Bacheloroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

Mikal Hjørungdal

# Mobilapplikasjon for å avdekke alvorlighetsgrad knyttet til brannskade

Bacheloroppgave i Dataingeniør
Veileder: Di Wu
Mai 2020

NTNU
Kunnskap for en bedre verden

Mikal Hjørungdal

# Mobilapplikasjon for å avdekke alvorlighetsgrad knyttet til brannskade

Bacheloroppgave i Dataingeniør
Veileder: Di Wu
Mai 2020

Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

NTNU
Kunnskap for en bedre verden

NTNU

Kunnskap for en bedre verden

# Bacheloroppgave

**IE303612 Bacheloroppgave Data**

**Mobilapplikasjon for å avdekke alvorlighetsgrad knyttet til brannskade**

Kandidater: 460044

Totalt antall sider inkludert forsiden: 55

Ålesund, 20.Mai 2020

NTNU

Kunnskap for en bedre verden

# Obligatorisk egenerklæring/gruppeerklæring

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

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

# Publiseringsavtale

**Studiepoeng: 20**

**Veileder: Di Wu**

## Fullmakt til elektronisk publisering av oppgaven

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

**Jeg/vi gir herved NTNU i Ålesund en vederlagsfri rett til å gjøre oppgaven tilgjengelig for elektronisk publisering:** ☒**ja** ☐**nei**

**Er oppgaven båndlagt (konfidensiell)?** ☐**ja** ☒**nei**
**(**Båndleggingsavtale må fylles ut)
- Hvis ja:
**Kan oppgaven publiseres når båndleggingsperioden er over?** ☐**ja** ☐**nei**

**Er oppgaven unntatt offentlighet?** ☐**ja** ☒**nei**
(inneholder taushetsbelagt informasjon. Jfr. Offl. §13/Fvl. §13)

**Dato: 20.05.20**

# FORORD

Denne rapporten er skrevet av en student i forbindelse med avsluttende utdanning for studieretningen dataingeniør ved institutt for informasjonsteknologi og elektroteknikk ved NTNU Ålesund.

Oppgaven er levert av en lege i spesialisering.
Oppgaven går ut på å lage en mobilapplikasjon som kan kalkulere kroppsoverflaten til en brannskadet person og hvor stor prosentandel av kroppen er brannskadet.

Jeg har valgt denne oppgaven siden det virket ut som at det kom til å bli en utfordring for meg sjøl både innen å lære et nytt programmerings språk og utvide lærdommen min innen app utvikling.

Målet med denne oppgaven er å lage en rask og enkel løsning som du kan ha tilgjengelig til enhver nødvendig tid dersom en brannskade oppstår. Applikasjonen er bygd for å kunne brukes med estimat fra bruker om nødvendig, men er i hovedsak designet for bruk på akuttmottaket pasienten kommer innpå for å få nøyaktig informasjon som appen trenger for kalkulasjonen.

Jeg ønsker å takke Eilif Folland Lønnebakke for muligheten til å kunne lage en app som kan være veldig viktig innen fagfeltet han spesialiserer seg i.

Jeg ønsker også å takke Di Wu ved NTNU Ålesund som har vært veileder. Hennes bidrag til oppgaven har vært nyttig da hun gjennom hele arbeidsprosessen har bidratt med interessante perspektiver, gode innspill og viktig erfaring. Dette har vært avgjørende for å få til et godt resultat. Takket være alle bidragsytere sitter jeg igjen med kunnskap om et interessant felt jeg ikke hadde stor kjennskap til tidligere.

# INNHOLD

# SAMMENDRAG

Formålet med denne oppgaven er å lage en mobilapplikasjon som gir en nøyaktig kalkulasjon av kroppsoverflate og brannskade. Applikasjonen gjør det også mulig for at du kan utføre denne prosessen i en nødsituasjon før eller når en ambulanse dukker opp til situasjonen, men er primært laget for bruk på akuttmottak.

Rapporten vil beskrive grunnlaget, fremgangen og sluttresultatet av bachelor oppgaven. Oppgaven er gitt av Eilif Folland Lønnebakke som er en lege i spesialisering.

Denne applikasjonen er dermed laget for et felt innenfor medisin som har hatt behov for bedre løsninger og mer utvikling. Denne mobilapplikasjonen skal være et bidrag til å løse noen av disse utfordringene. Dette ved å gjøre noen av de situasjonen som kan oppstå knyttet til brannskader mer forutsigbare og enklere å kunne anslå alvorlighetsgraden i en brannskade.

Målet var å utvikle en mobilapplikasjon som kan lokalisere området brannskaden befinner seg hurtig og presist. Med denne informasjonen blir der gitt umiddelbar tilbakemelding til brukeren om hva som er alvorlighetsgraden til skaden.

Utviklingen av produktet har vært gjort i samarbeid med oppdragsgiver og veileder ved studieinstitusjonen. Men med mindre innspill fra oppgavegiver grunnet Covid-19 situasjonen som pågikk under prosjekt tiden.

# TERMINOLOGI

## *Begreper*

| | |
|---|---|
| Flutter | Flutter er et open source UI-programvareutviklingssett laget av Google. Den brukes til å utvikle applikasjoner for Android, iOS, Windows, Mac, Linux, Google Fuchsia og nettet. (Flutter u.d.) |
| Dart | Dart er et klientoptimalisert programmeringsspråk for apper på flere plattformer. Den er utviklet av Google og brukes til å bygge mobil-, desktop-, server- og webapplikasjoner. Dart er et objektorientert, klassebasert, søppelsamlet språk med syntaks i C-stil. Dart kan kompilere til enten innfødt kode eller JavaScript. (Wikipedia 2020) |
| Kunstig intelligens | Kunstig intelligens omfatter alle intelligente systemer. Et grovt skille kan trekkes mellom *regelbaserte modeller* (ekspertsystemer) og *datadrevne modeller* (maskinlæring). Regelbaserte modeller forstår begreper gjennom regler, som ofte er programmerte før modellen brukes. I datadrevne modeller lærer maskinen i stedet for å bli programmert. (Tidemann 2020) |
| Applikasjon | Program, en serie instruksjoner som forteller en datamaskin hva den skal gjøre. Et dataprogram kan integreres i selve maskinvaren, eller i en uavhengig form som må lastes inn i en datamaskins hukommelse og «kjøres». (Eirik Rossen 2019) |
| Data | I daglig tale kjensgjerninger, faktiske opplysninger, for eksempel personlige data som alder, høyde og liknende. I forbindelse med databehandling betegner data enhver fysisk representasjon av opplysninger, viten, meninger etc. i motsetning til innholdet, som kalles informasjon. (papirleksikonet 2018) |

| | |
|---|---|
| API | API, hjelpeverktøy ved programmering, et grensesnitt mot en eller flere tjenester i et operativsystem, en databasetjener eller lignende. (Rossen, snl.no 2019) |
| UI | Brukergrensesnitt, betegnelse på kontaktflaten mellom brukeren og datamaskinens operativsystem og programmer, avgjør hvordan brukeren styrer programmene. (Rossen, snl.no 2019) |
| Hacking | En hacker er en person som er i stand til å løse problemer og utfordringer ved hjelp av informasjonsteknologi på svært kort tid. En hacker er god på å *hacke*. I motsetning til tradisjonell, disiplinert teknologiutvikling, består dette i å bygge opp raske, snedige løsninger som fungerer der og da, uten å ta hensyn til andre ting, for eksempel om de vil fortsette å fungere i fremtiden eller om de fungerer sammen med andre komponenter. (Dvergsdal 2019) |
| Semantikk | Semantikk er læren om språkets innhold, sammenhengen mellom ord, fraser og setninger og deres betydning eller mening. (Hanne Gram Simonsen 2019) |
| Syntaks | Regler for hvordan en kodelinje eller en kommando i et programmeringsspråk eller et skriptspråk skal settes sammen. Alle dataspråk er konstruert slik at én og samme melding må være entydig. Brytes syntaksreglene, kan kommandoen eller kodelinjen ikke tolkes. Program med syntaksfeil vil ikke kunne kjøres. Kompilerte språk gir melding om syntaksfeil under kompilering. Tolkede språk og skriptspråk gir gjerne slike tilbakemeldinger under kjøring. (papirleksikonet 2009) |
| Open source | Åpen kildekode viser til programvare som distribueres under forutsetning av at kildekoden skal være tilgjengelig for brukerne. Åpen kildekode skiller seg fra programvare som utelukkende distribueres i binærform, hvor kildekoden i de fleste tilfeller er opphaverers hemmelighet. (Gramstad 2020) |

| | |
|---|---|
| Boolean | Boolean, eller boolean logikk, er en undergruppe av algebra som brukes til å lage sanne / falske utsagn. Boolske uttrykk bruker operatørene AND, OR, XOR og NOT for å sammenligne verdier og gi et sant eller falskt resultat. (Christensson, Techterms.com 2011) |
| Widget | En widget er et element i et grafisk brukergrensesnitt (GUI) som viser informasjon eller gir en spesifikk måte for en bruker å samhandle med operativsystemet eller et program. (Margaret Rouse 2015) |

## *Forkortelser*

| | |
|---|---|
| IDE | Integrated development environment |
| App | Application (Applikasjon) |
| API | Application programming interface |
| Sql | Structured Query Language |
| GPU | Graphic processing unit (Grafikkprosessor) |
| CPU | Central processing unit |
| RAM | Random-access memory |
| AI | Artificial intelligence (Kunstig intelligens) |
| UI | User interface (Brukergrensesnitt) |

# 1 INNLEDNING

## *1.1 Oppgaven*

Oppdragsgiveren ønsket å få utviklet en App som presist kan kalkulere kroppsoverflate (Total Body Surface Area) og hvor stor prosentandel av kroppen som er forbrent i en brannskade.

Oppgaven er utført av Mikal Olai Hjørungdal ved NTNU Ålesund våren 2020 etter kravspesifikasjoner gitt av Eilif Folland Lønnebakke.

## *1.2 Rammebetingelser*

Appen har ingen andre krav enn at den skal kunne brukes på mobil. Det blir ikke tatt høgde for problemstillinger som dårlig dekning, at applikasjonen er nedlastet på forhånd eller tilsvarende situasjoner.

## *1.3 Problemstilling*

Problemstillingen i denne oppgaven var å finne en løsning på hvordan et mobil-kamera skal kunne skanne en person og kunne se forskjellen på uskadet hud, klær og brannskade.

Løsningen må være enkel for brukeren å bruke og må kunne få nøyaktige resultat. Appen må også være ryddig, enkel og brukervennlig uten avanserte forklaringer.

Uten noen spesifikasjon om hvilke plattformer Appen skal støtte så er konklusjonen at Appen burde være tilgjengelig for alle leger, helsepersonell i nødetaten og spesialister i redningstjenesten. Det er to rådende operativsystemer for mobile enheter som den burde utvikles for, Android og IOS.

Den største problem stillingen er at der ikke finnes andre løsninger som bruker en mobil til å skanne en person uten eksterne maskiner eller avanserte database AI-løsninger. Dette må dermed bli utviklet med lite data tilgjengelig som hjelpemiddel.

Dette er problemstillinger jeg har kommet frem til gjennom oppgavearbeidet og etter møter med oppgavegiver og veileder.

## *1.4 Mål*

Målet er å levere en løsning som tilfredsstiller kravene til oppgavegiver. Det må være en App som er funksjonell og lett å navigere.

## *1.5 Kravspesifikasjoner fra oppdragsgiver*

- Det må være en app som er lett å bruke
- Kan utføre nøyaktig og raske svar på hvor stor en skade er
- Appen må kalkulere ved bruk av kjønn, vekt, høyde, alder og fasong

Siden oppgavegiver ikke har oppgitt mange krav, legger jeg til flere krav og strukturerer dem for å ha et mer strukturert prosjekt og flere arbeids mål.

### 1.5.1 Mobil Plattformer

Siden appen skal brukes av leger og muligens sivile så er plattformene fleksible, dermed blir appen utviklet til å fungere på hovedsakelig Android og IOS. Der eksisterer også løsninger for bruk på Windows Phone om nødvendig.

### 1.5.2 Brukervennlig grensesnitt

Appen må være strukturert på en ryddig og forstårlig måte sånn brukeren kan lett fylle ut alle felt nødvendig for å utføre kalkulasjonen.

### 1.5.3 Konstruksjon av appen

Appen må kunne operere raskt og kalkulere så raskt som mulig.

Koden må være ryddig og forståelig for meg sjøl og fremtidige oppgraderinger fra tredje parti om det skal være nødvendig.

# 2 TEORETISK GRUNNLAG

## *2.1 Mobilapplikasjon*

**Mobilapplikasjon** (Technopedia 2018) (**app**, **mobilapp**, **miniprogram**) er et dataprogram som kan installeres på smarttelefon, nettbrett eller andre mobile enheter. Programmene kan vanligvis lastes ned fra et nettsted som drives av utvikleren av mobiltelefonoperativsystemet, hvorav de største er Apple App Store og Google Play (J.Clement 2020). Mange apper er tilgjengelig for gratis nedlastning, mens andre kjøpes gjennom den aktuelle nettbutikken. Mange apper har to versjoner, der en må du betale for, mens den andre er gratis – på bekostning at den kommer med reklame og/eller begrensede funksjoner. Dette er en strategi som brukes av noen utviklere for å senke terskelen for å teste ut en applikasjon. (Dev 2019)

Mobilapplikasjoner som er utviklet som webapplikasjoner vil kunne fungere på flere forskjellige typer mobiltelefonoperativsystem uten tilpassing. Apper som er programmert og kompilert for et bestemt mobiloperativsystem (såkalt "native app") vil bare kunne installeres på eget operativsystem, med mindre de er utviklet i programmeringsverktøy som har støtte for å kompilere versjoner for flere mobilsystemer.

## *2.2 Brukergrensesnitt*

Utvikling av nettsider handler om å skape en god brukeropplevelse (Aaberge 2017). Det vil si at brukerne av et nettsted skal forstå strukturen og klare å orientere seg, slik at de finner det de leter etter.

Når nettsider eller app-er for mobile enheter blir planlagt, er det to innfallsvinkler som er vanlige å ha.

- Det ene er å være opptatt av hvordan brukeren navigerer, beveger seg og finner fram. Målet er å gi en opplevelse av flyt og at brukeren ikke føler seg stoppet i systemet.
- Det andre er å ha fokus på utseende. Hvilke farger og skrifttyper skal brukes? Hvilken stil skal det være på bilder og video? Det er viktig å sikre at nettstedet har et design som samsvarer med formålet, bedriften eller organisasjonens profil.

## *2.3 Universell utforming*

**Universell utforming** (Norskdesign 2010) er utforming av produkter og omgivelser på en slik måte at de kan brukes av alle mennesker, i så stor utstrekning som mulig, uten behov for tilpassing og en spesiell utforming.

Design for alle er en strategi for design og brukersentret innovasjon. Hensikten er god og brukervennlig design som tar utgangspunkt i en brukerforståelse basert på mangfold. En brukerfokusert designprosess bidrar til nyskaping og resulterer i mer brukervennlige løsninger for en utvidet kundegruppe.

Foretak som er proaktive i forhold til nye krav og direktiver innen universell utforming kan oppnå økt konkurransekraft i et globalt marked og samtidig bidra til et mer inkluderende samfunn.

### 2.3.1 De syv prinsippene for universell utforming

En gruppe amerikanske arkitekter, produktdesigners, ingeniører og forskere har utarbeidet syv prinsipper for universell utforming. (Norskdesign.no 2007)

1. **Enkel og intuitiv i bruk** – Utformingen skal være lett å forstå uten hensyn til brukerens erfaring, kunnskap, språkferdigheter eller konsentrasjonsnivå.
2. **Forståelig informasjon** – Utformingen skal kommunisere nødvendig informasjon til brukeren på en effektiv måte.
3. **Toleranse for feil** – Utformingen skal minimalisere farer og skader som kan gi ugunstige konsekvenser, eller minimalisere utilsiktede handlinger.
4. **Like muligheter for alle** – Utformingen skal være brukbar og tilgjengelig for personer med ulike ferdigheter.
5. **Fleksibel i bruk** – Uansett individuelle preferanser og ferdigheter. Den synshemmede skal kunne høre, den hørselshemmede se og så videre.
6. **Lav fysisk anstrengelse** – Utformingen skal kunne brukes effektivt og bekvemt med minimum besvær.

