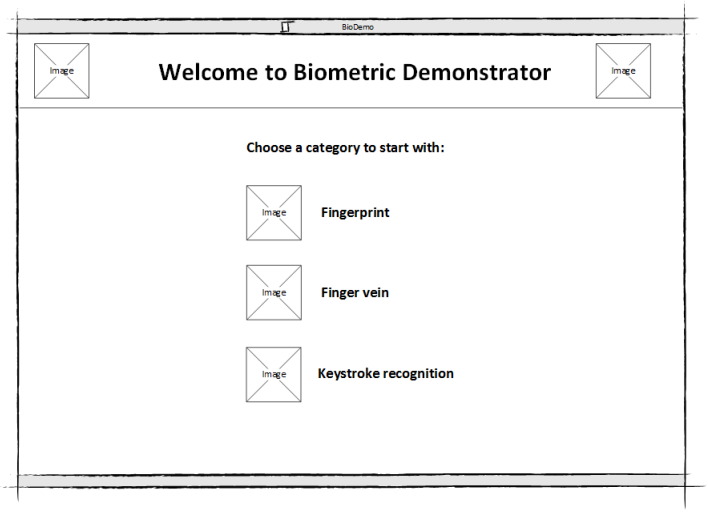
Vi arbeidet på samme måte med tasturgjenkjenningsmodulen. Begynte med skisser på ark, og deretter benyttet Visio. Selve grunnstrukturen på designet er det samme som i fingeravtrykkmodulen. Figurene under viser hvordan den endelige versjonen av tastaturgjenkjenningsmodulen ble seende ut (se fig.18 - Som viser hvordan vinduet der brukeren taster inn det utvalgte ordet og fig. 19 - som viser resultatet av sammenligningen mot databasen.) Merk at der det står ”match” ble det byttet ut med ”Distance” på grunn av hvordan denne utregningen foregår. Resten av skissene er lagt ved som vedlegg K.

3.6.2 Designprinsipper

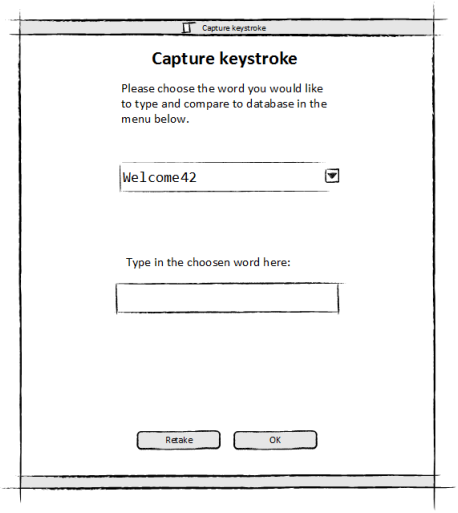
Vi valgte å benytte oss av flere designprinsipper i utviklingen av BioDemo. Nedenfor har vi tatt for oss de viktigste og forklart hvordan vi har valgt å bruke disse i vårt design:

Prinsipper hentet fra ”The essential guide to user interface design” [15]:

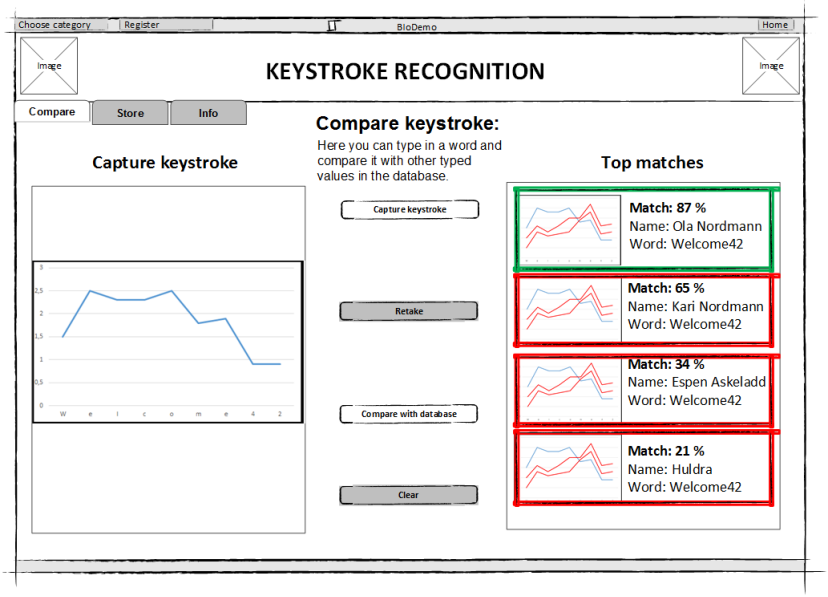
Forutsigbarhet



Figur 17: Nyeste versjon av hovedmeny



Figur 18: Endelige versjon av tastaturgjenkjenning - Vinduet der brukeren kan legge inn sine tastetrykk.



Figur 19: Endelige versjon av tastaturgjenkjenning - Viser match etter sammenligning med database.

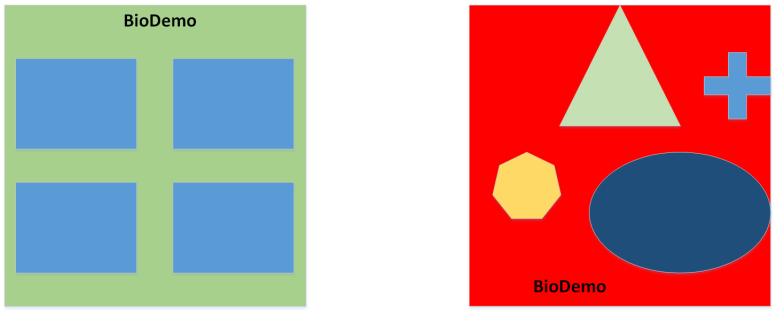
Oppnås ved å være konsekvent og følge et bestemt mønster. Funksjonalitet grupperes og plasseres på bestemte steder på skjermen. I BioDemo har vi vært konsekvente med hvilke ikoner som benyttes, og hvor knappene og menyene plasseres på skjermen.

Sekvensialitet
Oppnås ved at elementer arrangeres slik at blikket blir ført gjennom skjermen på en logisk, rytmisk og effektiv måte. Blikket dras mot: Lyse elementer før mindre lyse, isolerte elementer før grupperte elementer, grafikk før tekst, farger før svart og hvitt, mettede farger først, mørke elementer før lyse elementer, store elementer før små, osv. Boken viser også til studier som sier at det første stedet øyet beveger seg, er øverst i venstre hjørne på skjermen, for deretter beveger seg raskt med klokken over skjermen. Noe som vil si at et naturlig punkt å starte programflyten, sett fra brukerens perspektiv, er i venstre hjørne, for deretter å bevege seg nedover mot høyre.

Dette har også blitt benyttet i BioDemo, der de viktigste elementene plasseres øverst i høyre hjørne (”Categories” - menyen der brukeren velger hvor vedkommende vil begynne, samt fanene med ”Compare”, ”Store” og ”Info”). I tillegg foregår selve programflyten i hver modul fra øverst mot venstre, og beveger seg nedover mot høyre hjørne. F.eks. når et fingeravtrykk skal sammenlignes i fingeravtrykkmodulen, vil man bli presentert for et livescan-område til venstre på skjermen, midt på skjermen får man alternativene ”scan”, ”analyze” og ”compare” i synkende rekkefølge, for deretter å få ut resultatet på høyre side av skjermen. Enkelte elementer understrekes også med farger: hovedvinduet ble fremhevett ved å bruke en lys farge, samt tydelig aktivering (farge) og deaktivering (fargen forsvinner) av knapper når det er aktuelt. Dette er også prinsipper som ble fulgt i øvrige moduler

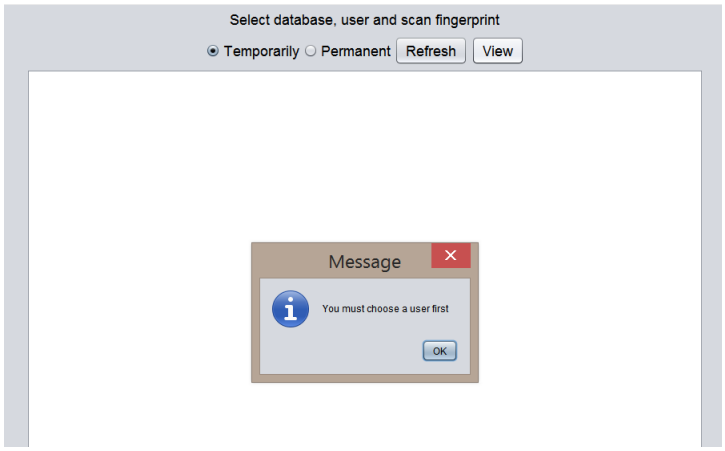
Enhet
Oppnås ved å benytte lignende former, størrelser og farger for info som er relatert til hverandre. I BioDemo følges dette prinsippet ved konsekvent bruk av ikoner, der samme oppgave skal ha samme ikoner. Menyene er også satt opp konsekvent: Med en meny som alltid befinner seg øverst i høyre hjørne. Uavhengig av hvilken modul som velges, vil brukeren få opp fanene: ”compare”, ”store” og ”info”. Disse fanene sammenligner henholdsvis biometriske kjennetegn, lagrer biometriske kjennetegn på en bestemt bruker, og får opp informasjon om valgt modul. Vi ha også, som nevnt, fokusert på selve innlesningen av biometriske kjennetegn til venstre på skjermen, valgknapper i midten, og resultatet til høyre. På denne måten sikres det at brukeren lett kjenner seg igjen og når vedkommene har lært seg programflyten i en modul, vil resterende moduler kunne læres så og si av seg selv.

Nedenfor (fig. 20) kan man se en illustrasjon av enhet og sekvensialitet (til venstre) og det motsatte (til høyre).

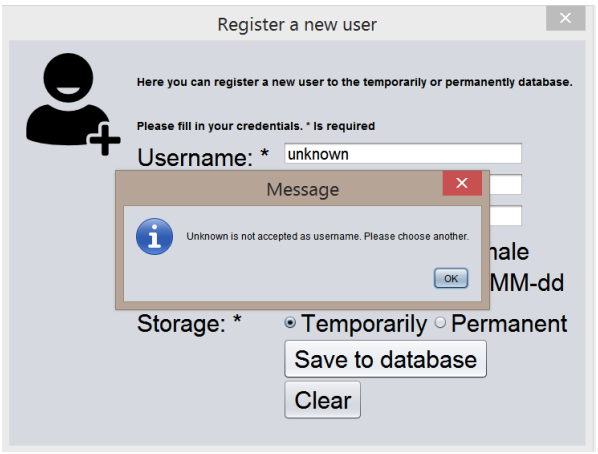


Figur 20: Eksempel der enhet og sekvensialitetsprinsippene følges og ikke følges

«Tilgivelse»
Alle brukere kan gjøre feil, og disse feilene må designet kunne tilgi. Når brukerfeil forekommer, må brukeren få opp konstruktive meldinger, samtidig som man må unngå for mange eller avanserte feilmeldinger, som vil gjøre det vanskelig for brukeren å forstå hva som har skjedd.



Figur 21: Viser hva som skjer når man ikke har valgt bruker fra databasen.



Figur 22: Viser hva som skjer - når ugyldig brukernavn (unknown) ved registrering.

Gjennom hele BioDemo har vi fulgt prinsippet med en clear-knapp, som aktiveres når brukeren har skannet, og som lar brukeren fjerne data fra skjermen og begynne prosedyren på ny. Vi har også vært nøye med å legge inn enkle, klare meldinger, både når operasjoner lykkes og misslykkes. Dette kan ses eksempel på i fig.22 og fig. 22.

Fargevalg:
Et viktig designprinsipp er også fargevalg. Det er viktig med god kontrast mellom forgrunn og bakgrunn for å sikre god leselighet. Samtidig kan farger i forskjellige steder i spektere understreke ulikhet. Vi valgte i denne forbindelse ut en palett med rgb-farger som fungerer godt sammen og gir god kontrast. Lyse farges brukes for å dra fokus mot elementer, mørke brukes til det motsatte. Samtidig kan farger brukes til å gruppere elementer. Fargene som benyttes i BioDemos GUI og deres bruksområder er som følger:

● rgb 33,54,104 Mørk blå grunnfarge - Bakgrunnsfarge for faner etc. som det ikke foregår interaksjon med.

○ rgb 240, 239, 239 Lys grå grunnfarge - Bakgrunnsfarge i interaksjonsområder.

● rgb(95, 109, 144) Mellomfarge for overnevnte- drar fokus mot steder som kan klikkes.

● rgb(102,61,153) Lillafarge - Markerer grenser. Gir kontrast til interaksjonsbakgrunn.

● rgb(0,0,0) Svart - Markerer grenser og benyttes som skriftfarge. Gir kontrast til interaksjonsbakgrunnen.

○ rgb(255,255,255) - Tekst og logo på mørk bakgrunn, og gir god kontrast til dette.

I tillegg benyttes fargene rød og grønn som understreking for å indikere match.

Prinsipper som er hentet fra GUI bloopers don‘ts and do‘s for software developers and web designers" [16]:

Bekrefte at det som utvikles samsvarer med brukerens måte å se oppgaven på
Her er det viktig å observere brukerne og deres handlinger. Det er viktig å få til et samarbeid. Det neste man må vurdere er hva som virker naturlig for brukeren. Det er viktig å unngå at brukeren må utføre oppgaver som virker unaturlige, da dette vil medføre vanskleligere læring, unødvendig bruk av tid og frustrasjon.

Et godt eksempel her er sjakkeksempelet fra boken som prinsippene er hentet fra (se over). For å flytte en sjakkbrikke, må man spesifisere hvilken brikke man vil flytte, og hvor den skal flyttes. Slik det er i det virkelige liv. Tenker man på GUI, er også dette det eneste brukeren skal trenge gjøre. Alt utenom dette blir en unaturlig handling. Man må derfor fokusere på å ikke gi brukeren flere valg enn det som er nødvendig for å utføre oppgaven.

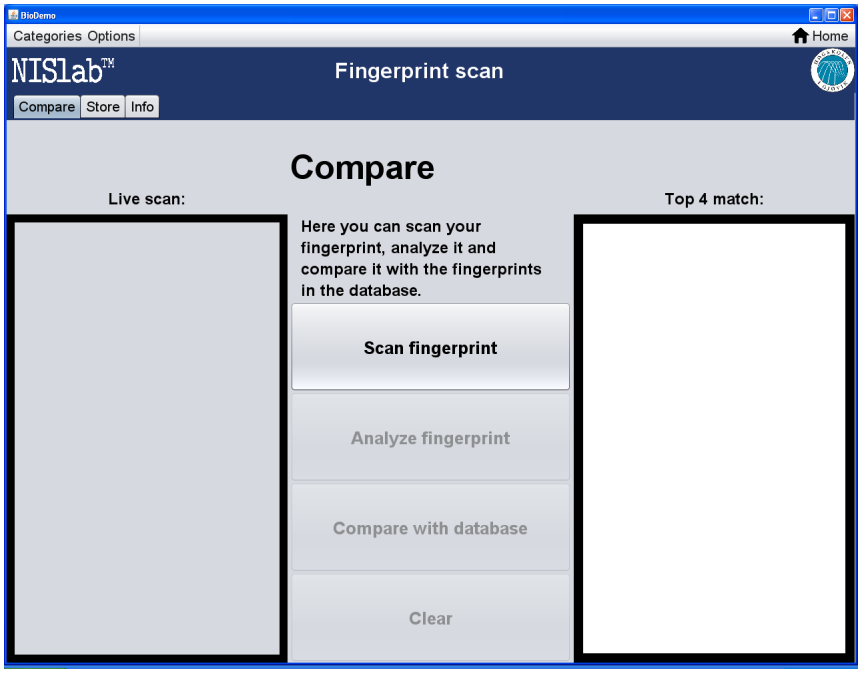
Man må også tenke på språkbruk, - det er viktig at språket enkelt skal kunne forstås av forskjellige brukergrupper.

Disse punktene har vi spesielt tatt hensyn til i forbindelse med brukertesting, i tillegg til demonstrasjoner for oppdragsgiver, eksperter og veileder. Vi gjorde på forhånd vurderinger rundt hva som virket naturlig for oss, og hva vi mente var enkleste måte å utføre

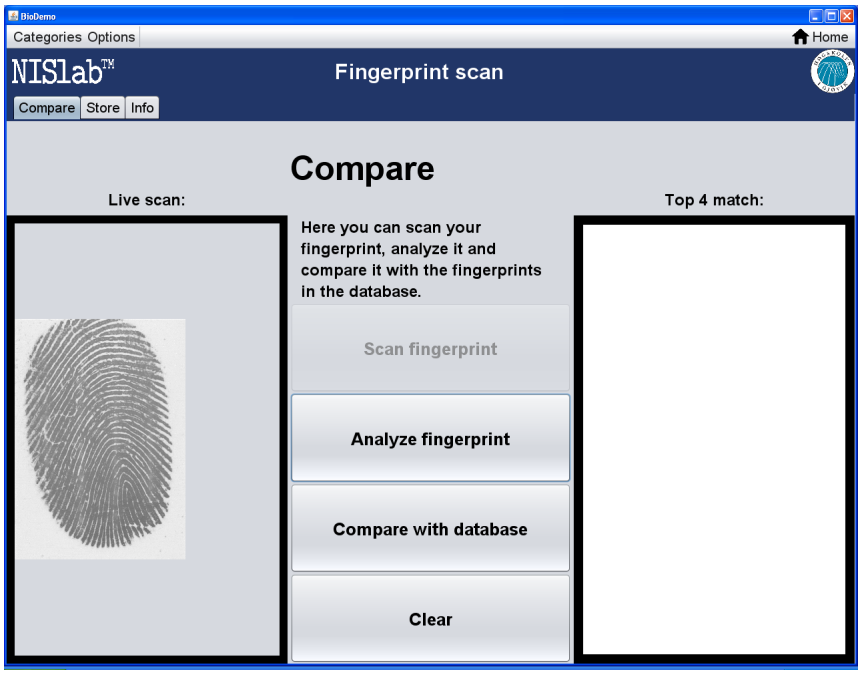
en oppgave på, og deretter tatt til oss de tilbakemeldingene vi har fått. Les mer om dette i kapittel 5.2

Dialogbokser:
Disse bør poppe opp når det er behov for dem, og forsvinne først når de ikke er nødvendige for brukeren lenger. På denne måten sikres det at brukeren rekker å se beskjedene. I tillegg bør overdrevne mengder dialogbokser unngås. En dialogboks kan aktivere en annen dialogboks, men normalt ikke videre enn det. I BioDemo har vi sørget for at brukeren må foreta seg noe for at dialogboksen skal lukke seg. Dette kan være ved å klikke ”OK”, eller som i dialogboksen som dukker opp når brukeren skal registrere seg, som blir værende til man er ferdig med å registrere brukere, og krysser den ut. Vi har også holdt oss unna dialogbokser som genererer flere dialogbokser, for å sikre at programmet holder seg ryddig og oversiktelig.

Knapper:
Brukeren bør tydelig vises når en knapp aktiveres og deaktiveres. I tillegg vil for mange knapper og knapper med forkortelser føre til forvirring og bør unngås. I BioDemo har vi fulgt disse prinsippene ved å kun ha de knappene som er nødvendige for å underbygge en naturlig programflyt. Knapper som ikke er aktuelle gråes ut (Et eksempel på dette, er ”Compare with database”-knappen, som gråes ut frem til brukeren har lagt inn biometrisk kjennetegn (se fig. 23). Når knappene så blir aktuelle å bruke, får de tilbake fargen (se fig. 24).



Figur 23: Viser hvordan knapper er deaktivert før fingeravtrykk har bitt skannet inn.



Figur 24: Viser at knappene aktiveres når fingeravtrykk har blitt skannet inn.

4 Programutvikling

Denne delen vil forklare hvordan BioDemo ble realisert og implementert. Vi vil gå litt mer i dybden på begge modulene og forklare hvordan vi har løst disse. I tillegg vil vi også her trekke frem hvordan implementasjonen og tilpassningen til de forskjellige analyse- og prosesseringsprogrammene vi inkluderte i vårt program. Merk at kildekoden til flere av disse ”eksterne” programmene ikke er inkludert i denne rapporten etter oppdragsgivers ønske.

4.1 BioDemo

I kapittel 3.1 omtaler vi hvordan vi delte opp programmet. I tillegg til at vi delte opp programutviklingen etter moduler, utviklet vi de komponentene som ikke falt under noen spesefikk modul samtidig som vi jobbet med modulene. I denne delen vil vi trekke frem noen løsninger som faller inn under den generelle delen av BioDemo, og som ikke tilhører en spesefikk modul.

4.1.1 Programmeringsmønstre

Utillity

Et av mønstrene som ble anvendt i BioDemo, var utillity-klasser. Dette er en klasse med relaterte funksjoner som blir kalt på en statisk måte. Dette er en vanlig måte å programmere på når tilstand ikke er en faktor og man ønsker å unngå redundans i koden. Ikke alle i programmeringsmiljøet er enige i at dette er et mønster, og kaller det heller et anti-mønster (siden de hevder det strider mot objektorientert tankegang) [17]. Gruppen valgte å benytte utilities-klasser for å lage hjelpefunksjoner som utfører oppgaver flere klasser kan ha behov for. Vi mener selv at dette både har gjort utviklingen mer effektiv og koden mer lesbar. Vi ser likevel at noen av utility-klassene burde blitt delt opp i mindre klasser for å lette vedlikeholdet av koden.

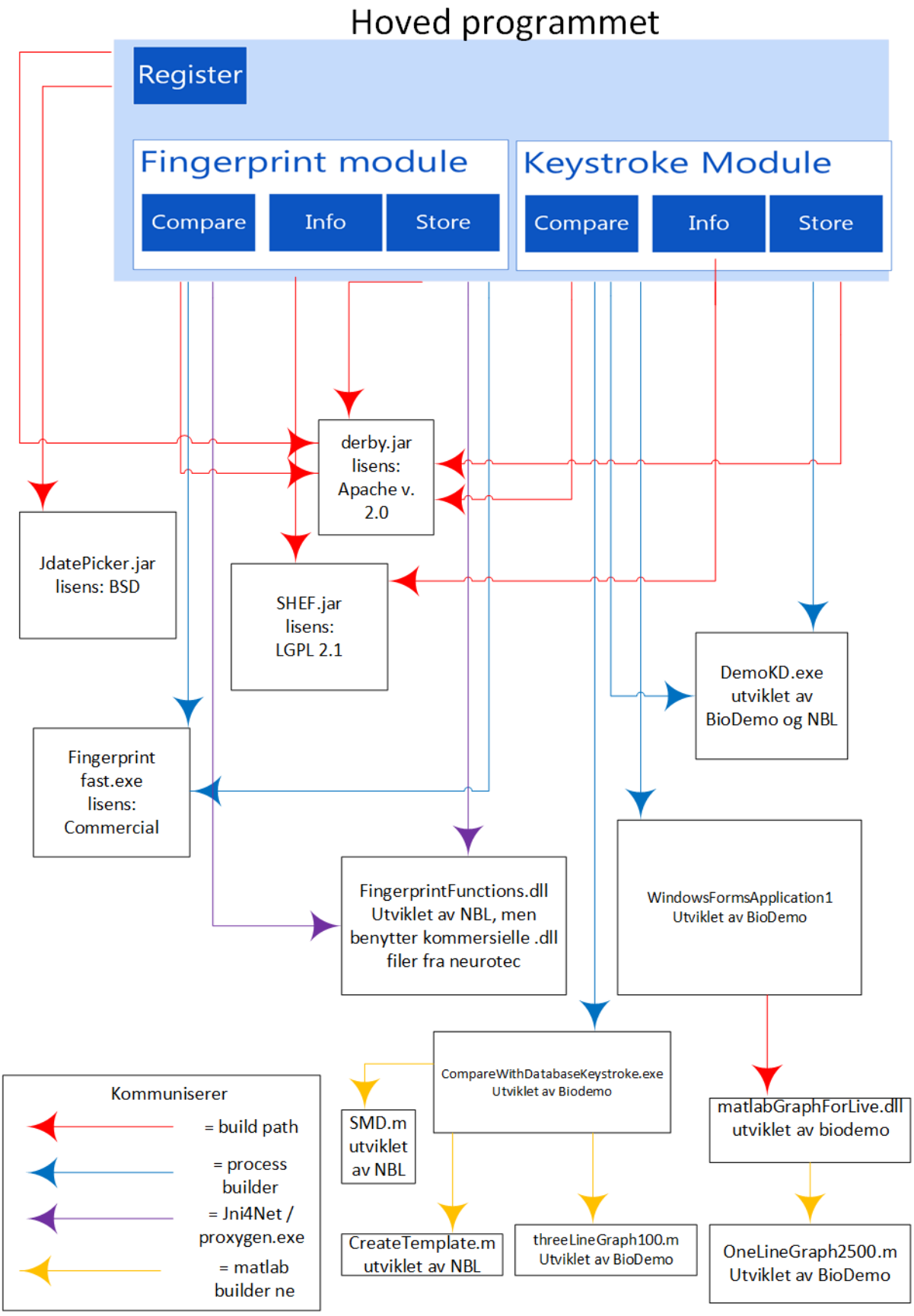
Utility-klassene som ble utviklet er som følger:

- **UtillityClass :** All behandling av filer (sletting, omdøping av filnavn, sortering av filer og oppdatering av metadata) Hjelpefunksjon for plassering av GUI- komponenter med GridBaglayout og XML-behandlingsfunksjoner.
- **Queries:** Inneholder alle databasespørringene.

4.1.2 Det store bildet

Figuren under (fig. 25) illustrerer hvilke delprogrammer (komponenter) som inngår i BioDemo, hvor de blir kalt på, hvordan de kommuniserer, og med hvilke programmer de kommuniserer med. Pilene indikerer hvilke deler av hovedprogrammet som benytter delprogrammer for å utføre deler av sine oppgaver. Som du kan se i nederste boks i figuren, har piler i ulike farger, forskjellige betydninger.

⟵ Denne komponenten er enten en JAR- fil som er satt i build path for Java-prosjektet, eller en .dll-fil som er satt som referanse i et C# -prosjekt.



Figur 25: Delprogrammene som inngår i demostratoren

⟵ En ny prosess startes ved hjelp av processBuilder i Java, av et kjørbart program (.exe). Vi konfigurerte hvor prosessen skal starte fra (hvilken directory) og sendte med eventuelle parametere til programmets main-funksjon.

⟵ Jni4Net sammen med Proxygen lager en bro mellom C# og Java. Dette gjør det mulig å benytte .dll-filer skrevet i C# i et Java -program.

⟵ MATLAB-builderne lager wrapper kode i form av en .dll. Denne kan brukes som en referanse i C#-prosjekt og gir mulighet for å kalle på MATLAB kode fra C#.

Tabellen (tabell 1) viser en oversikt over disse delprogrammene.

Tabell 1: Beskrivelse av underprogram

Programnavn	**kalt fra**	**Brukes til**	**Ekstra info**
JDatePicker.jar	Registrer	Sikrer lovlige verdier når Brukeren oppgir en fødselsdato.	Klassen DateLabelFormatter definerer datoformatet. Denne funksjonaliteten ble bygget rundt JDatePicker (Dens kildekode har ikke blitt endret på av gruppa)
Derby.jar	Registrer	Lagre data som brukernavn, fornavn, etternavn, fødselsdato og kjønn i en database.	
Derby.jar	FP.Store	Lagre fingeravtrykk i databasen og hente ut fingeravtrykk så man kan se hva som er lagret av fingeravtrykkdata på en bruker.	
Derby.jar	FP.Compare	Hente ut alle fingeravtrykk, som skal brukes i sammenligning av gjestebrukerens fingeravtrykk (live scan)	
Derby.jar	KS.Store	Lagre hvilke ord som en bruker har lagret tastaturgjenkjenningstemplate på og en referanse til XML-filen som holder disse dataene	
Derby.jar	KS.Compare	Henter ut informasjon om hvilke brukere som har registrert et gitt ord	Spørringene for å gjøre dette har blitt lagd, men den endelige løsningen baserer seg ikke på dem.

Fortsettelse på neste side

Tabell 1 – ... Fortsettelse

Program navn	kalt fra	Brukes til	Ekstra info
SHEF.jar	Infotab (alle)	Tilbyr et grafisk grensesnitt som genererer HTML for brukeren.	Kildekoden til SHEF.jar har ikke blitt endret på, men vi bygget funksjonalitet rundt for å kunne skrive til en (.html)-fil og lese fra den. På denne måten blir endringer gjort i en infotaben bevart, også neste gang programmet startes.
Fingerprint fast.exe	FP.Store	Tilbyr et grafisk grensesnitt for å velge hvilket fingeravtrykkleser som skal benyttes og hvilken finger det skal tas fingeravtrykk av. Produserer et bilde av hver fingeravtrykk som har blitt tatt	Laget kode for å endre filnavnet til bilde så det passer inn i programmets navnemal (benytter brukernavnet + databaseNavn + fingernummer).
Fingerprint fast.exe	FP.compare	Samme som ovenfor	Laget funksjonalitet rundt som henter ut det aller siste fingeravtrykket som ble tatt. Dette benyttes i livescan (filene er sortert på tiden filen ble opprettet)
Fingerprintfunctions.dll	FP.Store	Analyserer kvaliteten til et fingeravtrykkbilde og returnerer en score-verdi. Tar imot en byte-array av et bilde og returnerer en byte-array av det samme bildet, men med minutiae-punkter.	Laget kode som sjekker om score-verdien er lav, og om så er tilfelle, sletter bildefilen. Laget kode som leser bildet ut i fra byte-arrayen og viser fingeravtrykket med menutiae points.
Fingerprintfunctions.dll	FP.Compare	Samme som ovenfor Sammenligner to fingeravtrykk og gir en matchscore	Samme som ovenfor Laget kode som henter alle fingeravtrykkene fra databasen og sammenligner disse med live scan- fingeravtrykket. Avtrykkene blir lagt i en liste, og listen blir sortert på matchscore. De fire bildene med høyest matchscore blir presentert for brukeren.

Fortsettelse på neste side

Tabell 1 – ... Fortsettelse

Program navn	kalt fra	Brukes til	Ekstra info
DemoKD.exe	KS.Store	Programmet startes med processBuilder med brukernavn og upperValue (heltall) som parametere. Brukeren taster inn et ord valgt fra en dropdown-liste så mange ganger som ”UpperValue” tilsier. Hvis feil ord ble registret på et av forsøkene, må vedkommende taste dette forsøket på nytt. Det blir generert data om ”key down” og ”key up”. Disse blir lagret i en XML-fil og filnavnet benytter brukernavnet som ble sendt med som parameter.	XML-dom benyttes for å lese en XML-fil for en gitt bruker. Gjennomsnittlig duriation og latency regnes ut, og dette presenteres for brukeren.
DemoKD.exe	KS.Compare	Brukernavn er satt til _live og ”UpperValue” er satt til 1.	Leser som ovenfor, men dataene som blir hentet er data som skal benyttes for å lage grafer.
Windows Form Application1.exe	KS.Compare	Leser XML for _live.xml og sender disse dataene videre.	
MATLAB Graph For live.dll	Windows Form Application1.exe	Å kommunisere med MATLAB-kode. Tar inn x,y og label-verdier for å kunne lage graf.	
OneLiveGraph2500.m	matlab Graph For Live.dll	Lager grafen for live keystroke capture.	
threeLineGraph100.m	compare With Database Keystroke.exe	Produserer graf og png-bilde av grafen. Denne grafen har live capture og +- standardavvik som grafverdier	Nummeret på slutten av filnavnet identifiserer hvilken bruker på top fire grafen tilhører.
SMD.m	compare With Database Keystroke.exe	Kalkulerer distanse mellom bruker i database og livescan-bruker	
CreateTemplate.m	compare With Database Keystroke.exe	Kalkulerer standardavvik og gjennomsnitt for en bruker.	Brukes som grafdata og for å kalkulere distanse.

Fortsettelse på neste side

Tabell 1 – ... Fortsettelse

Program navn	kalt fra	Brukes til	Ekstra info
Compare With Database Keystroke.exe	KS.Compare	Leser XML for alle XML-filer som inneholder det aktuelle ordet. Kalkulere durations og latency for alle oppføringer i disse XML-filene. Disse dataene blir lagret i en array.	Sorterer en liste av XML-filer ut i fra den distanseverdien som ble returnert av SMD.m. Skriver så en XML- fil som inneholder filnavnene til XML-filene i lista, brukernavnet til brukeren XML-dataene ble generert om og distansen retunert fra SMD.m Bruker arrayen til å lage, finne standardavik og gjennomsnitt og kalkulere distanseverdi mellom denne brukeren og live-brukeren. XML-filen som opprettes blir lest og topp fire match blir valgt ut i fra den.

Lisens	Kan	Kan ikke	Må
MIT	• Kommersielt bruk • Modifisere • Distrubuere • Benyttes sammen med en restrektiv lisens • Privat bruk	• Saksøke forfatteren av koden.	• Inkludere copyright. • Inkludere lisens
BSD 2	• Kommersielt bruk • Modifisere • Distrubuere • Benyttes sammen med en restrektiv lisens • Privat bruk • Tilby garanti	• Saksøke forfatteren av koden.	• Inkludere copyright
Apache 2	• Kommersielt bruk • Modifisere • Distrubuere • Benyttes sammen med en restrektiv lisens • Privat bruk • Tilby garanti • Beskytte patenten sin når den blir utfordret.	• Saksøke forfatteren av koden. • Kan ikke bruke logoer, utvikler navn eller annet for å promotere vårt produkt.	• Inkludere copyright • Inkludere lisens • Notere vesentlige endringer i koden til den opprinnelige programvaren. • Hvis programvaren kommer med en notice fil så må denne også tas med når vi distrubuerer programvaren.
LGPL v2.1	• Kommersielt bruk • Modifisere • Distrubuere • Benyttes sammen med en restrektiv lisens	• Saksøke forfatteren av koden.	• Inkludere copyright • Inkludere lisens • Notere vesentlige endringer i koden til den opprinnelige programvaren. • Hvis programvaren kommer med en notice fil så må denne også tas med når vi distrubuerer programvaren. • Gi muligheten til å oppdatere biblioteket med en nyere versjon.

Tabell 2: Et sammendrag av hva slags muligheter, restreksjoner og krav som de ulike lisensene vi bruker i vårt prosjekt har. Denne tabellen er kompilert fra disse subsidene vist på fotnotene: ([1], [2], [3], [4])

Lisenskrav og kompatibilitet med kravspesifikasjon

I tabell 2 vises en forenklet oversikt over de kravene som de ulike eksterne programenes lisenser medfører for BioDemo. I denne delen er vi interessert i å finne ut om disse kravene kan stride med de som er satt hos oppdragsgiver. MIT-, BSD- og Apache-lisensene gir stor frihet til å gjøre hva man vil med dem, men med noen forbehold. Disse begrensningene går ut på å ikke bruke produktutviklerne til å promotere BioDemo, i tillegg til at lisensfiler, notisfiler og copyright må inkluderes. Mens de nevnte lisensene tilbyr stor frihet til å benytte disse slik man ønsker, så krever GPL en stor grad av åpenhet rundt kildekoden og at produktet kun blir brukt sammen med et annet open source program. LGPL må ikke eksplisitt benyttes sammen med en open source lisens. Den kan benyttes i kommersielle produkter, men krever en viss grad av åpenhet rundt kildekoden.
[18], [19]

Det konkrete kravet om åpenhet som LGPL medfører, er at den man distribuerer programmet til, skal ha mulighet til å benytte en siste versjon av LGPL-biblioteket. Man skal

[1] https://tldrlegal.com/license/mit-license
[2] https://tldrlegal.com/license/bsd-2-clause-license-(freebsd)
[3] https://tldrlegal.com/license/apache-license-2.0-(apache-2.0)
[4] https://tldrlegal.com/license/gnu-lesser-general-public-license-v2.1-(lgpl-2.1)

legge til rette for dette. Da kan man eksempel sende med kildekode eller objektkode, slik at den som tar i bruk programmet selv kan være ansvarlig for skaffe nyeste versjon av biblioteket. Om det resulterende programmet da ikke fungerer, er ikke den som distribuerte produktet ansvarlig for dette. Lisensen krever ikke at man må vise proprietær kode. Det strider derfor ikke mot oppdragsgivers krav om at enkelte deler av programmets kode ikke må vises åpent.
[20]

BioDemo skal kun distribueres for NBL-PCen. Om produktet skal distribuere videre, må denne problemstillingen tas opp av BioDemos eiere.

Hvis oppdragsgiver skulle ønske å distribuere en JAR-fil videre uten å vise kildekode eller objektkode, kan man skrive kode som oppdaterer build.xml (build path styres av denne) eller sende med build.xml slik at den det distribueres til kan oppdatere denne. Funksjonalitet for å gjøre dette kan være aktuelt for en fremtidig bacheloroppgave.

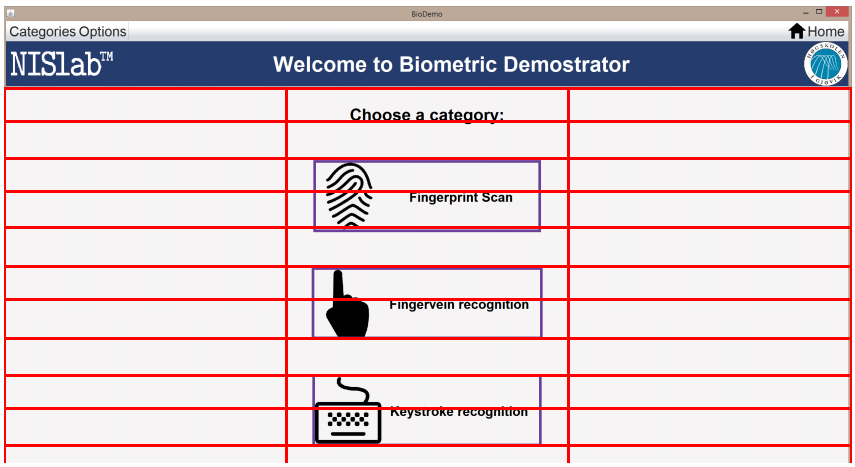
Løsninger underlagt linsenser

For info-tab-løsningen benytter vi et program som heter ”SHEF”. Dette er et HTML-redigeringsprogram for Java Swings grafiske brukergrensesnitt. [21] Dette programmet er underlagt LGPL 2.1 lisensen (se tabell 2).

Ikonene i BioDemo er hentet fra ”Freepik” - et nettsted som inneholder mange gratis grafiske ressurser. Lisensen lot oss bruke ikonene gratis i BioDemo, men må inkludere lisens og opplyse om at ikonene er ”created by freepik”. [22]

4.1.3 HomePanel

HomePanel er velkomstpanelet som brukeren først blir presentert for ved oppstart av BioDemo. Det er også dette panelet brukeren kommer tilbake til når vedkommende klikker på Home- knappen.



Figur 26: Oppdelingen av HomePanel vist med tabell

For å sette opp layouter i BioDemo og unngå duplisering av kode, ble det opprettet en egen funksjon for dette formålet: ”addComponent”. Parameterene i denne funksjonen setter GridBagConstraints for layouten, noe som vil si plasseringen av komponentene på panelet. Selve addComponent-funksjonen og en beskrivelse av dens parametere kan ses i kodeuttdrag 2.

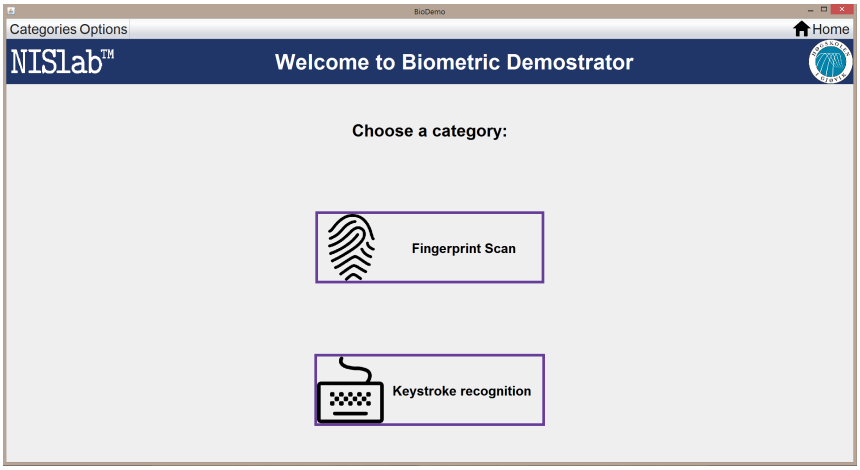
```
1
2   /**
3    * Utility for gridbagconstraint layout. Place a
4    * component with given GridBagcontraints
5    * @param panel - the panel where the components
6    * should be added to
7    * @param component - the component to be added
8    * @param row - the row the component should be placed in
9    * @param column - the column the component is placed
10   * @param width - the components width
11   * @param height - the components height
12   */
13  public static void addComponent(JPanel panel,
14    Component component, int row, int column,
15    int width, int height, int anchor) {
16
17      GridBagConstraints gbc = new GridBagConstraints();
18      gbc.insets = new Insets(0, 0, 0, 0);
19      gbc.weightx = 1;  gbc.weighty = 0.1;
20      gbc.gridx = column;
21      gbc.gridy = row;
22      gbc.gridwidth = width;
23      gbc.gridheight = height;
24
25      if(anchor != -1)
26        gbc.anchor = anchor;
27
28      panel.add(component, gbc);
29
30    }
```

Kodeutdrag 2: addComponent: Funksjonen som setter en gridBagLayout for et gitt panel.

”AddComponent” - funksjonen har også blitt benyttet i HomePanel for å tilpasse og legge til labels for de forskjellige modulene. Som det vises i fig 26 ble HomePanel delt opp i tre kolonner og tolv rader. Når en kolonne er fylt opp, tar hver modul- label opp en kolonne og to rader. I tillegg brukes en rad som mellomrom. Dette medfører at hver kolonne har plass til tre moduler. Ved færre moduler i en kolonne, vil mellomrommene tilpasse seg etter disse, som vist i fig 27. Et mål med BioDemo er at programmet enkelt skal kunne utvides til flere moduler. Dette er også noe vi har tatt hensyn til i HomePanel, der det er plass til å legge moduler også i den første og siste kolonnen. På denne måten kan HomePanel enkelt utvides for å huse opp til ni moduler, kun ved å sende med andre verdier for row og column (altså tredje og fjerde parameter i addComponent). (Se kode-utdrag 2).

