Høgskulen på Vestlandet

Fakultet for ingeniør- og naturvitskap

Institutt for datateknologi, elektroteknologi og realfag

# BACHELOROPPGAVE

## Maskinell Transkribering av Snapchat Skjermopptak

## Automatic Transcription of Snapchat Screen Recordings

**Thea Christianslund**
**Anne Furubotten Drottning**
**Vilde Martine Færøy**

Bachelor, Dataingeniør
Fakultet for ingeniør- og naturvitenskap
Institutt for datateknologi, elektroteknologi og realfag
Remy Andre Monsen
Innleveringsdato: 23.05.2022

Høgskulen på Vestlandet

Fakultet for ingeniør- og naturvitskap

Institutt for datateknologi, elektroteknologi og realfag

Høgskulen på Vestlandet

Fakultet for ingeniør- og naturvitskap

Institutt for datateknologi, elektroteknologi og realfag

# TITTELSIDE FOR HOVEDPROSJEKT

| | |
|---|---|
| *Rapportens tittel:*<br>Maskinell Transkribering av Snapchat Skjermopptak | *Dato:*<br>*23.05.2022* |
| *Forfatter(e):*<br>Vilde Martine Færøy, Anne Drottning og Thea Christianslund | *Antall sider u/vedlegg: 37*<br>*Antall sider vedlegg: 3* |
| *Studieretning:*<br>Dataingeniør | *Antall disketter/CD-er:* 0 |
| *Kontaktperson ved studieretning:*<br>Remy Andre Monsen | *Gradering:* Ingen |
| *Merknader:*<br>Ingen | |

| | |
|---|---|
| *Oppdragsgiver:*<br>Politiet (DPA) | *Oppdragsgivers referanse:* |
| *Oppdragsgivers kontaktperson:*<br>Eirik Reglund Thorsen | *Telefon:* 924 51 079 |

*Sammendrag:*

Bachelorprosjektet går ut på å utvikle programvare for automatisk transkribering av tekst fra skjermopptak av Snapchat samtaler. Programvaren skal brukes av etterforskere hos Vest politidistrikt for å redusere tid brukt på manuell transkribering, og for å raskere finne skjermbilder som inneholder et søkeord. Løsningen er programvare som kan kjøres på Windows og Unix maskiner gjennom et brukervennlig brukergrensesnitt.

*Stikkord:*

| | | |
|---|---|---|
| Python | Windows | Tesseract |
| Programvareutvikling | Unix | OpenCV |
| OCR | Søkefunksjonalitet | Bildeprosessering |
| Bildehåndtering | | |

Høgskulen på Vestlandet

Fakultet for ingeniør- og naturvitskap

Institutt for datateknologi, elektroteknologi og realfag

# FORORD

Rapporten dokumenterer arbeidet gjort i bachelorprosjektet "Maskinell Transkribering av Snapchat Skjermopptak". Prosjektet er gjennomført av Vilde Martine Færøy, Thea Christianslund og Anne Drottning.

Vi vil takke oppdragsgiver for et spennende og utfordrende prosjekt hvor vi har fått muligheten til å lære mer innen programmering og nye teknologier. Takk til Eirik Reglund Thorsen og Kjell Ove Riple for veiledning og innblikk i hvordan programvarer hjelper hverdagen hos politiet. Også takk til Vilde Margrethe Jacobsen og Petter Osland Aarberg for gode tilbakemeldinger på brukertestingen.

Vi vil også takke Remy Andre Monsen for hjelp underveis i prosjektet, gode tilbakemeldinger og hjelpsom idémyldring når vi har stått fast.

En siste takk går til Monica Færøy for hjelp til å lese over rapporten før levering.

# INNHOLDSFORTEGNELSE

# 1 Innledning

## 1.1 Kontekst

I Norge øker bruken av sosiale medier blant befolkningen. I 2018 ble det rapportert at 80% av nordmenn mellom 16 og 79 år bruker sosiale medier, hvorav ca. 60% bruker sosiale medier nesten hver dag. Daglig bruk av sosiale medier er høyest i aldersgruppen 16-24 år (Røgeberg, 2018). De tre største sosiale medietjenestene brukt daglig i Norge av mennesker over 18 år er Facebook hvor 65% bruker den daglig, og Snapchat og Messenger hvor 43% bruker dem daglig (Ipsos, 2022). Snapchat er en tjeneste hvor det er mulig å sende bilder og chat-meldinger til enkeltpersoner og grupper. Bildene er synlig når de åpnes, og chat-samtaler slettes automatisk etter 24 timer, med mindre de lagres. Snapchat har blant annet blitt brukt til salg av narkotika (Braaten & Fossheim, 2021) og salg av seksuelle tjenester (Eliassen, 2021). Etterforskere i politiet kan bruke informasjon fra sosiale medier som bevis videre i etterforskningen, og videre til rettssak.

## 1.2 Motivasjon

Politiet har ansvar for å forebygge, avdekke og stanse kriminalitet i samfunnet og i tråd med den digitale utviklingen beveger kriminaliteten seg fra fysiske åsted til digitale rom (Stranden, 2021). Digitale rom kan også benyttes til planlegging av kriminelle handlinger og fungere som kommunikasjonsmedium mellom de kriminelle.

Politiet har et eget avsnitt Digitalt Politiarbeid (DPA) som bistår i oppklaringen av enkelte saker under etterforskning. Som en del av bevissikring filmer DPA en gjennomgang av Snapchat kommunikasjon på mobiltelefon. I saker som fremlegges for retten må innholdet i videoen transkriberes. I dag utføres transkriberingen manuelt av etterforskere i avsnittet, og er en tidkrevende oppgave. DPA sine bidrag i straffesaker er større enn tidligere, på grunn av den digitale utviklingen. Effektivitet i digitalt politiarbeid påvirker saksbehandlingstid og er spesielt viktig i tidssensitive saker.

Ved å utvikle en programvare som kan automatisere transkribering kan bevissikring av Snapchat kommunikasjon effektiviseres. Automatisk transkribering fører til kortere saksbehandlingstid og frigjøring av ressurser til annet arbeid.

## 1.3 Prosjekteier

Prosjektgruppen vil være den offisielle eieren av det endelige produktet, med forbehold om at oppdragsgiver vil få tilgang.

Oppdragsgiver er avdeling for Digitalt Politiarbeid (DPA) i Vest Politidistrikt. DPA bistår i oppklaringen av de mest alvorlige sakene som etterforskes i Vest Politidistriktet. Grunnet den digitale utviklingen har DPA sine bidrag til straffesaker økt.

## 1.4 Problembeskrivelse og mål

Ut fra kontekst og motivasjon ble det bestemt en problemstilling: *Er det mulig å automatisere transkribering fra skjermopptak av chat-samtaler for å spare tid og ressurser sammenlignet med manuell transkribering?*

Den transkriberte teksten brukes videre i rettssaker og innen etterforskning, dermed er det viktig at transkriberingen blir korrekt. I samtaler på sosiale medier er det vanlig å bruke slang, forkortelser og emojier som bidrag til innholdet. Korrekt transkribering krever at det er tatt hensyn til all informasjonen sendt mellom brukere, hvem som har sendt de ulike beskjedene og i hvilken rekkefølge utvekslingen har forløpt.

Det endelige målet for å løse problemstillingen er å utvikle en fungerende programvare med høy korrekthet på den transkriberte teksten, representert på et format som fremstiller innhold og forløp til samtalen.

For å oppnå dette endelige målet er det satt fire delmål som vil hjelpe for å løse det endelige målet. Delmål 1 er å utvikle et program hvor skjermopptaket blir delt opp til skjermbilder som representerer den fullstendige samtalen, og at teksten fra hvert skjermbilde transkriberes. Delmål 2 er å unngå duplisert tekst hvor flere skjermbilder viser samme del av samtalen, slik at resultatet er uten unødvendige gjentakelser. Delmål 3 er at programvaren skal kunne håndtere ulike alfabet, emojier og forstå forkortelser, og transkribere disse korrekt. Det siste delmålet, delmål 4, er at den transkriberte teksten skal formateres på en oversiktlig måte basert på oppdragsgiver sine krav. Det er to relevante forskningsspørsmål til problemstillingen:

1. *"Er det mulig å lage en applikasjon som korrekt transkriberer tekst fra en løpende video?"*

2. *“Er det mulig å hente ut bilder fra en video-fil som kun presenterer unikt innhold?”* Skjermopptak av Snapchat kommunikasjonen inneholder unødvendige bilder med duplisert innhold som må fjernes før tekstgjenkjenning prosessen.

## 1.5 Oppbygging av rapporten

Videre i rapporten gis det en detaljert beskrivelse av prosjektet. Kapittel 2 spesifiserer krav og avgrensninger i prosjektet i tillegg til nødvendige ressurser. I Kapittel 3 er det fokus på ulike idéer for hvordan prosjektet skal løses og en beskrivelse av planlagt utviklingsmetode og prosjektplan. Kapittel 4 beskriver løsningen til prosjektet i detalj hvor den er delt inn i tre kategorier; video og mappe behandling, transkribering og eksportering og GUI. Resultatene og evalueringen av prosjektet er beskrevet i Kapittel 5. Diskusjon om resultater i forhold til forventinger til prosjektet finnes i Kapittel 6. Til slutt er konklusjonen og forslag for videre arbeid i Kapittel 7.

# 2 Prosjektbeskrivelse

## 2.1 Praktisk bakgrunn

### 2.1.1 Tidligere arbeid

Per dags dato er det ingen programvare som tilbyr transkribering av tekst direkte fra video i sin helhet og som oppfyller kravene satt av oppdragsgiver. Hvis det undersøkes programvarer som transkriberer videoer er disse ofte relatert til transkribering av lyd fra video. Her finnes det mange ulike verktøy som ikke er relevant for problemstillingen i dette prosjektet. For å vurdere tidligere arbeid deles problemet inn i to arbeidsoppgaver hvor det er tilgjengelig eksisterende løsninger.

Den første oppgaven er å dele en video opp i bilder, hvor det finnes mange løsninger på nett og ved programvare. De fleste operativsystemer kommer med videoredigeringsverktøy allerede installert som kan brukes.

Den andre oppgaven er å transkribere tekst fra et bilde. For å løse denne oppgaven finnes det verktøy på nett som bruker maskinlæring for å gjenkjenne tekst i bilder.

### 2.1.2 Initielle krav

Det initielle kravet satt av oppdragsgiver er en programvare som transkriberer tekst fra video uten å miste informasjon. Dette generelle kravet består av mindre, men mer detaljerte krav. Transkriberingen skal håndtere symboler fra flere alfabeter, emojier og inkludere fotnoter som beskriver betydningen av vanlig brukte forkortelser. Teksten som skal transkriberes skal tolkes direkte, og ikke kjøres gjennom stavekontroll eller lignende siden det er den nøyaktige teksten på videoen som er av interesse.

Det er også nødvendig at programvaren skal være kjørbar uavhengig av hvilket operativsystem som brukes.

### 2.1.3 Initiell løsnings-idé

Problemet løses ved å utvikle programvare som deler opp skjermopptak til bilder som representerer den fullstendige informasjonen, transkriberer hvert bilde, formaterer teksten og eksporterer den fullstendige teksten.

For å gjøre programvaren brukervennlig uavhengig av teknisk bakgrunn vil det utvikles et interaktivt brukergrensesnitt.

## 2.2 Avgrensninger

Programvaren som utvikles i prosjektet vil ikke ta hensyn til animasjoner, bilder, emojier og videoer som er sendt i chat-samtalene. Dette er grunnet tidsbegrensninger og at det er andre krav stilt til det endelige produktet som har høyere prioritet.

For å gjennomføre transkribering med tekstgjenkjenning må det defineres hvilket alfabet som er brukt. I programvaren settes alfabetet til norsk slik at de norske bokstavene kan kjennes igjen, i tillegg til resten av bokstavene i det latinske alfabetet. Programvaren tar ikke hensyn til tegn og bokstaver fra andre alfabeter.

## 2.3 Ressurser

Utviklingen av programvaren krever ulike ressurser. For å starte utviklingen var det nødvendig med videoer av samme kvalitet og format som skal brukes i det endelige produktet. Politiet bruker Elgato Game Capture for å skaffe disse skjermvideoene. Elgato Game Capture er et verktøy som kobles til en mobiltelefon for å ta skjermopptak, i tillegg er det mulig å konfigurere innstillinger etter brukeren sine krav (Elgato, 2022).

Optisk tegngjenkjenning (OCR) er en metode som tolker tekst fra bilder og er bygget på maskinlæring teknologi. For Python eksisterer det et OCR bibliotek, basert på åpen kildekode, kalt Tesseract som inkluderer varierte metoder relatert til problemstillinger rundt OCR (PyPI, 2022). For å håndtere delene i programvaren som består av håndtering og prosessering av digitale medier brukes biblioteket OpenCV (Open Source Computer Vision Library) for Python. Programvaren er avhengig av flere biblioteker enn de to viktigste som er nevnt her, og mer utfyllende informasjon om disse er tilgjengelig i Vedlegg 9.1.