7. **Størrelse og plass for tilgang og bruk** – Hensiktsmessig størrelse og plass skal muliggjøre tilgang, rekkevidde, betjening og bruk, uavhengig av brukerens kroppsstørrelse, kroppsstilling og mobilitet.

## *2.4 Metoder*

Programvareutvikling (Doshi 2019) refererer til en iterativ logisk prosess som tar sikte på å lage en programmert programvare for å oppfylle unike forretnings- eller personlige mål, mål eller prosesser. Målet oppnås ved at en programvareutvikler skriver datakode. Imidlertid innebærer det også flere trinn som forskning, utforming av data og prosessflyt, skriving av teknisk dokumentasjon, omfattende testing, feilsøking og pressing av iterativt til å leve. Denne prosessen er kjent som programvaren utvikling livssyklus.
Det er flere metoder tilgjengelige for programvareprosjektledelse som Agile, Waterfall, Scrum, Kanban, og noen få andre.

### 2.4.1 Agile

"Agile programvare utvikling" (Kent Beck u.d.) refererer til en gruppe programvareutviklingsmetodologier basert på iterativ utvikling, der krav og løsninger utvikler seg gjennom samarbeid mellom selvorganiserende tverrfunksjonelle team. Begrepet ble myntet i 2001 da Agile Manifesto ble formulert.

Agile programvareutvikling bruker iterativ utvikling som grunnlag, men tar til ordre for et lettere og mer menneske sentrisk synspunkt enn tradisjonelle tilnærminger. Agile prosesser inkluderer grunnleggende iterasjon og kontinuerlig tilbakemelding som den gir for å videreutvikle og levere et programvaresystem.

### 2.4.2 Waterfall

Waterfall (Powell-Morse 2016) er en programvareutviklingsprosess. Fossemodellen understreker at det tas en logisk progresjon av trinn gjennom programvarenes livssyklus (SDLC), omtrent som de overlappende trinnene nedover en inkrementell foss. Mens populariteten til fossefallsmodellen har avtatt de siste årene til fordel for mer smidige metoder, kan den logiske karakteren av den sekvensielle prosessen som brukes i fossefallmetoden ikke nektes, og det er fortsatt en vanlig designprosess i industrien.

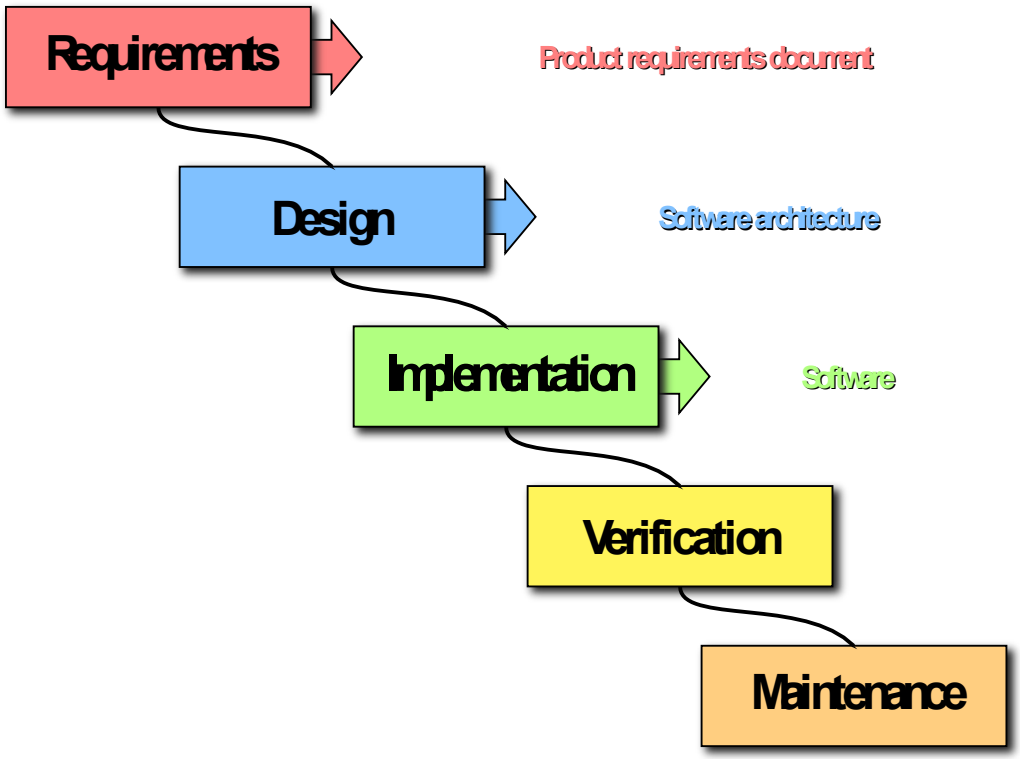
"The six stages of falling water":

- Krav
- analyse
- Design
- Koding
- Testing
- Distribusjon (eller installasjon)

Fordelene med å bruke Waterfall:

- tilpasser seg til nye grupper: Selv om det ikke nødvendigvis bare er spesifikt for Waterfall modellen, gjør bruk av waterfall at prosjektet som helhet kan opprettholde et mer detaljert, robust omfang og designstruktur på grunn av alle planleggings- og dokumentasjonsfaser på forhånd.
- Tvinger strukturert organisering: Selv om noen kan hevde at dette er en byrde snarere enn en fordel, gjenstår faktum at Waterfall modellen tvinger prosjektet, og til og med organisasjonsbygget til det nevnte prosjektet, til å være ekstra disiplinert i sin utforming og struktur. De fleste betydelige prosjekter vil, etter behov, inneholde detaljerte prosedyrer for å styre alle aspekter av prosjektet, fra design og utvikling til testing og implementering.
- Gjør det mulig for tidlige designendringer: Selv om det kan være vanskelig å gjøre designendringer senere i prosessen, egner Waterfall seg godt til endringer tidlig i livssyklusen.
- Passer for fokusert utvikling: På grunn av den iboende lineære strukturen til et Waterfall prosjekt, er slike applikasjoner alltid godt egnet for organisasjoner eller team som fungerer godt under et målsatt og datofokusert prosjekt. Med klare, konkrete og godt forståtte stadier som alle på teamet kan forstå og forberede seg på, er det relativt enkelt å utvikle en tidslinje for hele prosessen og tildele bestemte markører og milepæler for hvert trinn og til og med fullføring.

Den første formelle beskrivelsen av metoden siteres ofte som en artikkel publisert av Winston W. Royce i 1970, selv om Royce ikke brukte begrepet "waterfall" i denne artikkelen. (W. W. Royce 1970)



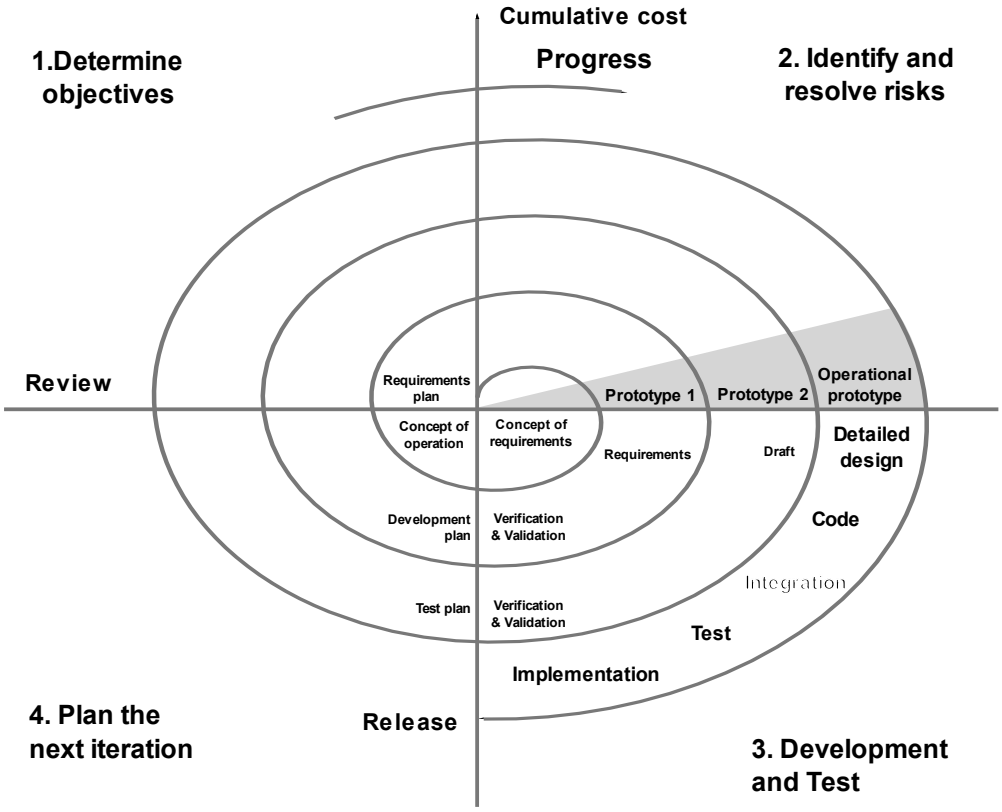

*Figur 1: Waterfall modell (Peter kemp 2010)*

### 2.4.3 Spiral

Etter at Barry W. Boehm (Ionos 2019) for første gang presenterte sitt konsept for utvikling av komplekse applikasjoner i 1986, publiserte den amerikanske programvareingeniøren sin modell i 1988 i publikasjonen "A Spiral Model of Software Development and Enhancement" i et mer omfattende rammeverk. I publikasjonen beskriver han spiralmodellen som et mulig alternativ til den tidligere etablerte Waterfall modellen, som også fungerte som grunnlag for opplevelse. I motsetning til fossemodellen, antar spiralmodellen ikke at oppgavene til programvareutvikling er designet lineært - men ser dem som iterative oppgaver. I spiralmodellen kjøres derfor ikke fasene en gang trinnvis, men flere ganger i spiralform. Selv om denne konjunkturrepetisjonen gjør at prosjektet nærmer seg målene satt relativt langsomt, minimeres risikoen for en mislykket utviklingsprosess avgjørende takket være de jevnlige kontrollene.

Spiral modellen bruker fire sykluser:

- Bestem mål og alternativer og beskriv rammebetingelser
- Identifisere og løse risikoene
- Utvikle og teste alternativer
- Planlegge neste syklus




*Figur 2: Spiral modell (Boehm 1988)*

## *2.5 Brukertesting*

**Brukertesting** er en evaluering der man observerer og analyserer hvordan funksjoner i en løsning blir brukt av faktiske brukere.

### 2.5.1 Om brukertesting

Brukertesting (Digitaliseringsdirektoratet 2015) brukes for å sikre at nye løsninger fungerer, men kan også gi grunnlag for å forbedre eksisterende løsninger.

Hvis en har kommet fram til at en digital tjeneste, et skjema, standardbrev, nettsidene eller virksomhetens nye app mv. ikke fungerer godt nok og dessuten generer ekstra henvendelser, kan brukertesting være med på å avdekke hva som bør forbedres.

Enkelte brukertester kan virksomheten enkelt gjennomføre selv, mens det vel er mer vanlig at mer omfattende brukertester av f.eks. digitale tjenester settes ut til profesjonelle firmaer. Da gjennomføres alt fra enkle brukstester på et sted der brukerne vanligvis befinner seg når de utfører tjenesten, eller ved at ulike typer brukere inviteres til testlaboratorium der de f.eks. studerer bruksmønster, intervjuer etc.

I likhet med alle andre datainnsamlingsmetoder er det også her viktig å dokumentere hvordan testen gjennomføres, lage oppsummeringer og analysere testresultatene.

### 2.5.2 Test metode

Ved brukertesting er det viktig at flere testpersoner tester tjenesten. Anbefalt antall varierer fra 5-10 testbrukere (Nielsen 2000). Kjører man flere iterasjoner og flere brukertester i løpet av utviklingsprosessen, kan man tillate seg å gå ned på antall testpersoner. Jacob Nielsen anbefaler å teste med 5 testpersoner, men å gjennomføre flere brukertester i løpet av utviklingsløpet. Begrunnelsen er at når et par tre brukere har avdekket alvorlige svakheter med tjenesten er det mer hensiktsmessig å utbedre disse, enn å bruke ressurser på å teste flere brukere.

## *2.6 Database*

En database (Christensson, techterms.com 2009) er en datastruktur som lagrer organisert informasjon. De fleste databaser inneholder flere tabeller, som hver kan inneholde flere forskjellige felt. For eksempel kan en bedriftsdatabase inneholde tabeller for produkter, ansatte og økonomiske poster. Hver av disse tabellene vil ha forskjellige felt som er relevante for informasjonen som er lagret i tabellen.

Nesten alle e-handelsnettsteder bruker databaser for å lagre produktbeholdning og kundeinformasjon. Disse nettstedene bruker et databasesystem (eller DBMS), for eksempel Microsoft Access, FileMaker Pro eller MySQL som "backend" til nettstedet. Ved å lagre nettstedsdata i en database, kan data enkelt søkes, sorteres og oppdateres. Denne fleksibiliteten er viktig for e-handelsnettsteder og andre typer dynamiske nettsteder.

Tidlige databaser var relativt "flate", noe som betyr at de var begrenset til enkle rader og kolonner, som et regneark. Imidlertid gir dagens relasjonsdatabaser brukere tilgang til å oppdatere og søke informasjon basert på forholdet mellom data lagret i forskjellige tabeller. Relasjonsdatabaser kan også kjøre spørsmål som involverer flere databaser. Mens tidlige databaser bare kunne lagre tekst eller numeriske data, lar moderne databaser også brukere lagre andre datatyper som lydklipp, bilder og videoer.

## *2.7 Sikkerhet*

Mobilappsikkerhet (FraudWatch International 2019) er målet til å forsvare apper mot digital svindel i form av skadelig programvare, hacking og annen kriminell manipulering. Mobilappsikkerhet kan implementeres med teknologiske midler, personlige responser og bedriftsprosesser som er ment å sikre digital integritet på mobile enheter.

Imidlertid er mobilapper en stor kanal for sikkerhetstrusler, spesielt når de er koblet til bedrifter. Disse er ofte målrettet mot kriminelle elementer som søker å tjene penger på bedriftene og ansatte som bruker mobile enheter, men ikke driver med riktige sikkerhetsprosesser for mobilapper.

### 2.7.1 Beste praksis for mobil applikasjonssikkerhet

- Opplæring i digital sikkerhet
- Proaktivt overvåke for useriøse apper
- Last ned fra pålitelige kilder
- Forbedre datasikkerhet
- Unngå å lagre passord

### 2.7.2 Sosial Manipulering

Sosial manipulering (Nettvett 2020) utnytter menneskelig kontakt og sosiale evner for å få tak i eller påvirke informasjon. Vi snakker her om "menneskelig informasjonssvindel". Forretningshemmeligheter, personopplysninger og informasjon om IT-systemer kan være verdifulle for andre og misbrukes til svindel eller kriminelle handlinger.

### 2.7.3 Hashing

Hashing (Rouse 2005) er forandring av et ord til en kortere fast lengdeverdi eller status som representerer det opprinnelige ordet. Hashing brukes til å indeksere og hente gjenstander i en database fordi det er raskere å finne elementet ved hjelp av den kortere hashen enn å finne det ved å bruke den opprinnelige verdien. Det brukes også i mange krypteringsalgoritmer.

