Høgskulen på Vestlandet

BACHELOROPPGAVE

Programvare for fremstilling av lungetest-data på Haukeland Universitetssykehus

Software for collecting and presenting lung test data at Haukeland University Hospital

Amanda Midttun Søreide, Fredrik Fidjestøl Mathisen

Dataingeniør/informasjonsteknologi
Institutt for data- og realfag

Fakultet for ingeniør- og naturvitenskap
Antall ord: 11 301
Innleveringsdato: 03.06.19

Jeg bekrefter at arbeidet er selvstendig utarbeidet, og at referanser/kildehenvisninger til alle kilder som er brukt i arbeidet er oppgitt, *jf. Forskrift om studium og eksamen ved Høgskulen på Vestlandet, § 10.*

TITTELSIDE FOR HOVEDPROSJEKT

Rapportens tittel: Programvare for fremstilling av lungetest-data på Haukeland universitetssykehus	*Dato: 03.06.2019*
Forfatter(e): Fredrik Fidjestøl Mathisen, Amanda Midttun Søreide	*Antall sider u/vedlegg: 49* *Antall sider vedlegg: 5*
Studieretning: Dataingeniør/Informasjonsteknologi	*Antall disketter/CD-er: ingen*
Kontaktperson ved studieretning: Sven-Olai Høyland	*Gradering: ingen*
Merknader: ingen	

Oppdragsgiver: Haukeland universitetssykehus	*Oppdragsgivers referanse:*
Oppdragsgivers kontaktperson: Lars Peder Bovim Ola Drange Røksund	*Telefon:* +47 976 82 402 +47 55 58 55 79

Sammendrag:

Bachelorprosjektet går ut på å utvikle en programvare for fremstilling og innhenting av data fra en lungetest (CLE-test). Programvaren skal brukes til medisinsk forskning og diagnostisering på Energisenteret for barn og unge ved Haukeland Universitetssykehus. Løsningen er en Windows-applikasjon som skal gjenspeile eksisterende løsning, men med ytterlig tilleggsfunksjonalitet samt forbedret brukergrensesnitt. Den nye løsningen åpner også for et større bruksområde innenfor medisinske undersøkelser og forskning.

Stikkord:

Programvare	Medisinsk utstyr	.Net
Fullstack utvikling	CLE-test	FFmpeg, FFME

Høgskulen på Vestlandet, Fakultet for ingeniør- og naturvitskap

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 rapporten dokumenterer bachelorprosjektet "Programvare for fremstilling av lungetest-data på Haukeland universitetssykehus". Denne rapporten beskriver prosessen for å utvikle en programvare for innhenting og fremstilling av data fra en lungetest på Haukeland universitetssykehus. Prosjektet ble gjennomført av Amanda Midttun Søreide og Fredrik Mathisen våren 2019.

Vi ønsker å rette en stor takk til prosjekteier for å ha gitt oss en veldig spennende, viktig og utfordrende oppgave. Stor takk til Lars Peder Bovim, Ola Røksund, Hege Clemm, Victoria Isern og resten av teamet på Haukeland for et innblikk i krysningspunktet mellom medisin og teknologi, samt motivasjon gjennom hele prosjektarbeidet.

Vi ønsker også å takke vår interne veileder ved Høgskulen på Vestlandet, Sven-Olai Høyland, for gode råd og veiledning gjennom hele prosjektperioden.

En siste takk rettes til Steinar Søreide for alle tekniske innspill og veiledning.

Innholdsfortegnelse

1 INTRODUKSJON

Dette kapittelet gir en introduksjon for bakgrunnen til prosjektet. I kapitlene under vil problemstilling, mål og motivasjon, kontekst for prosjektet, begrensninger og ressurser bli gått gjennom nærmere.

1.1 Mål og motivasjon

1.1.1 Bakgrunn

Mange kan oppleve pusteproblemer under hard trening. Dette kan gjenkjennes ved høyfrekvente pustelyder, kortpustethet og følelse av at halsen tetter seg. Mange av de som oppsøker lege kan bli diagnostisert med anstrengelsesastma og dermed få medisiner mot dette. Det er likevel mange som blir feildiagnostisert. Exercised Induced Laryngeal Obstruction (EILO) og astma har mange likhetstrekk, men astmamedisin har ikke har noen effekt på EILO. Strupehodet fungerer som en flaskehals for luftveien hvis man har EILO. På Haukeland Universitetssykehus har de utviklet og patentert en test for å diagnostisere EILO. Denne testen heter Continuous Laryngoscopy Excercise Test (CLE) og er en maksbelastningstest på tredemølle.

1.1.2 Problemstilling

Den nåværende applikasjonen de bruker for å fremstille og lagre data fra CLE-tester på Haukeland fungerer ikke optimalt. Videostrømmene har dårligere kvalitet enn hva utstyret tilsier. Det skal også settes en forsinkelse på noen av videostrømmene ettersom den ene videostrømmen allerede kommer med cirka 0,7 sekunder forsinkelse. Dette er ikke løst på en god nok måte, og dermed er ikke videostrømmene synkroniserte. I tillegg er brukervennligheten i applikasjonen dårlig, og en del tilleggsfunksjonalitet mangler. Haukeland ønsker å kunne trekke inn rådata fra alle de aktuelle kildene som brukes i

dagens oppsett. Oppdragsgiver etterspør et nytt og mer stabilt innsamlingsoppsett enn det som eksisterer i dag.

1.1.3 Mål

Målet for dette prosjektet er å utvikle en ny og forbedret applikasjon som Haukeland kan bruke. I første omgang vil vi jobbe mot å lage en applikasjon som har samme funksjonalitet som den eksisterende løsningen, men med forbedret brukervennlighet. Dersom vi får tid til å utvikle applikasjonen videre utover dette, vil vi holde et nytt møte sammen med de involverte fra Haukeland for å diskutere hvilken funksjonalitet vi skal prioritere å implementere.

Data som blir samlet inn blir ikke bare brukt til å diagnostisere pasienter, men også til et forskningsprosjekt som omhandler EILO. Haukeland ønsker å bruke den nye løsningen til å forbedre nøyaktigheten på forskningsdata som blir samlet inn. Ved å levere et fungerende produkt vil vi bidra til at Haukeland kan samle inn mer presise forskningsdata, noe som igjen vil bidra til økt kunnskap rundt denne kroniske tilstanden. Vi vil få et innblikk i krysningspunktet mellom klinikere, leverandører og driftsteknisk personell. En annen motivasjonsfaktor i dette prosjektet er læringspotensialet. Ved å utvikle en applikasjon fra start til slutt er vi med på å utvikle hele teknologistacken, noe vi sjeldent har fått gjøre før. Haukeland ønsker å distribuere programvaren når den er ferdigstilt. Dette vil bli gjort i samarbeid med Olympus, som er en produsent av medisinsk utstyr. Disse to faktorene bidrar til at motivasjonen for å lykkes er stor.

1.2 Begrensninger

Det er naturlig at det er begrensninger ved et prosjekt som dette. Dette grunnet begrenset med tid for å gjennomføre prosjektet. Gruppen hadde ikke jobbet med multimedieelementer på denne måten før. Dette førte til at det måtte brukes en del tid på å sette seg inn i teknologien som skulle brukes. En annen ting som også var problematisk var tidspunktene testlaben på Haukeland er i bruk. Disse punktene vil bli beskrevet nærmere i punktene nedenfor.

1.2.1 Bruk av ny teknologi

Som tidligere nevnt hadde ingen på gruppen jobbet spesielt mye med håndtering av multimedieelementer tidligere. Det ble brukt mye tid i startfasen av prosjektet til å lese oss opp på nye teknologier og rammeverk for å øke forståelsen rundt dette. Dette medførte også en del prøving og feiling før vi begynte å skjønne hvordan videostrømmene ble behandlet. Ettersom rammeverket som ble tatt i bruk snakker med kommandolinjen, måtte vi sette oss inn i hvordan disse kommandoene kan sendes på tvers av programmer. Alt dette medførte at utviklingen av applikasjonen tok mer tid i starten enn forventet. Haukeland var tidlig ute og sa at de mest sannsynlig ikke kunne tilby tilstrekkelig teknisk kompetanse, så dette var vi klar over helt fra prosjektstart.

1.2.2 Testlaben

For å kunne dokumentere at utviklingen gikk riktig vei var gruppen avhengige av å kunne teste applikasjonen på Haukeland sin PC. Denne PC-en står i kjelleren i en lab på Energisenteret for barn og unge på Haukeland. PC-en blir brukt til å dokumentere CLE-tester på pasienter. Dette medførte restriksjoner på når applikasjonen kunne testes på det faktiske systemet den skal brukes på. Et annet problem gruppen støtte på med denne datamaskinen var at den tidvis ikke hadde tilgang til internett. Dette medførte problemer i forbindelse med synkronisering av koden vi hadde liggende på GitHub. Vi løste problemet med å føre hele prosjektet over på minnepenn da vi gjennomførte endringer på koden. Prosjektet ble så manuelt lagt inn på PC-en på laben.

1.3 Ressurser

For å kunne gjennomføre dette prosjektet var vi avhengig av å bruke mange forskjellige verktøy. For utvikling av selve applikasjonen brukte vi C#. FFMPEG og FFME ble brukt for å kunne lese og fremstille de forskjellige multimedia strømmene. Siden utviklingen av koden ville foregå på forskjellige maskiner var det viktig å kunne dele koden lett og

effektivt. Det ble derfor bestemt at Git skulle brukes som versjonskontrollverktøy. For utviklingen av selve koden ble Visual Studio brukt. Når det kom til deling av dokumenter ble Google Drive brukt. Databasen var SQL Server Express. Disse verktøyene vil bli gjennomgått i mer detalj i kapittel 3.3.

Videre var programmet avhengig av å ha en del forskjellige kameraer for testing. Til dette lånte gruppen en del webkameraer, brukte de innebygde webkameraene og testet med det faktiske utstyret på CLE-laben. Det ble også testet å få opp skjermbilde av en ekstern datamaskin, ved å koble den til datamaskinen på laben gjennom en HDMI-kabel og en splitter.

Det var også viktig å ha tilgang til datamaskinen på CLE-laben for testing, ettersom det er der programvaren i hovedsak er ment til å kjøre. En del elementer i programvaren er hardkodet etter datamaskinen på CLE-laben.

1.4 Organisering av rapporten

Rapporten er organisert på følgende måte:

Kapittel 1: En introduksjon til prosjektet.

Kapittel 2: En mer detaljert beskrivelse av prosjektet.

Kapittel 3: Prosjektets design, samt diskusjon rundt alternative prosjektløsninger.

Kapittel 4: En detaljert beskrivelse av løsningen som ble brukt i prosjektet.

Kapittel 5: Evaluering av prosjektet og resultatet av evalueringen.

Kapittel 6: Diskusjon rundt resultatet sett opp mot prosjektets mål.

Kapittel 7: Rapportens konklusjon

Kapittel 8: En oversikt over all litteratur som har blitt brukt for å komme frem til svarene I oppgaven.

Kapittel 9: Vedlegg

2 PROSJEKTBESKRIVELSE

2.1 Praktisk bakgrunn

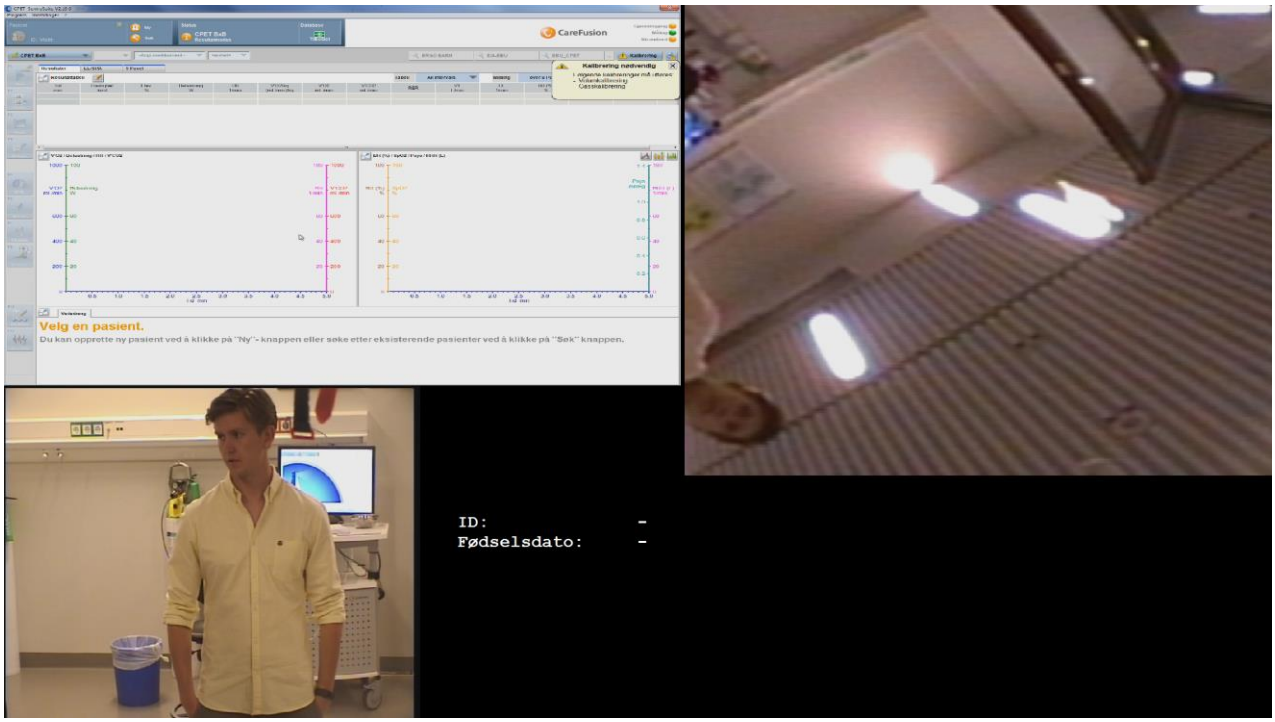
2.1.1 Prosjekteier

Prosjektgruppen har ingen eierskap knyttet til produktløsningen. Prosjektet er eid av EILO gruppen Bergen. EILO gruppen Bergen er den del av West Pead Research. Dette er en forskningsgruppe som ble startet i 2009 av Trond Merkestad. De arbeider med forskning rundt vekst-, hjerte- og luftveisfysiologi for barn og ungdom. Forskningsgruppen består av leger, sykepleiere og fysioterapeuter på Haukeland Universitetssykehus. De fokuserer også på oppfølgingsstudier av barn som er født for tidlig (Westpead, u. å).

