Høgskulen på Vestlandet

BACHELOROPPGAVE

Tidlig oppdagelse og varsling: Utvikling av en mobilapplikasjon for brannsikkerhet i sårbare befolkninger

Early detection and notification: Developing a mobile application for fire safety in vulnerable populations

Gruppe D10:

Torstein Alsaker Eide

Magnus Ivarsen

Rosa Nguyen

DAT191

Fakultet for ingeniør- og naturvitskap

Institutt for datateknologi, elektroteknologi og realfag

Dataingeniør/Informasjonsteknologi

Veileder: Volker Stolz

Innleveringsdato: 22. Mai 2023

Høgskulen på Vestlandet

Fakultet for ingeniør- og naturvitskap

Institutt for datateknologi, elektroteknologi og realfag

TITTELSIDE FOR HOVEDPROSJEKT

Rapportens tittel:	*Dato:*
Tidlig oppdagelse og varsling: Utvikling av en mobilapplikasjon for brannsikkerhet i sårbare befolkninger	*22. Mai 2023*
Forfatter(e):	*Antall sider u/vedlegg: 34*
Torstein Alsaker Eide Magnus Ivarsen Rosa Nguyen	*Antall sider vedlegg:* *62*
Studieretning:	*Antall disketter/CD-er:*
Dataingeniør & Informasjonsteknologi	0
Kontaktperson ved studieretning:	*Gradering:*
Volker Stolz	Ingen
Merknader:	
Ingen	

Oppdragsgiver:	*Oppdragsgivers referanse:*
Building design for At-risk groups (BUILDER)	Ingen
Oppdragsgivers kontaktperson:	*Telefon:*
Arjen Kraaijeveld	+47 52 70 26 57

Sammendrag:

Målet med oppgaven var å utvikle en mobilapplikasjon som kan hjelpe med tidlig deteksjon og varsling av brann. Denne rapporten tar for seg utviklingen av mobilapplikasjonen «Omkring» som lytter etter lyden fra røykvarslere i bakgrunnen, og varsler forhåndsdefinerte kontakter dersom en røykvarsler er oppdaget. Det ble utviklet en maskinlæringsmodell for å kjenne igjen lyden av en røykvarsler.

Stikkord:

Mobilapplikasjon	Brannsikkerhet	Maskinlæring

Høgskulen på Vestlandet, Fakultet for ingeniør- og natuvitskap
Postadresse: Postboks 7030, 5020 BERGEN Besøksadresse: Inndalsveien 28, Bergen
Tlf. 55 58 75 00 Fax 55 58 77 90 E-post: post@hvl.no Hjemmeside: http://www.hvl.no

FORORD

Denne bacheloroppgaven markerer avslutningen på våre studier innen dataingeniør- og informasjonsteknologi ved Høgskulen på Vestlandet. Vi ønsker å uttrykke vår takknemlighet ovenfor våre forelesere Carsten Gunnar Helgesen og Per Christian Engdal, som har gitt oss verdifull kunnskap om hvordan vi skal gå frem i arbeidet med en bacheloroppgave og hva vi må være oppmerksom på underveis. Vi vil også takke vår veileder, Volker Stolz, for hans veiledning og rådgivning i løpet av hele prosessen. Til slutt vil vi takke vår arbeidsgiver, Arjen Kraaijeveld, for å ha støttet og utfordret oss i vårt arbeid med å løse dette problemet.

INNHOLDSFORTEGNELSE

ORDLISTE

Ord	Beskrivelse
Fast Fourier Transform	En algoritme som brukes for å analysere lydsignal, tar det fullstendige lydsignalet og konverterer det til individuelle frekvenser
YAMNET	Dyplærings modell som brukes til lyd klassifisering
Dyplæring	En form for maskinlæring som bruker nevrale nettverk for å autonomt lære etter mønstre i dataset
Spektogram	En visuell framstilling av lyd over tid
Device token	Id som unikt identifiserer en applikasjon på en enhet, brukes for å sende varsler
Arduino	Programmerbar ettkortsdatamaskin, mikrokontroller
IoT enhet	«Internet of Things» enhet, generelt mindre enheter som sensorer
TensorFlow Lite	Maskinlæringsbibliotek for å utvikle maskinlæringsmodeller til mobile enheter
Jupyter Notebook	Interaktivt dokument som lar bruker skrive tekst og kode, og lar brukeren kjøre kode i dokumentet, brukes med python
P2P	«Peer-to-peer», desentralisert kommunikasjon der hvert individ kan kommunisere direkte med hverandre, uten behov for en dedikert server
Desibel (dB)	Måleenhet for lydstyrke
Foreground Service	En tjeneste i Android applikasjoner som lar utviklere kjøre deler av applikasjonen i bakgrunnen

1 INNLEDNING

1.1 Kontekst

Direktoratet for samfunnssikkerhet og beredskap (DSB) har registrert branner i Norge siden 1979. De har påpekt at spesielt sårbare grupper som eldre, pleietrengende, personer med nedsatt funksjonsevne, og rusavhengige, utgjør en stor andel av de som omkommer i branner. Statistikken til DSB viser at rundt 75 prosent av alle brannrelaterte dødsfall i Norge involverer disse gruppene (DSB, 2023).

Faktorer som påvirker dødsrisikoen for disse sårbare gruppene inkluderer alder, alkohol, røyking og redusert fysisk og mental helse (Eivind L, 2022).

Personer over 70 år har fire til fem ganger høyere risiko for å omkomme i brann enn yngre personer (DSB, 2023). SSB-statistikker viser også at en høy andel av eldre, med henholdsvis 87 prosent og 79 prosent for aldersgruppene 65-74 år og 75-79 år, bruker mobiltelefon eller smarttelefon til internettformål (SSB, 2021).

1.2 Motivasjon

I 2019 viste Statistisk sentralbyrå (SSB) at 53 prosent av individer i aldersgruppen 80 år og oppover var bosatt alene i privathusholdninger (SSB, 2019). Dermed har de med nedsatt boevne ulike utfordringer som kan påvirke deres evne til å evakuere under en brann. Dette kan øke risikoen for at brannen oppdages for sent, og føre til at de ikke er i stand til å evakuere i tide.

En løsning for å øke sikkerheten for eldre og sårbare grupper under brannsituasjoner kan være å utvikle av en mobilapplikasjon som kan lytte og kjenne igjen lydsignalet fra røykvarslere, og deretter varsle forhåndsdefinerte kontakter som venner, naboer og familiemedlemmer. En slik mobilapplikasjon kan potensielt bidra til å redde liv og redusere skader ved å oppdage og varsle om brann, før det blir for sent.

Selv om brannvarslingssystemer som kan installeres i boliger allerede eksisterer, kan høye kostnader og andre faktorer gjøre det vanskelig for eldre og sårbare grupper å få tilgang til slike systemer. Dermed kan en mobilapplikasjon være et mer tilgjengelig og rimelig alternativ for å øke brannsikkerheten i hjemmene deres.

1.3 Prosjekteier

Building design for At-risk groups (BUILDER) tar sikte på å øke brannsikkerheten for personer som er spesielt sårbare i brannsituasjoner, som eldre og personer som sliter med rus og psykiske lidelser (HVL, 2023). Prosjektet er ledet av Høgskulen på Vestlandet, målet er å utvide kunnskapen og forståelsen av de utfordringene som disse risikogruppene møter, og samtidig foreslå sikkerhetsløsninger for bygg og innredninger (HVL, 2023).

Arjen Kraaijeveld har jobbet i mange år med brannsikkerhet og er en del av BUILDER-teamet (HVL, 2023). Han er prosjekteier og vil veilede teamet i å utvikle ideer og ressurser som kan bidra til å nå målet.

Vårt prosjekt har til hensikt å utvikle en mobilapplikasjon som kan oppdage røykvarslere i personers hjem. Hvis røykvarsleren går av, skal applikasjonen automatisk varsle forhåndsdefinerte kontakter som venner, naboer og familiemedlemmer via mobilen. En av utfordringene er å kunne registrere lyden fra røykvarsleren. Dermed kan prosjekteieren bidra med kunnskap, ideer og ressurser for å hjelpe teamet i å oppnå prosjektmålene.

1.4 Problembeskrivelse og mål

Målet med denne oppgaven er å utvikle en mobilapplikasjon som kan lytte etter lydsignalet fra en røykvarsler, og varsle forhåndsdefinerte personer (naboer, familie, venner) under brannsituasjoner.

Problemstillingen blir da:

"Hvordan utvikle en mobilapplikasjon som hjelper med tidlig oppdagelse og varsling av brann?"

Det er flere relevante forskningsspørsmål til prosjektet:

- «Er applikasjonen brukervennlig nok til at den er effektiv?»

Applikasjonen avhenger av at brukere er i stand til å forstå brukergrensesnittet i den grad at det ikke blir en hindring i bruk av applikasjonen. Dette er veldig relevant grunnet målgruppen for applikasjonen (eldre, personer med nedsatt funksjonsevne, rusavhengige).

- «Er applikasjonen god nok til å gjenkjenne lyden av røykvarslere til at den kan fungere som et verktøy til tidlig varsling til naboer i situasjoner der umiddelbar assistanse er avgjørende?»

For at applikasjonen skal bli tatt i bruk, må hovedfunksjonaliteten være fungerende og effektiv. Hvis applikasjonen ikke kjenner igjen lyden fra røykvarsleren kan det ende opp i personskade, og i verste fall død.

- "Er applikasjonen energieffektiv nok til å ikke tømme batteriet urimelig raskt?"

For at applikasjonen skal fungere kontinuerlig i bakgrunnen, er det essensielt at den ikke forbruker overflødig strøm fra mobilens batteri.

1.5 Oppbygging av rapporten

Rapporten er delt inn i 7 hovedkapitler:

- Dette kapittelet ga en innledning til prosjektet og dets bakgrunn. Kapitlet tar for seg motivasjonen bak prosjektet, og hva som gjør det viktig og relevant.

Problemstillingene som prosjektet vil adressere og målene for produktet ble også introdusert. Oppdragsgiver og deres behov ble nevnt.

- Følgende kapittel gir en beskrivelse av prosjektet, inkludert tidligere arbeid som er gjort på emnet, kravene som må oppfylles for å utvikle en løsning, samt avgrensninger som må tas hensyn til. Videre er det også beskrevet hvilke ressurser som er nødvendige for å gjennomføre prosjektet, og et litteratursøk om problemstillingen.
- Kapittel 3 presenterer designet av prosjektet hvor ulike alternative løsninger for både frontend, backend og lydgjenkjenning vurderes og drøftes, før en endelig løsning blir valgt. Deretter blir valg av verktøy, prosjektmetodikk og evalueringsplan definert.
- Kapittel 4 omhandler designet av produktet. Brukstilfeller blir definert, og prototyper blir laget for visuell tilbakemelding. Arkitektur blir valgt og applikasjonen blir utviklet. Autentisering, database, lydgjenkjenning og varsling blir implementert.
- Kapittel 5 gir en evaluering av prosjektet, og fremhever de viktigste oppnådde resultatene.
- Kapittel 6 presenterer en kritisk diskusjon av prosjektets løsning i sammenheng med andre løsninger tilgjengelig i markedet. Det blir diskutert styrker og svakheter ved hver tilnærming, og hvordan løsningen vår står opp mot disse alternativene.
- Kapittel 7 presenterer prosjektets konklusjon, der vi fremhever hvordan løsningen vår adresserer forskningsspørsmålene presentert tidligere, og oppsummerer de viktigste funnene og bidragene.

2 PROSJEKBESKRIVELSE

2.1 Praktisk bakgrunn

2.1.1 Tidligere arbeid

Dette prosjektet bygger videre på en semesteroppgave som ble gjennomført innen systemtenkning og innovasjon emnet i 2022. Oppgaven omhandler utformingen av applikasjonen, inkludert dets funksjonalitet, personvern, og økonomi rundt utvikling og vedlikehold. Semesteroppgaven ble skrevet av en gruppe studenter på HVL Campus Haugesund. Gruppen inkluderer studentene Hågen Solbakken, Sebastian Dimmen, Erica Lucia T. Hauge, Eirik R. Nordstrand, Ingrid Vedø og Åsne Almenning.

Det finnes også produkter, og systemer med funksjonalitet som ligner på applikasjonen vi ønsker å utvikle, både i form av programvare og dedikerte fysiske løsninger. Mobilapplikasjoner som Verisure (Verisure, 2019) og Vestfold Audio (Vestfold Audio, 2023) tar i bruk fysisk hardware sammen med en mobilapplikasjon for å varsle om brann. Apple har også funksjonalitet for å varsle brukeren av mobilen når røykvarsleren går av, og man kan bruke Apple Shortcuts for å sende varsel videre til kontakter (Apple, 2023).

2.1.2 Initielle krav

Mobil applikasjonen skal ha følgende funksjonalitet/initielle krav:

1. **Lytte etter lydsignalet fra røykvarsleren:**

Mobilapplikasjonen skal kunne lytte til, og kjenne igjen lydsignalet fra en røykvarsler.