### 2.7.4 Salting

I kryptografi (Svein Johan Knapskog 2019) er et salting (Fernandez 2019) en sekvens av bytes som blir lagt til passordet før det blir fordøyd. Dette gjør fordøyelsene våre forskjellige fra hva de ville være hvis vi krypterte passordet alene, og som et resultat beskytter oss mot angrep fra ordbøker. Vi kan ta i bruk to forskjellige strategier angående salt:

- Bruk et fast salt, en sekvens med bytes som vi vil bruke til å fordøye hvert passord. Vi kan holde dette saltet skjult og betrakte det som en ekstra sikkerhetsverdi, men det kan gjøre systemet vårt mer sårbart for bursdagsangrep og generelt angrep drevet mot vår database med passord som helhet.
- Bruk et variabelt salt, som vanligvis er et tryggere alternativ (bedre hvis det er tilfeldig). Dette genereres eller beregnes separat for hvert passord som blir fordøyd, og det gjør at hvert lagret passord kan kobles fra de andre, noe som skaper en sterkere totalbeskyttelse og forbedrer sikkerheten for angrep drevet mot vår database med passord som helhet.

I praksis er tilfeldig (eller i det minste variabelt) salt en mye bedre ide, selv om det er tilfeldig vil det tvinge oss til å lagre det ukryptert sammen med fordøyelsen (slik at vi kan gjenopprette det), og dette vil gjøre det trivielt for en angriper for å vite det, det vil fortsatt la hvert brukers passord forbli avkoblet fra resten, slik at de må angripes separat.

### 2.7.5 SQL Injection

SQL-injeksjon (W3Schools u.d.) er en kodeinjeksjonsteknikk som kan ødelegge databasen din. Det er en av de vanligste teknikkene for netthacker. SQL-injeksjon er plassering av ondsinnet kode i SQL-setninger, via inndata på websiden.

SQL-injeksjon skjer vanligvis når du ber en bruker om inndata, som brukernavnet, og i stedet for et navn eller id, gir brukeren deg en SQL-setning som du ubevisst vil kjøre i databasen.

## *2.8 Programmeringsspråk*

Programmeringsspråk (Rossen, programmeringsspråk 2019), konstruert språk til å skrive programmer til datamaskiner.

Et programmeringsspråk omfatter ulike typer grunnleggende kommandoer som en datamaskin kan utføre. Ord i et programmeringsspråk er alltid entydige, og de er gjerne hentet fra grener som matematikk og logikk.

Syntaksen i et programmeringsspråk må følges nøye. Det minste avvik fra «grammatikken» fører til at programmet går i stå eller datamaskinen gjør feil.

### 2.8.1 SQL

SQL (Structured Query Language) (Encylopedia Britannica u.d.) er et dataspråk designet for å få fram informasjon fra databaser.

SQL fungerer ved å gi en måte for programmerere og andre databrukere å få ønsket informasjon fra en database ved å bruke noe som ligner vanlig engelsk. På det enkleste nivået består SQL av bare noen få kommandoer: Velg, som henter data; Sett inn, som legger til data i en database; Oppdatering, som endrer informasjon; og Delete, som sletter informasjon. Andre kommandoer eksisterer for å opprette, endre og administrere databaser.

SQL brukes i alt fra regjeringsdatabaser til nettsteder på e-handel. Etter hvert som SQLs popularitet vokste, fortsatte programmerere og informatikere å optimalisere måten relasjonsdatabaser fungerer på.

### 2.8.2 Flutter

Flutter (Wikipedia 2020) er et open source UI-programvareutviklingssett laget av Google. Den brukes til å utvikle applikasjoner for Android, iOS, Windows, Mac, Linux, Google Fuchsia og nettet.

Den første versjonen av Flutter hadde kodenavnet "Sky" og kjørte på Android-operativsystemet. Det ble avduket på toppmøtet for Dart-utvikleren i 2015. (Flutter u.d.)

#### 2.8.2.1 Dart

Flutter-apper er skrevet på Dart-språket (Dart 2013) og bruker mange av språkets mer avanserte funksjoner.

Dart (Wikipedia 2020) er et klientoptimalisert programmeringsspråk for apper på flere plattformer. Den er utviklet av Google og brukes til å bygge mobil-, desktop-, server- og webapplikasjoner.
Dart er et objektorientert, klassebasert, søppelsamlet språk med syntaks i C-stil. Dart kan kompilere til enten innfødt kode eller JavaScript. Den støtter grensesnitt, mixins, abstrakte klasser, tingliggjøring genetikk og type inferens. (Leler 2017)

## *2.9 Backup*

Sikkerhetskopi (Liseter 2014), kopi av programmer og data fra et datamaskinsystem, tatt kontinuerlig f.eks. ved at alle data lagres på forskjellige lagringsenheter, daglig eller med jevne mellomrom. Hensikten er at programmer og data skal være mest mulig intakte og tilgjengelige for fortsatt drift dersom systemets ordinære lagringsenheter skulle svikte eller data bli slettet, f.eks. som følge av virusangrep eller feil. Medier for sikkerhetskopier kan være andre disker, magnetbånd eller optiske enheter som CD-er eller DVD-er. Diskspeiling er en form for sikkerhetskopiering der systemet alltid sørger for at alle endringer på én disk umiddelbart kopieres til en annen, slik at de to diskene alltid har identisk innhold.

# 3 MATERIALER OG METODE

## *3.1 Metode*

### 3.1.1 Utviklingsmetode

Under utviklingen av prosjektet har jeg brukt en simplifisert variant av spiral metoden. Arbeidet ble planlagt med 14 dagers iterasjoner. Alle iterasjoner startet med oppdatering på forrige iterasjon deretter en 14 dagers plan på hva som skal videreutvikles på og hvilke nye funksjoner som må implementeres.

Før første iterasjon ble det holdt et møte mellom meg og veileder. I dette møtet laget vi en plan for hvordan fremtidige møter skulle settes opp siden prosjektgiver flyttet fylke og var for det meste utilgjengelig for faste samtaler og møter.

Vi laget en startfase for prosjektet og senere hadde jeg en privat samtale med prosjektgiver. Etter første møte fikk jeg masse innspill fra veileder der vi planla en basis for hva som burde være et bra startpunkt og hvilke spørsmål som burde formidles til prosjektgiver for å danne et klart innsyn om hva han tenker på.

Etter første møte med veileder kontaktet jeg prosjektgiver via telefon der han fikk formidle hva han forventet og tenkte, hvordan denne Appen kom til å fungere. Jeg fikk også muligheten til å spørre spørsmål. Jeg formidlet også hva veileder, og jeg hadde diskutert for å høre hans perspektiv om hva vi tenkte om første iterasjon.

Siden prosjektgiver ikke var mulig å få kontakt med i vårt første planlagte møte, ble da tankene og kravene til prosjektgiver tatt opp med veileder på et nytt raskt møte for å finne ut hvilke forandringer som burde bli gjort i startplanen.

Hver iterasjon startet med oppdatering angåendes forrige iterasjon og vurdering over arbeidet som hadde blitt utført. Deretter dokumenterte jeg ned tanker for hva som burde forandres og legges til. På møtene flyttet vi eller slettet noen planer, enten fordi arbeidet ville dele opp prosjektet for mye eller for at det ikke passet inn i henhold til kravene fra prosjektgiver eller mine egne krav.

Vi startet første iterasjon med hvilken programvare som passet best til å lage denne appen, deretter et basis design for å vise om progressen lignet på det prosjektgiver så for seg. Prosjektgiver ga tilbakemelding som ble brukt til å videre utvikle en prototype. Disse tilbakemeldingene ble sendt via e-post, med et telefonmøte om mulig. I slutten av arbeidsperioden ble et møte mellom meg og veileder opprettet der jeg informerte hva prosjektgivers tanker var og hvilke forandringer han forventet. Dersom der var forandringer ble disse gjennomført i neste iterasjon.

### 3.1.2 Programmeringsspråk

Prosjektet er en mobilapplikasjon og ble dermed laget i Android studio med programvareutviklings verktøyet Flutter som er laget for å bruke Dart (Dart 2013). Dart er et programmeringsspråk for apper på flere plattformer.

Valget av Flutter er for at det støtter IOS og Android med en samlet kode. Dette eliminerer den normale situasjonen der to forskjellige prosjekt blir laget for hver mobil plattform.

## *3.2 Verktøy*

### 3.2.1 Android studio

Android Studio (Android u.d.) er det offisielle integrerte utviklingsmiljøet (IDE) for Googles Android-operativsystem, bygget på JetBrains 'IntelliJ IDEA-programvare og designet spesielt for Android-utvikling. Det er tilgjengelig for nedlasting på Windows, macOS og Linux-baserte operativsystemer. Det er en erstatning for Eclipse Android Development Tools (ADT) som den primære IDE for naturlig Android-applikasjonsutvikling. Støtter Android Virtual Device (Emulator) for å kjøre og feil søke apper i Android-studio. (Wikipedia 2020)

#### 3.2.1.1 Simulering plattform

En Android Virtual Device (AVD) (Android u.d.) er en konfigurasjon som definerer egenskapene til en Android-telefon, nettbrett, Wear OS, Android TV eller Automotive OS-enhet som du vil simulere i Android Emulator. AVD Manager er et grensesnitt du kan starte fra Android Studio som hjelper deg å opprette og administrere AVD-er.

En AVD inneholder en maskinvareprofil, systembilde, lagringsområde, utseende (skins) og andre egenskaper.

**Maskinvareprofilen** definerer egenskapene til en enhet som sendes fra fabrikken. AVD Manager leveres forhåndsinnlastet med visse maskinvareprofiler, for eksempel Pixel-enheter, og du kan definere eller tilpasse maskinvareprofilene etter behov.

Et **systembilde** merket med Google APIer inkluderer tilgang til Google Play-tjenester.

AVD har et eget **lagringsområde** på din utviklingsmaskin. Den lagrer enhetsbrukerdata, for eksempel installerte apper og innstillinger, samt et emulert SD-kort. Om nødvendig kan du bruke AVD Manager til å slette brukerdata, slik at enheten har de samme dataene som om de var nye.

Et **emulatorutseende** spesifiserer utseendet til en enhet. AVD Manager gir noen forhåndsdefinerte «skins». Du kan også definere dine egne, eller bruke skins levert av tredjepart.

### 3.2.2 Microsoft office 365

Microsoft office 365 (Microsoft u.d.) er en pakke med programvare som inneholder et bredt utvalg av produkter. Disse produktene tilbyr ulike løsninger til privat bruk og bedrifter. Programmer inkludert i office pakken er: Word, Excel, PowerPoint, OneNote, OneDrive, Outlook, Teams.

Word og OneDrive ble brukt for å lage denne rapporten.

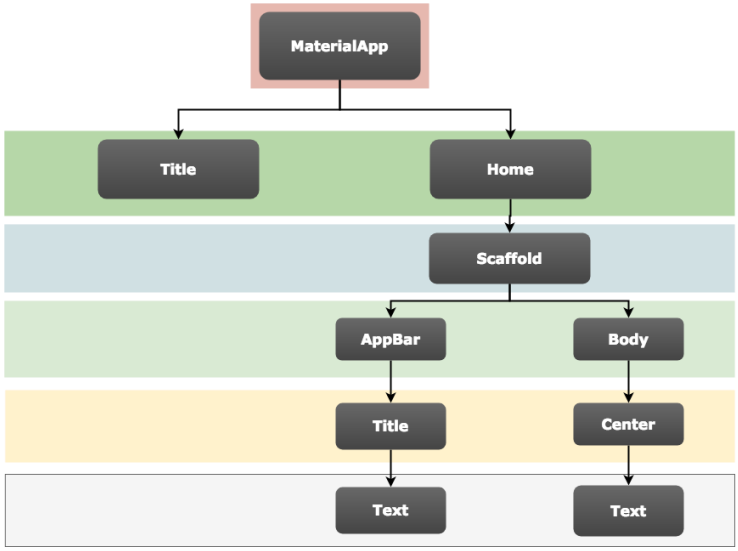
### 3.2.3 Dart.dev

Nettverks plattform brukt for å finne informasjon om dart og testing av kode. (Dart 2013)

### 3.2.4 Flutter

Flutter (Flutter u.d.) er et programvareutviklingssett som er designet rundt programmeringsspråket Dart. Dette er et verktøy som bruker et «Widget»-system. Widgets er de grunnleggende byggesteinene i en Flutter-apps brukergrensesnitt. Hver widget er en uforanderlig erklæring om en del av brukergrensesnittet. I motsetning til andre rammeverk som skiller visninger, visningskontrollere, oppsett og andre egenskaper, har Flutter en konsistent, enhetlig objektmodell: widgeten.

Best representert med et bilde:




*Figur 3: Widget hierarchy (Gupta 2018)*

## 3.3 Testing

For testing av applikasjonen ble egen maskin brukt. Dette er for Android studio har muligheten for å simulere en virtuell telefon ved bruk av prosessoren til pc-en. Dette er en mer funksjonell og raskere metode til å teste, enn ved bruk av egen telefon. Dette på grunn av overføring av data og feil som kan oppstå i Android studio når apper blir lastet ned til en enhet.

Maskin som ble brukt har spesifikasjoner:

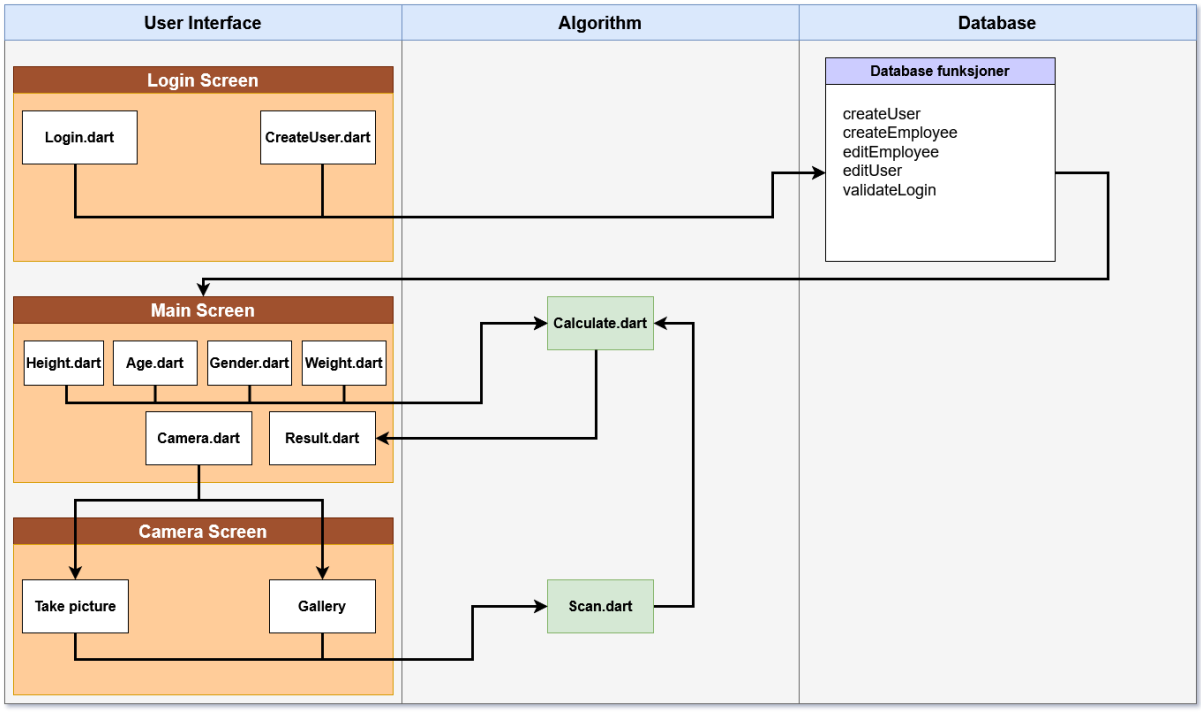
*Tabell 1: Pc spesifikasjoner for bruk for testing*

| | |
|---|---|
| Operativ system | Windows 10 Pro 64-bit |
| CPU | Intel Core i7-8086K |
| RAM | 16,0 GB |
| Hovedkort | ASUSTeK COMPUTER INC. ROG MAXIMUS XI FORMULA |
| GPU | Nvidia RTX 2080 |

# 4 RESULTATER

## *4.1 Konseptmodell*

Nedenfor er en konseptmodell over mobilapplikasjonen. Denne følger ingen standard som UML eller lignende standarder, men er for å vise en sammenheng mellom forskjellige funksjoner, algoritmer og databasen. Brune ruter representerer hva brukeren av mobilapplikasjonen kan se på skjermen, mens de grønne rutene er algoritmene som blir tatt i bruk når forskjellige funksjoner blir iverksatt. Innloggingen navigerer til den blå ruten som representerer databasen for enten å lage en ny bruker, editere en bruker eller logge inn på systemet.