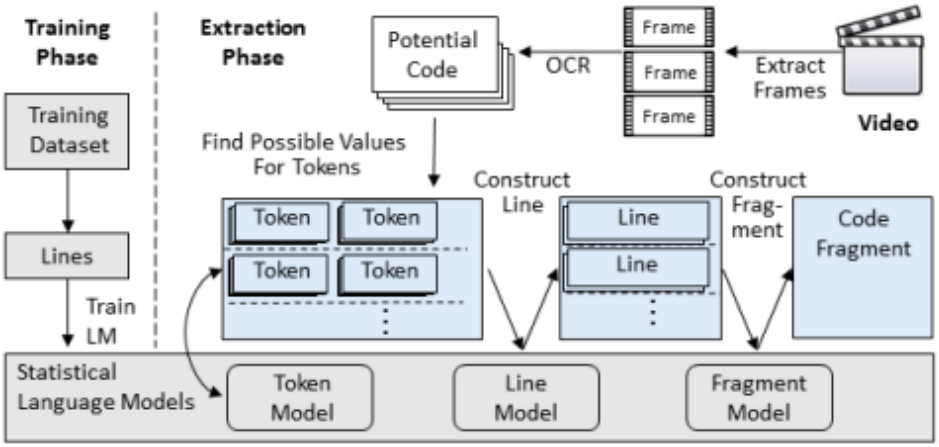
GitHub brukes for å dele kildekoden til programvaren for å opprettholde versjonskontroll og spore endringer utført av ulike prosjektmedlemmer.

Oppdragsgiver og prosjektveileder fungerer som ressurspersoner i løpet av prosjektet. Oppdragsgiver bidrar med spørsmål omhandlende brukerbehov og spesifisering av krav til produktet. Oppgaveveileder er tilgjengelig for tekniske spørsmål og veiledning gjennom eventuelle utfordringer.

I situasjoner hvor det ikke er mulig med fysiske møter brukes Zoom og Discord for å kommunisere digitalt.

## 2.4 Litteratur om problemstillingen

Det finnes ikke produkter som gjør nøyaktig det samme som skal utføres i dette prosjektet, men lignende problemstilling er beskrevet i (Yadid & Yahav, 2016) hvor det ønskes å hente ut kode fra en videoleksjon om koding. Det fokuseres på to hovedutfordringer; å slå sammen blokker som tilhører samme kode og korrigere resultat fra OCR for å oppnå høyere presisjon. Prosessen anvendt vises i Figur 1.




*Figur 1 Beskrivelse av implementasjon for tolkning av kode i videoer*

I implementasjonen hentes først bilder ut fra video ved uniform utvelging, deretter preprosesseres bildene og det brukes maskeringsteknikker for å minimere støy i bildene. Som tegngjenkjenningsverktøy brukes Tesseract OCR biblioteket. De velger å fokusere på leksjoner om Java og teksten hentet fra bildene filtreres derfor basert på om de inneholder minst et av Java symbolene: ; eller {}. Etter at teksten er filtrert brukes tre maskinlæringsmodeller: Token Model, Line Model og Fragment Model for å behandle teksten.

Løsningen til problemstillingen i dette prosjektet vil innebære bruk av OCR, denne teknologien tillater automatisk gjenkjenning av tegn gjennom en optisk mekanisme (Mithe, Indalkar, & Divekar, 2013). OCR er en teknologi som fungerer på lignende måte som menneskers evne til å lese, selv om OCR ikke kan konkurrere med menneskers leseevne da dens ytelse er direkte avhengig av kvaliteten til inndata. OCR er designet for å prosessere bilder som i hovedsak består av tekst og med begrenset ikke-tekst støy. Optisk tegngjenkjenning er en teknologi som lar deg konvertere filer som PDF, bilder og skannede dokumenter til redigerbare og søkbare data.

# 3 DESIGN AV PROSJEKTET

## 3.1 Forslag til løsning

Utviklingen av programvaren deles opp i tre deler, hvor hver av de kan ha ulike løsninger. Første del omhandler hvordan skjermopptak skal deles opp til skjermbilder som representerer den fullstendige samtalen. Den andre delen handler om hvordan teksten på hvert skjermbilde skal transkriberes, og den siste delen handler om hvordan den transkriberte teksten skal formateres. I de neste underkapitlene vil det presenteres ulike løsninger som har blitt vurdert i løpet av prosjektet for hver av de tre delene.

### 3.1.1 Alternative løsninger for skjermopptak til skjermbilder

#### 3.1.1.1 Bruk kun hvert n-te bilde

Den første løsningen som ble vurdert var å ikke bruke hvert eneste bilde i videoen, men hente ut hver n-te bilde. Antallet bilder som hoppes over bestemmes ut fra det totale antallet bilder i videoen.

Metoden tar ikke hensyn til hastighet på skrolling og kan resultere i at informasjon blir repetert flere ganger, eller at informasjon blir hoppet over.

#### 3.1.1.2 Sammenligning av bilder

Ved å sammenligne bilder er det mulig å vurdere om bilder er unike, og dermed unngå overlapp. Dette er en krevende løsning da skjermopptak av chat-samtaler inneholder mye ensfarget bakgrunn som betyr mye likhet mellom alle bildene.

#### 3.1.1.3 Skrolle-hastighet

Ved å jobbe med en video sin verdi for bilder per sekund (FPS), varighet og forskyvning av tekst i bilder kan det være mulig å beregne skrolle-hastighet for en video. Hastigheten kan da brukes for å beregne ved hvilket intervall det skal hentes ut bilder som er unike.

### 3.1.2 Alternative løsninger for transkribering av skjermbilde

#### 3.1.2.1 Egendefinert tegngjenkjenning

Ved å utvikle en egendefinert OCR metode kan den skreddersys til typen av videoer som brukes og detaljer om teksten. En slik metode kan utvikles ved å trene en maskinlæringsmodell til å gjenkjenne tekst i bilder fra videomateriale som produktet er ment for. Dette er en krevende løsning da det tar lang tid å utvikle en metode med høy kvalitet og det kreves mye data for å trene og teste en maskinlæringsmodell.

#### 3.1.2.2 Tegngjenkjenning fra bibliotek basert på åpen kildekode

Python pakken Tesseract tilbyr de funksjonalitetene som prosjektet krever relatert til OCR. Tesseract er et bibliotek med utfyllende dokumentasjon og lisenser.

### 3.1.3 Alternative løsninger for formatering av transkribert tekst

#### 3.1.3.1 Enkel tekstfil

Å skrive den transkriberte teksten til en '.txt'-fil er en løsning som krever lite arbeid. Transkriberingen lagres slik den kommer ut fra OCR metoden, uten ekstra formatering ellers forklaring av innholdet.

#### 3.1.3.2 Strukturert tekstfil

Lagring av transkribert tekst til et strukturert format medfører kategorisering av innholdet. Det strukturerte formatet som foreslås er JSON (JavaScript Object Notation) da dette er et standardisert format.

### 3.1.4 Diskusjon av alternativene

#### 3.1.4.1 Skjermopptak til skjermbilder

Den beste løsningen for å dele opp et skjermopptak til skjermbilder er å hente ut minst antall bilder som representerer fullstendig innhold, uten overlapp i tekst. Hvilket alternativ som skal velges vil være avhengig av hastigheten til metoden, korrekthet i endelig tekst og antall bilder som brukes. Det vil også bli vurdert hvor mye overlapp som blir transkribert.

#### 3.1.4.2 Transkribering av skjermbilde

Prosjektet er omfattende, og det er ønskelig å inkludere ekstra funksjonalitet for å gjøre programvaren best mulig ut fra oppdragsgiver sine behov. Å utvikle en egen OCR pakke vil kreve mye arbeid og ressurser, noe som ikke er ideelt på et tidsbegrenset prosjekt. En bedre løsning er å bruke en gratis og fullstendig pakke som utfører den samme oppgaven, da dette har høyere kvalitet enn hva gruppen kan produsere basert på kompetanse. Ved å bruke en eksisterende pakke er det mulig å fokusere på prosjektet i sin helhet, og ikke bare på en del.

#### 3.1.4.3 Formatering av transkribert tekst

Formatet til den endelige transkriberte teksten avhenger av hva teksten skal brukes til. I løpet av utviklingen og i testingen holder det med en enkel tekstfil for å undersøke om transkriberingen blir korrekt. Når den transkriberte teksten skal brukes utenom utvikling blir formateringen viktigere med hensyn til eksportering og lesbarhet.

## 3.2 Valgt løsning

Løsningene som det ble valgt å jobbe videre med er OCR for transkribering basert på åpen kildekode, fordi utvikling av en egen OCR er for tidkrevende for dette prosjektet. For deling av skjermopptak til bilder ble det utført flere tester før en løsning ble valgt. Oppdelingen utføres ved å sammenligne bilder ved bruk av bilde-hash. For formateringen av den transkriberte teksten vil begge løsningene brukes i ulike iterasjoner av utviklingen. Målet er at den endelige løsningen skal bruke et strukturert format slik at det er mulig å skille mellom dato, klokkeslett, avsender og meldingens innhold. En slik strukturering gjør det lettere å gjennomføre testing av transkriberingens korrekthet og eksportering til PDF-fil.

Mot slutten av prosjektet ble kravene til oppgaven oppdatert av oppdragsgiver. Oppdragsgiver ønsket å skifte fokus fra fullstendig transkribering til å kunne søke etter ord i den transkriberte teksten. Hvis det oppdages et fullstendig eller delvis treff for ordet, returneres skjermbildene for treffene. Mesteparten av den valgte løsningen er enda blitt anvendt i den endelige programvaren, uten formatering av den transkriberte teksten. Det nye kravet krever mindre fokus på fullstendig unike skjermbilder og den eksporterte PDF-filen trenger kun informasjon om videoen som transkriberes og de relevante skjermbildene.

## 3.3 Prosjektmetodikk

### 3.3.1 Utviklingsmetodikk

Programvaren som skal utvikles bygger i grunn på et konsept; å transkribere tekst fra video. For å tilfredsstille kravene fra oppdragsgiver vil ekstra funksjonalitet inkluderes for å forbedre resultatene. På grunn av dette ble det valgt å jobbe med en iterativ utviklingsmetodikk, inspirert av den agile utviklingsmetoden (Cockburn & Highsmith, 2001).

Prosjekter som bruker en agil utviklingsmetode tar utgangspunkt i en iterativ og inkrementell prosess for å oppnå delvis fullført programvare etter hver iterasjon, hvor iterasjonene bygger på hverandre (Zhong, Liping, & Tian-en, 2011). Agil arbeidsteknikk gjør det mulig å få raskere tilbakemeldinger fra oppdragsgiver og gir mulighet for å legge til ekstra funksjoner uten at eksisterende kode må endres radikalt. Hver iterasjon består av flere oppgaver som må utføres for å nå det overordnede målet til iterasjonen.

I starten ble det valgt å dele utviklingen inn i 6 ulike iterasjoner, hvor detaljert informasjon er gitt i Tabell 1. En ekstra iterasjon ble inkludert da det ble spesifisert nye krav fra oppdragsgiver etter at iterasjon 6 var gjennomført. I iterasjon 1 er det fokus på at de mest grunnleggende kravene skal oppnås; et skjermopptak deles inn i skjermbilder, og transkribering av ett enkelt skjermbilde. Iterasjon 2 fokuserer på å kombinere metodene fra iterasjon 1 slik at det er mulig å få ut transkribert tekst for hele videoen. Fokuset i iterasjon 3 er kvaliteten på den transkriberte teksten slik at denne blir korrekt i forhold til hvordan den er i skjermopptaket.

Iterasjon 4 og 5 har som mål å videre forbedre den transkriberte teksten og oppdeling av video. I tillegg er det fokus på formateringen av den transkriberte teksten slik at resultatet er mer forberedt til eksportering. I iterasjon 5 skal det også utvikles et første utkast for GUI slik at brukere lettere kan bruke programvaren.

Iterasjon 6 var den siste iterasjonen og handler om tilpasning av programvaren for brukeren. Et brukervennlig brukergrensesnitt skal fungere som ønsket, og resultatet skal eksporteres på en god måte basert på oppdragiver sine krav.

For iterasjon 7 var det fokus på å få implementert metode for å finne skjermbilder med total eller delvis treff på søkeord gitt av brukeren. Det var også et mål å få laget en installasjonsfil som kan kjøres av brukeren for å installere de bibliotekene som er nødvendig for at programvaren skal kjøre.