2. **Varsling til forhåndsdefinerte kontakter:**

Når mobilapplikasjonen oppdager et lydsignal fra røykvarsleren, skal den automatisk sende en varsel til forhåndsdefinerte kontakter slik som naboer, familie og venner.

3. **Enkelt brukergrensesnitt:**

Mobilapplikasjonen skal ha et enkelt og intuitivt brukergrensesnitt som gjør det lett å sette opp og bruke.

4. **Autentisering:**

Brukere skal kunne registrere seg med navn, e-postadresse og passord. Registrerte brukere skal kunne logge seg inn med å benytte e-postadresse og passord.

5. **Lavt strømforbruk:**

Mobilapplikasjonen skal kunne være aktiv lenge uten å tømme batteriet urimelig raskt.

6. **Kompatibilitet med flere plattformer:**

Applikasjonen skal være kompatibel med både iOS og Android for å nå så mange brukere som mulig.

2.1.3 Initiell løsnings-idé

Mobilapplikasjonen har som formål å kunne identifisere lyden av en røykvarsler, og må dermed ha tilgang til mikrofonen på enheten. For å oppnå dette kan det utvikles en maskinlæringsmodell som kan gjenkjenne lyden av en røykvarsler. Applikasjonen vil periodisk utføre lytteoperasjoner i fastsatte intervaller mens den opererer i bakgrunnen, med formålet å begrense strømforbruket. Det vil også bli brukt geolokasjon for å være aktiv bare i personens hjem for å spare strøm. Mobilapplikasjonen skal også ha et brukervennlig grensesnitt.

For å sikre bred tilgjengelighet skal applikasjonen kunne brukes på både iOS og Android. Det vil være nødvendig å ha funksjoner som registrering og innlogging med e-postadresse og passord. En innlogget bruker må kunne legge til kontakter som de ønsker å varsle, eller motta varsler fra.

2.2 Avgrensninger

Selv om en mobilapplikasjon kan være en lovende løsning, må det også tas hensyn til flere utfordringer og begrensninger som kan påvirke påliteligheten og effektiviteten til applikasjonen. Mobiltelefonen må ha dekning, og den må være i god tilstand for å sende og motta varsler. Mobilapplikasjonen må være brukervennlig for de eldre og sårbare personer som eventuelt mangler teknisk kompetanse. En annen utfordring er å utvikle applikasjonen til å unngå falske alarmer. Mobilapplikasjonen vil heller ikke tilby å ringe automatisk til brannvesenet i tilfelle av en brann, og dermed er avhengig av forhåndsdefinerte kontakter, eller brukeren selv for å kunne kontakte brannvesenet i en krisesituasjon.

Statistisk sentralbyrå har rapportert en økning i bruken av internett via mobiltelefoner (SBB, 2021). Dette har ført til en ide for å utvikle en mobilapplikasjon som kan hjelpe eldre og sårbare personer som bor alene til å motta nødhjelp i en brannsituasjon.

2.3 Ressurser

Prosjektet er avhengig av ulike ressurser for å sikre en effektiv utvikling. Et integrert utviklingsmiljø er avgjørende for å håndtere utviklingsoppgaver. Videre kreves det en simulator for hver plattform, inkludert en iOS-simulator og en Android-simulator, som gjør det mulig å åpne applikasjonen på datamaskinen under utviklingen og se endringer direkte, uten behov for å distribuere den til en mobiltelefon. Samarbeidet under utviklingen blir muliggjort gjennom bruk av Git og Github for versjonskontroll.

Når det gjelder opplæring av maskinlæringsmodellen, er det nødvendig å ha tilgang til lyder fra røykvarslere for trening. Modellen blir trent opp med tanke på den standarden som brukes i Norge, men det er også viktig å inkludere lyder som ikke kommer fra

røykvarslere for en mer omfattende opplæring. Videre er tilgang til en iOS- og en Android-mobiltelefon nødvendig for testing av applikasjonen.

Designet av det grafiske brukergrensesnittet baseres på sketcher og wireframes, med verdifulle tilbakemeldinger fra oppdragsgiver. For å støtte utviklingsprosessen gis det teknisk veiledning fra vår interne veileder, Volker. Videre er det behov for frivillige deltakere for brukertesting, som bidrar til å evaluere brukervennligheten og ytelsen til applikasjonen.

2.4 Litteratur om problemstillingen

Røykvarslere er en viktig del av brannsikkerheten i hus og hjem. En røykvarsler er en enhet som er designet for å oppdage røyk og varsle ved hjelp av en høy lyd som gir et signal om fare til de som befinner seg i hjemmet. Lyden som spilles av når røykvarsleren aktiveres er vanligvis en gjentakende lyd på rundt 85 desibel, som skal være høy nok til å høres i hele huset eller bygningen.

Frekvensen til lyden som spilles av når røykvarsleren aktiveres kan variere, men generelt ligger den mellom 3 kHz og 3,5 kHz. Dette er et frekvensområde som skal være godt innenfor det menneskelige hørselsområdet, og som skal gjøre det mulig for personer å høre alarmen selv om de befinner seg langt unna røykvarsleren. En studie fra 2018 undersøkte frekvensområdet til røykvarsleralarmer og fant derimot ut at de mest vanlige røykvarslerne med frekvensområde mellom 3 kHz og 3,5 kHz ikke var de mest effektive for varsling av personer i forskjellige aldersgrupper og hørselsnivåer. (Smoke alarm efficiency Waking sleeping occupants, n.d.)

En metode for å analysere lydsignaler, for eksempel fra en røykvarsler, er å bruke Fourier-transformasjon, som kan skille et komplekst signal i en rekke enklere komponenter. Fast Fourier-transformasjon (FFT) er en vanlig algoritme som brukes for å beregne Fourier-transformasjonen til et lydsignal over et tidsintervall. FFT konverterer lydsignalet fra tidsdomene til frekvensdomene, og gir informasjon om frekvensene som er tilstede i signalet.

Dyplæring er en undergren av maskinlæring som fokuserer på å bygge og trene nevrale nettverk med flere lag for å kunne lære komplekse mønstre og representasjoner fra data. En vanlig tilnærming i dyplæring for lydgjenkjenning er å bruke spektrogrammer som inngangsdata til nettverket. Spektrogrammet er en todimensjonal representasjon av lyden som viser frekvensinnholdet, og gir en visuell representasjon av frekvenskomponentene til lyden og hvordan de endrer seg over tid.

Ved å bruke dyplæring -modeller kan man trene nettverket på et stort datasett med lyder og deres tilhørende spektrogrammer. Gjennom trening lærer nettverket å gjenkjenne spesifikke mønstre og trekk i spektrogrammet som er karakteristisk for den ønskede lyden. Dette gjør det mulig å oppnå høy nøyaktighet og pålitelighet i gjenkjenningen av

spesifikke lyder. Ved å benytte dyplæring og spektrogrammer som verktøy kan man oppnå effektiv og presis gjenkjenning av spesifikke lyder, som for eksempel lyden av en røykvarsler. Dette legger grunnlaget for å utvikle en pålitelig applikasjon som kan identifisere og varsle i slike tilfeller (Salomon and Bello, 2017).

3 DESIGN AV PROSJEKTET

Kapittelet gir en oversikt over de foreslåtte løsningene for frontenden, backenden og lydgjenkjenningen av røykvarslere til mobilapplikasjonen. Ulike alternativer blir utforsket, og fordeler og ulemper blir diskutert. Endelig løsning og verktøy blir valgt, og prosjektmetodikk og evalueringsplan blir definert.

3.1 Forslag til front-end løsning

3.1.1 Alternativ front-end løsning 1, individuelle applikasjoner for iOS og Android

En mulig løsning for å utvikle en mobilapplikasjon som fungerer på både iOS og Android, er å utvikle separate applikasjoner for hvert operativsystem. Dette kan gjøres ved å bruke det anbefalte programmeringsspråket for hvert operativsystem. iOS tar i bruk programmeringsspråkene "Swift" og "Objective-C", Android bruker "Java" og "Kotlin") og implementere funksjonalitet for hver applikasjon. (Kotlin Help, 2023)

3.1.2 Alternativ front-end løsning 2, front-end rammeverk for begge platformene

En annen tilnærming for å utvikle en mobilapplikasjon som fungerer på både iOS og Android er å bruke en cross-platform løsning. Dette innebærer å bruke et felles programmeringsspråk og verktøy for å utvikle applikasjonen som kan kjøre på begge operativsystemene. Populære cross-platform rammeverk inkluderer React Native som bruker programmeringsspråket «JavaScript» eller «TypeScript», og Flutter som bruker programmeringsspråket «Dart» (Kotlin Help, 2023).

3.1.3 Diskusjon av front-end alternativene

En fordel med å utvikle en applikasjon for hvert operativsystem er muligheten for å tilpasse funksjonaliteten og designet for hver plattform. Dette gir oss utviklere mer kontroll over hvordan applikasjonen fungerer og ser ut på forskjellige enheter, og kan gi en mer optimal brukeropplevelse.

Ulempene med å utvikle separate applikasjoner for iOS og Android er at det kan være mer tidkrevende og ressursintensivt enn å utvikle en kryssplattformapplikasjon. Det krever også mer kunnskap og erfaring med de spesifikke programmeringsspråkene og utviklingsmiljøene for hver plattform. Videre kan det være mer utfordrende å vedlikeholde og oppdatere to separate applikasjoner. Hvis det er nødvendig å implementere nye funksjoner, må dette gjøres separat i hver applikasjon, og det kan føre til en høyere kostnad på sikt.

En kryssplattformapplikasjon kan dermed være en mer kostnadseffektiv og tidsbesparende løsning, da man kun trenger å utvikle en applikasjon som kan kjøre på flere plattformer. Dette gir også fordelen av en mer konsistent brukeropplevelse på tvers av ulike enheter og plattformer. En annen fordel er at man kun trenger å lære og bruke ett

programmeringsspråk og et utviklingsmiljø, noe som kan redusere læringskurven for oss som utviklere og gjøre det lettere å vedlikeholde og oppdatere applikasjonen på tvers av plattformer.

En ulempe med kryssplattformapplikasjoner inkluderer at man noen ganger kan støte på begrensninger i funksjonaliteten og ytelsen på ulike plattformer. Dette kan være spesielt relevant når det gjelder applikasjoner som krever tilgang til spesifikke maskinvarefunksjoner. I tillegg kan det være utfordrende å tilpasse applikasjonen for hver plattform, da noen funksjoner og designelementer kan fungere bedre på en plattform enn på en annen.

Valget mellom separate applikasjoner for iOS og Android og en kryssplattformapplikasjon er avhengig av flere faktorer, som tidsramme, budsjett, funksjonalitet, designkrav og ressursene tilgjengelige for prosjektet.

3.2 Forslag til back-end løsning

3.2.1 Alternativ back-end løsning 1, peer-to-peer

En P2P-løsning er en type distribuert system der flere enheter samarbeider og deler ressurser og tjenester direkte med hverandre, uten å være avhengig av en sentralisert server. (merriam-webster, 2023) Dette gjør P2P-løsninger attraktive for applikasjoner der det er behov for høy grad av desentralisering, sikkerhet, og autonomi. En måte å oppnå et P2P-nettverk for applikasjonen er ved å utnytte mobiltelefonenes innebygde SMS-tjenester og funksjoner for å sende og motta meldinger (SMPP, 2023). Ved å bruke SMS-protokollen kan mobiltelefonene kommunisere direkte med hverandre uten behov for en sentralisert server.

3.2.2 Alternativ back-end løsning 2, selvhostet server

Et annet alternativ er å sette opp en egen selvhostet server som er tjener for applikasjonens backend, og som driftes og vedlikeholdes av oss utviklere selv. Meldinger om at røykvarsleren har gått av blir sendt fra brukerens telefon til serveren, som deretter tar hånd om å varsle brukerens kontakter. En selvhostet server kan også gi større grad av kontroll og fleksibilitet, med full tilgang til og mulighet for endre konfigurasjonen av serveren og dens ressurser.

3.2.3 Alternativ back-end løsning 3, skytjeneste

En skyløsning er en type distribuert system der applikasjonens backend driftes og vedlikeholdes av en tredjeparts skytjeneste, som for eksempel Amazon Web Services (AWS) eller Google Cloud Platform (GCP). Dette kan gi mange fordeler for applikasjoner som trenger å skalere og håndtere større mengder trafikk, da skyløsninger ofte kan tilby høy tilgjengelighet, elastisitet og pålitelighet. (Microsoft, 2023) I tillegg kan skyløsninger også ha mange integrerte tjenester og verktøy for databehandling, analyse, sikkerhet,

autentisering og administrasjon. Det er en mindre kompleks løsning ettersom man verken trenger å sette opp P2P eller bygge og rulle ut en egen server. Imidlertid vil skyløsninger også ha noen ulemper som høye driftskostnader, avhengighet av en tredjepart, og mindre grad av kontroll og fleksibilitet over serverinfrastrukturen.

3.2.4 Diskusjon av back-end alternativene

Når det gjelder valget mellom disse tre alternativene, er det viktig å vurdere fordeler og ulemper ved hver løsning. En peer-to-peer-løsning kan være en enkel og billig måte å implementere systemet på, da det ikke krever noen ekstra infrastruktur. Det kan også være raskt og effektivt å sende data direkte mellom enheter. Imidlertid kan det være en utfordring å opprettholde påliteligheten i en slik løsning, da tilgjengeligheten og stabiliteten til andre enheter kan variere.