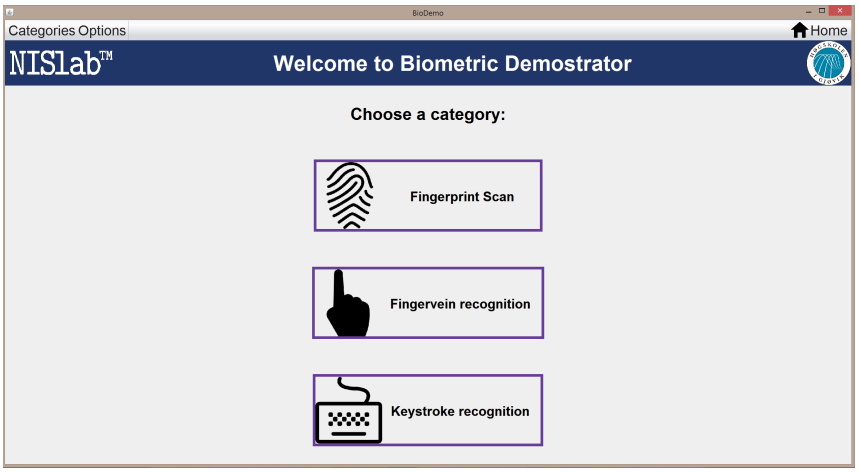
Fig. 27 viser et eksempel på to moduler fordelt på en kolonne (slik BioDemo ser ut i

dag). Parameteret column settes til 1 for begge moduler, og row settes til henholdsvis 3 og 6, og tilpasser seg automatisk (Altså med et noe større mellomrom enn ved tre moduler plassert i en kolonne).



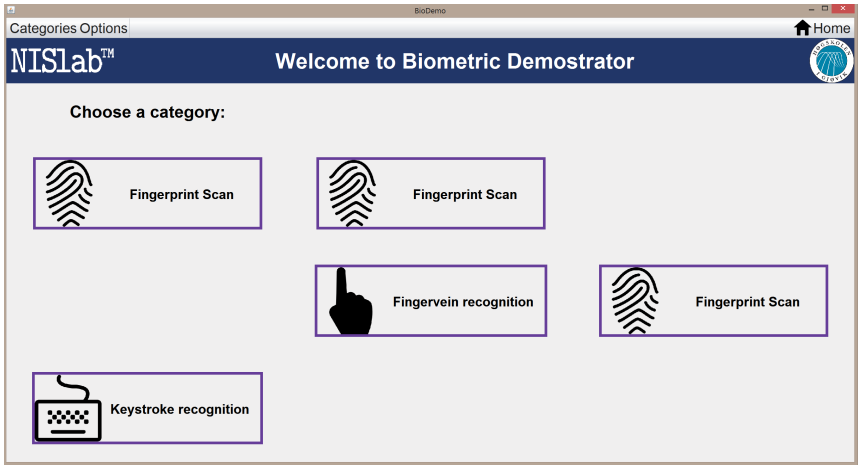
Figur 27: To moduler fordelt på en kolonne

Fig. 28 viser et eksempel der hele den midterste kolonnen (kolonne 1) er fylt opp med moduler. Her er parametrene henholdsvis column: 1 for alle moduler, og row: 3 for ”Fingerprint Scan”, 6 for ”Fingervein recognition” og 9 for ”Keystroke recognition”.



Figur 28: Tre moduler fordelt på en kolonne

Fig. 29 viser et eksempel på fem moduler som har blitt fordelt utover tre kolonner. Her ser man også tydelig hvordan ni moduler kan plasseres. (Vi replikerte ”Fingerprint scan”-boksen for å vise dette)



Figur 29: Fem moduler fordelt på tre kolonner

Kodeutdrag 3 viser hvordan en modul legges til på panelet. Her blir et ikon (i dette tilfellet ”fingerPrintScanIcon”) lagt på en JLabel med tekst (i dette tilfelelt fingerprint-Label). Labelen markeres med farge, og det legges på mouseListener, som reagerer på museklikk: sender brukeren til modulen og bevegelse over labelen, det bytter farge for å markere dette. Skal flere moduler legges til, kan denne kodesnutten kopieres, og man legger inn ikon, label, endrer på plasseringen (som man ser i linje 13), og angir hva som skal skje når labelet blir klikket på eller sveipet over.

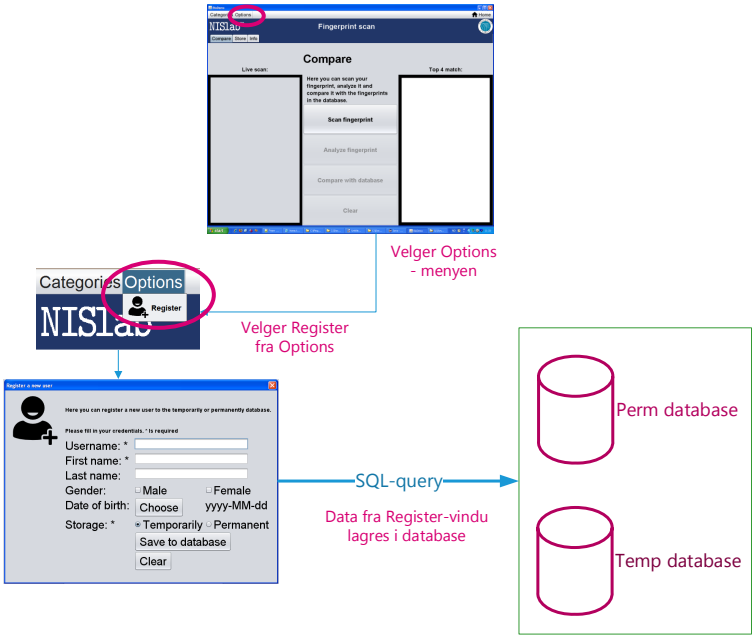
I kodeutdrag 3 blir et panel for fingeravtrykkmodulen lagt til i HomePanel. Ikon har vi fått fra en creative commons-kilde som det kan leses mer om i kapittel 4.1.2. På rad 12 og 13 blir ”addComponent”-funksjonen kalt med de parameterne den trenger. For dette objekter skal den plasseres i rad 3 og kolonne 1. Hvis det skulle bli behov for å justere plasseringen, er det i hovedsak kun snakk om å endre disse parameterene.

```
1   // The fingerprint option:
2   // Setting icon in a label with description:
3   fingerPrintScanIcon = new ImageIcon
4     (((new ImageIcon(getClass().getResource
5       ("/img/ikonCC/ikonFpCC.png")))).getImage().getScaledInstance
6         ((int) (screenWidth * 150 /1920),
7   (int) (screenWidth * 150 /1920), 0));
8   fingerprintLabel.setIcon(fingerPrintScanIcon);
9
10  constraints.weighty = 0.001;
11
12  RegisterPermOrTempUser.addComponent
13    (this, fingerprintLabel, 3, 1, 1, 2, GridBagConstraints.CENTER)
14
15  fingerprintLabel.setBorder(BorderFactory.createLineBorder
16    (UtillityClass.bluePurple, 8));
17
18  // Adding mouse listeners to the fingerprint label:
19  fingerprintLabel.addMouseListener(new MouseAdapter() {
20    @Override
21    public void mouseExited(MouseEvent e) {
22      fingerprintLabel.setForeground(Color.BLACK);
23    }
24    @Override
25    public void mouseEntered(MouseEvent e) {
26      fingerprintLabel.setForeground(UtillityClass.lightGray);
27    }
28
29    @Override
30      public void mouseClicked(MouseEvent e) {
31      // Gets the FPMainMenu object created at startup:
32      Start.fpMainMenuObj.changeBanner();
33    }
34  });
```

Kodeutdrag 3: Hvordan et panel i HomePanel legges til

4.1.4 Registrering av bruker

I henhold til kravspesifikasjonen (se kapittel 2.1.1) ”brukeren skal når som helst i programmet ha mulighet til å registrere seg i databasen midlertidig eller permanent”. Fig. 30 viser hvordan en bruker kan registrere seg i databasen. Brukeren har hele tiden tilgang til menybaren øverst i venstre hjørne. Herfra velger brukeren ”Options”, og deretter ”Register”. Herfra får brukeren opp et registreringsvindu. Brukernavn og fornavn benyttes for å unikt identifisere brukeren og må derfor tastes inn. Brukeren må også velge om vedkommende ønsker å bli lagret midlertidig eller permanent i databasen. I tillegg har brukeren mulighet til å taste inn fødselsdato, samt kjønn og etternavn, men dette er valgfritt. Etter at brukeren har taset inn sine data og trykket ”Send to database”, vil brukeren bli lagret i den permanente eller den midlertidige databasen, avhengig av hva vedkommende valgte under registreringen.



Figur 30: Viser hvordan en bruker registreres i databasen.

Brukernavn benyttes for å unikt identifisere hver bruker i databasen. Av denne grunn må brukernavnet være unikt. Gruppen har løst dette ved å ha en sjekk på om ønsket brukernavn finnes fra før når dette skal legges inn. Om så er tilfelle, vil brukeren få beskjed om dette, og blir bedt om å taste inn nytt brukernavn. Kodeutdrag 4 viser denne koden.

```
    if(Queries.userExists(username.getText(),
              (tempoarliyStorageRadioButton.isSelected() ?
                  CreateTempDb.URL_DB : CreatePermDb.URL_DB
                  )) == true)
          {
            JOptionPane.showMessageDialog(null, "Unfortuantly the"
                + "username '" + username.getText()
                + "' is already taken.
                + \n Please choose another username. ");
            username.setText(UtillityClass.EMPTY);
```

Kodeutdrag 4: Utdrag fra sjekk om brukernavn allerde finnes

Funksjonen ”userExists()”, som foretar selve sjekken på om brukernavnet allerede finnes. Kodeutdrag 5 viser denne koden.

```

  public static boolean userExists(String username,
    String url) throws SQLException {
    Connection con = null;
    java.sql.PreparedStatement prepS = null;
    String query =
        "Select username from userTable where username =?";


    try {
      // open connection, set value for placeholder, execute query
      con = DriverManager.getConnection(url);
      prepS = con.prepareStatement(query);
      prepS.setString(1, username.toLowerCase());
      java.sql.ResultSet res = prepS.executeQuery();

      while(res.next()) {          //loop thorugh
                        //if there is a match
        if(res.getString("username").equalsIgnoreCase(username))

          return true;         //return user exist
      }
```

Kodeutdrag 5: Utdrag fra spørringen som sjekker om brukernavnet eksisterer, og i såfall ignorerer forespørselen

Funksjonen returnerer true hvis brukernavnet allerede finnes, og da blir heller ikke spørringen sendt til databasen.

4.1.5 Sletting av midlertidig database

Det var et spesefikt krav fra oppdragsgiver at det skulle være mulig å lagre midlertidig data i databasen. Se kapittel 2.1.1 i kravspesifikasjonen for mer detaljer om dette. Det var viktig å sørge for at det aldri skulle være data i den midlertidige databasen ved oppstart av programmet. I kapittel 3.5 omtaler vi vurderingene vi gjorde ved design av databasen. Vårt valg falt på to databaser, der den ene var for midlertidig data og skulle slettes ved avsluttning av programmet, mens den andre var for permanent lagring. Vi vil nå forklare hvordan vi løste dette i BioDemo.

Ved avslutning av BioDemo kjøres det en funksjon som prøver å hente ut tabellene og sletter dem hvis dette er mulig. Vi har benyttet oss av "setDefaultCloseOperation", en funksjon av klassen JFrame i Java. Denne funksjonen gjør det mulig å definere selv hva som skal skje når man trykker på "x" i hjørnet av programvinduet [23]. Denne funksjonen kjøres også ved oppstart. Hvis det ikke finnes noen tabeller ved oppstart, betyr dette at slettingen ved avslutning var vellykket. Kodeutdrag 6 viser deler av denne slettefunksjonen. Funksjonen tar i mot en URL for den databasen som det skal slettes tabeller fra. På denne måten er det også mulig å slette all data fra den permanente databasen hvis det skulle bli behov, uten å måtte skrive en egen funksjon for dette.

```
1  /** Recives the database url, open connection and delete tables
2  * @param url - the location for the current database
3  * @throws SQLException
4  */
5  public static void dropTables(String url) throws SQLException {
6    try {
7      //Open connection
8      con = DriverManager.getConnection(url);
9
10     //Get the database metadata
11     java.sql.DatabaseMetaData dbm = con.getMetaData();
12
13     //get all the tables in APP schema
14     ResultSet tables = dbm.getTables(null, "APP", null, null);
15
16     //loop to go through all tables and exe DROP
17     while ( tables.next() ) {
18
19       //gets the current table name
20       String tableName = tables.getString("TABLE_NAME");
21
22       //create a statement
23       Statement stmt = con.createStatement();
24       //Drop if table exist
25       stmt.execute("DROP TABLE " + tableName);
26
27       stmt.close();          //close statement
28   }
29 /**Kode klippet ut
30 **/
```

Kodeutdrag 6: Sletting av tabeller i database

I tillegg til sletting av selve tabellene, er noen filer knyttet til den midlertidige databasen som også må slettes, som for eksempel fingeravtrykkbilder knyttet til en bruker i databasen. Kodeutdrag 7 viser hvordan denne slettingen foregår.

```
1
2     /** removes livescan and temp images
3     * Due to the NTFS file system, we need
4     * to update the timestamp for timeCreation for the creation time
5     *  to reflect the actual creation time of  the new files
6     * with the same names as the deleted ones
7     * "unknown_live_"
8     * @param f
9     * @param startText
10    */
11    public static void removeLiveScanImages(List<File> f,
12                                    String startText) {
13      for(int i = 0; i < f.size(); i++) {
14        if(f.get(i).isFile() && f.get(i).getName().
15                              startsWith(startText)) {
16          updateCreationTime(f.get(i));
17          f.get(i).delete();
18          f.remove(i);
19        }
20      }
21    }
```

Kodeutdrag 7: Sletting av fingeravtrykk-filer tilhørende midleretidige brukere eller ”live-scan”

4.1.6 Sikring av databasespørringer

I forbindelse med registrering av bruker vil brukeren ha anledning til å taste inn data (fornavn, etternavn, brukernavn, fødselsdato, kjønn etc.), i denne forbindelse er det viktig at brukeren ikke kan taste inn tegn som kan gjøre databasen korrupt, enten det skjer ved et uhell eller ved SQL-injection. SQL-injection er den vanligste SQL-relaterte sårbarheten, og skjer i forbindelse med at input tas fra brukerens inntastede data og settes sammen til en string. Denne stringen vil da inneholde spørringen som benyttes mot databasen [24]. Hvis stringen skulle inneholde databasespørringer, kan dette i verste fall medføre autroisert endring av databasen. Derfor er det viktig med tiltak som hindrer at ufiltrert input kan havne i stringen og benyttes mot databasen.

En vanlig metode for å forebygge SQL-injection er å benytte parameteriserte spørringer [24]. Dette vil si at det benyttes plassholdere for variabelparametre, og at disse parametrene bindes til en bestemt datatype før statementet utsendes. På denne måten hindres parametrene, uavhengig av innhold, i å bli tolket som SQL-spørringer. I Java finnes det en egen klasse, PreparedStatements, der hvert objekt representerer en ferdigbygget SQL-spørring. På denne måten kan samme spørring kjøres flere ganger uten å trenge å kompilere for hver gang. Dette var også noe vi så på som en fordel, da dette forbedrer ytelsen. På bakgrunn av dette, valgte vi å benytte PreparedStatements, for alle spørringer mot databasen.

Et eksempel på hvordan vi valgte å løse dette i forbindelse med brukerinput kan sees i kodeutdrag 8. Her utfører ”ExecuteUpdate()” oppdateringen av databasen, og returnerer 0 hvis ingen rader ble oppdatert, eller et positivt tall for det antall rader som ble oppdatert. Derfor får brukeren beskjed om at vedkommende er registrert når et tall større enn 0 returneres.

```
1
2   // The query for inserting user data as a string:
3   String insertQuery =
4   "INSERT INTO UserTable (userName, "
5   + "firstName, lastName, DOB, gender) "
6   + "VALUES(?,?,?,?,?)";
7
8   try{
9   //Connection to database
10  con = DriverManager.getConnection(url);
11  prepS = con.prepareStatement(insertQuery);
12  // Inserts username into first column
13  prepS.setString(1,username.toLowerCase());
14  // Inserts users firstname into second column
15  prepS.setString(2,firstName);
16  // Inserts users lastname into third column
17  prepS.setString(3,lastName);
18
19
20
21  //exe the sql statement and
22  //message if row was updated
23  if(prepS.executeUpdate() > 0)
24
25  JOptionPane.showMessageDialog(null,
26  +"You have been succesfully registered.");
```

Kodeutdrag 8: Utdrag fra kode med preparedStatements som brukes i forbindelse med brukerinput.

4.2 Fingeravtrykk

Fingeravtrykksmodulen var den første modulen vi begynte på og består av flere forskjellige klasser og funksjoner. I vedlegg F vises en detaljert oversikt over aller klasser, variabler og funksjoner for denne modulen. Vi har valgt å fremheve noen av funksjonalitetene for denne modulen for å gå i mer detalj om hvordan vi løste den spesefikke utfordringen.

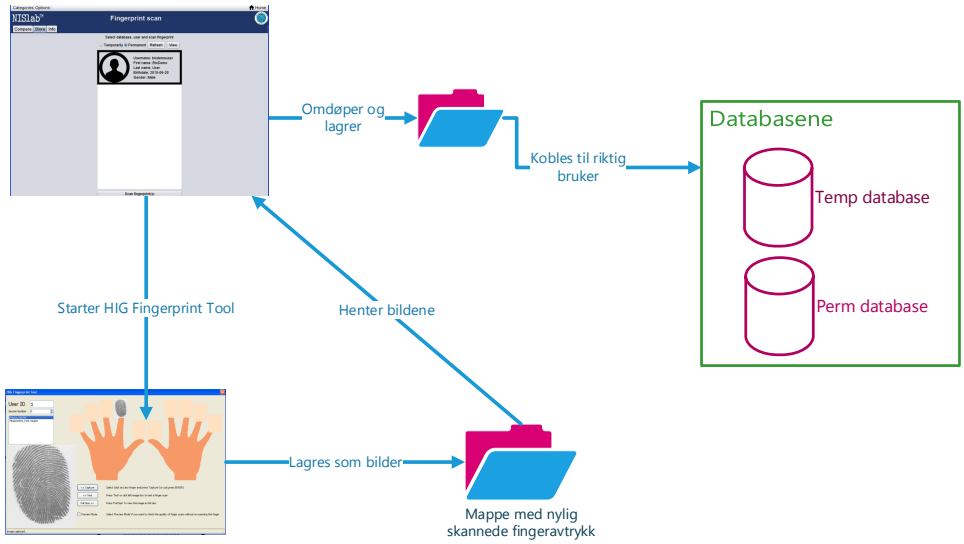
4.2.1 Fingeravtrykkmodul

Skanning av fingeravtrykk

Vi har lagt opp til at brukere kan skanne sitt fingeravtrykk uten å måtte registere seg i databasen. I tillegg har vi utviklet programmet slik at brukere kan velge å registere seg midlertidig eller permanent i databasen og skanne og lagre opp til ti fingeravtrykk i databasen.

Vi har illustrert hele denne prosessen i fig. 31. Denne figuren er en grovt oversiktlig

figur over alle aktørene i prosessen. Skanning av fingeravtryk skjer i et eksternt program (HIG Fingerrpint Tool, se kapittel 4.2.2), men i BioDemo kalles dette programmet og det kjøre mens BioDemo ”venter” til det er ferdig. Kodeutdrag 9 viser akurat denne koden. I HIG Fingerprint Tool blir de nyinnskannede fingeravtrykkene lagret som bilder i en mappe. Men for å tilpasse slik at vi kunne utnytte disse bildene i BioDemo ble vi nødt til å flytte på disse bildene og omdøpe dem. Denne prosessen kan det leses om i kapittel 4.2.3.



Figur 31: Prosessen for å skanne inn en finger for en bruker i databasen

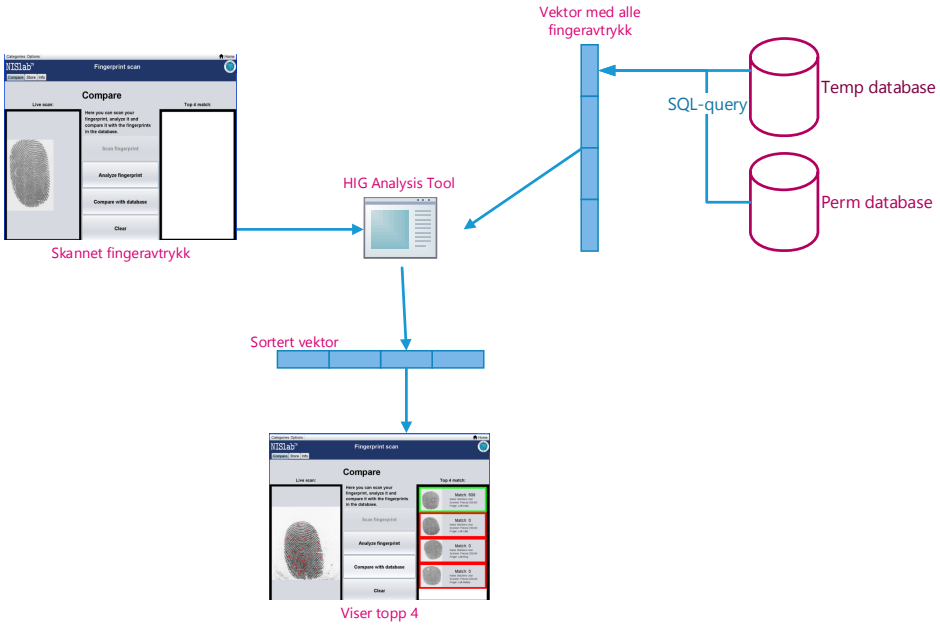
```
1  /**
2  * Start the capturing software and waits for it to finish
3  * @param username the username i want the image to be renamed with
4  * @param mode the mode i want the image to have in its name
5  * @return the return value from listFileRename(...)
6  */
7  public static List<File> startScanSoftware(String username,
8            String mode) {
9
10   // needed to start the capture program
11   File pathToExecute = new File
12               ("C:/Documents and Settings/Administrator/" +
13                 "Desktop/Scanner_program/" +
14                 "Fingerprints Fast24052012/Fingerprints Fast/" +
15                 "bin/Debug/Fingerprints Fast.exe");
16
17   ProcessBuilder builder = new ProcessBuilder
18                      (pathToExecute.getAbsolutePath());
19   builder.directory( new File
20               ("C:/Documents and Settings/Administrator/" +
21                 "Desktop/Scanner_program/" +
22                 "Fingerprints Fast24052012/Fingerprints Fast/" +
23                 "bin/Debug/").getAbsoluteFile());
24
25   builder.redirectErrorStream(true);
26   try {
27     Process process = builder.start();
28
29     // Wait to run the code beneath until the capturing program
30     // has been closed
31     process.waitFor();
32     return listFileRename("C:/Fingerprints/", username, mode);
33
34 /**
35 *Klippet ut kode
36 **/
37 return null;
38 }
```

Kodeutdrag 9: Starter fingeravtrykkskannerprogrammet og venter til den er avsluttet

Sammenligning av fingeravtrykk

Ved en typisk demostrasjon vil en gjeste-bruker eller NBL-bruker skanne sitt fingeravtrykk ved hjelp av det eksterne programmet HIG Fingerprint Tool. Når fingeravtrykket har blitt vellykket skannet, kan dette sammenlignes med databasene og alle deres registertes fingeravtrykk. Deretter vil de med topp fire matchtall vises i en topp fire-liste. Hver av disse fire skal ha fargeindikatorer (rød og grønn) som forteller om det er match eller ikke. for at en match skal bli grønn, må matchtall være over eller lik 48.

Denne prosessen er ganske omfattende og innebærer større SQL-spørringer, og mange sammenligninger. For å forenkle denne prosessen, har vi lagt til en figur som illusterer dette, (fig.32). Her sees de to databasene på øvre høyre side. En SQL-spørring kjøres som henter ut alle fingeravtrykk og legger det i en vektor. Denne spørringen kjøres på begge databaser. Kodeutdrag 10 viser denne spørringen. Her blir resultatet av spørringen lagt i en vektor ”data” av klassen ”FPMatcher”. Denne vektoren blir så videre brukt i sammenligningsanalysen. Sammenligningen skjer i et eksternt program HIG Analysis Tool som du kan lese mer om i kapittel 4.2.2. Hver enkelt objekt i ”data” blir sammenlignet med det nylig innskannede fingeravtrykket, og et matchtall blir produsert. Utfra dette matchtallet blir alle objektene sortert i en ny vektor. Til slutt blir kun de fire første i denne vektoren vist i vinduet for brukerene. Hver av disse fire blir omringet av enten en rød eller grønn firkant, avhengig om de har en match overstiger 48 eller ei (Se kravspesifikasjonen i kapittel 2.1.2).



Figur 32: Prosessen for å sammenligne ett fingeravtrykk med databasen

```
1
2       /**
3       * Get data from the database on a current user and h*s fingers
4       * @param url - location of the database to be queried
5       * @return data - A vector contains the data from the
6       *      database within
7       * FPMtacher objects
8       * @throws SQLException
9       */
10      public static Vector<FPMatcher> selectFingerItem(String url)
11                    throws SQLException {
12
13      Connection con = null;
14      java.sql.PreparedStatement prepS = null;
15      String insertQuery = null;
16
17      Vector<FPMatcher> data =
18          new Vector<FPMatcher>();
19
20      StringBuilder sB = new StringBuilder();
21      sB.append("SELECT ut.username, ut.firstname, ut.lastname, ");
22      sB.append(" fp.fingerprintno, fp.scannertype, fp.fingerprintpict,
23           fp.quality ");
24      sB.append(" FROM usertable ut ");
25      sB.append(" JOIN fingerprint fp ");
26      sB.append(" on ut.username = fp.username " );
27
28
29      String query=sB.toString();
30
31      try {
32        con = DriverManager.getConnection(url);
33
34        prepS = con.prepareStatement(query);
35        java.sql.ResultSet res = prepS.executeQuery();
36
37      // loop as long there is data to process
38      //add object to vector containing the result (res)
39        while (res.next()) {
40          data.addElement(
41            new FPMatcher(
42              res.getString("username"),
43              res.getString("firstname"),
44              res.getString("lastname"),
45              res.getString("scannertype"),
46              res.getInt("fingerprintno"),
47              res.getString("fingerprintpict"),
48              res.getInt("quality")));
49        }
50
51      /**
52      ** Kode klippet ut
53      **/
54      }
55      return data;            //return the vector
56      }
```

Kodeutdrag 10: SQL-spørring for å hente ut alle fingeravtrykk i databaser

4.2.2 Eksterne programmer

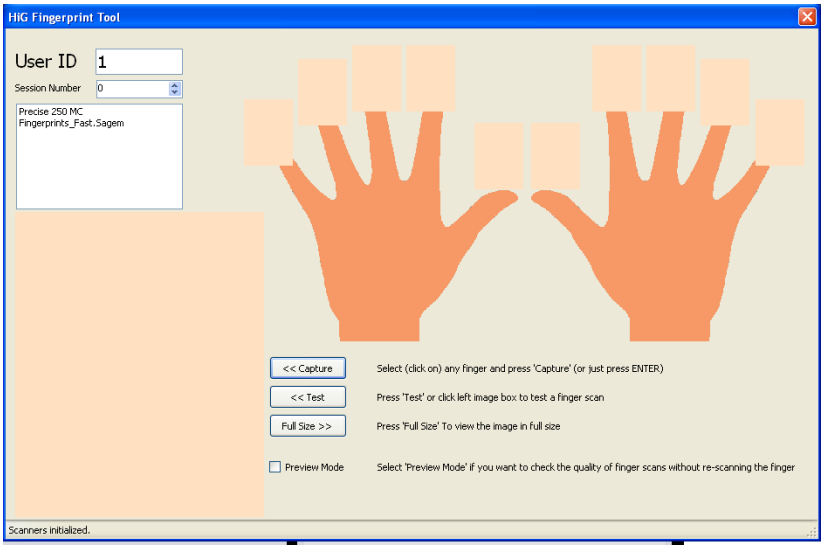
I fingeravtrykksmodulen benyttet vi oss av to eksterne programmer, der begge var skrevet i C#. Begge programmene var også bundet til lisenser, noe som medførte at flere hensyn måtte tas, både i forbindelse med tilpassning og offentlig visning.

Skanneprogrammet

Skanneprogrammet som skulle benyttes er HiG Fingerprint Tool. Dette programmet presenterer brukeren for et enkelt grafisk grensesnitt, som hovedsakelig består av to hender med en rute over hver finger der fingeravtrykket som har blitt skannet inn vises i fig 33. Programmet lar brukeren skanne inn ett fingeravtrykk pr. finger. Dette skjer ved at brukeren legger på en finger på skanneren og samtidig klikker på den fingeren på bildet som skal skannes inn.

På grunn av lisensering kunne dette programmet kun benyttes på NBL-PCen, så all utviklingen av denne delen måtte skje på denne PCen. Programmet kunne heller ikke endres eller tilpasses, og måtte benyttes i sin helhet. Dette medførte endringer i vårt design, da vi hadde regnet med å kode opp GUI for denne delen selv. Løsningen innebar at man klikker på ”Scan fingerprint”- knappen i BioDemo og derfra kjøres skanneprogrammet i et popupvindu.

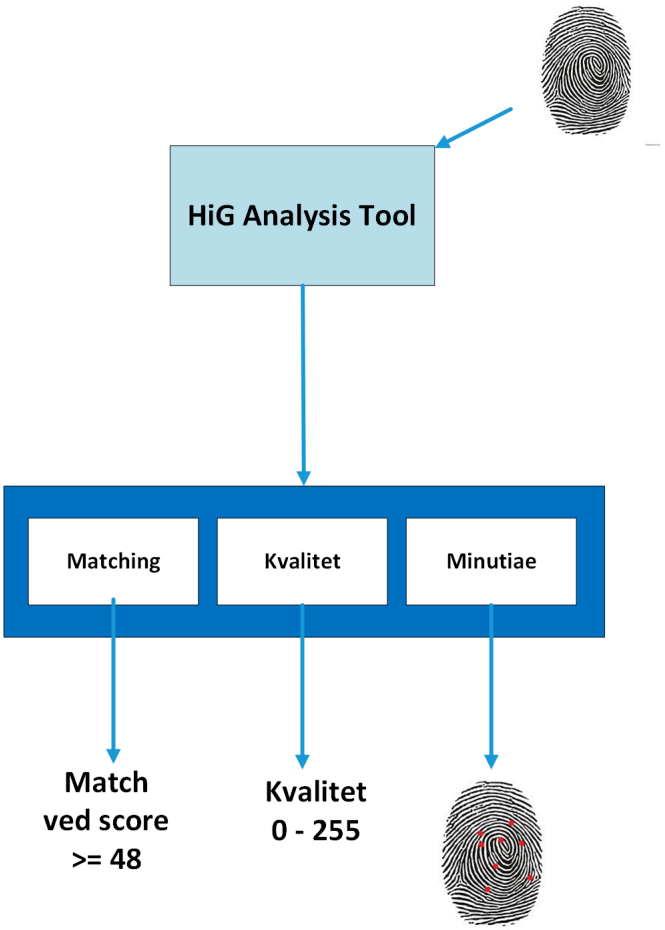
En av de større utfordringene med dette programmet var at det var ganske ustabilt og skapte feilsituasjoner som vi ikke direkte kunne fikse. Disse feilsituasjonene og hvordan vi håndtere den er nevnt i kapittel 6.3.



Figur 33: HiG Fingerprint Tool

Analyseprogrammet

HIG Analysis Tool er et C#-program bestående av funksjoner som analyserer et fingeravtrykksbilde og returnerer en match-score. I tillegg tar det i mot et fingeravtrykksbilde, og returnerer det samme bildet, men markert med minutiae-punkter. Disse funksjonene er etter ønsker fra oppdragsgiver ikke offentliggjort, og kunne derfor ikke vises i rapporten. Fig. 34 viser at HiG Analysis Tool tar inn et fingeravtrykksbilde (øverst i høyre hjørne på figuren), og forskjellige funksjoner returnerer følgende til BioDemo: fingeravtrykksbildet markert med minutiae-punkter, kvaliteten på fingeravtrykksbildet - en verdi mellom 0 og 255, samt matcscore, der en score på 48 eller høyere indikerer match. Når det skal sammenliges to fingeravtrykk blir to bilder av de bestemte fingeravtrykkene sendt med og resulterer i en tallverdi.

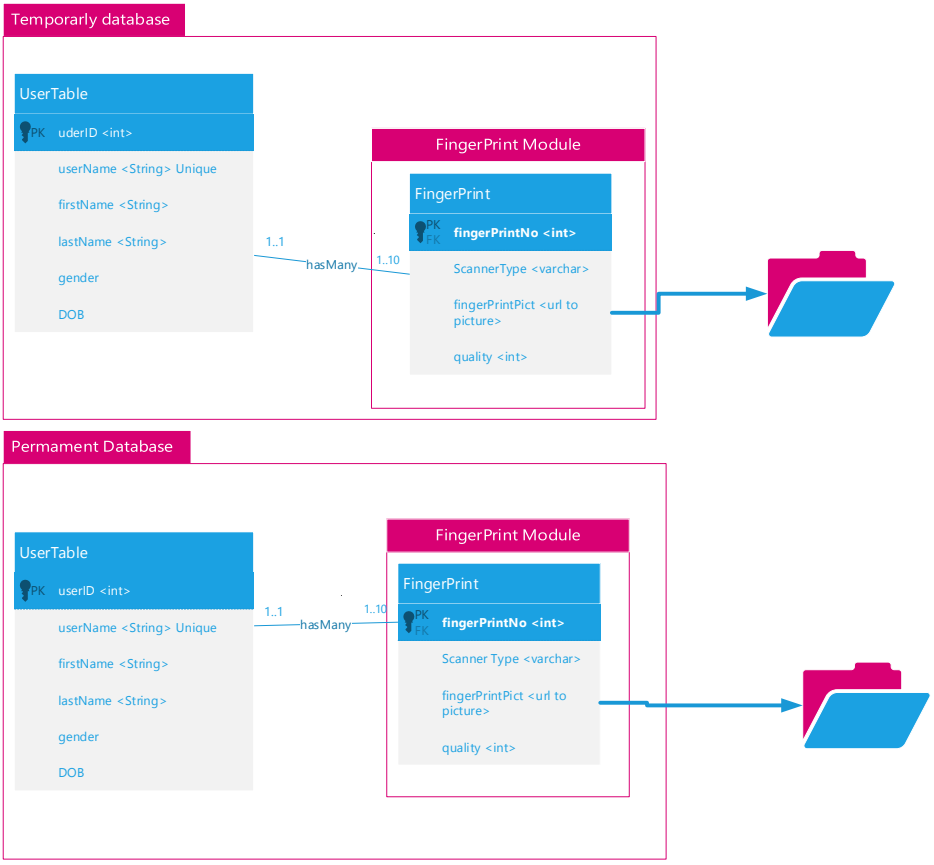


Figur 34: Viser hvordan analyseverktøyet benyttes.

Noen utfordringer oppstod i forbindelse med dette programmet. Dette dreide seg i hovedsak om at programmet nektet å analysere bilder av for dårlig kvalitet. I tillegg noterte vi oss at programmet har utfordringer med å gi bilder noen annen kvalitetsscore enn 0 og 255 - som tilsvarer dårlig og god kvalitet. Hvordan vi forholdt oss til flere av disse problemene er forklart i kapittel 6.3.

Databasetilpasninger

Fig. 35 viser det konseptuelle designet på databasen når det gjelder fingeravtrykkmodulen. Som figuren viser, kan vi se at det er to forskjellige seperate databaser; en for midlertidig og en for permanent lagring. ”fingerPrintPict” i ”Fingerprint”-tabellen består av en string som peker på filen hvor det aktuelle fingeravtrykkbildet ligger. Dette vises i figuren med piler til mappene.



Figur 35: UML av databaseutseende for Fingeravtrykk-modulen

4.2.3 Filhåndtering knyttet til fingeravtrykkmodulen

Som tidligere nevnt, tar HiG Fingerprint Tool seg av innskanning av fingeravtrykk og lagring av fingeravtrykksbildene som blir generert i forbindelse med dette. Dette programmet oppretter en mappe for hver tilkoblede sensor og navngir bildefilen på følgende måte: 1_0_Fingernummer. Et bilde tatt av høyre pekefinger vil f.eks. få filnavnet: ”1_0_2”.

På grunn av linsensering var det ikke mulig å tilpasse dette. Brukeren angir hvilken sensor som skal benyttes, og skanner inn sitt fingeravtrykk. Programmet tar så bildet som ble generert og lagrer det i den forhåndsdefinerte mappen med et autogenerert navn. Utfordringen ble da å kunne knytte dette bildet til en spesifikk bruker i databasen. Dette ble løst på følgende måte:
Funksjonen ”listFileRename”, som er av typen List, og tar inn mappenavn, brukernavn og om det er midlertidig, permanent eller livescan (se kodeutdrag 11). Kodeutdraget viser bare de to første casene: for første og andre finger, men casene blir tilsvarende for resterende fingre. Denne funksjonen går gjennom alle filene i mappen, endrer filnavnet og plasserer filene i ny mappe.
Selve navneendringen foregår på følgende måte:

For livescan, vil filnavnet se ut som følger: ”unknown” etterfulgt av ”live” og en tilfeldig verdi under 100 000 (mer om dette under), deretter fulgt av fingernummeret. Filnavnet vil f.eks. kunne se slik ut: ”unknown_live_503_2”.

For fingeravtrykk som skal lagres på en bestemt bruker i databasen, vil filnavnet se ut som følger: Brukernavnet etterfulgt av ”temp” eller ”live” (avhengig av om brukeren er registrert midlertidig eller permanent), og en tilfeldig verdi under 100 000 (mer om dette under), deretter fulgt av fingernummeret. Filnavnet vil f.eks. kunne se slik ut: ”BioDemoUser_perm_1008_5”.

Som man ser, følger modusen bak brukernavnet. Modusen kan ha tre verdier: ”temp” - markerer midlertidig lagring, ”perm”-markerer permanent lagring, eller ”live”- markerer livescan. Det siste tallet representerer fingernummer, og tilsvarer det siste tallet i det opprinnelige filnavnet. Fig. 31 illustrerer hele denne prosessen.

Disken hadde ytelsesutfordringer (se mer i kapittel 6.1.1). I denne forbindelse valgte gruppen å benytte memory mapped I/O, fordi RAM er, som kjent, raskere enn disk, vi benyttet dette for å øke ytelsen. Et problem som dukket opp på bakgrunn av dette, var etter sletting av filer. Hvis en av fingeravtrykksbildene ble slettet, og det ble lagret et nytt bilde med samme filnavn (F.eks. i tilfeller der det var snakk om samme bruker og samme finger), så ville bildet som allerede var slettes returneres, nettopp fordi det fremdeles lå i RAM. Denne utfordringen ble løst ved å legge til en tilfeldig verdi mellom 0 og 100 000 i filnavnet. Dette anså gruppen som tilstrekkelig for å hindre at flere filer kunne ha samme navn, og sikre at ikke et bilde som alt var slettet ble returnert.

Et annet hensyn som måtte tas, var kvaliteten på bildene som ble lagret. I kapittel 6.3.1 omtales potensielle feilsituasjoner, og under ”Feil skannet fingeravtrykk i skanneprogram” vises det at analyseprogrammet ikke vil utføre analyser på fingeravtrykk som

```
/**
 * Modified code from
 * www.stackoverflow.com/questions/14676407/list-all
 * -files-in-the-folder-and-also-sub-folders
 * by user Cyrille Ka
 * Recursivly seaRch through all subdir of a folder and find all
 * files should start like this 1.0.* = id.session.finger id and
 * session will be constant replace existing replaces a file if
 * that already exists
 * @param directoryName the directory we will start to seach from
 * @param username the user that owns that fingerprint
 * (can be unknown)
 * @param mode perm || temp || live
 * @throws IOException
 */
public static List<File> listFileRename(String directoryName,
    String username,
      String mode) throws IOException {
  File tmp = null;
  File directory  = null;
  // where we start to search
  directory = new File(directoryName);

  // store it in this list
  List<File> resultList = null;
  resultList = new ArrayList<File>();

  // getting all the files from the directory
  File[] fList = directory.listFiles();

  for (File file : fList) {
     if(file.isFile()) {
      // remove the dot and the file extension
      // (so that we can use this code for all image file  types)

        switch(file.getName().replaceFirst("[.][^.]+$", "")) {

        case "1_0_1" : Files.move(file.toPath(),
            (tmp = new File(file.getParent(), username
                + "_" + mode + "_1"
                  + getFileExtension(file))).toPath(),
                    REPLACE_EXISTING);
          resultList.add(tmp);

      break;
      case "1_0_2": Files.move(file.toPath(),
          (tmp = new File(file.getParent(), username
                 + "_" + mode + "_2"
                 + getFileExtension(file))).toPath(),
                REPLACE_EXISTING);
          resultList.add(tmp);

      break;
  /**Kode klippet ut
  **/
```

Kodeutdrag 11: Funksjon som endrer navn på fingeravtrykksfilene

har for dårlig kvalitet. På bakgrunn av dette, ble det behov å slette disse filene, slik at det ikke oppstod konflikter med analyseprogrammet. Kodeutrag 12 for dwne løsningen.

```
  /**
  * Goes through all the image files (fingerprints) and delete the
  * images with too low quality
  */
  public static void deleteFilesWithBadQuality() {
    StringBuilder builder = new StringBuilder();

    try {
      List<File> lookForUserData =
        listFileRename("C:/Fingerprints/", "", "");

    // connecting to program to see quality
      try {
        Bridge.setVerbose(true);
        Bridge.init();
        Bridge.LoadAndRegisterAssemblyFrom
        (new File("./jni4netFolderForFingerprint
          +/FingerprintFunctions.j4n.dll"));

// delete all files with to bad quality to be stored in the database
        for (File file : lookForUserData) {
          if(file.isFile() && file.getName().contains("_")) {


            byte[] fileContent = Files.readAllBytes(file.toPath());

// delete file if too low quality
               if(FingerprintFunctionalities.
                 ObtainSampleQualityByImage(fileContent) == 0) {
                 builder.append(file.getAbsolutePath());
                 builder.append("\n");
                 // append file abbsolute path in Stringbuilder
                 // display in JOptionpane in a JTextArea
                 file.delete();
               }

            }
          }

          if(builder.length() > 0)
// print all the removed files in a Joptionpane JtextArea combination
            JOptionPane.showMessageDialog(null,
              new JTextArea(builder.toString()),
                "Deleted files " +
                "from disk because of bad quality",
                JOptionPane.WARNING_MESSAGE);
```

Kodeutdrag 12: Funksjon som sletter fingeravtrykk med dårlig kvalitet

4.3 Tastaturgjenkjenning

Tastaturgjenkjenning var den modulen vi tok for oss etter fingeravtrykksmodulen. Klassene i denne modulen er satt opp mye på samme måte som fingeravtrykksklassene.