*Figur 4: Konseptdiagram for system*

## *4.2 Grensesnitt konsept*

Nedenfor er et grensesnitt design for appen. Denne ble brukt for å ha et rammeverk å arbeide med når appen ble konstruert. Den hadde masse innspill på sluttresultatet.

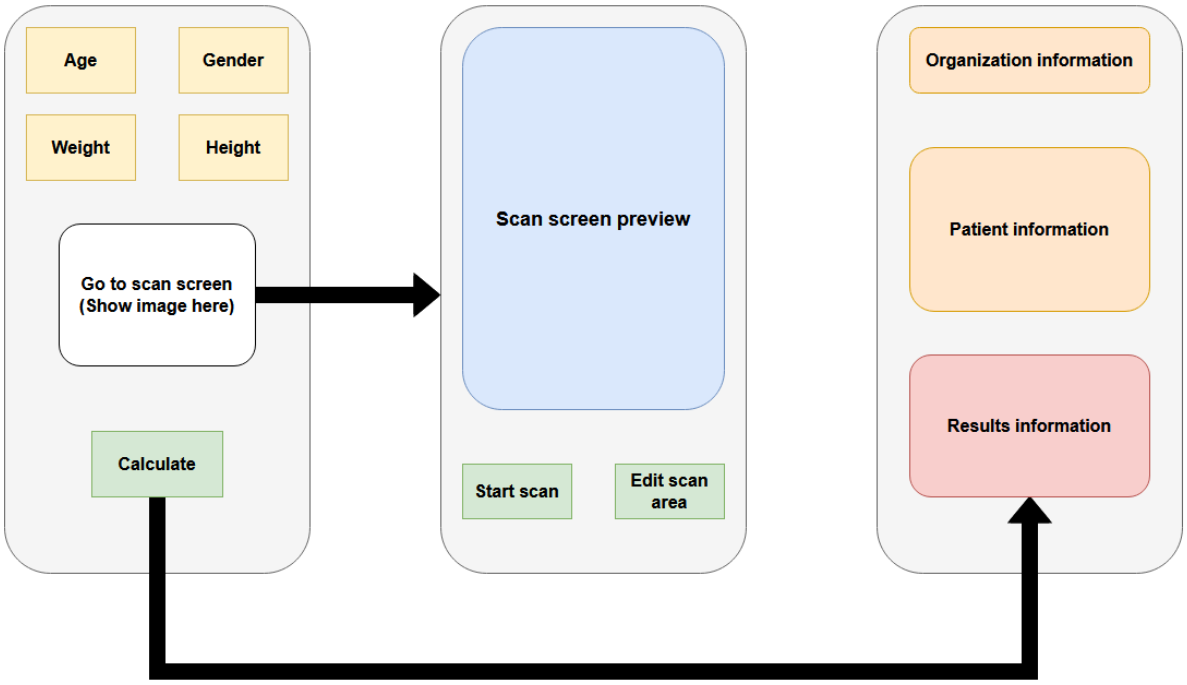
Gule ruter er felt som fylles inn med nummer. Verdiene som blir utfylt her, er data som kommer fra en pasient.

Grønne felt er knapper som utfører forskjellige funksjoner.

Oransje felt er gir informasjon til brukeren etter funksjoner er ferdig kjørt.

Røde felt er informasjon som er oppgitt etter du har trykket på kalkuler.

Det blå feltet representerer kameraet til telefonen som blir navigert til via den hvite knappen.




*Figur 5: Grensesnitt design*

## *4.3 Applikasjonens oppbygning*

### 4.3.1 Applikasjon start

Applikasjonen kjører en klasse med navnet «MyApp» med en innebygd Flutter funksjon som har navnet «runApp». «runApp» funksjonen starter opp den gitte widgeten og viser den på skjermen (Flutter u.d.).

I denne applikasjonen er den gitte widgeten «LoginPage» og er hva som blir opprettet på skjermen.

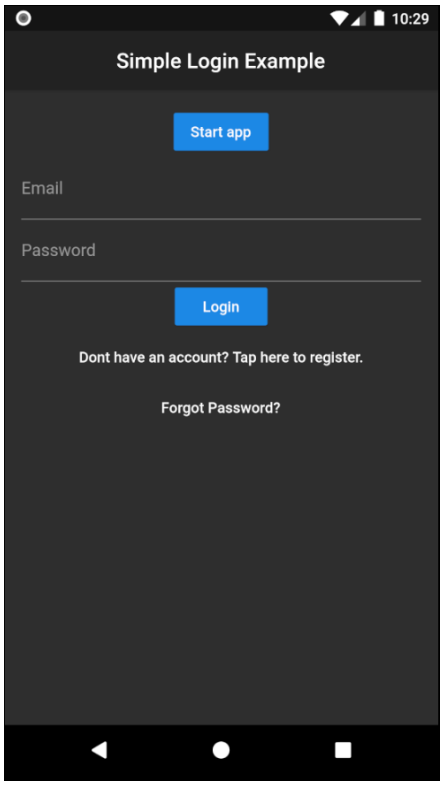
```dart
void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'MainPage',
      debugShowCheckedModeBanner: false,
      theme: ThemeData.dark(),
      home: LoginPage(),
    ); // MaterialApp
  }
}
```

*Figur 6: Funksjon for å starte applikasjonen*

### 4.3.2 Innlogging

Når det navigeres til «LoginPage» er det mulig å logge på med en e-post og et passord eller opprette en ny konto hvis en database er koblet til. Dette er et eksempel for å vise hvordan siden kunne blitt strukturert om prosjektgiver ville hatt med innlogging.




*Figur 7: Eksempel for innloggings skjerm laget av (ahmed-alzahrani 2018)*

Siden dette eksempelet ikke er koblet opp til en database, kan den ikke brukes til å logge inn på eller lage en konto. Som en midlertidig løsning er en «start app» knapp lagt til for å starte applikasjonen i stedet.
Denne innloggings skjermen har mulighet til å koble seg opp til en database eller, ved behov fjernes helt om den ikke kommer i bruk.

### 4.3.3 Innlastingsskjerm

Applikasjonen bruker en funksjon som heter «_LoadingState» for å gi en klar beskjed til brukeren at de navigerer til forskjellige skjermer og at applikasjonen prosesserer informasjon i bakgrunnen. Denne skjermen er brukt i applikasjonen for å kunne navigere til hovedskjermen som er navngitt «HomePage». «_LoadingState» er en «State» (Flutter u.d.) som kan navigere til oppgitt lokasjon ved bruk av «initState» metoden (Flutter u.d.).

Når denne funksjonen kjører så er det en fastlagt tid på ett sekund før den navigerer til neste skjerm. Ventetiden kan endres sånn den venter til applikasjonen har funnet frem informasjonen den trenger og dermed navigere til neste skjerm. Siden applikasjonen kjører raskt uten en database koblet til, kan det virke ut som at der er en feil i systemet når det vises for raskt. Derfor ligger det en fast tid på ett sekund.

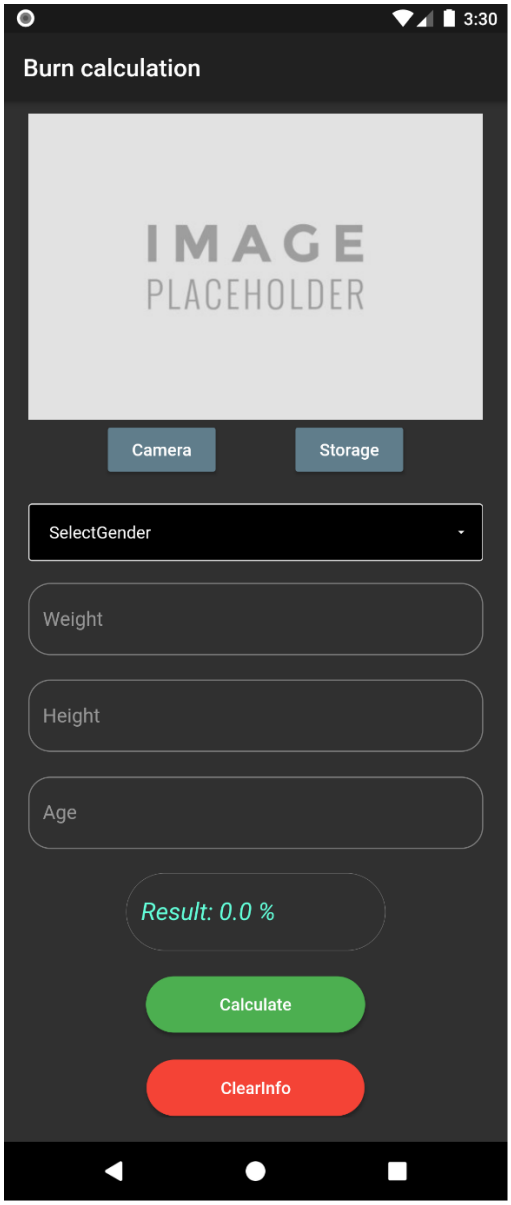
```
class _LoadingState extends State<Loading> {

  @override
  void initState() {
    super.initState();
    new Future.delayed(
        const Duration(seconds: 1),
            () => Navigator.push(
          context,
          MaterialPageRoute(builder: (context) => HomePage()),
        )); // Future.delayed
  }
```

*Figur 8: Funksjon for Innlastingsskjerm når bruker navigerer applikasjonen*

## *4.4 Sluttdesign for grensesnitt*

Sluttdesignet er dannet med tanke på konseptet. I motsetning til konsept designet er det blitt laget en løsning som har alle felt på en skjerm. Dette gir en mer lesbar struktur og gir brukeren klarere navigasjon mellom funksjonene. Alle feltene er fordelt utover med hint på hvert felt som forteller hva som må fylles ut. Kameraet er på en separat skjerm som kan navigeres til med «Camera» knappen. Hvis kamera funksjonen ikke trengs er en «Storage» knapp der for å hente bilde fra galleri eller sky lagring.




*Figur 9: Grensesnitt over applikasjonen*

Applikasjonen er delt opp i fem hoved funksjoner. Disse funksjonene er kamera, kjønn, vekt, høyde og kalkulering. Hvert felt har en egen jobb som må utføres før kalkulering kan gjennomføres.

## *4.5 Kamera*

Kamera funksjonen er delt opp i tre forskjellige funksjoner. Disse funksjonene er å ta et bilde med kameraet til telefonen, velge et bilde fra telefonen sinn lagrings enhet eller sky lagring og til slutt en beskjærings mulighet for å innstille bilder som blir tatt eller hentet fra lagring.

### 4.5.1 Måter å anskaffe bilde

Kamera funksjonen er en programvareutvidelse som heter «image_picker». (cyanglaz 2020) Denne programvareutvidelsen er brukt for å kunne utnytte kamera funksjonen i Flutter. Den gir tilgang til kameraet og lagrings enheten til telefonen.

```
getImageWidget(),
Row(
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  children: <Widget>[
  MaterialButton(
      color: Colors.blue,
      child: Text(
        "Camera",
        style: TextStyle(color: Colors.white),
      ), // Text
      onPressed: () {
        getImage(ImageSource.camera);
      }), // MaterialButton
  MaterialButton(
      color: Colors.blue,
      child: Text(
        "Storage",
        style: TextStyle(color: Colors.white),
      ), // Text
      onPressed: () {
        getImage(ImageSource.gallery);
      }) // MaterialButton
  ], // <Widget>[]
), // Widget to add buttons to get/take picture // Row
```

*Figur 10: Kode for plassering av bilde skjerm og knapper*

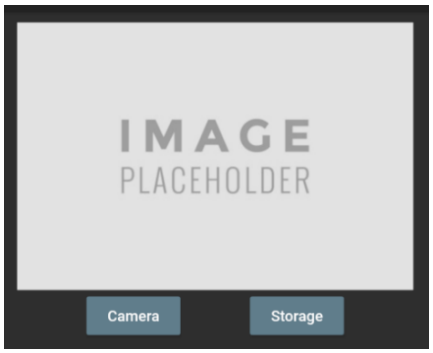
Koden over oppretter en bilde rute og knapper som funksjonene blir knyttet til.

For å gi brukeren informasjon om hvor bildet blir vist kjører en «getImageWidget» funksjon som sjekker om du har tatt eller valgt et bilde. Hvis du ikke har tatt eller valgt et bilde, viser den et midlertidig bilde.

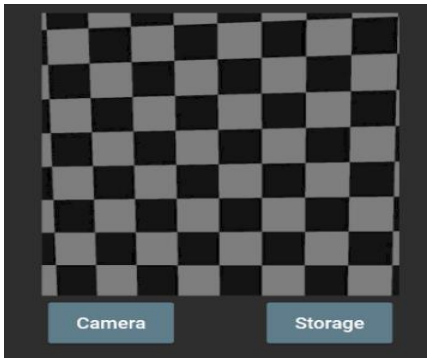
```
Widget getImageWidget() {
  if (_selectedFile != null) {
    return Image.file(
      _selectedFile,
      width: 250,
      height: 250,
      fit: BoxFit.fitHeight,
    );
  } else {
    return Image.asset(
      "assets/placeholder.jpg",
      width: 250,
      height: 250,
      fit: BoxFit.cover,
    );
  }
} // Display image from selection. If not, return placeholder image
```

*Figur 11: Funksjon for å vise bildet som er valgt eller hentet fra lagring.*

Under vises eksempel bilder som brukeren av applikasjonen ser under prosessen fra valget mellom kamera eller lagring til et bilde er valgt. «Placeholder» bildet er hentet fra (Alphasaith u.d.).