En selvhostet server-løsning gir mer kontroll og stabilitet, siden man kan kontrollere og overvåke trafikken på serveren selv. Det gir også mer fleksibilitet når det gjelder funksjonalitet og skalerbarhet, siden man kan legge til og fjerne ressurser etter behov. Imidlertid kan det være kostbart å sette opp og drifte en egen server. I tillegg kan det være en utfordring å sikre at serveren alltid er tilgjengelig, for eksempel hvis man har behov for å oppdatere programvare eller komponenter.

En skytjeneste-løsning kan være enkel å sette opp og skalere, da man kan leie ressurser fra en tredjepart. Dette gir også høy tilgjengelighet, siden skytjenester ofte har redundante servere som sikrer at tjenesten alltid er tilgjengelig. (Microsoft, 2023) Imidlertid kan det være kostbart å leie ressurser fra en tredjepart, spesielt hvis antall brukere øker betydelig. Videre kan man miste noe kontroll over sikkerhet og personvern, da dataene vil bli lagret på servere som man ikke selv kontrollerer.

3.3 Forslag til røykvarsler lydgjenkjenning løsning

3.3.1 Alternativ røykvarsler lydgjenkjenning løsning 1, frekvens gjenkjenning

En metode for å gjenkjenne lyden til en røykvarsler er å bruke Fast Fourier Transform (FFT) for å finne den dominante frekvensen i lyden. FFT er en algoritme som finner ut hvilke frekvenser som er til stede i et lydsignal og deres relative intensitet. Ved å analysere lyd som en funksjon av frekvensen, kan man identifisere den dominante frekvensen og sammenligne den med røykvarsleren sin frekvens. FFT er en relativt enkel metode for å gjenkjenne lyden til en røykvarsler.

3.3.2 Alternativ røykvarsler lydgjenkjenning løsning 2, dyplæring

En annen metode for å gjenkjenne lyden til en røykvarsler er å bruke en dyplæring modell. Dyplæring er en form for maskinlæring som tar i bruk nevrale nettverk med flere lag (Howard, J. og Gugger, S., 2020). Dyplæring brukes ofte på bilder for å kjenne igjen hva klasse et bilde hører til i, og det kan brukes samme metoden på lyd ved å framstille lyden

som et bilde, ved hjelp av FFT algoritmen (Howard, J. og Gugger, S., 2020). For å trene en dyplæring modell, er det behov for store mengder lyddata fra røykvarslere, slik at modellen kan gjenkjenne lyden fra en røykvarsler på egenhånd. Det er også behov lyder som ikke er fra røykvarslere for å trene opp hva som er røykvarslere og hva som ikke er røykvarslere.

3.3.3 Diskusjon av røykvarsler lydgjenkjenning alternativene

Begge alternativene for gjenkjenning av lyden til en røykvarsler har sine fordeler og ulemper. Fast Fourier Transform-metoden er en enklere og mindre ressurskrevende løsning som kan implementeres på en måte som er mindre data- og prosesseringskrevende. Den kan også gi en nøyaktighet som er tilstrekkelig for mange situasjoner. På den andre siden kan denne metoden være mindre nøyaktig enn en dyplæring modell, spesielt når det kommer til å skille mellom forskjellige lydkilder. I tillegg kan denne metoden ha vanskeligheter med å skille mellom andre alarmer som produserer høye lyder med lignende frekvenser som røykvarslere.

Dyplæring modeller har blitt stadig mer populære fordi de kan gi bedre nøyaktighet enn tradisjonelle metoder. Disse modellene kan skille mellom forskjellige lydkilder, inkludert andre alarmer som produserer lyder med lignende frekvenser som røykvarslere, noe som gjør dem til et mer pålitelig alternativ. Imidlertid krever disse modellene store mengder data og prosessering for å bli trent og kan være ressurskrevende. Implementeringen av disse modellene kan også være mer krevende, spesielt hvis det kreves spesiell maskinvare for å kjøre dem i sanntid. Dermed vil valg av løsning avhenge av en rekke faktorer, inkludert krav til nøyaktighet og tilgjengelige ressurser.

3.4 Valgt løsning

Ved begynnelsen av utviklingen av mobilapplikasjonen, ble kryssplattform-rammeverket React Native valgt for frontend, da det ville spare tid og arbeid framfor å utvikle separate applikasjoner for iOS og Android. Imidlertid oppsto tekniske utfordringer knyttet til implementering av lydgjenkjenningsløsningen, som førte til avgjørelsen å bytte løsning til separate applikasjoner for hvert operativsystem. Etter uker med utvikling så Android løsningen mer lovende ut enn iOS løsningen, det ble deretter vurdert i samarbeid med veileder at fokuset skulle bli å utvikle en applikasjon for Android for å sikre ferdigstillelse av et produkt.

For backend ble det valgt en skyløsning, da dette medførte muligheten til å skalere applikasjonen etter behov og redusere behovet for vedlikehold. Ved å bruke en skyløsning ble det også mulig å benytte et bredt spekter av tjenester og verktøy som allerede var tilgjengelige i skyen, som autentisering av brukere, persistering av brukerdata, og å sende meldinger mellom brukerne.

Når det gjaldt lydgjenkjenning ble både FFT og dyplæring alternativet utforsket. Valget falt på dyplæring på grunn av dens høyere nøyaktighet og evne til å håndtere ulike lydkilder og miljøforhold. Selv om det krever mer ressurser og tid for å trene en slik modell, ble det konkludert at det ville gi en mer pålitelig og presis løsning for å oppdage røykalarmer, som igjen ville føre til en bedre brukeropplevelse og mer pålitelige resultater.

3.5 Valg av verktøy

Valget av verktøy for utvikling av mobilapplikasjonen var basert på flere faktorer. For Android utvikling ble Android Studio valgt, som er den offisielle IDEen fra Google for utvikling av Android applikasjoner. Xcode ble valgt for utvikling av iOS applikasjonen, da dette er det primære verktøyet for utvikling av iOS applikasjoner og tilbyr en rekke funksjoner for å optimalisere ytelsen til applikasjonen.

For lydgjenkjenning ble Tensorflow Lite valgt, et open-source maskinlæringsbibliotek som tilbyr optimalisering av ytelse for mobile enheter.

Vi valgte Firebase som skytjeneste-leverandør for å enkelt kunne håndtere autentisering, lagring av data og push-notifikasjoner, og for å unngå behovet for å bygge og administrere en egen server.

Disse digitale verktøyene ble valgt med omhu for å sikre best mulig resultat for våre brukere. Samlet sett var valget av disse verktøyene basert på deres funksjonalitet, popularitet og pålitelighet, samt deres evne til å effektivt håndtere de tekniske utfordringene som oppstod under utviklingen og testingen av applikasjonen.

3.6 Prosjektmetodikk

3.6.1 Utviklingsmetodikk

I prosjektet ble Scrum valgt som den foretrukne prosjektmetodikken. Ramsøy (2022) nevner at scrum-metoden består av tre grunnleggende roller, som er produkteieren, utviklingsteamet og scrum masteren. Scrum masteren fungerer som prosjektleder og har ansvaret for å sørge for at alle i teamet har det de trenger for å utføre arbeidet sitt på en effektiv måte. I hennes blogg påpeker hun at metoden er også kjent for å dele opp arbeidet i sprinter, hvor hver sprint har spesifikke oppgaver som må utføres. Hun fremhever også at Scrum-metoden har vist seg å være effektiv for å jobbe med høy innsats og være åpen for utforskning, samtidig som den gir en strukturert tilnærming til arbeidet. Dette bidrar til at teamet kan jobbe mer effektivt og følge en plan som fører dem nærmere målet på en mer smidig måte. Den gir et rammeverk som fremmer samarbeid, kommunikasjon og fleksibilitet (Ramsøy, 2022).

Ved å bruke Scrum-metoden, kan teamet enkelt justere kursen basert på utfordringer som oppstår underveis, samtidig som de kan holde seg fokusert på det overordnede målet.

Denne tilnærmingen til prosjektledelse har vist seg å være effektiv for å oppnå målene på en strukturert og samarbeidsorientert måte.

3.6.2 Prosjektplan

Prosjektplanen (vist i prosjekthåndboken, Vedlegg 1) er delt inn i flere epoker. Den første er oppstart som hovedsakelig inneholder møter med oppdragsgiver og veileder. Den neste er forberedelse, her blir problemet definert, og en løsning planlagt. Videre kommer utvikling, hvor selve arbeidet med å skape applikasjonen skjer. Til slutt kommer evaluering og rapportskriving, her presenteres, evalueres og diskuteres resultatet, og presenteres til oppdragsgiver.

Prosjektplanen ble endret like før påske. Gruppen startet å utvikle applikasjonen med React Native, et front-end rammeverk som lar utviklere lage en applikasjon som fungerer både på iOS, Android og web. Det ble funnet flere biblioteker for håndtering av lyd i React Native, men det ble oppdaget under utvikling at ingen av disse hadde tilfreds funksjonalitet eller dokumentasjon til å bli tatt i bruk uten omfattende modifikasjon. Det ble da bestemt at det utvikles selvstendige applikasjoner til hver plattform.

Med dette ble skjemaet skjøvet flere uker tilbake. Maskinlæringsmodellen og dataen som ble samlet var fortsatt nyttig, det var også skyløsningen som ble satt opp i Firebase, men hele frontend-delen måtte lages på nytt.

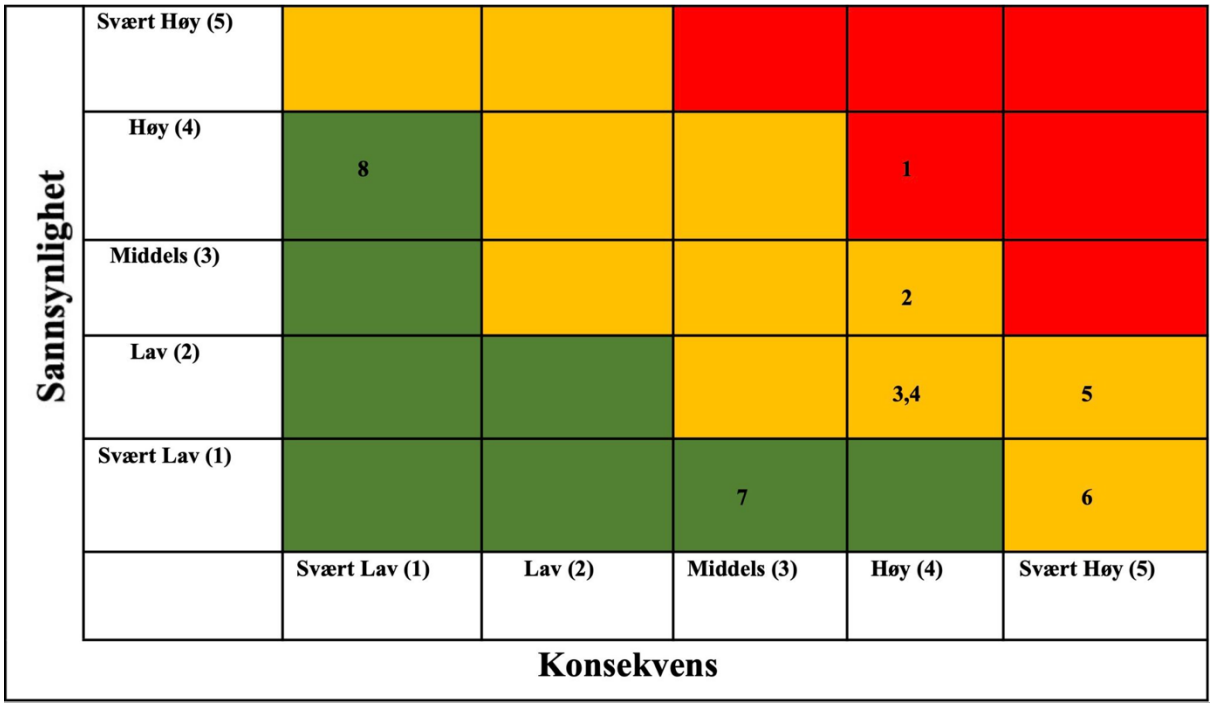
3.6.3 Risikovurdering

Risikoanalysen, illustrert i tabell 3.1, viser årsaken, sannsynligheten, og konsekvensen av de forskjellige risikoene, og definerer tiltak dersom det skulle oppstå.