4.3.1 Tastaturgjenkjenningsmodul

Definisjon av tastaturgjenkjenningsord:

Denne modulen lar brukeren taste inn et bestemt ord, og så lage en template på brukeren ut fra inntastingsdata, (hvor lang tid det tar fra brukeren trykker ned tasten og frem til vedkommende slipper tasten). Som vist i kravspesifikasjonen i kapittel 2.1.4, skal brukeren kunne velge et bestemt ord fra et forhåndsdefinert utvalg i databasen, som vedkommende ønsker å taste inn. For å angi hvilke ord som skal benyttes for tastaturgjenkjenning, valgte vi å opprette en array i "Utilities"-klassen inneholdende disse ordene. Denne arrayen kan ses i kodeutdrag 13. Arrayen sendes så til databasen, slik at radene for de gitte radene opprettes.

Ønskes det å legge til flere ord for tastaturgjenkjenning, legges disse til på slutten av arrayen. Skal allerede eksisterende ord editeres, må tabellen "keystroke" slettes før det utføres flere tastaturgjenkjenninger, slik at det ikke blir konflikter med ordene.

```
1  /*
2   * array for the keywords to choose between in keystroke module.
3   * To add more please fill in after the last.
4   * NOTE if the existing words are edited the
5   * values in database makes no sense.
6   * Please delete the "keystroke" table(or current row)
7   * in permDB before editing the words.
8   */
9  public static final String[] KEYWORDS =
10   new String[] {"welcome42", "password", "Password", "Aidu50Xv",
11   "Longer sentences are also possible", "123456"};
```

Kodeutdrag 13: Viser arrayen i Utilities-klassen som angir hvilke ord som kan tastes inn av brukeren.

Vi ser at vi heller burde ha valgt en mer dynamisk løsning, og ikke hardkode verdier inn i en array, men på grunn av begrenset med tid, var dette noe vi valgte å ikke prioritere.

Inntasting og lagring av ord:

En bruker kan legge inn tastetrykk på to måter:
En eksisterende bruker kan velge "Store" - fanen i keystrokemodulen, herfra velge sin bruker, og deretter velge å legge inn et ord på denne brukeren. Hvert ord må her tastes inn ti ganger, slik at det kan genereres et gjennomsnitt. Brukeren har også mulighet til å gå rett til "Compare"-delen, og begynne inntastingen øyeblikkelig uten å registrere seg, men da tastes ordet inn en gang.

I forbindelse med inntasting, vil det eksterne programmet DemoKD startes opp (Mer om hvordan dette fungerer i kapittel 4.3.2) og registrerer inntastingen av ordet. Kodeutdrag 14 viser hvordan dette lagres i databasen.

For hver bruker opprettes det en egen XML-fil som inneholder alle brukerens inntastede ord, og måledata (”key up”, ”key down”, ”latency”) for inntasting av hvert tegn i dette ordet. Når et ord skal legges inn i databasen (se.fig 14), legges det inn sammen med en URL til denne XML-filen (”String xmlFile”), som holder på ordets måledata. Mer om selve filhåndteringen kan det leses om i kapittel 4.3.5.

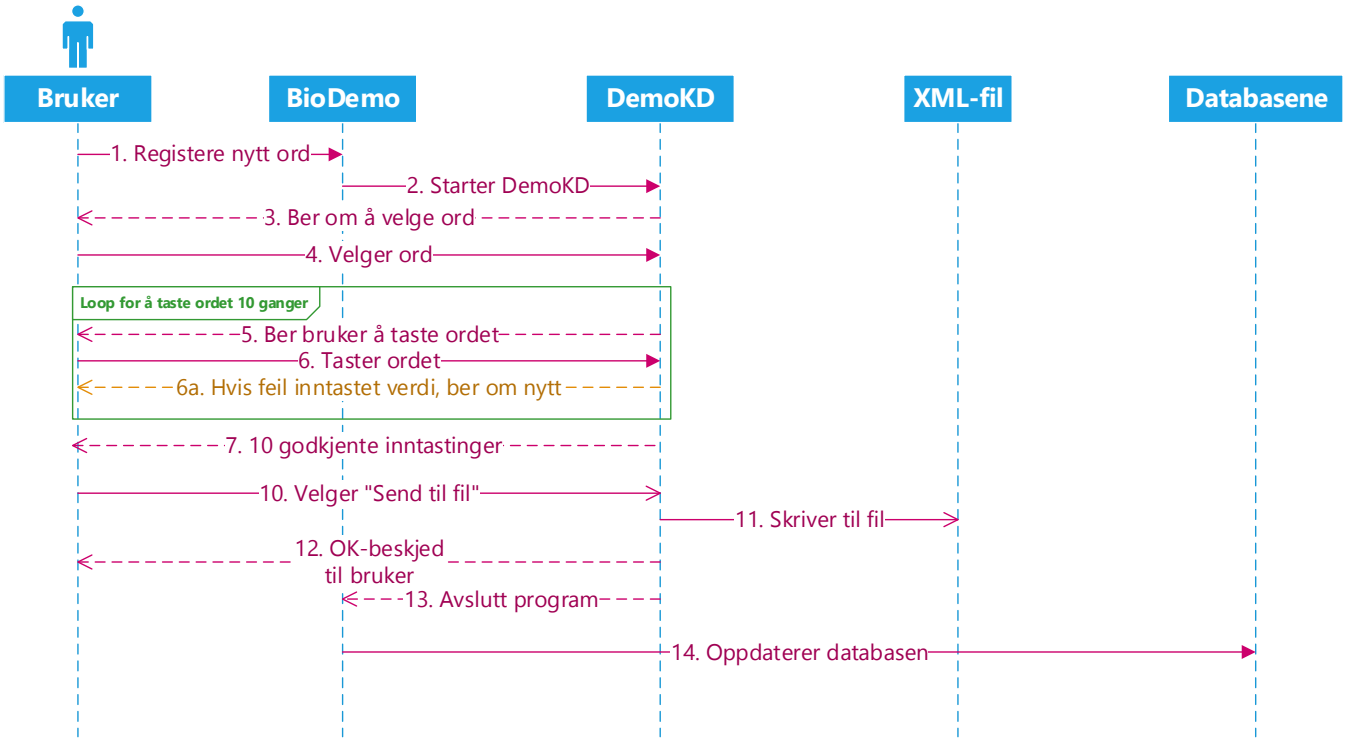
```
/**
* Function that insert a current word to one user
* @param username - the current username.
* MUST exist in the "usertable" beforehand
* @param word - The current word that is going to get inserted
* @param xmlFile - url to the xml file with keystroke values
* @param dbUrl - url to current database (temp or perm)
* @throws SQLException
*/
public static void insertKeyStrokeWord(String username,
    String word, String xmlFile, String dbUrl)
        throws SQLException {

  Connection con = null;
  java.sql.PreparedStatement prepS = null;

  String insertQuery =
  "INSERT INTO keystroke(word, username, wordvalues)"
  + "VALUES(?,?,?)";

  try{
    //connection to database
    con = DriverManager.getConnection(dbUrl);
    prepS = con.prepareStatement(insertQuery);

    //insert the values to the right columns
    prepS.setString(2,username.toLowerCase());
    prepS.setString(1,word);
    prepS.setString(3,xmlFile);

    //exe the statement
    prepS.executeUpdate();
  }
/** Kode klippet ut (catch og finally)
*/
```

Kodeutdrag 14: Viser hvordan et ord legges til på brukeren i databasen.

Sekvensdiagram for inntasting av ord

Sekvensdiagrammet i fig. 36 forutsetter at brukeren allerede har registert seg som bruker i databasen. Mesteparten av kommunikasjonen skjer mellom bruker og DemoKD, men prosessen starter ved at brukeren ber om å taste inn ett nytt ord. BioDemo starter da DemoKD-programmet og videre kommunikasjon skjer mellom bruker og DemoKD. I mellomtiden ”venter” BioDemo på at denne prosessen er ferdig. Når det gjelder brukere som skal registreres i databasen, blir brukeren bedt om å skrive ordet inn ti ganger, og derav den grønne boksen i figuren. DemoKD skriver til en XML-fil når ti godkjente ord er

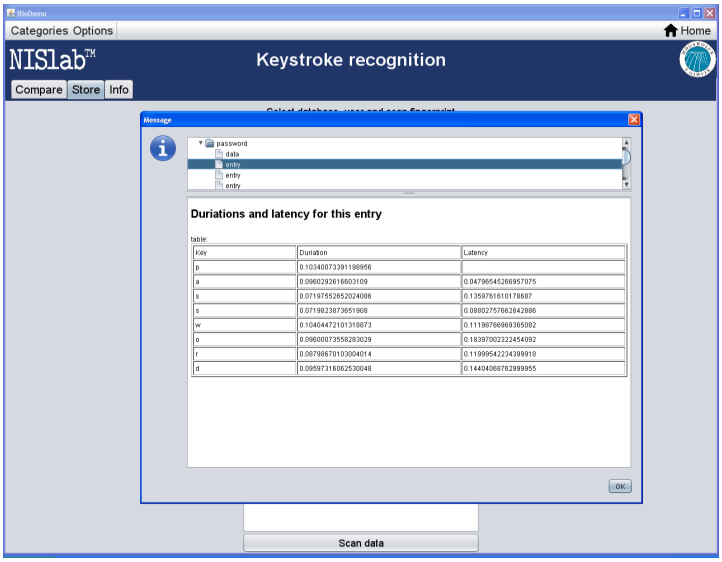
skrevet inn, og gir beskjed om dette til bruker. Deretter avsluttes DemoKD og BioDemo oppdaterer databasen med de nye endringene.



Figur 36: Sekvensdiagram for inntasting av ord

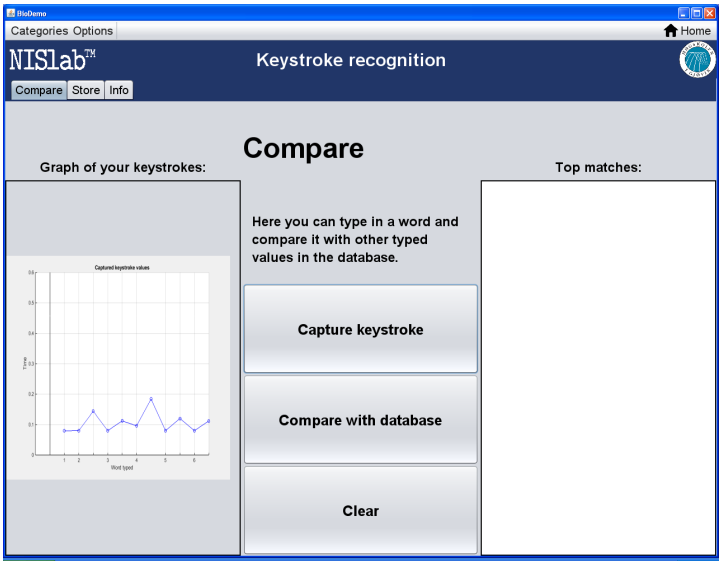
Visning av eksisterende data på en bruker

Under ”Store”-fanen er det mulig for brukeren å sjekke hvilke ord som vedkommende har fra før og evtentuelt det nylige inntastede ordet. Løsningen for dette er et vindu som viser dataene i den aktuelle XML-filens innhold. Dataene er gjort lettleslig for brukere og verdiene er i sekunder. Fig. 37 viser hvordan det ser ut når en bruker ber om å få se data som er lagret om vedkommendes inntastinger.



Figur 37: Viser hvordan XML-filen til brukeren vises.

Sammenligning og opptegning av graf



Figur 38: Viser grafen som genereres ut fra brukerens inntasting

Etter inntasting, skal brukeren kunne sammenligne sin inntasting mot databasen. I denne modulen har fire MATLAB-funksjoner blitt benyttet for å sammenligne distansetallene som blir generert i forbindelse med kjøring av DemoKD-programmet. To av disse funksjonene har NBL laget. Disse kan ikke vises i rapporten, men er beskrevet nærmere i kapittel 4.3.2. Gruppen har derimot også laget to egne funksjoner, som kan vises i denne rapporten. Disse grafene er som følger:

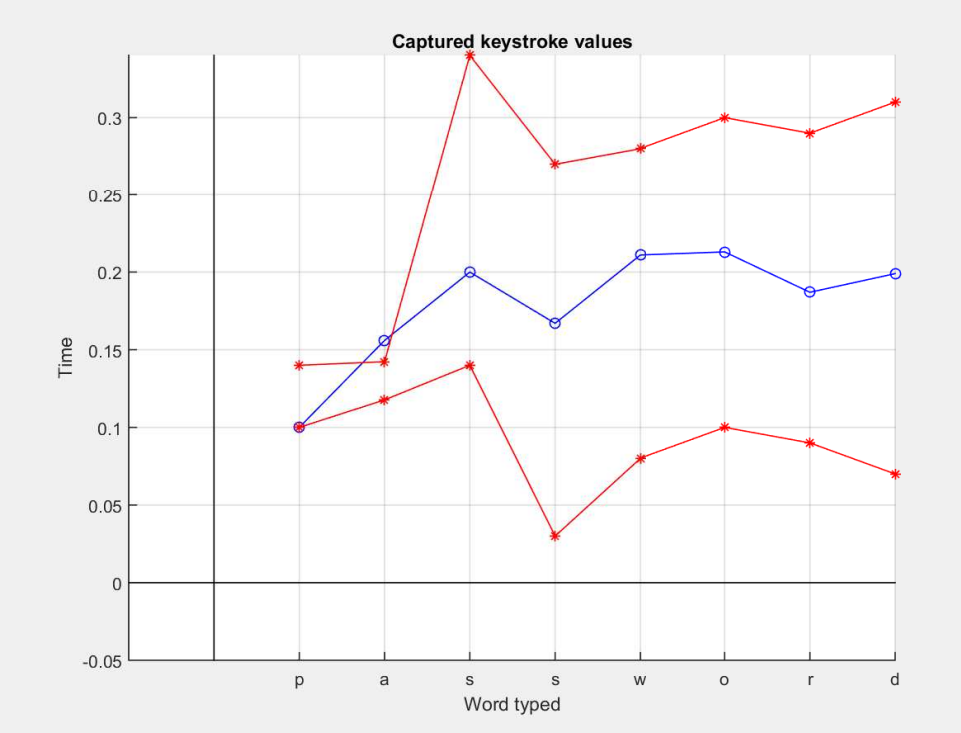
OneLineGraph2500.m (se kodeeksempel 15), tar i mot ”x1”, ”y1” og ”word” som argumenter og tegner opp en graf ut fra verdiene. Denne funksjonen benyttes for å vise grafen

til verdiene som brukeren tastet inn. ”word” er ordet er inntastet og settes på x-aksen. Fig 38 viser hvorden en slik graf ser ut i BioDemo.

```
1
2  %A one line graph, Takes the x-values, y-values and
3  %the words as arguments
4  function [] = oneLineGraph(x1, y1, word)
5  opengl software;
6  fig = figure;
7  set(fig,'Visible', 'off');
8  hold on;
9
10 %Plot the values in a blue line with circles
11 plot(x1,y1,'-bo');
12
13 %specify the axis (x and y)
14 len=length(x1);
15 axis([-1 len 0 0.6]);
16
17 %add a line through 0
18 plot([-10 len], [0 0], '-k',[0 0], [-10 10], '-k');
19
20 %formatting the graph with squares, labels and title
21 grid on;
22 ylabel('Time');
23 xlabel('Word typed');
24 title('Captured keystroke values');
25 % Increment by one on x axis instead of five and increment to len
26 set(gca,'XTick',[0:len]);
27 set(gca, 'XTickLabel', word);
28
29 % outputs a byte stream of the figure
30 % keep same background
31 set(gcf,'InvertHardCopy','off');
32 saveas(gcf, 'output', 'png');
33
34 end
```

Kodeutdrag 15: Matlab-funksjonen oneLineGraph

ThreeLineGraph100.m (se kodeeksempel 16 og fig. 39) Denne funksjonen tar i mot det aktuelle ordet, ”x1” og ”y1” - verdiene for gjestebrukeren, og ”mu” og ”sigma” - gjennomsnitt og standardavvik for en bestemt bruker i databasen og ”number” brukes til å lagre bilde av grafen. Ut fra disse verdiene genereres tre grafer i samme diagram som viser sammenhengen mellom gjestebrukerens målinger og maksimum- og minimumsverdien i databasen. Ordet plasseres i x-aksen for å vise hvilke knapper som gjelder for hver y-verdi.



Figur 39: Et eksempel på hvordan en graf ser ut.

4.3.2 Eksterne programmer og funksjoner

Tastaturgjenkjenningsmodulen i BioDemo består i likhet med fingeravtrykksmodulen av eksterne komponenter. I use case-diagrammet i kravspesifikasjonen (se fig)2) illustreres de eksterne programmene som en aktør: Keystroke Analysis. I virkeligheten består denne aktøren av C#- program tilpasset av gruppen, i tillegg til flere MATLAB-funksjoner. To av disse funksjonene tilhører NBL og kan ikke vises offentlig.

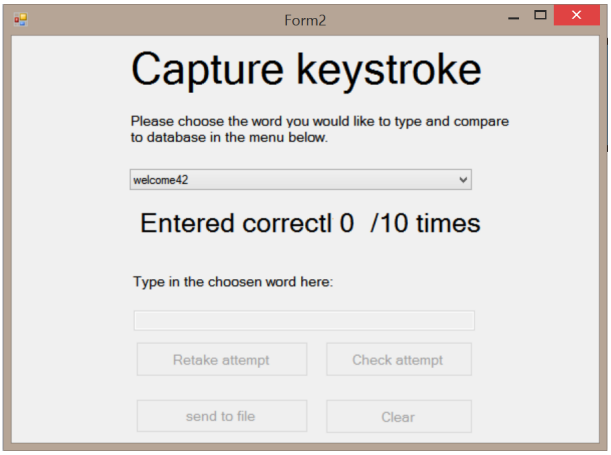
DemoKD

Vi benyttet et eksternt C# program som tar i mot og registrerer brukerens inntastinger, samt holder tellingen på hvor mange ganger ordet har blitt tastet inn, og sjekker om inntastingen ble godkjent. Selve funksjonaliteten ble benyttet i sin helhet, mens programmets utseende ble tilpasset i henhold til designprinsippene vi valgte å følge (se kapittel 3.6.2). Når brukeren velger ”Capture keystroke” fra ”Compare”-delen eller velger dette for sin bruker i ”Store”-delen, vil DemoKD kjøres. Fig. 40 viser hvordan dette vinduet ser

```
%Function that takes arguments to make a three line graph
%Word - the current word
%x1 - the values for x-axis
% y1 - values for the guest (blue line)
% mu og sigma - gjennomsnitt og standardavvik for en bestemt bruker
% i databasen
function [] = threeLineGraph(word, x1, y1, mu, sigma, number)

opengl software;
fig = figure;
set(fig,'Visible', 'off');
hold on;
% min
y2 = mu-sigma;

% max
y3 = mu + sigma;

minArr = [min(y1), min(y2), min(y3)];
maxArr = [max(y1), max(y2), max(y3)];

%Plot the values in a blue line with circles
plot(x1,y1,'-bo', x1, y2, '-r*', x1, y3, '-r*');

len=length(x1);

% adjust the axis after the max and min values of the plot
axis([-1 len min(minArr) max(maxArr) ]);
plot([-10 len], [0 0], '-k',[0 0], [-10 10], '-k');
%formatting the graph with squares, labels and title
grid on;
ylabel('Time');
xlabel('Word typed');
title('Captured keystroke values');
% Increment by one on x axis instead of five and increment to len
set(gca,'XTick',[0:len]);
set(gca,'XTickLabel',word);
% keep same background
set(gcf,'InvertHardCopy','off');
saveas(gcf, strcat('output', number), 'png');
end
```

Kodeutdrag 16: Matlab-funksjonen threeLineGraph

ut når det blir kjørt fra BioDemo.



Figur 40: Viser C# -programmet DemoKD som startes fra BioDemo

I BioDemo kalles DemoKD som en .exe -fil og kommer opp som et eget vindu. I kodeeksempelet 17 kan man først se at at .exe- filen kalles (linje 11 og 12). BioDemo vil så vente på at prosessen DemoKD skal bli ferdig (linje 15), og det opprettes en XML-fil som inntastingsdata lagres i (linje 17). Koden til DemoKD kan ikke vises i denne rapporten etter ønske fra oppdragsgiver.

```
1
2  private class CustomKeystrokeAction implements ActionListener {
3
4    @Override
5    public void actionPerformed(ActionEvent arg0) {
6      // remove data that where there before
7      UtillityClass.removeTempAndLiveKeystrokeData();
8      setButtonEnabledEquals(true);
9
10
11     ProcessBuilder builder =
12       new ProcessBuilder("cmd", "/c","Keystroke files/DemoKD.exe",
13         "1", "_live");
14     try {
15       Process process = builder.start();
16       process.waitFor();
17
18       File xmlFile = new File("Keystroke files/data/_live.xml");
19       Document doc = UtillityClass.getDOMSStructure(xmlFile);
```

Kodeutdrag 17: Utdrag som viser hvordan DemoKD kalles, og xml- dokument opprettes

MATLAB-funksjoner utviklet av NBL

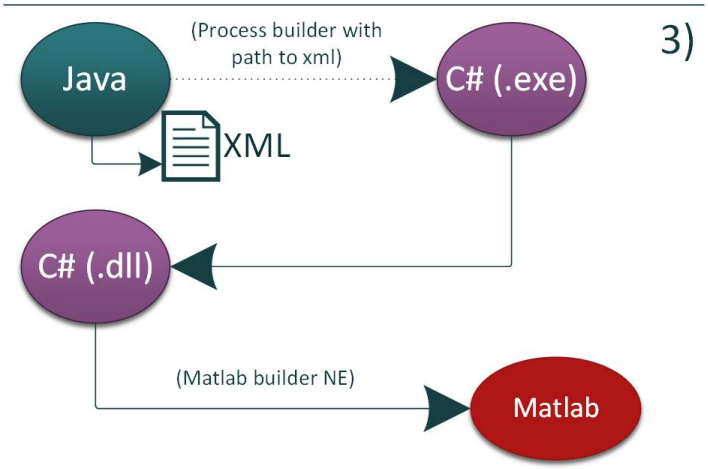
I tastaturgjenkjenningsmodulen ble det benyttet to funksjoner laget av NBL. Koden til disse kunne ikke vises i denne rapporten.

CreateTemplate.m: Denne funksjonen tar data fra brukerens inntasting som argumenter og genererer en template for denne brukeren. Templaten utgjør de tallene som skal sammenlignes mot brukeren inntastede ord i ”Compare”.

SMD.m: Dette er en algoritme (Scaled Manhattan Distance) som tar templaten til brukeren som ble generert i CreateTemplate- funksjonen og regner ut avstanden mellom gjestebrukerens verdier og databasen. Funksjonen returnerer en distanseverdi.

4.3.3 Utforfordringer med MATLAB

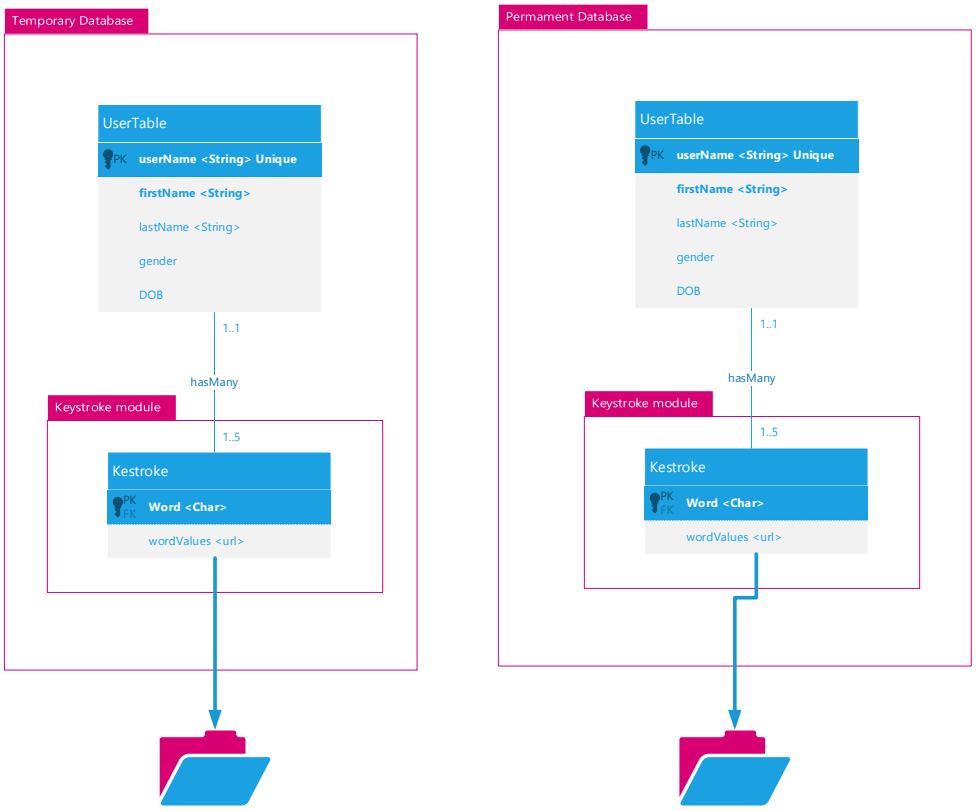
For å kunne benytte oss av MATLAB-funksjoner i prosjektet vårt var vi nødt til å implementere disse på en litt spesiell måte. Gruppen testet integrasjon mellom MATLAB og Java ganske tidlig i prosjektperioden på en av de private PC-ene, og i et 64-bits OS. I dette tilfellet fungerte koblingen mellom disse to bra. Da vi skulle gjøre det samme for NBL-PCen fikk vi problemer, og det lot seg ikke gjøres slik vi hadde testet på forhånd. I kapittel 6.1.4 omtaler vi disse utfordringene og den endelige løsninger i mer detalj. Fig 41 viser den endlige løsningen som fungerte.



Figur 41: Den endelige løsningen for implementasjon av MATLAB

4.3.4 Databasetilpassninger

For tastaturgjenkjenningsmodulen har vi i likhet med fingeravtrykkmodulen lagt opp til å kunne bruke både den midlertidige og permanente databasen, til tross for at det mest sannsynlig kun vil være bruk for den permanente databasen. Dette på grunn av at det kreves betraktelig mer tid fra en bruker å registere data knyttet til denne modulen, fordi en bruker må taste inn ett av de forhåndsbestemte ordene ti ganger før det lagres i databasen. Fig 42 viser det konseptuelle designet av databasen. Tabellen for tastaturgjenkjenning er ganske enkel, hvor det er kun to attributter; ordet det gjelder (”Word”) og url-adressen til XML-filen hvor tallverdiene ligger (”wordValues”). Dette er illustert med piler.



Figur 42: UML av databasen for tastaturgjenkjenningsmodulen

4.3.5 Filhåndtering knyttet til tastaturgjenkjenning

Når en bruker kjører DemoKD fra BioDemo blir det lagd en XML-fil hvor tasteverdiene knapp ned og opp blir lagret. Hvis en bruker er tilknyttet til en database, blir denne filen knyttes til riktig bruker i filen og det aktuelle ordet. Ifølge kravspesifikasjonen må brukerene i databasen skrive det valgte ordet ti ganger, før det kan sendes til databasen. Det betyr at XML-fila må inneholde data fra alle ti rundene. Kodeutdrag 18 viser et eksempel på en slik fil. Her er det ordet ”password” som er brukt, og ”number” forteller hvilken runde det er snakk om. Deretter følger det en liste med åtte ”<key_down>”-verdier, en for hver bokstav i ordet ”password”. Deretter følger ”<key_up>”-verdiene. I kodeutdraget er de resterende ni rundene og andre ord som brukeren eventuelt har data på blitt klippet ut.

Navngivingen av filene følger samme mønster som på fingeravtrykkmodulen, altså med brukernavn og hvilken database filen tilhører. For LiveScan er filen anonymisert og heter ”_Live.xml”. Ettersom en fil innholder alle dataene fra alle ordene er det ikke behov for mer informasjon i filnavnet. Et eksempel på ett filnavn er: BioDemoUser_perm.xml”.

```
<words>
  <word>
    <value>password</value>
    <data>
      <entry>
        <number>0</number>
        <key_down_list>
          <key_down>3,70757828547798</key_down>
          <key_down>3,89897864136473</key_down>
          <key_down>4,0989572564139</key_down>
          <key_down>4,25093149054728</key_down>
          <key_down>4,41893805690846</key_down>
          <key_down>4,82695268585893</key_down>
          <key_down>5,0269514398376</key_down>
          <key_down>5,210949624481</key_down>
        </key_down_list>
        <key_up_list>
          <key_up>3,78698743034728</key_up>
          <key_up>3,97895902206331</key_up>
          <key_up>4,15496657045472</key_up>
          <key_up>4,31495399328326</key_up>
          <key_up>4,52296002623203</key_up>
          <key_up>4,9149450522552</key_up>
          <key_up>5,09897179681039</key_up>
          <key_up>5,32294585425351</key_up>
        </key_up_list>
      </entry>
  <? Klippet ut kode  ?>
```

Kodeutdrag 18: Eksempel på en XML-fil til en bruker i databasen

En av de viktigste egenskapene løsningen for å lagre disse dataene måtte ha, var at både Java og C# måtte kunne hente og lagre dataene. Lagring direkte i Derby databasene ville medføre utfordringer med kommunikasjon mellom C# da Derby er utviklet i Java og benytter databasedriveren JDBC [25]. I utgangspunktet kommuniserer ikke C#

med JDBC, så for å løse det må det implementeres en bindeledd mellom disse.

Av denne grunn anså gruppen at det var en bedre løsning å benytte en av to muligheter; lagre dette i en ren tekstfil eller lagre den i en XML-fil. Begge løsningene ville kunne kommunisere med både Java og C# uten problemer, og eventuelle andre språk hvis dette skulle bli et behov. Ved en ren tekstfil måtte de forskjellige dataene skilles fra hverandre ved hjelp av ett tegn (vanligvis mellomrom), og når dataene ble hentet fra filen måtte man skille verdiene når mellomrom dukket opp. Dette var ikke noe problem for en gruppe som kjenner hvordan strukturen i filen er, og vet hvordan data lagres. Men for en som ikke kjenner strukturen kan det være krevende å ”lese” innholdet i tekstfilen og skille de forskjellige dataene fra hverandre. For XML gjelder ikke dette. Det er ikke behov for å ha ett bestemt tegn som skal skille dataene fra hverandre og det er lettere for en utenforstående å lese innholdet så lenge typene er navngitt logisk. Ved hjelp av en parser i Java og i C# vil programmet være i stand til å hente ut dataene i XML-filen.

5 Testing og kvalitetssikring

Det har gjennom prosjekttiden blitt gjennomført tester og kvalitetsikringer på flere områder for å sikre at BioDemo fungerer i henhold til kravene som er satt i kravspesifikasjonen, og for å identifisere feil.

5.1 Testing av BioDemo

Gruppen startet med å vurdere muligheten for å bruke test først-utvikling (TFU). Denne prosessen går ut på å lage tester før funksjonene implementeres, for så å lage kode som tilfredsstiller både funksjonelle kravene og sikrer at testen kan kjøre vellykket. I TFU benytter man verktøy for automatisk regresjonstester[1]. Et eksempel på ett sånt verktøy er JUnit i Java. En av hovedfordelene med TFU-prosessen er at man oppdager feil tidlig i utviklingen av programmet og at man lettere kan identifisere problemer knyttet til for eksempel refaktorering av kildekode [13], [26]. Hovedgrunnen til at vi ikke kunne gå fullt inn for TFU er at de automatiserte testene ikke kan fange opp hvorvidt brukeren blir presentert for de riktige grafiske elementene til enhver tid. Vi bestemet derfor å kun bruke automatiserte tester der vi anser dem som hensiktsmessige.

De ulike metodene vi benytter for å teste Biodemo er:

- Automatiserte regresjonstester (JUnit)
- Gray box-testing
- Review
- Statisk kodeanalyse

5.1.1 Automatiserte regresjonstester

Det var flere grafiske komponenter som programmet skulle benytte seg av for å vise egendefinerte objekter. Disse komponentene skulle holde på objektene dynamisk under kjøring av BioDemo, ved at de legges inn når brukeren trykker på en knapp og fjernes når vedkommende trykker på en annen. For å løse dette, fikk hver av disse komponentene sin egendefinerte datamodell som holdt på objektene. Her ble automatiserte tester brukt, fordi vi mente at dette var en god måte å sjekke om objektene ble korrekt lagt inn og fjernet fra modellene.

5.1.2 Gray Box-testing

Funksjonalitet som krever interaksjon med brukeren, ble testet ved hjelp av Gray Box-testing. Gray Box-testing er en kombinasjon av White Box- og Black Box-testing. I White Box-testing kjenner man til den interne strukturen i kildekoden, mens med Black Box har man ingen slik kunnskap. I Gray Box har man delvis kjennskap til strukturen [27]. Når gruppen testet koden, hadde ikke gruppemedlemmene 100% kunnskap om all kode, da vi valgte å utnytte muligheter ved å ha tre utviklere ville det være områder som ikke alle hadde kjennskap til. På denne måten startet testingen som Gray Box-testing. Etter hvert som gruppemedlemmen fikk mer kjennskap til hverandres kode gjennom kode-reviews,

[1] Sjekker at den samme testen kan kjøres uten feil

gikk testingen mer i retning av White Box.

I vedlegg I vises en samling av tabeller som dokumenterer hvordan vi gjennomførte denne prosessen, samt testenes resultater.

Den første tabellen (se fig. I.7) inneholder en oversikt over testene som ble utført av gruppen. I fig. 43 er det et kort utdrag fra denne tabellen. Resten kan sees i vedlegget som det ble referert til ovenfor.

Test nummer	user story/ user task/ operasjonelle krav	filer/klasser innvolvert	Funksjoner innvolvert	Beskrivelse	Test data
1		/img/* /img/ikonCC/* no/hig/main/start.java no/hig/main/HomeMenu.java	Start : InitClass(), setupMenues(), setupStartWindow(), setScreenSizeAndVisible(), updateWIndowUtility(), HomeMenu: ALT	Test for å sjekke om vi får: 1. Vist et hovedvindu 2. Vist meny 3. Vist banner med hig logo, beskjed (sentrert), og Nislab logo 4. Vist forside med navigering til ulike moduler 5. SetLookAndFeel har blitt satt til nimbus 6. Fargevalg er satt riktig 7. Ikoner er satt 8. Størrelsen på programmet skal ta hensyn til windows oppgavelinje	1.Test med ulike pcer med forskjellige skjerm størrelser. 2. endre orienteringen på oppgavelinja i windows
2		C:/Fingerprints/* (Reg no: 03699) no/hig/main/Start.java no/hig/main/UtilityClass.java database/DropTablesForGivenDatabase.java no/hig/FingerprintModule/FPCompTabWin.java derby.jar	Start : setCostomCloseOperation(), main(...), UtilityClass : removeLiveScanImages(...), deleteTempFiles(...), listFileRenamet(...), getFileExtension(...), DropTablesForGivenDatabase: ALT FPCompTabWin : setupButtonActionListeners() - clear : JButton	Sletting av middeltidige data dvs : 1. Bilder fra livescan (etter vi har trykket clear) 2. Bilder fra livescan ved oppstart av programmet 3. Bilder fra livescan ved normal avsluttning av programmet. 4. Bilder fra temp brukere ved oppstart av programmet 5. Bilder fra temp brukere ved normal avsluttning av programmet. 6. Data i temp databasen ved oppstart 7. Data i temp databasen ved normal avsluttning av programmet.	1. Klikk på clear knappen : klikk event 2.avslutt program ved å krysse det ut 3. avslutt programmet uten å krysse ut (CTRL + ALT+ DELETE) 4. Ha bilder lagret på C:/Fingerprints i ulike mapper og ulike filformater 5. ha data i databasen 6. ikke ha data i databasen

Figur 43: Utdrag fra oversikt over gjennomførte tester

Hver test fikk et unikt test-nummer som ble brukt for å koble flere rader med resultatdata til en og samme test. I kolonnen ”user story/user task/operasjonelle krav” tok vi med alle funksjonelle og ikke- funksjonelle krav som denne testen dekker. I kolonnene ”filer/klasser involvert” og ”funksjoner involvert” forklares hvilke deler av kildekoden og ressurser som ble berørt av denne testen. I ”beskrivelse”-kolonnen beskrives forventet (eller ønsket) resultat, og i ”test data”-kolonnen nevnes alle inndata, hendelser og situasjoner som ble benyttet for å teste koden.

I neste tabell (se vedlegg I.8) presenteres resultatene av testingen. I fig. 44 kommer vises et utdrag fra denne tabellen, resten kan sees i vedlegget referert til overfor.

Resultat for	Resultat beskrivelse	Gjennomført	Handling
1	3. Beskjeden er ikke sentrert 4. Ikke på plass 6. Fargevalg ikke bestemt enda 7. Ikoner ikke valgt enda 8. Ikke tilpasset ulike maskiner	04.02.2015	3. Bruk html formattering for å sentrere 4. Gjenta test når dette er på plass 6. Velg hvilke farger skal brukes for hva og sett disse som conster 7. Finn Creative commons som kan benyttes 8. Finn løsning hvor man kan dynamisk finne ut hvor oppgave linja er orientert, hvor stor den er osv og lag fullscreen ut ifra dette
1	4. Ikke på plass 6. ST	04.02.2015	

Figur 44: Utdrag fra oversikt over resultatene av testene

”Resultat for”-kolonnen peker på hvilken test dette er resultat for, ”Resultat beskrivelse” beskriver eventuelle avvik fra det forventede resultatet, ”Gjennomført”-kolonnen beskriver datoen testen ble utført og i ”Handling” beskrives tiltak som ble gjort for å fikse avvik.

5.1.3 Review

For å avklare feil tidlig i utviklingsprosessen hadde gruppen flere runder med gjennomganger for å se om den konseptuelle modellen for databasene tilfredsstilte kravene satt til systemet. Vi hadde også flere runder der vi så på sammenhengen mellom våre designskisser og de funksjonelle kravene til systemet. Vi benyttet også mye designskisser for å kommunisere kravene til systemet for oppdragsgiver. Det var derfor viktig å avklare misforståelser som kunne ha oppstått når dette skulle oversettes til user stories. Vi brukte gjennomgang av hverandres kode for å øke vår forståelse, forbedre kvaliteten, finne logiske feil, og rette opp i eventuelle mangler på f.eks. dokumentasjonen.

5.1.4 Statisk kodeanalyse

Ved statisk kodeanalyse går et program gjennom kildekoden til et system, uten å måtte kjøre den. Dette benyttet vi blant annet for å finne vanlige kodefeil, unngå overhead i forbindelse med manuell gjennomgang av hver linje av kildekoden, og for å finne tilfeller der vi ikke fulgte gode kodestandardprinsipper. [28]

I vedlegg J vises resultatene fra kjøring av det statisk kildeanalyseprogrammet FindBugs på BioDemo [29]. Feilene er delt inn i to kategorier ”Troubling” og ”of concern”. Disse kategoriene er igjen delt opp i ulike grader som representerer hvor stor sannsynlig det var en bug (Ulike grader av dette blir nok presentert, med tanke på at statisk kode analyse programvare har en tendens til å ha en høy grad av falske positiver). Vi rettet opp i tre typer feil som vi så var aktuelle å gjøre noe med:

- *“value is null and is guaranteed to be dereferenced on exception path “*:
  I vårt tilfelle betydde dette at siden vi lukker en forbindelse til databasen i en finally clause og en SQL-exception ble kastet før finally ble kjørt, kunne forbindelsen selv være null. Det måtte derfor legges til en sjekk på at forbindelsen ikke er null, før den stenges. Se kodeutdrag 19 for hvordan vi valgte å løste dette.
- *“Dead store to local variable “*: Representerer ubrukte variabler. Disse variablene ble fjernet.
- *“Code contains hard coded reference to a absolute pathname “* :
  Vi hadde hardkodet noen stier til foldere på NBL-PCen. Dette hadde vi laget conster for, som kan endres i koden om det blir aktuelt å kjøre koden på en annen maskin enn NBL-PCen i fremtiden.

```
1 } finally {
2   if(prepS != null)
3     prepS.close();       //Close statement
4   if(con != null)
5     con.close();       //Close database connection
6 }
```

Kodeutdrag 19: Kode for å sikre at det ikke kommer en null pointer exception når en databaseforbindelse stenges

5.2 Testing av GUI

Testing av GUI var viktig for å sikre at programmet faktisk oppfylte kravet til brukervennlighet (se kravspesifikasjon i kapittel 2) Denne testingen har foregått på forskjellige stadier i utviklingsprosessen, og disse kan det leses om under vedlegg D. Ettersom BioDemo skal benyttes av personer med forskjellig bakgrunn, anså vi det som viktig å teste skissene på personer med forskjellig bakgrunn, både når det gjaldt alder, kjønn og bakgrunn.

5.2.1 Brukertesting av skisser

Etter anbefaling av Ole Wattne (Se møtereferat C.3)) ble den første brukertesten utført på ark. Her benyttet vi nyeste versjon av designskissene og nummererte disse i henhold til programflyten. Alle disse designskissene kan ses i vedlegg K.