Prosjektet har også mottatt støtte fra Vestlandets Innovasjonsselskap AS (VIS). VIS er det nye navnet til Bergen teknologioverføring (BTO) som ble startet i 2004. VIS er en organisasjon som ønsker å hjelpe bedrifter, studenter, forskere og gründere med å utvikle ideer med innovativt potensiale. Ideene må være levedyktig i fremtiden, og tilføre samfunnet nytteverdi i form av tjenester, produkter eller prosesser. Selskapet er eid av en rekke institusjoner med base på Vestlandet, blant annet Helse Bergen, HVL, UiB og NORCE (Christian Michelsen Research / Uni Research). Til sammen har institusjonene 4000 forskningsårsverk og en forskningsbase på 5 milliarder NOK. (VIS innovasjon, u. å).

2.1.2 Tidligere arbeid

Programmet Haukeland bruker til CLE-testing nå, er laget av Profitek AS, et selskap som ikke eksisterer lenger. Den nåværende løsningen er et program som viser tre forskjellige videostrømmer (Figur 2-1). Brukerne har også mulighet for å legge inn pasientinformasjon til de separate innspillingsøktene. Programmet spiller også inn et skjermopptak av det som vises på skjermen for å kunne lagre videoene og pasientinfo i én og samme fil. Det er også mulig å legge inn en forsinkelse på skop-kameraet (til høyre) og portrett-kameraet (nede) i den eksisterende programvaren.



Figur 2-1 Skjermbilde fra eksisterende programvare

Det er ingen indikasjon på om forsinkelsen er skrudd av eller på under kjøring. Dette har ført til at innsamlet data ikke har vært synkronisert. Haukeland har da løst dette ved å redigere filmene i ettertid, men med varierende resultat. Videre lagres opptakene i Haukelands database. Det er ingen mulighet for innlogging, og ingen logging på hvem som har gjort undersøkelsen og hvem som har sett på videoopptakene i ettertid.

2.1.3 Kravspesifikasjon

I den opprinnelige kravspesifikasjonen som står i oppgavebeskrivelsen stod det at gruppen bare skulle utvikle et program med helt lik funksjonalitet som det nåværende programmet. Etterhvert ble det sett at det faktisk var tid og mulighet for å legge inn ekstra funksjonalitet, som da ble lagt til i den offisielle kravspesifikasjonen. Disse tilleggsfunksjonalitetene ble brukstilfelle 2 og 3, som er beskrevet mer nøyaktig i kapittel 4.1.2 og 4.1.3. Den var da at brukeren av programvaren skal kunne gå inn og se ferdige innspilte CLE-tester, og å kunne bestemme et antall videostrømmer som skal vises i programmet.

Oppdraget ble dermed å lage en forbedret versjon av det nåværende programmet, men med ny tilleggsfunksjonalitet. Programmet skulle kunne ta inn et variabelt antall videosignal og et lydsignal, og da vise disse lyd- og videostrømmene synkronisert. Brukergrensesnittet skulle også lages slik at brukeren kan endre størrelsen på vinduene som viser videoene. Brukeren skal videre kunne legge inn mer info per pasient. Denne infoen består av fødselsnummer, kjønn, forskningsnummer og om det er en lege eller annet personell som gjennomfører undersøkelsen.

2.2 Litteratur bakgrunn

Når data brukes til forskning er det ekstremt viktig at dataa er korrekt. For å sikre at data er korrekt, er det viktig med et presist og intuitivt innsamlingsoppsett. Heimdal (2009) forklarer at alle data som blir samlet inn av programmet blir vurdert i ettertid av gjennomført test. Det er derfor viktig at kamerakildene er synkroniserte. Hvis videostrømmene ikke er synkrone vil det bli vanskeligere å oppdage symptomene som gjenkjenner EILO. Dette problemet kan også løses med redigering av filmene. Dette er som regel tidkrevende og resultatene er varierende.

Innsamlingsoppsettet må også være brukervennlig for å forhindre brukeren å gjøre feil, som i verste fall kan gjøre data korrupt. Masteroppgaven “User Interface design - Methods and qualities of a good user interface design” skrevet av Ravi Chandra Chaytania Guntupalli (2008) ved Högskolan Väst i Trollhättan presenterer syv punkter for å måle kvaliteten på en programvare.

- **Pålitelighet** - Programvaren skal oppføre seg likt hver gang det kjøres.
- **Effektivitet** - Et mål på hvor godt programvaren deler de tilgjengelige ressursene.
- **Knapphet** - Programvaren skal ikke gjøre eller vise mer enn det er laget for.
- **Fleksibilitet** - Programvaren skal ikke være låst til et system.
- **Sammenhengende** - Utseende på forskjellige sider i programvaren skal følge en uniform mal.
- **Vedlikeholdbarhet** - Programvaren skal fungere selv om hardware og software oppdateres.

- **Forståelighet** - Programmet skal være lett å håndtere for sluttbrukeren.

3 PROSJEKTDESIGN

3.1 Mulige løsninger

Det første steget i prosjektet var å velge verktøy. Gruppen startet med valg av kodespråk. Ettersom det som skulle produseres var en Windows-applikasjon var C# et naturlig valg. Det ble også bestemt at Windows Presentation Foundation (WPF) skulle brukes. Valget falt på WPF fordi det gjør jobben med å lage brukergrensesnitt lettere. Det egner seg også svært godt å bruke delegatfunksjoner med videostrømming.

Det ble fort klart at Windows DirectShow skulle brukes for videostrømmingen. Det er tilgjengelig i Windows SDK, og er det mest brukte rammeverket for mediastrømming innen Windows-applikasjoner. I tillegg til dette støtter de fleste enheter DirectShow, og det var dermed det mest logiske verktøyet å ta i bruk. For å bruke DirectShow er det mange forskjellige alternative tilnærminger når det kommer til bruk av bibliotek, programmer og verktøy.

3.1.1 Alternativ løsning 1: C# wrappers for DirectShow

I utgangspunktet var planen å bruke DirectShow uten hjelp fra andre kodebiblioteker , for å få alt av videocapture og lagring til å fungere så optimalt som mulig. Det ble dermed lastet ned en wrapper for DirectShow i C# slik at programmet fikk tilgang til funksjonaliteten til DirectShow i C# kode. På denne måten har gruppen hatt stor kontroll over DirectShow på et lavt nivå.

Etter mye utprøving og lesing av wrapper-koden ble det konkludert med at dette ble for tungvint måte å gjøre ting på, og gruppen bestemte seg for å lete etter verktøy som bygger på DirectShow. Ettersom det relativt sett er ganske generelle ting som skal gjøres med DirectShow, ansees det ikke som nødvendig å ha så detaljert kontroll over softwaren. Siden tiden for prosjektet er begrenset, var det mer strategisk å bruke ferdige bibliotek.

3.1.2 Alternativ løsning 2: FFmpeg og FFME

Etter en del undersøkelse ble FFmpeg oppdaget. FFmpeg er et gratis software-prosjekt som inneholder bibliotek og programmer for å håndtere mediefiler og strømmer. Det viste seg å være av god kvalitet, og enkelt å bruke etter at gruppen hadde brukt en del dager på å sette seg inn i kommandoene. Det ble valgt til å brukes for lagring, strømming og konvertering av video og lyd både rett fra kamerakilder og lokale filer.

For å spille av video og lyd i applikasjonen, ble det lastet ned et kodebibliotek kalt Fast Forward Media Element (FFME). FFME er skrevet for å brukes i C#, og bruker FFmpeg for å lese og dekode mediastrømmer. Ulempen ved FFME er at den har en standard forsinkelse på 0,5 sekunder når man spiller av video uansett kilde. Dermed finnes det muligens bedre verktøy for å vise video i C#, men det viste seg vanskelig å finne et som ikke krever lisens.

Videre var det en utfordring å lagre video til fil samtidig som man viste den i applikasjonen gjennom FFmpeg og FFME. Etter mye lesing av dokumentasjonen var tilsynelatende den eneste måten å gjøre dette på å strømme videoen over User Datagram Protocol (UDP) fra input, samtidig som den ble skrevet til fil. Etter mye testing var det ansett som sannsynlig at forsinkelsen dette ville ha medført ville vært altfor stor i forhold til kravet.

3.1.3 Alternativ løsning 3: FFmpeg og FFME med screencap av mediaelementene

På grunn av forsinkelsen som kom av UDP-strømmingen i alternativet over, ble et tredje alternativ undersøkt. Teorien var at FFME og FFmpeg også ville bli brukt her, men for å lagre videoene ville gruppen heller prøve å gjøre screencap av mediaelementene slik at UDP-strømming kunne droppes. Dersom dette hadde fungert, ville det bare ha vært rundt 0,5 sekunders forsinkelse på videoavspillingen på det tidspunktet. Ulempen var at det muligens kunne blitt en dårligere kvalitet på den lagrede videoen. Minimering av forsinkelse var imidlertid prioritert framfor videokvalitet av den lagrede videoen.

3.1.4 Alternativ løsning 4: C++ og Qt

Dersom C++ hadde blitt brukt i stedet for C#, ville forsinkelsen kunne blitt minimert betraktelig. Grunnen til dette er at FFmpeg er skrevet i C++, og dermed ville alt kunne blitt gjort på et mye lavere nivå og mer tilpasset prosjektets bruksmål. Men da ville det ha vært en større utfordring å lage brukergrensesnittet, ettersom Qt da sannsynligvis ville ha blitt brukt som rammeverk for front-enden. Qt er greit å bruke og fungerer veldig bra, men gruppen har ikke en så god kjennskap til denne programvaren, og dermed vil WPF være enklere og mer effektivt å programmere i.

3.1.5 Alternativ løsning 5: Et eventuelt alternativ til FFME

Et bedre alternativ til FFME kunne ha vært å finne et bibliotek med C# kode som kunne fange og spille av video og lyd. Et slikt bibliotek kunne ha vist seg å være raskere og bedre enn FFME. Det ble satt av en god del tid til å lete etter dette, men uten hell. Planen var helst at det skulle kunne brukes i WPF, og dermed ble utvalget av bibliotek allerede begrenset.

3.1.6 Diskusjon rundt alternative løsninger

Det som hele tiden var klart for prosjektet var at ønske om å gjøre video-rendering med så liten forsinkelse som mulig. Det var også å lage et program med et enkelt forståelig brukergrensesnitt, av god nok kvalitet til å bli tatt i bruk av sykehus. Valget falt da på alternativ 2; FFmpeg og FFME. Det beste ville jo ha vært å gå for en wrapper-klasse i C# for FFmpeg. Men kompetansen rundt videobehandling og FFmpeg innen gruppen i forkant av prosjektet var svært liten, og det hadde da blitt for vanskelig og tidkrevende å få det til på denne måten. Det samme gjelder i enda større grad for alternativ 1, altså å bruke wrapperklasser for DirectShow i C#. Det ville ha vært enda vanskeligere å sette seg godt nok inn i enn FFmpeg. Gruppen fant aldri et bedre alternativ til FFME, og dermed falt valget på nettopp det rammeverket. Bruk av FFME medfører automatisk bruk av FFmpeg, og FFmpeg har også blitt brukt litt utenfor FFME. Etter mye arbeid og utvikling av kode basert på FFME, ble det funnet en løsning på forsinkelsen som innspillingen medførte. I

stedet for å bruke FFmpeg til å spille inn videostrømmene samtidig som de ble sendt over UDP, viste det seg mulig å gjøre dette ved hjelp av FFME-biblioteket. Dette gjennombruddet kom da gruppen sendte inn et spørsmål til en av FFME's utviklere Mario Di Vece vedrørende dette temaet, og fikk tilbake kode som hjalp som en veiledning til å få dette til på egenhånd. Dermed bortfalt også alternativ 3; altså å kjøre en screencap av alle mediaelementene som et alternativ til å lagre videostrømmene på en bedre måte.

FFME har hele tiden bydd på nye utfordringer. Det er et rammeverk som fortsatt er under utvikling, noe som har ført til en god del forvirring etter versjonsoppdateringer. Fordelen er at det faktisk er mulig å be utviklerne om hjelp, noe som viste seg svært nyttig da det ble gjort. Det er også svært effektivt og bra å bruke når man først lærer seg det. Men det er en utfordring å tilegne seg kunnskap om det ettersom tilgjengelig dokumentasjon er veldig overfladisk og mangelfull. Det virker også som at utviklerne av FFME antar at brukerne har svært god kjennskap til FFmpeg. Men ettersom forsinkelsen i videofremvisningen nå bare er på rundt 0,3 sekunder, og alt generelt sett fungerer bra, er gruppen svært fornøyde med valg av alternativ. Til tross for at det er vanskelig å lære seg rammeverket, er det mer effektivt å bruke enn et annet rammeverk på et lavere nivå.