	Hendelse /Risiko	Årsak	Sannsynlighet	Konsekvens	Risiko-produkt	Tiltak
1	For ambisiøse tekniske løsninger, kommer ikke i mål	Lite innsikt i tilgjengelig teknologi	Lav (4)	Høy (4)	**16**	Lese oss opp på og lær om tilgjengelig teknologi
2	Applikasjonen blir ikke tatt i bruk.	Ineffektiv og lite brukervennlig brukerdialog.	Middels (3)	Høy (4)	**12**	Gjennomføre brukertesting og markedsføre
3	Oppfyller ikke oppdragsgivers visjon	Lite effektiv kommunikasjon underveis	Lav (2)	Høy (4)	**8**	Planlegge og gjennomføre jevnlige møter
4	Utviklingen tar for lang tid, kommer ikke i mål	Dårlig planlegging	Lav (2)	Høy (4)	**8**	Detaljert framdriftsplan
5	Oppfyller ikke regler for sikkerhet og personvern	Dårlig planlegging	Lav (2)	Høy (5)	**10**	Detaljert framdriftsplan
6	Datainnbrudd / Hackere	Dårlig datasikkerhet	Svært Lav (1)	Svært Høy (5)	**5**	Sette oss inn i datasikkerhet
7	Intern konflikt mellom gruppemedlemmer	Uenigheter	Svært Lav (1)	Middels (3)	**3**	Kontinuerlig kommunikasjon
8	Sykdom/fravær	Sykdom/jobb	Høy (4)	Svært Lav (1)	**4**	Spise vitaminer / Skulke jobb

Tabell 3.1: Risikoanalyse

Risikoene blir framstilt i en risikomatrise i figur 3.1.



Figur 3.1: Risikomatrise

Risikoer blir vurdert av risikoproduktet, som er beregnet ved:

Sannsynlighet * Konsekvens = Risikoprodukt

3.7 Evalueringsplan

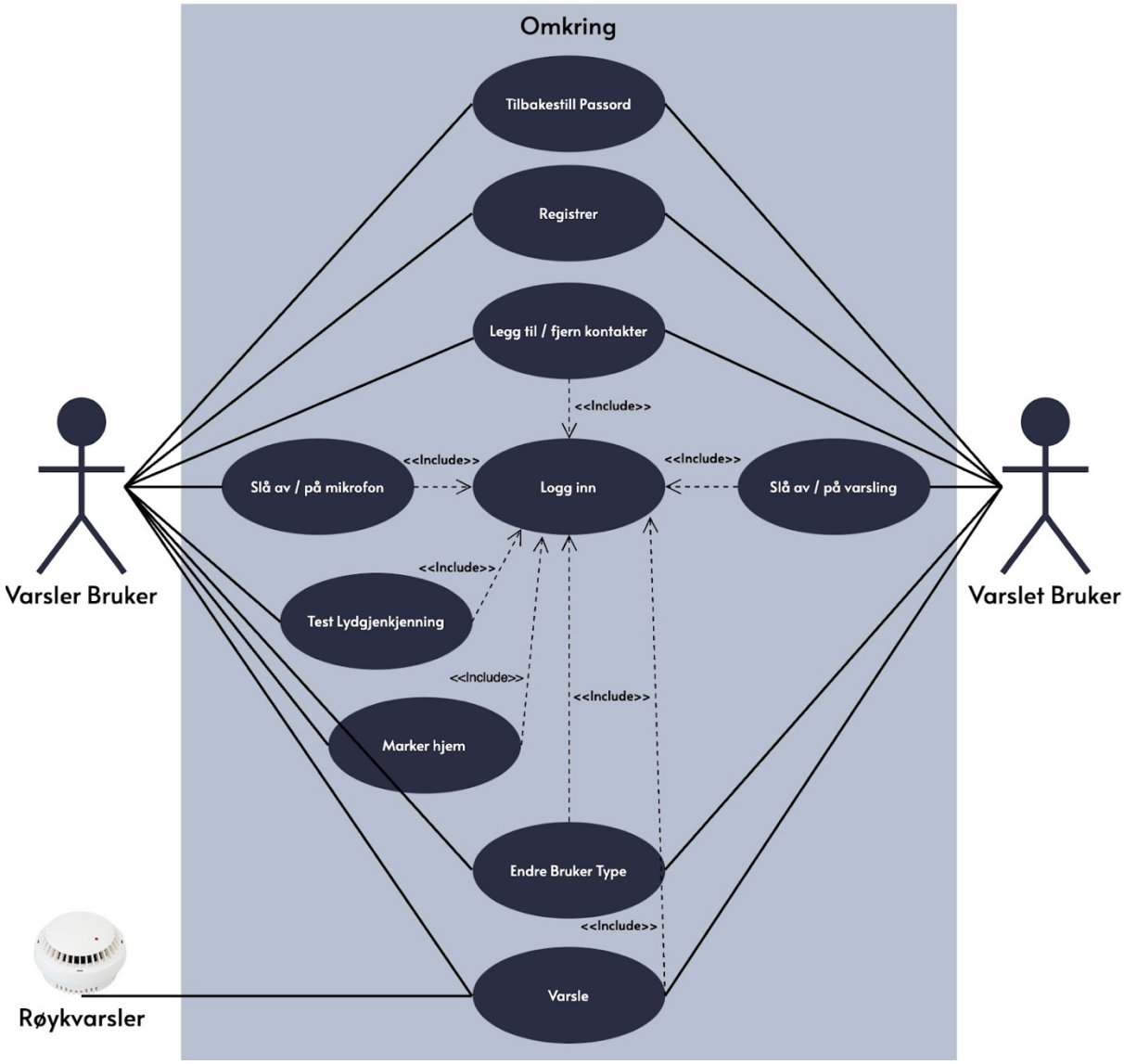
Funksjonaliteten av applikasjonen skal bli testet underveis i utviklingen. Registrering, innlogging, tilbakestilling av passord, og persistering av brukerdata skal bli testet. Brukertesting skal bli utført for brukergrensesnittet (registrering, innlogging, oppsett av varsling og legge til kontakter) for å teste om brukergrensesnittet er lett nok til å forstå og bruke for målgruppen til applikasjonen.
En akseptansetest for applikasjonen skal bli gjort, der røykvarsler-gjenkjenningen testes på ekte røykvarslere med flere mobiler, i diverse rom, med diverse avstander fra røykvarsleren. Dette ble så presentert for oppdragsgiveren.

4 Produktdesign

Kapittelet fokuserer på produktdesignet av prosjektet, og dekker brukstilfeller, aktører og funksjoner. Det blir lagt vekt på brukerautentisering, kontaktsystemet, push-varsling og røykvarsler lydgjenkjenning. Kapittelet utforsker designet av brukergrensesnittet gjennom skisser, wireframing og en Figma-prototype. Kapittelet beskriver også applikasjonsutvikling, inkludert registrering/pålogging, dashbordfunksjoner og implementeringen av lydgjenkjenningen i mobilapplikasjonen.

4.1 Brukstilfeller

Brukstilfellediagramet, illustrert i Figur 4.1, viser de forskjellige aktørene, og brukstilfellene som kan taes i bruk.



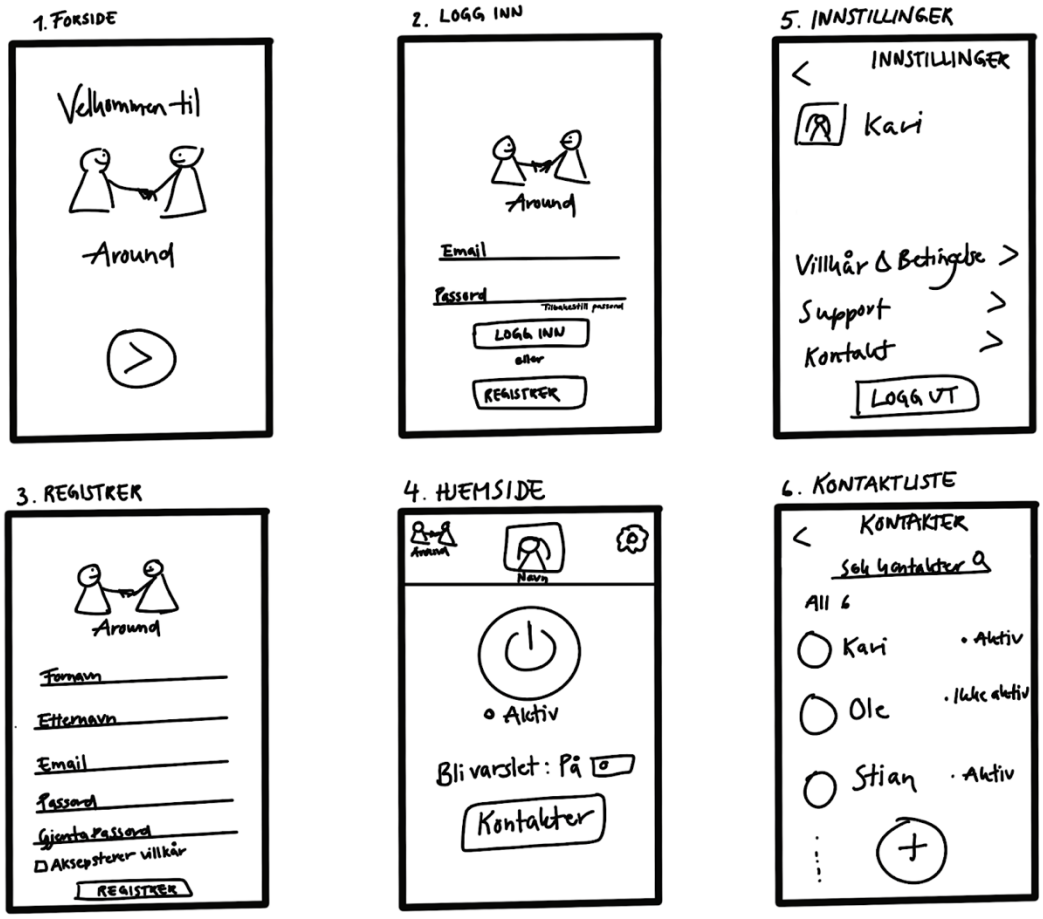
Figur 4.1: Brukstilfellediagram

Det er hovedsakelig 2 aktører i applikasjonen, en bruker som sender varsler til naboer/familie/venner når brukerens røykvarsler går av og blir oppdaget, og en bruker som mottar varsler når en kontakt sin røykvarsler går av. Brukstilfellebeskrivelsene er inkludert i kravdokumentet (Vedlegg 3), men et kort sammendrag er inkludert her.

For å ta i bruk mesteparten av funksjonaliteten må brukeren være innlogget, dette blir gjort med e-postadresse og passord. For å kunne logge inn må brukeren først ha registrert en konto med navn, e-postadresse og passord. Hvis brukeren har glemt passordet skal brukeren kunne tilbakestille passordet ved å melde ifra i applikasjonen, og deretter motta og klikke på lenken i en e-post. Når en bruker er logget inn i applikasjonen kan brukeren legge til kontakter som skal bli varslet. For å forsikre oss at brukeren sin røykvarsler er kompatibel med vår maskinlæringsmodell skal det lages en test modus for lydgjenkjenningen. Test modusen skal ikke varsle brukerens kontakter, men viser om røykvarsleren aktiverer lydgjenkjenningen eller ikke. Brukeren skal kunne legge inn posisjonen til sitt eget hjem, slik at mikrofonen ikke kjører aktivt når brukeren ikke er hjemme, dette med tanke på både personvern og batterieffektivitet. Det er også mulig for brukeren å slå av og på lydgjenkjenningen.