Vi valgte å benytte testmetoden Concurrent think aloud - noe som vil si at testansvarlig går gjennom programmets naturlige flyt ved hjelp av ark som er nummererte. Testansvarlig skulle kort forklare hva som skjedde på det aktuelle arket, og påtok seg rollen som den personen fra NBL som skal delta på demonstrasjonen i praksis. Testansvarlig skal så langt det er mulig være nøytral og ikke uttrykke sine egne meninger eller oppfatninger underveis i testen. [30]

Vi endte opp med fire testpersoner: to med IT-bakgrunn, to med annen bakgrunn, to menn og to kvinner, og disse var i alderen 16-56. Ingen av personene hadde noen spesiell biometrikunnskap fra før, utover å ha hørt lite om det.

Etter testen valgte vi å benytte retrospective probing, noe som vil si at testansvarlig stiller noen standardiserte spørsmål til testperson i tillegg til å svare på eventuelle spørsmål fra testpersonen selv. Dette benyttes for å oppsummere testen. Spørsmål som ble stilt her var:

- Se for deg at du brukte det faktiske programmet, forstår du nå hvordan fingeravtrykkprosessen foregår?
- Synes du at du har mer forståelse for biometri nå etter at du har gjennomgått dette programmet?

Under vises et referat fra en av brukertestene, der vi kun noterte de tingene som dukket opp underveis. De andre tre kan sees i vedlegg D.

Brukertest 4:
Mann, 25 år, IT-student, kjenner litt til biometri. Mest info og litt over gjennomsnittet.

2. Synes det er naturlig å trykke på registrer før man tar compare. Skal jo først registrere seg for å kunne sammenligne med databasen.
4. Vil gjerne ha mer informasjon om fingeravtrykket sitt når man trykker på analysis. Litt lite info. Hva betyr Good?
6. Trenger ikke å ha «streker» mellom treffene. Så lenge det er de røde punktene så kan man se hva som er unikt og hva som er ikke unikt.
7. autosjekk på username
8. Litt forvirring mellom scan og klikke på finger. Dropp « selected again»
11. Mer informasjon, samme som 4.
Generelt: Lært mer om prosessen, men ikke så mye om informasjon.

Testene viste at vi i stor grad hadde klart å utvikle et design som var naturlig og intuitivt for brukerene. De punktene som kom fram, var enkelte formuleringer som kunne vært klarere, samt nødvendighet av å understreke hvilke knapper som er aktive og inaktive. Dette var noe vi tok med oss i neste utkast og i selve implementasjonen.

5.3 Risikovurdering

Vi hadde to runder med risikovurderinger underveis i løpet av prosjekttiden. Vi opparbeidet oss en liste over alle risikoene vi så på de tidspunktene vurderingene ble gjennomført. For hver av risikoene ble det vurdert konsekvens og sannsynlighet på en skala fra liten, middels og stor, og sjelden, og for sannsynlighet: kan skje og svært vanlig på sannsynlighet. For hver runde ble alle risikoene vurdert på nytt, hvor de fra tidligere runder ble vurdert etter tiltak der det ble gjort. I de tilfellene risikoene økte fra forrige runde, ble dette kommentert eksplisitt og vurdering om tiltak ble gjort i gruppa. Hele risikotabellen etter siste versjon (runde to) er lagt med vedlegg E.

5.3.1 Første runde

Første runde for risikovurdering ble gjennomført i januar, før sprintene og programutviklingen begynte. På det tidspunktet var det krevende å se noen risikoer knyttet til selve programutviklingen, men vi så flere risikoer knyttet til selve prosjektarbeidet og det grunnleggende. Derfor er de første risikoene preget av dette. I risikotabell E er disse risikoene markert med en i første tall i ”Nr”-kolonnen. Merk at vuderingen av sannsynlighet og kosekvens er justert etter runde to og kan ha en annen vurdering enn ved runde en. De største risikoene i denne runden vises i fig. 45.

Nr	Risiko	Sannsynlighet	Konsekvens	Kommentar	Tiltak
1.1	Tap av prosjektdokumenter	Kan skje	Stor	Dette kan skje pga menneskelig eller tekniske feil.	Alle medlemmer av gruppa må bruke versjonskontroll aktivt + lagre viktige dokumenter på felles server.
1.3	Får ikke tilgang til programmer og/databaser som er nødvendig	Kan skje	Stor	Pga oppgavens art er det helt nødvendig vi får tilgang og støtte rundt programmer og databaser fra NBL.	Ha jevnlige faste møter med Patrick Bours og NBL, være klare på hva vi trenger fra dem.
1.9	Mangel på kunnskaper	Svært vanlig	Middels	Biometri, MatLab, GUI, IO osv er fagfelt som vi er ganske uerfarne i.	Forberede oss med å finne og lære oss disse nye områdene som vi ser vil bli aktuelle å kunne. Fordele spesialfelt mellom medl for å være mer effektive.

Figur 45: Utdrag av de høyeste risikoene i runde 1

Alle de tre risikoene som kom opp hadde vi ganske gode tiltak på, og vi implementerte dem alle. For 1.1 valgte vi å benytte oss av versjonskontroll i form av BitBucket som resulterte i at vi hadde tre lokale kopier av arbeidsdokumenter, kodefiler og rapporten på våre egne private PCer, i tillegg til lagring hos Bitbucke sine servere med backup-lagring med hjelp av Raid6. For 1.3 og 1.9 var det snakk om å planlegge og få oversikt over hva som trenges og så være klare på hva vi i gruppen måtte gjøre og hva vi måtte få hjelp til fra NBL.

5.3.2 Andre runde

Andre runde for risikovurdering ble gjennomført i mars, godt inni programutviklingen. På dette tidspunktet var flere risikoer knyttet til BioDemo og utviklingen klare. Vi tok også en ny vurdering av risikoene som ble identifisert i runde en. De fleste av risikoene fra runde en ble redusert, mest fordi vi iverksatte flere av tiltakene som ble foreslått da. Men en risiko økte, og sammen med de nye riskoene (de som starter på 2 i "Nr"-kolonnen) viser i fig. 46.

Nr	Risiko	Sannsynlighet	Konsekvens	Kommentar	Tiltak
1.8	For høye arbeids/prosjektkrav, Tidsfrister som overskrides	Svært vanlig	Middels	Økning. Pga forsinkelser med PC i begynnelsen så har vi blitt hengende litt etter. Vi er nå straks halvveis og jobber fortsatt med fingeravtrykkmodulen. Skal begynne på neste så fort som mulig	Tilpasse oss etter behov, gjøre vårt beste for å levere og hvis dette ikke er nok: få dette med i rapporten.
2.11	Sikkerhetssvakheter med WinXP	Svært vanlig	Stor	WinXP er en utfaset OS fra Microsoft, og er veldig sårbar mot trusler, spesielt fra internett	IKKE koble til internett. Bruke minnepinner for å flytte over filer ol. Patrick er positiv til at vi tilpasser til nyere versjoner også.
2.15	Database ikke blir slettet som det skal	Kan skje	Stor	Det kan skje feil i kjøring av program / kodefeil som fører til at dne imidlertidige databasen ikke blir slettet som den skal.	Ha en "dobbelsjekk" ved oppstart av programmet som sletter hvis databasen finnes.
2.16	Gjester lagrer i den permamente databasen	Kan skje	Stor	Når en bruker skal registere seg, så må h*n velge mellom den midlertidigie og den permamente databasen. Hvis en gjest klikker feil, og lagrer seg selv i den permamente databasen, så er det vanskelig å fjerne denne personen igjen.	For å fjerne brukern, så må man hardkode en SQL-spørring som sletter denne spesefikke personen. Det er ikke laget en funksjon for dette, og det er kun en funksjon som sletter hele tabellen på plass. Ved en slik situasjon så må eier av BioDemo vurdere om tabellene skal slettes eller om personen skal være i databasen. Merk at det er snakk om flere steg av godkjenning før fingeravtrykk blir lagret.

Figur 46: Utdrag av de høyeste risikoene i runde 2

I denne runden ble risiko nr 1.8 vurdert i forhold til i runde 1. Dette fordi vi så at vi ikke hadde så bra med tid som vi hadde regnet med. Vi brukte mer tid på ting som vi ikke hadde regnet med skulle ta så lang tid, og fingeravtrykkmodulen ble fort mer omfattende enn det vi i utgangspunketet hadde sett for oss. Dette i tillegg til at NBL-PCen ble over tre uker forsinket og, ikke ble tilgjengelig for oss før på slutten av februar. På grunn av linsenser knyttet til fingeravtrykkskannerprogrammet som ikke kunne overføres til noen av våre private PCer, var vi ganske begrenset i denne perioden. Av de andre nye risikoene som ble identifisert er 2.11 spesiell. NBL-PCen kjører Windows XP som operativssystem. Dette er i seg selv en stor risiko fordi Microsoft har avsluttet støtten for denne versjonen [7]. For å sikre PCen mot dette, ble det svært viktig å ikke koble den til internett.

Avvik i tiltak

Det var behov for å verifisere sertifikatet til MATLAB, og dette var det ingen annen løsning enn å koble til internett. Vi valgte å la oppdragsgiver ta beslutningen om dette skulles gjøres eller ikke. Alternativet var å ikke utvikle tastaturgjenkjenningsmodulen på

NBL-PCen. Oppdragsgiver tok besluttningen om å la NBL-PCen kobles til internett mens MATLAB ble lastet ned og verifisert. Gruppen opplyste klart for oppdragsgiver risikoen for dette og hva alternativene var. NBL-PCen ble koblet fra så fort dette var fullført.

6 Realisering og implementasjon

6.1 Utviklingsmiljø

6.1.1 NBL-PC

I forbindelse med prosjektarbeidet fikk gruppen tildelt en PC fra NBL som BioDemo skal kjøres på under demostrasjoner, og som er tilknyttet de lisenser som benyttes i BioDemo.

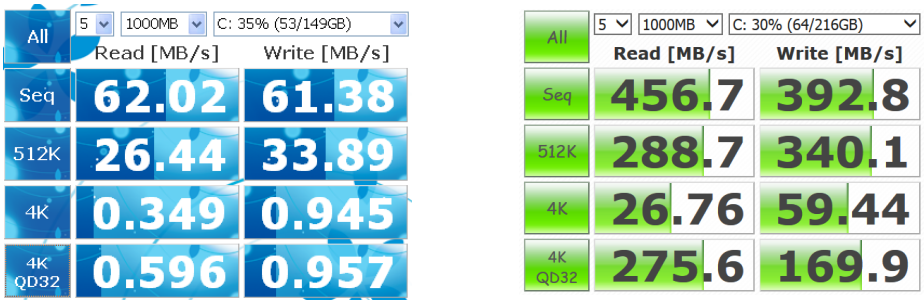
Systemspesifikasjoner for denne NBL-PCen er som følger:

- Operativsystem: Windows XP Professional Version 2002, Service Pack 3
- Prosessor: Intel Core 2 Quad Q6600 - 2,4 GHz
- RAM: 3,25 GB
- Disk: HDD. WDC WD1600AAJS-75PSA0 148 GB

Den første utfordringen vi så, var operativsystemet. Microsoft tilbyr ikke lenger support av Windows XP, og dette medfører at Windows XP blir fem ganger så sårbar for sikkerhetstrusler, et tall som bare vil fortsette å stige [7]. Oppdragsgiver var klar over problemstillingen, og for å hindre at NBL- PCen fra å bli infisert ble det bestemt at den ikke skulle kobles til nettet. Dette innebar at programvare som skulle installeres samt programkode til BioDemo måtte overføres fra våre egne maskiner til NBL-PCen med minnepenn. Samtidig kunne vi ikke utføre versjonskontroll med Bitbucket på samme måte som med våre egne PCer. Vi løste utfordringene ved å utføre en manuell versjonskontroll på NBL-PCen ved å flytte over utviklingsmiljøet fra Bitbucket i form av egne Javaprosjekter. Ved installasjon på NBL-PCen opprettet vi et nytt prosjekt fremfor å overskrive det gamle. På denne måten fikk vi flere prosjektversjoner, og muligheten for å gå tilbake til tidligere versjoner hvis det skulle bli behov for det. Ved endring av enkeltfiler fra Bitbucket, brukte vi Eclipses innebygde sammenligingsverktøy for å finne forskjellene og kopierte disse manuelt.

En annen utfordring var disken. Dette var noe vi allerede merket ved begynnelsen av utviklingsperioden av prosjektet, da disken på NBL-PCen måtte overskrives før vi kunne få tilgang til den. Dette tok over tre uker, noe som medførte forsinkelser i utviklingen av BioDemo. På bakgrunn av dette utførte vi målinger av lese - og skrivehastigheten på disken. Her benyttet vi programmet CrystalDiskMark [31]. Figurene under 47 viser først NBL-PCens lese/skrive-hastighet i fig 47a. I tillegg valgte vi å gjennomføre identisk test på en av våre private PCer hvor mesteparten av utviklingen skjedde for å vise hvor stor forskjell det faktisk er. Fig 47b viser lese/skrive-hastigheten på den nyeste av våre PCer, som består av en SSD-disk.

Målingene viste store forskjeller på lese/skrivehastighet, noe som innebar at å legge til brukere, hente ut brukere fra databasen, og ikke minst analysere biometriske kjennetegn ville ta mye lengre tid på NBL-PCen. På bakgrunn av dette, var det viktig at alle programmets funksjoner ble testet på NBL-PCen for å forsikre oss om det gav akseptabel responstid. I tillegg tok vi flere hensyn under programutviklingen. Både av sikkerhets - og ytelsesgrunner valgte vi å benytte PreparedStatemets, som det kan leses mer om i



(a) NBL- PC

(b) Nyere PC med SSD disk

Figur 47: Måling av lese/skrivehastighet på 2 forskjellige PCer

kapittel 4.1.6. I tillegg valgte vi å benytte memory-mapped I/O, se mer i kapittel 4.2.3 om dette.

6.1.2 Programmeringsspråk

Oppdragsgiver hadde ingen preferanser når det kom til programmeringsspråk, og derfor stod gruppen relativt fritt til å velge programmeringsspråk etter eget ønske. Fra før hadde alle gruppemedlemmene erfaring med å programmere i Java og C++, og en av gruppemedlemmene hadde i tillegg erfaring med C# 1.4.

For å finne det språket som passet oss best, satte vi opp en liste med våre behov:

- Vi ønsket et språk som var type safe. Dette vil i korte trekk si et språk som gir feilmeldinger når det forsøkes å utføres operasjoner på verdier som ikke støttes av operasjonene. Programmet vil f.eks. hindre at "a"+7 blir utført når den egentlige hensikten er at to inter skal adderes. [32]. Det viktigste for oss var at programmet skulle returnere feilmlding når en array går utenfor grensene, noe som blant annet hindrer at buffer overflow fra å forkomme.
- Vi måtte forholde oss til eksterne programmer som var skrevet i C# og MATLAB. Språket vi valgte måtte derfor kunne kommunisere med disse språkene.
- Språket måtte tilby et grafisk rammeverk som kunne realisere våre designskisser.
- Med begrenset tid, måtte vi velge et språk som ikke medførte unødvendig lang utviklingstid.

BioDemo skulle i hovedsak benytte seg av eksterne programmer som registrerte biometriske kjennetegn, utførte analyser og sammenligninger. En viktig del av prosjektet var å utvikle et grafisk grensesnitt. Vurderingen gikk derfor ut på å bestemme hvilket språk vi skulle bruke for å lage det grafiske grensesnittet og primært kommunisere med de komponenten som var skrevet i C# og MATLAB.

Med hensyn til utviklingstid, ble vi enige om å benytte et språk vi hadde kunnskaper om fra før, og valget stod derfor mellom C++, Java og C#. Alle disse språkene kunne også kommunisere med C# og MATLAB. Java og C# er type safe, men det er derimot ikke C++ og dette språket kunne derfor utelukkes.

Java tilbyr GUI-rammeverket Swing - som gruppemedlemmene hadde erfaring med fra

faget programvareutvikling. Swing er også enkelt å tilpasse med et bredt utvalg av look and feels, som gav mange muligheter for å manipulere utseendet på de fleste GUI-komponentene.

Basert på disse vurderingene, falt valget på Java, som oppfylte alle våre krav fra listen ovenfor.

6.1.3 Diverse utfordringer med flerspråks programmer (våre erfaringer)

En av utfordringene i forbindelse med integrasjon av C# -kode (.dll) med BioDemo, var at språkene oppfører seg noe forskjellig. En av disse forskjellene er hvordan lagring i byte memory foregår. En byte i Java er signed(fra -127 til 127) [33], mens i C# er en byte unsigned (fra 0 til 255) [34]. Dette medførte ingen kompilasjonsfeil, men problemet oppstod når BioDemo forsøkte å lese en C#- byte med verdi 255 i en Java byte. Verdien ville her ikke få plass og dette resulterte i at programmet representerte 255 som -1. Siden BioDemo er avhengig av denne byte-verdien for å ta beslutninger, medførte denne lille forskjellen mellom språkene en uønsket programflyt. For å finne denne feilen uten å ha stålkontroll på hvordan begge språkene fungerte under panseret, måtte det debugges på både Java- og C#- siden. Dette er en av grunnene til at integrasjon mellom utviklingsspråk kan være utfordrende. [35], [36]

Fra BioDemo benyttet vi også ProcessBuilder for å starte kjørbare filer (.exe). ProccessBuilder kan jobbe med relative stier. Det er flere måter å skrive disse stiene på. En filvariabel i Java kan deklareres sånn: File f = new File(" /folder/resurs.png "). Dette hadde ikke vært en gyldig sti for ProcessBuilder. ProcessBuilder hadde oppfattet dette som "C:/folder/ressurs.png ". En akseptabel relativ sti for processBuilder vil være "./folder/ressurs.png".

Det var også en utfordring å benytte Unicode characters mellom C# og MATLAB. MATLAB har utfordringer med å skrive tegn fra dette tegnsettet, men det er mulig å bruke Java- kode fra MATLAB som kan gjøre denne jobben.

I ulike land har man forskjellige måter å skrive inn en dato, tallverdier og mer. For å sikre at man får det ønskede formatet tilbyr ofte et programmeringsspråk en lokal API. Om programmet benytter seg av dette APIet, tar man ofte hensyn til standardspråket til maskinen applikasjonen kjører på. Problemet vi møtte på var at Eclipse (Java utviklingsverktøyet vårt) og Visual Studio (C# utviklingsverktøyet vårt) gav ulik format på desimaltall. I Eclipse ble desimaltall representert med punktum (.), mens i Visual Studio ble komma (,) brukt. Dette ble et problem når vi skulle lese fra en fil som den andre programeringsplattformen hadde skrevet. Vi løste dette med enkel string-håndtering. [37] (Se kodeutdrag 20).

```
1 //  fra  Java
2 doubleAsString.replace(',','.')
3
4 // fra C Sharp
5 doubleAsString.replace('.',',')
```

Kodeutdrag 20: String håndtering av locale dependent verdier

6.1.4 MATLAB-integrasjon

MATLAB er et matematisk høy-nivå programmeringsspråk som benyttes mye hos NBL, og av denne grunn ble det nødvendig å tilpasse BioDemo slik at MATLAB-filer kunne integreres. Jobben vi gjorde for å finne løsninger for integrasjon av MATLAB var viktig for å sikre at BioDemo skal bli et langlivet produkt.

Løsninger

Vi vurderte løsninger basert på hvordvidt de var godt dokumentert og om de fremdeles var under utvikling. De to løsningene som var aktuelle for oss, var MATLAB Builder JA (MBJA) og MATLAB Builder NE (MBNE) [38] [39]. Ved å kompilere en MATLAB-funksjon gjennom MBJA, blir det produsert en JAR-fil som fungerer som en wrapper for å kommunisere mellom MATLAB og java. MBNE produserer wrapper koden i form av en DLL og dette gjør at MATLAB og C# kan kommunisere.

Utfordringer med implementasjonen av MATLAB

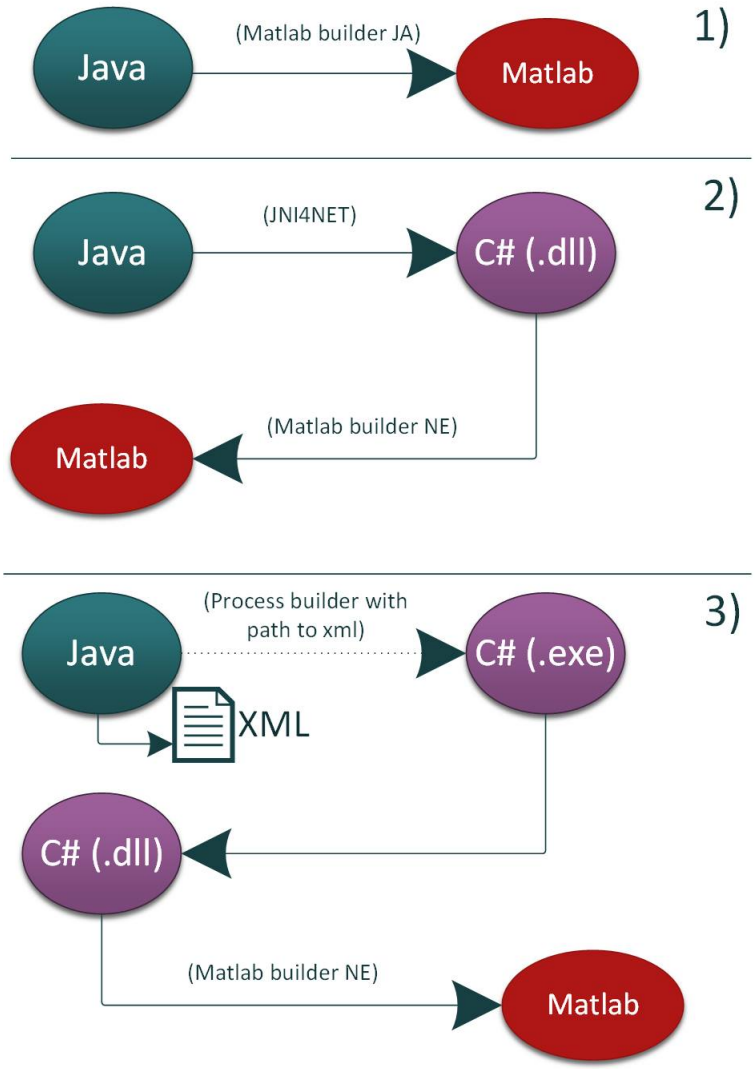
Ved implementasjon var det flere utfordringer som dukket opp og flere der vi ikke fant årsaken til problemet. Dette gjorde implementasjonen av arbeidet krevende og vi brukte lengre tid enn planlagt på dette. Noen av utfordringene vi møtte var:

- Det var utfordrende å rekonstuere situasjonen der feilsituasjonene oppstår.
  - To Java-prosjekter som var konfiguert med samme JRE, operte under samme maskin, jobbet på samme wrapper og koden fungerte på den ene men ikke den andre.
  -vUtvikleren har liten kontroll hvilken return-type wrapper-koden har.
- Det var utfordrende å avklare hvor eller under hvilket programmeringsspråk som førte til at en exception ble kastet under kjøring.
- Vi oppdaget en bug på NBL-PCen ved kjøring av MATLAB. I utgangspunktet skal MATLAB starte OpenGL (et program for å vise frem figurer). Denne funksjonaliteten funker fint på våre private PCer, men ikke for NBL-PCen, og vi måtte eksplisitt si at OpenGL skulle starte ved hjelp av OpenGL programvare. Her var det flere hensyn vi måtte passe på:
  - Miljøvariablene måtte være riktig satt
  - Java-versjonen måtte støtte den versjoen av MATLAB vi jobbet med
  - Det måtte være samme arkitektur og samme bit-versjon av MATLAB som NBL-PCen.
  - NBL-PCen måtte ha samme versjon av MATLAB-kompilatoren som wrapper-koden ble kompilert med.
  - JAR-filer er i utgangspunktet uavhenige av arkitekturen, men wrapper-koden er avhenig av arkitekturen.

Denne tabellen viser hvordan vårt oppsett er på NBL-PCen.

Java	C#	MATLAB	NBL-PC
32 bit JRE 8 Eclipse	Visual C# Studio 2010 Express (2005 versjonen gir oss ikke mulighet til å styre target Framework)	32-bit MATLAB R2014B	JAVA_HOME peker på 32 bit JDK 1.7.0_75, 32-bit arkitektur, Windows XP 32-bit, NET Framework 4.0

Fig. 48 viser de forskjellige løsningene vi prøvde, før vi til slutt endte med den siste (3). Alternativ nr 1 er det vi fikk til på våre private PCer i "dummyprosjekter", og antok ville funke på BioDemo-prosjektet og på NBL-PCen. Denne løsningen ville ikke fungere på NBL-PCen, til tross for at MBJA var eksplisitt lagd for å løse dette. [38].



Figur 48: Viser de tre alternativene gruppen så på for integrasjon av Matlab

6.2 Vurdering av sikkerhetsmekanismer

Personopplysninger defineres i loven som «opplysninger og vurderinger som kan knyttes til en enkeltperson». Vår database vil inneholde biometriske kjennetegn, som havner inn under kategorien «fødselsnummer og andre entydige identifikasjonsmidler». Dette er sensitive personopplysninger, og det er derfor viktig å sikre at denne informasjonen ikke kommer på avveie, og kun benyttes til det formål eieren har gitt godkjenning for. Vi brukte derfor tid på å gå gjennom aktuelle sikkerhetsmekanismer, og avgjøre behovet for disse. I tillegg var det ytelseshensyn som måtte tas i forhold til NBL-PCen. For BioDemo valgte vi å benytte Derby-databaser. Derby tilbyr forskjellige former for sikkerhetsmekanismer. Vi har både tatt for oss de mest aktuelle av disse og andre sikkerhetsmekanismer.

Autentisering og autorisering:
Både brukerautentisering og brukerautorisering støttes av Derby. Ved brukerautentisering verifiserer Derby brukeren før vedkommende får tilgang til databasen. Brukerautentisering er ikke aktivert ved default, men man må først definere en repository med de brukerene som skal ha tilgang, og velge om denne kun skal være gjeldene for databasen, eller for hele systemet. Brukerautorisasjon var også et alternativ vi så på, og her kan man spesifisere hvilke brukere som har tilgang på hva. I denne forbindelse måtte vi ta en vurdering på om vi skulle ha både en admin, som kunne slette tabeller etc. Og en «vanlig» bruker som hadde et begrenset sett med muligheter, eller kun en type bruker, med alle disse rettighetene. [40]

Kryptering:
Kryptering av disk var også et alternativ som ble vurdert, først og fremst med tanke på at uvedkommende kunne få fysisk tilgang til disken. Det tilbys flere tjenester for dette, og vi undersøkte flere muligheter, inkludert Derbys innebygde alternativer, Java JCE, Bouncy Castle Crypto API og BitLocker.

Det finnes også forskjellige løsninger for nettverkskryptering- og autentisering, men vi valgte å ikke fokusere på dette, da NBL-PCen ikke under noen omstendigheter skulle kobles til nettet.

Backup:
BioDemo vil lagre informasjon om brukerne i den permanente databasen. Tap av data kan f.eks. skje hvis harddisken tar kvelden. I så tilfelle ville data mest sannsynlig gå tapt hvis backup ikke er tilstede. Backup var ikke et krav fra oppdragsgiver, så dette var en vurdering gruppen selv måtte ta. Den permanente databasen vil ikke inneholde forskningsdata, men kun data som benyttes i forbindelse med sammenligning av biometriske data. Skulle disse dataene gå tapt, ville konsekvensene bli at det i kommende demonstrasjoner ville være mindre å sammenligne opp mot i databasen, frem til permanente brukere fikk registrert sine biometriske kjennetegn på ny. Selve koden til BioDemo og .jar-filen kommer også til å lagres på minnepinne, noe som innebærer at selve BioDemo-programmet kan settes opp på nytt. På bakgrunn av dette, bestemte vi å utelate backup av databasen.

Tiltak mot SQL-injection:

Brukere får anledning til å legge inn data når de registrerer seg, og derfor ble det nødvendig å undersøke løsninger som kunne forhindre SQL-injection. Valget vi falt på, var PreparedStatements, som det kan leses mer om i kapittel 4.1.6, og på denne måten unngå dynamiske spørringer og som en ekstra bonus: bedret ytelse.

Samtale med oppdragsgiver:
Etter samtale med oppdragsgiver den 19. februar, der vi tok opp mulige sikkerhetsmekanismer, ble det klart at disken på NBL-PCen hadde treg lese - og skrivehastighet, (noe som også kommer fram i 6.1.1). Vi måtte derfor ta hensyn til ytelsesproblematikk. Kryptering av disken og/ eller databasen ville medføre unødvendig treg responstid. I tillegg ville maskinen være låst inne på laben, noe som reudserte risikoen for at uvedkommende skulle få tilgang til disken. På grunn av dette var det et sterkt ønske fra oppdragsgiver om å unngå kryptering.

Våre valg:
Vi kom sammen med oppdragsgiver frem til at den sikkerhetsmekanismen som var mest aktuell å bruke, var brukerautetisering. Brukerautentisering vil foregå ved pålogging på maskinen. Databasen skal kun lagres lokalt på maskinen, og brukeren ville uansett ha tilgang til denne. Med hensyn til ytelsesproblematikk og faren for treg responstid, samt tatt i betraktning at NBL-PCen til enhver tid ville være innelåst på laben, med kun NBL-personell som har adgang, valgte vi i samråd med oppdragsgiver å se bort fra kryptering av både disk og database. Vi konkluderte med at autentisering i forbindelse med pålogging var tilstrekkelig, og at NBL-personell sikrer dette ved å låse maskinen (Windows + L) når de forlater maskinen.

Vi så også behovet for å beskytte databasen mot SQL-injection, Derfor bestemte vi oss for å benytte PreparedStatements i forbindelse med spørringer mot databasen. Dette kan det leses om i kapittel 4.1.6.

6.3 Feilsituasjoner

I utviklingsprosessen av BioDemo ble flere potensielle feilsituasjoner oppdaget ved testing. Mange av disse valgte vi å ta tak i, mens for noen gjorde vi ikke dette. I dette kapittelet vil vi presentere feilsituasjonene, samt hvordan vi valgte å løse dem (eller valgte å ikke gjøre det).

6.3.1 Håndterte feilsituasjoner

Opptatt brukernavn

I register-vinduet var det mulig å taste inn et brukernavn som allerede fantes i databasen. Dette ville medføre problemer når data skulle sendes til databasen. Derfor innførte vi en sjekk på om inntastet brukernavn er unikt i databasen. Hvis dette ikke er tilfellet, får brukeren beskjed om at brukernavnet er opptatt og at det må velges et nytt.

Tomme verdier i register bruker

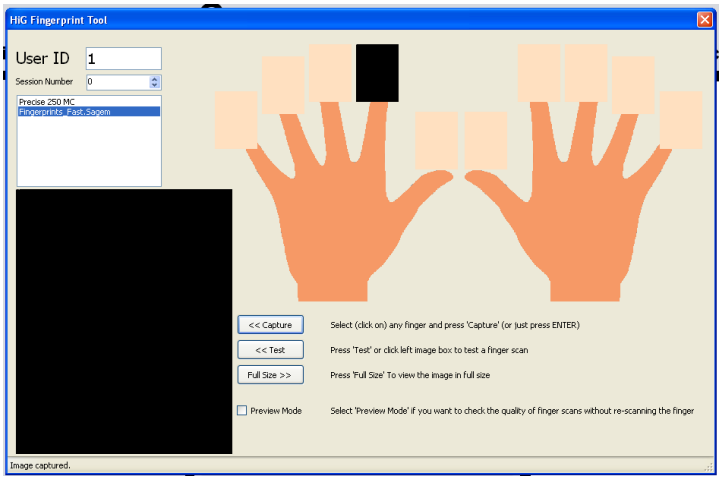
Det var ikke krav om at brukeren må taste inn alle verdiene det blir bedt om i ”Register”-vinduet. Disse verdiene er blant annet etternavn, kjønn og fødselsdato. Når brukeren

velger å ikke taste inn en eller flere av disse verdiene, er det behov for en ekstra funksjon som går gjennom alle disse verdiene og ”nuller” ut de verdiene som skal være null. Hadde vi ikke gjort dette, ville databasen klaget på dette og brukerdata ville ikke sendes til databasen som det skulle. Bruker ville ikke få beskjed om at det var verdier som hadde blitt nullet ut, i og med at bruker selv valgte å ikke fylle disse ut.

Fingeravtrykkmodul

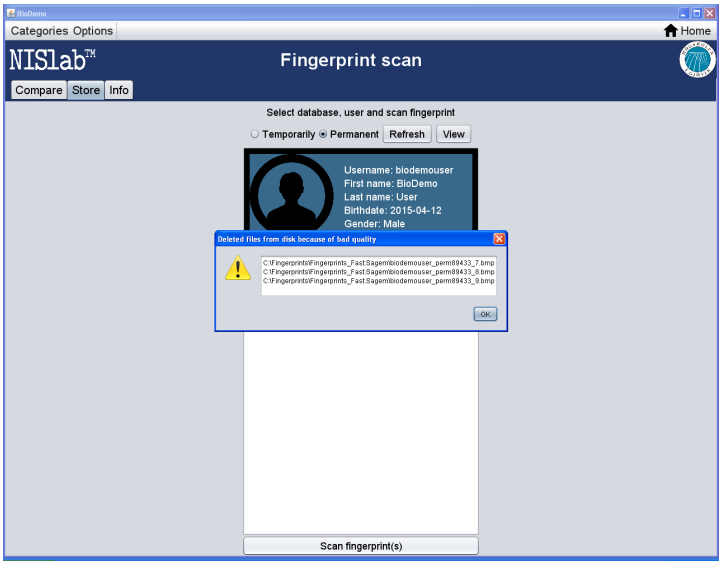
Feil skannet fingreavtrykk i skannerprogram

En av de større utfordringene med kommunikasjonen og samarbeidet mellom BioDemo og de eksterne programmene i fingeravtrykkmodulen, var blant annet at skannerprogrammet ikke håndterte feilsituasjoner som ville påvirke BioDemo. Det var mulig å skanne inn fingeravtrykk av for dårlig kvalitet eller ”tomme” bilder, uten at dette ble håndtert av skannerprogrammet. Fig.49 viser et tilfelle av dette. Her har skannerprogrammet skannet og godtatt et dårlig bilde (den sorte boksen), og bildet blir lagret i den respektive mappen på NBL-PCen. Videre ble dette bildet håndtert av analyseprogrammet, men dette programmet ville ikke kjøre analyse på innskannende fingeravtrykk av for dårlig kvalitet. Her måtte vi lage en slettefunksjon på de fingeravtrykkene som fikk for lav kvalitetsverdi, rett etter at skanneprogrammet var ferdig, for å unngå problemer med analyseprogrammet.



Figur 49: Screenshot fra skannerprogrammet

For å kunne kommunisere til brukerne at bildet ble slettet på grunn av dårlig kvalitet, valgte vi å vise dette gjennom en pop-up vindu som kommer opp etter at brukeren har gått ut av skannerprogrammet. I fig. 50 ser vi dette pop-up vinduet. Her er det oppdaget tre bilder av for dårlig kvalitet, og disse blir slettet.



Figur 50: En pop-up som forteller brukerne bilder er slettet

Keystroke module: DemoKD.exe

Etter å ha validert XML-filene som ble generert av DemoKD.exe ble det observert at noen av disse ikke hadde riktig antall <keyup> merker eller <Keydown>. Dette medførte at DemoKD.exe ikke kunne garantere for at alle inntastingsdata ble lagret i skjemaet. For å løse denne problemstillingen ble det innført sjekker på om brukeren taster inn riktig ord, og hvis ikke programmet klarer å registrere alle verdiene, måtte brukeren taste inn ordet på nytt. Brukeren får beskjed når denne feilen oppstår. En ulempe med en slik løsning er at den gjør programmet mindre brukervennlig, ved at en bruker må taste inn igjen et ord som vedkommende anser som riktig. (Se kodeutdrag 21).

```
        /**Kode klippet ut
        **/
       // check if word match
             if (textBox1.Text.ToString().
             Equals(stringAtIndex[comboBox1.SelectedIndex]) &&
                  typedCounter < 10)
             {
                // check if mismatch between registered values
                 if (keyUp[typedCounter].Count !=
                 keydown[typedCounter].Count)
                 {
                      MessageBox.Show("unfortunately there was a
                      mismatch for the current entry " +
                          "there was a count of " +
                          keyUp[typedCounter].Count + " Key up
                          and  " +
                          keydown[typedCounter].Count + " key
                          down \n please enter again "
                          );
                      retakebtn.PerformClick();
                      return;
                 }
    /**Kode klippet ut
    **/
     private void retakebtn_Click(object sender,
     EventArgs e)
        {
             resetKeyUpDownCounters();
             // empty out textbox
             textBox1.Text = "";

             // set the list to empty (handle initate state
             where typedCounte
             // == -1
             keydown[(typedCounter == -1? 0: typedCounter)] =
             new List<double>();
             keyUp[(typedCounter == -1 ? 0 : typedCounter)] =
             new List<double>();

             // remove all items for current working lists
             textBox1.Focus();
        }
```

Kodeutdrag 21: Håndtering av feilsituasjonen i DemoKD.exe

6.3.2 Ikke håndterte feilsituasjoner

I BioDemo er det enkelte potensielle feilsituasjoner vi har valgt å ikke gjøre noe med. I dette avsnittet har tatt for oss disse, samt begrunnet hvorfor vi valgte å ikke håndtere dem.

Fingeravtrykkmodul

Skanne 2 fingre samtidig i samme bilde

I skannerprogrammet er det mulig å skanne to fingre på samme bildet. Dette er noe som ikke håndteres av skannerprogrammet, og analyseprogrammet gir fingeravtrykket en kvalitet som er godkjent (Over 0). Det betyr at vi ikke har noen mulighet til å kunne

skille disse feilskannede fingeravtrykkene fra de riktige uten manuell sjekk. Derfor har vi kommet frem til at dette er noe som den NBL-ansatte som er med på demostrasjonen skal kunne oppdage og be gjestebrukeren om å skanne på nytt. Hvis det skjer at en bruker skanner inn to fingre på ett bilde, så må NBL-ansatte kunne se dette når bildet vises. Det er ganske lett å se at dette er tilfellet, og fig. 51 viser dette.

Figur 51: To fingre skannet i samme bilde

Bilder av dårlig kvalitet over 0 kan brukes i sammenligning

Analyseprogrammet vil som oftest måle kvaliteten på fingeravtrykket i to kategorier: 0 (dårligst) og 255 (best). Vi har klart å få verdier mellom disse, men det er i hovedsak enten veldig bra eller veldig dårlig. Fingeravtrykk som skannes inn og der kvaliteten måles til 0 vil bli slettet i det skanneprogrammet lukkes og før den eventuelt kan bli sendt til analyseprogrammet. Det kan derimot være mange dårlige fingeravtrykk som ikke går under kategorien 0 (over 0) og dermed ikke slettes. Når disse så brukes videre i sammenligning medfører dette at match-score blir veldig lav, og ofte 0 selv om det er samme finger fra samme person som har blitt sammenlignet. Sagem-skanneren er best på å skanne fingeravtrykk, men det kreves fremdeles at man er svært nøye med å plassere fingeren, har riktig trykk og unngår å vri på fingeren når bildet tas. For å sikre en god sammenligning må NBL-ansatte gi nøye instrukser om hvordan fingeravtrykket skal tas. Se kapittel 3.3.1 om hvor stor forskjell det faktisk kan bli mellom samme finger i forhold til kvalitet.

Info-fane

Tabeller i HTML-filen

Info-taben er en fane i hver modul som lar NBL-ansatte legge inn og redigere informasjon om de forskjellige modulene, enten i form av tekst eller HTML. I BioDemos ”Info”-fane blir programmet SHEF benyttet. (se kapittel 4.1.2) SHEF lar brukeren lett legge inn tabeller i infosiden ved enten å bruke det grafiske grensesnittet i ”Edit” eller ved å bruke HTML-kode direkte. Et problem som oppstår, er når brukeren vil slette hele tabellen, fordi dette kun er mulig ved å redigere HTML direkte og ikke gjennom ”Edit”- funksjonen. Vi anser ikke dette som et stort problem og har valgt å ikke endre på dette. Dette fordi brukeren ikke trenger å forlate SHEF og BioDemo for å redigere HTML-koden. Vi velger heller å nevne dette i brukermanualen, slik at NBL-ansatte er oppmerksomme på dette.

7 Avslutning

I dette kapitlet vil vi drøfte og avklare hvorvidt gruppen har klart å levere på det som skulle leveres og nådd målene som var satt i henhold til kravspesifikasjonen i kapittel 2. I tillegg vil vi drøfte muligheter for videreutvikling av BioDemo etter at gruppen har avsluttet utviklingen. Til slutt vil vi drøfte gruppearbeidet vårt.

7.1 Drøftinger

7.1.1 Resultater

Prosjektets formål var å lage et program som skulle demonstrere hvordan biometriske data ble skannet inn, for så å kunne sammenligne og identifisere personer. NBL trengte et program de kunne bruke for å demonstrere for personer utenfor NBL hva og hvordan biometriske innsamlinger og analyse foregår. Bakgrunnen for dette er at NBL og NisLab ofte får gjester og besøkende som gjerne vil lære om hva som foregår hos NBL, og før BioDemo var det ingen måte å kunne vise dette enkelt og forståelig. Les gjerne om hvordan det var for demostrasjoner før og etter BioDemo i kapittel 1.1.1.

Resultatet av dette prosjektet er et program - BioDemo, som kan benyttes til slike demostrasjoner. BioDemo består 15.mai 2015 av to moduler; en for fingeravtrykk og en for tastaturgjenkjenning. Fingeravtrykkmodulen er klar til bruk. Tastaturgjenkjenningsmodulen kan også brukes, men det gjenstår mindre elementer og småfiks. Fingerårermodulen fikk gruppen ikke begynt på. Årsaken omtales i kapittel 3.1.1. I utviklingen gav oppdragsgiver gruppen stor frihet til å løse problemstillingen, men ønsket også å ta del i utviklingen gjennom jevnlige møter der det var rom for å komme med innspill. Dette har resultert i et produkt som gruppen mener har høy kvalitet. I tillegg til at det er tilrettelagt for å utvide BioDemo med ytterligere moduler. BioDemo er produsert i en slik form at NBL kan ta i bruk programmet ved prosjektes slutt, noe gruppen håper blir gjort. Alle krav satt i kravspesifikasjonen, med unntak av fingerårermodulen (se kapittel 2) har blitt løst av gruppen.