*Tabell 1 Beskrivelse av planlagte utviklingsiterasjoner*

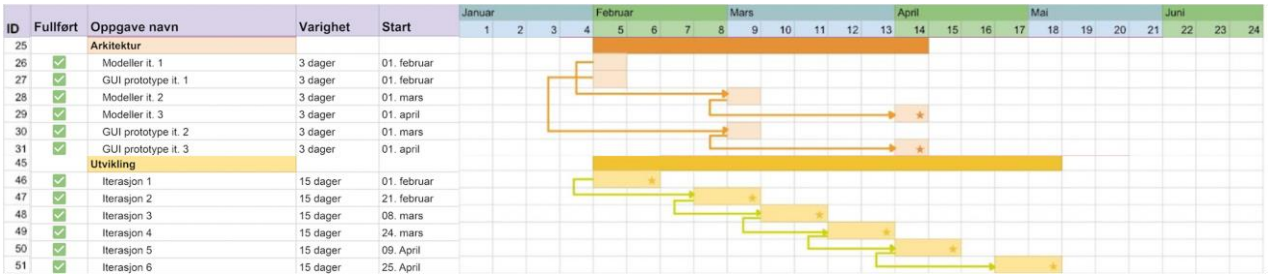
| It. | Mål for iterasjonen | Oppgaver i iterasjonene | Tester | Hva skal man oppnå |
|---|---|---|---|---|
| 1 | Fungerende kode som deler skjermopptak opp i skjermbilder, og fungerende kode som transkriberer tekst fra ett enkelt skjermbilde.<br>Metodene trenger ikke samarbeide. | Skrive en klasse som transkriberer tekst fra et enkelt skjermbilde uten å ta hensyn til formateringen av teksten.<br>Dele skjermopptaket opp i færre bilder uten å miste innhold, det er akseptert med duplisert tekst.<br>Skrive en metode som sammenligner den automatisk transkriberte teksten med manuelt transkribert tekst for å beregne hvor korrekt resultatet er. | Skrive testklasser for metoder skrevet, der det er mulig. | Etter iterasjon 1 er det ønskelig å ha en skjelettstruktur for programvaren hvor de helt grunnleggende funksjonene fungerer korrekt. |
| 2 | Metode som kan kjøre de klassene fra iterasjon 1 samtidig.<br>Forbedre resultatet med hensyn på korrekthet. | Utvide koden for transkribering av skjermbilde slik at den kan transkribere flere skjermbilder og lagre resultatet fra alle skjermbildene.<br>Forbedre koden som deler opp skjermopptak til færre bilder for å redusere mengden duplisert tekst.<br>Skrive en main-metode som kjører programmet i sin helhet slik det er nå.<br>Forbedre kvalitet på skjermbildene slik at det er lettere for OCR å kjenne igjen tegn. | Undersøke korrekthet nivået.<br>Utvide tester metodene i forhold til ekstra metoder lagt til.<br>Teste koden med flere dummy videoer. | Etter iterasjon 2 skal programmet kunne fungere for flere input videoer. |
| 3 | Transkribering metoden skal fungere på en bedre måte slik at resultatet blir mer korrekt.<br>Undersøke om det er mulig å håndtere emojier i transkriberingen i forbindelse eller som en del av OCR pakken som brukes. | Legge til at transkribering klassen skal kunne håndtere duplikater og riktig alfabet. | Test klasse for å se at koden fungerer riktig i forhold til alfabet og duplikater.<br>Sjekke korrekthet til resultatet. | Forbedret korrekthet i den transkriberte teksten.<br>Redusere antall duplikater.<br>Ingen problemer med spesialtegn for det norske alfabetet. |
| 4 | God formatering av den transkriberte teksten til en JSON fil. | Lagre den transkriberte teksten til en JSON fil som kan eksporteres.<br>Videre forbedre oppdeling av video til bilder. | Test metoder for å sjekke at de nye tilleggene fungerer. | Fungerende metode som lagrer resultatet på en god måte som enklere kan eksporteres. |

| | | | | |
|---|---|---|---|---|
| 5 | Ha et første utkast på en fungerende GUI.<br>Inkludere at transkribering kan utføres på mappe av bilder i tillegg til video. | Utvikle et enkelt GUI med fokus på funksjon, ikke design.<br>Utvikle egen funksjonalitet for håndtering av mappe med bilder. | Brukertest av GUI på møtet med politiet.<br>Sjekk korrekthet med nye tillegg. | En fungerende GUI som er blitt testet av brukere. |
| 6 | Fungerende eksportering av JSON filen til et godt og leselig format i tekstdokument (helst PDF).<br>Godt fungerende GUI. | Videreutvikling av GUI etter brukernes ønsker, inkludert design.<br>Skrive metode for å kunne eksportere JSON filen til en oversiktlig PDF-fil som er lett å lese. | Møte med oppdragsgiver for å vise endelig funksjonalitet og design av GUI og resultat-fil. | Fungerende programvare med høy korrekthet og resultat som tilfredsstiller oppdragsgiver. |
| 7 | Mulighet for å søke etter ord i den transkriberte teksten og eksportering av skjermbilder med dette ordet, eller lignende ord. | Oppdatere GUI slik at det er mulig å legge inn søkeord.<br>Skrive metode for å finne søkeord med nøyaktig treff, eller delvis treff.<br>Finne skjermbildene hvor søkeordet finnes og eksportere disse til en PDF-fil.<br>Lage en installasjonsfil for å installere pakker som er nødvendig for å kunne kjøre programvaren. | Møte med oppdragsgiver for å gjennomføre brukertesting. | Programvare som finner søkeord i de korrekte skjermbildene og eksporterer disse på et nivå som tilfredsstiller oppdragsgiver. |

### 3.3.2 Prosjektplan

Fremdriftsplanen for prosjektet er organisert i et Gantt-diagram, sammen med en mer detaljert beskrivelse i Kapittel 9.2. Oppgavene som skal utføres i løpet av prosjektet er delt inn i fem ulike kategorier; dokumentasjon, presentasjon, arkitektur, møter og utvikling. Figur 2 viser et utsnitt av Gantt-diagrammet med fasene arkitektur og utvikling.

I løpet av prosjektet er det satt 13 milepæler som representerer viktige hendelser i prosjektforløpet. I dokumentasjonsfasen er det 4 milepæler som representerer viktige innleveringer i forbindelse med prosjektet. Presentasjonsfasen har 1 milepæl for presentasjonen av det fullførte prosjektet. Det er 2 milepæler i arkitekturfasen, en for fullført arkitektur og en for fullført brukergrensesnitt. Resten av milepælene er i utviklingsfasen og representerer hver fullførte utviklingsiterasjon.



| ID | Fullført | Oppgave navn | Varighet | Start |
|---|---|---|---|---|
| 25 | | **Arkitektur** | | |
| 26 | ☑ | Modeller it. 1 | 3 dager | 01. februar |
| 27 | ☑ | GUI prototype it. 1 | 3 dager | 01. februar |
| 28 | ☑ | Modeller it. 2 | 3 dager | 01. mars |
| 29 | ☑ | Modeller it. 3 | 3 dager | 01. april |
| 30 | ☑ | GUI prototype it. 2 | 3 dager | 01. mars |
| 31 | ☑ | GUI prototype it. 3 | 3 dager | 01. april |
| 45 | | **Utvikling** | | |
| 46 | ☑ | Iterasjon 1 | 15 dager | 01. februar |
| 47 | ☑ | Iterasjon 2 | 15 dager | 21. februar |
| 48 | ☑ | Iterasjon 3 | 15 dager | 08. mars |
| 49 | ☑ | Iterasjon 4 | 15 dager | 24. mars |
| 50 | ☑ | Iterasjon 5 | 15 dager | 09. April |
| 51 | ☑ | Iterasjon 6 | 15 dager | 25. April |



*Figur 2 Utsnitt av Gantt diagrammet*

### 3.3.3 Risikovurdering

Det ble gjennomført en risikoanalyse for å få innsikt i hvilke risikoer som er relatert til prosjektet, hvordan de oppstår og tiltak som kan gjøres for å unngå dem. Den fullstendige risikoanalysen er detaljert i Kapittel 9.3. De to største risikoene er synlig i Tabell 2. Risikoene relatert til personer involvert i prosjektet er mulig å forhindre ved å opprettholde god kommunikasjon, planlegge godt og jobbe jevnt i løpet av prosjektet. De tekniske risikoene må løses med god planlegging av iterasjoner og ved å finne gode hjelpemidler til å løse vanskelige oppgaver.

*Tabell 2 De største risikoene fra den fullstendige risikoanalysen*

| | **Hendelse** | **Årsak** | **Sannsynlighet** | **Konsekvens** | **Risiko produkt** | **Tiltak** |
|---|---|---|---|---|---|---|
| 2 | Ulike språk på videoene | Ulikt språk på kommunikasjonen.<br>Ulike språkinnstillinger på telefoner. | Svært høy<br>(5) | Middels (3) | 15 | Tilpasse koden til å akseptere flere alfabet.<br>Bruke kode som kan gjenkjenne språk. |
| 6 | Dårlig oppfølging fra arbeidsgiver. | Oppdragsgiver har ikke tilstrekkelig med tid til oppfølgingsmøter | Middels (3) | Svært høy (5) | 15 | God planlegging og god kommunikasjon mellom partene. |

## 3.4 Evalueringsplan

Evaluering utføres på de ulike komponentene og funksjonalitet til programvaren. Det blir være egen evaluering for oppdeling av video til unike skjermbilder, transkribering, søkefunksjonalitet og brukervennlighet.

Oppdeling av video til unike skjermbilder vurderes ut fra antallet skjermbilder nødvendig for å inkludere fullstendig innhold funnet manuelt og maskinelt. I tillegg til å sammenligne antallet undersøkes det på manglende innhold.

Programvaren evalueres ut fra effektiviteten og korrektheten til transkriberingen. Evalueringen utføres ved å sammenligne maskinell transkribering med manuell transkribering på tekst og tid. For manuell transkribering beregnes tiden ut fra gjennomsnittlig skrivehastighet på 40 ord i minuttet (TypingPal, 2022) og antallet ord som transkriberes. For å sammenligne korrekthet vil den totale transkriberte teksten sammenlignes på ord og tegn.

Søkeordfunksjonaliteten evalueres ved en sammenligning mellom skjermbilder gitt av programvaren og skjermbilder vurdert visuelt som inneholder søkeordet eller lignende.

Under sluttfasen av prosjektet vil produktet bli testet av etterforskere i DPA for å evaluere dens brukervennlighet og funksjonalitet.

# 4 DETALJERT LØSNING

## 4.1 Video- og Mappe håndtering

Før en mappe av bilder eller en video kan transkriberes må innholdet behandles som beskrevet av kapitlene under.

### 4.1.1 Video til bilde

For å kunne transkribere innholdet i en video må den først deles opp i bilder. OpenCV biblioteket inneholder en klasse VideoCapture med ulike metoder for videobehandling, ved å opprette et VideoCapture objekt av videofilen kan klassens read() metode benyttes for å hente ut bildene fra videofilen. Dette steget er ikke nødvendig for mapper da innholdet allerede er bildefiler.

### 4.1.2 Beskjæring av bilder

Bilder hentet fra video eller mappe må beskjæres slik at elementer som kan forstyrre transkriberingen av chat innholdet fjernes. Et forstyrrende element kan for eksempel være mobilens verktøylinje, plasseringen og størrelsen av denne kan variere avhengig av mobilens størrelse og operativsystem. For at beskjæringen skal fungere etter hensikt må den være tilpasset mobilen som er brukt til bevismateriale, derfor inkluderer GUI en funksjon hvor bruker kan markere den delen av bilde som skal beholdes. Bildene fra mappen eller videoen beskjæres i henhold til inndata fra bruker.

### 4.1.3 Filtrering av bilder

Det ønskes kun å transkribere bilder med unikt innhold, men spesielt i bilder hentet fra video vil det finnes gjentagende innhold i flere av bildene. Til å filtrere ut bilder som inneholder gjentagende innhold benyttes det perseptuell hashing som baserer seg på at lignende entiteter skal bli hashet til lignende hash verdier (Drmic, Silic, Delac, Vladimir, & Kurdija, 2017). Perseptuell hashing differer fra de kryptografiske hashfunksjoner hvor små forskjeller kan gjøre store utslag på hash-verdien. ImageHash biblioteket i Python inneholder metoder for forskjellige perseptuelle hash algoritmer og metoder for å sammenligne hash-verdier. Filtreringene forklart i de to neste kapitlene anvender dette bibliotekets P-hash algoritme som baserer seg på diskret cosinus transformasjon.

#### 4.1.3.1 Første filtrering

For å redusere antall bilder før den mer komplekse filtreringen forklart under, utføres først en grov filtrering. For å avgjøre om et bilde skal beholdes så kalkuleres differansen mellom bildet sin hash verdi og hash verdien til neste bilde, om denne differansen er over en bestemt verdi beholdes bilde. Den filtreringen krever n-1 sammenligninger for n bilder.

#### 4.1.3.2 Andre filtrering

En grundigere filtrering krever en mer kompleks løsning for å luke ut bilder med gjentagende innhold. Dette gjøres ved å generere hash verdien til del nederst av et bilde med høyde 100 piksler og deretter søke gjennom de etterfølgende bildene etter en del med lignende hash verdi. Visst et bilde ikke inneholder en slik del så beholdes bildet og nederste del av bildet blir brukt til videre søk.

## 4.2 Transkribering og eksportering

### 4.2.1 Transkribering

Som tidligere beskrevet i oppgaven utføres transkriberingen ved OCR teknologi. Tesseract biblioteket gir bedre resultater ved tydelig svart tekst på hvit bakgrunn. Derfor er det nødvendig å preprosessere bildene før de transkriberes. Når preprosesseringen er utført sendes det prosesserte bildet videre til OCR teknologien.