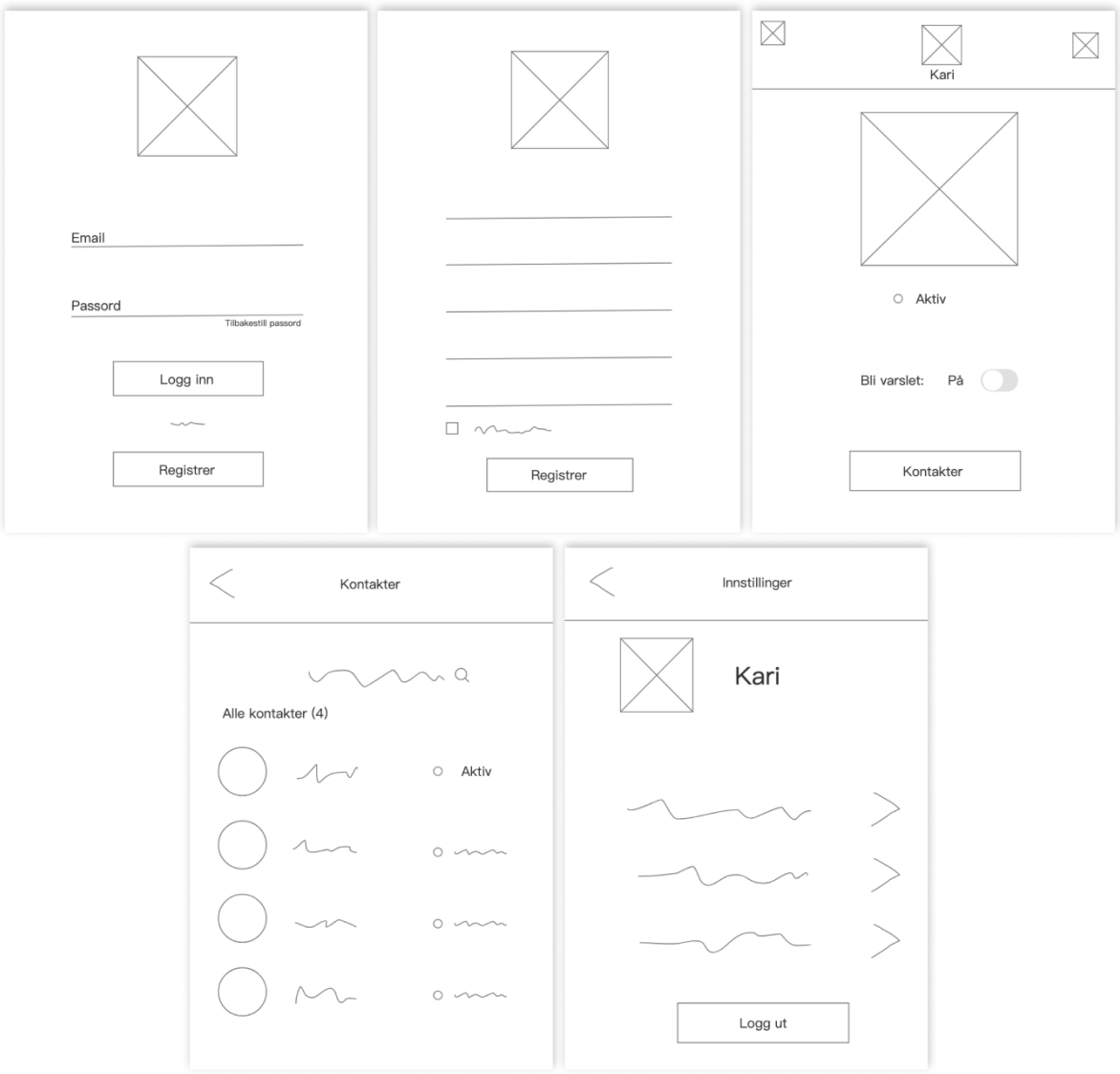
4.2 Brukergrensesnitt

4.2.1 Sketsj og wireframes



Figur 4.2: Sketsj av mobilapplikasjonen

Å tegne sketsjer av brukergrensesnittet er en del av startfasen som hjelper mot visualisering for frontend-utviklere. Først blir de tegnet på forhånd og senere blir det gjort til wireframes i utviklingsfasen når man har kommet frem til det beste løsningsalternativet. Det er kun den generelle strukturen og logikken for designet som vises frem her. Dette gjør at det åpner mot ulogiske løsningsalternativer som gjør det lettere for utviklere å finne effektive løsninger for brukergrensesnittet.

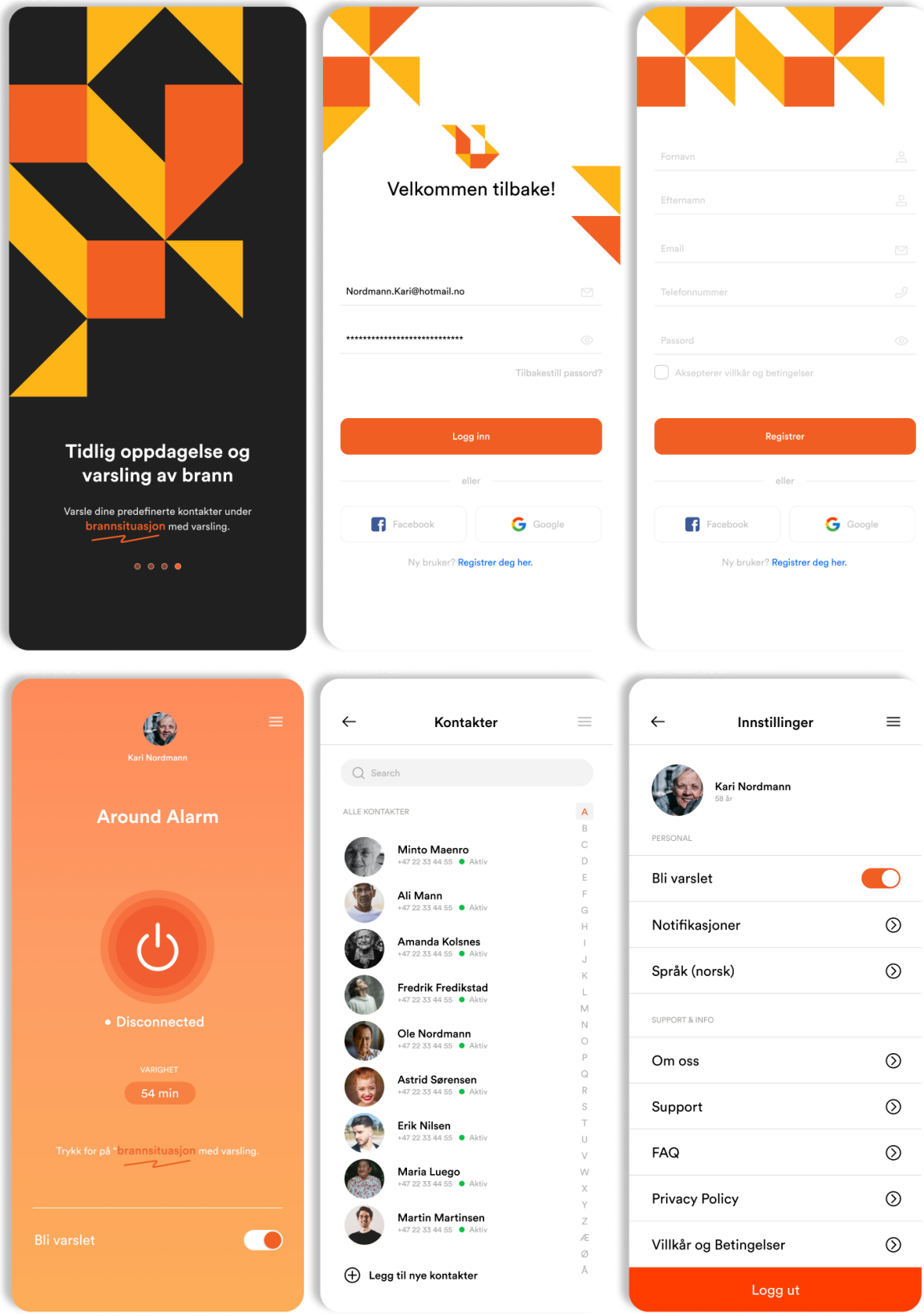


Figur 4.3: Wireframe

Etter å ha laget en sketsj kommer wireframes, som er vist i figur 4.3. Wireframes er en måte å skissere en app og blir brukt som verktøy for utviklingen av grensesnittet. Meningen er å gi en visuell forståelse av sidestruktur, layout, brukerflyt og funksjonalitet (UX design institute, 2022).

Wireframesene ble inspirert av semesteroppgaven i systemtenkning og innovasjon, samtidig integrerte vi våre egne ideer og konsepter. Wireframes skal gi et inntrykk på hvordan den endelige applikasjonen vil se ut og hjelpe til med å ta viktige beslutninger om layout og funksjonalitet før selve programmeringen starter. Den gir også en mulighet til å diskutere ideer og konsepter på tidlig stadium av prosjektet.

4.2.2 Figma prototype



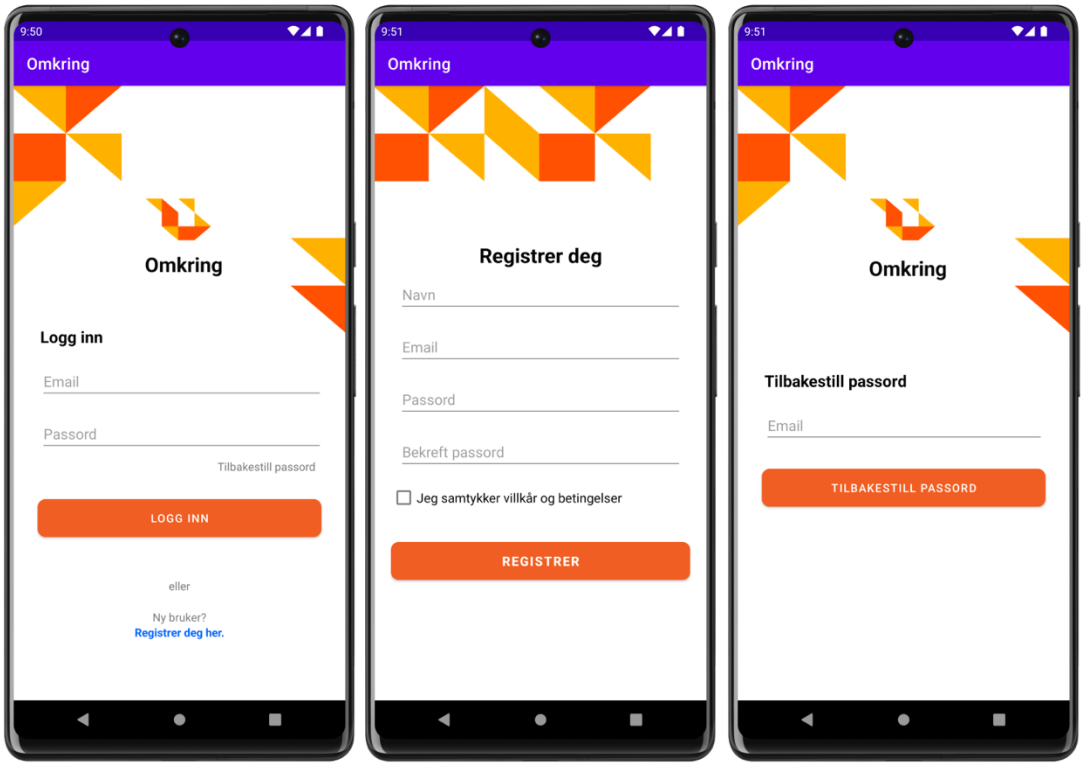
Figur 4.4: Figma prototype

Prototyping kan være en verdifull ressurs for å utvikle og evaluere designløsninger på et tidlig stadium i utviklingen av en applikasjon. Ved å utvikle en prototype kan man teste ut designet og funksjonaliteten før selve programmeringen starter. Dette kan bidra til å identifisere potensielle utfordringer og forbedringsmuligheter på et tidlig stadium av prosjektet (Interaction design foundation, 2023).

Resultatet av prototypen, vist i figur 4.4, er tatt i hensyn med tanke på plassering av funksjonene, skriftstørrelse, fargebruk og fonttype for lesbarhet. Dette kan føre til en mer brukervennlig applikasjon og redusere risikoen for designfeil i den endelige applikasjonen. I tillegg ble det også inkludert tilbakemeldinger fra oppdragsgiveren på wireframes for å kunne tilfredsstille brukerens behov og ønske.

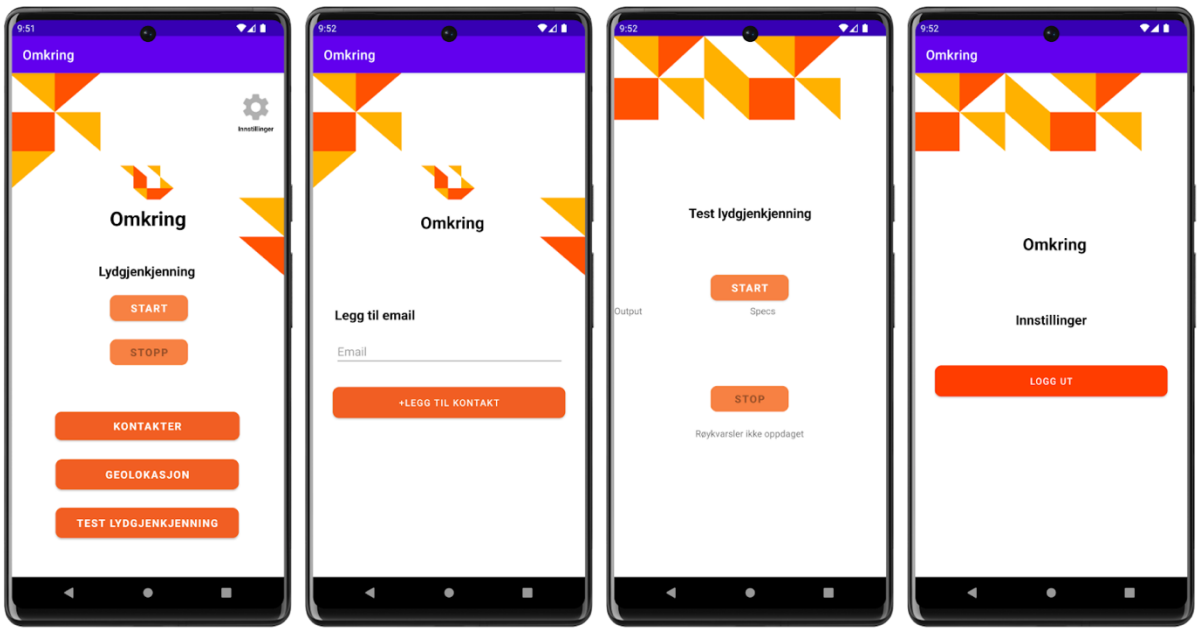
4.2.3 Endelig applikasjon

Når appen åpnes kommer det opp en innloggingsside hvor brukeren kan logge inn med e-postadresse og passord. For å kunne logge inn må en bruker ha registrert seg, dette kan brukeren gjøre ved å trykke på "Registrer deg her" og vil bli tatt til en ny side der du blir bedt om å oppgi personlige detaljer som navn, e-postadresse og passord. Deretter klikker man på "Registrer" for å fullføre registreringen.