Som en del av at Scrum-modellen ble fulgt i prosjektet, ble det holdt møter med oppdragsgiver annenhver uke, ved slutten av hver sprint. Oppdragsgiver ønsket å få jevnlige oppdateringer og gode muligheter til å komme med innspill underveis. Denne kontakten mellom gruppen og oppdragsgiver har vært veldig bra, og gruppen har tatt kontakt med oppdragsgiver også utenom disse møtene for å få svar på uklarheter eller spørsmål. Oppdragsgiver har under hele prosjekttiden vært veldig imøtekommende, og bistått så godt han kunne med det vi lurte på. Ved sprint update-møte ble grafisk brukergrensesnitt og skisser gjennomgått og det ble holdt en grafisk utvelgelse, fremfor å gå gjennom backloggen. Dette fordi det var enklere for oppdragsgiver å gi innspill på hva som burde gjøres og på utført arbeid. Det var to demostrasjoner for oppdragsgiver underveis i prosjekttiden, der de to modulene ble presentert og oppdragsgiver fikk prøve de respektive moduelene. Resultatet av disse demostrasjonene var at oppdragsgiver kom med mange

konstruktive tilbakemeldinger, som ble tatt hensyn til i utviklingen.

Vi fulgte Gantt-skjemaet godt (se vedlegg A) , også frem til siste uke. Gruppen gjennomførte forprosjekt, to delleveranser, seks sprinter, brukertesting og hadde en intensiv rapportskrivingsperiode etter siste sprint. Arbeidet på rapporten ble utført kontinuerlig, men med gradvis opptrapping mot slutten av prosjekttiden. En fordel med å jobbe etter Scrum, var de fastsatte møtene med oppdragsgiver. Dette tvang gruppen til å følge rutinemessige runder med sprint update-møter ved sprintstart, utvelgelse av oppgaver for neste sprint, gjennomgang av fullførte oppgaver og eventuelle utfordringer ved slutten av en sprint.

Integrasjonen mellom de eksterne programmene og komponentene og vår del av BioDemo var kort sagt krevende. Det var mange eksterne komponenter som skulle integreres og fungere bra med BioDemo, og denne integrasjonen tok mye tid. Fig 25 viser hvordan løsningen til slutt ble med alle komponentene. For fingeravtrykkmodulen var det et linsensiert program som blant annet begrenset vår påvirkning til hvordan fingeravtrykkene ble skannet. I tillegg løste vi flere feilsituajsoner knyttet til dette. I kapittel 6.3 omtaler vi de feilsituasjonene som ble håndtert, men også de som av forskjellige grunner ikke håndteres. For tastaturgjenkjenning var det blant annet utfordringer med MATLAB-funksjoners kommunikasjon med Java, som medførte en del omveier hvor C#-funksjoner fungerte som ”broen” mellom Java- og MATLAB-funksjonene. Les mer om dette i kapittel 6.1.4.

7.1.2 Alternativer

En av de største utfordringene gruppen møtte på, var at vi måtte vente på mye av utstyret vi trengte til dette prosjektet. I begynnelse av utviklingstiden måtte vi vente litt over tre uker på NBL-PCen som hadde både programmene, tilhørende lisenser og utstyret til fingeravtrykkskanning. Hvis vi hadde visst dette på forhånd, hadde vi mest sannsynlig begynt med tastaturgjenkjenningsmodulen. Dette fordi den ikke krever noe spesielle programmer bundet av linsenser eller noe spesielt utstyr (kun tastatur). Når det gjelder fingeravtrykkmodulen som vi ikke rakk å starte på, var dette på grunn av at en annen student trengte å bruke utstyret tilknyttet til fingeråre-skanning. Dette medførte at vi aldri fikk startet på denne modulen.

7.2 Videre arbeid

Det er flere forbedringsmuligheter i BioDemo, både i forbindelse med å bygge på med flere moduler og utvide funksjonaliteten. Det var også et ønske fra oppdragsgiver at BioDemo ble utviklet med tanke på at det skulle være mulig å legge til flere moduler utviklet av andre enn gruppen, etter at prosjektet var avsluttet. Oppdragsgiver har også kommunisert at det i mai skulle være en annen student som skal jobbe med en annen modul med webkamera og irisgjenkjenning.

Punkter som gruppen anser som forbedringspotensialer for BioDemo:

- **Bedre brukervennlighet for endringer i BioDemo.**
  Det er flere muligheter for å tilpasse bedre for endringer i BioDemo. Dette gjelder for eksempel om NBL-brukeren ønsker å legge til, fjerne ord eller endre eksisterende ord. Gruppen har ikke tilpasset noe GUI-funksjonalitet for dette. For å legge

til ord, må NBL-brukeren inn i Java-prosjektet og legge til disse i en array der alle ordene ligger definert. En løsning for å gjøre dette mer brukervennlig, er å benytte konfigurasjonsfiler, hvor alle ordene som skal benyttes til tastaturgjenkjenningsmodulen ligger. Ved oppstart av programmet kunne det være en sjekk på om det var noen endringer i denne filen og som så oppdaterte databasen hvis dette var tilfellet.

- **Ytterligere tilpasninger i databasene**
  Databasene er tilpasset endringer når det gjelder å legge til nye tabeller knyttet til en ny modul. For å gjøre dette, må utvikleren lage nytt SQL-statement i en klasse, og så vil de nye tabellene opprettes i databasene. I tilegg er slettefunksjonen tilpasset en situasjon der det er flere tabeller enn første versjonen ved 15. mai 2015. Men noe som kan forbedres med databasen er:
  - Bedre tilpassede løsninger for sletting av en bruker i den permanente databasen. Det har ikke blitt lagd noen løsning for dette i prosjektet.
  - I tillegg kom oppdragsgiver ganske sent i utviklingsprosessen med et innspill om å gjøre det mulig å kunne lagre forskjellige versjoner av samme finger for en bruker i databasen. Dette fordi det er ganske store forskjeller mellom typer fingeravtrykksskannere. For å løse dette, må det legges til en ekstra tabell i fingeravtrykk-delen av databasene med alle skannere, som konbles til den aktuelle fingeren og brukeren.
- **Hjelpefunksjon i programmet**
  Det var planlagt å lage en dynamsik hjelpefunksjon i BioDemo basert på samme struktur som det er på info-taben i programmet. Dette ble til slutt bortprioritert på grunn av at det ikke var noen krav om å ha en slik løsning og tiden ikke strakk til.
- **Ytterligere moduler**
  Som nevnt tidligere, ble fingerårermodulen aldri startet på, men denne modulen kan utvikles og legges til BioDemo etter at prosjektperioden er avsluttet. Dette gjelder også andre moduler. Det er mange biometri-områder som NBL jobber med som kan passe i BioDemo, blant annet; musebevegelser, iris- og ansiktgjenkjenning.

I og med at tasturgjenkjenningsmodulen mangler den siste ”finpussen” har gruppen avtalt med oppdragsgiver at dette skal gjøres etter at rapporten og prosjektperioden er avsluttet. Dette skjer på frivillig basis, og gruppen er villig til å gjøre dette fordi ingen av gruppens medlemmer ønsker å avslutte et så stort prosjekt når det bare er småpuss som gjenstår. Gruppen ønsker å levere et fullstendig produkt, klar til bruk for NBL.

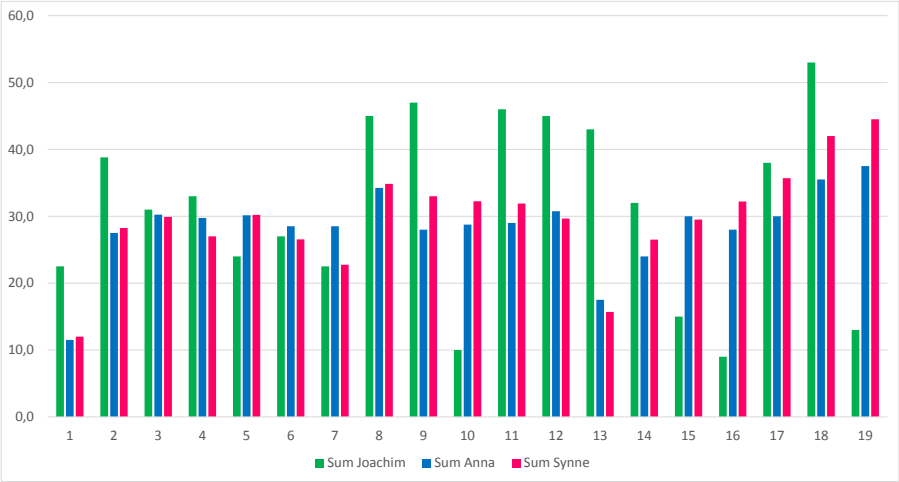
7.3 Evaluering av gruppearbeidet

Gruppen har gjennom hele prosjekttiden hatt møter og felles ”arbeidsdager” der vi satt sammen og jobbet. Oppgavefordelingen har hovedsaklig vært individuell, men noen har blitt utført sammen med en eller hele gruppen. Eksempler på slike arbeidsoppgaver var brukertester av grafisk grensesnitt, og testing av BioDemo. Det har vært fokus på å la hver av gruppemedlemmene ”spesialisere” seg og få sine ansvarsområder. Eksempler på disse var utforming av grafisk grensesnitt, databaseutforming og SQL-spørringer, MATLAB og I/O. Det var i tillegg tildelt forskjellige roller fra begynnelsen, noe som ble opprettholdt under hele prosjekttiden. Innad i gruppen var det lav terskel for å spørre om hjelp og bistand, og uenigheter ble alltid løst med avstemning der fordelene med å være tre på gruppen kom tydelig frem. Det var en av medlemmene som var tildelt rollen som gruppeleder, men gruppens organisasjon var i praksis veldig horisontal.

Det har blitt gjennomført faste møter med både oppdragsgiver og veileder; oppdragsgivermøtene var som nevnt tidligere, annenhver uke, ved slutt og start av sprintene, mens veiledertimene har vært hver tirsdag - med noen få unntak. Timene med veileder ble brukt til å ta opp utfordringer og spørsmål, i tillegg til innspill på rapportutkast og andre løsninger under utviklingen.

7.3.1 Disponering av tid

Fra starten av prosjektet ble det besluttet at hver av gruppemedlemmene skulle investere minst 30 timer hver uke på BioDemo. Dette var noe alle medlemmene mer eller mindre overholdt under hele prosjekttiden. Tilsammen jobbet gruppen nesten 1700 timer, fordelt på ca 595 på Joachim, ca 540 på Anna og Synne jobbet ca 565 timer. (Se vedlegg L for ekstakte timelister) Det var noen uker der noen av medlemmene jobbet mindre enn 30 og ofte over 30 timer. Grafen i fig 52 viser arbeidstimer pr uke pr. gruppemedlem.



Figur 52: Antall arbeidstime pr gruppemedlem

Det var flere elementer som tok mye ekstra tid i utviklingen av BioDemo, enn det gruppen hadde planlagt. En av utfordringene som tok mest tid var implementasjonen og kommunikasjonen av de eksterne programmene med våre utviklede deler av BioDemo. Det var blant annet, som nevnt over, store problemer med integrasjonen mellom MATLAB og tastaturgjenkjenningsmodulen. Denne problemstillingen tok mye tid og førte direkte til at gruppen ikke fikk ”finpusset” på tastaturgjenkjenningsmodulen før utviklingsperioden var avsluttet. Andre ting som tok tid, var brukertester og testing, noe som vi valgte å bruke tid på fordi vi mente dette ville resultere i bedre kvalitet i sluttproduktet. Et annet element som påvirket tiden, var forsinkelsene med NBL-PCen, men gruppen prøvde å tilpasse dette ved å gjennomføre de oppgavene som var mulig uten denne PCen. Av denne grunn ble det ikke fullstendig stopp i utviklingen, men det preget helt klart utviklingen.

7.4 Konklusjon

Sluttproduktet for denne bacheloroppgaven er et både kjørbart og brukbart program. BioDemo, som viser enkelt og forståelig hvordan man skanner og samler utvalgte bio-

metriske data og bruker disse i analyse og sammenligning. BioDemo består pr. 15. mai 2015 av to moduler; fingeravtrykk- og tastaturgjennkjenningsmodulen. BioDemo oppfyller alle krav spesifisert i kapittel 2, med unntak av kravene under fingerårermodulen. Årsaken til dette er fordi denne modulen ble aldri igangsatt på grunn av eksterne faktorene. Det er flere forbedrings- og videreutviklingsmuligheter i BioDemo. Noen løsninger har gruppen valgt bort på grunn av tidsmangel, andre har dukket opp underveis og ikke prioritert da det ikke er satt som krav i kravspesifikasjonen.

BioDemo gjør jobben lettere for NBL når de skulle ønske å demostrere visuelt hva de jobber med, og i tilfelle Statsminister Erna Solberg stikker innom, er det bare å starte programmet og imponere henne. Videre håper gruppen at NBL tar i bruk BioDemo, men også benytter muligheten til å videreutvikle og utvide med ytterligere moduler. Kanskje dette kan være en aktuell bacheloroppgave for neste år?

Bacheloroppgaven har vært utfordrende og lærerrik for gruppen. Samtlige medlemmer sitter igjen med mye ny erfaring, forbredrede kunnskaper og en god samvittighet for et vellykket og godt gjennomført bachelorprosjekt. Vi mener alle at BioDemo er et godt produkt og håper vår kunde og oppdragsgiver blir fornøyde med resultatet.

Bibliografi

[1] Paolo, Azzoni. and Stefano, Gosetti. and Luca, Geretti. and Konstantinos, Rantos. and Balázs, Berkes. Voice/facial recognition demonstrator validation and verification report (online). URL: http://www.newshield.eu/wp-content/uploads/2015/01/NSHIELD-D7.10_Voice_Facial_Recognition_demonstrator_Validation_Verification_Report.pdf (Visited 22 Jan 2015).

[2] Nicolas, Tsapatsoulis. and Zenonas,Theodosiou. and Marios, Milis. and Anastasis, Kounoudes. Polybio: Multimodal biometric data acquisition platform and security system. Technical report, SignalGeneriX Ltd og Cyprus University of Technology, Limassol, Cyprus.

[3] Davide Maltoni [et al.]. 2009. *Handbook of fingerprint recognition*. Springer, 2 edition.

[4] Hafez Barghouthi. 2009. Keystroke dynamics - how typing characteristics differ from one application to another. URL: http://idtjeneste.nb.no/URN:NBN:no-bibsys_brage_9845.

[5] Erna på gjøvik-visitt: Vi har store utfordringer (online). 2014. URL: http://www.oa.no/Erna_p__Gj_vik_visitt-5-35-9806.html (Visited 2015 Mar 23).

[6] Trello. Inc . Trello (online). 2015. URL: https://trello.com (Visited 13 Apr 2015).

[7] Microsoft. Kundestøtten for windows xp er avsluttet (online). 2014. URL: http://windows.microsoft.com/nb-no/windows/end-support-help (Visited 2015 Apr 02).

[8] Lov om behandling av personopplysninger (personopplysningsloven) (online). 2000. URL: https://lovdata.no/dokument/NL/lov/2000-04-14-31 (Visited 2015 Feb 18).

[9] E R. Henry. 1900. *Classification and uses of Fingerprints*. George Routledge and Sons, 1 edition. URL: http://www.clpex.com/Information/Pioneers/henry-classification.pdf.

[10] Wikipedia. Fingerprint recognition — wikipedia the free encyclopedia (online). 2015. URL: http://en.wikipedia.org/w/index.php?title=Fingerprint_recognition&oldid=642143857 (Visited 2015 Apr 02).

[11] Fingerprinting. Live scan fingerprinting (online). 2015. URL: http://www.fingerprinting.com/live-scan-fingerprinting.php (Visited 27 Apr 2015).

[12] ID Control. Biometric authentication method pro's and con's (online). 2015. URL: http://www.idcontrol.com/keystroke-biometrics/biometric-authentication-method-pros-and-cons (Visited 01 Mai 2015).

[13] Ian Sommerville. 2011. *Software Engineering*. Pearson, 9 edition.

[14] Thomas M. Connolly. 2010. *Database systems : a practical approach to design, implementation, and management*. Addison-Wesley, 5 edition.

[15] Wilbert O.Galitz. 2002. *The essential guide to user interface design*. Wiley, 2 edition.

[16] Jeff Johnson. 2000. *GUI bloopers don'ts and do's for software developers and web designers*. Morgan Kaufmann Publishers, 1 edition.

[17] Yegor Bugayenko. Oop alternative to utility classes) (online). 2014. URL: http://www.yegor256.com/2014/05/05/oop-alternative-to-utility-classes.html (Visited 12 Mai 2015).

[18] David A. Wheeler. The free-libre / open source software (floss) license slide (online). 2007. URL: http://www.dwheeler.com/essays/floss-license-slide.html (Visited 11 Mai 2015).

[19] gnu.org. Why you shouldn't use the lesser gpl for your next library (online). 2014. URL: http://www.gnu.org/philosophy/why-not-lgpl.html (Visited 11 Mai 2015).

[20] David Turner. The lgpl and java (online). 2014. URL: https://www.gnu.org/licenses/lgpl-java.html (Visited 11 Mai 2015).

[21] SHEF. Shef - swing html editor framework (online). URL: http://shef.sourceforge.net/ (Visited 12 Mai 2015).

[22] freepik. freepik - graphic resource (online). URL: http://www.freepik.com/ (Visited 12 Mai 2015).

[23] Oracle. Java(tm) platform standard ed.7 (online). 2014. URL: http://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html#setDefaultCloseOperation%28int%29 (Visited 28 Apr 2015).

[24] Mark Dowd, John McDoald, Justin Schun. 2013. *The art of software security assessment*. Addison-Wesley Professional, 5 edition.

[25] Microsoft Developer Network. Database acces (c sharp vs java) (online). URL: https://msdn.microsoft.com/en-us/library/ms228366%28VS.80%29.aspx (Visited 12 Mai 2015).

[26] Robert, Martin. . When tdd doesn't work. (online). 2014. URL: http://blog.8thlight.com/uncle-bob/2014/04/30/When-tdd-does-not-work.html (Visited 29 Jan 2015).

[27] Shivani, Acharya. and Vidhi, Pandya. . 2012. Bridge between black box and white box – gray box testing technique. In *International Journal of Electronics and Computer Sc ience Engineering*.

[28] Ivo, Gomes. and Pedro, Morgado. and Tiago, Gomes. and Rodrigo, Moreira. 2015. An overview on the static code analysis approach in software development. URL: http://paginas.fe.up.pt/~ei05021/TQSO%20-%20An%20overview%

20on%20the%20Static%20Code%20Analysis%20approach%20in%20Software%20Development.pdf (Visited 02 Apr 2015).

[29] FindBugs. Findbugs (tm) - find bugs in java programs (online). URL: http://findbugs.sourceforge.net/ (Visited 12 Mai 2015).

[30] Usability.gov. Running a usability test (online). 2015. URL: http://www.usability.gov/how-to-and-tools/methods/running-usability-tests.html (Visited 05 Apr 2015).

[31] Crystal Dew World. Crystaldiskmark (online). URL: http://crystalmark.info/software/CrystalDiskMark/index-e.html (Visited 12 Mai 2015).

[32] What is the definition of type safety? (online). 2012. URL: http://www.quora.com/What-is-the-definition-of-type-safety (Visited 2015 Apr 27).

[33] Oracle. Primitive data types (online). URL: https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html (Visited 12 Mai 2015).

[34] Microsoft Developer Network. byte (c sharp reference) (online). URL: https://msdn.microsoft.com/en-us/library/5bdb6693.aspx (Visited 12 Mai 2015).

[35] Oracle. Primitive data types (online). 2015. URL: https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html (Visited 12 Mai 2015).

[36] Microsoft Developer Network. byte (csharp reference) (online). 2015. URL: https://msdn.microsoft.com/en-us/library/5bdb6693.aspx (Visited 12 Mai 2015).

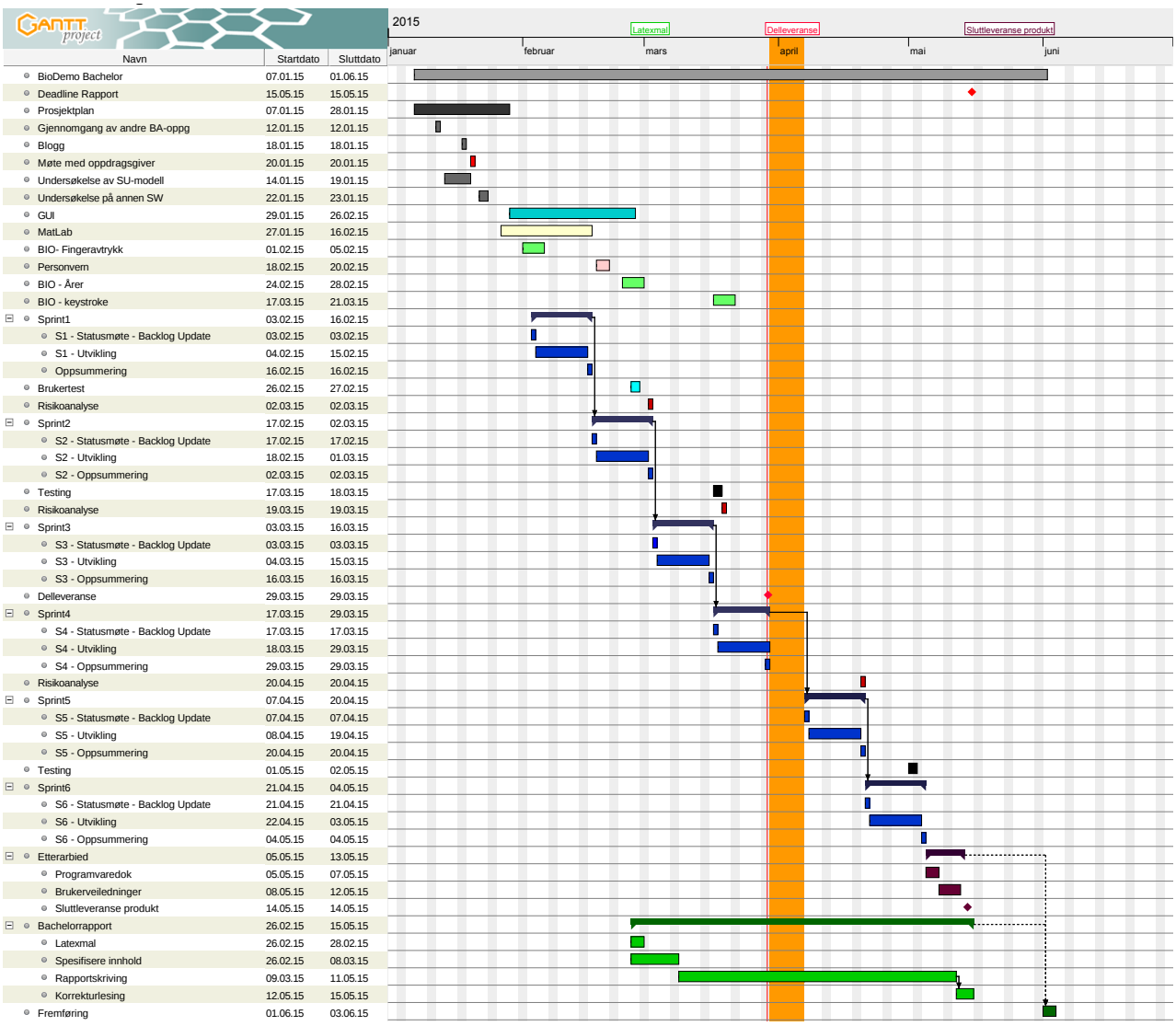
[37] Oracle. What is a locale?) (online). 2012. URL: http://docs.oracle.com/cd/E23824_01/html/E26033/glmbx.html (Visited 12 Mai 2015).

[38] TechSource. Matlab builder ja (for java language (online). 2015. URL: http://techsource-asia.com/products-a-solutions/products/45-matlab-builder-ja-for-java-language.html (Visited 01 Mai 2015).

[39] TechSource. Matlab builder ne (for microsoft .net frameworks (online). 2015. URL: http://techsource-asia.com/products-a-solutions/products/44-matlab-builder-ne-for-microsoft-.net-framework.html (Visited 01 Mai 2015).

[40] Apache Derby. Apache derby: Documentation v10.11 (online). URL: https://db.apache.org/derby/docs/10.11/security/index.html (Visited 12 Mai 2015).

A Ganttskjema



Navn	Startdato	Sluttdato
BioDemo Bachelor	07.01.15	01.06.15
Deadline Rapport	15.05.15	15.05.15
Prosjektplan	07.01.15	28.01.15
Gjennomgang av andre BA-oppg	12.01.15	12.01.15
Blogg	18.01.15	18.01.15
Møte med oppdragsgiver	20.01.15	20.01.15
Undersøkelse av SU-modell	14.01.15	19.01.15
Undersøkelse på annen SW	22.01.15	23.01.15
GUI	29.01.15	26.02.15
MatLab	27.01.15	16.02.15
BIO- Fingeravtrykk	01.02.15	05.02.15
Personvern	18.02.15	20.02.15
BIO - Årer	24.02.15	28.02.15
BIO - keystroke	17.03.15	21.03.15
Sprint1	03.02.15	16.02.15
S1 - Statusmøte - Backlog Update	03.02.15	03.02.15
S1 - Utvikling	04.02.15	15.02.15
Oppsummering	16.02.15	16.02.15
Brukertest	26.02.15	27.02.15
Risikoanalyse	02.03.15	02.03.15
Sprint2	17.02.15	02.03.15
S2 - Statusmøte - Backlog Update	17.02.15	17.02.15
S2 - Utvikling	18.02.15	01.03.15
S2 - Oppsummering	02.03.15	02.03.15
Testing	17.03.15	18.03.15
Risikoanalyse	19.03.15	19.03.15
Sprint3	03.03.15	16.03.15
S3 - Statusmøte - Backlog Update	03.03.15	03.03.15
S3 - Utvikling	04.03.15	15.03.15
S3 - Oppsummering	16.03.15	16.03.15
Delleveranse	29.03.15	29.03.15
Sprint4	17.03.15	29.03.15
S4 - Statusmøte - Backlog Update	17.03.15	17.03.15
S4 - Utvikling	18.03.15	29.03.15
S4 - Oppsummering	29.03.15	29.03.15
Risikoanalyse	20.04.15	20.04.15
Sprint5	07.04.15	20.04.15
S5 - Statusmøte - Backlog Update	07.04.15	07.04.15
S5 - Utvikling	08.04.15	19.04.15
S5 - Oppsummering	20.04.15	20.04.15
Testing	01.05.15	02.05.15
Sprint6	21.04.15	04.05.15
S6 - Statusmøte - Backlog Update	21.04.15	21.04.15
S6 - Utvikling	22.04.15	03.05.15
S6 - Oppsummering	04.05.15	04.05.15
Etterarbied	05.05.15	13.05.15
Programvaredok	05.05.15	07.05.15
Brukerveiledninger	08.05.15	12.05.15
Sluttleveranse produkt	14.05.15	14.05.15
Bachelorrapport	26.02.15	15.05.15
Latexmal	26.02.15	28.02.15
Spesifisere innhold	26.02.15	08.03.15
Rapportskriving	09.03.15	11.05.15
Korrekturlesing	12.05.15	15.05.15
Fremføring	01.06.15	03.06.15

Figur A.1: Gantt-skjema for prosjektet

B Backlog

Hvis figuren er litt vanskelig å lese så er det lurt å benytte seg av zoome-funksjonaliteten i pdf-leseren og zoome inn på ca +120%.

r. *	Som en...	Ønsker jeg...	Slik at...	Del
1	Bruker (alle)	At resultatet presenteres under sammenligningsbildene i tekst og verdi for match score	Bruker forstår resultatet av sammenligningen.	FP
2	Bruker (alle)	A få opp en meny som viser mine valgmuligheter	Jeg har oversikt på det jeg gjør.	Hele
3	Bruker (NBL)	A få opp en feilmelding når programmet mislykkes i å hente data	Jeg blir bevisst på at et problem har oppstått.	Hele
4	Bruker (alle)	A få opp en hyggelig velkomstmelding når programmet startes	Jeg blir motivert for å fortsette.	Hovedvindu
5	Bruker(Erna)	A bli presentert for et oversiktelig grensesnitt, med to bilder ved siden av hverandre som viser sammenligningsprosessen	Jeg forstår hva som foregår.	Hele
6	Bruker(NBL)	A kunne hoppe frem og tilbake mellom forskjellige moduler uten at innsamlet data forsvinner	Kan bruke applikasjonen dynamisk.	Hele
7	Applikasjon	Opprette en kobling med lokal database ved oppstart	Jeg har noe å sammenligne input med.	Hele
8	Bruker(NBL)	At det er manuelt fremsteg i demo	Jeg får tid til å fortelle hva som skjer underveis og kan øke kunnskapen til Erna.	Hele
9	Bruker(Gjest)	A kunne ta et fingeravtrykk, få det displayet og kunne velge om jeg vil lagre det midlertidig i databasen (så lenge programmet kjører)	Det lagrede fingeravtrykket kan brukes i sammenligning så lenge programmet kjører.	FP
10	Bruker(Gjest)	A se mitt eget fingeravtrykks særegenheter	Jeg kan se hva som skiller mitt fingeravtrykk fra andres.	FP
11	System	Å kunne slette fingeravtrykk ved utgang av programmet	Jeg unngår at fingeravtrykket blir ulovlig lagret i NBL-databasen.	FP
12	Bruker(NBL)	Å starte programmet enkelt og gå inn til fingeravtrykkdelen enkelt og raskt	Jeg raskt kan sette opp demoen.	FP
13	Bruker(NBL)	Å få informasjon om alle tilkoblede sensorer	Jeg kan se hvilke sensorer som har blitt detektert av programmet.	
14	Bruker(Gjest)	A kunne lagre opplysninger som navn, bilde ol. i en "profil"	Dette kan brukes hvis et fingeravtrykk matcher mitt.	
15	Bruker(Gjest)	Å kunne sammenligne fingeravtrykket mitt mot databasen	Jeg kan bruke det til sammenligning og få opp resultatet av sammenligningen.	FP
16	Bruker(Gjest)	A kunne sammenligne to fingeravtrykk (ikke sjekk i database, men kun de nyscanna fingeravtrykk)	Jeg kan se dem opp mot hverandre	FP
17	System	Kunne huske tidligere brukernavn	Slippe å skrive det samme omigjen	
18	Bruker(Gjest)	a bli presentert for et enkelt og intinuitivt grensesnitt som ligner på øvrige moduler	jeg lettere forstår programflyten og fort kan sette meg inn i funksjonene	hele
19	Bruker(Gjest)	å komme direkte til Compare-del der jeg får se live scan av fingeren min	kan følge med på det som skjer	FV
20	Bruker(Gjest)	å kunne trykke på en clear-knapp som rensker skjermen (data)	at jeg når som helst kan begynne forfra og skanne på nytt	FV
21	Bruker(Gjest)	å kunne klikke på knapp (sammenlign) og få opp resultat for treff mot databasen	kan se hvilke som er like/ligner på mine	FV
22	Bruker(Gjest)	å kunne klikke på en kapp som gjør at bilde av fingerårer lagres i mappe (capture)	dette etterpå kan brukes i sammenligningen	FV
23	Bruker(NBL)	å kunne registere mine fingerårer i databasen	de senere kan brukes til sammenligning	FV
24	Bruker(NBL)	å legge til flere forskjellige fingre (1-10)	jeg kan ha flere fingre fra meg i databasen	FV
25	Bruker(Gjest)	å se tall/verdier om matchprosent og/eller om finger	jeg kan få mer info om fingeren	FV
26	Bruker(Gjest)	å se og scanne fingeren min	jeg kan se årene i finger og få forklaring om dette av NBL	FV
27	Bruker(Gjest)	å få opp 2 bilder av fingerårer side ved side	jeg kan se klart forskjellen mellom fingerene	FV
28	Bruker(Gjest)	å starte på nytt med et annen ord	jeg kan prøve igjen	KD
29	Bruker(NBL)	å velge hvilket ord som erna skal taste inn	jeg kan selv velge dette	KD
30	Bruker(Gjest)	å få sammenlignet mitt ord(verdier) med databasen	jeg kan se hvem som kom nærmest meg	KD
31	Bruker(Gjest)	å få info om hvilket ord jeg skal skrive og når jeg kan skrive det	jeg vet hva jeg skal skrive	KD
32	Bruker(Gjest)	å få opp en graf som viser mitt ord(verdier) og databasens match	jeg kan se hvor likt/forskjellig det er	KD
33	Bruker(Gjest)	å kunne gjenskrive ordet	hvis jeg skriver feil el lign kan gjenta	KD
34	Bruker(NBL)	å skrive inn ett ord flere ganger	tallverdier kan lagres og gjennomsnitt utregnes og legges i databasen	KD
35	Database	å ha noen få ord som brukere må taste inn (ordliste)	kan unngå for stor database med for mange verdier	KD
36	Bruker(NBL)	å velge hvilke ord i ordliste som jeg skal skrive	jeg kan ha flere forskjellige ord i databasen på meg	KD
37	Bruker(NBL)	registerere (min egen) tastegjenkjenning i databasen	at det senere kan brukes til sammenligning og opptegning av graf	KD
38	Bruker(Gjest)	å gå direkte til en Compare del hvor jeg blir bedt om å skrive ett ord	jeg kan taste dette inn	KD

Figur B.2: Backlog i form av user stories for BioDemo

C Møtereferater

C.1 Oppdragsgivermøter

Her følger alle referater fra oppdragsgivermøte med Patric Bours.

20.01.2015

Referat oppdragsgiver 20.01.2015
Alle til stede - Forventninger: Patrick ønsker en demostrator som viser hva vi gjør på biometrilabben (den trenger ikke å være selvforklarende siden når demostratoren skal displayes vil folk fra labben være der)

- Patrick ønsker at vi skal få med så mye som mulig, men om vi skulle startet å prioritere så bør vi starte med det som absolutt må være med (fingeravtrykk, blood veins, tastatur gjenkjenning) Ønske om at det skal se litt fancy ut.
- Patrick ønsker at alle skal få samme informasjon. Det forventes ikke av brukerne våre at de er eksperter på biometri.
- Patrick ser for seg et program for alt (med menyer for å navigere... (snakker ofte om en hovedmeny)) Patrick svarte at kodesnuttene var skrevet primært i mathlab, men kjenner til kode i både java og C (fikk intrykket tidligere at også python kunne være et språk.) Kodesnutter som han også snakket om var java kode for språkgjenkjenning og kontinuelig gjenkjenning. Patrick sier at utviklingsmodellen er opp til oss og at han kan delta mye/eller lite ettersom vårt behov.
- Patrick snakket om løsninger som gjør at du kan skille mellom å permanent lagre data og en annen om å slette etter det ikke lenger er behov for (demostrator slås av / programmet avsluttes). Dette kan eventuelt gjøres med flere databaser eller moduser og triggere (eller andre løsninger) Viktig at noe kan slettes og andre kan beholdes. Vi står fritt til å modifisere deres kode slik vi selv ønsker, men må være klar over å benytte ulike metrics kan føre til ulike resultater. Patrick skulle forespørre biometri labben om hva slags informasjon om biodemo skal være tilgjengelig for alle.
- Patrick som oppdragsgiver sier at han ofte endrer mening og det er derfor mer naturlig med en smidigere modell. Vi kommer til å jobbe tett med eksperter (ekspert på fingeravtrykk, iris, osv) Patrick ønsker at vi skal ha en åpen dialog. Dvs at vi står fritt til å spørre ofte om hvordan ting skal se ut, hvordan programmet skal oppføre seg. Patrick nevnte at det passer best med møter på tirsdager klokka 09.00 eller på onsdag morgen.

03.02.2015

Referat, 03.02.15 **Referent:** Anna Kaardal
Til stede: Alle

Før møtet hadde gruppen valgt ut elementer fra user stories til backloggen. De elementene vi hadde vurdert til høyest prioritet, markerte vi. Vi presenterte backloggen for oppdragsgiver, og han hadde ingenting å legge til på denne. Vi har konkludert med at vi kan selv ha hovedansvaret for backloggen, og at det beste er å vise oppdragsgiver hva vi faktisk har tenkt til å gjøre i sprinten, f.eks. med tegninger, noe som gjør det mulig å vise mer konkret hva vi mener, og deretter evnt. tilpasse backloggen etter tilbakemeldinger på det vi har vist.

Punkter i forbindelse med funksjonalitet og GUI i fingeravtrykksmodulen:

- Brukeren får ett fingeravtrykk scannet, og deretter blir dette testet opp mot alle fingeravtrykk i databasen.
- Registrer: Her registreres bruker for permanent lagring (kan f.eks. være forskere ved NBL) -> Vi har også diskutert muligheten for å kunne hake av for om man vil at avtrykket skal lagres permanent i databasen, eller slettes når programmet lukkes. En ting som også er viktig å få med her, er et alternativ for valg av scanneren som skal brukes.

- Compare: Skal kunne foreta live-scanning av fingeravtrykk, og deretter vise fingeravtrykket til brukeren. I tillegg skal et bilde vises ved siden av, der man viser alle fingeravtrykkene (ett og ett) som sammenlignes med brukerens fingeravtrykk. Målingspunkter for disse skal også vises. Etter sammenligningen, blir brukeren presentert for topp 4 match", med bilde og match score. Det vil blinke rødt rundt de som ikke matcher, og grønt ved match.
- På grunn av at resultatet vises under "Compare", trenger vi ingen Resultside, slik som i de første designutkastene våre.

Design og farger:
- NISLab-logo skal vises. Fargebruk er opp til oss.

Annet om programutviklingen:

- Det eksisterer allerede et program som detekterer sensorer. Anlyseprogramvare for fingeravtrykksscanning er skrevet i C Shar. Vi må kartlegge hvordan dette fungerer med Java.
- P.g.a. lisensering må vi benytte oss av en spesifikk PC, som blir satt opp for oss. Denne er ikke koblet til nettet.
- Vi må vurdere om vi vil bruke ordet registereller enroll førstnevnte enklest å forstå for Erna", mens enroller ordet som brukes på fagspråket.
- Oppdragsgiver nevnte at det kan bli behov for en tredje tab: den skal ikke være interaktiv, og forklare brukeren hvordan fingeravtrykksscanning foregår. Dette kan f.eks. innebære å vise et eksempelfingeravtrykk med målingspunkter. -> Dette kan legges til på et senere tidspunkt.
- Vi skal fokusere på at programmet vi utvikler skal kunne videreutvikles av andre på et senere tidspunkt. (Dokumentasjon etc.)

Databaselagring:

- Når fingeravtrykket lagres i databasen, skal også fingernummerering være med (1-10).
- Det skal også lagres informasjon om hvilken av sensorene som har blitt brukt til å lagre fingeravtrykket.

17.02.2015

Referat fra oppdragsgivermøte 17.02.15:
Til stede: Alle
Referent: Anna og Synne

Tilbakemeldinger på design:

- Det er ønskelig at det plasseres en info-tab ved siden av Register+Compare. Denne skal kun inneholde statisk informasjon om fingeravtrykkskanning.

- For register: Ikke bruk "Check validity", men heller "Check validity of username". **For register:**
- Gjør det klarere at man registrerer noe i databasen, men også har mulighet til å gå rett til Compare og utføre live scan. En god løsning kan være at man starter på Compare-siden, og heller har mulighet til å gå til Register om man ønsker å registrere.
- For register vil det også være hensiktsmessig med en knapp Fingerprint OK?"/"Check fingerprintsom sjekker kvaiteten på fingeravtrykket (dette finnes det allerede programvare for). Brukeren skal da få tilbakemelding på avtrykkets kvalitet (dårlig/normal/-bra), og får mulighet til å velge om man vil lagre et dårig fingeravtrkk likevel, eller ta et nytt (store/retake"). Kvalitetssjekken gir verdier fra 1 til 5 hvor 1 er den beste og 5 dårligst. Oversette tallene til forståelige tilbakemeldinger. **For compare:**

- Man viser hele tiden fingeravtrykket på venstre side. Den delen skal ikke gjøres inaktiv. Til høyre på skjermen skal man ikke ha en kopi av sitt eget fingeravtrykk. Kun topp 4-score med bilder, og helst plassert under hverandre (liste), for å bedre illustrere hvilke som har høyest match. Ved siden av hvert bilde skal det plasseres tilhørende info om fingeravtrykkets eier, altså brukernavn, hvilken skanner som har blitt brukt, fingernummer og score. Det er ønskelig å "oversettefingeravtrykknr til feks left thumb".
- Det er også ønskelig at det plasseres en knapp under live scan Find measure points"/"Analysesom viser punkter på fingeravtrykket som sammenlignes. Deretter kan brukeren velge å sammenligne med database.
- Det er også ønskelig med en knapp der man får opp informasjon om fingeravtrykkets kvalitet. Dette kan evt. komme opp som en popup i forbindelse med at man velger Show measure pointsel. Når sammenligningen er ferdig vises topp 4-liste eller topp-4 boks (2x2) og bruker kan klikke på et fingeravtrykk match for å "zoome"ut ett fingeravtrykk. Enten som et popup som dekker akkurat sammenligningsbolken eller i selve bolken (AN: er dette mulig?) Ønsker å ha en hjem-knapp perm i vinduet.
- Ferdig-knapp i compare-siden isteden for en popup. (vil da tømmesiden) **Tilbakemeldinger på database:**

- Sikker sletting av database: Ved oppstart skal det sjekkes om det finnes en midlertidig database. Hvis denne finnes, skal denne slettes.
- Gjestedatabasen skal være lik som den permanente databasen.
- Hver entry i databasen skal ha en unik ID. Slik at bruker har en ID - finger sin ID - og scanner sin: eks 01-01-01 (bruker 1, sin finger1 og på scanner 1)
- Informasjon som også bør lagres i databasen (i tillegg til det vi alt har): kjønn og fødselsdato. MEN må dobbeltsjekkes med NBL om de er villige til dette.
- Info om fingeravtrykkets kvalitet trenger ikke å lagres i databasen.
- Ett fingeravtrykk fra samme finger, fra samme bruker, fra samme skanner skal kunne lagres i databasen. Forsøkes det å registrere et nytt fingeravtrykk på samme bruker, med en allerede eksisterende finger og samme skanner, vil det som er lagret bli overskrevet, eller evt. kan man lagre fingeravtrykket med best kvalitet (altså overskrive hvis det nye er av bedre kvalitet, eller beholde det gamle, hvis det er av best kvalitet.)