*Figur 12: "Placeholder" bilde (Alphasaith u.d.)*




*Figur 13: Eksempel bilde når brukeren har valgt eller tatt ett bilde*

«Camera» og «Storage» knappene kjører en funksjon som heter «getImage». Denne funksjonen henter bildet fra valgt kilde. Den sjekker deretter om der er en kjørende prosess som heter «_waitForProcess» og avslutter med en beskjæring funksjon for å tilpasse bildet for mer nøyaktig bruk.

```
getImage(ImageSource source) async {
  this.setState((){
    _waitForProcess = true;
  });

  File image = await ImagePicker.pickImage(source: source);
  if(image != null){
    File cropped = await ImageCropper.cropImage(
        sourcePath: image.path,

        aspectRatio: CropAspectRatio(
            ratioX: 1, ratioY: 1),
        maxHeight: 700,
        maxWidth: 700,
        compressQuality: 100,
        compressFormat: ImageCompressFormat.jpg,

        androidUiSettings: AndroidUiSettings(
          toolbarColor: Colors.deepOrange,
          toolbarTitle: "Crop picture to desired size",
          statusBarColor: Colors.deepOrange,
          backgroundColor: Colors.grey,
        )
    );
```

*Figur 14: Funksjon for å hente og skalere bildet som er valgt*

### 4.5.2 _waitForProcess

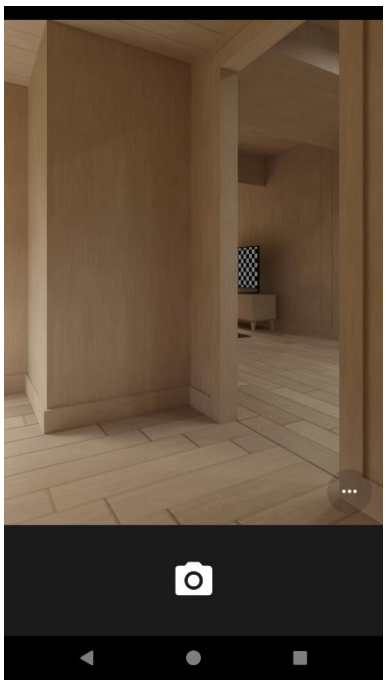
For å unngå at applikasjonen stopper opp, legges det inn en «_waitForProcess» funksjon som legger en innlastingsskjerm mens applikasjonen prosesserer bildet. Hvis denne ikke er på plass kan brukeren ta to bilder på samme tid som vil låse applikasjonen. Dette er en boolean som sjekker om du tar eller henter et bilde.

```
    (_waitForProcess)?Container(
      color: Colors.white,
      height: MediaQuery.of(context).size.height * 0.95,
      child: Center(
        child: CircularProgressIndicator(),
      ), // Center
    ):Center() // Added loading screen to avoid app crash when selecting image // Container
```

*Figur 15: Boolean for innlastingsskjerm når bilder blir tatt eller hentet fra lagring*

«_waitForProcess» funksjonen legger en hvit skjerm med en vente indikator over kamera funksjonen når du har valgt eller tar ett bilde. Denne varer ut prosessen telefonen bruker til å vise bildet til brukeren.

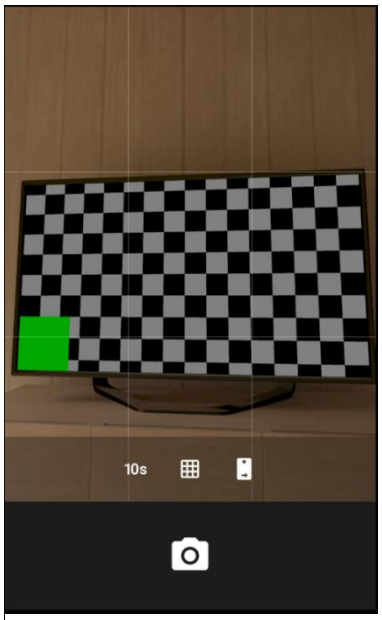
Under vises eksempler på hvordan brukeren vil gjennomgå prosessen av å kunne ta et bilde eller velge et bilde fra forskjellige kilder som er tilgjengelig.



**Kamera eksempel**

Brukeren har en generell kamera funksjon som kan brukes til å ta bilder. Denne kamera funksjonen støtter noen valgmuligheter for situasjoner som kan trenge dem.

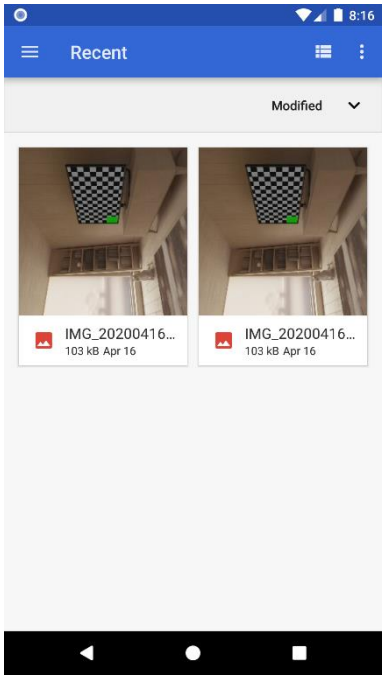
*Figur 16: Kamera eksempel*



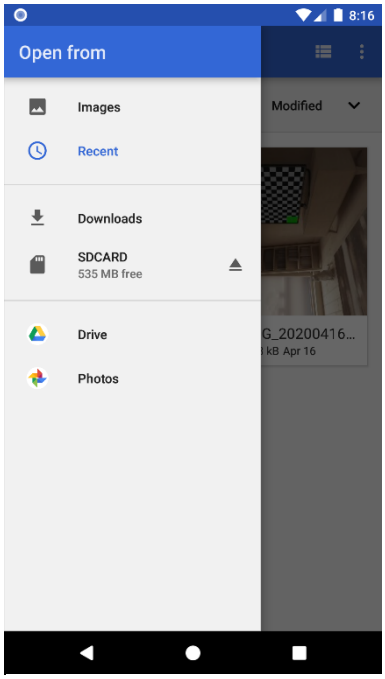

**Valgmuligheter med kameraet**

Kamera funksjonen tilbyr noen få valgmuligheter om det er ønskelig. Brukeren kan skru på en nedteller, opprette et rutenett eller bruke kameraet på fremsiden av telefonen.

*Figur 17: Ekstra valgmuligheter kamera tilbyr*




*Figur 18: Lagring visning eksempel*




*Figur 19: Sky lagring visning eksempel*

Valg av bilder kan gjøres fra lagringsenheten til telefonen eller fra sky lagringer som Google Drive. Der er også en mulighet for å bruke nedlastede bilder.

### 4.5.3 Skalering av bilder

Når applikasjonen ser at du har anskaffet et bilde (Referer til Figur 14: Funksjon for å hente og skalere bildet som er valgt) vil den kjøre en annen programvareutvidelse som heter «ImageCropper». (Yalantis 2020)

«ImageCropper» kan dermed skalere bildet som er valgt og komprimere det til å passe inn i ruten på skjermen. Du har fleksibilitet til å skalere størrelse, rotasjon og plassering av bildet. Når brukeren er ferdig med skaleringen, vil bildet vises basert på størrelse av enheten. Funksjonen komprimerer bildet til størrelsesforholdet enhet har, for å vise den beste bildekvaliteten til brukeren. Bilderuten er skalert til 700 ganger 700 piksler for å unngå visning av bildet utenfor mobilskjermen. Denne kan forandres til en prosent skalering hvis nødvendig.

## *4.6 Kalkulering*

Kalkulerings funksjonen er brukt til å kjøre en algoritme som bruker alle verdier oppgitt og leverer et resultat. Dette viser skadeomfanget av pasienten til brukeren av applikasjonen. Algoritmen trenger dermed informasjon fra kjønn, vekt, høyde, alder og bilde feltene.

### 4.6.1 doAddition

Algoritmen har blitt navngitt «doAddition». Denne skal brukes til å finne skadeomfanget med informasjon fra fire felt. Kjønn, høyde, alder og vekt.

Den femte funksjonen som skanner etter brannskade på bildet, ble ikke implementert i tide.

Derfor er formelen fleksibel og kan forandres til planlagt bruksområde når informasjon er anskaffet.

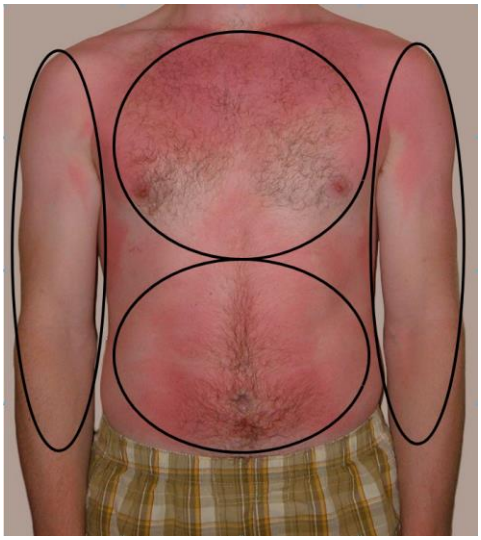
```
doAddition() {
  setState(() {
    genderSelect = double.parse(selectedKey);
    num3 = double.parse(t3.text);
    num2 = double.parse(t2.text);
    num1 = double.parse(t1.text);
    res = num1 + num2 + num3 + genderSelect;
  });
} // The formula for calculation
```

*Figur 20: doAddition funksjonen*

#### 4.6.1.1 Hvordan ville brannskade skannen fungert?

Når bildene har blitt prosessert av mobiltelefonen og brukeren er fornøyd med resultatet, vil en funksjon starte som identifiserer området som er skadet. Den starter med å lage et utklipp av området til kroppsdelen som er påvirket, og avslutter med å informere brukeren av applikasjonen hvor stor skaden er på den lokasjonen.

Eksempelvis ville applikasjonen med bruk av bildet under delt opp personen i fire regioner. To armer, bryst og mage. Den vil dermed undersøke hvilke områder som er utenfor vanlig hudkontrast og gi tilbakemelding over hver kroppsdels totalprosent skade. Den avslutter med en samlet total skade over hele kroppen.



*Figur 21: Førstegrads forbrenning (Wikipedia 2020)*

Prosessen må også kunne se forskjell på de forskjellige stadiene av forbrenning. Under vises en andregrads forbrenning.

*Figur 22: Andregrads forbrenning (Villines 2019)*

### 4.6.2 Kjønn

For å velge kjønn brukes en menyknapp. Dette er en programvareutvidelse med navnet «menu_button» (hugoextrat.com 2020).

Denne funksjonen lar brukeren av applikasjonen velge mellom mann eller dame. Dette skal utgi to forskjellige formler når informasjon om hvilke formler som kommer i bruk blir oppgitt.

```
MenuButton(
  child: button,
  items: keys,
  topDivider: true,
  itemBuilder: (value) => Container(
      height: 40,
      alignment: Alignment.centerLeft,
      padding: const EdgeInsets.symmetric(vertical: 0.0, horizontal: 16),
      child: Text(
        value,
        style: TextStyle(color: Colors.white.withOpacity(1.0)),
      )), // Text, Container
  toggledChild: Container(
    color: Colors.black,
    child: button,
  ), // Container
  divider: Container(
    height: 1,
    color: Colors.white,
  ), // Container
  onItemSelected: (value) {
    setState(() {
      selectedKey = value;
    });
  },
```

*Figur 23: Funksjon for meny knapp*

Meny knappen er separert ut i forskjellige liste som heter «keys». Når bruker trykker på meny knappen viser den tilgjengelige «keys». Når brukeren har valgt fra listen ser applikasjonen hvilket kjønn som er valgt med «initState» funksjonen (Flutter u.d.) (Se Figur 24: InitState funksjon som ser hvilken key er valgt) som forandrer verdien til «selectedKey» stringen. Valget blir lagret som tekst og blir konvertert til brukbare nummer (Se Figur 20: doAddition funksjonen).

```
@override
void initState() {
  selectedKey = keys[0];
  super.initState();
} // Set selected state of the key chosen
```

*Figur 24: InitState funksjon som ser hvilken key er valgt*

Denne meny knappen kan byttes ut med en klasse Flutter støtter selv som heter «DropdownButton» (Flutter u.d.). Grunnen til bruk av «menu_button» programvareutvidelsen er for å få designet mer ryddig og noe av funksjonalitetene er mer komfortable å bruke. Det anbefales å bruke Flutters egen standard, siden programvareutvidelsen for «menu_button» er veldig ny og fremdeles under utvikling.

### 4.6.3 Vekt, høyde og alder

For å la bruker legge inn vekt, høyde og alder er samme funksjonen brukt for alle. Dette er en standard klasse fra Flutter som heter «TextField» (Flutter u.d.). Denne kan ha en «controller» (Se Figur 28: Kontrollere som sjekker om tekst er fylt inn i vekt, høyde og alder feltene) som sjekker om du har skrevet inn tekst i feltet og kan dermed bli gitt et navn som i denne applikasjonen er «t1». Dette navnet blir brukt som en lytter for informasjon.

```
Container(
  width: 200,
  child: TextField(
    keyboardType: TextInputType.number,
    cursorColor: Colors.tealAccent,
    controller: t1,
    decoration: InputDecoration(
      labelText: 'Weight',
      fillColor: Colors.white,
      hintText: 'Enter weight here',
      border: OutlineInputBorder(
        borderRadius: BorderRadius.circular(18.0),
      ), // OutlineInputBorder
    ), // InputDecoration
  ), // TextField
), //Widget to add Weight setting // Container
```

*Figur 25: Funksjon for vekt feltet*

Høyde og alder funksjonene bruker derfor samme oppsett som vekt funksjonen siden dette er konkrete verdier som blir oppgitt når applikasjonen blir brukt. Høyde og alder funksjonene bruker også en «controller» med navn «t2» og «t3».

```
Container(
  width: 200,
  child: TextField(
    keyboardType: TextInputType.number,
    cursorColor: Colors.tealAccent,
    controller: t2,
    decoration: InputDecoration(
      labelText: 'Height',
      fillColor: Colors.white,
      hintText: 'Enter height here',
      border: OutlineInputBorder(
        borderRadius: BorderRadius.circular(18.0),
      ), // OutlineInputBorder
    ), // InputDecoration
  ), // TextField
), //Widget to add Height setting // Container
```

```
Container(
  width: 200,
  child: TextField(
    keyboardType: TextInputType.number,
    cursorColor: Colors.tealAccent,
    controller: t3,
    decoration: InputDecoration(
      labelText: 'Age',
      fillColor: Colors.white,
      hintText: 'Enter Age here',
      border: OutlineInputBorder(
        borderRadius: BorderRadius.circular(18.0),
      ), // OutlineInputBorder
    ), // InputDecoration
  ), // TextField
), //Widget to add Age setting // Container
```

*Figur 27: Funksjon for høyde feltet*

*Figur 26: Funksjon for alder feltet*

### 4.6.4 Text controller

Når brukeren modifiserer tekst inne i en «TextField» som er koblet til en «controller», blir «TextField» oppdatert og varsler lytterne som er «t1», «t2» og «t3».

```
TextEditingController t1 = TextEditingController(text: '');
TextEditingController t2 = TextEditingController(text: '');
TextEditingController t3 = TextEditingController(text: '');
```

*Figur 28: Kontrollere som sjekker om tekst er fylt inn i vekt, høyde og alder feltene*

## *4.7 Clear info*

Hvis bruker av applikasjonen har gjort en eller flere feil kan de kjøre en funksjon som nullstiller alle verdier oppgitt, med unntak av kamera siden dette må føres inn på samme måte uansett hvor langt i prosessen du har kommet.