#### 4.2.1.1 Preprosessering

##### 4.2.1.1.1 Gråskala

OCR biblioteket som brukes krever at bildene som sendes inn er i gråskala. Gråtoner gjør det bedre for videre prosessering som terskel prosessering (thresholding). Det er da ikke nødvendig å ta hensyn til om fargene er kraftige eller ikke. Figur 3a viser et eksempel på et skjermbilde før det sendes gjennom preprosessering og Figur 3b viser det samme bildet etter at det er gjort om til gråskala.

Bildene i eksempelet viser en sirkel på skjermen som heter Assistive Touch, denne fungerer som en hjem-knapp. Den er inkludert for å vise hvordan uønsket informasjon på bilder påvirkes av preprosesseringen.


*(a) Originalt skjermbilde* *(b) Skjermbilde i gråskala*

*Figur 3 Skjermbilde før og etter gråskala transformasjon*

#### 4.2.1.1.2 Skalering

Bildene skaleres slik at teksten i bildene skal bli tydeligere og slik at det er bedre for OCR å lese den av. Bildet sin størrelse i piksler blir økt uten at tydeligheten går tapt. For å sikre at bildet ikke reduseres i kvalitet brukes bi-kubisk interpolasjon for å bestemme fargeverdien til nye piksler. Bi-kubisk interpolasjon tar et vektet gjennomsnitt på de 4x4 pikslene på området rundt for å komme frem til den nye verdien.

Figur 4a viser eksempelbildet i gråskala før skalering og Figur 4b viser eksempelbildet etter skaleringen. Ut fra Figur 4b er det synlig at teksten blir tydeligere og lettere å lese etter skaleringen. Hvor høy skaleringsverdi som brukes ble bestemt ved testing av ulike verdier mellom 2 og 14, til slutt ble det valgt å bruke 10 da denne verdien ga god kvalitet uten at videre prosessering tok for lang tid.


*(a) Skjermbilde i gråskala* *(b) Skjermbilde etter skalering*

*Figur 4 Skjermbilde før og etter skalering*

#### 4.2.1.1.3 Thresholding

Thresholding anvendes for å bestemme hvilke piksler som skal være tydelige og hvilke som skal gli inn med bakgrunnen. Ut fra en terskelverdi bestemmer metoden at piksler med verdi lavere enn terskelen settes til hvit (bakgrunn) og resten settes til sort. Etter dette vil teksten være tydeligere i bildet og noe støy fjernes.

Hvilken verdi som brukes som terskel krevde mye testing med ulike bilder og ulike verdier. Den endelige verdien som ble brukt er den verdien som er størst av gjennomsnittlig pikselverdi til bildet og 84% av den høyeste pikselverdien i bildet. Denne løsningen fører til at teksten blir tydeligere i flertallet av bilder uten at informasjon med lysere skrift går tapt.

Figur 5a viser det skalerte bildet i gråskala slik det er før thresholding og Figur 5b viser det samme bildet etter gjennomført thresholding. I Figur 5b er det synlig at teksten blir kraftigere etter utført thresholding og informasjon som ikke tolkes av programvaren som Assistive Touch og emojier blir helt sorte.


*(a) Skalert skjermbilde* *(b) Skjermbilde etter thresholding*

*Figur 5 Skjermbilde før og etter thresholding*

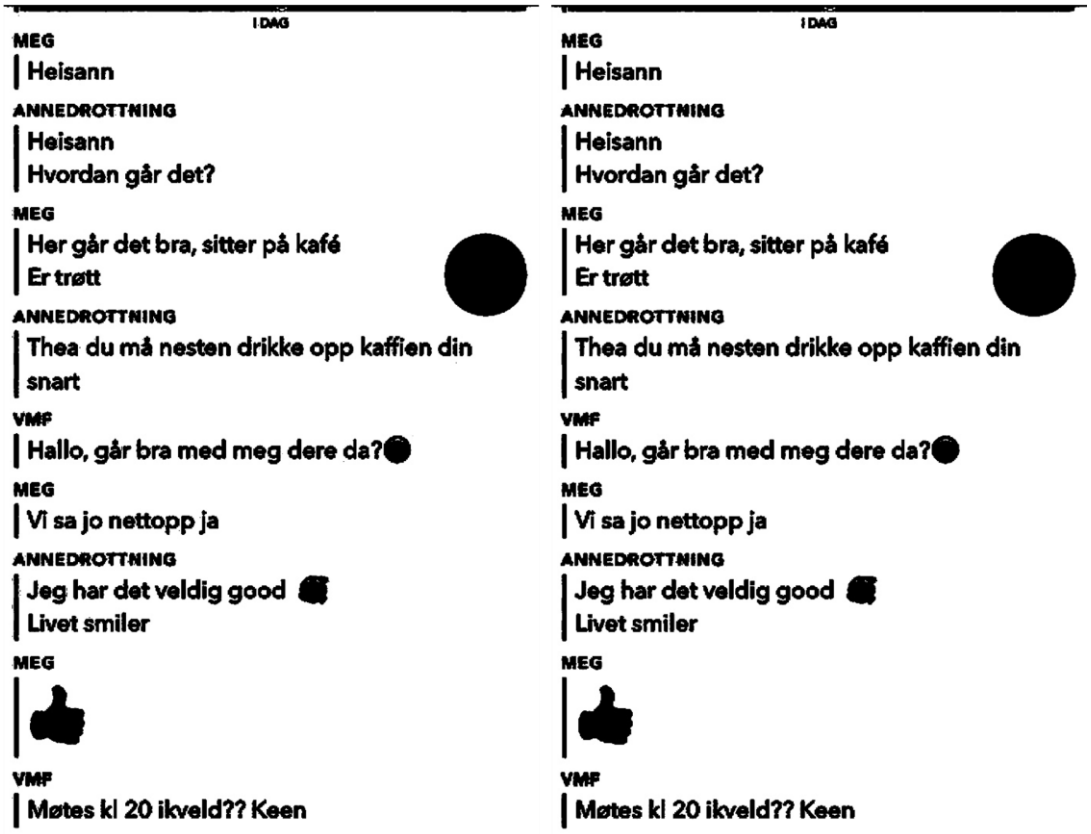
#### 4.2.1.1.4 Fjerning av støy

Støy er tydelige piksler som det ikke er ønskelig å transkribere som tekst. Dette er ønskelig å fjerne og det brukes da to teknikker i kombinasjon; fortynning (dilution) og erosjon (erosion). Begge metodene eksisterer som innebygde metoder i OpenCV pakken til Python.

Fortynning brukes for å fremheve elementer i bildet. Det brukes en kjerne som kombineres med bildet hvor piksler blir satt til sort (fremhevende) om minst en piksel under kjernen er satt til sort. Teksten i bildet vil da bli større.

Erosjon fjerner grenser satt i forgunnsvinduet som vil gjøre teksten tydeligere. Denne teknikken bruker også en kjerne som kombineres med bildet, hvor pikslene blir satt til sort kun hvis alle pikslene i kjernen er sort. Piksler som har blitt satt til sort i primære bakgrunnsområder ved en feil vil da fjernes.

Figur 6a viser eksempelbildet etter at det har blitt gjort til gråskala, skalert og gjennomgått thresholding, og Figur 6b viser dette bildet etter de to metodene beskrevet over. Eksempelbildet i Figur 6a er påvirket av noe støy som er borte i Figur 6b. Dette er synlig blant annet foran og over første 'MEG' og mellom T-ene i første 'ANNEDROTTNING'.




*(a) Thresholdet bilde* *(b) Skjermbilde etter støyfjerning*

*Figur 6 Skjermbilde før og etter støyfjerning*

Dette er siste trinnet i preprosesseringen og bildet i Figur 6b er det som vil sendes som inndata til tegngjenkjenningsteknologien.

### 4.2.2 Eksportering

Før den transkriberte teksten kan eksporteres må teksten prosesseres. Formatet teksten kommer ut fra OCR er ren tekst uten struktur. Teksten må prosesseres for å finne forfattere, klokkeslett og meldinger. Det kan også være ekstra informasjon som datoer som må håndteres i tillegg. For enklere eksportering skal teksten lagres på strukturert JSON format.

For å kjenne igjen forfattere brukes navnene oppgitt av brukeren i kombinasjon med gjenkjenning av klokkeslett da disse vanligvis kommer på samme linje. Klokkeslett kjennes igjen ved bruk av regulære uttrykk. Meldingene sendt er da teksten som står igjen.

Fra strukturert format hentes dataene ut og representeres oversiktlig i et PDF-dokument produsert med Python pakken FPDF. Teksten fra hvert bilde skrives ved siden av bildet det hører til.

I dokumentet vil det oppgis informasjon om videoen eller mappen som er blitt transkribert på forsiden før den transkriberte teksten. Dette inkluderer MD5-hash av videoen eller mappen, antall bilder transkribert og datoen for transkriberingen. For videoer inkluderes også rammer per sekund og varighet på videoen.

I den endelige programvaren brukes ikke metoden beskrevet over for eksportering da det ikke er nødvendig for de oppdaterte kravene. Den transkriberte teksten trenger ikke prosesseres da det unødvendig for å finne igjen søkeord. Teksten for hvert bilde lagres i en tabell slik at det er mulig å søke etter søkeordet i hver bildetekst. For å finne treff på ordet brukes et regulært uttrykk av søkeordet med et bestemt antall lovlige substitusjoner. Antallet lovlige substitusjoner er satt til halvparten av lengden til ordet rundet ned. Hvis søkeordet oppdages i en bildetekst vil bildet lagres for eksportering og det hoppes direkte videre til neste bilde siden det ikke er nødvendig å lete videre i dette bildet. Hvis søkeordet ikke oppdages i bildeteksten, går søket videre til neste bilde.

Når søket er ferdig med alle bildene eksporteres hvert bilde til en PDF-fil med beskrivelse på hvilken video eller mappe som ble brukt og søkeordet på forsiden. For hvert bilde som eksporteres skrives også navnet på bildet slik at det er mulig å finne dette igjen i sekvensen av bilder.

Etter at PDF-filen er opprettes åpnes den direkte slik at brukeren kan undersøke den med en gang.

## 4.3 Grafisk brukergrensesnitt

Grafisk brukergrensesnitt, også omtalt som GUI som er en forkortelse for engelske Graphical User Interface, blir brukt slik at brukeren kan samhandle med programmet ved hjelp av anordninger. Det grafiske brukergrensesnittet utvikles ved å bruke Python biblioteket tkinter.

### 4.3.1 Sideoppbygging

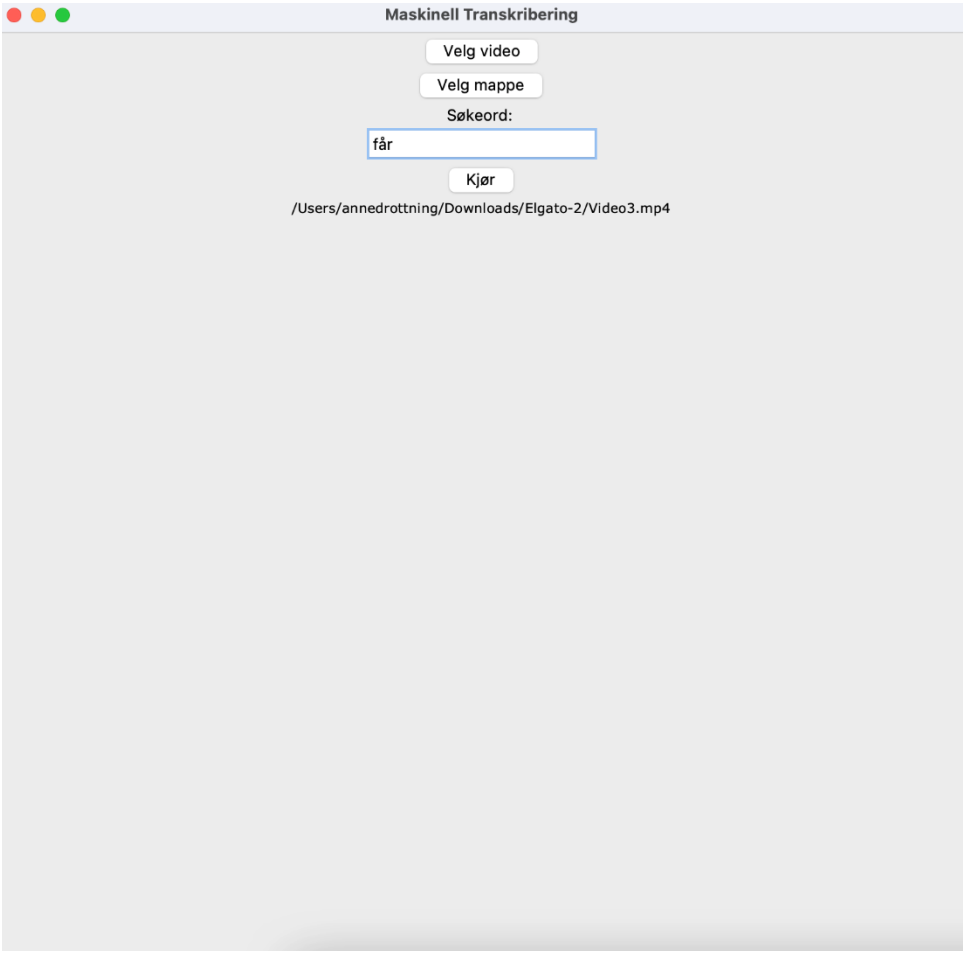
Før implementering av ulike sider og funksjoner trenger man å importere selve tkinter og opprette et vindu som det er mulig å jobbe med videre. Når det første vinduet opprettes settes små avgrensninger, som størrelse, bakgrunnsfarge og tittel på vinduet.