Tilbakemeldinger på konvertering fra C Sharp til Java: Det finnes allerede en løsning i Java sdk som konverterer C Sharp til Java. Dette kan vi se nærmere på med Guaquiang Li når PCen er klar. (Altså ingen konvertering nødvendig for denne modulen.)

05.03.2015

Sprint Review Update
Demo
Viste en demo av hvor langt vi har kommet.

Compare:

- Fingerprint bildet må være større og sentrert. MEN pass på å ikke ødelegg forholdet i bildet (strekker det for langt eller bredt)
- Når man klikker på fingerprint match - så fyll heller boksen til venstre enn å ha en popup. Her skal man få opp minutiae på begge. Slik at man kan se forholdet dem imellom. Og så må man kunne klikke frem og tilbake.

- Categories knappen. Hvit skrift når den klikkes på fingerprint. (nå er den svart og synes nesten ikke)
- Compare - Stor C

Når man skanner en finger
- Sjekker kvalietet.
- Sjekker om det finnes en fra før
- Sjekker hvem som har best kvalitet.
- Og den som har høyest vinner.

De andre modulene
- Finger Vein
- Ganske lik fingerprint.
- Ha en database med bilder av fingerårer. (mye færre her enn med fingeravtrykk)
- Ikke aktuelt å legge inn gjester
- Ha en Compare-del hvor man viser hvordan fingerårene ser ut. (dette vises live med hjelp av den blå boksen"og et program) Så må man klikke for å ta et bilde. Dette lagres i en mappe. Så sammenligne med databasen. (usikkert med tall verdier)
- Utstyret som brukes til å skanner fingerårer brukes for tiden av en annen student og er opptatt. Må vente med denne modulen.

Keystroke
- En del anderledes enn de to andre. MEN ikke noe linsensiert programvare. (YAYYY :P) Har en database med få brukere og noen få ord. (orddatabase)
- De som legger inn her, må skrive ordet flere ganger. Dette produserer en graf en gjennomsnitten med litt svinn. (husk rød linje i grafen)
- Ikke aktuelt å legge inn gjester

I Compare delen:
- Ber bruker om å skrive ett ord fra orddatabasen en gang. Eks Welcome45
- Analyseprogrammet tar opp når man trykker ned, slipper knappen og hvor lang tid det tar imellom knappene. Og så sammenligne denne verdien med databasen og viser det nærmeste treffet (husk blå strek i graf)
- Ved match så vises det en graf med 2 røde streker som er gjennomsnittet til match-personen i databasen, og en blå strek som er gjestens instastede verdier. **Annet:**
- Patrick skal sjekke om masterstudentene er ferdige med fingerårer testen, og den blå boksen". Og så komme med tilbakemeldinger om dette.
- Det er fint om vi også tester på den store skjermen. Slik at de kan veksle mellom disse to.
- Aktuelt (men ikke kritisk) å sjekke om programmet kan kjøres på nyere OS.
- Det er ikke nødvendig på ha noe ekstra sikkerhetstiltak enn det som er nå. (lås og tilganskontroll på døra, login på pc)
- begynnelsen av mai kommer det en utvekslingsstudent som kanskje skal jobbe videre på vårt program. (mulig iris med webcam) . Patrick vil at vi er villige til å svare på spørsmål og hjelpe til hvis studenten trenger hjelp til noe ang programmet vårt.

24.03.2015

Sprint update

Vi gikk gjennom utkast på design til keystroke med Patrick. Han hadde noen innspill, men ellers små endringer:
Det regnes ikke "Matchscore i keylogging. Men Distance. Jo mindre tall her, jo mer treff er det.
Angående enrollment: Kan ha det samme vindu som "Capture keystroke", men ha en litt annen tekst og en counter. Denne counteren skal telle antall godkjente ord som sendes til databasen. "OKknappen må være passiv, så lenge det er mindre enn 10. En løsning at man registerer ordene når man trykker enter"istedenfor å måtte trykke på en ekstra knapp.
Ikke vis ordet når det tastes inn, men gjerne stjerner.
Patrick skal velge ut ett viss antall ord som skal kunne legges i db. Og så vil han ha muligheten til å legge til nye. Løses med at vi koder lett nok til at en kan legge til ord uten store problemer. (f.eks array med alle ordene)

Andre ting
Det registeres tallverdier når en bruker skriver inn ordene. Dette legges i databasen. Utifra disse tallene så må vi kunne lage en graf.
Merk at vi må kunne lage tre streker (se GUI-utkast) to røde (eller en annen felles farge) på databasetall og en blå strek på den som brukeren har tastet inn.

Patrick nevnte at det var utfordringer med java angående keylogging. -problemer med nøyaktigheten Derfor er kode skrevet i C Sharp Avtalte at vi skulle ha en demo for Patrick og Guoqiang når de kan av fingeravtrykkmodulen. Etter påske.

C.2 Veiledningsmøter

Her følger referater fra møter med veileder Tom Røise.

13.01.2015

- Det er viktig å avklare tidlig med Patrick om hvor dypt vi skal gjøre programmet.
- Skal det være veldig dypt på ett eller to funksjonaliteter eller veldig bredt og litt på mange?
- Avklare prioriteringer: Hva MÅ på plass, hva KAN være på plass, og hva kan v SUPPLIMERE etterpå?
- Fokus på å lage generelle og grunnleggende deler som lett kan bygges på for å utvide programmet. Legge opp til modulbasert.
- *Kan ha papir mockup —> gå over til verktøy —-> kode utseende. Det er viktig å vise at vi kan benytte oss av verktøy.
- Anbefales å finne inspirasjon fra andre ligende prosjekter / programvare. Se på hva kan vi bruke, hva funker? ikke?
- Diskutering og analyse noteres hele veien. Med kilder.
- Avklare hvem målgruppen er for programmet. Er det ansatte ved BioLaben eller hvermannsen?
- Avklare om vi skal / kan ta med bakgrunnsinfo. Hvordan utføres analysen, hva skjer osv. Litt opplæring/ kunnskap om temane.
- I/O. BioLaben har benyttet seg mye av I/O-kommunikasjon, er dette noe vi kan gjenbruke?
- Når vi skriver rapporten: Få frem hva vi fikk til. Hva er vi stolte av å fått till. Ikke fokuser så mye på det vi ikke fikk til, forklar det heller veldig kort.
- Avklare med Patrick ang teknologi. Enheter (pc, mobil, nettbrett??)
- Bruk vedlegg aktivt. Eks: Use Case —> Plukk ut de viktigste i rapporten, legg resten som vedlegg.
- Avklar med Patrick- Om programmet skal fungere i et lukket miljø, om det skal være tynne/tykke klienter (bare EN klient?)
- Få frem gode løsninger i rapporten. (ang kode) Kommenter hva som skjer og hvorfor.
- Bland gjerne sammen systemutviklingsmodeller (eks RUP/Scrum)
- Scrum vil kreve mer deltakelse fra BioLaben (product owner).
- Hvis vi velger scrum så prøv å følg de prinsippene som er der.
- Anbefales å kategorisere timer
- Endelig kravspesifikasjon i begynnelsen av rapporten, og hvis det dukker opp ting underveis så ta den i disusjonen. Ikke jobb med flere forskjellige kravspesisfiksjoner.
- InforSec —> Ettersom vi studerer inforsec så forventes det mer fra oss når det gjelder informasjonssikkerhet. Ta med sikkerhet fra bunn av når det gjelder kode. Forventes også mer infosec i diskusjonen. Ha risikoanlayse hele veien.
- Feks. Ha en infosec-vurdering for hver sprint.

20.01.2015

Referent: Anna Kaardal
Til stede: Anna Kaardal, Joachim Hansen.
Selve prosjektarbeidet: Det kan være en ide å legge inn tider i gant, som viser når vi skal snakke med eksperter. Disse legges inn som aktiviteter utenom SCRUM-sprintene.

GUI: Noen mener at designet bør ligge en sprint foran resten av utviklingen. (One Sprint Ahead UX.) Fikk mail av Tom, med anbefaling om å google dette, samt:

Sikkerhetshensyn: Vi bør diskutere med Patrick hva som blir konsekvensene, f.eks. om noen kan hente opp igjen det vi har slettet fra disk. Finne ut om det blir nødvendig å overskrive hele disken for å hindre dette. Det er viktig at vi gjør bevisste valg, ikke bare for å briefe med at "vi kan det". Det skal være et behov.

Risikoanalyse i Scrum: Vi bør asjourføre nye risikoer som dukker opp. (F.eks. hver 3.sprint). Det kan bli over-kill med fullstendig risikoanalyse hver gang. Revider heller. Vi bør sette grenser på hvor stor en user story kan være før den må brytes ned til user task (f.eks. 5 dagsverk).

Vi bør estimere tidsbruk for hver task i timer og dagsverk. Vi bør legge oppgavene i prioritert rekkefølge, der de med høyest prioritet ligger øverst.

Forslag til forbedringer på rapporten: For effektmål må vi tenke mer effekt. Det er nå for stort fokus på funksjonelle krav. Hva er høgskolens ønsker på sikt? -Medieoppslag? Bevisstgjøring? Rammer: Tom og Patric hører ikke til her. De skulle heller vært under prosjektorganisering. Her kan vi heller ha med hvilket utstyr vi har tilgang på, f.eks. PC'en vi har fått tildelt og som vi skal jobbe på.

Fagområde: Bør ha mer om biometri her. Systemutvikling bør ikke plasseres her. Noen steder bruker vi div. typer hardware"ol. Vi må være mer spesifikke. Under argumentasjon bruker vi forferdelig mange referenser fra Sommerville, finn en bedre måte å referere her.

Plan for statusmøter: Er dette egentlig nødvendig? Har vi ikke sagt dette tidligere? For figuren: Referer heller til section, ettersom figuren befinner seg flere sider frem. Bør forklare bedre HVA vi skal bruke GIT til. Altså at vi både skal bruke den til kildekode og admin.dokumenter, osv.

Risikoanalyse: Kan fokusere litt mer på biometrimiljøet, ikke bare innad i gruppen.

10.02.2015

statusrapport (bør lages 2-3 ganger og sendes ut ved en milepæl i prosessen) eksempler på dette ligger på nettet.

Vi må være mer pågående (være litt utålmodig for å mase om løsninger som vi trenger fra NBL) Må vurdere om vi trenger backup løsninger for database (under utvikling)

Bør lage et oversikt kart om hva slags kall skal gjøres i hvilke programmer, hvilke programmer trenger hvilke tilpassninger ...

Bør ikke bare tenke happy day scenarioer når vi tester teknologier/løsninger

Bør ikke vente med å teste teknologier/løsninger selv om ikke alt er på plass enda

Kan vurdere om hva skjer om strømmen går, hvordan blir det med sletting av sensitive filer da? Skal f.eks. det være en bakgrunn prosess som kjøres ved boot som sletter alle "middeltidgregistrerte database tabeller

Trenger ikke å lage de ultimate løsningene når det gjelder GUI, det vi har gjort så langt er bra nok (VISIO > Photoshop).

Bør ha et utvalg av mennesker fra ulike bakgrunner som skal teste GUI, denne prosessen bør foregå på en systematisk måte f.eks. intervju (men trenger ikke å dekke like mye som store vanlige prosesser)

24.02.2015

Veiledningstime m/Tom
Tilstede: Alle mann
Stikkord Skal ha info tekst. Patrick ville ha noe statisk (i første omgang) Vi må vurdere om det er kanskje best å ha en editor el lign for å gjøre det mer dynamisk.

Linsenser - dokumenter? Finnes det noe dokumentasjon som vi kan begynne å se på? Forberede oss på den kommende c sharp-koden.

Begynne å se litt på hvordan de andre modulene skal løses. Ikke gå i detaljer, men se om disse kan benyttes til stamdesignet.

Prøve å lage noen tricky komponenter -> lage testløsninger. Databasen Sikre og skjerme databasen. Det er mye sensitiv informasjon som må sørge for å beskytte. Det er kreves mer av oss ettersom vi er infosec.

Autentisering?
SQL-injection
Kryptering
Sensitivt informasjon = må sikres bedre.

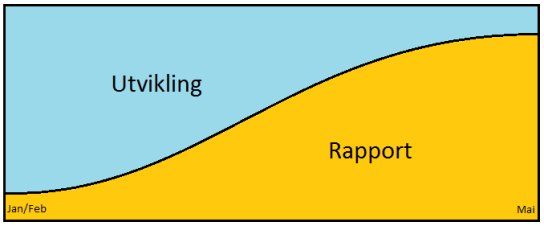
03.03.2015

Angående deling av arbeidsoppgaver
Det er et fokus at alle skal lære, ikke være mest mulig effektive. Det å dele fullstendig slik at det blir en som koder mesteparten og de andre skriver mesteparten av koden er litt unødvendig. Men det er fullt mulig å ha en som har hovedansvaret. Det er viktig at alle går inn i hverandres kode og lærer av det. Se på hva som kan gjenbrukes, hva kan endres litt på en annen modul? Se på hva som kan overlappes.

Testing
Anbefales å ha review av kode og testing. Kan gjøre dette med hverandres kode. 1 går gjennom review og testing på en del, en annen gjør det på en annen. Bestem hvordan vi skal gjennomføre Reviews. Felles, alene en og en, fordele oss i mellom osv. Test grenseverdiene innenfor og rett utenfor. Test alltid null verdier. Både systematisk og automatisk (jUnit). Se på svakheter og styrker på måten testingen er utført. White box vs black box. Ta dette med i rapporten.
Hvordan jobbe uten PC'en. Kan vi utvikle mest mulig på våre egne PC'er og så overføre det til PC'en i A151? Unngå kø og venting på å få bruke PC'en i A151. Backup på PC? Burde ha en løsning for dette.

Rapport Sette 2 datoer som vi har som deadline med rapportutkast. Ha en gjennomgang oss i mellom, og så en gjennomgang med Tom også. Tom må få vite datoen i godt tid slik at han kan få litt tid på å lese gjennom.

Antall moduler Ikke noe fasitsvar hvor mange moduler som er best. Kvalitet kontra kvantitet. Se de på de 3 MÅ modulene. De bør være hovedmålet, men når vi nærmer oss å bli ferdig så planlegg litt i forkant neste modul. (Om det er tid)
Det er viktig at det vi leverer har kvalitet, robusthet, testing og et er godt produkt. Dette bør være fokuset. Men må også ikke bli for pirketete på en modul. En balansegang. God grunnstruktur/grunnbase vil gjøre det lettere å bygge videre på neste moduler. Figuren under viser en god fordeling av utvikling og rapportskriving. Begynner med lite rapportskriving, og så halvveis er det ca 50/50 på hver og til slutt mesteparten rapportskriving. I den siste delen er det minimalt med utvikling, og mest mindre ting fremfor større oppgaver. Se illustrasjon under.



Rapport i forhold til utvikling

10.03.2015

Vi begynner med de andre modulene. Tar for oss GUI, user stories og får programvare som vi trenger.

Tester
Angående tester. Vi bør ta en vurdering på hva som skal testes. Om det skal være alt? Alt automatisert osv. Erfaringsmessig mener Tom at målet om å teste alt er altfor stor oppgave og vil kreve mye jobbing. Ta heler å konsentrer på noen viktige funksjoner eller klasser. Finne en balansegang mellom automatiske/systematiske og tidsbruk. Rekker ikke å være perfekte.

Databasen Selv om utgangspunktet og teoridesignet ikke nødvendigvis er identisk med den reelle løsningen så er det greit, så lenge vi kommenterer dette i rapporten. F.eks angående at vi har gått vekk fra normalisering. Vis idealet (utgangspunktet) og fortell hvordan vi kom frem til den faktiske løsningen. Det er verre å ikke kommentere det, og en sensor ser at databasen ikke er normalisert. Han kan tolke det da som dårlig design fremfor en tilpassning til oppdragsgiver ønsker som i vårt tilfelle. Brukertester?
Tom spurte om vi skulle ha noen brukertester på det faktiske programmet.Og anbefalte at vi tar en vurdering om dette. Det trenger ikke å være snakk om langvarige tester. F.eks 2-3 studenter som går gjennom en modul. En kort halvtime. Noter det som de synes er rart eller sliter med, og se om vi kan fikse det. (ala noe likt det vi gjorde med GUI-brukertest på papir)

Presentasjon i Juni
Se for dere at vi kan presentere /demostrere programmet vårt på presentasjonen. (om ikke hele /alle moduene) Litt kulere enn å vise på ppt hva som vi har gjort osv. Kanskje vi kan det opp på film? Moduler vs rapporten

Trenger ikke å være redd for å levere noe som ikke er helt ferdig. Det er ikke noe problem å ha noen påstartede moduler tilslutt i tillegg til de som er helt ferdige. F.eks GUI forslag til de neste modulene eller lignende. Må ikke bare levere det som er fullstendig ferdig. Tenkt på den gradvise utviklingen av rapport og programutvikling. Etter påske skal man ligge ca 50/50. Og så gradvis bli mer og mer rapport. Det er litt rart at man slutter å utvikle helt rett etter påske. Hvis vi jobber med siste modul, og ser at vi ikke blir ferdig så ta det med i rapporten. Og sett det under "Videreutviklingkapittel. Kanskje en student kan få det i sommerjobb å fullføre.

17.03.2015

Feilsituasjoner
Vi bør se på feilsituasjoner. Se på hvor dårligting kan gå, og fikse slik at skaden minskes eller unngås.
Sett opp en oversikt over de vi kan se, og del dem i hva vårt program skal kunne håndtere og hva de eksterne programme håndterer.
Ikke mulig å kunne ta tak i alt, men ta en vurdering over hvilke som er viktigst og hvorfor. Vis i rapporten at vi har sett flere feilsituasjoner, men har valgt de og de. Vis at vi har tatt en vurdering. Vær bevisste over de andre situasjonene. Hva er feilsituasjonen, hvordan kan den løses, skal vi løse den eller ikke? Som alt annet, mest fokus på kvalitet fremfor kvantitet, Lever god kvaliet på det som vi velger å fikse på.

Presentasjon Finn ut hvordan vi ønsker å demostrere. En live demo på presentasjonen? Eller ett opptak. TA med PC'en på forelseningssalen og kjør den der? Avklar med Patrick om vi kan vise til sensor. Skal han få lov til å få en egen demo på rommet?

Rapporten Bruk screenshot av GUI av programmet. Hvordan det endte opp. Fint å sammenligne med utkastene. Ha grafiske modeller og kart over programmet. Kjedelig med lange lister og regneark. Gjør det visuelt og lettlest.

User stories: Hvordan vise dette på en bedre måte? Use Case er en løsning. Trekk frem noen av modulene og vis gjennom dette. Trekk frem noen av de viktigste user stories og vis utviklingen fra første start til siste implementasjon med kode. (ta med møter med Patrcik, GUI-skisser, kode, og Screenshot)
Design vs implementasjon
Design er mer overordet
Mens implementasjon er mer detaljert. Ikke så ofte at man tar med ALT, men plukker ut det beste. F.eks en fin løsning på kodeutfordringer, eller design løsning osv. Her er det vanlig å vise til kodesnutter, og detaljerte figurer av struktur osv. Anbefalte igjen å se på andre rapporter, spes Belt og Drismo.

DEMO Det blir demo av fingeravtrykkmodulen neste tirsdag 24.04.15 med Tom (og evt Patrick) på A151.

C.3 Ekspertmøter

Her følger referater fra møter med eksperter innenfor forskjellige felter som vi har snakket med i løpet av prosjektet.

Møte med Ole Wattne

Møte med Ole Wattne 10.02.15
Til stede: Anna Kaardal og Synne Gran Østern

Under dette møtet startet vi med å presentere vi oppgaven vår, og diskuterte litt rundt dette, og fikk input på hvordan designprosessen bør foregå. Vi tok også med oss designskisser for fingeravtrykksscanning og fikk mange nyttige tilbakemeldinger på disse.

Designprosessen: Man starter med å kartlegge brukerens brukerens behov til funksjonalitet, lager brukerscenarioer og skriver ned steg for steg hvordan disse vil forløpe. Brukerhistorier: Hva er et vanlig hendelsesløp for en bruker. Fokus på Erna, ikke på NBL.
AN: Vi velger å forenkle denne prosessen betraktelig, men tar med brukertesting og benytter oss av brukerhistorier som en måte å finne hvordan vi skal utforme GUI.
Brukertesting: Brukertesting bør foregå så tidlig som mulig! Det kan være aktuelt å starte med å gi testpersonene ark med design og be om tilbakemeldinger på dette. Er det forståelig for utenforstående? Etter hvert kan man fortsette med interaktive oppgaver.
Farger og utforming: Vi bør vente med dette til slutt, og heller holde oss nøytrale i starten. Det er også viktig å huske på stor nok kontrast mellom bakgrunn og forgrunn, samt sørge for at skrift-og ikonsørrelse er tilstrekkelig. Ikoner er god støtte til tekst, men ytterst sjeldent noe bra alene. Bruk det der det passer. Vurdere i hvor stor del vi skal rette oss etter universiell utforming (AN: nja... Gidder vi det da?)
Arkitektur: Vi bør sette opp en arkitektur som viser hvilke funksjoner som er med i programmet, de forskjellige stegene, og hvordan man kommer seg gjennom programmet steg for steg (programflyt). Diagrammer.
Tilbakemeldinger på våre skisser: En vesentlig ting som er viktig å tenke på er at utenforstående kan være skeptiske med å gi fra seg biometri. Det bør kommuniseres klart hvorfor de skal gjøre dette. Modules (øverst i venstre hjørne) - dette er et lite forklarende stammespråkord. Vi må finne et ord som bedre representerer formålet med denne menyen. I tillegg kan "modules"i hovedvindu droppes ettersom det er vist i hovedbilde hvilken moduler man kan velge. Choose scanner: Gjør det vanskelig for brukeren å forstå hva som skal gjøres. Man bør vise første scanneralternativ i drop down menyen før brukeren har klikket på noe. Det kan også være en ide å gi scannerene navn/ nr. Og ev. klistre lapper på hver scanner. Capture fingerprint: Brukeren bør få en bekreftelse på at alt som skal fylles inn er ok, og alt er klart før vedkommende trykker capture fingerprint. Bekreftelse på de andre brukerinteraksjonene er noe som vi må ha med på alt. Bekreftelse at scanning er ok, ingen feil har skjedd. (men også feilmelding hvis det skulle skje)
Compare:
Finn et mer forklarende ord på knappene. Hva sammenlignes? - "Compare scan from database". Området der sammenligningen foregår bør være inaktivt (grået ut) frem til brukeren velger å sammenligne sitt fingeravtrykk med databasen. Det bør også komme en tydelig melding når sammenligningen er fullført og resultatet er klart, og så hvordan brukeren kommer videre. Kan det være aktuelt å sende brukeren tilbake til Welcome screen? Feks. med en Ferdig-notafiksjon og så en lenke til "hjemlike ved? Compare: Mer stykket opp de forskjellige funskjonalitetene (mer steg for steg) og en ting om gangen. F.eks. live sanningfor seg selv, når den er ferdig så kan man trykke på sammenlign -> sammenligner (alt det andre blir passivt) osv osv.

Faner:
Vi bør ha samme farge på bakgrunnen som på aktiv fane. Inaktiv fane kan f.eks. ha en mørkere farge. Skill tydelig ut modules, så den ikke ser ut som en fane. Det blir rotete.

Diverse: Husk at det er viktigere med brukervennlighet og en selvforklarende , fin sekvens av handlinger. Det må være rom for feiltoleranse. Man kan ikke gå ut fra at brukeren klarer alt riktig på første forsøk. Man må også ta hensyn til brukere med utfordringer. Gi beskjeder som ikke er for tekniske, og gi brukeren muligheten for å gå tilbake til forrige steg og rette opp det som ble feil. Ventetid: Programmet skal kjøre i et lukket"miljø, og kravene til at brukeren skal "aktiviseres"mens PCen foretar beregninger er ikke så viktig. Likevel må vi ikke la brukeren lure på om maskina har "hengt segfordi responsen ikke kommer øyeblikkelig. For kortere ventetider holder det med f.eks. et snurrende tegn. For lengre ventetid vil det beste være en progressbar som viser prosentvis framgang.

D Brukertester

I testingen av designet har vi benyttet skissene som finnes i K.

Resultater av testing av skisser

Brukertest 1:
Jente, 16. vgs- Helse og sosial. Vet ikke hva biometri er.

F-Compare(tom):
Ser først på «Compare with database», legger ikke merke til «Capture fingerprint» eller «Analysis of fingerprint» Foreslår å bruke en annen farge eller teksttype på disse knappene for at de skal skille seg mer ut.

F-Compare(resultat):
Skjønner at den med grønn er den som matcher, men hva betyr de med rødt? Det hadde vært greit med en tekst over som forklarer at dette er topp 4-match.

F-Register:
Det er vanskelig å forstå hva som menes med guest. Burde bruke et annet ord her, som forklarer at dette er for midlertidig lagring. Ikke helt forberedt på det som skjer i Capture fingerprint, ha en bedre forklaring på dette.

F-Register-Scan session:
(8,9) Det er mer naturlig å bare klikke på en ny finger enn å velge select again. (10) Unaturlig å ha capture fingerprint øverst. Legger ikke merke til den. Hadde vært bedre å ha den nederst på bildet.

Etter testen:
Synes jevnt over at dette er et oversiktelig og forståelig program, men med enkelte unntak (se over).

Brukertest 2:
Kvinne, 56. Ungdomsskolelærer. Vet ikke hva biometri er.

F-Compare(tom):
Legger ikke merke til capture fingerprint, men ser rett på analysis av fingerprint. Kan være en ide å også plassere denne knappen nede sammen med analysis.

F-Compare(resultat):
Det er forståelig at den øverste er en match, trenger ikke mer beskrivelse her.

F-Register:
Spesifiser at det er gyldighet av brukernavn som sjekkes. Forstår ikke hva som menes med Guest. Vær heller konsekvent, og bruk samme ord som i beskrivelsen over: temporarily.

F-Register-Scan session:
(8,9)Synes det er greit med select again, da forstår man at man faktisk har mulighet til å angre og heller velge en ny finger. (10) Ser heller ikke her capture fingerprint-knappen.

BrukerTest 3:
Mann, 27 år. IT-student. Grunnleggende biometri. (vet at vi har fingeravtrykk, iriser, men lite mer om dette)
Tallene er sidetall på arkene vi demonstrerte med.
1. Velkomstside
a. Bilde på fingerpint.
b. En hel knapp på kategoriene. (samle tekst og bilde)
c. Vise hva som velges.

2. F- Compare - tom
a. Mer bakgrunnsinfo – hva skjer her
b. Ha en «Scan» knapp som starter fingeravtrykklesningen
b.i. Eller noe infotekst om at nå kan brukern legge finger på skanneren.

3. F Compare – fingeravtrykk
a. Knappene «analysis of fingerprint» og «capture fingerprint» forsvinner
a.i. Også «Choose skanner»
a.ii. Tips om å samle alle knappene sammen på ett område.
a.iii. «Compare» er et dårlig navn fordi det sier at vi skal
a.iii.1. byttes med «match»?

4. F – Compare – analysis
a. Ønsker mer info om fingeravtrykket a.i. Få ut tall verdiene selv om de ikke gir mye betydning for hvermannsen

5. F – Compare – match 4
a. Prosent match er viktigst. Vis dette først og størst når det gjelder tekst.
b. Kanskje litt lite informasjon. Kanskje ha med tall verdier??? (evt på neste side)
c. Man får ikke inntrykk over at det har skjedd noe. Kanskje annerledes når det skjer live?
d. Om det er mulig, kanskje få noe info om hva gjør fingeravtrykkene forskjellige (kanskje best på neste side?)
e. Droppe «compare with database» når man har sammenlignet. Denne forsvinner.
f. Kanskje ha de to boksene mye større og samlet?

6. F – Compare – match en fokus
a. Her er det naturlig å ha de tingene som jeg nevnte over.
b. Men vil ha mer info / evt viser hva som treffer og hva som ikke treffer.

7. F – Register – Tom
a. Hvordan være anonym?
b. Guest er et rart ord her
c. Droppe username – heller ha en anonymous knapp
c.i. Og ha krav om firstname og lastname fremfor å ha username.
d. Register sier at jeg forbinner meg til noe, men det er ikke noe info om dette. Kanskje ha noe info om dette og hvordan data blir behandlet.
e. Ha save / start fingerprint scan fremfor capture. Kanskje enroll er et bedre ord (og profesjonelt) f. Ikon med diskett når det lagres? (kanskje mest relevant i scannesiden) 8, 9 ,10 og 11

F – Register – scan session
a. Unaturlig å ha hendene opp, snu dem heller.
b. Dropp «Select again»
c. Smart å ha fargevalg
d. Bruk denne når finger er skannet, eller hake eller noe lignende
e. Tungvint å måtte gå frem og tilbake for å skanne flere fingre
a.i. Heller ha en popup når finger er valgt og «Scan/Continue» er valgt
a.ii. Scan en finger og så kan velge «save» eller «Send to database»
a.iii. Så kan man gå tilbake til fingrene å velge en annen finger
a.iv. Og kanskje kunne klikke på en finger og se ditt fingeravtrykk som er tatt.

E Risikotabell

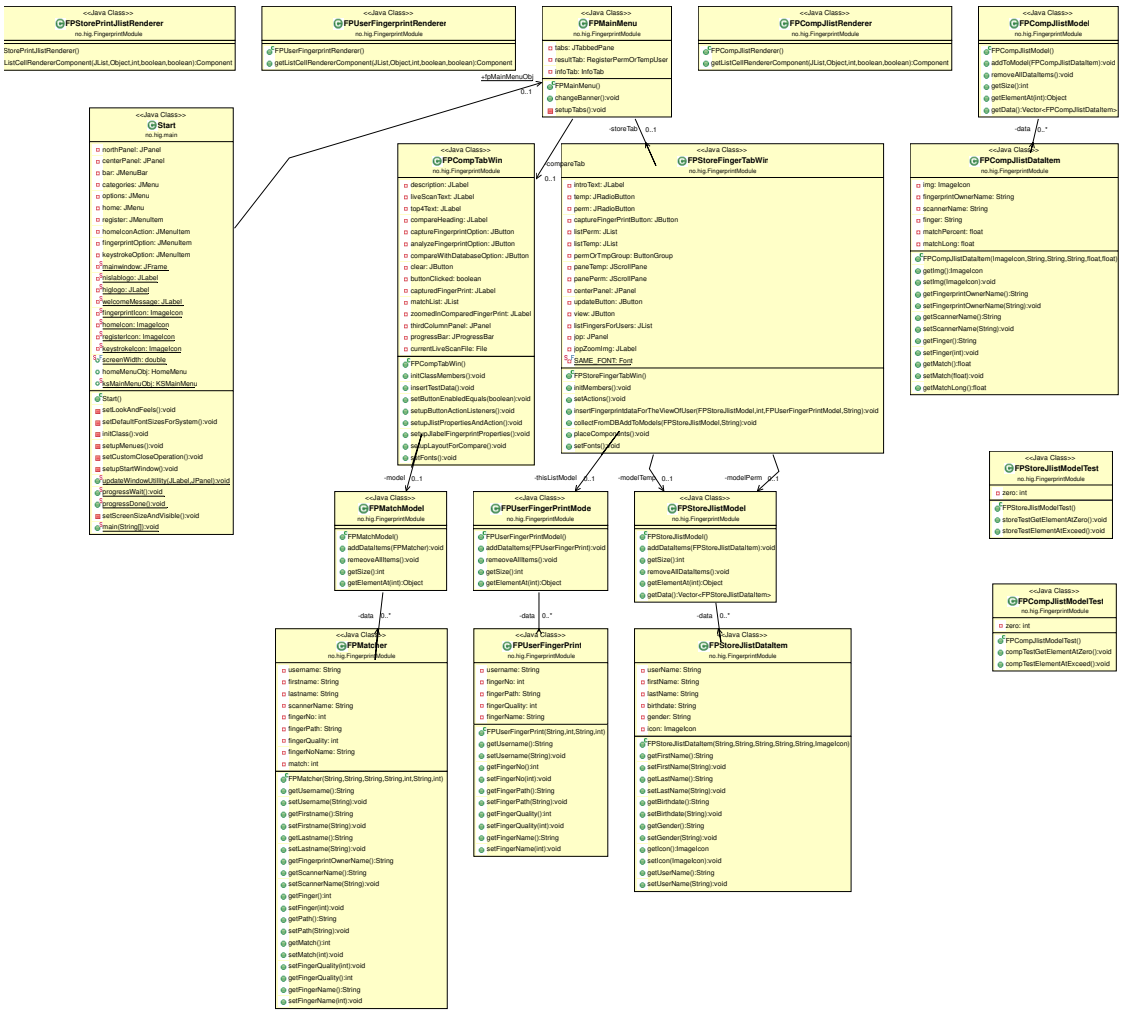
Hvis figuren er litt vanskelig å lese så er det lurt å benytte seg av zoome-funksjonaliteten i pdf-leseren og zoome inn på ca +175%.

Nr	Risiko	Sannsynlighet	Konsekvens	Kommentar	Tiltak
1.1	Tap av prosjektdokumenter	Sjelden	Middels	Reduksjon: på denne fordi vi har blitt flinkerer til å benytte sikrere lagringsløsninger, men ikke helt fjernet fordi det et litt varierernde hva vi lagrer og hvor.	Alle medlemmer av gruppa må bruke versjonskontroll aktivt + lagre viktige dokumenter på felles server.
1.2	Fravær ett eller flere gruppemedlem (sykdom eller skade)	Sjelden	Middels	Reduksjon: pga ikke ensidig fordeling av arb.oppg.	Gruppeeregler som dekker slike tilfeller, unngå at det blir en «single-point-of-failure» ved at en gr.mdl er den eneste som er ekspertise på et felt og de andre ikke kan ta over.
1.3	Får ikke tilgang til programmer og/databaser som er nødvendig	Kan skje	Middels	Reduksjon: pga at progutvikling er blitt fordelt på moduler. Hvis vi skulle henge opp i en , så kan vi jobbe med en annen paraellt.	Ha jevnlige faste møter med Patrick Bours og NBL, være klare på hva vi trenger fra dem.
1.4	Ingen eller liten deltakelse av gruppemedlem	Sjelden	Stor	Uendret. Fortsatt en stor konsekvens til tross for gode grupperegler	Grupperegler som dekker slike tilfeller. Klare krav om hva som skal gjøres. Andre mdl må aktivt si ifra om sine bekymringer.
1.5	Liten eller ingen kontakt med NBL	Sjelden	Middels	Reduksjon: pga jevnlige møter med oppdragsgiver og aktuelle personer.	Ha jevnlige faste møter med Patrick Bours og NBL, være klare på hva vi trenger fra dem. Vi må ta initiativet.
1.6	Tekniske problemer med lagring og versjonskontrollverktøy (Bitbucket, server hos IT-tjenesten)	Kan skje	Liten	Uendret, bruker godt brukte løsninger, men risikoen er fortsatt tilstedet.	Bruke velkjente og profesjonelle verktøy, og tillegg ha gode rutiner for bruk av disse.
1.7	Uenigheter som påvirker fremdrift og leveranse	Sjelden	Middels	Uendret	Benytte oss av gruppedynamikk for at det er 3 personer og at det vil uansett bli flertall såfremt ikke noen stemmer blankt.
1.8	For høye arbeids/prosjektkrav, Tidsfrister som overskrides	Svært vanlig	Middels	Økning. Pga forsinkelser med PC i begynnelsen så har vi blitt hengende litt etter. Vi er nå straks halvveis og jobber fortsatt med fingeravtrykkmodulen. Skal begynne på neste så fort som mulig	Tilpasse oss etter behov, gjøre vårt beste for å levere og hvis dette ikke er nok: få dette med i rapporten.
1.9	Mangel på kunnskaper	Kan skje	Liten	Reduksjon. Vi har fått god hjelp fra NBL + at vi er flinke til å tilegne oss kunnskap som trengs.	Forberede oss med å finne og lære oss disse nye områdene som vi ser vil bli aktuelle å kunne. Fordele spesialfelt mellom medl for å være mer effektive.
1.10	Sikkerhetshull som ikke oppdages	Kan skje	Middels	Økning: Pga forsinkelser og dårlig tid ser vi at testing og review har blitt utsatt til nå.	Følge kodestandarder, utføre flere forskjellige tester.
2.11	Sikkerhetssvakheter med WinXP	Svært vanlig	Stor	WinXP er en utfaset OS fra Microsoft, og er veldig sårbar mot trusler, spesielt fra internett	IKKE koble til internett. Bruke minnepinner for å flytte over filer ol. Patrick er positiv til at vi tilpasser til nyere versjoner også.
2.12	Tap av databasene	Sjelden	Stor	Databasen er ganske sensitiv pga biometriske data + kobling til personer. Det vil medføre store konsekvenser hvis denne kommer i hendene til uvedkommende.	Sikre databasen tilfredsstillende etter trusselbildet og oppdragsgiver (eier) ønsker og behov.
2.13	Krasj i programmet	Kan skje	Middels	Det er vist seg å være utfordrende å kjøre eksterne programmer samtidig med vårt, og demoPC hender krasjer en gang iblant.	Sikre på best mulig måte at programmet ikke krasjer mhja løsninger for eksempel pauser i kodekjøring osv.
2.14	Manglende sikkerhetstiltak pga dårlig kapasitet på PC	Kan skje	Middels	DemoPC'en er av en eldre art og er ikke i stand til å håndtere ekstra sikringstiltak som kryptering osv.	Risikoeier (oppdragsgiver) har akseptert denne risikoen, og synes det holder med adgangskontroll på dør og passord på innlogging.
2.15	Database ikke blir slettet som det skal	Kan skje	Stor	Det kan skje feil i kjøring av program / kodefeil som fører til at dne imidlertidige databasen ikke blir slettet som den skal.	Ha en "dobbelsjekk" ved oppstart av programmet som sletter hvis databasen finnes.
2.16	Gjester lagrer i den permamente databasen	Kan skje	Stor	Når en bruker skal registere seg, så må h*n velge mellom den midlertidigie og den permamente databasen. Hvis en gjest klikker feil, og lagrer seg selv i den permamente databasen, så er det vanskelig å fjerne denne personen igjen.	For å fjerne brukern, så må man hardkode en SQL-spørring som sletter denne spesefikke personen. Det er ikke laget en funksjon for dette, og det er kun en funksjon som sletter hele tabellen på plass. Ved en slik situasjon så må eier av BioDemo vurdere om tabellene skal slettes eller om personen skal være i databasen. Merk at det er snakk om flere steg av godkjenning før fingeravtrykk blir lagret.

Figur E.3: Siste versjon av risikotabellen

F UML for fingeravtrykkmodul

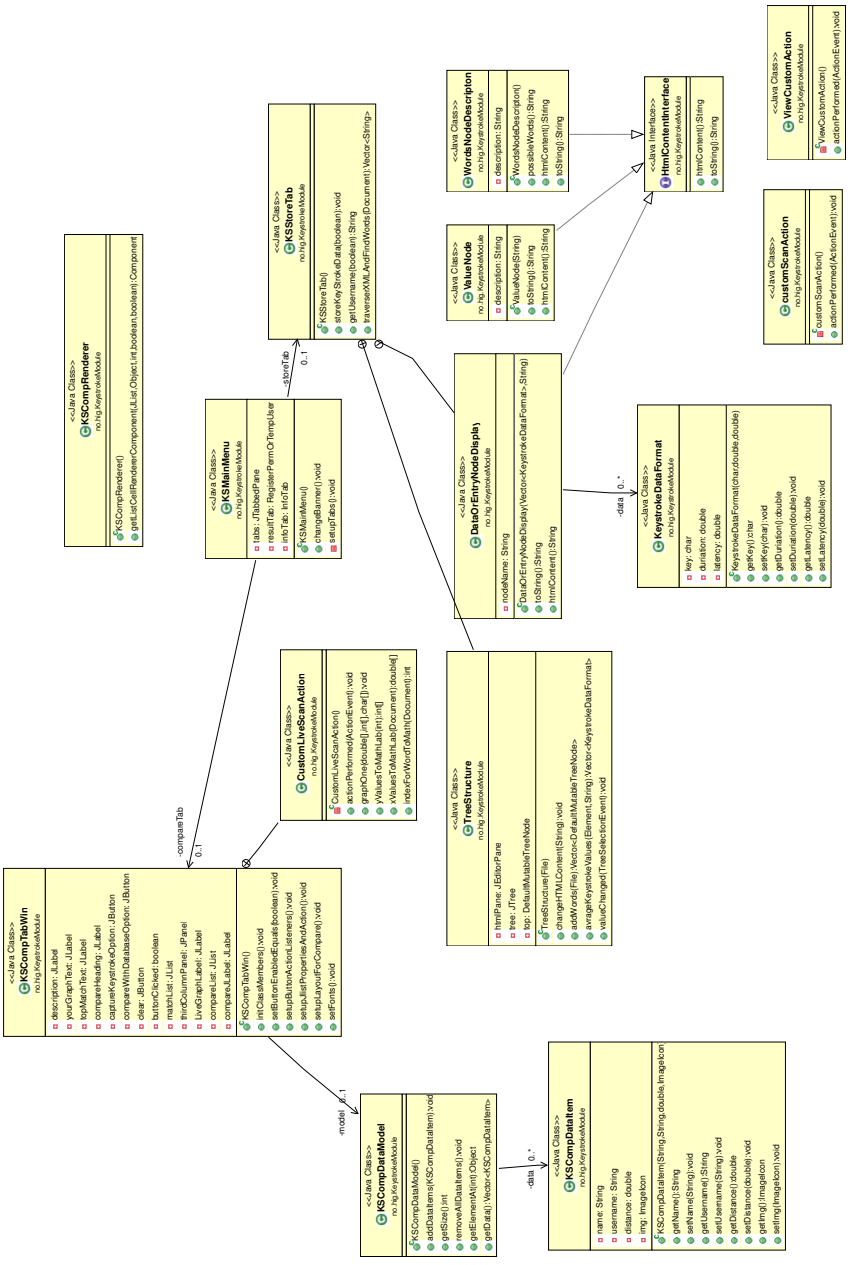
Hvis figuren er litt vanskelig å lese så er det lurt å benytte seg av zoome-funksjonaliteten i pdf-leseren og zoome inn på ca +200%.