## *4.8 Resultat skjerm*

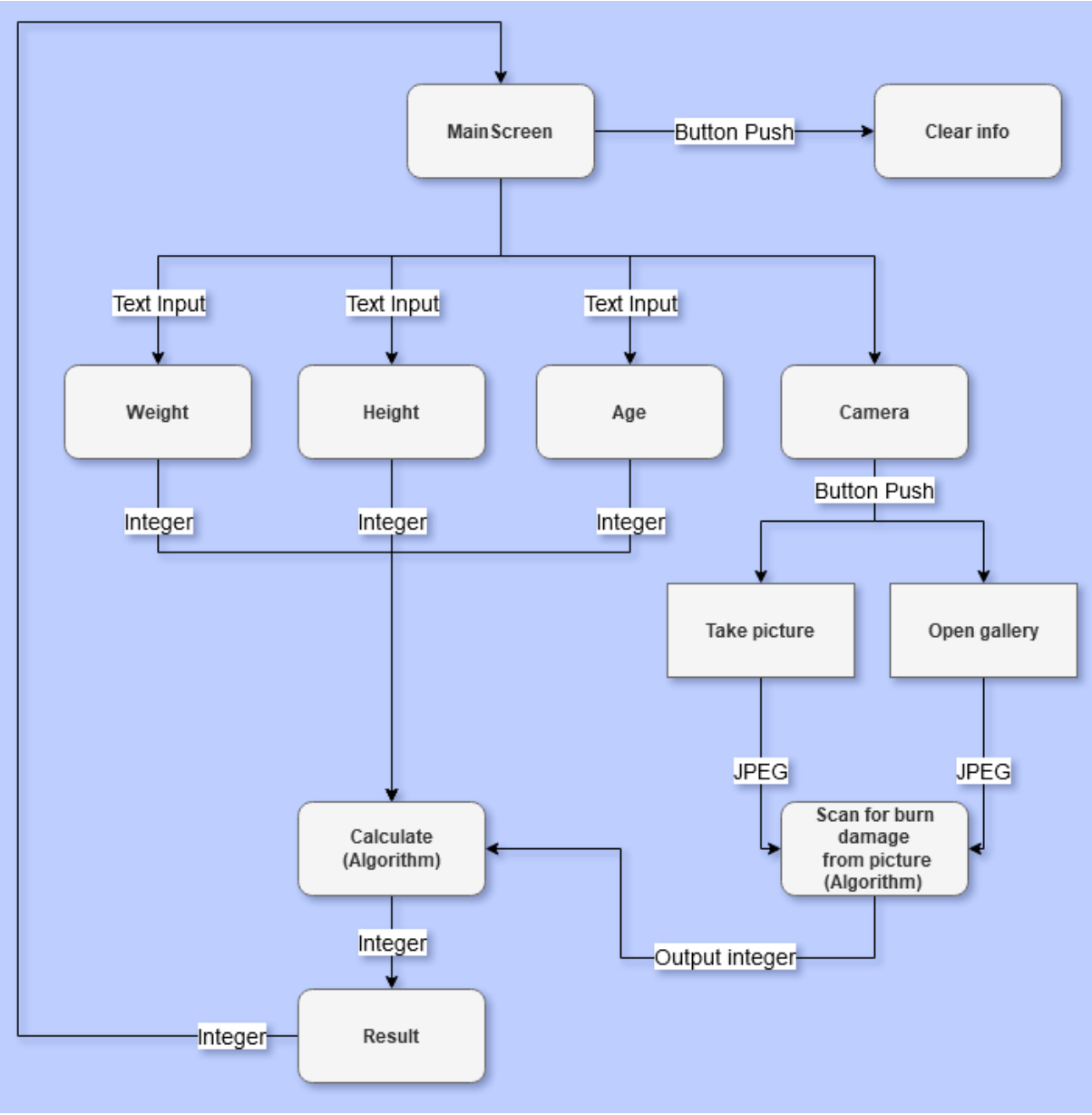
Når bruker har trykket på kalkulerings knappen vil det kjøres en algoritme ved bruk av alle oppgitte verdier. Etter den har kjørt igjennom algoritmen vil den visualisere hvor stor skaden er til brukeren.

Dette er gjort ved visning av resultatet navngitt «res» gjennom et tekstfelt.

```
Padding(
  padding: const EdgeInsets.symmetric(horizontal:80.0),
  child: TextField(
    keyboardType: TextInputType.number,
    cursorColor: Colors.tealAccent,
    enabled: false,
    //enableInteractiveSelection: false,
    decoration: InputDecoration(
      fillColor: Colors.white,
      hintText: 'Result: $res %',
      hintStyle: TextStyle(fontSize: 20.0, color: Colors.tealAccent,
          fontStyle: FontStyle.italic), // TextStyle
      border: OutlineInputBorder(
        borderRadius: BorderRadius.circular(40.0),
      ), // OutlineInputBorder
    ), // InputDecoration
  ), // TextField
), //Widget creating result screen // Padding
```

*Figur 29: Kode for visning av resultat*

## 4.9 Flowchart Diagram




*Figur 30: Flowchart diagram*

Over er et flowchart diagram som viser prosessen brukeren av applikasjonen ville gjennomgått for å utføre hele kalkulasjonen. Dette starter med hovedskjermen hvor vekt, høyde og alder må fylles ut. Deretter blir et bilde tatt og skannet for å identifisere skade, som er siste manglede verdi før kalkuleringen kan starte.

Vekt, høyde og alder tar inn tekst som brukeren skriver inn basert på pasienten det gjelder. Så velges kjønn fra en meny liste. Alle disse verdiene blir dermed konvertert til nummer som applikasjonen kan bruke i kalkulasjons algoritmen.

For kamera funksjonen kan brukeren velge mellom å ta et bilde eller hente et bilde. Disse vil sende ut et JPEG-format, dette går igjennom en algoritme der det blir skannet etter skadeområdet og utgir en tall verdi som kan brukes i kalkulering algoritmen.

Til slutt vil brukeren av applikasjonen trykke på en kalkulering knapp som bruker alle verdiene fra de andre feltene. Dette utgir et resultat i prosent som vises på hovedskjermen for å informere brukeren om skadeomfanget.

## *4.10 Resultat i henhold til krav*

Dette var en åpen oppgave og ble stilt få krav til. Dermed laget jeg egne krav til oppgaven og utførte dem til min beste evne.

### 4.10.1 Effektivitet

Applikasjonen er konstruert med enkle algoritmer og funksjoner med så få mulig mellom skjermer. Dette er for å unngå at prosesser blir gjennomført seint.

### 4.10.2 Plattformer

Siden applikasjonen er laget med bruk av Flutter har applikasjonen innebygd støtte for både Android og IOS systemer, med mulige Windows Phone løsninger.

### 4.10.3 Brukervennlig grensesnitt

Applikasjonen er laget sånn den viser alle felt på en skjerm. Denne strukturen gir en mer ryddig opplevelse for brukere av applikasjonen. Designet var oppstilt med felt som må fylles ut fra topp til bunn. Med kalkulering som siste valg mulighet.

Applikasjonen er fordelt mellom tre skjermer. En for innlogging, en for kamera funksjonene og resten på hoved skjermen. For å gi et mer behagelig utseende lar applikasjonen deg bla skjermen nedover, i stedet for å ha flere mellom skjermer med felt som må utfylles.

### 4.10.4 Oppbygning av appen

Siden applikasjonen bruker Flutter, er konstruksjonen av koden ryddig og forstårlig. Dette gir gode muligheter om det oppstår nødvendighet for utvikling videre.

# 5 DRØFTING

Applikasjonen trenger en spesifikk løsning for å kunne gjenkjenne skader på en person. Denne teknologien er avansert og har ikke mange løsninger tilgjengelig på nettet.

Den nærmeste teknologien som kan utnyttes er ansiktsgjenkjenningssystem. Problemet med å bruke denne teknologien er at den bruker en database som har en biometrisk kunstig intelligens. Å danne et system som bruker dette er avansert og utenfor perspektivet som denne applikasjonen er bygget rundt.

Noen systemer som ble sett på inkluderer Snapchat sitt ansiktsgjenkjenningssystem og Facebook sitt eget system med navnet DeepFace.

Snapchat anskaffet sinn løsning fra et nytt kompani med navnet Loosery. De kjøpte opp kompaniet (Sheremeta 2015) og har brukt deres løsning siden. Løsningen bruker koordinater for å lokalisere ansiktet til en person. Med X og Y koordinatene finner den frem ansikts landemerker og avslutter med å prosessere hvordan ansiktet beveger og former seg (Le 2018).

DeepFace er en løsning som Facebook har laget med bruk av et nettverk av simulerte nevroner som lærer seg å gjenkjenne ansikts former med nokk data. Den har en estimert 97,25 prosent nøyaktighet (CBSnews 2014).

Siden få løsninger med et fornuftig design var tilgjengelig ble oppgaven laget med egne løsninger og lærdommer. Sluttproduktet ble laget med tanke på å bruke et farge gjenkjennings system for å kunne identifisere skader, men ble ikke fullført i tide for prosjektet.

## *5.1 Problemstillinger*

Når prosjektgiver beskrev hva han ville ha som et produkt oppsto det noen problemer.

- Det første problemet er når en skal kunne utføre skanningen av en pasient; det eksisterer ingen mulighet for å få et fullstendig bilde av personen. Dette er problematisk hvis brannskaden er på baksiden eller rundt kroppen.
- Et annet problem var hudfarger i forhold til brannskade farger. Brannskader har store fordelinger av farger som kan gå fra rød, hvit, rosa, svart, brun eller lignende verdier.

Dette blir vanskelig å arbeide seg rundt uten AI-systemer som kan gjenkjenne forskjeller mellom uskadet og skadet hud.

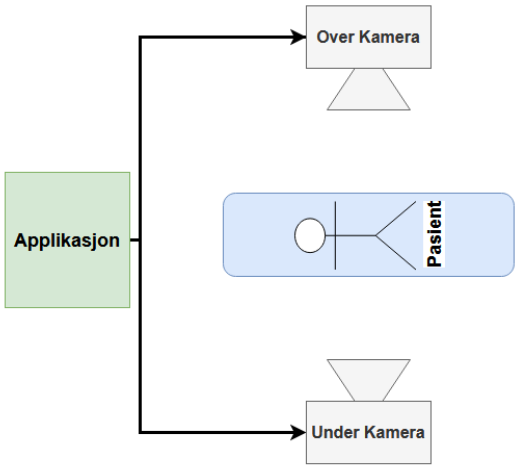
- Det største problemet som kommer med denne applikasjonen er at den per definisjon ikke kan bli godkjent til bruk i brannskade situasjoner. Dette kommer av at pasienter som kommer inn på akutten blir kled av så masse som mulig, normalt naken. Dette er for å identifisere hvor skaden er lokalisert og gi lettes tilgang for å fikse problemet. Reglene kommer fra Åndsverksloven: § 104. *Retten til eget bilde* (Lovdata 2018) *og NOU 2019: 10* (Regjeringen.no 2019).

Med de forskjellige problemstillingene ble mange løsninger snakket om og visualisert. Dette var for å gi prosjektgiver forståelse for hvilke modifikasjoner eller utviklinger som kan bli nødvendig for et lovlig og bedre sluttprodukt.

## *5.2 Løsninger for problem*

### 5.2.1 Fullstendig skann løsning

Den letteste konklusjonen var å lage et system som har telefoner eller kameraer i tak og gulv, med en gjennomsiktig båre som pasienten ligger på. Dette gir muligheten for to bilder som kan samles for en komplett skanning. Med situasjonen som den er nå så er dette ikke en realistisk løsning, siden modifikasjoner på akutter må utføres og det gir ikke den mobile løsningen prosjekt giver vil ha.




*Figur 31: Eksempel på en skanneløsning*

### 5.2.2 3D-skann

En løsning som ble nevnt var bruk av 3D-skanning verktøy. Dette blir utført med spesiallaget maskineri som kan maske ut formen av en person. Med god utvikling kan muligens verktøyet finne de irregulære områdene som er skadet og kalkulere hvor stor skaden kan være.

Der eksisterer løsninger for 3D-skanning med mobiltelefoner (Obudho 2020). Uten bred forståelse innen området for å utnytte eller reprogrammere hva skanneren leter etter er ikke dette en enkel eller billig løsning.

### 5.2.3 Farge identifikasjon

Den brukte løsningen for prosjektet er ved bruk av farge identifikasjon. Dette virker ut til å være den letteste funksjonen å implementere i et system der det trengs å identifisere skader med klare kjenne tegn. For å teste frem løsningen ble forskjellige datapakker til Flutter sett på. En som muligens kan brukes er «flutter_colorpicker» (Mchome 2020), men den vil trenge modifikasjoner og videre utvikling for bruk i lignende prosjekt.

De fleste datapakkene tilgjengelig brukes vanligvis for å gi fargealternativer til utviklerne av applikasjoner. Siden skade identifisering problemet ikke er et stort tema finnes få løsninger som kan utnyttes.

### 5.2.4 Ekstern maskinvare

Den beste løsningen for å utføre hva prosjektgiver vil ha, er med å lage egen maskinvare som er designet rundt å utføre denne prosessen. Det kunne blitt innebygd i operasjon bordet der den måler for eksempel vekt og høyde direkte. Den kan kjøre en fullstendig 3D-skann som utgir alt av informasjon en doktor kan trenge for å utføre nødvendige operasjoner.

Denne løsninger er den dyreste og mest avanserte. Der trengs spesialister innen flere fag områder for å lage en slik løsning, men det er den mest realistiske og praktiske løsningen for problemet.

## *5.3 Sikkerhet*

Når en utvikler en mobilapplikasjon, oppstår der flere problem innen sikkerhet. Normalt sett er der få løsninger som kan brukes for å sikre applikasjonen selv og brukeren av applikasjonen. Hvis det skal bli designet en løsning for sikkerhet kan applikasjonen begrenses til bedrift bruk og bli levert på bedrift installerte telefoner.

En viktig løsning hvis denne applikasjonen skal komme til bruk er å kunne stoppe skjermbilder og lagring av bilder lokalt på telefonen. Noen eksempler på hvordan dette kan utføres er Snapchat sitt informering system som sender en melding når det blir tatt skjermbilde av et bilde sendt via Snapchat. Det kan også lages løsninger som Netflix bruker der skjermbildet vil vise som svart hvis et bilde blir tatt av en serie eller film.

Mulighet for å bruke et sensurering filter er også en mulig løsning, men dette kan påvirke kalkulasjon data i visse tilfeller.

## *5.4 Resultatet*

### 5.4.1 Bruk av Flutter

Flutter har vært et utmerket utviklingsverktøy med lite komplikasjoner. Lære prosessen har vært fleksibel og godt forstårlig med Flutters egne kilder. Youtube kanaler som RetroPortal Studio (Studio 2018) og The Net Ninja (Ninja 2015) har også vært til stor hjelp for å komme i gang eller finne eksempler på struktur og funksjonaliteter i kode.

Siden dette er et nytt verktøy som brukes innen mobilapplikasjons utvikling har det imponert meg. Den største kritikken som kan bli sakt er at det er enda et ungt verktøy med få ressurser tilgjengelig for nå, men ser ut til å bli bedre for hver dag.

### 5.4.2 Evaluering av applikasjonen

Kriteriene gitt av prosjektgiver ble brukt som veiledning igjennom prosjektet. Alle krav oppgitt ble utfylt, med unntak av bilde skanning funksjonaliteten. Utvikling av dette prosjektet har vært en avansert prosess, som har gitt et godt rammeverk med evnen til å bli en komplett versjon av hva prosjektgiver forventet. Applikasjonen fikk mangel på innspill fra prosjektgiver i siste perioden.

Med formler, eksempler og tilbakemeldinger fra prosjektgiver i slutten av arbeidsperioden ville applikasjonen hatt bedre utviklings muligheter, men prosjektgiver ble utilgjengelig og dette er sluttresultatet.

### 5.4.3 Evaluering av brukervennlighet

Når applikasjonen ble planlagt i starten var det lite spørsmål om hvordan utseende burde være konstruert. Tidlige versjoner hadde flere skjermer som brukeren kunne gå imellom for å utfylle alle nødvendige data. Kamera funksjonen tok bare et enkelt bilde som ble lagret og

ikke informerte hvor det ble brukt og så ut. Der var ingen klare innlastingsskjermer mellom hver prosess som måtte fylles ut. Dette laget en dårlig flyt når applikasjonen ble brukt.

Sluttdesignet endte dermed opp med en løsning som viser alle element på en skjerm, der det kan navigeres opp og ned med berørings kontroller. Dette gir en klar beskjed om hva som må fylles ut og brukte færre mellomskjermer som sakket ned bruk av applikasjonen.

Med dårlig kontakt med prosjektgiver og klassekamerater på grunn av virus situasjonen, var det vanskelig å anskaffe tilbakemeldinger fra brukere som kunne testet ut applikasjonen. Hvis der var større muligheter for testing ville applikasjonen muligens forandret seg drastisk.

### 5.4.4 Avvik

Planen var å få implementert en skannefunksjon for å lokalisere skader på bilder. Denne ble ikke ferdig i tide både på grunn av tidsproblemer og vansker med å lage en funksjonell løsning.