Videre blir det bygget klasser for de ulike sidene. For at det skal være mulig å ha flere sider på brukergrensesnittet istedenfor vinduer, er det nødvendig med en hovedklasse som oppretter vinduet ved hjelp av en beholder som samler alle sidene ved å bruke den innebygde tkinter ramme metoden. Dette medfører at alle sidene kan samles i ett vindu ved bruk av arv. Her vil startsiden være foreldresiden.

I den endelige programvaren ble det ikke brukt sider da søkefunksjonaliteten og opplasting av video eller mappe utføres på en side. Det ble derfor kun brukt en startside som beskrevet under.

#### 4.3.1.1 Startsiden

På startsiden kan bruker velge om de vil transkribere en video eller bilder i en mappe. I tillegg må brukeren skrive inn søkeordet de vil bruke. Figur 7 viser hvordan startsiden ser ut etter at det er blitt valgt en video og skrevet inn et søkeord. Når brukeren trykker på ‘Velg video’ eller ‘Velg mappe’ vil det åpne seg en filutforsker slik at brukeren kan velge en lokal fil eller mappe. Etter at transkriberingsobjektet er valgt og søkeordet er skrevet inn trykker brukeren på ‘Kjør’.



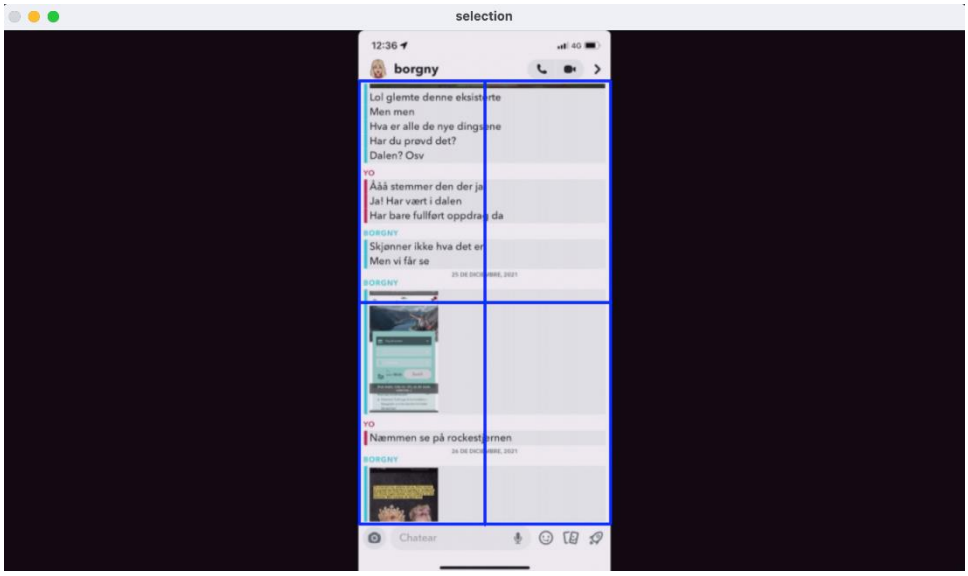

*Figur 7 Brukergrensesnitt til programvaren*

### 4.3.2 Utklipp og dimensjoner

Etter at brukeren har trykket 'Kjør' lastes filen eller mappen som skal transkriberes opp og brukeren får opp et interaktivt vindu med det første skjermbildet. Her må brukeren anvende et innebygd utklippsverktøy fra OpenCV kalt selectROI for å velge området av bildet som skal transkriberes. Dette er nødvendig for at programvaren ikke skal håndtere unødvendig informasjon og piksler som kan ødelegge for preprosessering og redusere effektiviteten til algoritmene brukt.

Figur 8 viser et eksempelbilde i det interaktive vinduet. For at det skal være mulig å velge område fra bilder med høyere oppløsning enn skjermen til maskinen brukt endres størrelsen til bildet slik at bredden er 1000 piksler, og høyden endres dynamisk ut fra denne verdien for at bildet enda skal representeres riktig.

I Figur 8 er det mye sort bakgrunn rundt området som skal transkriberes. Dette er en effekt av skjermopptak fra Elgato Game Capture. Det blå rektangelet er området valgt av brukeren. I tillegg til å fjerne den sorte bakgrunnen fjernes irrelevant informasjon som og skrivefelt.




*Figur 8 Eksempelbilde med utvalgt område*

Utklippsverktøyet returnerer koordinater som beskriver start x- og y-posisjon, bredden og høyden til det blå rektangelet. Før koordinatene returneres beregnes de fra den endrede størrelsen til den korrekte oppløsningen til bildet slik at de kan anvendes ved videre prosessering.

Videre sendes dimensjonene og filen til video- og mappehåndteringsmetodene beskrevet i Kapittel 4.1 og Kapittel 4.2.

# 5 RESULTATER

Resultater og evalueringer brukes for å utføre en vurdering av produktet i forhold til problemstillingen og forskningsspørsmålene. Det er ønskelig å svare på om den maskinelle transkriberingen sparer tid og ressurser sammenlignet med den manuelle transkriberingen.

I løpet av utviklingsprosessen er det brukt visuelle vurderinger på korrekt funksjon av metoder fordi et flertall av metodene anvendt er importert fra godt dokumenterte pakker med egne enhetstester. Dette gjorde det unødvendig å skrive egne enhetstester til disse metodene. I tillegg bruker flere metoder bilder som det var lettere å sammenligne visuelt enn digitalt.

Resultatene er utarbeidet i forhold til evalueringsmetodene beskrevet i Kapittel 3.4 og Kapittel 5.1.

## 5.1 Evalueringsmetoder

Evaluering av resultater er brukt ved fire metoder: brukertesting, sammenligning av unike bilder, sammenligning av transkribert tekst og hvor korrekt programvaren finner skjermbilder med søkeordet.

### 5.1.1 Brukertesting

Brukertesting går ut på at oppdragsgiver tester programvaren på videoer som ikke er brukt i utviklingsprosessen. Oppdragsgiver kan da gi tilbakemelding på brukervennlighet, oppfylte krav, kvalitet på resultatet i forhold til forventninger og kommentere ønsker eller mangler med produktet.

For brukertestingen er det skrevet en liste med elleve spørsmål som det er ønsket å få svar på:

- Før programvaren kan brukes kreves det installasjon av Python pakker som brukes, hvordan gikk denne installasjonen?
- Hvilke funksjoner er dere fornøyd med?
- Hvilke funksjoner skulle dere ønske var bedre?
- Hvilke funksjoner skulle dere ønske var inkludert?
- Hvordan er effektiviteten sammenlignet med manuell gjennomføring?
- Er det lett å forstå hvordan man bruker programvaren ut fra brukergrensesnittet?
- Hvor godt løser programvaren oppgaven?

- Hvor lett tror dere det blir å huske hvordan produktet brukes hvis det ikke blir brukt på en stund?
- Føler dere at det tar lang tid å sette seg inn i hvordan produktet fungerer?
- Hvis det kommer feilmeldinger: hvor forståelig var feilmeldingene?
- Generell vurdering av produktet (terningkast)

### 5.1.2 Sammenligning av unike bilder og bilder hentet ut

Beregning av filtreringens korrekthet gjennomføres ved å manuelt finne bildene i en video som representerer innholdet i chat-samtalen uten duplikater. Antallet av de unike bildene sammenlignes med antall bilder programmet henter ut.

### 5.1.3 Tekstsammenligning

For å kunne beregne hvor korrekt den transkriberte teksten er sammenlignes den manuelt transkriberte teksten med den maskinelt transkriberte teksten. Det er da mulig å se om videoene er delt opp på en måte som beholder all informasjonen og om transkriberingen har blitt utført optimalt eller ikke. Dette gjennomføres ved å utføre tekstsammenligning på tegn og på ord.

I programvaren er det lagt inn tidsstempler slik at tiden programvaren bruker på oppgaven kan hentes ut og dette vil sammenlignes med tiden det tar å transkribere tekst manuelt.

### 5.1.4 Skjermbilder med søkeordet

Beregning av korrekthet til søkefunksjonen i programvaren gjennomføres ved å manuelt finne skjermbildene hvor det er søkeordet finnes og sammenligne med skjermbildene funnet av programvaren. Dette vil gi en prosent som beskriver hvor bra programvaren finner søkeordet og beskrive situasjoner hvor programvaren ikke finner søkeordet.

## 5.2 Evalueringsresultat

### 5.2.1 Brukertesting

Brukertestingen ble gjennomført 19. Mai 2022 med to etterforskere hos DPA: Vilde Margrethe Jacobsen og Petter Osland Aarberg. Etterforskerne kommer ikke fra tekniske bakgrunner og vil få nytte av programvaren i sine arbeidsoppgaver. Programvaren ble lastet ned på Vilde Margrethe sin datamaskin og ble kjørt på videoer som ikke er brukt i utviklingsprosessen.

Brukertestingen startet med installasjon av Python og de nødvendige bibliotekene. Installasjon av bibliotekene utføres ved å kjøre en Windows batch fil i ledeteksten før programvaren kjøres fra samme kommando prompt. Grunnet manglende teknisk bakgrunn på brukerne ble dette gjennomført i hovedsak av gruppen. Tilbakemelding fra brukerne viste et ønske om en kjørbar '.exe' fil som starter programvaren da de ikke var kjent med kommandoer i ledeteksten. Hos oppdragsgiver er det en driftsansvarlig som setter opp miljøet på maskinene brukt av etterforskere, og som kan ta ansvar for installasjon nødvendige pakker. Brukerne uttalte at med en detaljert brukermanual på installasjon vil det være mulig for dem å gjennomføre dette selv.

I brukertestingen ble det oppdaget problem med skalering på vinduet hvor utklippsverktøyet anvendes da oppløsningen på bildet var større enn skjermen brukt og det ikke var mulig å redusere størrelsen på vinduet. Dette problemet ble løst før prosjektet var ferdig.

Søkeordet ble ikke funnet igjen i alle skjermbildene hvor det var, og brukerne meldte interesse av å vite hvor korrekt denne funksjonaliteten er da det er relevant for deres arbeid. Korrektheten til søkefunksjonalitet er beskrevet i Kapittel 5.2.4.

Brukerne var spesielt fornøyd med programvaren sin evne til å automatisk og effektivt dele video opp til skjermbilder som representerer den fullstendige teksten på et begrenset antall bilder. Etterforskere bruker mye tid på å finne skjermbilder manuelt og brukerne fortalte at dette vil kunne spare oppdragsgiver for mye penger og tid. Vest politidistrikt har mer digitale ressurser enn mange andre politidistrikt og programvaren vil være et godt tilskudd til videre effektivisering av arbeidsoppgaver.

Tilbakemelding på programvaren i seg selv og brukergrensesnittet var at det var intuitivt å forstå hvordan det fungerte og at programvaren løste oppgaven på en god måte. Om det går lang tid uten å bruke programvaren mente brukerne at det ikke var problematisk å sette seg inn i funksjonaliteten igjen, spesielt ved hjelp av en detaljert brukermanual.

Oppsummert ga brukerne programvaren terningkast 5, og det ble nevnt gjentatte ganger at programvaren vil spare penger og effektivisere arbeidet betydelig.

### 5.2.2 Sammenligning av unike bilder og bilder hentet ut

Etter programmets bildefiltrering er gjennomført sammenlignes antall bilder hentet ut av programmet med antall manuelt valgte bilder. Sammenligningene ble gjennomført på 16 videoer som er beskrevet i mer detalj i Tabell 3. Felles for alle videoene er at oppløsningen er på

1280x720 piksler da dette er standarden for Elgato Game Capture. Antallet bilder som hentes ut fra video er en tilnærmet verdi, som betyr at verdien ikke stemmer nøyaktig med rammer per sekund (FPS) multiplisert med varighet på videoen. Som synlig i tabellen er det fokusert på videoer av Snapchat samtaler med lyst tema da Snapchat samtaler var utgangspunkt for oppgaven og lyst tema er standardinnstillingen.