Figur F.4: Fullstendig UML av klasser, funksjoner og variabler knyttet til fingeravtrykk-modulen

G UML for tastaturgjenkjenningsmodul

Hvis figuren er litt vanskelig å lese så er det lurt å benytte seg av zoome-funksjonaliteten i pdf-leseren og zoome inn på ca +200%.



Figur G.5: Fullstendig UML av klasser, funksjoner og variabler knyttet til tastaturgjenkjenningsmodulen

H UML for C# programmene

Figur H.6: Det som står som konfidensielt har ikke vi utviklet. Diagram over alle C# programmene og deres klassediagram

I Gray box testing

Hvis figuren er litt vanskelig å lese så er det lurt å benytte seg av zoome-funksjonaliteten i pdf-leseren og zoome inn på ca +300%.

Test nummer	user story/ user task/ operasjonelle krav	filer/klasser innvolvert	Funksjoner innvolvert	Beskrivelse	Test data
1		/img/* /img/ikonCC/* no/hig/main/start.java no/hig/main/HomeMenu.java	Start : InitClass(), setupMenues(), setupStartWindow(), setScreenSizeAndVisible(), updateWIndowUtillity(), HomeMenu: ALT	Test for å sjekke om vi får: 1. Vist et hovedvindu 2. Vist meny 3. Vist banner med hig logo, beskjed (sentrert), og Nislab logo 4. Vist forside med navigering til ulike moduler 5. SetLookAndFeel har blitt satt til nimbus 6. Fargevalg er satt riktig 7. Ikoner er satt 8. Størrelsen på programmet skal ta hensyn til windows oppgavelinje	1.Test med ulike pcer med forskjellige skjerm størrelser. 2. endre orienteringen på oppgavelinja i windows
2		C:/Fingerprints/* (Reg no: 03699) no/hig/main/Start.java no/hig/main/UtilityClass.java database/DropTablesForGivenDatabase.java no/hig/FingerprintModule/FPCompTabWin.java derby.jar	Start : setCostomCloseOperation(), main(...), UtilityClass : removeLiveScanImages(...), deleteTempFiles(...), listFileRename(...), getFileExtension(...), DropTablesForGivenDatabase: ALT FPCompTabWin : setupButtonActionListeners() - clear : JButton	Sletting av middeltidige data dvs : 1. Bilder fra livescan (etter vi har trykket clear) 2. Bilder fra livescan ved oppstart av programmet 3. Bilder fra livescan ved normal avsluttning av programmet. 4. Bilder fra temp brukere ved oppstart av programmet 5. Bilder fra temp brukere ved normal avsluttning av programmet. 6. Data i temp databasen ved oppstart 7. Data i temp databasen ved normal avsluttning av programmet.	1. Klikk på clear knappen : klikk event 2.avslutt program ved å krysse det ut 3. avslutt programmet uten å krysse ut (CTRL + ALT+ DELETE) 4. Ha bilder lagret på C:/Fingerprints i ulike mapper og ulike filformater 5. ha data i databasen 6. ikke ha data i databasen
3		no/hig/main/RegisterPermOrTempUser.java JDatePicker Software/jdatepicker-1.3.4.jar database/Queries.java derby.jar	RegisterPermOrTempUser: ALT Queries : userExists(...) InsertUserData(...)	Registerere bruker i en av databasene: 1. Få lagt inn en ny bruker om man fyller inn det som kreves 2. Få beskjed om en eksiterende brukes finnes på en og samme brukernavn som jeg prøvde om dette er tilfelle. 3. Få lagt inn samme brukernavn over de to ulike databsene 4. Få lagret i perm 5. Få lagret i temp	1. Trykk på knappen for å submitte uten å ha fylt inn noe data 2. Fyll inn bare for brukernavn 3. Endre verdier 4. Fyll inn alt 5. Fyll inn bare noe 6. Fyll inn samme brukernavn som en eksisterende bruker 7. Fyll inn tall på tekstfeltene og spesialtegn 8. kombiner lowercase og uppercase
4		C:/Fingerprints/* (Reg no: 03699) no/hig/main/UtilityClass.java no/hig/FingerprintModule/FPCompTabWin.java jni4netFolderForFingerprint/* FingerprintFunctions.j4n.jar (class : FingerprintFunctionalities) jni4net.j-0.8.8.0.jar	FPCompTabWin : setupButtonActionListeners() (Bryr oss om action listeneres for: - clear : JButton - captureFingerprintOption : JButton - analyzeFingerprintOption : JButton) FingerprintFunctionalities : obtainSampleQualityByImage(...), markMinutiaeInSample(...) UtilityClass : lastFingerprint(...), listFileRename(...), getFileExtension(...), startScanSoftware(...),	livescan + analyze: 1. Kan starte scanne programmet riktig 2. Om vi avslutter uten å ta et bilde, så kan vi få siste bilde tatt som ligger på disk. Om det ikke ligger noe på disk skal brukeren få en melding om at må ta et bilde. 3.om brukeren har avsluttet programmet med å ta et bilde og dette bilde harr for dårlig kvalitet skal bli slettet, etter at det har blitt displayet for brukeren. 4. Om brukeren avslutter scan programmet og har tatt et bilde med god kvalitet, skal man må dette displayet og man skal kunne trykke på analyze for å kunne få opp menutia points på dette bilde 5. Når man avslutter scan programmet (livescan) skal den alltid gi oss siste bilde som ble tatt (ser på file creation time)	1. Start scanne programmet 2. start programmet avslutt uten å ta et bilde 3. Start programmet ta bilder fra en gitt sensor og avslutt. 4. Start programmet ta bilder fra flere sensorer 5. start programmet ta bilder med dårlig kvalitet (gi noe annet enn fingeravtrykket ditt) 6. start programmet uten at det ligger noe på disk, avslutt uten å ta et bilde og så trykk på analyze knappen
5		no/hig/FingerprintModule/FPMainMenu.java no/hig/FingerprintModule/FPCompTabWin.java no/hig/FingerprintModule/FPStoreFingerTabWin. java no/hig/main/HomeMenu.java no/hig/main/InfoTab.java no/hig/main/Start.java infoHTML/fingerprintInfo.html	Start: main(...), updateWindowUtility(...), setupMenues(), HomeMenu: ALT FPMainMenu : changeBanner()	Navigering uten å miste data - fingeravtrykkmodulen	1. Sjekk om hvordan det er å navigere med menu 2. Sjekk om du kan navigere gjennom velkomst vinduet 3. Hvis vi navigerer tilbake til start, jobber vi fortsatt med samme objekt?
6		no/hig/main/RegisterPermOrTempUser.java	RegisterPermOrTempUser: isNotAllowed(...), setActionListeners() -confirmButton : JButton	navne convensjon filer og database username er oppretthold?	1. Prøv uppercase og lowercase på unknown 2. prøv _ som brukernavn 3 prøv _ i starten, i midten og slutten av et brukernavn 4. Prøv unkown i et ord (aka unknowned)
7		no/hig/FingerprintModule/FPCompTabWin.java no/hig/main/UtilityClass.java database/Queries.java no/hig/main/Start.java no/hig/FingerprintModule/FPMatchModel.java jni4netFolderForFingerprint/* FingerprintFunctions.j4n.jar (class : FingerprintFunctionalities) jni4net.j-0.8.8.0.jar	FPCompTabWin : setupButtonActionListeners() (Bryr oss om action listeneres for: - clear : JButton - captureFingerprintOption : JButton - analyzeFingerprintOption : JButton) - CompareWithDatabaseOption : JButton FingerprintFunctionalities : obtainSampleQualityByImage(...), markMinutiaeInSample(...), comparisonBetweenTwoSamples(...), UtilityClass : lastFingerprint(...), listFileRename(...), getFileExtension(...), sortOnMatch(...), Queries : selectFingerItem(...) Start : progressWait(), progressDone(),	Sammenlign live scan med databasen: 1. Når du trykker sammenlign etter du har valgt et bilde fra scan image skal du få en beskjed om databasen er tom om dette er tilfelle. 2. Når du trykker på compare skal ikke den siste fila bli sammenlignet, med databasen, men imageIcon som allerede er satt fra liveScan 3: hvis du har trykket compare with database gjentatte ganger skal dette ikke medføre at mer enn 4 displayes i top 4 match den må derfor tømme modellen for hver gang i starten av compare 4. Det skal tas hensyn til top 4 match for bildene i begge databasene 5. Bilde med høyest match skal havne først (bli lagt først i modellen) 6. Hvis man sammenligner og databasen ikke er tom så skal den globale cursoren for dette programmet bli til en hourglass og satt tilbake til normalen når denne tasken er ferdig 7. Kan trykke på top 4 match lista for å få opp menutiae punkter for et gitt bilde og trykke på bilde igjen for å få opp lista igjen	1. Test å sammenligne med en tom database 2. Legg til nye fingeravtrykk på en eksisterende bruker før compare har blitt trykt 3. samme som ovenfor, men etter compare har blitt trykt og trykk compare igjen. 4 lag en ny bruker før du trykket compare og legg inn nye fingeravtrykk 5. Lag en ny bruker etter du har trykket compare, legg inn fingeravtrykk og trykk compare igjen. 6. Trykk på analyze fingerprint 7. Trykk på clear og gjenta det ovenfor 8. Trykk på compare når hour glass vises
8		no/hig/main/UtilityClass.java database/Queries.java no/hig/main/Start.java no/hig/FingerprintModule/FPUserFingerPrint no/hig/FingerprintModule/FPStoreFingerTabWin. java jni4netFolderForFingerprint/* FingerprintFunctions.j4n.jar (class : FingerprintFunctionalities) jni4net.j-0.8.8.0.jar	FPStoreFingerTabWin : ALT FingerprintFunctionalities : obtainSampleQualityByImage(...), markMinutiaeInSample(...), comparisonBetweenTwoSamples(...), UtilityClass : listFileRename(...), getFileExtension(...), startScanSoftware(...), permOrTempStorage(...), insertOrSetFingerprints(...), highestQualityFile(...), listOffAllFilesForGivenFingerForGivenUser AndForgivenMode(...), getFileExtension(...), FPUserFingerPrint : getFingerNo(), Queries : getFingerprintsOfUser(...), ifRowExists(...), insertFingerData(....), Start : progressWait(),	Store: 1. Sørg for at riktig bruker / database ble valgt 2. Sørg for at rename kjørte riktig 3. Hvis bilder som ble tatt har dårlig kvalitet skal disse slettes 4. Insert (bilde med best kvalitet som også er av den fingern) om det ikke finnes et fingeravtrykk for en gitt finger og på gitt bruker. (bilder med for dårlig kvalitet er slettet før dette steget) 5. Update database for en gitt finger om det finnes en ny en med bedre kvalitet... Ved lik kvalitet skal det som stå der fra før bli værende. 6. Vis hour glass cursor når tasken kjører og gå tilbake til vanlig cursor når den er ferdig 7. om vi skifter database så skal ikke brukeren vi har higlighet fra en database, bli highlighta noe mer 8. view fungerer slik at du kan se hvilke bilder som har blitt lagret for en gitt bruker, om ikke noe har blitt lagret viser den en tom liste, ellers lar den deg trykke på de bildene i lista for å få opp minutiae punktene dens	1. Velg en vilkårlig bruker fra temp databasen 2. Velg en vilkårlig bruker fra perm databasen 3. Bytt mellom databsene og se om brukeren du valgte i 1 og 2 fortsatt er valgt/highligta 4. Trykk flere ganger 5. Start scanne programmet for en bruker i temp og så i Perm og avslutt scan programmet før du har tatt et fingeravtrykk for brukeren. 6. Insert nye bilder for alle fingra for en bruker 7. Velg samme bruker som i og insert nye bilder 8. Ta bilder med ulike scannere
9		no/hig/main/InfoTab.java	ALT fra infotab	InfoTab kunne bruke SHEF til å redigere HTML og gi et grensesnitt for å automatisk generere HTML ut ifra et grafisk grensesnitt. 1. Kan lagre til disk 2. Kan lese fra disk 3, Kan fortsette å redigere der vi slapp uten å miste endringer 4. Kan automatisk generere html 5. kan skrive HTML selv	1. Opprett en ny infotab 2. Trykk edit og editer noen endringer 3. se om endringene har blitt gjort tilgjengelig, med en gang vi lukker SHEF 4. Trykk på grensesnittet for å generere HTML 5. Skriv inn html selv
10		filer som slutter på Rendrer i filnavnet under fingerprintModule	ALT i disse	Viser renderene riktig informasjon?	1. Bruk view i Store, Compare with database I live scan og scan I store

Figur I.7: Testdokumentasjon over aktuelle tester

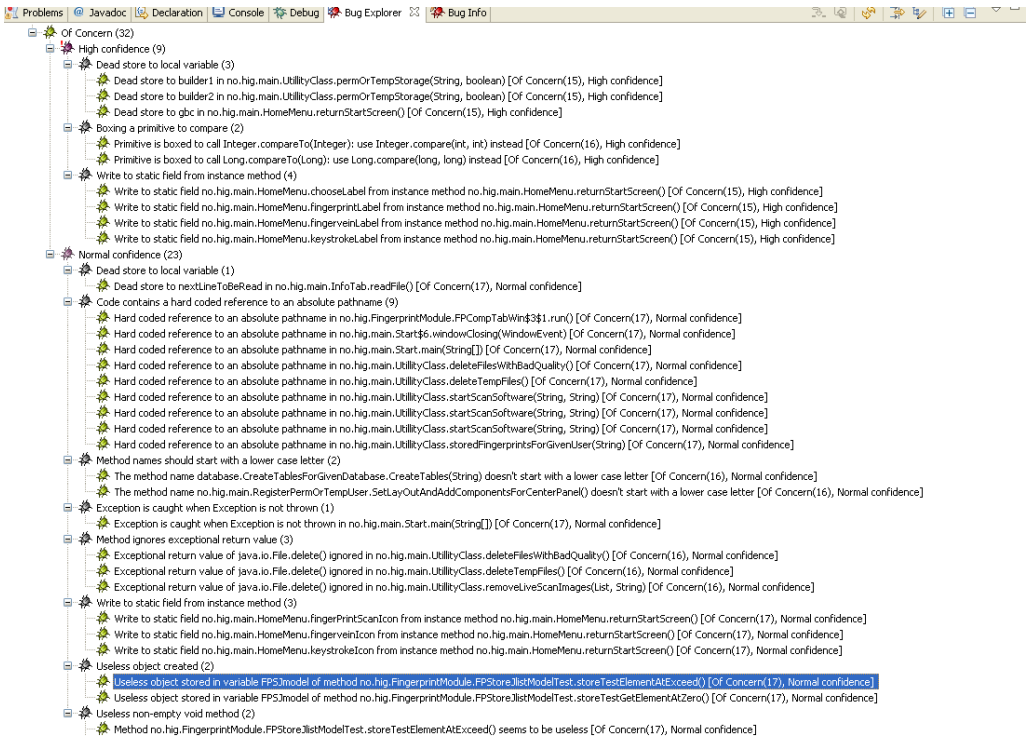
Resultat for	Resultat beskrivelse	Gjennomført	Handling
1	3. Beskjeden er ikke sentrert 4. Ikke på plass 6. Fargevalg ikke bestemt enda 7. Ikoner ikke valgt enda 8. Ikke tilpasset ulike maskiner	04.02.2015	3. Bruk html formattering for å sentrere 4. Gjenta test når dette er på plass 6. Velg hvilke farger skal brukes for hva og sett disse som conster 7. Finn Creative commons som kan benyttes 8. Finn løsning hvor man kan dynamisk finne ut hvor oppgave linja er orientert, hvor stor den er osv og lag fullscreen ut ifra dette
1	4. Ikke på plass 6. ST	04.02.2015	
1	4. Lagt til i ressurser, men får null pointer exception når vi prøver å sette de. 6. ST 7. samme som i 4	02.03.2015	Løs feil i filpath for ressurser
1	6. ST	03.03.2015	
1	OK	08.03.2015	
1	4: null pointer exception	13.03.2015	Skyldes merging feil
1	OK	13.03.2015	
2	1 - 5 ikke implementert	12.02.2015	ta igjen test når 1-5 er implementert
2	6: SQL exception (already established connection) 7: ----------||----------	13.02.2015	Glemte å ta med defaultCloseOperation på Jframe
2	1-5 ST 6: OK 7: OK	13.02.2015	
2	OK	19.02.2015	
3	2. Funker ikke om man skriver samme brukernavn i ulike case	08.02.2015	lagret brukernavn i databasen i lower case så det ikke blir mulig å sammenligne i feil case
3	OK	10.02.2015	
3	1. Kunne lagre "" (tom input) i databasen	31.03.2015	Endre måten vi sammenligner på. Bruker nå equals funksjonen til String istendenfor !=
3	OK	31.03.2015	
4	2-5 ikke implementert enda 1 Programmet viser ikke alle grafiske elementer riktig	30.01.2015	Dette skyldes at vi må starte prosessen fra riktig mappe som startmappe /enviroment
4	2-5 ST 1 OK	30.01.2015	
4	2 Ikke testet enda 3 og 4 OK 5: Gir ikke alltid det siste bilde som ble tatt	06.02.2015	Må sortere på creation Time istendenfor lastModified på fil attributter. Må ta hensyn at ved memory mapped IO så blir det problematikk med oppdatering av creationTime og nye filer med samme navn som tidligere slettet filer (får samme creation time på NTFS fil systemet) å spørre java med å starte garbage collector fungerte ikke som en løsning setter i en løsning med å appende en random tall verdi på filnavnet for å få filas navn til å bli unik
4	5 OK	19.02.2015	
4	2 Får null pointer exception om det ikke ligger noen filer på disk før og etter scan programmet har blitt kjørt	20.03.2015	sjekk om siste fila er null
4	OK		
4	Får null pointer exception om vi starter scanne programmet uten at det ligger noe på disk og vi avslutter uten å ha tatt et bilde for så å trykke på analyze eller compare	01.04.2015	Hvis det ikke ligger noe på disk før og etter scanne programmet har blitt avsluttes så benytter vi clear.doClick() på sjekken om siste fila er null
4	OK		
5	Benyttet ulike instancer av samme klasse av FPMainManu (Start og HomeMenu)	16.03.2015	Sørg for at vi bruker samme instance
5	OK	16.03.2015	
6	Feil med at den er case sensitiv	01.04.2015	benytt toLowerCase() i isNotAllowed(...)
6	OK	01.04.2015	
7	funskjonene innvolvert har eksistert lenge, men har blitt betydelig modefisert for å ikke benytte seg av den siste fila som ble tatt, men for å benytte seg av fila som ble tatt i live scan. en god del forbedringer på koden som også gir oss hour glass cursor. Siden endringene på hvordan funskjonene skal oppføre seg har endret seg så betydelig så telles 7 som en test i som en helhet fra idag selv om mye av dette var på plass tidligere OK	01.04.2015	
8	4,5,8 : SQL syntax feil	18.03.2015	Fikset syntax feil
8	OK	20.03.2015	
9	3. Endringer skjer ikke med en gang vi lukker Joptionpane(shef)	10.02.2015	Endre rekkefølgen på noen statements
9	OK	10.02.2015	
10	OK	16.02.2015	
10	Endring av koden medførte casting feil	24.03.2015	Endret klassa som vi skulle caste til og funskjoner som blir kalt
10	OK	25.03.2015	

Figur I.8: Testdokumentasjon over aktuelle tester

J Statisk kode analyse resultat

Hvis figuren er litt vanskelig å lese så er det lurt å benytte seg av zoome-funksjonaliteten i pdf-leseren og zoome inn på ca +150%.

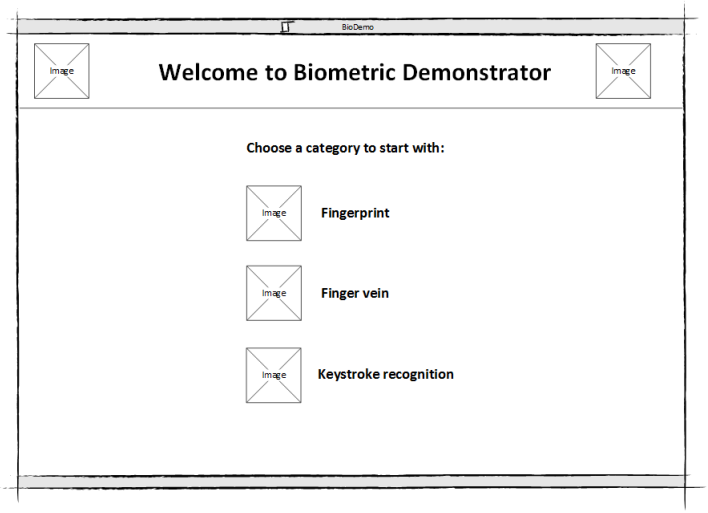
Figur J.9: Resultat av oppdagede ”troubling” bugs før fiks



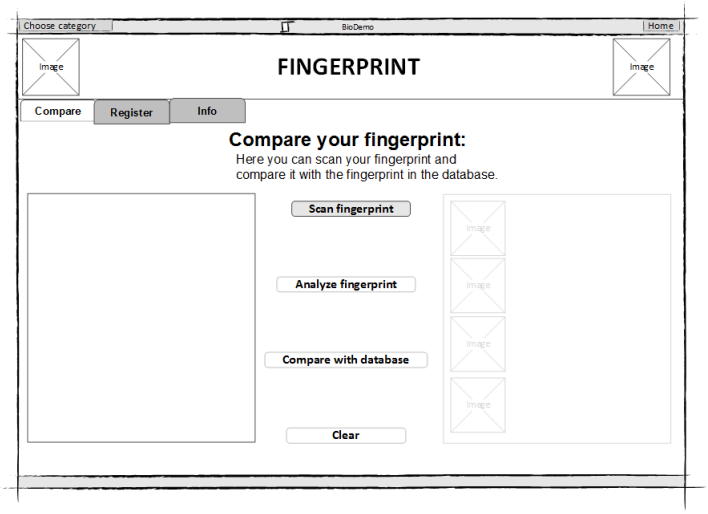
Figur J.10: Resultat av oppdagede ”concern” bugs før fiks

K GUI-skisser

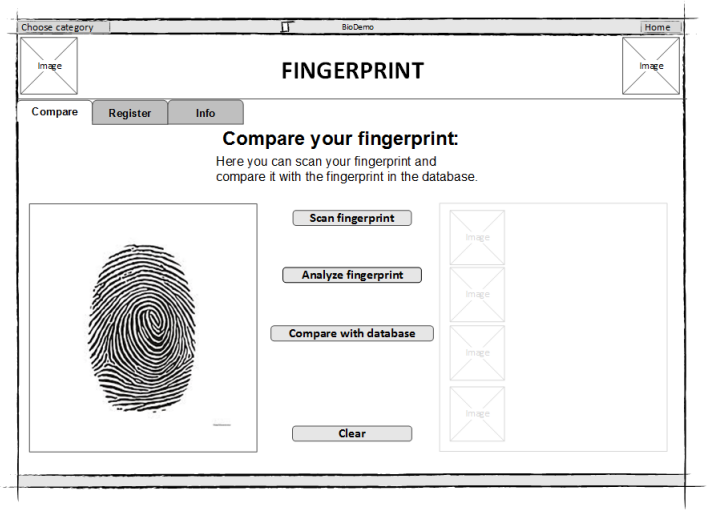
Dette er skissene etter siste versjon. Det var noen skisser som ble på et tidspunkt ikke aktuelt å implementere, men er med i denne fremstillingen på bakgrunn av at det gir et helthetlig bilde av programmet.



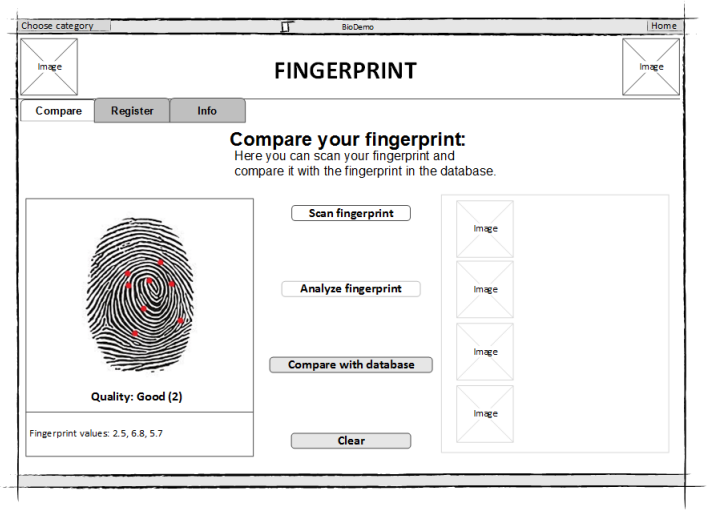
Figur K.11: GUI-Velkomstskjerm



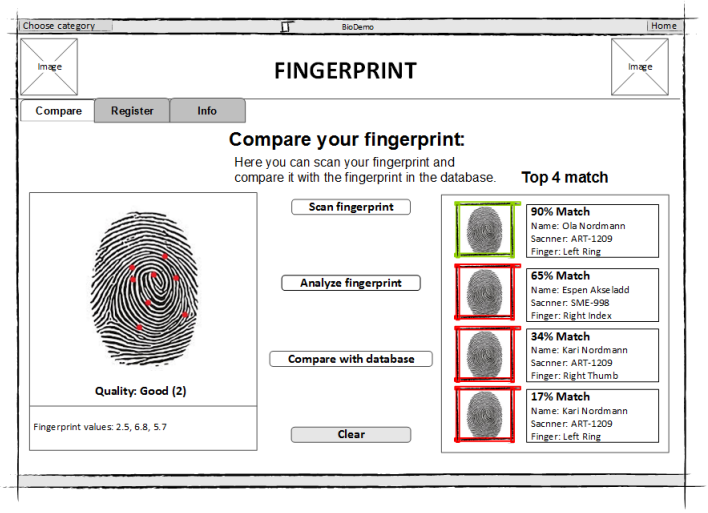
Figur K.12: GUI-Sammenlign før skanning



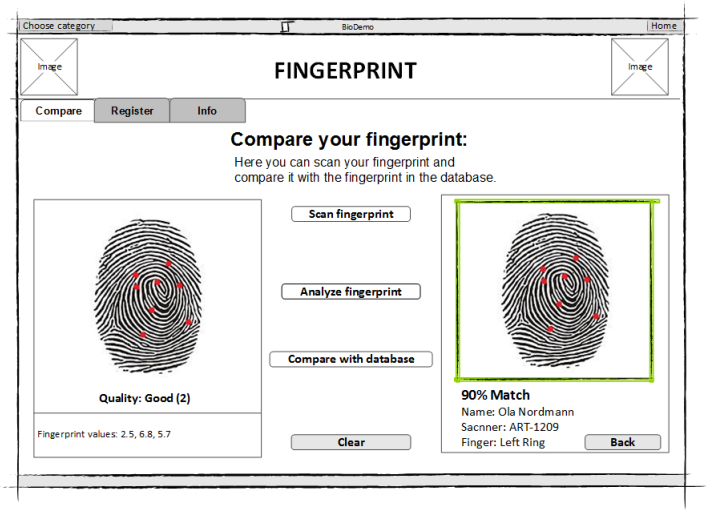
Figur K.13: GUI-Livescan av fingeravtrykk



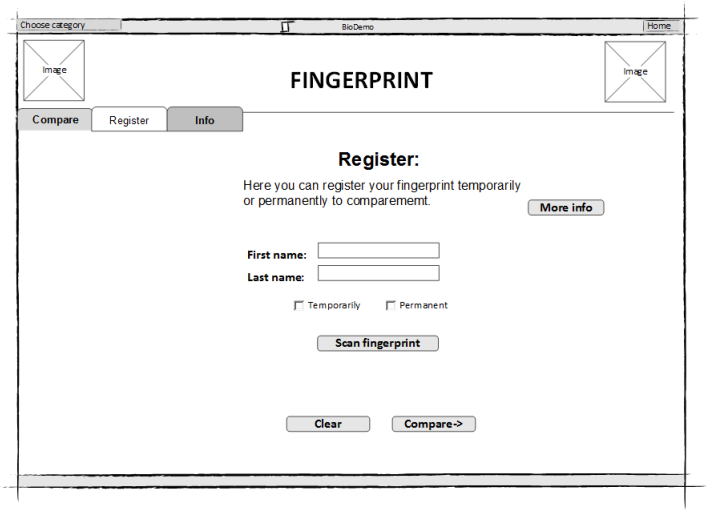
Figur K.14: GUI-Analyse av fingeravtrykk



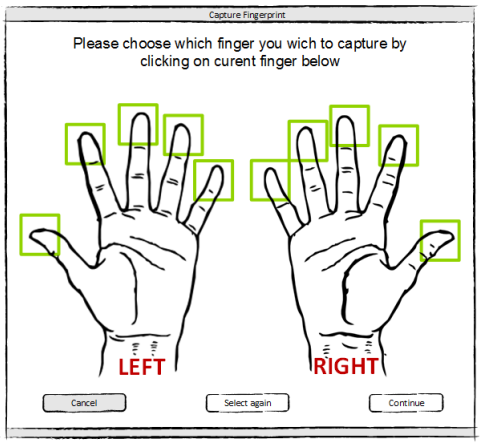
Figur K.15: GUI-Resultat av sammenligningen mot databasen



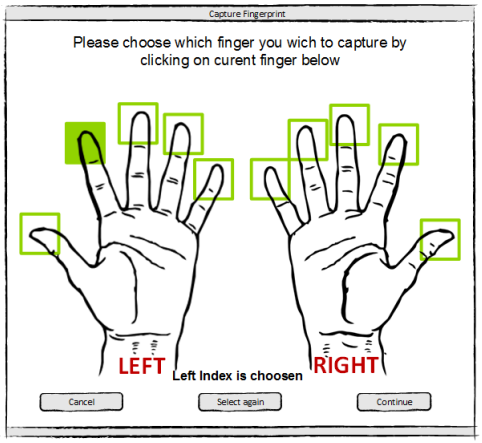
Figur K.16: GUI-Forstørrelse av element i resultatlisten



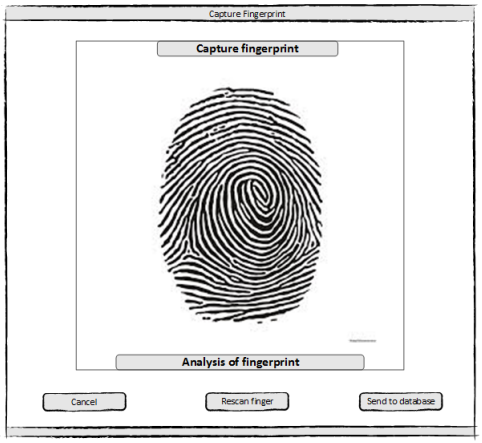
Figur K.17: GUI-Registrer bruker



Figur K.18: GUI-Velg finger - ikke implementert

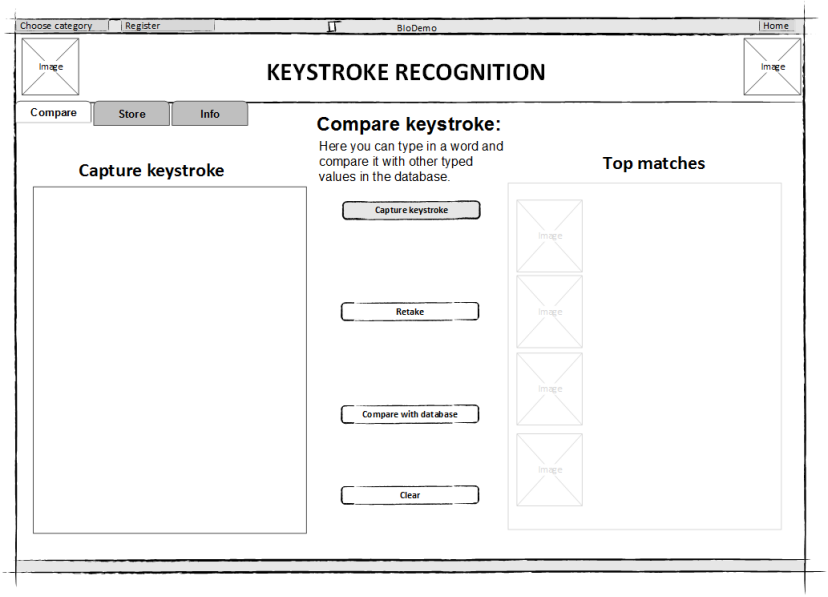


Figur K.19: GUI-Valgt finger- ikke implementert

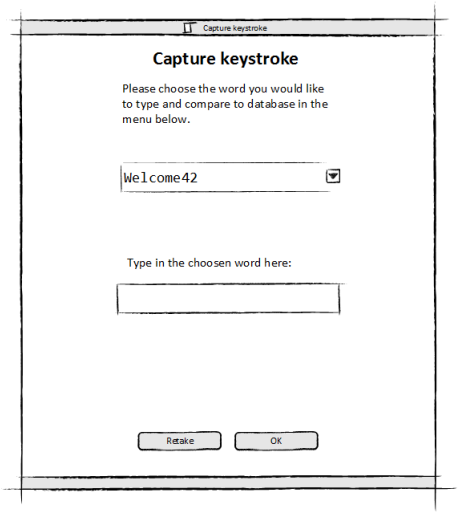


Figur K.20: GUI-Skannet finger - ikke implementert

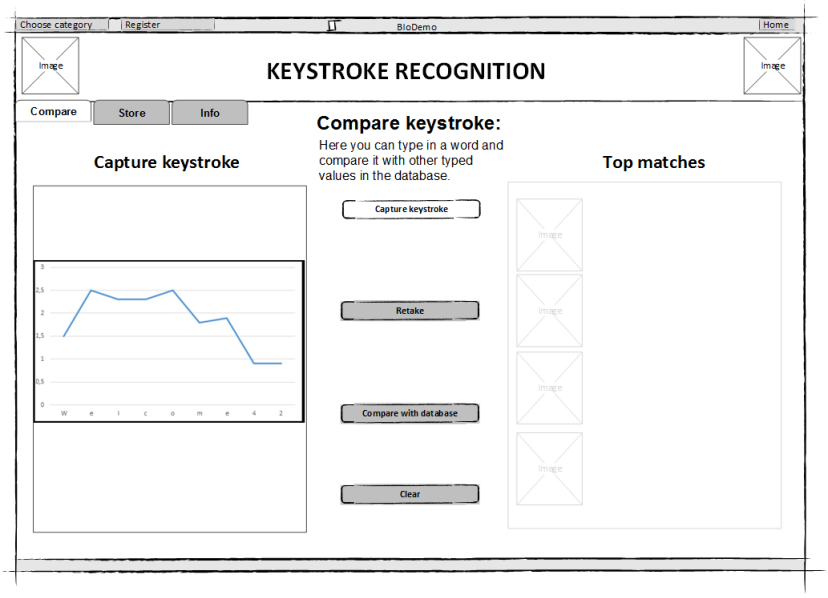
Figur K.21: GUI-Analyser skannet finger- ikke implementert



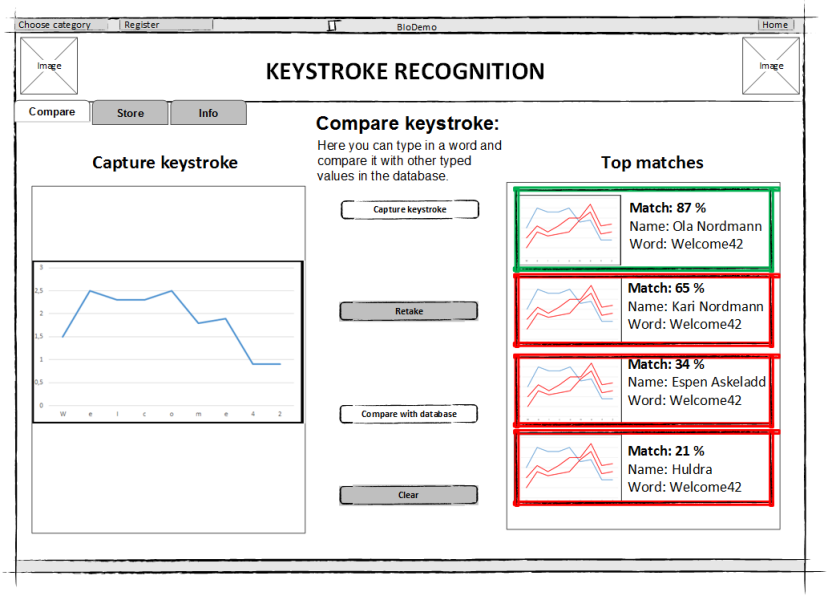
Figur K.22: GUI-Compare-del for keystroke



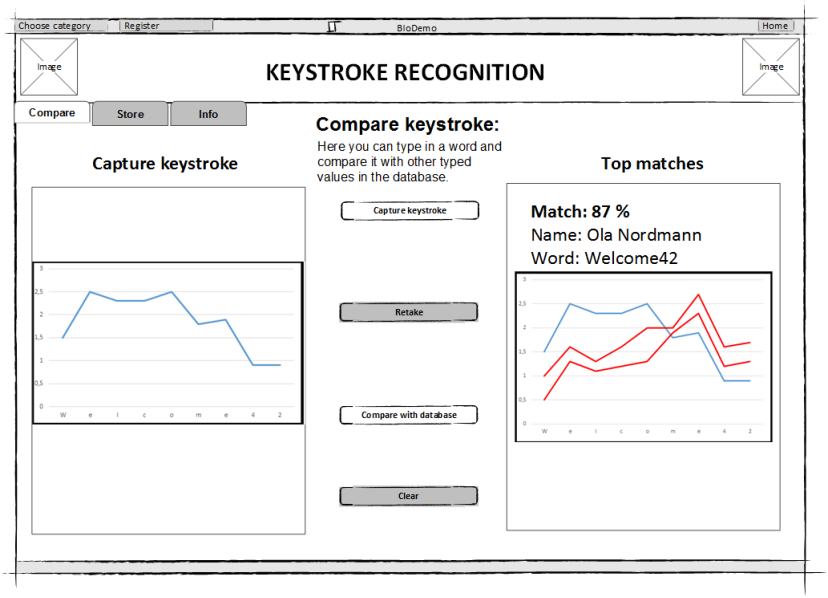
Figur K.23: GUI-Brukerinntasting for keystroke



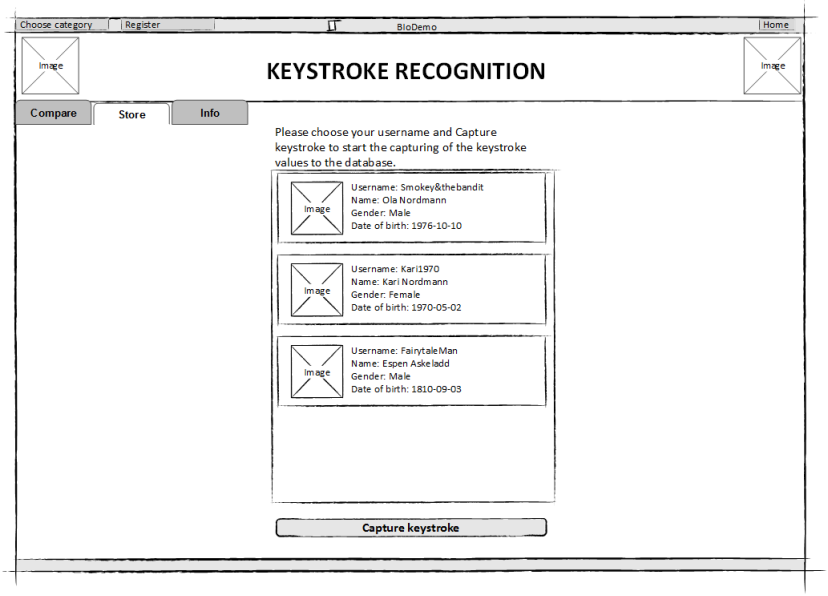
Figur K.24: GUI-Viser grafen av brukerens inntasting



Figur K.25: GUI-Viser taste-match etter sammenligning med database



Figur K.26: GUI-Viser innzoomet match-graf



Figur K.27: GUI-Viser lagring av tastetrykk på bestemt bruker

L Timeliste

Her er en oppsummert timeliste fordelt pr gruppemedlem pr uke. Uke 1 tilsvarer første prosjektuke (start 5 februar 2015) og uke 19 tilsvarer siste uke før denne rapporten ble levert (slutt 15 mai 2015).

Uke nr	Sum Joachim	Sum Anna	Sum Synne	Summering
1	22,5	11,5	12,0	*46,0*
2	38,8	27,5	28,3	*94,6*
3	31,0	30,3	29,9	*91,2*
4	33,0	29,8	27,0	*89,8*
5	24,0	30,2	30,2	*84,4*
6	27,0	28,5	26,6	*82,1*
7	22,5	28,5	22,8	*73,8*
8	45,0	34,3	34,9	*114,1*
9	47,0	28,0	33,0	*108,0*
10	10,0	28,8	32,3	*71,0*
11	46,0	29,0	31,9	*106,9*
12	45,0	30,8	29,7	*105,4*
13	43,0	17,5	15,7	*76,2*
14	32,0	24,0	26,5	*82,5*
15	15,0	30,0	29,5	*74,5*
16	9,0	28,0	32,2	*69,2*
17	38,0	30,0	35,7	*103,7*
18	53,0	35,5	42,0	*130,5*
19	13,0	37,5	44,5	*95,0*
SUM	**594,8**	**539,4**	**564,4**	**1698,6**
Gj.Snitt pr Uke	*31,3*	*28,4*	*29,7*	*89,4*

Figur L.28: Timeliste for arbeid i prosjekttiden

M Prosjektkontrakt

HØGSKOLEN I GJØVIK

PROJECT AGREEMENT

between Gjøvik University College (GUC) (education institution),

Patrick Bours / NISLAB (employer), and
Synne Gran Østern (120922)
Anna kaardal (120918)
Joachim Hansen (120983) (student(s))

The agreement specifies obligations of the contracting parties concerning the completion of the project and the rights to use the results that the project produces:

1. The student(s) shall complete the project in the period from 05.01.15 to 03.06.15.

   The students shall in this period follow a set schedule where GUC gives academic supervision. The employer contributes with project assistance as agreed upon at set times. The employer puts knowledge and materials at disposal necessary to complete the project. It is assumed that given problems in the project are adapted to a suitable level for the students' academic knowledge. It is the employer's duty to evaluate the project for free on enquiry from GUC.