3.2 Valg av verktøy og programmeringsspråk

Vi har valgt å bruke C# som programmeringsmiljø, WPF som rammeverk for brukergrensesnitt, Visual Studio som IDE og FFmpeg som verktøy for å capture video og lyd.

3.2.1 C#

C# er et programmeringsspråk utviklet av Microsoft som brukes for utvikling av applikasjoner under .Net rammeverket. Dette er et objektorientert og typesikkert programmeringsspråk. C# stammer fra C familien av programmeringsspråk. Dette medfører mange likhetstrekk mellom C#, C, C++, Java og JavaScript. Mange av C# sine innebygde funksjoner bidrar til at man kan lage robuste applikasjoner. "Garbage collection" bidrar til automatisk frigjøring av minne. Feilhåndtering bidrar til strukturerte

og konkrete feilmeldinger. Det typesikre designet til C# gjør at man ikke kan lese fra variabler som ikke har blitt initialisert (Microsoft. 05. April 2019).

3.2.2 WPF

Windows Presentation Foundation (WPF) er en grafisk komponent av .Net rammeverket. Det er denne komponenten som lar deg lage brukergrensesnitt til Windows-baserte applikasjoner. WPF bruker DirectX for å fremstille eventuelle objekter som skal opptre i brukergrensesnittet. Ved å bruke DirectX slipper programmereren å tenke på hvordan maskinvaren skal motta forskjellige informasjon da dette fikses av DirectX. WPF deler opp business logic og brukergrensesnittet, slik som lignende XML orienterte objektmodeller gjør. I .Net rammeverket, spesielt WPF, brukes XAML til å definere og linke sammen forskjellige grafiske elementer (Microsoft, 11. april 2016).

3.2.3 Visual studio

Visual Studio er en IDE laget av Microsoft. Programmet kan brukes til å utvikle skrivebordsprogrammer, websider, webapplikasjoner og mobilapplikasjoner. C#, C++, Visual Basic og XAML er bare noen av de 36 forskjellige språkene IDEen støtter. Grunnen til at programmet kan håndtere så mange forskjellige språk er fordi de kompileres ned til samme lavnivåspråk. Visual Studio har også en kraftig feilsøker som fungerer på både høynivå- og lavnivåspråk (Microsoft, u. å).

3.2.4 FFmpeg

FFmpeg er et ledende rammeverk i stand til å dekode, kode, krysskode, multiplekse, demultiplexe, strømme, filtrere og spille av multimedia. Rammeverket støtter alle format, alt fra de eldste og utdaterte opp til den nyeste teknologien i feltet. FFmpeg kombinerer de beste løsningene av gratis programvare. Dette blir gjort fordi rammeverket skal prestere likt for både utviklere og sluttbrukere (FFmpeg, u. å).

3.2.5 FFME

FFME er en avansert erstatning for Microsofts WPF MediaElement Control bibliotek. Standard MediaElement bruker DirectShow for avspilling av multimedia, mens FFME bruker det langt kraftigere FFmpeg. Dette gjør at man slipper å installere forskjellige codecer på maskinen som skal kjøre programmet (Unosquare, u. å).

3.2.6 Git

Git er et distribuert versjonskontrollsystem som brukes til å spore kodeendringer under programvareutvikling. Ved å bruke "Branches" simplifiseres problemene rundt parallelt arbeid på samme prosjekt. Versjonskontrollsystemet er designet for å være distribuert, sikkert og raskt (Git, u. å).

3.2.7 Microsoft Server Express

Microsoft Server Express er en versjon av Microsofts SQL relasjonsdatabase. Denne programvaren er gratis og egnet for små og integrerte applikasjoner. Serveren kan kjøre i både Windows, Linux og i docker-containers. Dette betyr at programvaren er veldig fleksibel. Dette er også den minst sårbare databasen på markedet (Microsoft, u. å).

3.3 Prosjektutviklingsmetode

3.3.1 Utviklingsmetode

En smidig utviklingsmetode ble ganske raskt valgt som utviklingsmetode i startfasen av prosjektet. Ettersom gruppen består av to stykker som skal jobbe i tett samarbeid, anså gruppen det som at dette ville være den beste tilnærmingen. Gruppen planla å drive en iterativ, modulbasert utvikling der det ville egne seg godt å dele opp arbeidet på denne måten. I startfasen var det derimot ikke relevant å dele opp arbeidet slik, ettersom alt arbeidet gikk ut på å teste ut forskjellige verktøy og programmeringsspråk.

Agile programvareutvikling er mer enn bare et rammeverk, slik som scrum og extreme programming. Dette er et sett med verdier og fremgangsmåter som beskriver hvordan prosjektgruppen skal håndtere arbeidet. Det som skiller Agile fra andre utviklingsmetoder er fokuset på menneskene som er involvert. Den endelige løsningen utvikler seg gjennom samarbeid mellom selvorganiserte og tverrfaglige grupper, som tar i bruk fremgangsmåten som best passer oppgaven. Hver for seg har ikke gruppemedlemmene noen spesiell rolle. Når hele gruppen er samlet bedømmes helhetsinntrykket, og man ser om den riktige kunnskapen eksisterer innad i gruppen for å kunne utføre prosjektet. Det er nettopp dette som gjør en slik gruppe tverrfaglig. En leder for en slik prosjektgruppe vil ofte være mindre tilstedeværende. Lederen vil heller ta et steg tilbake og la gruppen komme frem til en fremgangsmåte seg imellom. (Agile Alliance, u. å)

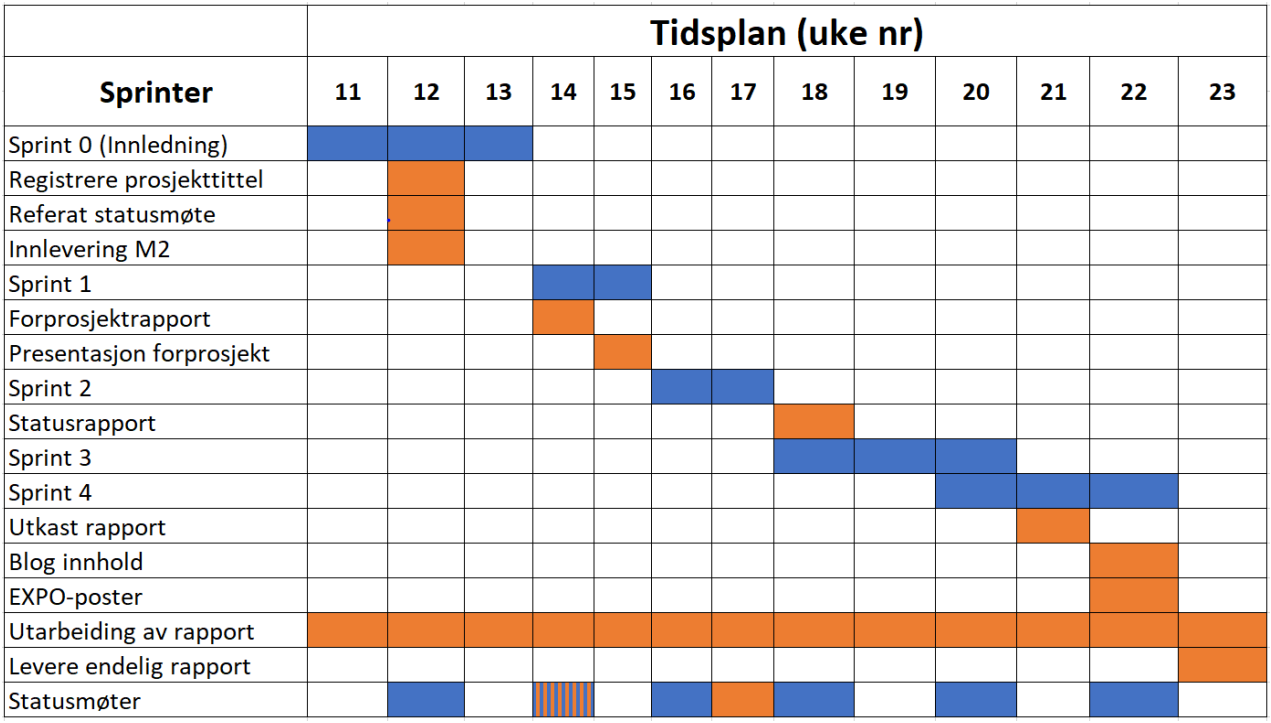
Agile programutvikling er bygger på 12 grunnprinsipper. Disse ligger vedlagt i kapittel 9.3.

3.3.2 Prosjektplan

Fargekoder:

Blå: Arbeid som omhandler applikasjonen.

Oransje: Arbeid som omhandler rapporten.



Figur 3-1 GANTT diagram

Forklaring GANTT diagram

Sprint 1

Denne sprinten inneholder innledningen til prosjektet. Det innebærer valg av rammeverk samt lage en mappe til prosjektet på GitHub. Det viktigste punktet i denne fasen er å få all nødvendig programvare installert på våre egne PCer og PC-en til Haukeland. Denne fasen inneholder også innlevering av prosjekttittel, et referat fra statusmøte og M2.

Sprint 2

I denne sprinten er hovedfokuset å få sendt og fremstilt videostrømmene fra de forskjellige kameraene. Her må vi fokusere på å finne de riktige parameterne slik at både kvalitet og responstiden blir best mulig.

I denne fasen skal det også leveres en forprosjektrapport der vi presenterer de viktigste temaene i bacheloroppgaven vår. Det skal også avholdes en presentasjon av denne rapporten på HVL.

Sprint 3

Sprint 3 innebærer å få fremstilt et brukergrensesnitt til programvaren. Her er det viktig at vi avholder møter med oppdragsgiver slik at de kan komme med ønsket design. En del av oppgaven er å gjøre programvaren mer brukervennlig. Det er derfor viktig at vi får tilbakemeldinger fra personene som faktisk skal bruke systemet.

Sprint 4

I sprint 4 vil fokuset vårt være å forbedre forsinkelsen og synkronisering på videostrømmene. Programvaren skal ikke bare brukes til CLE-test, men også til spirometri. Denne testen er avhengig av forsinkelse som er tilnærmet lik null. Det er også viktig at alle videostrømmene er synkronisert ettersom at data skal brukes i forskningsprosjekter.

I denne sprinten skal det også leveres en statusrapport som inneholder eventuelle endringer og status på arbeidet.

Sprint 5

Sprint 5 innebærer å implementere lagring på database og server. Grunnet nye regler rundt GDPR må programvaren være i stand til å lagre data på en server. Vi planlegger å lagre pasientdata i en database. Videoene som blir lagret fra testen vil ligge på en annen server. Vi planlegger så å koble disse opp mot hverandre. Det eksisterende systemet har ikke noen god organisering av lagringen.

I denne sprinten skal det også leveres et utkast av bachelor rapporten, blogginnhold samt en plakat som skal brukes på EXPO.

Utarbeiding av rapport

Bacheloroppgaven vil bli utarbeidet gjennom hele perioden. Det skal leveres forskjellige delinnleveringer parallelt med de fem sprintene. Det skal også skrives en blogg som dokumenterer arbeidet på oppgaven. Denne bloggen har også som oppgave å holde veileder oppdatert på arbeidet som utføres. Oppgaven skal helt til slutt presenteres på EXPO.

Statusmøter

Det vil bli avholdt statusmøter gjennom hele perioden. Både med intern veileder og oppdragsgiver. Møtene med oppdragsgiver vil forekomme annenhver uke. Dette gjøres for å hele tiden holde oppdragsgiver oppdatert på arbeidet som utføres. Møter med intern veileder på HVL vil bli avholdt etter behov.

3.3.3 Risikohåndtering

Det ble tidlig valgt å lage en risikoanalyse for å avdekke eventuelle risikoer og konsekvenser av disse. Risikoene ble rangert fra 1-5, der 1 er liten sannsynlighet for at de inntreffer og 5 er stort sannsynlighet for at de inntreffer. Konsekvensene ble også rangert fra 1-5, der 1 er ubetydelig konsekvens, mens 5 er kritisk konsekvens. Risikofaktoren er risikoen multiplisert med konsekvensen slik at mulige verdier er i området 1-25. Er risikofaktoren under 10 anses risikoen som uproblematisk og trengs ikke å tas tak i umiddelbart. Ligger risikofaktoren i intervallet 10-20 anses risikoen som problematisk og forebyggende tiltak burde opprettes umiddelbart. Hvis en risiko har risikofaktor på 20+ vil det være kritisk for prosjektets levetid å ta tak før den inntreffer.

I risikoanalysen er det identifisert risikoer og tiltak for å unngå at disse skal inntreffe. Ved å lage en slik analyse med tiltak vil man kunne både forebygge og håndtere risikoene bedre hvis de skulle inntreffe. Risikoanalysen ligger som vedlegg i kapittel 9.1

De risikoene med størst konsekvens for prosjektet vil bli presentert under. Det vil også bli presentert hvordan gruppen håndterte risikoene underveis i arbeidet

Dårlig håndtering av tid

For å passe på at prosjektet fulgte den gitte tidsrammen ble det tidlig laget et GANTT diagram (Figur 3-1). Hvis tidsrammene ikke ble fulgt risikerte gruppen å ikke kunne levere tilfredsstillende løsning til prosjekteier. For å unngå dette måtte prosjektet prioriteres over andre aktiviteter. En annen faktor som hjalp gruppen å opprettholde tidsfristene var god kommunikasjon. Ved å ha fokus på god og åpen kommunikasjon hjalp dette gruppen å innse når ekstra tid måtte legges ned for å fullføre oppgavene.