Figur 4.5: Grafisk brukergrensesnitt før inlogging

Etter at brukere er logget inn lander de på dashboard siden, vist til venstre i figur 4.6, her styres all hovedfunksjonaliteten til applikasjonen. Når dashbordet åpnes kommer det opp flere forespørsler. Applikasjonen spør om tilgang til mikrofon, lokasjonstjenester, og push-varsler.



Figur 4.6: Grafisk brukergrensesnitt etter innlogging

For å forsikre oss at lydgjenkjenningen fungerer på brukerens telefon, har vi laget en testmodus. Her kan brukeren starte og stoppe lydgjenkjenningen uten at den sender varsler til brukerens kontakter. Brukeren får også opp litt mer informasjon, som hvor sikker modellen er på klassifiseringen til de forskjellige lydene. En start og stopp knapp blir vist for å ta på og av lydgjenkjenningen. Teknisk informasjon om opptaket av lyden er inkludert. Og en tekstboks som viser om en røykvarsler er oppdaget eller ikke.

For å starte og stoppe hovedlydgjenkjenningen er det en knapp på dashbordet brukeren kan trykke på. Når lydgjenkjenningen kjører kommer det opp en varsel om at den kjører aktivt i bakgrunnen. Denne varselen kommer som resultat av at applikasjonen tar i bruk en foreground service, som er en måte å la applikasjoner kjøre i bakgrunnen, mer om dette senere.

4.3 Lydgjenkjenning

4.3.1 Definere problemet

Det er satt som mål å utvikle en modell som er i stand til å gjenkjenne lyden til alarmen fra en røykvarsler. Denne modellen skal bli implementert i en mobilapplikasjon som vil fungere på smarttelefoner. Hovedformålet med applikasjonen er å sende varsler til brukerens familie, venner og/eller naboer når alarmen fra en røykvarsler blir oppdaget. Lydene fra røykvarslere har visse standardiserte karaktertrekk, men det finnes ulike standarder avhengig av produsent og region.

For å oppnå dette målet, er det planlagt å ta i bruk en eksisterende dyplæring modell kalt YAMNET som et utgangspunkt, og bygge videre på denne ved hjelp av såkalt transfer learning. YAMNET modellen er spesielt laget for å kjøre på mobile enheter, dette er viktig

for våres applikasjon med tanke på personvernshensyn - at lydene ikke blir sendt vekk til en server for å bli analysert. Et minimumskrav er å kunne gjenkjenne en standard brannalarmer med et frekvensområde rundt mellom 3,0 og 3,5 kHz når smarttelefonen befinner seg i samme rom. Ideelt sett så oppnås pålitelig gjenkjenning selv når telefonen er plassert i brukerens lomme, i et annet rom, eller når det er andre lyder som bakgrunnsstøy.

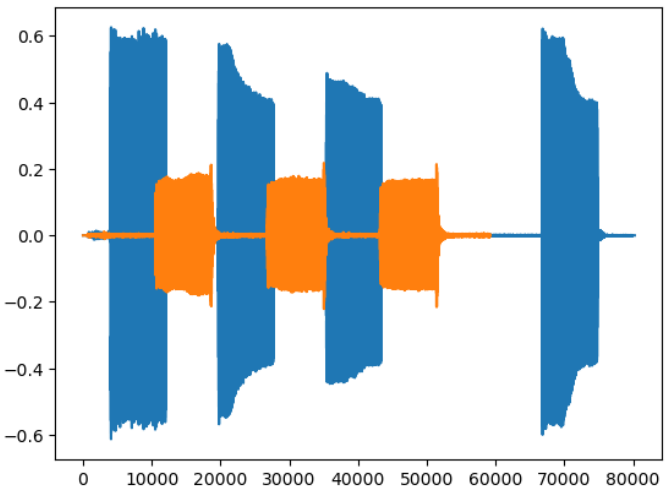
4.3.2 Samle data

Et nytt datasett blir opprettet ved å samle inn lyder fra røykvarslere tilgjengelig på nettet, samt ta opptak ved hjelp av våre egne mobiltelefoners innebygde mikrofoner. For å bygge et pålitelig datasett er det behov for flere hundre opptak som merkes som røykvarslere. Disse opptakene vil variere, noen vil inneholde klare og uavbrutte røykvarslerlyder, mens andre vil bli blandet med bakgrunnsstøy som lyd fra TV, musikk, samtale, støvsuging osv. Det vil også inkluderes eksempler på andre lyder som kan forveksles med røykvarslerlyder, for eksempel sirener fra utrykningskjøretøy, pipelyder fra mikrobølgeovn, vekkerklokke, osv. Samtidig blir det også inkludert lyder som ikke er røykvarslerlyder. For å øke variasjonen i datasettet kan det også genereres lyder ved å blande tidligere røykvarslerlyder med ikke-røykvarslerlyder.

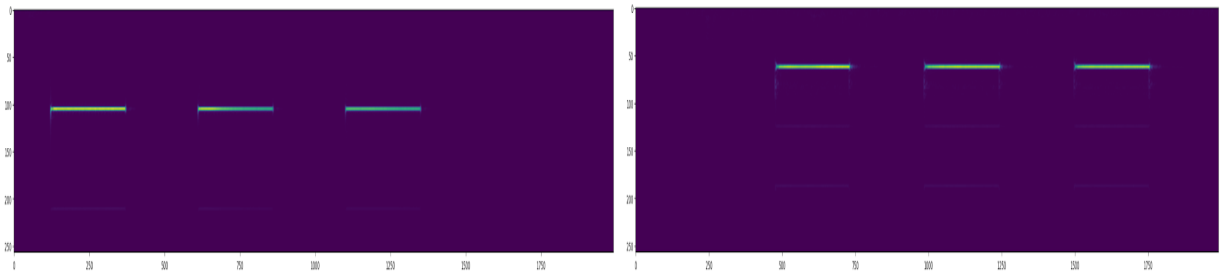
4.3.3 Utforske data

Strukturen av datasettet består av to hovedmapper i rotmappen, en for trening og en for testing. Hver av disse hovedmappene er delt inn i to undermapper: en for lyder fra røykvarslere og en for lyder som ikke er fra røykvarslere.

Det finnes flere måter å skille lydene mellom røykvarslere og andre lyder på. I figur X.X vises to lyder: en mikrobølgeovnlyd i oransje og en røykvarslerlyd i blått. Begge lydene slår seg av og på med en lignende rytme, så rytmen til lyden alene kan ikke brukes til å identifisere en røykvarsler. Imidlertid er lydnivået betydelig høyere på røykvarsleren sammenlignet med mikrobølgeovnen.



Figur 4.7: Lydnivå av røykvarsler (i blå) og mikrobølgeovn (i orange)



Figur 4.8: Spektogram av røykvarsler og mikrobølgeovn

Hvis vi representerer de samme lydene som spektrogrammer ved hjelp av en Fast Fourier Transform (FFT), får vi enda mer informasjon. I figur 4.8 vises to spektrogrammer: til venstre vises mikrobølgeovnlyden, og til høyre vises røykvarslerlyden. Her ser vi tydelig en forskjell i frekvensen på lydene. Dermed kan vi klassifisere lydene basert på styrken, frekvensen, og rytmen mellom av og på.

Ved å analysere spektrogrammet kan ulike egenskaper utnyttes som igjen kan hjelpe oss med å skille mellom lydene. For eksempel er det mulig å se etter spesifikke frekvensområder som er karakteristiske for røykvarslerlyden. Det kan også observere mønstre i tids- og frekvensdomenet som skiller seg fra andre lyder. Ved å kombinere informasjonen fra styrke, frekvens og mønstre er det mulig å på en pålitelig måte klassifisere lyder som enten røykvarslerlyder eller andre typer lyder.

Det å benytte seg av spektrogrammer i lydgjenkjenning gir oss et mer detaljert bilde av lydens egenskaper og gjør det mulig å utnytte mer avanserte maskinlæringsteknikker for å oppnå nøyaktig og pålitelig klassifisering av røykvarslerlyder.

4.3.4 Trene modell

For å trene en ny modell kan man utvide datasettet ved å legge til flere eller endre lyder, og deretter kjøre den vedlagte Jupyter Notebooken. Jupyter Notebook er en teknologi som gir brukerne muligheten til å opprette interaktive dokumenter som kombinerer tekst og Python-kode (Jupyter.org, 2019), der Python-koden kan kjøres direkte i notebooken. Den vedlagte Jupyter Notebooken er spesifikt utformet for å eksportere en modell som er egnet for bruk i en mobilapplikasjon, og eksporterer den i TensorFlow Lite-formatet. TensorFlow Lite-formatet er optimalisert for å kjøre modeller på mobile enheter med begrenset ressurser, og sikrer dermed at den trente modellen kan brukes effektivt i mobilapplikasjonen.

4.3.5 Eksportering av modell

Modellen eksporteres i TensorFlow Lite-formatet for å kunne brukes i mobilapplikasjonen. For å oppdatere og trene en ny lydgjenkjenningsmodell kan lydene i datasettet oppdateres. Da trenger du bare å kjøre gjennom Jupyter Notebook-en for å trene og eksportere en ny modell. Denne effektive arbeidsflyten gjør det enkelt å iterere og forbedre modellen ved

behov, og sikrer at den oppdaterte modellen kan implementeres enkelt i mobilapplikasjonen.

4.4 Arkitektur

4.4.1 Autentisering

For å få tilgang til hovedfunksjonaliteten til applikasjonen må brukeren være innlogget. Om brukeren ikke allerede har en konto må brukeren registrere seg med e-postadresse og passord. Registreringen blir håndtert i RegistrerActicity.java. Det er lagt inn et regulært uttrykk for å sjekke om det er en e-postadresse som blir gitt, og det er satt krav til lengde på passord, og at bekreft passord er identisk. Hvis kravene er fullført, vil e-postadresse og passord bli sendt videre til Firebase. Ved hjelp av Firebase sin createUserWithEmailAndPassword() autentiserings metode, blir en ny bruker opprettet. Det blir også lagt til e-postadresse og device token i Firebase sin Realtime Database. Deretter blir brukeren sendt videre til dashbordet.

Dersom brukeren allerede er registrert, kan brukeren bruke e-postadresse og passord for å logge inn. Dette blir gjort i MainActivity.java ved hjelp av Firebase sin signInWithEmailAndPassword() autentisering metode. Hvis både e-postadresse og passord er godkjent, vil brukeren bli sendt videre til dashbordet, hvis ikke kommer en feilmelding opp på skjermen.

Om brukeren har glemt passordet sitt kan det bli tilbakestilt. I GlemtPassordActivity.java blir dette gjort med Firebase metoden sendPasswordResetEmail(). Metoden tar en e-postadresse som parameter, og sender en epost der brukeren kan trykke på en lenke for å tilbakestille sitt passord.

4.4.2 Legg til kontakter

For å muliggjøre varsling av kontakter må brukeren legge til en kontakt som en "venn" i applikasjonen. Dette kan enkelt gjøres via kontaktaktiviteten. Brukeren har muligheten til å skrive inn e-postadressen til en nabo eller venn, som deretter blir lagt til i en kontaktliste som er lagret i brukerens database.

Ved å legge til kontakter på denne måten, oppnås en enkel og brukervennlig måte for brukeren å administrere og oppdatere sin liste over venner og naboer som skal varsles i tilfelle en nødsituasjon. Dette gir også fleksibilitet, slik at brukeren kan legge til eller fjerne kontakter etter behov, og sikrer at varslingen når de rette personene når det er nødvendig.

4.4.3 Geolokasjon

For å tilrettelegge for strømsparing og redusert ressursbruk, gis brukeren muligheten til å lagre sitt hjemområde i applikasjonen ved hjelp av en geolokasjonstjeneste. Dette gjøres

ved at brukeren trykker på en knapp når han/hun er hjemme. Når hjemområdet er lagret, kan applikasjonen identifisere når brukeren befinner seg innenfor dette området.

Geolokasjon ble implementert ved hjelp av å hente ut lengdegradene og breddegradene ved hjelp av SharedPreferences klassen. Ved å benytte denne tjenesten kan applikasjonen nøyaktig og pålitelig fastslå brukerens nåværende posisjon. For å sjekke om brukeren er hjemme kalles brukerErHjemme() metoden, som tar nåværende posisjon og avstand som parameter. Det ble valgt å bruke 100 meter som avstand, for å ta forbehold om større bygg, og unøyaktighet i geolokasjon dataen.

Når brukeren befinner seg innenfor sitt hjemområde, aktiveres lyttemodus for lydgjenkjenning av røykvarsler. Dette betyr at applikasjonen vil begynne å lytte etter lyder kun når brukeren er hjemme, noe som resulterer i strømsparing og optimal bruk av enhetens ressurser. Når brukeren forlater hjemområdet, vil applikasjonen automatisk deaktivere lyttemodus og spare ytterligere strøm ved å ikke gjennomføre unødvendig lydovervåkning.