2. The costs of completion of the project are covered as follows:
   - Employer covers completion of the project such as materials, phone/fax, travelling and necessary accommodation on places far from GUC. Students cover the expenses for printing and completion of the written assignment of the project.
   - The right of ownership to potential prototypes falls to those who have paid the components and materials and so on used to make the prototype. If it is necessary with larger or specific investments to complete the project, it has to be made an own agreement between parties about potential cost allocation and right of ownership.

3. GUC is no guarantor that what employer have ordered works after intentions, nor that the project will be completed. The project must be considered as an exam related assignment that will be evaluated by lecturer/supervisor and examiner. Nevertheless it is an obligation for the performer of the project to complete it according to specifications, function level and times as agreed upon.

4. The total assignment with drawings, models and apparatus as well as program listing, source codes and so on included as a part of or as an appendix to the assignment, is handed over as a copy to GUC who free of charge can use it in lessons and in research purpose. The assignment or appendix cannot be used by GUC for other purposes, and will not be handed over to an outsider without an agreement with the rest of the parties in this agreement. This applies as well to companies where employees at GUC and/or students have interests.

   Assignments with grade C or better are registered and placed at the school's library. An electronic project assignment without attachments will be placed on the library part of the school's website. This depends on that the students sign a separate agreement where they give the library rights to make their main project available both on print and on Internet (ck. The Copyright Act). Employer and supervisor accept this kind of disclosure when they sign this project agreement, and they must possibly give a written message to students and dean if they during the project period change view on this kind of disclosure.

5. The assignment's specifications and results can be used by the employer's own work. If the student(s) in its assignment or while working with it, makes a patentable invention, relations between employer and student(s) applies as described in *Act respecting the right to employees' inventions* of 17th of April 1970, §§ 4-10.

6. Beyond the publicising mentioned in item 4, the student(s) have no right to publicise his/hers/theirs assignment, fully or partly or as a part of another work, without consensus from the employer. Equivalent consent must be made between student(s) and lecturer/supervisor regarding the material placed at disposal by the lecturer/supervisor.

7. The students shall hand in the assignment with attachments electronic (PDF) in Fronter. In addition the students shall hand in a copy to the employer.

8. This agreement is drawn up with one copy to each party. On behalf of GUC it is dean/vice dean who approves the agreement.

9. In each case it is possible to enter separate agreement between employer, student(s) and GUC who closer regulate conditions regarding issues such as ownership, further use, confidentiality, cost coverage, and economic utilisation of the results.

   If employer and student(s) wish an additional or new agreement, this will occur without GUC as a party.

10. When GUC also act as employer, GUC accede to the agreement both as education institution and as employer.

11. Possible disagreements concerning understanding of this agreement are solved by negotiations between the parties. If consensus is not achieved, the parties agree that the disagreement is solved by negotiations, according to provision in Civil Procedure Act of 13th of August 1915, no 6, chapter 32.

12. Participants by project implementation:

GUCs supervisor (name): Tom Roise

Employers contact person (name): Patrick Bours.

Student(s) (signature): Anna Haardal date 23/01/2015

Sigmund Egeran Olsen date 28.01.2015

Jo H date 25.01.2015

date

Employer (signature): Patrick Bours. date 28/1/2015

IMT Dean/Vice Dean (signature): date

Revised 25th of November, 2010, Hilde Bakke

Vedlegg 1 – Spesifisering av åpenhet og deling

Modifisering av §4 i kontraktmal

I bacheloroppgaven «Biometric Demostrator» har oppdragsgiver bedt om å forandre litt i prosjektkontraktmalen vedrørende åpenhet og deling av kildekode som produseres i løpet av prosjektet.

Årsaken til denne forespørselen er at vi skal benytte oss av programvare som er beskyttet av opphavsrett til tredjepart-leverandører og styrt under lisenser som bare Norwegian Biometrics Laboratory (NBL) har rettigheter til. I tillegg er det et ønske om at analyseprogramvaren som behandler og analyserer de biometriske dataene vi skal bruke i programmet ikke skal deles med utenforstående.

Vi vil derfor, etter ønske fra oppdragsgiver, unnlate å ta med alt av kildekode i prosjektrapporten. Vi vil spesifisere i rapporten hvilke deler av kildekoden som er utelatt og hvorfor. De delene som vil bli utelatt vil være basert på avsnittet over. Alt av kildekode som legges med i prosjektrapporten kan deles etter §4 bestemmelser.

N Prosjektplan - BioDemo

Prosjektplan

Joachim Hansen, Anna Kaardal, Synne Gran Østern

10. februar 2015

1 Mål og Rammer

1.1 Bakgrunn

I forbindelse med bacheloroppgaven våren 2015 falt vår gruppes valg på en oppgave fra Patrick Bours ved Norwegian Biometric Laboratory som går ut på å lage en biometridemonstrator. Biometri brukes i informasjonssikkerhet som autentiseringsmetoder avhengig av målbare fysiske egenskaper. Norwegian Biometric Laboratory (NBL) er en viktig del av NISlab (Norwegian Information Security Laboratory) ved Høgskolen i Gjøvik (HiG), og arbeider med forskning på biometri. De jobber også aktivt for å øke kunnskapen rundt dette feltet i Norge [1]. Her foregår forskning på flere forskjellige felter innenfor biometriens verden, slik som:

- Fingeravtrykk - og åregjenkjenning
- 2D og 3D ansiktsgjenkjenning
- Tastegjenkjenning
- Tanngjenkjenning
- Irisgjenkjenning
- Øregjenkjenning
- Ganglagsgjenkjenning
- Signaturgjenkjenning

Tilhørende HiG er flere databaser som inneholder data fra de forskjellige fagfeltene ovenfor. NBL har utviklet programmer som analyserer og sammenligner disse dataene. Det har til nå kun vært fokus på effektiviteten til disse programmene, men svært lite på hvordan outputen som genereres av disse fremstilles. I dag genereres output som tall og enkle grafer, [1], noe som i mange tilfeller vil være vanskelig å forstå for andre enn forskerne ved NBL.

NBL ønsker at flere skal kunne få en demonstrasjon av, og forstå hva som skjer i forskningen hos NBL. I denne forbindelse er det ønskelig at det skal utvikles en applikasjon med et grafisk brukergrensesnitt som presenterer dette på en forståelig måte, også for personer utenfor NBL.

1.2 Prosjektmål

1.2.1 Effektmål

Vi forventer at det ferdge produktet skal:

- Kunne brukes for å promotere NBL ved ekte demostrasjon av forskningen for gjester
- Kunne demostrere og gi opplæring i biometriprosesseringer for utenforstående slik at de forstår hva som skjer
- Bli brukt til å øke støtten og omdømmet til NBL

1.2.2 Resultatmål

Ved prosjektets slutt den 15.05.15 skal produktet kunne:

- Demostrere biometriprosesseringer med MÅ-ha funksjonaliteter:

  Fingeravtrykksgjenkjenning

  Åregjenkjenning i fingre

  Tastegjenkjenning

- Kunne innhente data fra eksisterende database og sammenligne med innhentet brukerdata.
- Applikasjonens kode må være kommentert og dokumentert slik at NBL kan videreutvikle den.
- Kodens formulering skal følge de standardeene vi har valgt å følge
- Kunne sammenligne biometriske data fra demostrator opp mot eksisterende database
- Må kunne slette biometriske data som er innhenetet i løpet av demostrasjonen (ikke data fra databasene til NBL)

1.3 Rammer

- Selve prosjektarbeidet skal foregå i perioden 07.01.15 og frem til 15.05.15, som er innleveringsfristen for sluttlevering av rapporten. Etter dette følger en muntlig presentasjon av oppgaven (1-3).06.15
- Vi får tildelt en PC til disposisjon. Programmet vi utvikler skal utelukkende bli installert og kjørt på denne enheten.
- Vi vil få "låne" forskjellig type hardware: Se del 2.3 for liste av disse.
- På grunn av at vi har fått tildelt en spesefikk PC til disposisjon og tilhørende biometriscannere/lesere, så har vi vært så heldige å få tildelt rom A151 til å jobbe med dette prosjeketet.
- På grunn av linsensrettigheter på programvare tilknyttet de forskjellige scannerne, og etter ønske fra oppdragsgiver , så vil deler av kildekoden bli utelatt fra den offentlige rapporten. Vi vil ta med så mye vi kan, og presisere der det er utelatt kildekode om det.

2 Omfang

2.1 Fagområde

Hovedområdet for oppgaven er biometri. Vi skal produsere en bacheloroppgave innen informasjonssikkerhet, og derfor vil informasjonssikkerhet vektlegges underveis i oppgaven. Dette vil blant annet innebære risikovurderinger i deler av prosjektet, fra prosjektplan og helt frem til det ferdige produktet. Vi vil også komme inn på en rekke andre fagområder, slik som:

- Databasehåndtering
- Interaksjonsdesign

- Programmering i flere språk (i hovedsak Java)
- Programvareutvikling
- Biometri

2.1.1 Biometri

Etter ønske fra oppdragsgiver, så skal vi prøve å dekke mest og bredest mulig over de mest brukte biometriområdene. De fleste av NBL forskningsområder er nevnt i del 1.1, og av de så kommer vi til å komme innpå områder som fingeravtrykk, fingerårer, tastaturmetode, ansikt, iris og ører. For å kunne møte oppdragsgivers ønsker, så vil vi bruke tid på å sette oss inn i de forskjellige områdene. Les mer i del 3.1 om dette.

2.2 Avgrensninger

Applikasjonen skal kun utvikles for Windows operativssystemet. Det er ikke noe krav om noe særlig portabilitet, siden applikasjonen primært skal kjøres på en maskin hos NBL. Den kjørende applikasjonen er ikke ansvarlig for å forklare brukeren ukjent termologi, for ansatte ved NBL vil ved behov forklare dette og være tilgjengelig for å veilede gjennom prosessen.

2.3 Oppgavebeskrivelse

Det skal i denne oppgaven utvikles en demonstrator som viser grafisk hva som foregår når forskjellige former av biometriske kjennetegn sammenlignes og analyseres. Vi skal utvikle en applikasjon som tar i mot output fra de forskjellige analyseprogrammene utviklet av NBL og presenterer disse i et grafisk brukergrensesnitt, slik at det blir enkelt for brukeren å følge med på og forstå hele prosessen, både når det gjelder hva som sammenlignes, og resultatet av sammenligningen.

Vi vil utvikle i moduler, en for hver type biometrisk kjennetegn. Disse vil i sin tur sammenstilles i en større applikasjon. Vi vil utvikle demonstratorer for så mange av analyseprogrammene vi får mulighet til, men oppdragsgiver har satt følgende må-krav: fingeravtrykksgjenkjenning, åregjenkjenning og tastegjenkjenning. Se del 1.2 for resultatmålene våre. Vi vil få tilgang til en PC med Windows som operativsystem, der applikasjonen skal kunne kjøres. De eksisterende analyseprogrammene er hovedsakelig skrevet i Matlab, og konvertering til andre programmeringsspråk kan bli nødvendig. En del av oppgaven vil også være å få applikasjonen vi utvikler til å være i stand til å registrere forksjellige typer hardware, dette fordi vi er avhengige av forskjellige typer sensorer som registrerer biometriske kjennetegn.

De forskjellige typene som er nevnt av oppdragsgiver er:

- Flere forskjellige modeller av fingeravtrykkscannere
- Årescanner for finger
- Fotokamera for iris, ansikt og øreavbildning
- Tastatur til tastegjennkjenning (denne er standard tilhørende tildelt PC)

Det vil i tillegg bli behov for å etablere en database med de nødvendigste biometriske dataene for å kunne gjennomføre sammenligningen og analysene. Denne databasen vil bli delt i to, en permament og en imidlertidig for gjester. Den imidlertidige vil bli slettet ved utgang av applikasjonen.

Vi vil også fordype oss litt i interaksjonsdesign, og grunnleggende prinsipper i forbindelse med dette. Det er viktig at målgruppen for vår applikasjon faktisk forstår det vi prøver å kommunisere.

3 Prosjektorganisering

3.1 Ansvarsforhold og roller

For å være mest mulig effektive og unngå å ha doble roller og oppgaver har vi valgt å fordele spesialroller og ansvar til gruppemedlemmene.

3.1.1 Prosjektleder - Anna Kaardal

Hensikten med denne rollen er å ha en person i gruppa som skal ha oversikt over hvem som gjør hva og når, hvor mye tid hver enkelt jobber, og hvilke frister gruppa må forholde seg til. Dette er nødvendig for å kunne ta gode beslutninger på hvordan gruppa skal disponere sine ressurser. Den som har denne rollen skal bidra til en rettferdig arbeidsfordeling, og samtidig være ansvarlig for å ta opp bekymringsmeldinger med gruppa og veileder. Denne personen har også endelig beslutningskraft om gruppa ikke klarer å fatte en beslutning i felleskap. Vi har valgt å tilegne en fast person prosjektlederrollen.

3.1.2 Programdriftansvarlig - Joachim Hansen

Hensikten med denne rollen er å spare gruppas ressurser ved at utfordringer med programvare kan løses primært av en person, i stedet for at hele gruppa står stille og alle prøver å løse utfordringen på ulike måter. Vedkommende med denne rollen er ansvarlig for å finne alternativer til programvare som gruppa har behov for, sette seg inn i teknologier, hjelpe gruppa med å konfigurere ulike verktøy på deres maskin, samt lage veiledninger for resten av gruppa.

3.1.3 Kommunikasjonsansvarlig - Synne Østern

Hensikten med denne rollen er å sikre en ryddig kommunikasjonsprosess med våre veiledere og oppdragsgivere. Kommunikasjonsdirektøren er vårt ansikt utad og er ansvarlig for ekstern kommunikasjon med veileder, institusjon og oppdragsgiver. Denne rollen har også ansvar for å holde prosjektets blogg oppdatert. Og være ordstyrer ved fellesmøter.

3.1.4 Spesialfelt/ekspertområde

Vi har funnet tre aktuelle spesialfelt som vi vil fordele innad i gruppa. Spesialfeltene ble opprettet fordi vi anser det å kunne mye om alt som mer utfordrende enn å kunne mye om noe og litt om alt. De tre spesielfeltene går på I/O (Joachim), GUI-design (Anna) og Matlab (Synne). Ekspertene er ansvarlige å tilby opplæring og ”support” til de andre gruppemedlemmene.

I tillegg vil vi fordele biometriområder oss imellom slik at vi kan dekke mest mulig av dette omfattende området. Dette for å kunne møte fagpersoner med god bakgrunnsinformasjon slik at det ikke vil bli store behov for å få grunnleggende "opplæring" om fagområdet, men heller fokus på hva vi skal benytte oss av i selve applikasjonen. Og med denne kunnskapen har også bedre grunnlag for å kunne oppdage feil eller mangler i utviklingen.

3.2 Rutiner og regler i gruppen

Vi skal ha minimum to til tre statusmøter/sprint daily meetings i løpet av en uke. Vi ønsker å ha fleksibilitet når statusmøtene og felles arbeidsøkt skal være og bruker Google sin kalenderapplikasjon for å plotte inn tidene vi har avtalt. Denne kalenderen er delt med hele gruppen og hver av gruppemedlemmene har redigeringsrettigheter.

Veiledningsmøter er satt til tirsdager klokka 13:15 og møter med oppdragsgiver er annenhver tirsdag klokka 09:00 i forhold til sprintene. Se 3 for mer detaljer om sprinter og statusmøter med oppdragsgiver. Utover dette, vil det også ved behov være mulig å ha møter med aktuelle personer ved NBL og ellers ved IMT. Disse tar vi kontakt med fortløpende ved behov, men vi skal unngå å la prosjektet gå i still pga ventende svar fra ressurspersoner.

En referent vil bli satt opp for hvert møte, og er ansvarlig for å legge ut dette dokumentet på gruppas wikiside på bitbucket.com.

Vi bruker Microsoft Excel for å loggføre hva vi har gjort, når vi har gjort det og hvor mye tid vi har brukt på ulike aktiviteter som f.eks. undersøkelse og prosjektskriving. Det å holde oversikt på hvor mye ressurser vi bruker på en gitt aktivitet kan hjelpe oss å fordele ressursene mer fordelaktig på viktigere deler av prosjektet.

4 Planlegging, oppfølgning og rapportering

4.1 Hovedinndeling av prosjektet

I denne delen har vi argumentert og kommet frem til hvilken systemutviklingsmodell vi velger å jobbe etter, og hvordan dette påvirker planen for statusmøter og beslutningspunkter.

4.1.1 Argumentasjon for SU-modell

Vi har vurdert flere forskjellige systemutviklingsmodeller til bruk i prosjektet. Gruppen og oppdragsgiver ble enige om en modell som støtter endringer ut i prosjektetarbeidet, da vi ser for oss at det kan bli behov for å gjøre endringer i kravspesifikasjonen underveis. Ettersom oppgaven består av flere moduler, se del 2.3, og i tillegg er usikre på hvor mange analyseprogrammer vi vil rekke å gjennomføre, anser vi det som vanskelig å kunne sette noen eksakte datoer i prosjektperioden (med unntak av gitte frister for innlevering). Det vil også være viktig for oss at forskere ved NBL kan ta del i prosjektet underveis, slik at vi kan utvikle et program som oppfyller deres behov, og samtidig unngå større missforståelser.

Dette er vårt største prosjekt til nå, og det er første gang vi jobber etter oppdragsmodell med en faktisk kunde, noe som betyr at vi på bakgrunn av vår manglende erfaring har utfordringer med å

detaljert bestemme på et tidlig stadium hva vi vil gjøre på bestemte tider i løpet prosjeket, noe som plandrevne systemutviklingsmodeller krever.

På bakgrunn av disse punktene kunne vi raskt utelukke plandrevne modeller, slik som fossefallsmodellen. [2]

Overnevnte punkter medfører behovet for en modell som tilbyr fleksibilitet, og det vil derfor være naturlig å vurdere forskjellige smidige (agile) modeller. Disse tillater endringer underveis, og lar samtidig oppdragsgiver ta aktivt del i prosjektet, og komme med endrede ønsker underveis. For et vellykket resultat ved bruk av en slik modell, er det viktig at oppdragsgiver har muligheten til å engasjere seg ta aktivt del i prosjektet. Etter første møte med oppdragsgiver, ble det klart at han var motivert for dette, og hadde gode erfaringer fra tidligere prosjekter.

Agile metoder gjør oss i stand til å levere i inkrementer, noe som vil være hensiktsmessig da vi skal utvikle demonstratorer for forskjellige typer analyseprogrammer, og det vil være naturlig å splitte opp disse i forskjellige moduler, og sikre at vi faktisk er i stand til å levere noen moduler som fungerer enkeltstående, før vi leverer den siste stabile versjonen av applikasjonen i mai 2015.

Scrum er en smidig modell. Her har man en product backlogg å forholde seg til, og det er oppdragsgiver som velger ut hvilke oppgaver som skal bli med i hver sprint [2]. Vi vil da være avhengige at oppdragsgiver er villig til å ta aktivt del, og ha jevnlige møter med oss. En fordel er at prosjektet deles opp i sprinter, og vi leverer et nytt inkrement for hver sprint, noe som står i stil med våre planer. Samtidig støtter denne modellen jevnlige møter, for å holde øvrige gruppemedlemmer oppdatert på status, noe som sikrer god oversikt, samt sikrer at oppdragsgiver/forskere holdes oppdatert.

XP er også en smidig modell vi har vurdert. Her sitter programmererne sammen i par og utvikler produktet. Modellen støtter felles eierskap av all kode [2]. Felles eierskap av kode mener vi er en godt prinsipp, da vi anser det som svært viktig at alle gruppens medlemmer setter seg inn i all kode, også den de ikke har programmert selv. På en annen side, er XP mer tidkrevende, og vi har begrenset med tid. Vi ser også for oss at det blir mer krevende å administrere parprogrammering i en gruppe med tre medlemmer.

RUP er også en aktuell modell å benytte. Den er også smidig og støtter inkrementell levering, samt risikovurdering for hver fase, noe som er positivt. Ulempen er at det er et omfattende rammeverk å sette seg inn i.

Gjenbruk: Uavhengig av hvilken hovedmodell vi velger, vil det i dette prosjektet bli nødvendig å benytte gjenbruksorientert systemutvikling. Det er ikke vår oppgave å utvikle analyseprogrammene, men benytte de som allerede eksisterer. Vi må sørge for at disse delene fungerer sammen med programmet vi utvikler, og vi må være forberedt på at det kan bli behov for modifikasjoner/ konvertering til et annet programmeringsspråk for allerede eksisterende kode.

Etter nærmere diskusjon med oppdragsgiver og veileder, kom vi fram til at en tilpasset Scrum, vil være en god løsning. Vi kommer til å dele arbeidet opp i sprinter på to uker, og etter hver sprint vil vi ha et møte med oppdragsgiver og evt andre aktuelle personer, der vi kort viser hva vi har utviklet i den tilbakelagte sprinten, samt velger ut oppgaver fra backloggen til neste sprint. Det vil ikke kun være oppdragsgiver som velger ut disse oppgavene, det vil skje ved at gruppens medlemmer gjør dette i samråd med oppdragsgiver. Vi vil ikke praktisere korte, daglige statusmøter, men heller litt lengre møter 2-3 dager i uka. Vi kommer også til å benytte elementer fra XP, noe som innebærer at vi

i perioder vil praktisere parprogrammering, samt at vi gjennom hele prosjektet vil praktisere felles eierskap av kode. Vi ser også for oss å benytte risikoanalyser, slik som i RUP, men tilpasse dette mer til Scrum, og ha risikoanalyser flere steder i prosjektet. Del 6.1 omtaler planen for prosjektet etter scrummodellen, og noen spesielle datoer.

5 Organisering av kavlitetssikring

5.1 Dokumentasjon, standardbruk og kildekode

Vi har gjennom disse årene ved HIG opparbeidet oss en del erfaring og kunnskaper om arbeidsverktøy som vi kommer til å benytte. Det har vært endel gode erfaringer, men også en del mindre gode erfaringer. Vi vil i denne delen oppsummere veldig kort hvilke hovedverktøy vi kommer til å bruke, men det kommer til å bli aktuelt å benytte oss av ytterligere verktøy som vi ikke omtaler her.

5.1.1 Kildekode

I dette prosjektet kommer vi til å bruke JavaDoc til å dokumentere kildekoden, i tillegg til kommentering i selve koden. JavaDoc generer automatisk API-dokumentasjon i HTML format, noe som gjør jobben vår lettere med tanke på dokumentasjon og at det vil være behov for at andre enn oss skal kunne lese seg opp på hva som skjer i koden.[3]

Når det gjelder kjøring av JavaDoc-dokumentet skal dette gjøres en gang i uken ved ukeslutt. Dette for å sikre at vi har oppdatert dokumentasjon kontinuerlig i prosjektet.

Ettersom vi ønsker at det skal være enkelt for andre enn oss som har utviklet applikasjonen å forstå hva som skjer i koden, velger vi å følge en kodestandard for Java. Oracles egen kodestandard er fra 1999 og ikke lenger helt oppdatert og vedlikeholdt av Oracle. Men denne standarden forteller ganske grundig blant annet hvordan Java-kode skal formatteres og kommentering av kode. [4] Vil vi i tillegg følge en mer nyere og anerkjent standard for Java; Googles Java Style. [5] Kombinasjonen mellom den offisielle Java dokumentasjonen fra Oracle og den nyere veiledningen fra Google mener vi vil gjøre at koden vår blir mer tilpasset dagens behov.

5.1.2 Analyse og diskusjon

Når det gjelder dokumentasjon av undersøkelse, analyse og diskusjonsdelen av prosjektet, så vil vi underveis i arbeidet dokumentere vårt arbeid i våre respektive ”arbeidsmapper” på felles lagringsserver gitt av IT-tjenesten ved HIG.[1]

Vi vil ta vare på kildeinformasjon kontinuerlig, enten det skulle være URL-adresser eller bøker. Dette må gjøres for at vi skal kunne unngå å bruke ekstra tid til å ”lete” etter kildene i ettertid.

Når vi så begynner på selve rapportskrivingen, vil vi benytte oss av GIT-området for prosjektet. Dette fordi man har her muligheter til versjonskontroll som er velidg nyttig i forbindelse med denne typen arbeid. Se del 5.2 for mer om GIT.

[1] Adressen til vårt tilegna område er: `\\emu.stud.hig.no\Hovedprosjekter\Projectdata\V2015\imt\is\biodemo`

5.1.3 Retningslinjer

Vi vil følge "Retningslinjer for Mastergradsoppgaver og større studentoppgaver på Bachelornivå ved Høgskolen i Gjøvik". [6] Dette betyr at vi f.eks. siterer med Vancouver og fortløpende i teksten. Det skal på denne måten komme klart frem hva som er hentet fra andre hog hva som vi selv har kommet frem til og produsert.

5.1.4 Programutviklingsverktøy

Ettersom vi har falt for Java når det gjelder hovedspråk, så vil vi bruke Eclipse som utviklingsmiljø. Dette er et såkalt integrert utviklingsmiljø (IDE), som støtter flere språk Java deriblant. Eclipse fungerer bra med GIT og BitBucket og kan i tilegg legge til ekstra plug-ins etter behov. [7]

5.2 Versjonskontroll

Det viktigste vi ser etter i vår versjonskontrollsløsning er hvorvidt den kan leve opp til tre kriterier: Feiltoleranse, lagringsplass og brukervennlighet.

For vår gruppe stod valget mellom å benytte SVN-løsning levert av IT-tjenesten ved HiG og GIT som versjonskontroll. SVN har en sentralisert løsning og GIT som har en distribuert løsning. Kort sagt betyr dette at med SVN jobber alle direkte mot en felles repository. Med GIT har alle en lokal repository hver og vil kunne synkronisere eventuelle endringer mellom disse.

Hva er konsekvensene ved å ende opp med en defekt/ødelagt repository med de ulike løsningene? Ved å bruke SVN vil alle filene i respository bli utilgjengelig, ettersom denne er sentralisert. Om det samme skulle skje ved GIT, vil bare repositoryen til en av gruppemedlemmene være påvirket og de siste endringene til denne personen vil kunne gå tapt. Vi konkluderer derfor med at ved å bruke GIT istedenfor SVN sikrer vi en større grad av feiltolleranse. [8]

Vi har også prøvd med å bli kjent med begge løsningene og har kommet fra til at vi er mer komfortable med å bruke GIT. GIT vil i dette prosjektet bli brukt til å lagre og jobbe med kildekoden til applikasjonen, selve rapportskrivingen, prosjektplanen og andre elementer som er relevante til disse punktene.

5.3 Risikoanalyse

Denne delen omtaler en risikoanalyse med tiltak. Figur 1 viser tabellen over de risikoene vi ser på som relevante for dette prosjektet. Vi har valgt å bruk en 3x3 matrise hvor rødt felt er det som regnes som ikke akseptabelt og må gjøres risikoreduserende tiltak på.

5.3.1 Tiltak

Det er tre risikoer som faller under rødt felt i matrisen, i tillegg har vi valgt å utføre tiltak på flere av de risikoene som har kommet under gult felt. Se figur 2

Nr 1: Tap av prosjektdokumenter Denne risikoen er ganske alvorlig, for vi er helt avhengige å kunne både gjenopprette og være trygge på at prosjektdokumenter blir lagret sikkert. Som tiltak her har vi valgt å bruke GIT (ref til versjonskontroll) og nettverkslagringen som vi har fått tilbydd av IT-tjenesten ved HIG. Disse anser vi som løsninger som er sikre nok for vårt behov.

Nr	Risiko	Sannsynlighet	Konsekvens	Kommentar	Tiltak
1	Tap av prosjektdokumenter	Kan skje	Stor	Dette kan skje pga menneskelig eller tekniske feil.	Alle medlemmer av gruppa må bruke versjonskontroll aktivt + lagre viktige dokumenter på felles server.
2	Fravær ett eller flere gruppemedlem (sykdom eller skade)	Sjelden	Stor	Her er det i hovedsak snakk om langvarig sykdom/skade utover 2 uker. Konsekvensen kan være at de andre gr.medl ikke får fortsatt arbeidet pga fraværet.	Grupperegler som dekker slike tilfeller, unngå at det blir en «single-point-of-failure» ved at en gr.mdl er den eneste som er ekspertise på et felt og de andre ikke kan ta over.
3	Får ikke tilgang til programmer og/databaser som er nødvendig	Kan skje	Stor	Pga oppgavens art er det helt nødvendig vi får tilgang og støtte rundt programmer og databaser fra NBL.	Ha jevnlige faste møter med Patrick Bours og NBL, være klare på hva vi trenger fra dem.
4	Ingen eller liten deltakelse av gruppemedlem	Sjelden	Stor	Gruppemedlem kan bidra ulikt, og jobbe forskjellig.	Grupperegler som dekker slike tilfeller. Klare krav om hva som skal gjøres. Andre mdl må aktivt si ifra om sine bekymringer.
5	Liten eller ingen kontakt med NBL	Kan skje	Middels	Pga oppgavens art er vi helt avhengige av at NBL er villige til å dele og hjelpe oss med informasjon.	Ha jevnlige faste møter med Patrick Bours og NBL, være klare på hva vi trenger fra dem. Vi må ta initiativet.
6	Tekniske problemer med lagring og versjonskontrollverktøy (Bitbucket, server hos IT-tjenesten)	Kan skje	Liten	Det kan dukke opp problemer med commit, push og deling av filer, som kan føre til at arbeids blir tapt eller endret på.	Bruke velkjente og profesjonelle verktøy, og tillegg ha gode rutiner for bruk av disse.
7	Uenigheter som påvirker fremdrift og leveranse	Sjelden	Middels	Uenigheter vil dukke opp, men det kan komme situasjoner hvor det går i lås.	Benytte oss av gruppedynamikk for at det er 3 personer og at det vil uansett bli flertall såfremt ikke noen stemmer blankt.
8	For høye arbeids/prosjektkrav, Tidsfrister som overskrides	Kan skje	Middels	Ved en sprint eller i løpet av hele prosjektet oppdager at vi ikke er i stand til å levere på det vi ønsker i utgangspunktet.	Tilpasse oss etter behov, gjøre vårt beste for å levere og hvis dette ikke er nok: få dette med i rapporten.
9	Mangel på kunnskaper	Svært vanlig	Middels	Biometri, MatLab, GUI, IO osv er fagfelt som vi er ganske uerfarne i.	Forberede oss med å finne og lære oss disse nye områdene som vi ser vil bli aktuelle å kunne. Fordele spesialfelt mellom medl for å være mer effektive.
11	Sikkerhetshull som ikke oppdages	Sjelden	Middels	Det stilles høyere krav til oss ang sikkerhet i programvaren.	Følge kodestandarder, utføre flere forskjellige tester.

Figur 1: Risikotabell

		Sannsynlighet		
		Sjelden	Kan skje	Svært vanlig
Konsekvens	Liten		6	
	Middels	7, 11	5, 8	9
	Stor	2, 4	1, 3	

Figur 2: Risikomatrise

Nr 3: Får ikke tilgang/kopier til programmer/databaser som er nødvendig I dette prosjektet er vi helt avhengige av å få tilgang til eller kopier av programmer og databaser som gjør de biometriske utregningene som vi trenger til demostratoren fra NBL. Hvis vi ikke får dette kan vi ikke utvikle et program etter oppdraget. Som tiltak her har vi satt opp jevnlige møter med oppdragsgiver og NBL, og i tillegg presisere hva vi trenger av dem for å få dette programmet til å gjøre det som er behovet.

Nr 9: Mangel på kunnskaper Biometri, MATLAB og GUI er noen av fagområdene vi kommer til å møte i dette prosjektet, og vi har liten kunnskaper og erfaring innenfor disse feltene. Som tiltak her har må vi først få en oversikt over de feltene som det er behov for mer kunnskap, og så må vi sette av tid til å kunne studere disse feltene. Vi har valgt å dele ut ”spesialfelt” til hver av gruppemedlmenne for å gjøre dette mer effektiv. (Ref til delen om roller)

Nr 2: Fravær hos ett eller flere gruppemedlem Siden vi er en gruppe på 3 så er vi desverre sårbare hvis ett eller flere gruppemedlem skulle ha ett lengre fravær fra prosjektet. For å unngå at dette påvirker prosjektet i altfor stor grad vil vi sørge for at alle lagrer sine arbeidsdokumenter på felles nettverksserver og unngå at ett gruppemedlem er det eneste som kan noe om et fagfelt i prosjektet. Gruppemedlemmene står som ansvarlige å lære opp de andre i ”Spesialfeltene”, slik at en

annen kan til nød ta over arbeidsoppgavene.

Nr 4: Ingen eller liten deltakelse av gruppemedlem Veldig lik risiko over, men her er det ikke snakk om fravær men heller liten deltakelse. Som tiltak her er det viktig å ha gode utviklet gruppe-regler som dekke slike tilfeller, og det er likeså viktig at de andre gruppemedlemmene reagerer hvis dette skulle skje. Grupperegler ligger vedlagt som vedlegg....

Nr 5: Liten eller ingen kontakt med NBL Denne risikoen er ganske lik nr 3 over, men går mer på kontakt mellom oss og NBL. Det er vesentlig at vi kan få hjelp og bistand fra ekspertene innenfor biometri, og kunne be om tilbakemeldinger om det vi gjør er riktig. Tiltaket er det samme som punkt nr 3.

Nr 8: For høye arbeidskrav/prosjektkrav Vi er såpass uerfarne på større projekt som denne oppgaven er, og derfor er det utfordrende for oss å kunne vite hva vi vil kunne klare å levere på og hva vi ikke rekker. Vi vil ta en nøye vurdering om hva vi mener, men vi vil i tillegg også benytte oss muligheten til å redusere kravene underveis hvis det skulle bli nødvendig. Drøftingen og begrunnelsen vil bli dokumentert i rapporten.

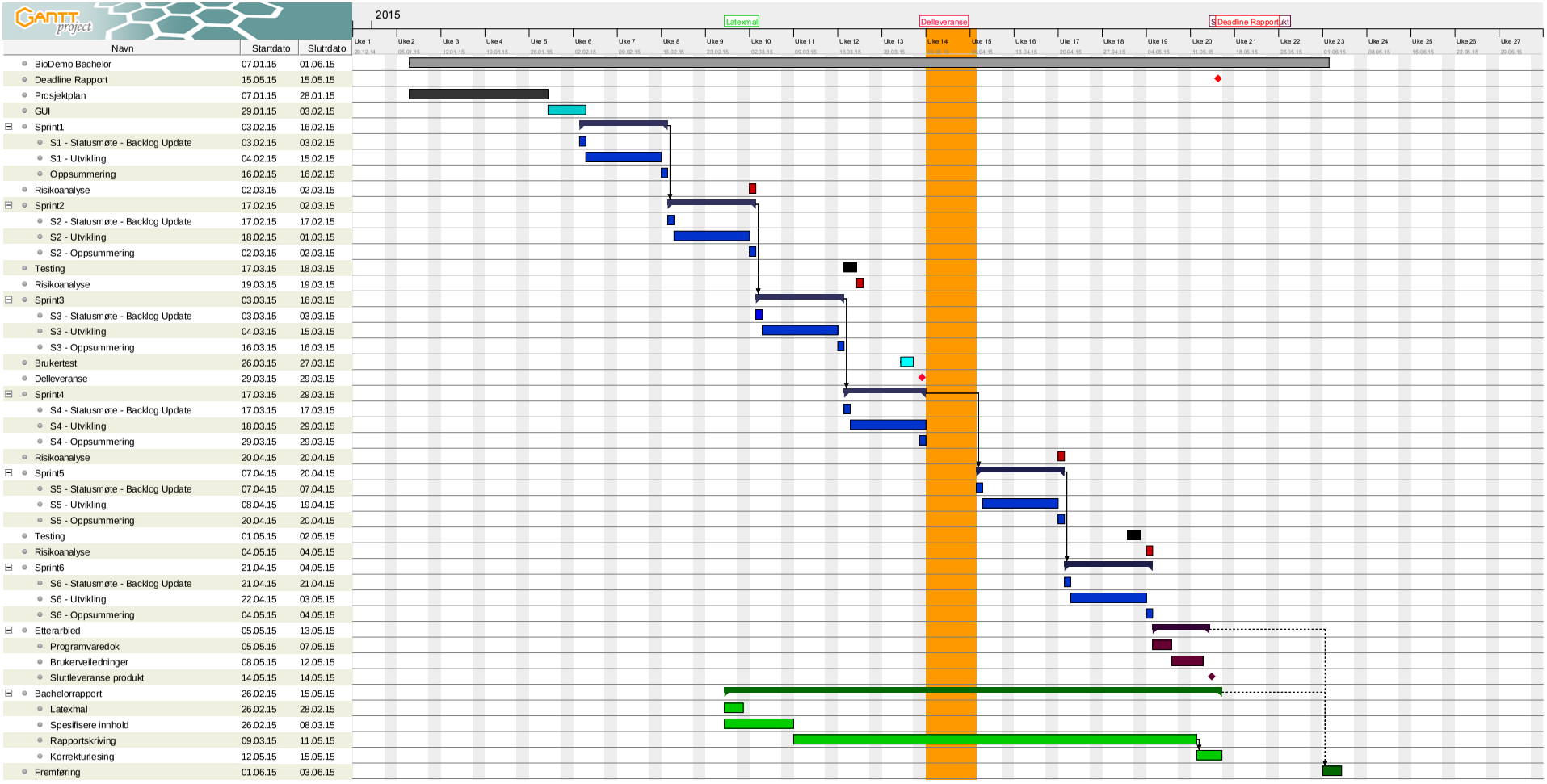
6 Plan for gjennomføring

6.1 Gantt-skjema

Figur 3 viser en plan for hele bachelorprosjektet. Vi regner med fra start den 05 januar 2015 til slutt den 03 juni 2015 da presentasjonene er satt. Som nevnt i del 4.1.1 bruker vi Scrum, noe som preger utformingen av planen vår fremover. I tillegg har vi lagt til noen ekstra aktiviteter som vi vil gjennomføre; som f.eks testing 2 ganger i perioden og risikoanalyse 4 ganger hvor 2 er i forbindelse med eventuelle funn fra testingen før. I tillegg har vi "skilt" ut selve rapportskrivingen fra programutviklingen, selv om disse to vil skje paraellt og ofte samtidig i samme sprint.

Viktige frister i denne planen er selvfølgelig selve rapport deadlinen den 15 mai, delleveringene som vi har satt til slutten av mars og ved sluttfasen av rapporten i midten av mai.

Ganttskjema



Figur 3: Gantt-skjema

7 Vedlegg

7.1 Grupperegler

7.1.1 Tidsarbeid

Det forventes at hvert gruppemedlem vil gjennomsnittlig benytte ca 30 timer i uka på prosjektet. Dette måles over tid f.eks. (25 t arbeid en uke medfører 35t en annen) og vil ikke være et absolutt antall. Gruppemedlemmer er ansvarlige for å lage bekymringsmeldinger når gruppemedlemmer ikke bidrar nok.

7.1.2 Kvaliteten på arbeidet

Gruppen har felles eierskap på det som blir produsert, og alle er ansvarlige for å bidra til arbeid holder høyt akademisk nivå både når det gjelder kodedelen og rapportdelen av prosjektet. Bekymringsmeldinger kan bli konsekvensen om noen leverer mangelfult arbeid og ikke har vilje eller evne til forbedring.

7.1.3 Bekymringsmeldinger

Om vedkommende som mottar gjentatte bekymringsmeldinger ikke viser tegn til forbedring, er dette grunnlag for å kontakte veileder og utkastelse vil bli en siste mulighet dersom samtale med veileder ikke løser konflikten.

7.1.4 Kontakttid og kontaktform

Kontakt over Skype og telefonsamtale kan kun forekomme mellom 10.00 – 21.00. Ellers kan kontakt over SMS og Facebook forekomme når som helst.

7.1.5 Møter

Gruppas medlemmer plikter å møte opp på faste møter, så sant ikke sykdom og vesentlige gode grunner. Om noen ikke møter opp på flere møter av ulike årsaker, er dette grunnlag for bekymringsmelding og skal tas opp i gruppen. Møteformen skal enten være i person eller over Skype. På møter utad av gruppen er alle ansvarlige for å være forberedt til, og det bør forbedres en agenda (trenger ikke å følges slavisk).

7.1.6 Beslutningskraft

To personer er nok til å ta en beslutning som angår gruppa hvis tredje medlem ikke er tilgjengelig eller tilstedet. I utgangspunktet skal man ta en beslutning i felleskap når denne angår flere, men gruppemedlemmer kan ta beslutninger som kun angår dem. Man må også kunne gå tilbake på en beslutning. Kontraktendringer kan fremkomme kun ved at to eller flere gruppemedlemmer er enig. Ved å skrive under på den originale gruppekontakten er man også med på eventuelle endringer.

Referanser

[1] The Norwegian Biometrics Laboratory [webpage]. NISLab HiG; 2015 [updated 08 des 2014; cited 20 Jan 2015]. Available from: http://www.hig.no/imt/research/nislab/biometrics_lab.

[2] Sommerville I. Software Engineering. 9th ed. Pearson;.

[3] JavaDoc Tool [article about JavaDoc]. Oracle; 2015 [updated 20 Jan 2015; cited 20 Jan 2015]. Available from: http://www.oracle.com/technetwork/articles/java/index-jsp-135444.html.

[4] Java SE Documentation [Documentation on Java]. Oracle; 1999 [updated April 1999; cited 20 Jan 2015]. Available from: http://www.oracle.com/technetwork/java/javase/documentation/codeconvtoc-136057.html.

[5] Google Java Style [Veiledning]. Google Inc; 2014 [updated 21 Mars 2014; cited 20 Jan 2015]. Available from: https://google-styleguide.googlecode.com/svn/trunk/javaguide.html.

[6] Oppgaveskriving [Veileding om akademisk oppgaveskriving]. Høgskolen i Gjøvik; 2014 [updated 04 Juni 2014; cited 20 Jan 2015]. Available from: http://www.hig.no/student/oppgaveskriving.

[7] Wikipedia contributors. Eclipse (software) [wikipage]. Wikipedia, The Free Encyclopedia; [cited 10 February 2015]. Available from: http://en.wikipedia.org/w/index.php?title=Eclipse_(software)&oldid=644278376.

[8] GIT vs SVN [webpage]. Codeforest; 2012 [updated 05 feb 2012; cited 27 Jan 2015]. Available from: http://www.codeforest.net/git-vs-svn.