*Tabell 3 Beskrivelse av videoer brukt for å finne resultater*

| **Videonavn** | **Varighet i sekunder** | **FPS** | **Antall bilder** | **Chat-medium** | **Tema** | **Elementer utenom tekst** | **Forfattere** | **Skrolle-hastighet** |
|---|---|---|---|---|---|---|---|---|
| SC-HoyFPS,3,emojis-versjon2.mp4 | 4 | 60 | 227 | Snapchat | Lyst | Emojier | 3 | Medium |
| SC-HoyFPS,3,emojis.mp4 | 4 | 60 | 206 | Snapchat | Lyst | Emojier | 3 | Medium |
| SC-MedFPS,3,emojis,skjermbilder,linker.mp4 | 27 | 60 | 1586 | Snapchat | Lyst | Emojier Bilder Lenker | 3 | Rolig |
| SC-NormFPS,2,emojis_skjermbilder.mp4 | 22 | 60 | 1305 | Snapchat | Lyst | Emojier Bilder | 2 | Rolig |
| SC-NormFPS,3,emojis.mp4 | 8 | 60 | 455 | Snapchat | Lyst | Emojier | 3 | Rolig |
| Video1.mp4 | 4 | 60 | 265 | Snapchat | Lyst | Emojier Bilder Lenker | 3 | Hurtig |
| Video2.mp4 | 37 | 60 | 2235 | Snapchat | Lyst | Emojier Bilder Lenker | 3 | Rolig Hakkete |
| Video3.mp4 | 18 | 30 | 541 | Snapchat | Lyst | Emojier Bilder Lenker | 2 | Medium |
| Video4.mp4 | 20 | 30 | 587 | Snapchat | Mørkt | Emojier Bilder Lenker | 3 | Medium |
| Video5.mp4 | 3 | 60 | 167 | Snapchat | Mørkt | Emojier Bilder Lenker | 3 | Hurtig |
| Video6.mp4 | 7 | 60 | 391 | Discord | Mørkt | Emojier Bilder | 2 | Hurtig Frem og tilbake |
| Video7.mp4 | 6 | 60 | 389 | Messenger | Mørkt | Emojier Lenker | 2 | Hurtig |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Video8.mp4 | 10 | 60 | 606 | Messenger | Mørkt | Emojier Bilder Lenker | 2 | Variasjon av hurtig og medium |
| Video10.mp4 | 12 | 60 | 700 | Snapchat | Lyst | Emojier | 2 | Medium |
| Video11.mp4 | 5 | 60 | 272 | Snapchat | Lyst | Emojier | 2 | Medium |
| Video12.mp4 | 13 | 60 | 759 | Snapchat | Lyst | Emojier Starter på utsiden av samtalen | 2 | Medium |

For hver video kjøres programmet tre ganger, dette er fordi beskjæringen av bildene som bestemmes av bruker i brukergrensesnittet påvirker programmets bildefiltrering. Det tas gjennomsnittet av antall bilder programmet henter ut for hver video, og dette gjennomsnittet sammenlignes med antall bilder hentet ut manuelt. Resultatet fra evalueringen vises i Tabell 4.

I gjennomsnitt hentes 120% av de korrekte bildene ut, dette betyr at programmet i høy grad har evne til å hente ut bilder med begrenset overlapp i innhold fra video, uten å miste innhold. Gjennomsnittet ligger over 100% som betyr at det hentes i snitt ut flere bilder enn nødvendig. For videoer med hurtig skrolle-hastighet blir det i snitt hentet ut mindre enn 100% av det korrekte antall bilder som tilsier at resultatene mangler innhold fra samtalen.

*Tabell 4 Sammenligning av antall bilder som hentes ut automatisk og manuelt for å dekke alt innhold i videoene*

| **Video tittel** | **Gjennomsnittlig antall bilder hentet ut av programmet** | **Korrekt antall hentet manuelt** | **Forhold mellom antall bilder hentet og korrekt antall i prosent** |
|---|---|---|---|
| SC-HoyFPS,3,emojis.mp4 | 3,60 | 2 | 180% |
| SC-HoyFPS,3,emojis-versjon2.mp4 | 7,30 | 5 | 146% |
| SC-MedFPS,3,emojis, skjermbilder,linker.mp4 | 35,00 | 26 | 135% |
| SC-NormFPS,2,emojis ,skjermbilder.mp4 | 7,60 | 4 | 192% |
| SC-NormFPS,3,emojis.mp4 | 3,30 | 2 | 167% |
| Video1.mp4 | 11,60 | 13 | 89% |
| Video2.mp4 | 27,30 | 21 | 130% |
| Video3.mp4 | 32,00 | 25 | 128% |
| Video4.mp4 | 33,30 | 27 | 123% |
| Video5.mp4 | 10,30 | 13 | 79% |
| Video6.mp4 | 3,70 | 6 | 61% |

| | | | |
|---|---|---|---|
| Video7.mp4 | 5,00 | 8 | 63% |
| Video8.mp4 | 15,60 | 23 | 68% |
| Video10.mp4 | 8,88 | 8 | 108% |
| Video11.mp4 | 7,00 | 6 | 117% |
| Video12.mp4 | 31,00 | 24 | 129% |
| **Gjennomsnitt av sum:** | | | **120%** |

### 5.2.3 Tekstsammenligning

Etter at en video eller mappe var blitt transkribert sammenlignes resultatet med manuelt transkribert tekst for samme filmen. For å få en god sammenligning mellom disse er det sammenlignet på tid, korrekthet i fullstendige ord og korrekthet i tegn. I sammenligningen blir det sett bort fra duplisert tekst da dette ikke har konsekvenser for den endelige søkefunksjonaliteten.

Resultatene fra evalueringen vises i Tabell 5 for hver video med prosentverdier for antall korrekte ord og antall korrekte tegn fra den automatisk transkriberte teksten. For videoene med mørkt tema er resultatene betydelig dårligere fordi tegngjenkjenningsteknologien er tilpasset for sort tekst på hvit bakgrunn og programvaren er ikke tilpasset for mørk modus.

Mengden korrekte tegn i automatisk transkribert tekst er høyere enn korrekte ord da feil transkribering av et tegn i et ord vil føre til feil transkribert ord. På grunn av avhengigheten mellom feil transkribering i tegn og ord brukes det tilnærmede søk i søkefunksjonaliteten. Gjennomsnittlig blir 90.08% av tegn gjenkjent rett som viser at transkriberingen har høy korrekthet.

*Tabell 5 Resultater fra sammenligning av automatisk og manuelt transkribert tekst på korrekthet*

| **Videonavn** | **Korrekte ord i transkribert tekst** | **Korrekte tegn i transkribert tekst** |
|---|---|---|
| SC-HoyFPS,3,emojis.mp4 | 76.21% | 97.40% |
| SC-HoyFPS,3,emojis-versjon2.mp4 | 59.02% | 88.56% |
| SC-MedFPS,3,emojis,skjermbilder,linker.mp4 | 84.25% | 96.31% |
| SC-NormFPS,2,emojis_skjermbilder.mp4 | 81.42% | 97.09% |
| SC-NormFPS,3,emojis.mp4 | 75.47% | 97.41% |
| Video1.mp4 | 75.06% | 92.12% |
| Video2.mp4 | 79.59% | 95.28% |
| Video3.mp4 | 83.31% | 96.08% |
| Video4.mp4 | 9.37% | 73.15% |
| Video5.mp4 | 8.79% | 70.93% |

| | | |
|---|---|---|
| Video6.mp4 | 49.33% | 83.07% |
| Video7.mp4 | 34.66% | 79.96% |
| Video8.mp4 | 35.02% | 80.28% |
| Video10.mp4 | 87.44% | 98.64% |
| Video11.mp4 | 88.31% | 97.96% |
| Video12.mp4 | 80.16% | 96.97% |
| **Gjennomsnitt verdier** | 62.96% | 90.08% |

Et viktig element for prosjektet omhandler effektivisering av arbeid ved bruk av programvaren. Det ble derfor beregnet manuell transkriberingstid før den sammenlignes med tiden brukt av programvaren. Tabell 6 viser sammenligningen på tid hvor det kommer frem at i gjennomsnitt bruker den automatiske transkriberingen 1.90% av tiden som den manuelle transkriberingen bruker. Dette viser at den automatiske transkriberingen reduserer tidsbruket til brukeren betydelig.

*Tabell 6 Oversikt over tidsbruk ved manuell og automatisk transkribering, og forholdet mellom de to*

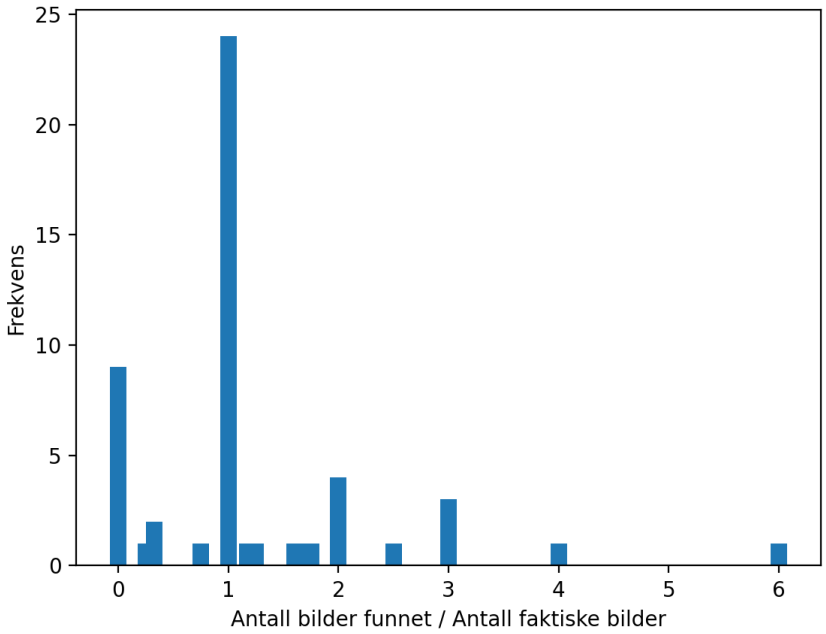
| **Videonavn** | **Manuell transkriberingstid (sekunder)** | **Automatisk transkriberingstid (sekunder)** | **Forhold (%)** |
|---|---|---|---|
| SC-HoyFPS,3,emojis | 347 | 8.91 | 2.57% |
| SC-HoyFPS,3,emojis-versjon2.mp4 | 153 | 4.38 | 2.86% |
| SC-MedFPS,3,emojis,skjermbilder,linker.mp4 | 1692 | 47.62 | 2.81% |
| SC-NormFPS,2,emojis_skjermbilder.mp4 | 417 | 12.05 | 2.89% |
| SC-NormFPS,3,emojis.mp4 | 153 | 4.37 | 2.86% |
| Video1.mp4 | 1044 | 18.41 | 1.76% |
| Video2.mp4 | 1950 | 37.19 | 1.91% |
| Video3.mp4 | 2634 | 46.53 | 1.77% |
| Video4.mp4 | 1955 | 33.22 | 1.70% |
| Video5.mp4 | 1721 | 11.79 | 0.69% |
| Video6.mp4 | 801 | 4.89 | 0.61% |
| Video7.mp4 | 788 | 9.03 | 1.15% |
| Video8.mp4 | 2430 | 29.13 | 1.20% |
| Video10.mp4 | 777 | 9.91 | 1.28% |
| Video11.mp4 | 573 | 8.81 | 1.54% |
| Video12.mp4 | 876 | 25.32 | 2.89% |
| **Gjennomsnitt** | **1144.43** | **19.4725** | **1.90%** |

### 5.2.4 Skjermbilder med søkeordet

For å teste om programvaren finner igjen ord i teksten ble det testet tre ulike ord for hver video hvor ordene varierte basert på frekvens i videoen og lengde på ordet. At ordene finnes igjen i skjermbildene avhenger av nøyaktigheten til transkriberingen og søkemetoden. På grunn av variert kvalitet på transkribering brukes et uklart ("fuzzy") regulært uttrykk. Et uklart regulært uttrykk tillater feil slik at et treff ikke trenger å være nøyaktig.

I programvaren er det definert et regulært uttrykk som tillater feil ut fra lengden til søkeordet. Antallet lovlige feil er halve lengden til søkeordet rundet ned til nærmeste heltall. Feilene som aksepteres inkluderer utbytting med andre tegn, manglende tegn og innsatte tegn.

Sammenligning mellom antallet bilder med ordet i seg og antallet bilder hvor programvaren fant ordet, vist i Figur 9, resulterer i at gjennomsnittlig blir det hentet ut flere bilder enn forventet. Gjennomsnittet på antall bilder som hentes ut ligger på 119%. Forholdet med høyest frekvens i resultatene er 100% som betyr at skjermbildene har blitt hentet ut korrekt.




*Figur 9 Stolpediagram over forhold mellom bilder hentet ut av programvaren og korrekt mengde bilder med søkeordet i*

Situasjonene hvor ingen eller for få bilder hentes ut oppstod i hovedsak for videoer med hvit tekst på mørk bakgrunn, noe programvaren ikke tar hensyn til og OCR teknologien ikke klarer å tolke på en god måte. I noen tilfeller oppstod det også grunnet dårlig transkribert tekst eller ved spesialtegn etter ordet det ble søkt etter.

Hvis søkeordet som brukes består av to eller tre bokstaver hentes det ut for mange bilder fordi endring i en bokstav på disse ordene kan føre til mange nye ord, som da vil telle som treff av det regulære uttrykket. Dette kan også oppstå ved lengre ord som “torsdag” hvor “tirsdag” også blir godkjent som et treff.