Mangel på kompetanse

Ingen av gruppemedlemmene hadde arbeidet med håndtering av multimedia før prosjektstart. Dette medførte at det ble satt av en periode på tre uker i starten av prosjektet som ble brukt til å finne og sette seg inn i de nødvendige rammeverkene og verktøyene (Figur 3-1). Under arbeidet støtte gruppen likevel på tekniske utfordringer. For å løse disse utfordringene etterspurte vi teknisk kompetanse fra Steinar Søreide. Hans kompetanse, prøving og feiling samt dokumentasjon gjorde at gruppen stort sett fikk løst de tekniske utfordringene som oppstod.

Utvikling av programvaren går utover rapportskriving

For at utviklingen av programvaren ikke skulle stjele for mye tid fra rapportskrivingen, ble det satt av egne perioder der rapporten skulle skrives. Det ble også bestemt at hovedfokuset skulle ligge på rapportskriving de siste tre ukene før innleveringsfristen 3. juni.

Skjev arbeidsfordeling

Det var ikke noen klar rollefordeling innad i gruppen. Siden gruppen bare består av to medlemmer var det viktig at det var god arbeidsfordeling. For å forebygge skjev arbeidsfordeling var det viktig at gruppemedlemmene hadde god kommunikasjon seg imellom. Det var viktig at gruppemedlemmene ga god og klar beskjed hvis det var misnøye med noe slik at det kunne bli tatt tak i tidlig.

3.4 Evalueringsmetoder

Det ble holdt en del møter underveis i prosjektet slik at det var sikkert at utviklingen stod i henhold til personellet på oppdragsgiver sine ønsker. Det ble gjennomført en test av programvaren 20. mai. Her ble programvaren vist frem til personalet på Haukeland. Personalet fikk også prøve seg frem og bruke de forskjellige funksjonene i programvaren. Etter gjennomført test fikk gruppen tilbakemeldinger på hva som eventuelt måtte forbedres, forandres eller legges til. Denne funksjonelle testen ga også god tilbakemelding på hvorvidt programmet fungerte som det skulle.

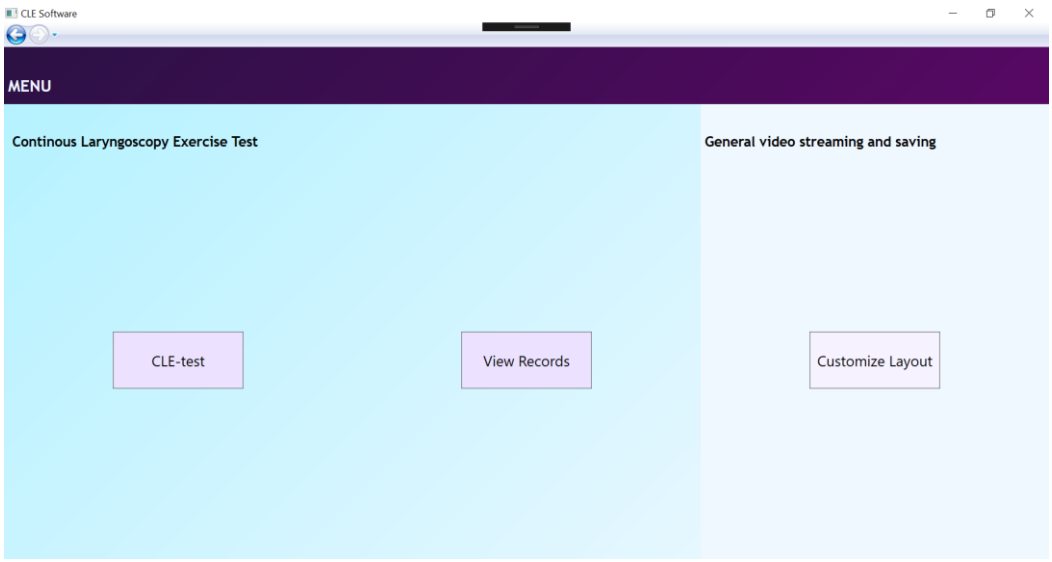
I tillegg til dette var det selvsagt planlagt å gjennomføre grundige tester på programvaren og kildekoden. Programvaren vil også kontinuerlig testes på laben til Haukeland. Dette blir gjort for å bekrefte at utstyret er kompatibelt med programvaren. Evalueringsmetoden og resultatet vil bli nærmere gjennomgått i kapittel 5.

4 DESIGN OG UTFORMING

Dette kapittelet omhandler designet og utformingen på den endelige løsningen av programvaren.

4.1 Utforming og design

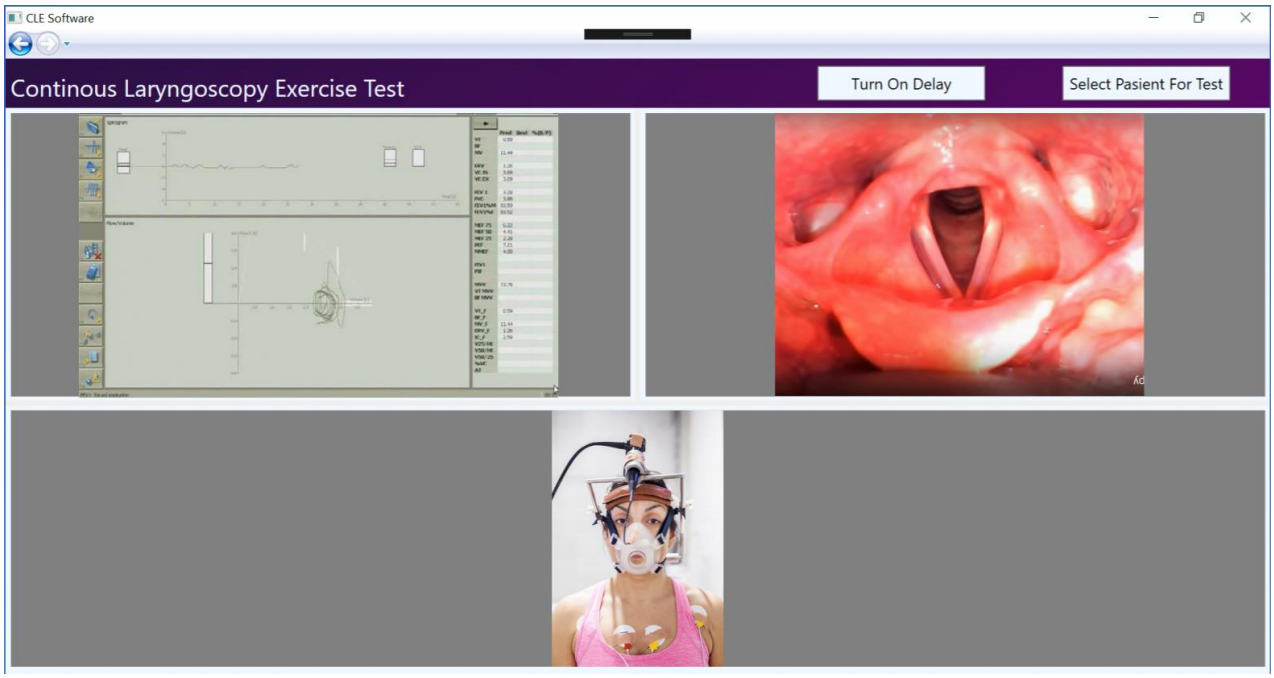
Programvaren er designet i henhold til oppdragsgivers ønsker, samt gruppens egne idéer for best mulig løsning med tanke på brukervennlighet og funksjonalitet. Fokuset har vært å ha et brukervennlig design, ettersom oppdragsgiver fra starten sa at de ville ha et intuitivt program som ikke krevde særlig innledende veiledning for å brukes. I utgangspunktet var programvaren bare ment for å brukes til CLE-tester. Men ettersom det viste seg å være ytterlige interessenter på Haukeland som ønsket et program med svært tilsvarende funksjonalitet til egne tester som innebærer videofremvisning- og lagring, la gruppen også inn støtte for dette. Dermed endte programmet opp med en meny der man kan velge mellom å gjennomføre standard CLE-test, å se ferdige tester og å bruke programmet til generell videostrømming- og lagring.



Figur 4-1 Menyen

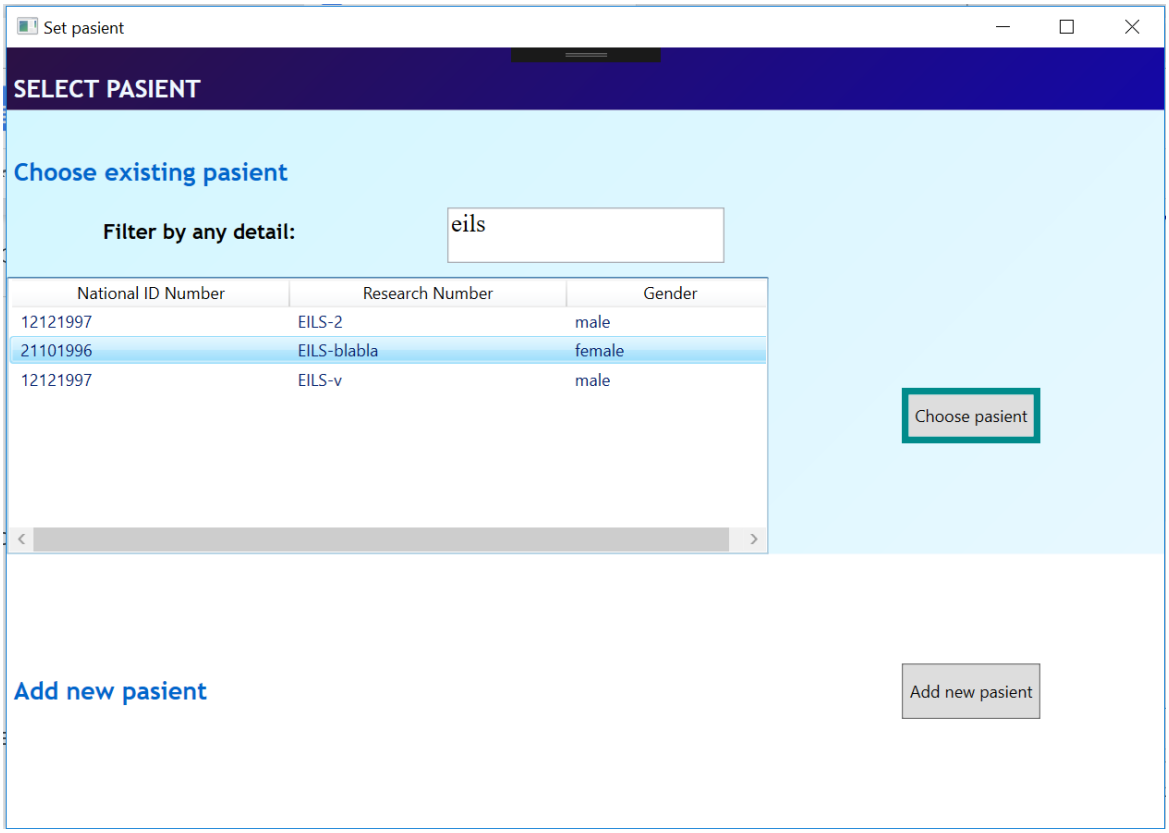
4.1.1 Brukstilfelle: standard CLE-test

Den første knappen, “CLE-test”, tar deg videre til et standard oppsett for CLE-test (Figur 4-1).