### 5.4.5 Alene arbeid

Dette prosjektet ble utført alene, med stor hjelp fra veileder. Dette har gjort det mulig å utvikle prosjektet så langt det har kommet. Arbeidet var delt utover to arbeids uker, med møter i slutten av hver periode. Nesten alt arbeid har blitt gjort hjemmefra siden skolens lokaler var låst ned i epidemi perioden. På rundt en fire ukers periode ble prosjektet arbeidet på utenfor heimsted siden jeg ble satt i karantene i den perioden. Dette gikk utover arbeids prosessen litt, men ble utnyttet så bra som var mulig i forhold til tidene.

Siden dette har vært eget arbeid har det ikke vært noen som kan gi tilbakemelding eller ideer direkte når arbeid ble gjort. Dette har gjort opplevelsen litt vanskelig når sjøl evaluering har vært nødvendig. Hjelp i prosjektet har kommet fra veileder og erfarne venner om nødvendigheter oppsto, som har kommet til stor hjelp.

Arbeids rytmen har holdt seg bra igjennom hele prosjektet. Dette kommer av interessen for å lære seg et nytt programmeringsspråk og testing av nye program. I situasjoner der løsninger kom til en stopp var der bra nokk ressurser tilgjengelig for å finne andre perspektiv eller løsninger. Dette ga muligheten til å komme seg tilbake i arbeid så raskt som mulig.

# 6 KONKLUSJON

Målet med dette prosjektet var å lage et produkt som kan utføre en fullstendig undersøkelse av en brannskade som har oppstått på en person. Gjennom prosjektet har jeg laget et produkt tilnærmet dette målet med spesifikasjoner gitt fra oppgavegiver. Kompromisser ble nødvendig underveis.

Applikasjonen har blitt oppbygget ved hjelp av Android studio og Flutter utviklingsverktøyet. Oppbygningen har brukt universell utforming prinsipper, med en arbeidsmetode som lignet spiral metoden. Dette ga et resultat som var strukturert, brukbart og leselig for de applikasjonen skal gjelde.
Funksjonaliteter for vekt, høyde, alder, kjønn, bilde, galleri og kalkulering har blitt implementert for å best utføre kravspesifikasjonene som ble oppgitt.

Bruken av en simplifisert spiral metode har fungert fint under utviklingen, men uten en gruppe som har oppgaver, roller eller oppdelt arbeid, har det gitt små vansker underveis. Med en fullstendig gruppe ville prosjektet blitt utviklet mer effektivt, med bedre eller mer avanserte løsninger.

Kontaktmulighetene med prosjektgiver var minimale eller ikke eksisterende i slutten av prosjektet. Dette ga lite muligheter for tilbakemelding angåendes om produktet representerte hva som var forventet. Tilbakemelding ble derfor utgitt ved bruk av veileder, familie eller venner som hadde mulighet til å sjå på eller teste applikasjonen.

Noen funksjonaliteter i prosjektet kunne blitt implementert bedre, og noen ble ikke implementert. Med mer erfaring og flere arbeidspartnere hadde dette kunne blitt gjennomført, men med tidsrammene gitt var det ikke gjennomførbart. Sluttproduktet har dermed de fleste funksjonalitetene som trengs og har muligheter for videre utvikling i fremtiden for et komplett prosjekt.

# 7 REFERANSER

Aaberge, Albertine. 2017. *Ndla.* 15 mai. https://ndla.no/subjects/subject:1/topic:1:186460/topic:1:169379/resource:1:160302.

ahmed-alzahrani. 2018. *github.* 16 september. https://github.com/ahmed-alzahrani/Flutter_Simple_Login.

Alphasaith. u.d. *Wikia.* Funnet Mars 15, 2020. https://vignette.wikia.nocookie.net/undertale-rho/images/5/5f/Placeholder.jpg/revision/latest?cb=20180213155916.

Android. u.d. *Android Studio.* Funnet mai 1, 2020. https://developer.android.com/studio/run/managing-avds.

—. u.d. *Developer.android.com.* Funnet Mai 6, 2020. https://developer.android.com/studio.

Bactra. 2012. *Three-toed Sloth.* 24 September. http://bactra.org/weblog/950.html.

CBSnews. 2014. *CBSnews.com.* 19 Mars. https://www.cbsnews.com/news/facebooks-deepface-shows-serious-facial-recognition-skills/.

Christensson, Per. 2009. *techterms.com.* 27 Oktober. https://techterms.com/definition/database.

—. 2011. *Techterms.com.* 12 Mars. https://techterms.com/definition/boolean.

cyanglaz. 2020. *Pub.dev.* 10 apirl. https://pub.dev/packages/image_picker.

Dart. 2013. *Dart.* 14 november. https://dart.dev/.

Dev, Apple. 2019. Oktober. https://developer.apple.com/app-store/freemium-business-model/.

Digitaliseringsdirektoratet. 2015. *Brukertesting.* 3 Februar. https://www.difi.no/fagomrader-og-tjenester/tidstyver/tidstyvdatabasen-verktoy-og-metoder/brukertesting.

Doshi, Kalpesh. 2019. *Understanding Software Development Process.* 5 August. https://www.browserstack.com/guide/learn-software-development-process.

Dvergsdal, Henrik. 2019. *snl.no.* 10 April. https://snl.no/hacker.

Eirik Rossen, Astri M. Lund. 2019. *snl.no.* 8 November. https://snl.no/program_-_IT.

Encylopedia Britannica. u.d. *SQL.* Funnet Mai 17, 2020. https://www.britannica.com/technology/SQL.

Fernandez, Daniel. 2019. *How to encrypt user passwords.* 26 Mai. http://www.jasypt.org/howtoencryptuserpasswords.html.

Flutter. u.d. *api.Flutter.dev.* Funnet Mai 14, 2020. https://api.flutter.dev/flutter/widgets/runApp.html.

—. u.d. *DropdownButton.* Funnet Mai 14, 2020. https://api.flutter.dev/flutter/material/DropdownButton-class.html.

—. u.d. *InitState.* Funnet Mai 14, 2020. https://api.flutter.dev/flutter/widgets/State/initState.html.

—. u.d. *showcase.* Funnet mai 1, 2020. https://flutter.dev/showcase.

—. u.d. *State.* Funnet Mai 14, 2020. https://api.flutter.dev/flutter/widgets/State-class.html.

—. u.d. *technical-overview.* Funnet Mai 14, 2020. https://flutter.dev/docs/resources/technical-overview.

—. u.d. *TextField.* Funnet Mai 14, 2020. https://api.flutter.dev/flutter/material/TextField-class.html.

FraudWatch International. 2019. *FraudWatch International Mobile App Security.* 5 februar. https://fraudwatchinternational.com/mobile-applications/what-is-mobile-app-security/.

Gramstad, Thomas. 2020. *snl.no.* 28 April. https://snl.no/%C3%A5pen_kildekode.

Gupta, Deepak K [Daksh]. 2018. *ProAndroidDev.* 2 mars. https://proandroiddev.com/flutter-from-zero-to-comfortable-6b1d6b2d20e.

Hanne Gram Simonsen, Arild H. Henriksen. 2019. *snl.no.* 24 April. https://snl.no/semantikk.

hugoextrat.com. 2020. *pub.dev.* 25 Mars. https://pub.dev/packages/menu_button.

Ionos. 2019. *Spiral model.* 5 April. https://www.ionos.com/startupguide/productivity/spiral-model/.

J.Clement. 2020. *Statista.* 22 april. https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/.

Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas. u.d. *Manifesto for Agile Software Development* . Funnet april 30, 2020. http://agilemanifesto.org/.

Le, James. 2018. *medium.com.* 29 Januar. https://medium.com/cracking-the-data-science-interview/snapchats-filters-how-computer-vision-recognizes-your-face-9907d6904b91.

Leler, Wm. 2017. *Why flutter uses dart.* 23 december. https://hackernoon.com/why-flutter-uses-dart-dd635a054ebf.

Liseter, Ivar M. 2014. *sikkerhetskopi.* 30 Juli. https://snl.no/sikkerhetskopi.

Lovdata. 2018. *Lov om opphavsrett til åndsverk mv. (åndsverkloven).* 20 Desember. https://lovdata.no/lov/2018-06-15-40/§104.

Margaret Rouse, David Higgins. 2015. *Techtarget.com.* Desember. Funnet Mai 15, 2020. https://whatis.techtarget.com/definition/widget.

Mchome. 2020. *pub.dev.* 18 april. https://pub.dev/packages/flutter_colorpicker.

Microsoft. u.d. *Office.com.* Funnet Mai 14, 2020. https://www.office.com/.

Nettvett. 2020. *Sosial Manipulering.* 13 februar. https://nettvett.no/sosial-manipulering/.

Nielsen, Jakob. 2000. *NNGroup.* 18 mars. https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/.

Ninja, The Net. 2015. *The Net Ninja.* 8 April. https://www.youtube.com/channel/UCW5YeuERMmlnqo4oq8vwUpg/about.

Norskdesign. 2010. *Hva er Design for alle?* 27 Mars. https://web.archive.org/web/20100327160956/http://www.norskdesign.no/hva-er-design-for-alle/hva-er-design-for-alle-article9930-583.html.

Norskdesign.no. 2007. *Prinsipper for design for alle.* 19 November. https://web.archive.org/web/20111028151057/http://www.norskdesign.no/hva-er-design-for-alle/prinsipper-for-design-for-alle-article2762-583.html.

Obudho, Brian. 2020. *all3dp.* 27 Mars. https://all3dp.com/2/5-best-3d-scanner-apps-for-your-smartphone/.

papirleksikonet. 2018. *snl.no.* 20 Februar. https://snl.no/data.

—. 2009. *snl.no.* 15 Februar. https://snl.no/syntaks_-_IT.

Peter kemp, Paul smith. 2010. *Waterfall model.* 14 Juni. https://en.wikipedia.org/wiki/Software_development_process#/media/File:Waterfall_model.svg.

Powell-Morse, Andrew. 2016. *Waterfall model.* 8 Desember. https://airbrake.io/blog/sdlc/waterfall-model.

Regjeringen.no. 2019. *NOU 2019: 10.* 30 April. https://www.regjeringen.no/no/dokumenter/nou-2019-10/id2643015/.

Rossen, Eirik. 2019. *programmeringsspråk.* 6 November. https://snl.no/programmeringsspr%C3%A5k.

—. 2019. *snl.no.* 7 November. https://snl.no/API.

—. 2019. *snl.no.* 11 November. https://snl.no/brukergrensesnitt.

Rouse, Margaret. 2005. *Hashing.* september. https://searchsqlserver.techtarget.com/definition/hashing.

Royce, Winston W. 1970. «Managin the development of large software systems.» *scf.usc.edu.* http://www-scf.usc.edu/~csci201/lectures/Lecture11/royce1970.pdf.

Royce, Winston W. 2002. *cartoon.* http://cartoon.iguw.tuwien.ac.at/fit/fit01/wasserfall/entstehung.html.

Sheremeta, Bozhena. 2015. *Kyivpost.* 16 September. https://www.kyivpost.com/article/content/ukraines-it-edge/snapchat-buys-looksery-in-reported-150-million-deal-making-it-biggest-acquisition-of-ukraine-tech-player-398002.html.

Studio, RetroPortal. 2018. *RetroPortal Studio.* 23 Desember. https://www.youtube.com/channel/UCW2ATgwtNrsBrE-piE2TIrA/about.

Svein Johan Knapskog, Øyvind Eilertsen. 2019. *Snl.no.* 30 November. https://snl.no/kryptografi.

Technopedia. 2018. *Mobile Application (Mobile App).* 3 Mai. https://www.techopedia.com/definition/2953/mobile-application-mobile-app.

Tidemann, Axel. 2020. *snl.no.* 8 januar. https://snl.no/kunstig_intelligens.

Villines, Zawn. 2019. *MedicalNewsToday.* 16 Mai. https://www.medicalnewstoday.com/articles/325189.

W3Schools. u.d. *SQL Injection.* Funnet mai 1, 2020. https://www.w3schools.com/sql/sql_injection.asp.

Wikipedia. 2020. *Android Studio.* 27 april. https://en.wikipedia.org/wiki/Android_Studio.

—. 2020. *Burn.* 26 April. https://en.wikipedia.org/wiki/Burn.

—. 2020. *Dart (programming language).* 16 Mai. https://en.wikipedia.org/wiki/Dart_(programming_language).

—. 2020. *Flutter (software).* 17 Mai. https://en.wikipedia.org/wiki/Flutter_(software).

Yalantis. 2020. *pub.dev.* 19 januar. https://pub.dev/packages/image_cropper.

# FIGUR LISTE

# TABELL LISTE

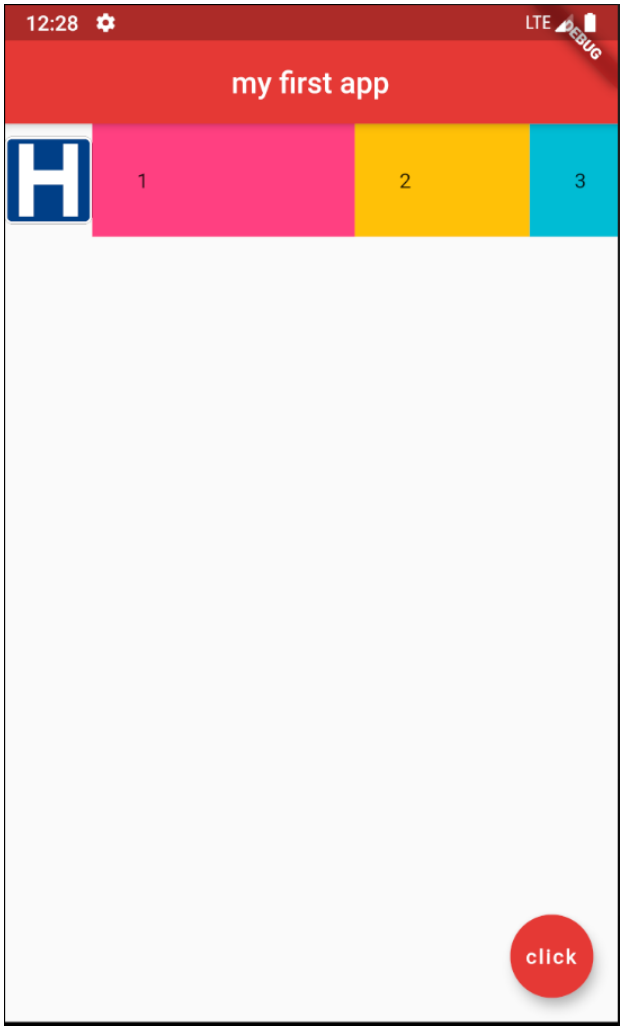
# VEDLEGG

Vedlegg 1 - Prototyper

Vedlegg 2 - Møtelogge

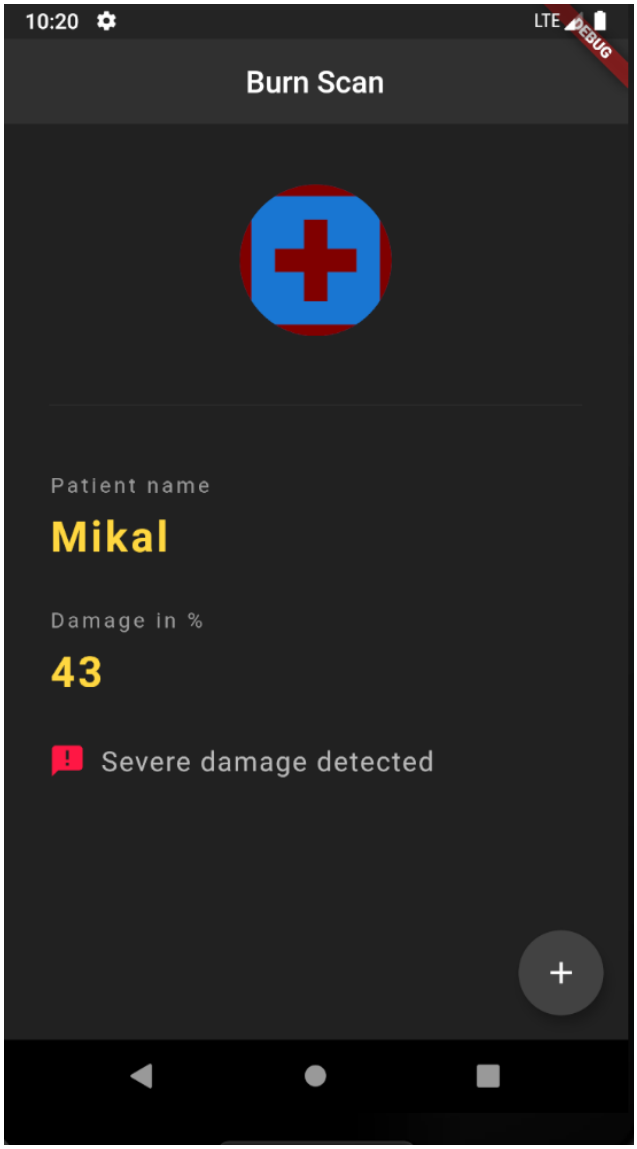
Vedlegg – 1
Prototyper

Prototype 1:
Design fleksibilitet test.