Totalt sett viser resultatene fra testingen at søkefunksjonen for det meste henter ut korrekte bilder hvis transkriberingen er god. Et forbedringspotensiale for søkefunksjonen er videre optimalisering det regulære uttrykket.

## 5.3 Prosjektresultat

Problemstillingen omhandler å automatisere transkribering fra videoer for å spare tid og ressurser. Problemet er blitt løst med hensyn på tidsbesparing, men løsningen er ikke feilfri. For å løse problemet må videoer deles opp til bilder med minst mulig overlapp og bildene må transkriberes til tekst med høyest mulig korrekthet.

Oppdeling av video til bilder som representerer det fullstendige innholdet i videoen uten noe overlapp er ikke blitt oppnådd. Det er brukt en generell metode for alle videoer og for å finne unike bilder for en mengde ulike videoer er ikke denne metoden god nok. Til tross for overlapp hentes det fullstendige innholdet ut for flertallet av videoer. Det som påvirker mangel av innhold er i hovedsak for hurtig skrolling på skjermopptaket i forbindelse med antallet bilder per sekund for videoen.

Transkriberingen gir gode resultater med høy nok korrekthet i forhold til programvaren sitt formål. I situasjon hvor tegn blir transkribert feil er det sikkerhet i søkefunksjonaliteten da denne bruker uklare regulære uttrykk. Dette fører til at søkefunksjonaliteten finner flere resultater enn forventet. Med hensyn på hvordan programvaren skal brukes er det en fordel over å hente ut for få resultater.

De viktigste kravene satt for prosjektet er blitt oppnådd til en viss grad, med tilfredsstillende kvalitet på resultater.

## 5.4 Prosjektgjennomføring

Tidligere i prosjektet ble det utført en risikoanalyse hvor flere av de noterte risikoene har medført reelle utfordringer. Den største utfordringen har vært tid. Flere oppgaver har tatt lengre tid enn forventet som har ført til tidsbegrensninger på andre oppgaver.

I starten medgikk det en del tid til å sette seg inn i teknologiene brukt i prosjektet og å søke opp feilmeldinger relatert til disse.

I risikoanalysen ble det ikke tatt høyde for en risiko hvor funksjonalitet til programmet kunne bli endret. En slik endring har vært tidskrevende og ført til omprioriteringer i sluttfasen av prosjektet. Grunnet disse endringene ble det nødvendig med nytt fokus på overlapp og duplikater. I starten av prosjektet var hovedfokuset å finne en løsning som transkriberte videomaterialet korrekt, mens de oppdaterte kravene hadde mer fokus på at videomaterialet ble transkribert tilnærmet korrekt slik at søkefunksjonaliteten kunne finne dem igjen.

Milepæler definert i Gantt diagrammet har i hovedsak blitt oppnådd til planlagt tid, med noen nødvendige justeringer underveis. Det har også vært milepæler som har blitt oppnådd før planlagt tid.

Til tross for utfordringen med tid og oppdaterte krav har det blitt jobbet jevnt med prosjektet og samarbeidet har vært en suksess. Arbeidsfordelingen har vært lik og rettferdig innenfor utvikling og skriving. I situasjoner hvor et eller flere medlemmer har av ulike grunner ikke kunne jobbet like mye som tidligere har det blitt vist gjensidig forståelse for dette.

Prosjektet har vist seg å være mer utfordrende og tidskrevende enn forventet ved prosjektoppstart som har ført til omprioritering av oppgaver og endringer i forventninger.

Oppsummert sees dette på som et godt gjennomført prosjekt.

# 6 DISKUSJON

Det viktigste kravet satt av oppdragsgiver var at programvaren skal håndtere å få inn et skjermopptak av en Snapchat samtale som skal tilnærmes korrekt transkriberes til en søkbar tekst. Initielle krav inkluderte også håndtering av vanlig brukte forkortelser og emojier. Programvaren skulle kunne brukes av flere mennesker uavhengig av teknisk bakgrunn.

For å løse dette ble det valgt å dele utviklingen av programvaren i oppdeling av video til bilder og transkribering av bilder. Disse to delene var hovedfokus i starten av prosjektet. Det ble raskt oppdaget at å dele en video opp til unike bilder er en utfordrende oppgave som tok lengre tid enn forventet. Etter ulike tilnærminger ble det valgt å bruke perseptuell hashing for å sammenligne bilder for å redusere antall bilder brukt i transkriberingen. Transkriberingen tok kortere tid, med noe mer fokus på optimalisering.

Det ble brukt Python pakker basert på åpen kildekode som er godt dokumentert med hensyn på funksjonalitet og testing.

I starten av prosjektet ble det lagt en grundig plan for prosjektarbeidet. I løpet av prosjekttiden har arbeidet utviklet seg i samsvar med denne, med noen avvik. Avvikene er relatert til begrensninger i tid, grunnet feil tidsprioritering i perioder og utfordringer i arbeidsoppgaver.

Grunnet mye fokus på å perfeksjonere oppdeling av video til bilder ble det mindre tid til andre ønskede funksjonaliteter som håndtering av forkortelser og emojier. Til slutt ble det akseptert at løsningen ikke fungerte perfekt og at det endelige produktet ville ha forbedringspotensialer. Samtidig ble det forsøkt å oppnå ekstra funksjonaliteten hvor noen av oppgavene ble for utfordrende å få til, som håndtering av emojier.

Oppgaven definert ut fra problemstillingen ble ikke fullstendig oppnådd siden den endelige prosessen ikke er en komplett automatisert løsning. Det kreves noe brukerinteraksjon og resultatene er ikke uten feil. Det som er blitt oppnådd er en løsning som oppfyller de viktigste kravene uten avvik som ødelegger funksjonalitet.

Sluttproduktet er en programvare med brukerinteraksjon kombinert med automatiserte løsninger som løser kjernen i problemstillingen. Ut fra resultatene vil programvaren spare tid og ressurser hos oppdragsgiver som ut fra brukertesting var fornøyd med løsningen. Løsningen krever at brukeren må gjøre mer enn opprinnelig planlagt, til tross for dette vil bruk av programvaren være mer effektivt enn å manuelt utføre arbeidsoppgavene.

# 7 KONKLUSJON OG VIDERE ARBEID

## 7.1 Konklusjon

De initielle målene satt for prosjektet er nådd til en viss grad. Det har blitt utført endringer i funksjonen til programvaren sent i prosjektet og noen funksjonaliteter har ikke blitt lagt til.

Oppgaven å dele opp til unike bilder var utfordrerne og ble ikke oppnådd i løpet av prosjektet på en fullstendig måte, men det tilfredsstiller fremdeles kravene og gir funksjonalitet som har verdi for oppdragsgiver.

Transkriberingsmetoden gir høy korrekthet med hensyn på tegn. Dette kombinert søkefunksjonaliteten sin bruk av uklare regulære uttrykk gir tilfredsstillende resultater for transkribering og søk.

Automatisk transkribering øker effektiviteten med hensyn på tid i stor grad, noe som var et viktig fokus hos oppdragsgiver. Oppdragsgiver kom med positive tilbakemeldinger på brukertestingen og et innblikk i hvorfor programvaren ville være til hjelp, til tross for at programvaren ikke gir perfekte resultater.

Programvaren som har blitt utviklet har begrensninger og har potensiale for videre arbeid. Om programvaren brukes med forbehold om avgrensninger, vil den fungere på en tilfredsstillende måte med hensyn til formålet.

## 7.2 Videre arbeid

Programvaren kan forbedres ved optimalisering av eksisterende teknikker og utvidet funksjonalitet. Teknikken som brukes for oppdeling av video til unike skjermbilder gir overlapp og i noen situasjoner manglende informasjon. Kvaliteten varierer basert på flere av videoens parametere. Ved å gjøre metoden mer dynamisk slik at den jobber basert på disse parameterne kan det være mulig å hente ut skjermbilder med mindre overlapp.

Transkriberingen har gode resultater, men disse kan forbedres. Før OCR teknologien anvendes preprosesseres skjermbildene på en delvis dynamisk måte basert på bildene. Denne prosessen kan gjøres mer dynamisk for at hvert bilde som hentes ut preprosesseres på en ideell måte.

Søkefunksjonaliteten gir tilfredsstillende resultater, men har enda forbedringspotensiale med hensyn på det uklare regulære uttrykket.

Inkludering av ekstra funksjonalitet kan gi mer eksakte resultater. Ved å inkludere håndtering av emojier i transkriberingsprosessen vil resultatene representere mer kontekst til den transkriberte teksten, og det kan være mulig å utføre søk på emojier.

Språket i videoen er satt til norsk og programvaren fungerer da på språk som bruker det latinske alfabetet. Ved å legge inn funksjonalitet for å velge språk, eller automatisk kjenne det igjen, er det mulig å transkribere videoer hvor samtalen bruker andre alfabeter.

I mange situasjoner sendes det bilder, animasjoner eller videoer i samtaler. Det er mulig å legge inn funksjonalitet for å hente disse elementene ut.

Skjermopptak i mørk modus gir dårlige resultater slik programvaren er nå. Ved å kjenne igjen når videoer er i mørk og lys modus er det mulig å invertere bildene i mørk modus slik at de er på rett format for transkriberingen.

# 8 REFERANSER


Braaten, M., & Fossheim, K. (2021, April 21). *Styrte hashbutikk fra Snapchat: "Ny topp godis inne"*. Hentet fra TV2 Nyheter: https://www.tv2.no/a/13952332/

Cockburn, A., & Highsmith, J. (2001, Desember). Agile software development, the people factor. *IEEE Computer*(34(11)), ss. 131-133.

Drmic, A., Silic, M., Delac, G., Vladimir, K., & Kurdija, A. S. (2017, Juli 13). Evaluating robustness of perceptual image hashing algorithms. *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, ss. 995-1000. Hentet fra https://ieeexplore.ieee.org/document/7973569

Elgato. (2022, Februar 22). *HD60 S+*. Hentet fra Elgato: https://www.elgato.com/en/game-capture-hd60-s-plus

Eliassen, H. (2021, Oktober 18). *Tiltalte: Vervet mindreårige jenter til sexsalg via Snapchat*. Hentet fra TV2 Nyheter: https://www.tv2.no/a/14293293/

Ipsos. (2022, Januar 25). *Ipsos SoMe-tracker Q4'21*. Hentet fra Ipsos: https://www.ipsos.com/nb-no/ipsos-some-tracker-q421

Mithe, R., Indalkar, S., & Divekar, N. (2013, Mars). Optical Character Recognition. *International Journal of Recent Technology and Engineering*(2-1), ss. 72-75.

PyPI. (2022, Februar 19). *pytesseract 0.3.9*. Hentet fra Python Package Index: https://pypi.org/project/pytesseract/

Røgeberg, O. (2018, August 31). *Fire av fem nordmenn bruker sosiale medier*. Hentet fra Statistisk sentralbyrå: https://www.ssb.no/teknologi-og-innovasjon/artikler-og-publikasjoner/fire-av-fem-nordmenn-bruker-sosiale-medier

Stranden, A. L. (2021, Juni 10). *Mer kriminalitet har blitt digital: - Etterforskere trenger ikke kunne svømme, løpe eller løfte tunge vekter*. Hentet fra forskning.no: https://forskning.no/kriminalitet-politi-utdanning/mer-kriminalitet-har-blitt-digital--etterforskere-trenger-ikke-kunne-svomme-lope-eller-lofte-tunge-vekter/1872330

TypingPal. (2022). *Typing Speed*. Hentet fra Typing Pal: https://www.typingpal.com/en/documentation/school-edition/pedagogical-resources/typing-speed

Yadid, S., & Yahav, E. (2016, Oktober 20). Extracting code from programming tutorial videos. *Association for Computing Machinery: Onward! 2016*, ss. 98-111.

Zhong, S., Liping, C., & Tian-en, C. (2011, Mai 5). Agile planning and development methods. *2011 3rd International Conference on Computer Research and Development*(1), ss. 488-491.