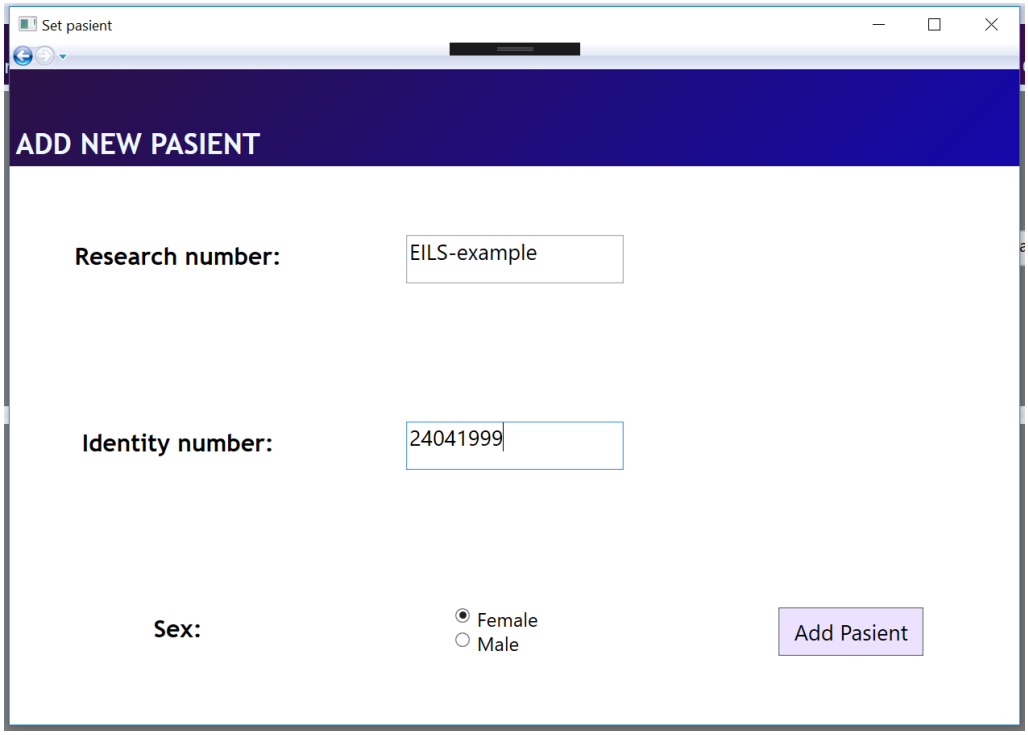
Figur 4-2 Standardoppsett for CLE-test

Kamera-oppsettet er hardkodet slik at dersom kameraene for CLE-test er tilkoblet, dukker de opp med en gang på riktig plassering slik at man ikke trenger å velge mediaelementene selv. Dersom man trykker på "turn on delay", slås en forsinkelse på 0,7 sekunder på for skop-kameraet (bilde oppe til høyre figur 4-2) og portrett-kameraet (bilde nede figur 4-2). Førstnevnte er da kameraet som filmer strupen, og sistnevnte det som filmer personen forfra. Forsinkelsen kan skrues av og på etter behov. Før man kan begynne å spille inn, må man velge pasient for undersøkelsen. Dermed trykker man på "select new pasient", og et popup-vindu åpner seg (figur 4-3).

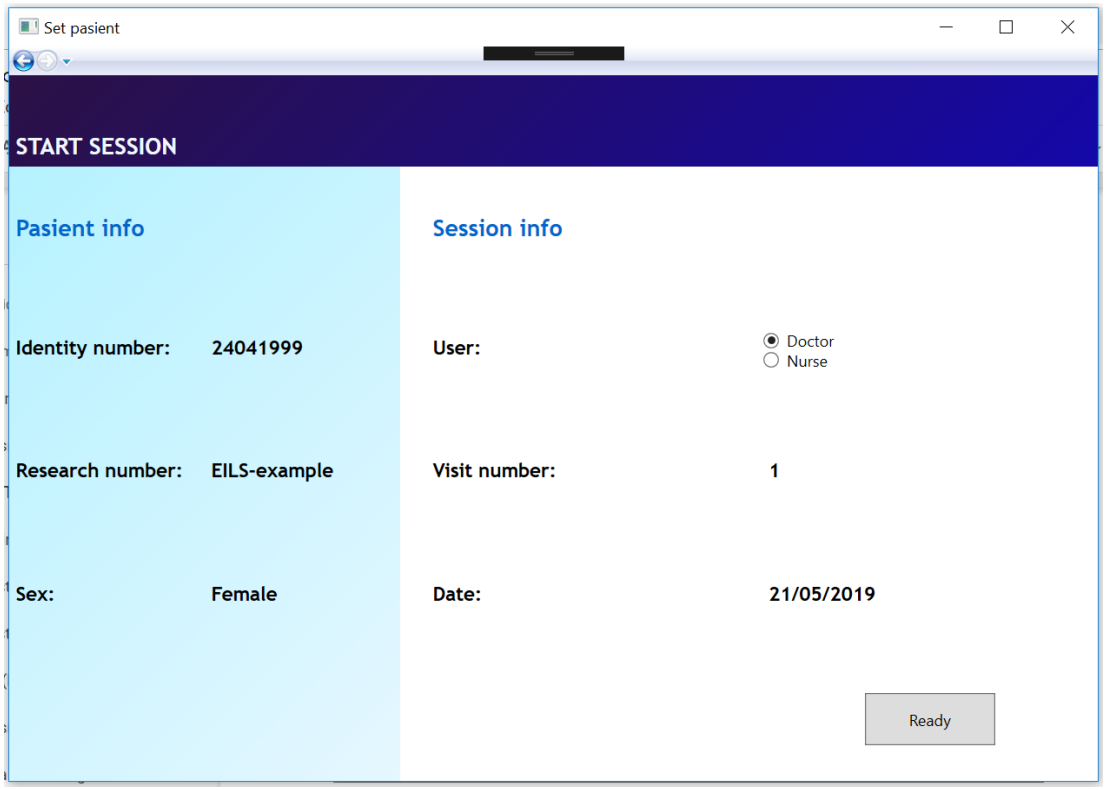


Figur 4-3 Siden for å velge eksisterende eller ny pasient

Her kan man enten velge en eksisterende pasient, eller opprette en ny. Det har blitt satt opp en tilkobling mot databasen, slik at alle pasienter som har tatt undersøkelsen før blir listet opp. Man kan også filtrere listen, noe som vil bli svært nyttig når man får et større antall pasienter i databasen. Dersom man velger en eksisterende pasient, blir man direkte satt videre til en side der man skriver inn nødvendig data for å starte undersøkelsen (figur 4-5). Om man i stedet trykker "add new pasient", må man legge inn data for en ny pasient før man kommer til denne siste siden (figur 4-4).



Figur 4-4 Siden for å legge til ny pasient

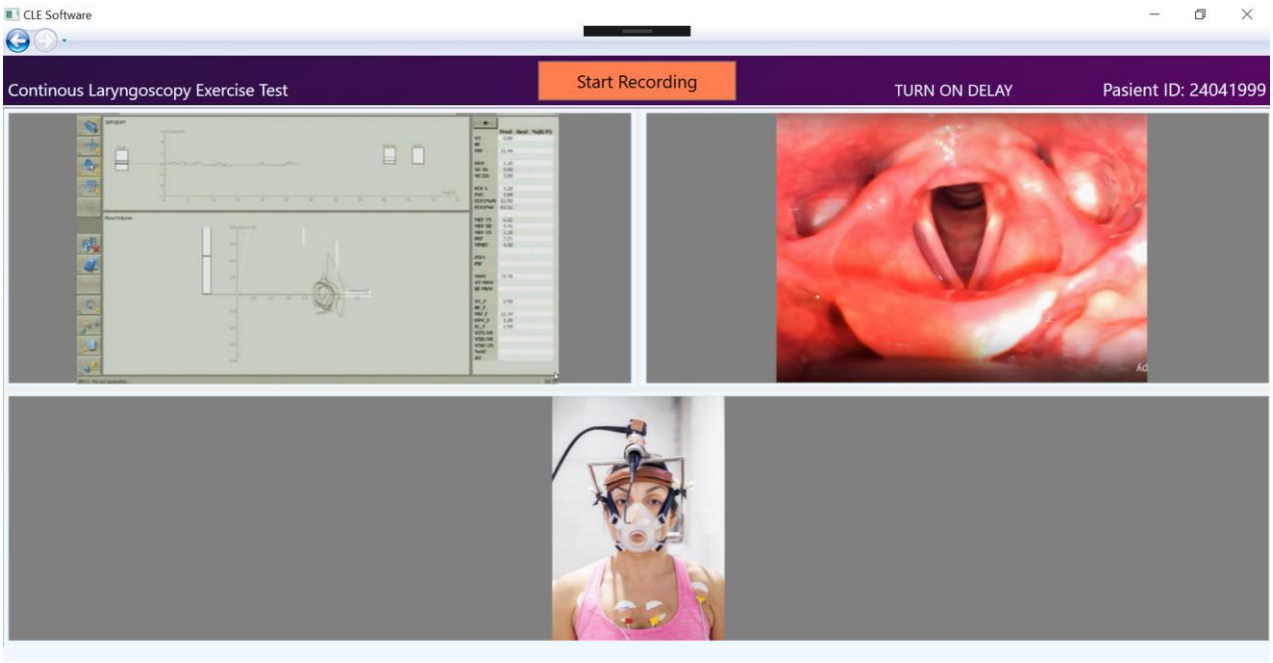


Figur 4-5 Siden for å starte innspilling av økt

I "start session"-siden (figur 4-5) er alt man trenger å gjøre å krysse av for hvem som skal utføre testen, og så trykke "ready". Noen pasienter tar testen flere ganger. "Visit number" viser da til hvilket nummer den aktuelle testen er. Dette tallet

genereres automatisk av en database-spørring. Når man så trykker "ready", stenges dette vinduet og informasjonen om denne undersøkelsen blir lagret i databasen.

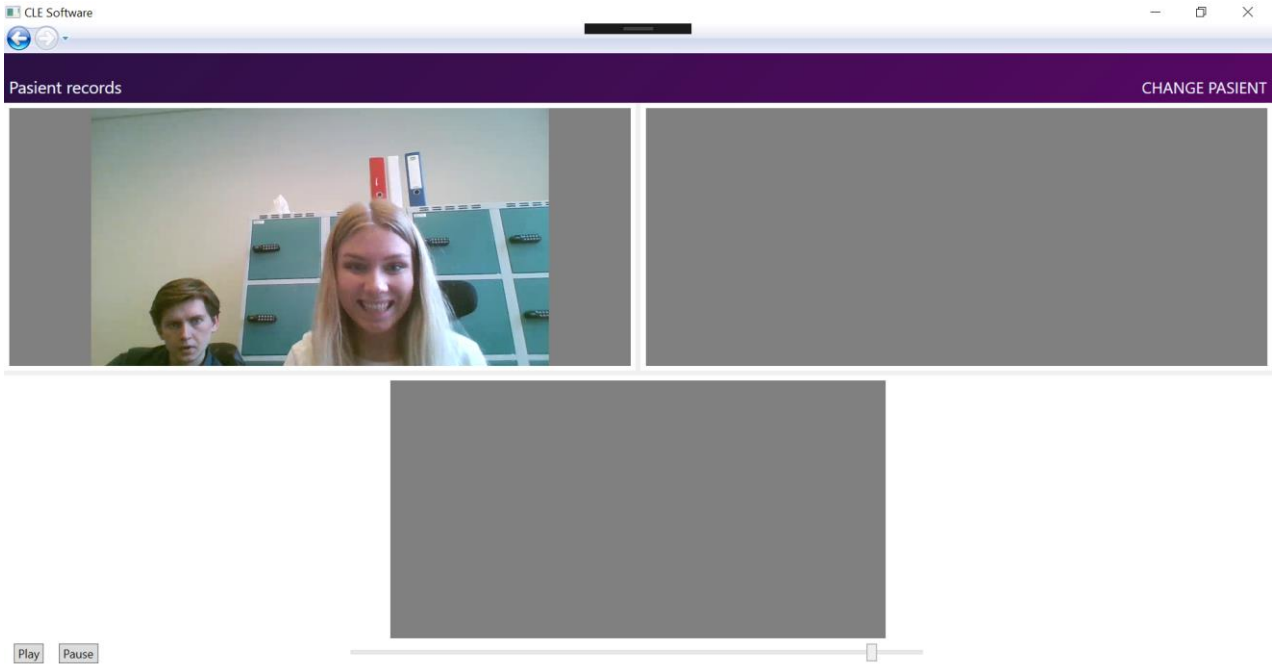
Nå byttes knappen "select new pasient" ut med en etikett med pasient-ID, slik at man ser hvilken pasient det er som gjennomfører testen. Knappen "start recording" blir nå synlig, ettersom alt er klart til å begynne å spille inn videoene. Når man trykker på den, lagres videostrømmen fra alle mediaelementene som har satt en videokilde. Knappen blir så erstattet med en knapp der det står "stop recording", og når man trykker på den stopper innspillingen (figur 4-6).



Figur 4-6 standardoppsettet for CLE-test med mulighet for å starte innspillingen

4.1.2 Brukstilfelle: se tidligere CLE-tester

Alternativ nummer to i menyen, "View Records", fører deg til modulen der du kan se tidligere gjennomførte CLE-tester. Denne er foreløpig ikke helt ferdigstilt, men funksjonaliteten for å vise videoer er der. Man kommer først til en side der man kan velge blant alle tidligere pasienter som har gjennomført testen. Når man så har valgt en pasient, skal man få opp alle testene denne pasienten har gjort, og så skal man kunne velge hvilken man vil se videoene fra. Det er denne delen som mangler. Etter at man har valgt test, kommer man til en side som viser videoene fra den. Der kan man starte, pause og spole i videoene (figur 4-7).