4.4.4 Teste og aktivere lydgjenkjenning

For å sikre at lydgjenkjenning kun aktiveres når det er en reell alarm fra røykvarsleren, har en test-funksjonalitet blitt implementert. Dette gjør det mulig for brukeren å teste lydgjenkjenningen uten å varsle kontakter. Testingen gir brukeren tilgang til den faktiske lydgjenkjenningsfunksjonaliteten, som i utgangspunktet kun kjører i bakgrunnen og overvåker potensielle alarmer fra røykvarslere.

Testen kan bekrefte at brukerens mikrofon fungerer, og at lyden fra røykvarsleren blir korrekt gjenkjent av maskinlæringsmodellen. Dette bidrar til å sikre at lydgjenkjenningen er pålitelig og i stand til å oppdage alarmer fra røykvarslere nøyaktig.

Når en bruker har testet lydgjenkjenningen og bekreftet at alt fungerer som forventet, og de har en kompatibel røykvarsler og kontakter lagt til i applikasjonen, kan brukeren aktivere lydgjenkjenningen. Gjennom testing-funksjonaliteten og aktiveringen av lydgjenkjenning øker sannsynligheten for at varslingssystemet kun reagerer på ekte alarmer fra røykvarslere, og reduserer risikoen for falske alarmer.

4.4.5 Lydgjenkjenning foreground service

Foreground service er en type tjeneste i Android utvikling som tillater applikasjoner å kjøre i bakgrunnen, selv når brukeren ikke er aktivt engasjert med applikasjonen. (Android Developers, n.d.) Denne funksjonaliteten er spesielt nyttig når applikasjonen utfører viktige oppgaver som skal fortsette å kjøre jevnt, som lydgjenkjenning og varsling i vårt tilfelle.

Ved oppstart av foreground servicen vil først onCreate() metoden bli kalt. Her blir lydklassifiseringmodellen lastet inn og initialisert, og det blir hentet referanser til databasen. Etter dette vil applikasjonen kontinuerlig lytte etter lydsignaler i bakgrunnen

ved hjelp av onStartCommand() metoden. Her blir lydopptaket kontinuerlig analysert ved hjelp av den trente lydklassifiseringmodellen. Dersom modellen gjenkjenner en lyd som samsvarer med en røykvarsler, vil varslingssystemet aktiveres. Dette innebærer å sende en varsling til brukeren og til alle brukerens kontakter om at en røykvarsler er blitt oppdaget. Ved å implementere en terskelverdi på 0.7 sikrer det at varselet utløses kun når lyden av en røykvarsler er pålitelig gjenkjent.

Gjennom bruk av foreground service kan applikasjonen vår fortsette å overvåke lyder i bakgrunnen mens den opprettholder en synlig tilstedeværelse for brukeren. Dette sikrer pålitelig og kontinuerlig lydgjenkjenning for å oppdage potensielle alarmer fra røykvarslere og varsle brukeren i sanntid.

4.4.6 Varsling

Når en bruker er innlogget og systemet er aktivt og oppdager lyden fra en røykvarsler, aktiveres varslingssystemet i mobilapplikasjonen for å informere brukerens pre-definerte kontakter. Den sender en push-varsel til kontaktene. Denne push-varselen viser hvilken bruker sin røykvarsler som er oppdaget, og tidspunktet varselen blir mottatt er automatisk notert. Dette gir kontaktene umiddelbar kunnskap om situasjonen og muligheten til å iverksette passende handlinger.

For å sikre at varslingen når fram til kontaktene, er det viktig å opprettholde en stabil og pålitelig kommunikasjon mellom applikasjonen og skytjenesten som håndterer push-varsler. Firebase har blitt valgt som skytjeneste-leverandør, da den tilbyr robuste funksjoner for push-varsler og en pålitelig infrastruktur for å håndtere varslinger på tvers av ulike enheter.

Varslingssystemet i applikasjonen er designet for å sikre rask og pålitelig respons på alarm fra røykvarslere. Ved å utnytte mobilens push-varsler og interne varslinger, kan det sikres at brukeren og deres definerte kontakter blir raskt informert om potensielle brannsituasjoner.

5 RESULTATER

Kapittelet presenterer resultatene av evalueringen. Evalueringen inkluderer å vurdere brukergrensesnittet gjennom en spørreundersøkelse og å evaluere funksjonaliteten og strømforbruket til applikasjonen gjennom akseptansetesting. Kapittelet diskuterer også evalueringsresultatene, prosjektresultatet og prosjektgjennomføring.

5.1 Evalueringsmetode

5.1.1 Spørreundersøkelse

Det ble laget en spørreundersøkelse som metode for å samle inn informasjon om tilbakemelding på brukergrensesnittet. Google-forms ble tatt i bruk som et plattform for å levere et spørreskjema. Spørreskjemaet ber først om informasjon om respondentens kjønn, alder, og om de har opplevd brann i hjemmet tidligere. Det ber også om tilbakemelding på hvor nyttig respondentene mener en slik mobilapplikasjon ville være, hvilke funksjoner de forventer å se i appen, og hvor komfortable de ville være med å bruke den. Deretter inkluderer spørreskjemaet spørsmål om brukervennlighet, navigasjon, synlighet og forståelse av knappene, eventuelle manglende funksjoner, designets samsvar med formålet, vurdering av appens evne til å hjelpe med tidlig evakuering, og om respondentene ville anbefale appen til andre. Det avsluttes med en åpen mulighet for andre kommentarer eller tilbakemeldinger om mobilapplikasjonen.

5.1.2 Akseptansetest

Under akseptansetesting ble det gjennomført en evaluering av lydgjenkjenningens funksjonalitet og applikasjonens påvirkning på strømforbruket. Formålet med disse testene var å sikre at lydgjenkjenningen fungerte som forventet og at applikasjonen kunne opprettholde en optimal balanse mellom ytelse og strømeffektivitet.

For å teste lydgjenkjenningen ble det brukt et utvalg av lydopptak som inkluderte både røykvarslerlyder og andre lignende lyder, som mikrobølgeovner og vekkerklokker. Modellen ble testet på disse lydene og nøyaktigheten og påliteligheten til gjenkjenningsprosessen ble evaluert. Det ble også utført praktiske tester der røykvarslere ble manuelt aktivert for å teste hvordan lydgjenkjenningen presterte under reelle forhold.

For å vurdere strømforbruket til applikasjonen, ble det gjennomført strømmålinger på ulike enheter. Strømforbruket ble målt både under aktiv bruk og i bakgrunnen for å få en helhetlig forståelse av applikasjonens energiforbruk. Dette omfattet å registrere strømforbruket under ulike scenarier, for eksempel når lydgjenkjenningen var aktivert og inaktiv, og når brukeren var både hjemme eller utenfor hjemmet.

5.2 Evalueringsresultat

Spørreundersøkelsen ble primært distribuert til en eldre målgruppe. Blant respondentene utgjorde aldersgruppen 45-54 år flertallet med en andel på 45.5%, mens etterfulgt av aldersgruppen 55-64 år med en andel på 27.3%. Dette støtter vårt fokus om at eldre utgjør en gruppe med høy dødsrisiko i brannsituasjoner. Videre viser undersøkelsen at 54.4% av respondentene har opplevd brann i eget hjem, dette bekrefter at eldre er sårbare for brann.

Majoriteten av respondentene er enig i å ha en mobilapp som oppdager røykvarsler og sender varsling til forhåndsdefinerte kontakter er nyttig, men det er varierende på hvor komfortabel de e å ta i bruk av en slik app.

Videre ble grensesnittet av mobilappen testet hvor det viste seg at 63.6% var enig med at det var lett å navigere seg frem på påloggingssiden og samtidig med 81.8% med at knappene var forståelig på hva de gjorde. Majoriteten anses funksjonene i mobilappen som nødvendige å ha. Til slutt er alle enig i at appen samsvarer med formålet til hva den gjør.

Resultatene fra akseptanstestene viste at lydgjenkjenningen fungerte og oppfylte oppdragsgivers forventninger. En fullstendig applikasjon som var klar til å rulles ut var ikke forventet, oppdraget handlet mest om å se om en slik applikasjon var mulig å lage. Maskinlæringsmodellen er i stand til å identifisere røykvarslerlyder og skille dem fra andre lignende lyder. Når det gjelder strømforbruket, viste testene at applikasjonen brukte cirka 5% av batteriets kapasitet per time på de forskjellige enhetene som ble testet. Dette ble regnet som litt mer en ønskelig fra oppdragsgiver.

5.3 Prosjektresultat

Resultatet av prosjektet er en mobilapplikasjon som kjører på Android smarttelefoner. Applikasjonen tar i bruk mikrofonen og en dyplæring modell for å gjenkjenne lyden av en røykvarsler. For å kjøre i bakgrunnen ble en foreground service tatt i bruk, og for å autentisere brukerene, lagre bruker data, og sende meldinger mellom brukerene ble firebase brukt.

Fra visjonsdokumentet (vedlegg 2) er flere krav definert, her ble 9/17 av de funksjonelle kravene tilfredstilt. Her er kravene som applikasjonen tilfredstiller:

- Brukeren må kunne registrere seg med navn, e-postadresse og passord
- Brukeren må kunne logge seg inn med e-postadresse og passord
- Brukeren må kunne logge seg ut
- Brukeren skal kunne legge inn nye kontakter
- Applikasjonen skal ta i bruk mikrofonen på mobilen for å lytte etter røykvarsler
- Brukeren skal kunne velge om applikasjonen får lov å lytte eller ikke
- Brukeren skal kunne se om applikasjonen lytter eller ikke
- Applikasjonen skal gjenkjenne lyden av en røykvarsler
- Applikasjonen skal varsle predefinerte kontakter dersom røykvarsleren går av

Kravene som mangler er:

- Brukeren skal kunne velge om han/hun ønsker funksjonaliteten av å varsle, eller å bli varslet, eller evt. Begge
- Brukeren må godkjenne våre rettningslinjer for å kunne bruke resten av applikasjonen
- Applikasjonen skal være kompatibel med både IOS & Android
- Brukeren skal kunne se reglene for personvern i applikasjonen
- Brukeren skal kunne fjerne kontakter slik at de ikke lenger blir varslet
- Brukeren må kunne se hvilken kontakter som er registrert
- Brukeren skal kunne se hvem som har varslene på eller av
- Brukeren skal kunne velge å slå av varslene

5.4 Prosjektgjennomføring

Prosjektet ble gjennomført i perioden januar 2023 til mai 2023. Til å begynne med besto prosjektet hovedsakelig av møter, og å diskutere hvordan en eventuell løsning skal se ut. Arbeidsoppgaver ble delt inn etter hver enkelt medlems styrker og svakheter. Rosa tok hovedansvaret for designet av fronteneden, Magnus tok hovedansvaret for maskinlærings modellen, og Torstein tok hovedansvaret for utviklingen av mobilapplikasjonen. Utvikling av applikasjonen startet med et kryssplattform rammeverk. Senere ble det oppdaget under utviklingen at dette var mer arbeid enn forventet for å oppfylle en eventuell løsning. Utviklingen av applikasjonen startet på nytt med individuelle applikasjoner for hver plattform. Dette førte til at planen ble satt tilbake flere uker. Etter et par uker til med utvikling hadde progresjonen kommet betydelig mye lenger på Android applikasjonen i forhold til iOS applikasjonen. Deretter ble det bestemt i samarbeid med veileder at fokuset skulle bli å utvikle Android applikasjonen ferdig.

6 DISKUSJON

Sluttproduktet endte opp med å være en Android mobilapplikasjon. Applikasjonen oppfyller mesteparten av de funksjonelle kravene nevnt i visjonsdokumentet (Vedlegg 2). Brukere kan registrere seg og logge inn med e-postadresse og passord, og kan tilbakestille passordet dersom de har glemt det. Innloggede brukere kan legge til kontakter, og teste lydgjenkjenningen opp mot sin egen røykvarsler. Brukeren kan lagre sin nåværende posisjon som hjemme. Når lydgjenkjenningen er aktiv og det oppdages en røykvarsler, så sendes varsler til alle brukerens kontakter.

Det er flere mangler å merke seg i forhold til kravene spesifisert i visjonsdokumentet (Vedlegg 2). Mesteparten av manglene oppsto som følge av tidspress, da det var etterslep i forhold til planen da utviklingen måtte startes på nytt. Strømforbruket til mobilapplikasjonen ligger på rundt 5% batteri i timen. Dette er noe høyere enn ønsket grunnet at applikasjonen skal kjøre konstant i bakgrunnen. Det ble bare utviklet en mobilapplikasjon til Android, og ingen til iOS. Kontaktsystemet har flere mangler. Det er ingen mulighet for å sende forespørsel, slik at man kan godkjenne eller avslå forespørselen, kontakter blir lagt til direkte. Det er heller ikke mulig å slette kontakter, eller å se alle kontaktene som er lagt til.