# 9 VEDLEGG

## 9.1 Kodebibliotek brukt i programvaren

Oversikt over kodebibliotek brukt i programvaren som krever installasjon, hentet fra Systemdokumentasjon.

| Bibliotek | Beskrivelse | Link til dokumentasjon |
|---|---|---|
| fpdf | Pakke som gir muligheter for generering av PDF-filer. | https://pyfpdf.readthedocs.io/en/latest/#fpdf-for-python |
| scikit-image | Pakke med en samling algoritmer for bildeprosessering. | https://scikit-image.org/docs/stable/ |
| imutils | Pakke med en samling funksjoner for enkel bildeprosessering. | https://github.com/PyImageSearch/imutils |
| pillow | Pakke med funksjoner for bildeprosessering. | https://pillow.readthedocs.io/en/stable/ |
| regex | Pakke med ekstra funksjonalitet relatert til regulære uttrykk. | https://pypi.org/project/regex/ |
| fonttools | Pakke for håndtering av skrifttyper. | https://pypi.org/project/fonttools/ |
| ffmpeg-python | Pakke for jobbing med mediefiler. | https://kkroening.github.io/ffmpeg-python/ |
| ImageHash | Pakke for å hashe bilder. | https://openbase.com/python/ImageHash/documentation |
| Image | En del av pillow rammeverket. | https://pillow.readthedocs.io/en/stable/reference/Image.html |
| photohash | Pakke for hashing av bilder. | https://github.com/bunchesofdonald/photohash |
| numpy | Pakke for jobbing med numeriske verdier i en- eller fler-dimensjonale lister. | https://numpy.org/doc/ |
| checksumdir | Pakke for hashing av mapper. | https://pypi.org/project/checksumdir/ |

## 9.2 Gantt-diagram

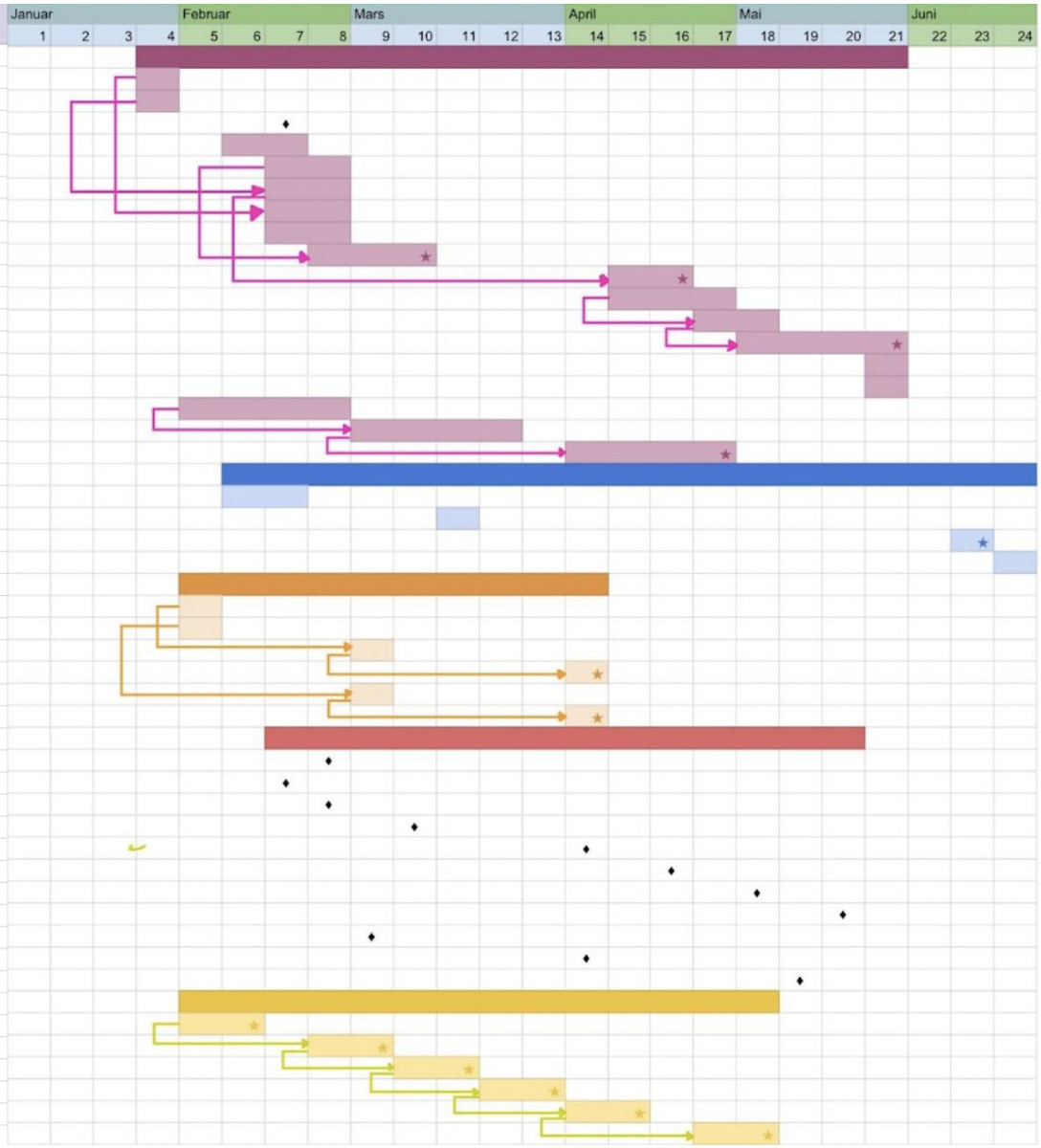
Det er definert 5 ulike faser som består av oppgaver relatert til fasen sitt tema. Den første fasen er dokumentasjon hvor det planlagte arbeidet er relatert til dokumenter som skal leveres i løpet av prosjektarbeidet og deres iterasjoner. Gantt diagrammet har også en fase for presentasjoner, som inkluderer alle obligatoriske presentasjoner som skal gjennomføres av prosjektgruppen.

Arkitektur fasen dekker planlegging og strukturering av produktet som skal utvikles. Dette er forarbeid til utviklingen av produktet som er en egen fase i diagrammet. Møter med veileder og oppdragsgiver spiller en sentral rolle under planlegging og utvikling av produkt da prosjektgruppen er avhengig av tilbakemelding fra sistnevnte for å forsikre at produktet møter ønskede krav. Gantt diagrammet inneholder en egen fase som omfatter disse møtene.

Utviklingsfasen inneholder seks iterasjoner og under hver iterasjon ferdigstilles en ny versjon av programvaren. Utviklingsarbeidet er delt opp i iterasjoner for å sikre tilbakemelding fra oppdragsgiver på hver utviklings iterasjon, slik at produktet kan tilpasses eventuelle tilbakemeldinger før kompleksiteten økes i videre iterasjoner. Før siste iterasjon (Iterasjon 6) er det ønsket å gjennomføre en brukertesting hos oppdragsgiver slik at iterasjon 6 har fokus på forbedringer.

Det er tretten milepæler som til sammen beskriver viktige hendelser i prosjektforløpet. Milepælene i dokumentasjons fasen er fullføringen av de sentrale dokumentene; forprosjektrapport, prosjekthåndbok og endelig rapport. I presentasjonsfasen er det en milepæl som omfatter den endelige presentasjonen til gruppen da dette er den siste delen av fasen. Arkitektur fasen har milepæler som beskriver ferdigstilling av GUI prototypen og arkitekturmodeller. Under utviklingsfasen vil hver iterasjon medføre en ny programversjon som vil være et steg nærmere en endelig løsning, dermed er hver iterasjon en milepæl.

| ID | Fullført | Oppgave navn | Varighet | Start |
|---|---|---|---|---|
| 1 | | **Dokumentasjon** | | |
| 2 | ☑ | Visjonsdokument it. 1 | 3 dager | 25. januar |
| 3 | ☑ | Prosjekthåndbok it. 1 | 3 dager | 25. januar |
| 4 | ☑ | Veilederkontrakt | 1 dag | 16. februar |
| 5 | ☑ | Analyse av tidligere bacheloroppgaver | 9 dager | 08. februar |
| 6 | ☑ | Forprosjektrapport it. 1 (ATV) | 10 dager | 17. februar |
| 7 | ☑ | Prosjekthåndbok it. 2 (V) | 10 dager | 17. februar |
| 8 | ☑ | Visjonsdokument it. 2 (A) | 10 dager | 17. februar |
| 9 | ☑ | Kravspesifikasjon it. 1 (T) | 10 dager | 17. februar |
| 10 | ☑ | Forprosjektrapport it. 2 | 12 dager | 27. februar |
| 11 | ☑ | Prosjekthåndbok it. 3 | 10 dager | 12. April |
| 12 | ☑ | Rapport utkast it. 1 | 14 dager | 11. April |
| 13 | ☑ | Rapport utkast it. 2 | 14 dager | 22. April |
| 14 | ☑ | Endelig rapport | 21 dager | 02. Mai |
| 15 | ☐ | Refleksjonsnotat | 4 dager | 20. Mai |
| 16 | ☐ | EXPO plakat | 6 dager | 23. Mai |
| 17 | ☑ | Systemdokumentasjon it. 1 | 28 dager | 01. Februar |
| 18 | ☑ | Systemdokumentasjon it. 2 | 28 dager | 01. Mars |
| 19 | ☐ | Systemdokumentasjon it. 3 | 28 dager | 01. April |
| 20 | | **Presentasjon** | | |
| 21 | ☑ | Analyse av tidligere bacheloroppgaver | 9 dager | 08. februar |
| 22 | ☑ | Forprosjekt | 7 dager | 08. mars |
| 23 | ☐ | Endelig presentasjon | 7 dager | 31. mai |
| 24 | ☐ | EXPO Bergen | 1 dag | 15. juni |
| 25 | | **Arkitektur** | | |
| 26 | ☑ | Modeller it. 1 | 3 dager | 01. februar |
| 27 | ☑ | GUI prototype it. 1 | 3 dager | 01. februar |
| 28 | ☑ | Modeller it. 2 | 3 dager | 01. mars |
| 29 | ☑ | Modeller it. 3 | 3 dager | 01. april |
| 30 | ☑ | GUI prototype it. 2 | 3 dager | 01. mars |
| 31 | ☑ | GUI prototype it. 3 | 3 dager | 01. april |
| 32 | | **Møter** | | |
| 33 | ☑ | Møte med oppdragsgiver 1 | 1 dag | 22. februar |
| 34 | ☑ | Statusmøte 1 | 1 dag | 15. februar |
| 35 | ☑ | Statusmøte 2 | 1 dag | 24. februar |
| 36 | ☑ | Møte med oppdragsgiver 2 | 1 dag | 08. mars |
| 38 | ☑ | Møte med oppdragsgiver 3 | 1 dag | 06. April |
| 39 | ☑ | Møte med oppdragsgiver 4 | 1 dag | 19. April |
| 40 | ☑ | Møte med oppdragsgiver 5 | 1 dag | 03. mai |
| 41 | ☑ | Møte med oppdragsgiver 6 | 1 dag | 19. mai |
| 42 | ☑ | Statusmøte 3 | 1 dag | 04. mars |
| 43 | ☑ | Statusmøte 4 | 1 dag | 08. April |
| 44 | ☑ | Statusmøte 5 | 1 dag | 12. mai |
| 45 | | **Utvikling** | | |
| 46 | ☑ | Iterasjon 1 | 15 dager | 01. februar |
| 47 | ☑ | Iterasjon 2 | 15 dager | 21. februar |
| 48 | ☑ | Iterasjon 3 | 15 dager | 08. mars |
| 49 | ☑ | Iterasjon 4 | 15 dager | 24. mars |
| 50 | ☑ | Iterasjon 5 | 15 dager | 09. April |
| 51 | ☑ | Iterasjon 6 | 15 dager | 25. April |



## 9.3 Risikoanalyse

| | Hendelse /Risiko | Årsak | Sannsynlighet | Konsekvens | Risiko produkt | Tiltak |
|---|---|---|---|---|---|---|
| 1 | Problemer med å lese av tekst | Mye variasjon i skrifttype.<br>Bruk av emojier.<br>Bilder i chat.<br>Dårlig oppløsning på video. | Høy (4) | Middels (3) | 12 | Funksjoner som håndterer variasjoner og unntak.<br>Bedre programvare for å ta skjermopptak. |
| 2 | Ulike språk på videoene | Ulike språk innstillinger på telefoner. | Svært Høy (5) | Middels (3) | 15 | Tilpasse koden til å akseptere flere alfabet.<br>Se etter kode som gjenkjenner språk ut fra tekst. |
| 3 | Forkortelser som ikke kjennes igjen | Hvis forkortelser ikke er i ordboken | Middels (3) | Lav (2) | 6 | Oppdatere ordboken med nye forkortelser.<br>Finne bedre oversikt over vanlige forkortelser. |
| 4 | Tekst blir tildelt til feil forfatter | Hvis tekst blir lagret til feil forfatter | Middels (3) | Høy (4) | 12 | Lagring av navn i egen struktur, slik at de kan kjennes igjen.<br>Finne ut hva som forårsaker problemet og løse det. |
| 5 | Fjerning av emojier | Transkribering inneholder ikke emojier brukt i originaltekst. | Middels (3) | Høy (4) | 12 | Transkribert tekst mister deler av kontekst. |
| 6 | Dårlig oppfølging fra oppdragsgiver | Oppdragsgiver har ikke tilstrekkelig med tid til oppfølgingsmøter | Middels (3) | Svært høy (5) | 15 | God kommunikasjon mellom partene og planlegging. |
| 7 | Fravær i prosjektgruppen | Sykdom, jobb eller andre personlige hendelser. | Middels (3) | Middels (3) | 9 | Jevn jobbing gjennom hele prosjektet og planlegging. |

| | | | | | | |
|---|---|---|---|---|---|---|
| 8 | Gruppen fullfører ikke obligatoriske oppgaver innen tidsfrist. | Fravær i prosjektgruppen eller andre interne problemer. | Lav (2) | Svært høy (5) | 10 | Starte å arbeide med oppgavene tidlig og planlegge med hensyn på tidsfrister. |