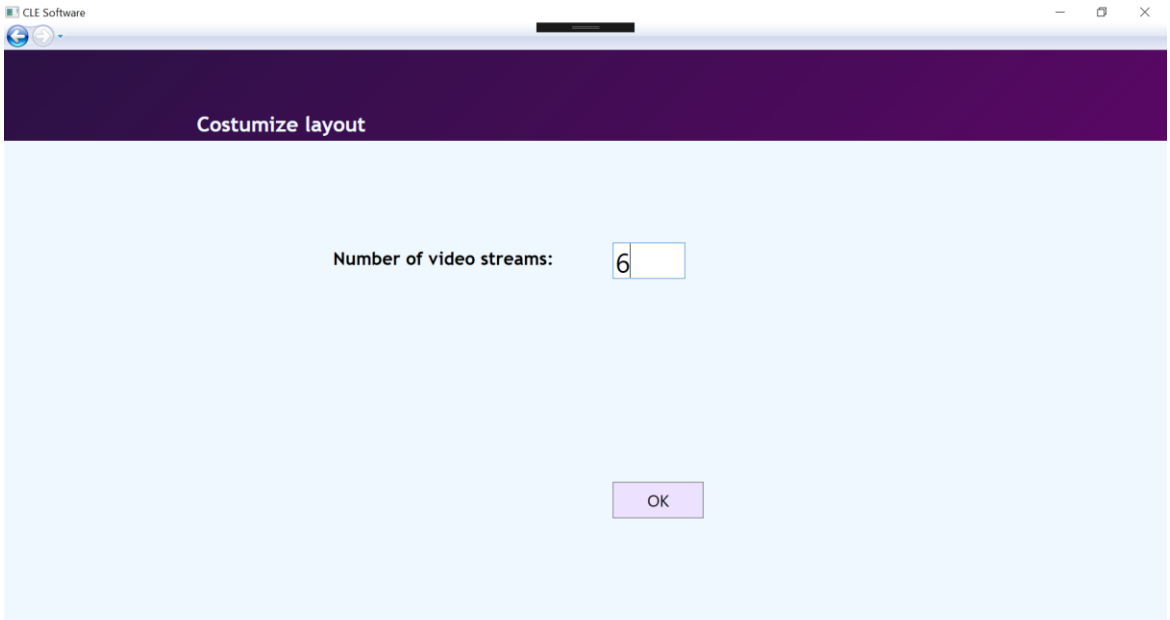


Figur 4-7 Siden for å vise tidligere innspilte filmer

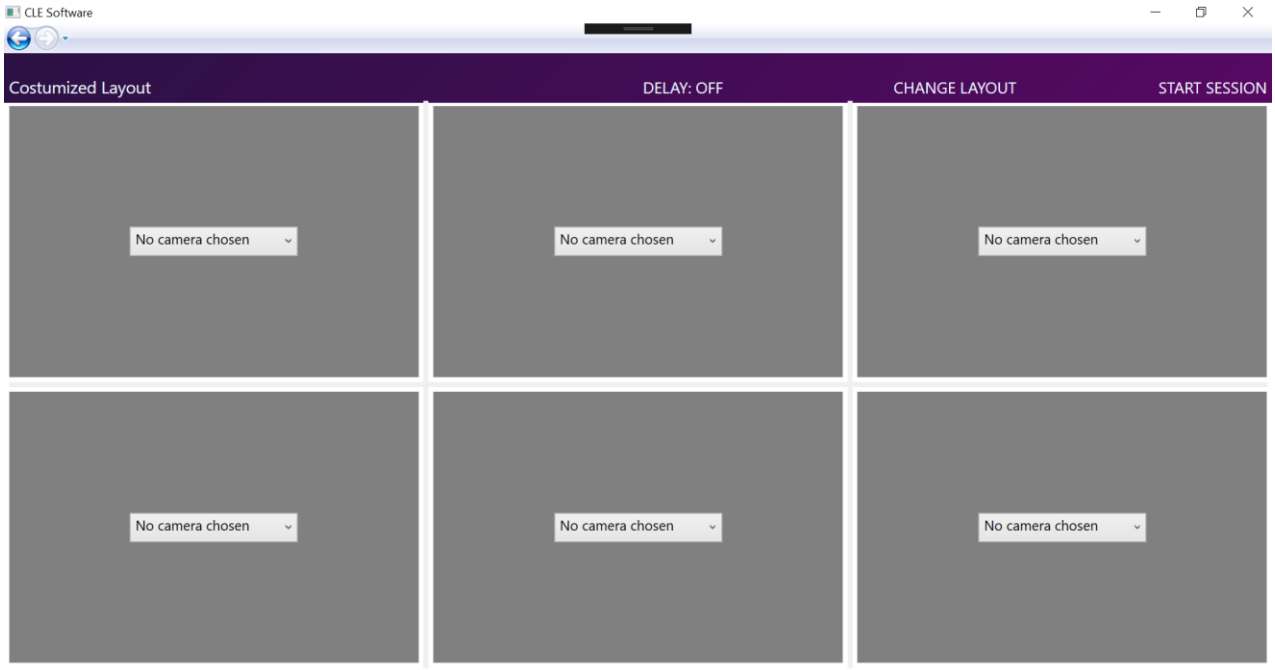
Man har også en knapp oppe til høyre for å bytte pasient, og dermed se andre videoer.

4.1.3 Brukstilfelle: generell videostrømming

Dersom man velger det siste alternativet i menyen, "Customize Layout", får man velge antallet videostrømmer man ønsker (figur 4-8). Deretter får man opp et tilsvarende antall videostrømmer i neste side.



Figur 4-8 Meny for valg av antall kamerakilder man ønsker at programvaren skal vise

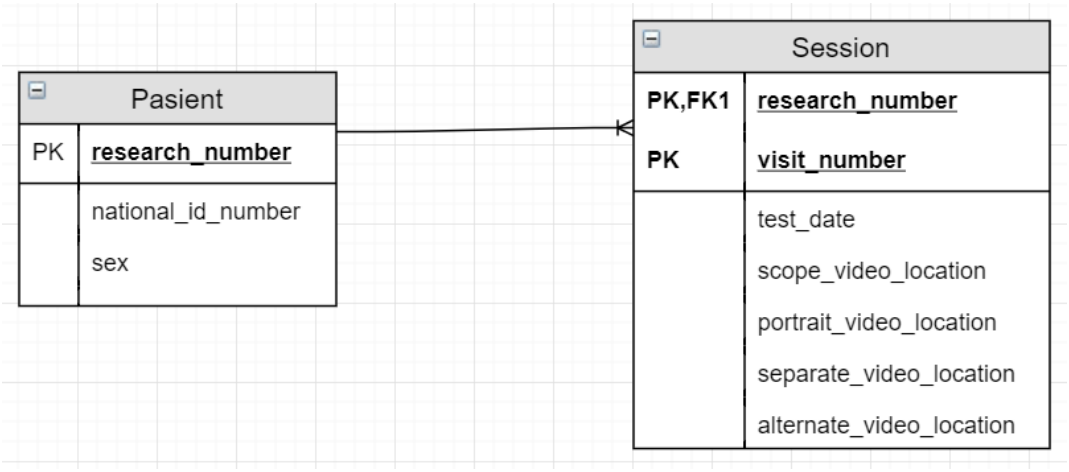


Figur 4-9 Siden med egendefinert antall kamerakilder

Det er da denne siden de andre interessentene skal bruke. Alle videokilder som er koblet til datamaskinen dukker opp i comboboxene, noe som gjør dette programmet helt generisk med tanke på videostrømming. Denne siden er heller ikke helt ferdig, ettersom det her skal implementeres funksjonalitet for å velge hvilke mediaelementer som skal ha forsinkelse på videostrømmen de viser. Man skal også kunne skrive inn nøyaktig hvor stor denne forsinkelsen skal være. Den blir da felles for alle videoer som har forsinkelse slått på.

4.2 Database

Gruppen har fått tilgang til et eget område på Haukelands database, og bruker da dette til lagring av detaljer for programmet. Dette er en SQL Server Express-database. I den har vi to entiteter, pasient og session.



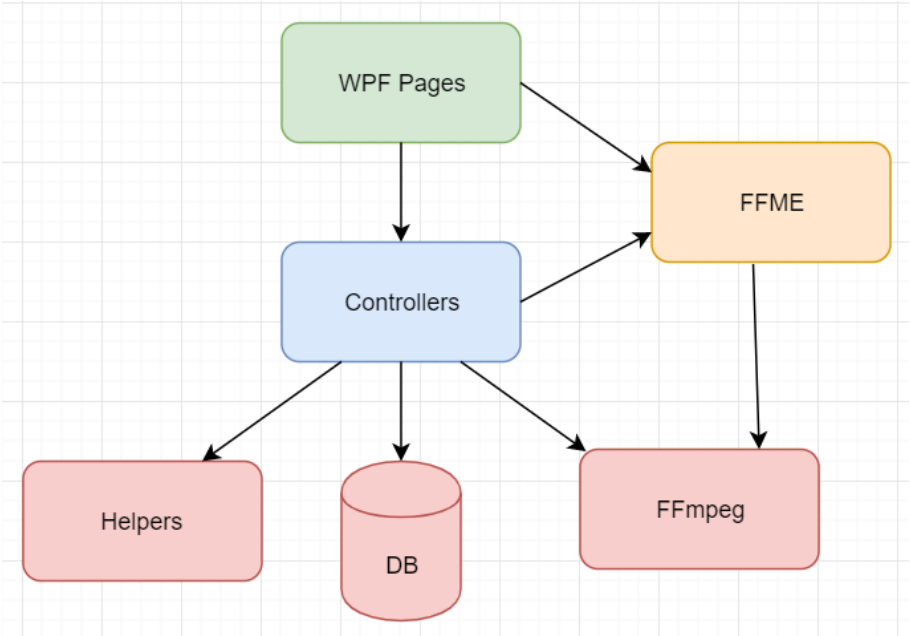
Figur 4-10 ER-diagram for databasen

Men ettersom programvaren muligens skal brukes til andre brukstilfeller enn CLE-test, kan det bli aktuelt å utvide databasen til å inneholde en entitet for lagring av generelle videoer. Den vil sannsynligvis også inneholde en fremmednøkkel til Pasient, men ha litt andre attributter enn Session.

Videoene fra innspillingen i programvaren lagres på en ekstern datamaskin via Remote Desktop Protocol (RDP), og fillokasjonen settes inn i databasen.

4.3 Teknologi og implementasjon

Kodearkitekturen vår er modulbasert, og den er ordnet i et hierarki som vist i figuren nedenfor.



Figur 4-11 Generell arkitektur for programvaren

På toppen av alt ligger WPF Pages, som er front-end i prosjektet. FFME brukes da som et GUI-element i de for å vise videostrømmene. WPF-sidene inneholder en del logikk, men er i hovedsak relatert til GUI-et. Implementasjonen av detaljer rundt videostrømmingen skjer i Controllers, som er navngitt nettopp det fordi de skal kontrollere det som skjer i front-enden når det kommer til videostrømming og databaseoperasjoner. Kontrollerne har igjen hjelpeklasse-modulen Helpers, som inneholder en del logikk og operasjoner på flere områder.

For å kommunisere med databasen brukes System.Data.SqlClient, et Microsoft-bibliotek. Det er enkelt å bruke i dette tilfellet, ettersom det fungerer godt til enkle SQL-spørringer.

Essensen i dette prosjektet er bruk av FFME. Det kan, som nevnt tidligere, brukes til både innspilling og visning av videostrømmer- og lyd.

```
<ffme:MediaElement x:Name="MediaElement1" LoadedBehavior="Play" UnloadedBehavior="Manual" MouseDown=
<ffme:MediaElement x:Name="MediaElement2" LoadedBehavior="Play" UnloadedBehavior="Manual" MouseDown=
<ffme:MediaElement x:Name="MediaElement3" LoadedBehavior="Play" UnloadedBehavior="Manual" MouseDown=
```

FFME er en utvidelse av Microsoft MediaElement (ME), som er et GUI-element som brukes til å vise media i WPF. Det er enkelt for oss å sette videostrømkilden til elementene, slik som gjøres i denne metoden (figur 4-12).

```
public void changeFFMESource(Unosquare.FFME.MediaElement mediaElement, string source)
{
    string original;

    if (mediaElement.Source == null)
    {
        original = null;
    }
    else { original = URIStringHelper.extractCameraNameFromURI(mediaElement.Source.OriginalString);

    if (source == CameraNamesAndAliases.getNoCameraChosenString())
    {
        mediaElement.Source = null;
    }
    else if (original == null && source != null || source != null && original != source)
    {
        mediaElement.MediaStateChanged += MediaElement_MediaStateChanged;
        mediaElement.MediaFailed += MediaElement_MediaFailed;
        mediaElement.MediaInitializing += MediaElement_MediaInitializing;
        mediaElement.MediaOpening += MediaOpening;
        mediaElement.Source = new Uri(@"device://dshow/?video=" + source);
    }
}
```

Figur 4-12 Utdrag fra MediaElementController. Metode for å sette en strømmekilde til et medieelement

FFME er, som så mye annet i WPF, basert på delegatfunksjoner. De er svært nyttige å bruke, ettersom det er slik de har valgt å stille parameterne til FFmpeg i FFME.

```
private void MediaOpening(object sender, Unosquare.FFME.Common.MediaOpeningEventArgs e)
{
    Unosquare.FFME.MediaElement me = sender as Unosquare.FFME.MediaElement;
    string cameraSource = getCameraOfMediaElementAndCameraPair(me);
    e.Options.MinimumPlaybackBufferPercent = delayController.getDelayOnCamera(cameraSource);
    e.Options.DecoderParams.EnableFastDecoding = true;
    e.Options.DecoderParams.EnableLowDelayDecoding = true;

    // use this for hardware
    //e.Options.VideoHardwareDevice
}
```

Figur 4-13 Delegatfunksjon for å sette forsinkelse på et medieelement

For eksempel er det ønskelig å ha så liten forsinkelse i videofremvisningen som mulig. Dekoderparametrene til FFmpeg settes derfor etter dette ønske. Man kan også sette en gitt forsinkelse ved å gi inn antall sekunder ønsket forsinkelse til playback-bufferen, noe som er essensielt å kunne gjøre i programmet ettersom noen videostrømmer skal ha en gitt forsinkelse i forhold til andre (figur 4-13).

5 EVALUERING

Dette kapittelet handler om hvordan gruppen valgte å evaluere prosjektet. Det vil også være et resultat av evalueringen holdt på Haukeland.