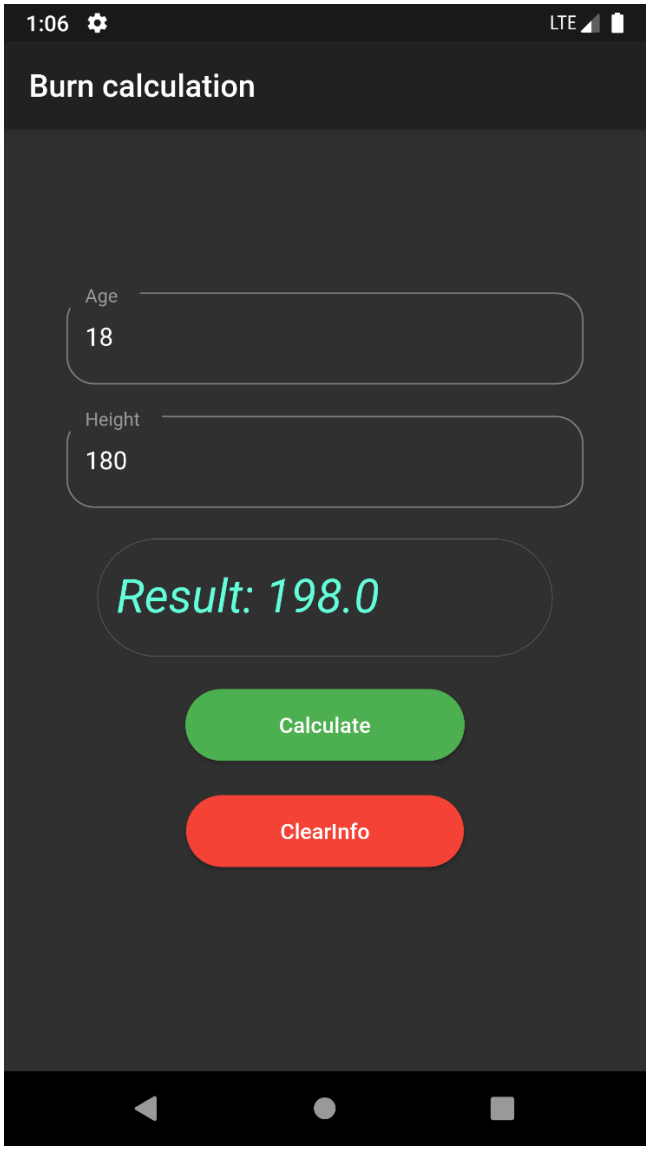

Første versjon av applikasjon brukt for å teste knappe funksjonaliteter, tekst felt og plassering av bilder eller knapper. Ingen funksjoner implementert i første versjon.

Prototype 2:
Resultat skjerm.



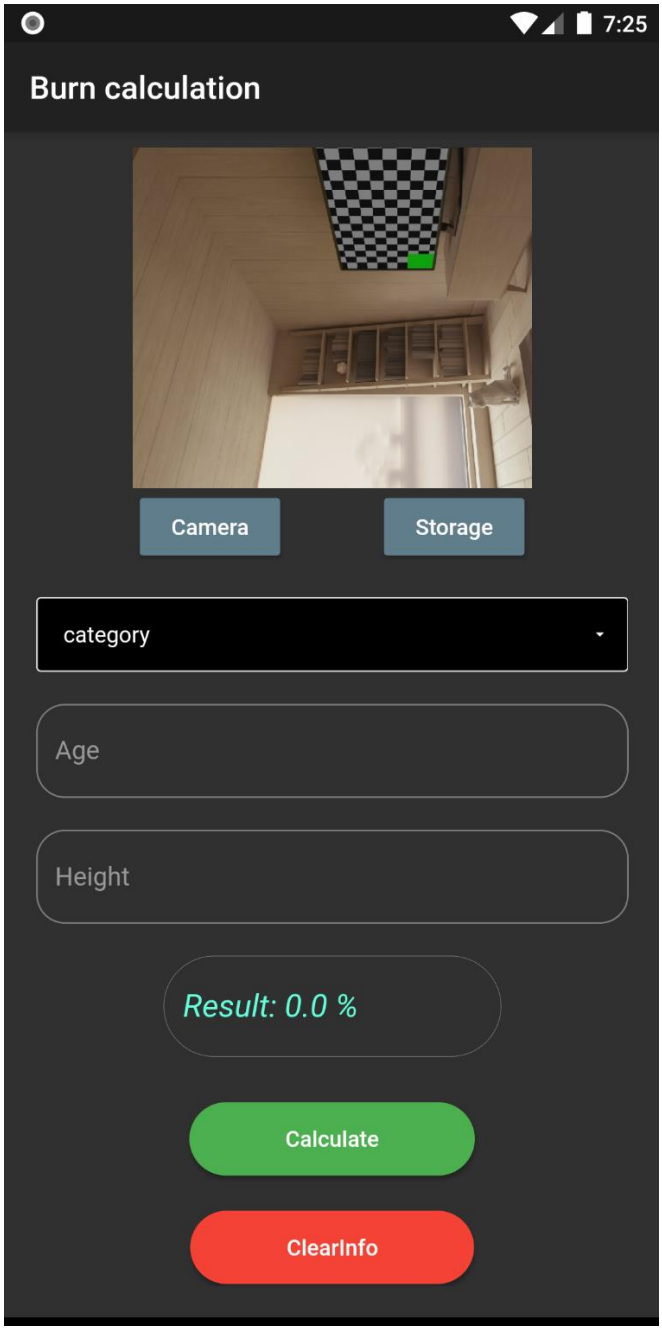

Første iterasjon av resultat skjerm. Viser informasjon på en skjerm separat fra hovedskjermen. Har informasjon om skade, grad av skade og generell informasjon.

Prototype 3:
Kalkulerings funksjon.




Første versjon av kalkulerings funksjon. Bruker en enkel formell for å legge sammen alder og høyde. Kan kalkulere med de to oppgitte feltene og nullstille applikasjonen.

Prototype 4:
Første funksjonelle versjon.




Første versjon av applikasjon som kan utføre alle generelle funksjoner.

Vedlegg – 2
Møtelogger

**Log 1 - 07.02.2020**

I det første møte med Di Wu gjekk vi gjennom planer for hvordan vi skal ha møte og hvordan eg skal framgå med prosjektet.
Til å starte med skal eg sette opp en forplan til prosjektet som skal forklare hvordan ting skal sjå ut og hvilke funksjoner blir nødvendig.
Trenger også å sette opp ei skisse som viser hvordan eg trur appen vil sjå ut.
Vi snakket om å forandre oppgaven i forhold til oppgavegivers forventninger som var et bilde og kalkulere ut fra bildet eller ta en skann med kameraet som beregner direkte en person som velger ut kroppsdelene som er brannskadet og dermed beregne basert på kjønn, vekt, høgde og fasong hvor mye prosent av kroppen er brent.
Den nye planen for nå er å sette opp bilde systemet. Du tar et bilde av pasienten, så vil det bildet bli opplastet til en database sentral som skal kunne sende resultatet tilbake til telefonen.

**Log 2 – 21.02.2020**

Det andre møte vi hadde gjekk vi igjennom oppsettet eg hadde og hvor langt inni prosessen eg var. Vi diskuterte noen ideer angåendes fremgangen med prosjektet og om hvilken forandring som muligens ville komme til bruk. Eg forklarte litt om hvordan flutter fungerte og hvordan eg kan utnytte noen innebygde funksjoner i flutter til å hjelpe med funksjonaliteten til programmet i fremtiden.
Vi repeterte noen punkt fra forrige møte om hvilke funksjoner vil være viktige når de skal bli implementert. Vi kom fram til at Kjønn, Vekt, Høgde, alder og fasong på personen blir inputen som skal brukes for å få kalkulasjonen av brannskade prosent tilbake.
For de neste 2 vekene blir planen å starte med en UI til programmet. Vi tenkte også da et simpelt og raskt design siden appen må kunne være raskere og presis nokk til å utarbeide en person som hadde gjort brannskade sjekken manuelt. Vi kom fram til et enkelt knapp system der du får store ruter opp som gir estimat som blir brukt i beregninga. Så du vil først få opp mann eller dame, deretter høgde som muligens er valg fra et estimat som vil sjå ut som 150-160 cm, 160-170 cm og lignende opp til antageligvis 2 meter høgde. Også med mulighet for manuell input hvis nødvendig. Så blir det samme med vekt oppgitt på samme måte som høgde. Til slutt så vil det komme inn mulighet for å velge fasong. Vi starter muligens med valg som, slim, average, fat, obese.

Eg skal også kontakte prosjekt giver for å muligens få eksempel bilder av brannskader vist mulig, eller eksempel sånn eg har test muligheter. Også må eg spør om formelen brukt for å kalkulere skaden og en feilmargin for programmet. Kanskje noe som 5-10% feilmargin. Også hvor lang tid en person bruker til vanlig på å gjøre dette manuelt og hvor nøyaktig det bruker å være.
Sist eg snakket med prosjekt giver angåendes appen så hadde han noen problemer med det å bruke en telefon til å ta bilde av en pasient. Vi kom fram til en mulig løsning for det problemet og det problemet med at vi ikke har mulighet alltid til å få et bilde av mer en fronten av kroppen. Løsningen blir å ha et kamera i taket av ambulansen der pasienten vil være liggende under og muligens et i gulvet som kan ta bilde under ifra hvis de har en båre som er gjennomsiktig design. Hvis vi bruker dette som en løsning vil vi ha et oppsett som alltid tar bilde fra samme vinkel og en mulighet for en fullstendig skanning av pasienten. Dette vil eg snakke med prosjekt giver om og finne ut hva han tenker om.

Med ny informasjon sies det at vekt og høgde kan være manuelt skreve inn. Vekt og høgde blir målt på akuttmottaket.

**Log 3 – 06.03.2020**

Formidlet informasjon gitt fra oppgavegiver når vi hadde forrige samtale. Ellers ble det bere bygget videre på prosjektets funksjoner
Spurte prosjektgiver om eksempel bilder på skade, formelen som må brukes for å gjøre kalkulasjonen. Ellers ble det informert om hvordan applikasjonen hadde kommet seg.

**Log 4 – 20.03.2020**

Applikasjon har nesten alle funksjoner inne. Siste funksjoner blir implementert på denne iterasjonen.
Ingen tilbakemelding om eksempel bilder eller formel.

**Log 5 – 03.04.2020**

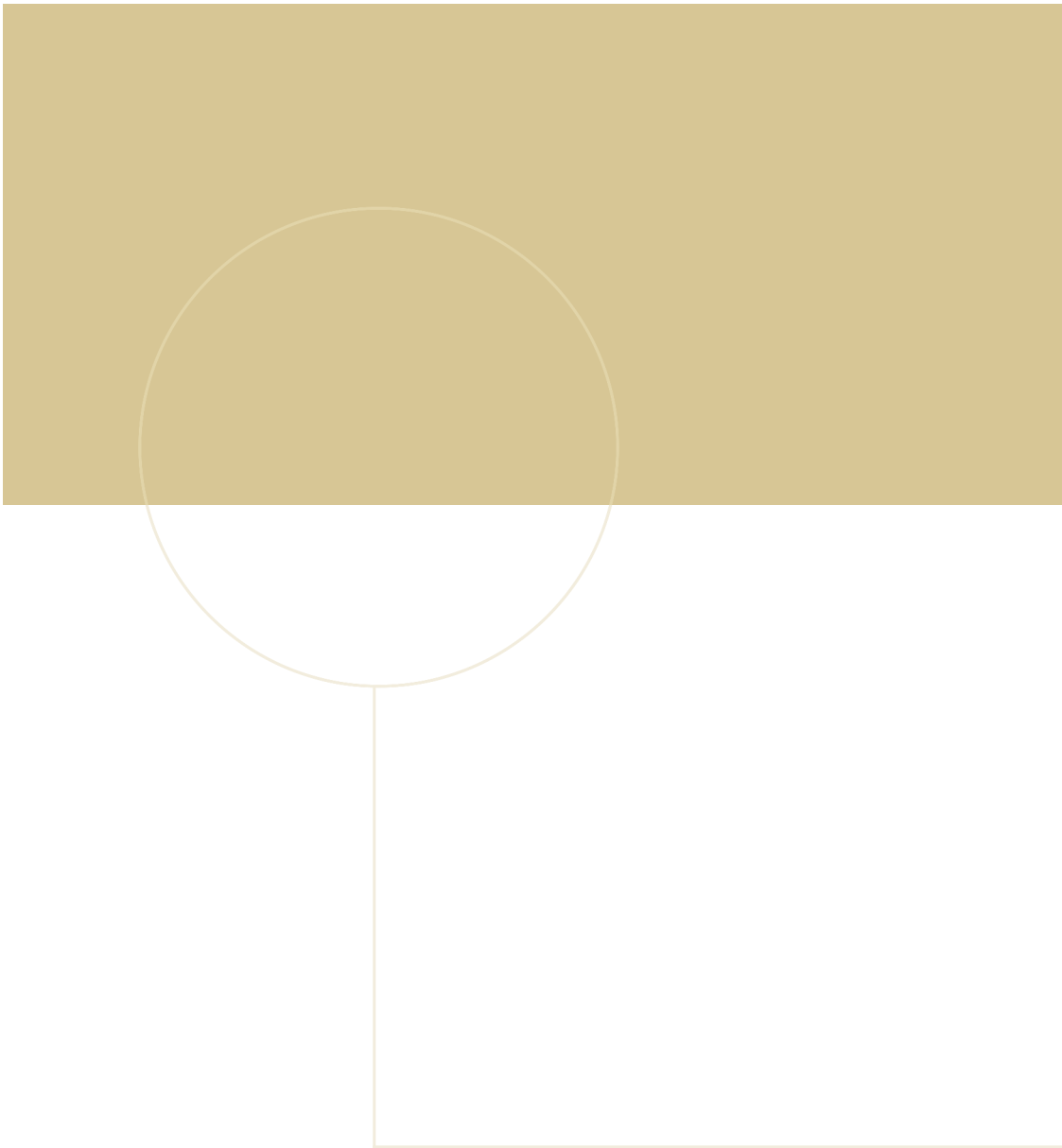
Denne iterasjonen blir kamera funksjonen fullført og rydding av design blir gjort. Legger til en innlogging meny for å gi et eksempel på hvordan dette kan brukes og se ut for oppgavegiver.
Prøvde å ta kontakt med oppgavegiver uten noe svar enda.

**Log 6 – 17.04.2020**

Starte skriving av rapporten. Fullføre noen små feil i koden.

**Log 7 – 01.05.2020**

Bruker tiden på å fullføre rapporten.



NTNU

Kunnskap for en bedre verden