Den eksisterende modellen, YAMNET, ble valgt å bruke som utgangspunkt for vår lydgjenkjnningsmodell. Gjennom transfer learning ble modellen tilpasset til å gjenkjenne røykvarslere, og den ble trent med et variert dataset. Det hadde vært mulig å utforske andre dyplæringsteknikker og modellarkitekturer for å forbedre ytelsen. Å utvide datasettet med flere lydvariasjoner og samle inn data fra flere kilder kunne også økt modellens generaliseringsevne. Grunnet uforutsette utfordringer ble ikke disse mulighetene utforsket/tiltakene gjort.

Utviklingen startet med å lage en React Native mobilapplikasjon. React Native ble vurdert som et godt frontend rammeverk og ga en god utvikleropplevelse. Med Expo synes oppdateringer i brukergrensesnittet fortløpende, uten behov for å bygge applikasjonen om igjen hver gang en endring ble gjort. Men etter komplikasjoner med å få lydgjenkjenningen koblet opp maskinlæringsmodellen ble det valgt å gå vekk fra denne løsningen.

Mens skiftet fra React Native til Android-utvikling gjorde at hele applikasjonen måtte utvikles på nytt, tilbød det en mer praktisk løsning for å møte det spesifikke kravet om kontinuerlig lydgjenkjenning av røykvarslere. Beslutningen om å dreie ble drevet av behovet for finmasket kontroll og tilgang til plattformspesifikke funksjoner, som Android-utvikling ved bruk av Java kunne gi. Det er ikke umulig å oppnå lignende resultat med React Native, men det var betydelig mer overkommelig å lage applikasjonen til Android med Java.

Vår lydgjenkjenningsløsning for røykvarslere kan sammenlignes med andre alternativer på markedet. Betalte løsninger tilbyr ofte avanserte funksjoner og omfattende støtte, men kan være kostbare og ikke nødvendigvis tilpasset spesifikke behov. iPhone shortcuts gir brukerne en innebygd løsning for lydgjenkjenning, og noe mulighet for å opprette tilpassede snarveier som løser varsling av kontakter. Smart-hjemløsninger, som stemmeassistenter, kan også integreres for lydgjenkjenning av røykvarslere. Dette kan være del av et større økosystem som gir automatisering og tilkobling til andre enheter i hjemmet. Ved vurdering av løsninger er det viktig å ta hensyn til kostnad, pålitelighet, brukervennlighet og tilpasningsevne for å sikre at den valgte løsningen passer best mulig til de spesifikke kravene og preferansene.

Hvis hele prosjektet skulle blitt gjort på nytt, hadde en del ting blitt gjort annerledes. Forarbeidet hadde blitt enda mer detaljert. Istedenfor å anta at rammeverkene som hørte til React-Native passet bra til lydgjenkjenningen, kunne en liten prototype som testet at lydgjenkjenningen funket før starten på selve utviklingen, blitt lagd.

Alt i alt var tilbakemeldingen fra oppdragsgiveren positiv. Den viktigste funksjonaliteten er på plass, og det viser at en slik applikasjon kan bli laget for å oppdage og varsle om brann automatisk.

7 KONKLUSJON OG VIDERE ARBEID

7.1 Konklusjon

I kapittel 1 definerte vi 3 forskningsspørsmål til prosjektet:

1. «Er applikasjonen brukervennlig nok til at den er effektiv?»
2. «Er applikasjonen god nok til å gjenkjenne lyden av røykvarslere til at den kan fungere som et verktøy til tidlig varsling av predefinerte kontakter i situasjoner der umiddelbar assistanse er avgjørende?»
3. "Er applikasjonen energieffektiv nok til å ikke tømme batteriet urimelig raskt?"

Spørreundersøkelsen har gitt oss et innblikk i hvordan applikasjonen blir oppfattet. Applikasjonen fremstår som lett å navigere, men noe uklarhet for hva knapper som "Geolokasjon" betyr. Svaret på forskningsspørsmålet «Er applikasjonen brukervennlig nok til at den er effektiv?», blir for det meste ja. Ut ifra dataen fra spørreundersøkelsen ville folk flest ikke ha et problem å sette opp mobilapplikasjonen sammenlignet med andre apper. Når det kommer til eldre, og gjerne mindre teknisk erfarne brukere, ville de antageligvis slitt med å sette opp applikasjonen alene. Men hvis de får hjelp fra en mer teknisk erfaren venn er alt de trenger å gjøre videre å trykke på start/stop-knappen.

Svaret på «Er applikasjonen god nok til å gjenkjenne lyden av røykvarslere til at den kan fungere som et verktøy til tidlig varsling av predefinerte kontakter i situasjoner der umiddelbar assistanse er avgjørende?» er ja. Røykvarslere har en distinkt lyd, og er lett å kjenne igjen for oss mennesker, det samme gjelder for maskinlæringsmodellen som ble utviklet. Applikasjonen plukker opp når en røykvarsler går av, og blir ikke lurt av andre lyder som mikrobølgeovner eller sirener.

Svaret på forskningsspørsmålet "Er applikasjonen energieffektiv nok til å ikke tømme batteriet urimelig raskt?" blir litt mer nyansert. 5% bruk av batteriet i timen på mobiltelefonene som ble testet på er nok ikke urimelig raskt, men regnes som i øverste lag man ville vært komfortabelt med å ha på kontinuerlig. Med dette forbruket ville det tatt 20 timer å tømme telefonen for batteri uten å bruke telefonen til noe annet. Konklusjonen er at nei, det er ikke urimelig raskt, men det er heller ikke optimalt.

Konklusjonen blir dermed at prosjektet ble fullført med et tilfredsstillende resultat.

7.2 Videre arbeid

Som nevnt i kapittel 5.3 ble ikke alle de funksjonelle kravene oppfylt, første prioritet ville vært å oppfylle de viktigste funksjonelle kravene som mangler, dette inkluderer:

- Forberede kontakt systemet. Nå skriver bare en bruker inn e-postadressen til ønsket kontakt, av flere grunner er det ønsket et "venneforespørsel" system, der man kan godta eller avslå en brukers forespørsel om å bli venner. Det må også bli mulig å slette kontakter.

- Legge til todelt bruker. Slik det er nå har alle brukere all funksjonaliteten, for å både sende og motta varsler. Det er ønsket at det er mulig at en bruker kan velge mellom å bare kunne varsle, eller bare kunne motta varsler, eller begge.
- Lage en iOS versjon av applikasjonen. Under utviklingen kom vi bak skjemaet som nevnt i kapittel 3.6, gikk vekk fra utviklingen av iOS applikasjonen og holdt fokuset vårt på Android. Men iPhone har en stor del av smarttelefonmarkedet, dermed er en applikasjon som kjører på iOS viktig.
- Sette opp personvernerklæring for brukeren som må bli godkjent under registrering. Personvern er en viktig ting å forstå med applikasjonen, mikrofonen blir tatt i bruk, og lyden blir analysert, det er viktig at brukeren skjønner at lyden ikke blir sendt videre og brukt noen andre steder.
- Liste opp alle kontakter. Slik det er nå får ikke brukeren sett hvilke kontakter som er lagt til. Det hadde også vært greit å kunne se hvilke kontakter som er aktive, og har applikasjonen på.

En ide som ble nevnt av oppdragsgiver under utviklingen var å lage en liten IOT enhet, mer lignende andre fysiske løsninger, som de nevnt tidligere fra Vestfold Audio og Verisure. Ideen var å bruke maskinlæringsmodellen for å utvikle en fysisk løsning som kan kobles opp mot telefonen og varsle selv om brukeren ikke er hjemme. Vi diskuterte hvordan en slik løsning kan bli prototypet ved hjelp av en Arduino med en mikrofon og maskinlæringsmodellen, og koble denne opp mot internett, og deretter koble den opp mot mobilapplikasjonen, eller integrere det som en smart-hjem løsning. Varslingen kommer da til å fungere selv om brukeren ikke er hjemme.

8 REFERANSER

DSB, Direktoratet for samfunnssikkerhet og beredskap. (2023) Omkomne i brann.
Tilgjengelig fra: https://www.dsb.no/menyartikler/statistikk/omkomne-i-brann/
[Hentet 24 Feb. 2023].

Verisure. (2019). Verisure App - styr hjemmet med mobilen.
Tilgjengelig fra: https://www.verisure.no/tjenester/verisure-app
[Hentet 24 Feb. 2023].

inVerita (2020). Flutter vs Native vs React-Native: Examining performance.
Tilgjengelig fra: https://medium.com/swlh/flutter-vs-native-vs-react-native-examining-performance-31338f081980
[Hentet 25 Feb. 2023].

SSB (2021). Internett og mobiltelefon.
Tilgjengelig fra: https://www.ssb.no/teknologi-og-innovasjon/faktaside/internett-og-mobil
[Hentet 25 Feb. 2023].

Rake, E.L., Jensen, K., Rimstad, N.Ø. and Hoel, O., (SNL) (2022). brann.
Tilgjengelig fra: https://snl.no/brann
[Hentet 26 Feb. 2023]

HVL (2023). Building design for At-risk groups (BUILDER)
Tilgjengelig fra: https://www.hvl.no/prosjekt/2496919
[Hentet 26 Feb. 2023]

SSB. (2021) Bruk av IKT i husholdningene.
Tilgjengelig fra: https://www.ssb.no/statbank/table/12349/chartViewColumn/
[Hentet 26 Feb. 2023]

Ramsøy, Carina (VISMA) (2022). En kort introduksjon til Scrum.
Tilgjengelig fra: https://www.visma.no/blogg/en-kort-introduksjon-til-scrum/
[Hentet 26 Feb. 2023]

SSB (2019). 1 av 3 eldre bor alene
Tilgjengelig fra: https://www.ssb.no/befolkning/artikler-og-publikasjoner/1-av-3-eldre-bor-alene
[Hentet 08 Mar. 2023]

Vestfold Audio. (2023). VEA Fire Safety.
Tilgjengelig fra: https://vestfoldaudio.no/vea-fire-safety/
[Hentet 9 Mar. 2023].

Apple Support. (2023). Gjenkjenn lyder med iPhone.
Tilgjengelig Fra: https://support.apple.com/no-no/guide/iphone/iphf2dc33312/ios
[Hentet 9 Mar. 2023].

Interaction design foundation. Prototyping
Tilgjengelig fra: https://www.interaction-design.org/literature/topics/prototyping
[Hentet 24 april. 2023].

Smoke alarm efficiency Waking sleeping occupants. (n.d.).
Tilgjengelig fra: https://risefr.no/media/publikasjoner/upload/2019/msb1332-smoke-alarm-efficiency.pdf
[Hentet 20 mai 2023].

Salamon, J. and Bello, J.P. (2017). Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification. IEEE Signal Processing Letters, [online] 24(3), pp.279–283.
https://arxiv.org/abs/1608.04363

Kotlin Help. (n.d.). Native and cross-platform app development: how to choose? | Kotlin. [online]
Tilgjengelig fra: https://kotlinlang.org/docs/native-and-cross-platform.html#what-is-native-mobile-app-development.

Kotlin Help. (n.d.). What is cross-platform mobile development? | Kotlin. [online]
Tilgjengelig fra: https://kotlinlang.org/docs/cross-platform-mobile-development.html#is-cross-platform-mobile-development-right-for-you.

www.merriam-webster.com. (n.d.). Definition of PEER-TO-PEER. [online]
Tilgjengelig fra: https://www.merriam-webster.com/dictionary/peer-to-peer.

smpp.org. (n.d.). SMPP - Short Message Peer-to-Peer Protocol. [online]
Tilgjengelig fra: https://smpp.org

Microsoft (n.d.). What Is Cloud Computing? A Beginner's Guide | Microsoft Azure. [online] azure.microsoft.com.

Tilgjengelig fra: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-cloud-computing

GitHub. (2023). The fastai book. [online]
Tilgjengelig fra: https://github.com/fastai/fastbook/blob/master/01_intro.ipynb
[Hentet 20 mai 2023].

Apple Support. (n.d.). Få varslinger fra røykvarslere og karbonmonoksidmålere. [online]
Tilgjengelig fra: https://support.apple.com/no-no/guide/homepod/apd1f1921722/homepod
[Hentet 20 mai 2023].

Jupyter.org. (2019). Project Jupyter. [online]
Tilgjengelig fra: https://jupyter.org

Android Developers. (n.d.). Foreground services. [online]
Tilgjengelig fra: https://developer.android.com/guide/components/foreground-services

Howard, J. and Gugger, S. (2020). Deep Learning for Coders with fastai and PyTorch. 'O'Reilly Media, Inc.'

UX Design Institute (2022). What is wireframing? A complete guide.
Tilgjengelig fra: https://www.uxdesigninstitute.com/blog/what-is-wireframing/#What-is-wireframing?
[Hentet 22 mai. 2023]

9 VEDLEGG

Følgende vedlegg er tilgjengelig som eksterne dokumenter:

Vedlegg 1 Prosjekthåndbok

Vedlegg 2 Visjonsdokument

Vedlegg 3 Kravdokumentasjon

Vedlegg 4 Systemdokumentasjon

Vedlegg 5 Spørreundersøkelse