5.1 Evalueringsmetoder

For å sikre at resultatet tilfredsstilte oppdragsgiver, har gruppen gjennom hele perioden drevet med funksjonell testing av programvaren. I startfasen av prosjektet var kravspesifikasjonene ganske løse. Dette medførte at de endret seg kontinuerlig gjennom prosjektperioden som resultat av møtene med oppdragsgiver.

Evalueringene ble gjennomført på slutten av hver iterasjon og ble ofte etterfulgt av et møte der neste iterasjon ble diskutert. På disse møtene ble det også diskutert om det skulle legges til ekstra punkter under kravspesifikasjonen. En endelig evaluering kom i form av en tre timers lang utprøvning på testlaben til Haukeland med både oppdragsgiver og andre interessenter til stede.

Evalueringen av prosjektet bestod av at gruppen først viste frem programvaren. De forskjellige brukstilfellene og designvalgene som hadde blitt tatt ble gjennomgått. Første brukstilfelle som ble vist var standard oppsett for CLE-test. Her gikk gruppen gjennom hvordan brukeren kan velge en ny pasient eller velge en allerede eksisterende pasient fra databasen. Videre ble det vist hvordan man setter på forsinkelse på videostrømmen fra portrettkameraet og skopet. Det ble også gjennomgått hvordan man starter innspillingen av de tre videostrømmene.

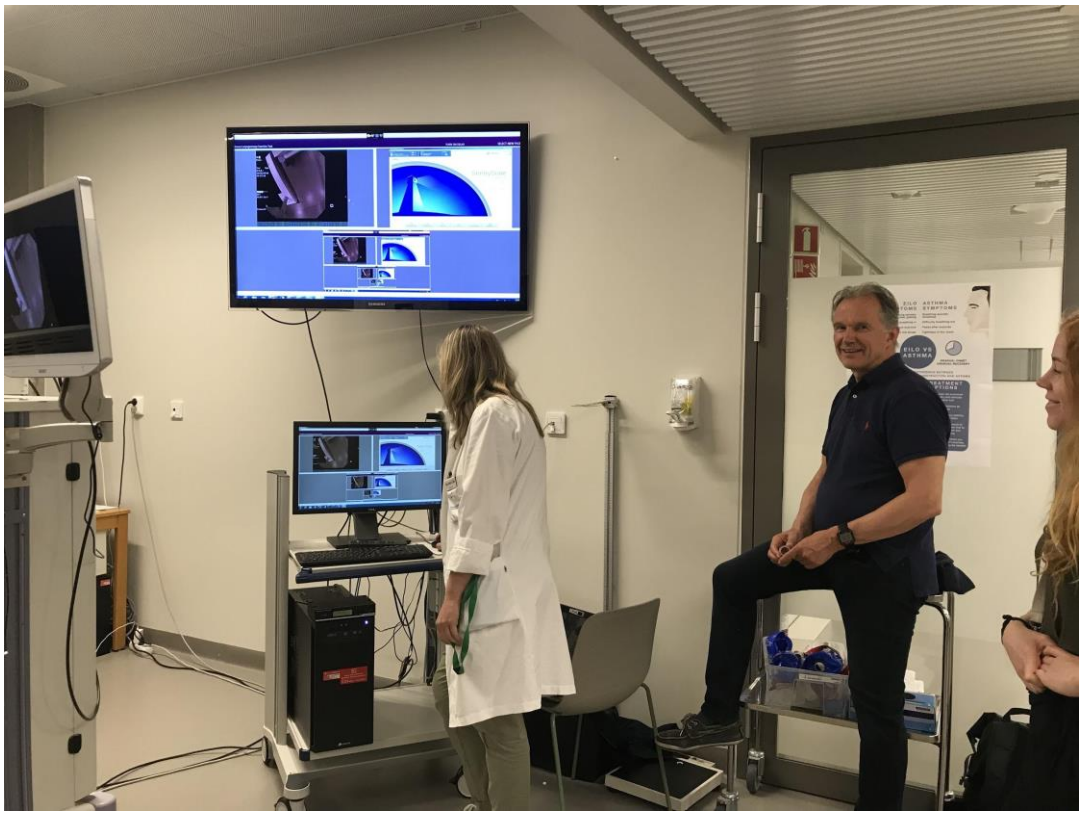
Deretter ble den delen av programmet der man kan spesifisere kameraoppsettet selv vist. Her velges det antall kamerastrømmen brukeren selv ønsker at skal vises. Denne siden har de samme funksjonene som CLE-oppsettet med forsinkelse som kan skrus av og på, valg av pasient og start/stopp av innspilling.

Det siste som ble vist var «View records». Her ble det vist frem hvordan brukeren kan velge en pasient, for så å få opp videoene som hører til pasienten. Under demonstrasjonen

ble det vist hvordan tre filmer kan bli avspilt synkronisert. Mulighetene for start, stopp og spoling ble også demonstrert.

Etter at gruppen hadde vist frem programvaren fikk de som hadde møtt opp til evalueringen prøve seg frem i programvaren. Dette ble gjort for å kunne evaluere om den nye programvaren faktisk er mer brukervennlig enn den eksisterende løsningen. Det viktigste som ble evaluert var hvorvidt brukergrensesnittet var oversiktlig. Personalet kom med god konstruktiv tilbakemelding på endringer som måtte gjøres før programvaren kunne ferdigstilles.

Det ble også testet noen nye brukstilfeller der ny teknologi ble utprøvd. Dette ble gjort for å sjekke om programvaren har større bruksområde på andre deler av Haukeland. Programvaren ble først testet ut med å koble til en BiPAP maskin. Denne maskinen har medfølgende programvaren brukt for avlesning av verdiene. Dette medfører at det må kobles inn en ekstern PC for at det skal kunne vises i programvaren som har blitt laget. Det siste som ble testet var teknologi for trykkmåling. Trykkmålingen var også avhengig av en ekstern PC grunnet medfølgende programvare.



Figur 5-1 Bilde fra utprøvning på Haukeland 20. mai. f. v Hege Clemm, Ola Røksund og Victoria Isern

5.2 Resultat fra evaluering

Etter første møte med oppdragsgiver ble det klart at ambisjonene for prosjektet var store. Det ble laget en liste med minimumskrav med store muligheter for at listen kunne utvides. Med tanke på at oppdragsgiver, i samarbeid med VIS, planlegger å kunne kommersialisere programvaren var det viktig at minimumskravene ble oppfylt. Det ble klart tidlig i utviklingsfasen at prosjektet kom til å nå minimumskravene for oppgaven. Dette medførte at oppdragsgiver kom med flere mål vi kunne jobbe opp mot.

Det at det ble gjennomført en utprøvning av programvaren ga oss god tilbakemelding fra oppdragsgiveren på arbeidet vårt. Den viktigste tilbakemeldingen var den som omhandlet designet av brukergrensesnittet. Programvaren skal brukes til innsamling av forskningsdata og det er derfor viktig at det ikke kan oppstå misforståelser ved bruk. Dette ble løst ved å gjøre alle knapper klare og tydelige. En annen forhåndsregel som ble tatt var at program skulle vise klart og tydelig når forsinkelsen og innspillingen var på.

Det som ble klart etter evalueringen den 20. mai var nytteområdet programvaren kunne ha på andre deler av sykehuset. Ved å legge inn en HDMI-splitter i kretsløpet av koblinger kan programvaren spille inn alt som skjer på en ekstern skjerm. Dette åpner for at bruken av programvaren kan utvides enormt.

Resultatene fra evalueringen legger grunnlaget for disse konklusjonene:

- Ved å ha tatt utgangspunkt i en modulbasert arkitektur er det lett for oppdragsgiver å videreutvikle og utvide programvaren hvis de ønsker dette.
- Den nye programvaren har et mye større spenn av bruksområder enn dagens oppsett.
- Programvaren som er laget har større funksjonalitet og er mer oversiktlig enn dagens oppsett.

6 DISKUSJON

Dette kapittelet vil ta for seg en diskusjon rundt gjennomføringen av prosjektarbeidet samt en refleksjon rundt resultatet av evalueringen.

6.1 Gjennomføring av prosjektet

Etter første møte med oppdragsgiver hadde gruppen noen ideer om hvordan oppgaven skulle løses. Gruppen kunne enten fortsette å jobbe på den eksisterende programvaren, eller så kunne vi begynne helt fra bunnen av. Valget falt på å starte fra bunnen av etter råd fra oppdragsgiver. Dette ble begrunnet med at gruppen ikke måtte bruke tid og energi på å sette seg inn i store mengder gammel kildekode. Oppdragsgiver formulerte minimumskrav for oppgaven og kort tid etter dette startet arbeidet med utviklingen. Oppgaven ble utført på en iterativ og modulbasert arbeidsmetode. Dette ble gjort for å muliggjøre utvidelse av programvaren når eventuelle mål ble nådd. Ved å sette opp en GANTT-diagram fikk gruppen en god oversikt over tiden som var til disposisjon. De forskjellige oppgavene og modulene fikk veldefinerte tidsintervaller der de skulle ferdigstilles. Gjennom utviklingen av programvaren har gruppen hele tiden vært à jour. Dette er en indikasjon på at gruppen har planlagt arbeidsoppgavene og disponert den tilgjengelige tiden godt.

6.2 Teknologi

6.2.1 FFmpeg og FFME

Valget å begynne helt fra bunnen av på programvaren medførte store konsekvenser. I startfasen av prosjektet ble det brukt mye tid på å finne egnet teknologi og arkitektur for programvaren. Eksempler på dette er rammeverkene FFmpeg og FFME. FFmpeg er et svært omfattende rammeverk. Dette medførte for gruppens del at det ble brukt mye tid på å lese seg opp på syntaks og bruksområder i APIen. Et annet problem vi støtte på med FFmpeg var måten rammeverket håndterte feilmeldinger. Siden dette rammeverket er laget for å bli eksekvert i kommandolinjen, kan debugging være problematisk. Dette medførte mye prøving og

feiling for å finne riktige parametere og alternativene for programvaren. Til slutt ble det konkludert med at FFmpeg ble for omfattende å bruke til selve videostrømmingen- og lagringen, og FFME ble tatt i bruk.

En av fordelene ved å bruke FFME var at dette rammeverket er kompatibelt med .Net-rammeverket. Det var for eksempel nødvendig å skrive i unsafe C# for å lage klassen som brukes til å spille inn video. Dette var helt nytt for begge gruppemedlemmene, og denne klassen fungerer fortsatt ikke helt. Det fungerer å spille inn video fra enkelte kamerakilder, men ikke alle. Dermed blir det nevnt i kapittel 7.2 at det gjenstår litt arbeid i denne klassen. Det ansees som en ulempe at videoinnspilling er såpass vanskelig ved bruk av FFME, og det var nødvendig å få hjelp fra utenforstående for å få klassen til å delvis fungere i utgangspunktet.

FFME gir mulighet for feilhåndtering under utvikling, noe som har vært svært nyttig. Et av problemene med FFME er som tidligere nevnt at det eksisterer lite og dårlig dokumentasjon for brukere som ikke kan FFmpeg godt nok til å forstå hva som menes med de forskjellige metodenavnene for FFME. Det virker også som om det ikke er spesielt mye brukt i og med at det har vært vanskelig å få svar på spesifikke problemer med koding. Dette førte til at gruppen måtte tilegne seg kunnskaper om FFME ved prøving og feiling, noe som har vært svært tidkrevende. Men til slutt har resultatet av bruken av biblioteket vært bra, og arbeidet var verdt det. Det har spart gruppen for mye tid å ha tilgang til et videostrømningsbibliotek på et høyere nivå.

6.2.2 Arkitektur og kodestruktur

Dette er den første gangen gruppemedlemmene har utviklet et så stort prosjekt, og hvor spesielt mangel på erfaring med planlegging av kodearkitektur har vært en utfordring. Etterhvert som prosjektet vokste og begynte å ta form førte dette til revidering av kodearkitektur, og ofte i etterkant av implementering. Dette har om ikke annet vært en svært nyttig erfaring som gruppemedlemmene tar med seg videre. Programvarens arkitektur er nå tydelig og fungerer bra. Dersom det hadde vært mer tid igjen for

videreutvikling ville det ha blitt brukt ressurser på å forbedre arkitekturen ytterligere. Det gjenstår også å flytte en god del metoder over i egne klasser. Det ble mye kode i Controller-modulen, som for ordens skyld hører hjemme i Helper-modulen. Dette er viktig å få gjort fordi koden da vil bli mye lettere å lese og forstå for eventuelle videreutviklere. Det ligger i planene til oppdragsgiver å faktisk utvikle programvaren videre enten med eller uten den nåværende prosjektgruppen, og dermed vil det settes fokus på kodeopprydning når fristen for prosjektslutt nærmer seg.

6.2.3 SqlClient versus LINQ

Da databasen var ferdigstilt, begynte arbeidet med å få kommunisere med den fra WPF-programmet. I begynnelsen ble det parallelt utviklet kommunikasjon ved bruk av både System.Data.SqlClient og LINQ, for å teste hva som ble det mest praktiske. Ettersom det var såpass lite informasjon som skulle lagres i databasen med en så enkel struktur, ble det konkludert med at SqlClient var enklest å bruke ettersom det er så lettvint og brukervennlig. LINQ ville krevd mer forkunnskap for å bli brukt i dette prosjektet, og det ble nedprioritert å sette av tid til å lære seg det. Dette er noe som i etterkant kan vise seg å ha vært en naiv løsning. Dersom databasen senere skal utvikles til å omfatte flere entiteter og avhengigheter, kan det vise seg mye mer praktisk å bruke LINQ, og det vil kreve mer arbeid å gå over til det. Men ettersom koden er modulbasert, vil det ikke ha noen innvirkning på de andre modulene.

6.3 Konsekvenser

Valget av arbeidsmetode medførte en del konsekvenser. Siden gruppen valgte en modulbasert fremgangsmåte, var det alltid en versjon av programvaren som oppdragsgiver kunne se. Denne fremgangsmåten gjorde det også lettere å endre på spesifikke moduler av programmet, uten særlig påvirkning på resten av programvaren. Dette forenklet også arbeidet rundt feilsøking under utviklingen av programmet. Da en ny modul ble implementert og programvaren stoppet å fungere, var det helt klart hvor feilen lå. Ved å dele opp prosjektet i veldefinerte sprinter var

det aldri et spørsmål hva som skulle gjøres. I en sprint ble det jobbet med én spesifikk oppgave, og man begynte ikke på nye oppgaver før den allerede påbegynte oppgaven var ferdigstilt.

Valgene rundt teknologi har også hatt innvirkning på prosjektarbeidet. Det ble brukt mye tid i starten av prosjektet på å sette seg inn i FFmpeg. Det ble som tidligere nevnt at denne teknologien var for omfattende. Resultatet av dette var at mye tid ble brukt på å sette seg inn i et rammeverk som er minimalt i bruk i den endelige løsningen. Hadde gruppemedlemmene hatt større kjennskap til håndtering av multimedia ved prosjektstart, kunne dette mest sannsynlig ha blitt unngått.

6.4 Drøfting av resultat

Resultatet av evalueringen har blitt beskrevet tidligere i 5.2. Med utgangspunkt i informasjonen som kom frem er vår oppfatning at programvaren kan ha stor verdi for oppdragsgiver. Det kan trekkes følge konklusjoner fra evalueringen:

Ved å ha tatt utgangspunkt i en modulbasert arkitektur, er det lett for oppdragsgiver å videreutvikle og utvide programvaren dersom de ønsker dette.

Hvis oppdragsgiver ønsker å videreutvikle programvaren, skal dette være lett å gjøre. Ved bruk av modulbasert arkitektur vil senere utviklere kunne legge til nye funksjonaliteter, teoretisk sett uten påvirkning på de allerede eksisterende modulene. Det har også blitt brukt en del hjelpeklasser for logikk og metoder som er felles for flere klasser, eller som har blitt flyttet ut av klasser for ordens skyld. Dette fører også til at det vil bli lettere for eventuelle senere utviklere å ta i bruk koden uten å måtte tenke på en del av operasjonene.

Den nye programvaren har et mye større spenn av bruksområder enn dagens oppsett.

Programvaren har allerede et sett med nye interessenter ved Haukeland, og det har blitt bevist at det vil fungere for dem å bruke den. Det er spådd at interessentgruppen vil vokse i takt med videreutviklingen av programvaren.

Programvaren som er laget har større funksjonalitet og er mer brukervennlig enn dagens oppsett.

Minimumskravet til dette prosjektet var å lage en programvare med samme funksjonalitet som den eksisterende, men med bedre brukervennlighet. Det ble tidlig klart at dette kom til å bli oppnådd, og derfor ble kravspesifikasjonene også utvidet. Det ble implementert en løsning med støtte opp til 9 kamerastrømmer. Det er også lagt inn funksjonalitet for å kunne se på tidligere innspilte videoer. Grunnet nye regler for håndtering av persondata ble det også implementert en databaseløsning. Her har brukeren mulighet til å enten velge en eksisterende pasient, eller opprette en ny pasient ved første besøk. Brukergrensesnittet i den nye programvaren er også mer intuitivt. Etter gjennomført evaluering var det fortsatt små endringer som måtte gjøres. Disse endringene omhandlet utseende på knapper og andre små endringer.

6.5 Refleksjon

Dette har vært et utrolig spennende og lærerikt prosjekt på veldig mange måter. Det har vært vanskelig og utfordrende, og det har til tider vært langt utenfor pensum som gruppen har vært gjennom hittil i utdanningen. Men det har også vært veldig givende, ikke bare med tanke på læringsutbyttet, men også for programvarens formål. Det å jobbe så tett på en oppdragsgiver som består av virkelig dyktige og motiverte ansatte ved Haukeland, har igjen motivert og inspirert gruppen til å jobbe ekstra hardt med prosjektet.

Gruppen har også fått en svært spesiell mulighet til å få arbeide med medisinsk-teknisk utstyr, og fått et innblikk i bruken av dette ved et sykehus. Prosjektet har gått innom et mangfoldig spekter av områder innen programutvikling, og gruppen har drevet fullstack utvikling. Gruppen har måttet tilegne seg kunnskap på et ganske detaljert nivå rundt videostrømming- og innspilling, og lært seg avansert .Net-programmering. Grafisk design og brukeropplevelse har også vært et viktig tema. Gruppen hadde lite kunnskap om dette fra tidligere og har derfor måttet være kreative.

Programvaren er fortsatt ikke helt ferdigstilt, men ettersom gruppen regner med å ferdigutvikle den i løpet av den resterende tiden ansees prosjektet som vellykket.

Programvaren vil sannsynligvis ikke bare bli brukt av oppdragsgiver, men også av flere andre interessenter ved Haukeland. Det er allerede bevist at programvaren vil fungere til

deres bruksmål, ettersom man per nå kan ta inn et ønsket antall videosignal. Da kan de ta inn en screencap av en ekstern datamaskin, der programmene deres kjører. Dette er jo ikke en optimal løsning, men svært anvendelig ettersom de er på utkikk etter funksjonaliteten programvaren allerede har.

7 KONKLUSJON OG VIDERE ARBEID

Dette kapittelet vil ta for seg konklusjonen av arbeidet som har blitt utført i prosjektperioden. Gruppen vil også komme med innspill til hva som kan gjøres for å videreutvikle programvaren.

7.1 Konklusjon

Hovedmålet til dette prosjektet var å utvikle en programvare med samme funksjonalitet som den eksisterende programvaren på Haukeland. Basert på resultater fra evalueringen kan minimumskravene ansees som fullført. Programvaren tar inn 3 forskjellige kamerastrømmer og viser disse. Det er også lagt inn mulighet for å skru av og på forsinkelse av kamerakildene som trenger dette. I tillegg er det nå støtte for lagring i database, noe som er nødvendig med nye regler rundt GDPR. Her kan brukeren velge om de skal bruke en allerede eksisterende pasient, eller opprette en helt ny en. Programvaren har nå også støtte for å vise valgfritt antall kamerakilder. Dette er en tilleggsfunksjon som den eksisterende programvaren ikke har. Dette åpner også mulighetene for at programvaren kan brukes av andre interessenter som samler inn digital data i form av video. Sist, men ikke minst, er det nå også mulighet for å spille av video i programvaren. Det er tiltenkt at brukeren skal kunne bruke programvaren for å sammenlikne videoer fra tidligere besøk.

7.2 Videre arbeid

For det første, er ikke programvaren helt klar for bruk på CLE-laben helt enda. Men det står klart for gruppen hva som gjenstår av arbeid, og man regner med at det vil bli ferdig innen et satt tidspunkt. Ettersom gruppen nå har såpass mye erfaring innen utviklingen av programvaren, vil det gå relativt fort å utvikle de siste detaljene.

Det som gjenstår er som tidligere nevnt for det første å gjøre det mulig å spille inn video fra alle typer kamerakilder. Dette er det som blir vanskeligst å gjennomføre, da klassen for videoinnspilling skrives i unsafe C#. Videre må gruppen bruke FFmpeg til å konkatenere

enkelte videofiler i ettertid av innspilling. Dersom brukeren skrur av eller på forsinkelsen satt på videostrømmene under innspilling, blir de lagrede videoene delt opp deretter. Men programvaren tar sikte på å lagre dette som sammenhengende video, og dermed må dette ordnes i ettertid. Utover dette gjenstår det litt småting, som å justere litt på brukergrensesnitt og litt databaselogikk rundt lagring av videoer fra programvaren.

Det er også slik at oppdragsgiver har et sett av ønsker for programvaren som går utenfor kravspesifikasjonen. I startfasen av prosjektet ble det holdt et møte mellom gruppen og oppdragsgiver der dette ble diskutert. Da ble det en enighet om at gruppen først skulle fokusere på en grunnpakke av funksjonalitet, som er den som står i kravspesifikasjonen. Dersom de klarte å ferdigstille all denne funksjonaliteten og fortsatt hadde mer tid igjen for prosjektarbeid, skulle de gå videre til en liste av ønsket tilleggsfunksjonalitet.

Men det viste seg at prosjektet var svært stort i utgangspunktet. Som tidligere vist i GANTT-diagrammet, ble det ikke regnet med at grunnpakken ville stå klar i tide til å videreutvikle programvaren til å inneholde tilleggsfunksjonaliteten. Det var derfor aktuelt med sommerjobb hos Haukeland, men det hadde gruppen dessverre ikke mulighet til. Det eksisterer fortsatt en mulighet for gruppen til å videreutvikle programmet i forbindelse med masteroppgaven deres, men det gjenstår å se om dette da blir aktuelt.

Denne videreutviklingen består da i å utvikle programvaren til å imøtekomme ønsket funksjonalitet fra andre interessenter ved Haukeland utover oppdragsgiver. Alle de nåværende interessentene er innen luftveisfysiologi. Men det har også vært snakk om andre, potensielle interessenter innen andre avdelinger på Haukeland. Uansett er ønsket om fremstilling av video felles for alle, og med litt enkel tilleggsfunksjonalitet spesielt tilpasset deres bruksområde. Programvaren kan dermed bli svært stor, og det hadde vært mulighet for å videreutvikle den i veldig lang tid.

8 LITERATUR OG KILDER

1. Agile Alliance (u. å) What is Agile Software Development? [Internet] tilgjengelig fra: < https://www.agilealliance.org/agile101/> [Lest 28. mai 2019].
2. Agile Alliance (u. å) 12 Principles Behind the Agile Manifesto [Internet] tilgjengelig fra: <https://www.agilealliance.org/agile101/12-principles-behind-the-agile-manifesto/> [Lest 28. mai 2019].
3. Atlassian (u. å) scrum [internett] tilgjengelig fra: <https://www.atlassian.com/agile/scrum> [Lest 03. mai 2019].
4. FFmpeg (u. å) About FFmpeg [internett] tilgjengelig fra: <https://ffmpeg.org/about.html> [Lest 03. mai 2019].
5. Geeks for Geeks (u. å) Software Framework vs Library [Internett] tilgjengelig fra: <https://www.geeksforgeeks.org/software-framework-vs-library/> [Lest 23. mai 2019].
6. Geeks for Geeks (u. å) Computer networks | User Datagram Protocol (UDP) [Internet] tilgjengelig fra: < https://www.geeksforgeeks.org/computer-network-user-datagram-protocol-udp/> [Lest 29. mai 2019].
7. Git (u. å) What is a git branch [internett] tilgjengelig fra: <https://git-scm.com/book/en/v1/Git-Branching-What-a-Branch-Is> [Lest 15. mai 2019].
8. Heimdal, J-H. Roksund, O D. Halvorsen, T. Skadberg, B T. Olofsson, J (2009) Continuous Laryngoscopy Exercise Test: A Method for Visualizing Laryngeal Dysfunction during Exercise, The Laryngoscope, 116, 1, s 52-57. tilgjengelig fra: <https://onlinelibrary.wiley.com/doi/full/10.1097/01.mlg.0000184528.16229.ba> [Lest 15. mai 2019].
9. Helsebiblioteket (07. juni 2016) Kvalitativ metode [internett] tilgjengelig fra: <https://www.helsebiblioteket.no/kunnskapsbasert-praksis/kritisk-vurdering/kvalitativ-metode> [Lest 15. mai 2019].
10. Helseforskning (u. å) Regionale komiteer for medisinsk og helsefaglig forskningsetikk [internett] tilgjengelig fra: <https://helseforskning.etikkom.no/prosjekterirek/prosjektregister/prosjekt?p_document_id=37472&p_parent_id=38422&_ikbLanguageCode=n> [Lest 15. mai 2019].
11. Helsekompetanse (u. å) BiPAP | Pust [Internett] tilgjengelig fra: <http://pust.helsekompetanse.no/nb/content/bipap> [Lest 21. mai 2019].
12. Microsoft (11. april 2016) Introduction to WPF [internett] tilgjengelig fra: <https://docs.microsoft.com/en-us/visualstudio/designers/introduction-to-wpf?view=vs-2019> [Lest 05. mai 2019].
13. Microsoft (u. å) Editing code in Visual Studio IDE [internett] tilgjengelig fra: <https://visualstudio.microsoft.com/vs/features/ide/> [Lest 05. mai 2019].
14. Microsoft (03. mars 2017) XAML overview (WPF) [internett] tilgjengelig fra <https://docs.microsoft.com/en-us/dotnet/framework/wpf/advanced/xaml-overview-wpf> [Lest 15. mai 2019].

15. Microsoft (u. å) What is .Net? An open-source developer platform [internet] tilgjengelig fra: <https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet> [Lest 15. mai 2019].
16. Microsoft (05. april 2019) A Tour of C# - C# Guide [Internett] tilgjengelig fra: <https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/index> [Lest 21. mai 2019].
17. Microsoft (u. å) SQLServer 2017 på Windows og Linux [Internett] tilgjengelig fra: <https://www.microsoft.com/nb-no/sql-server/sql-server-2017> [Lest 23. mai 2019].
18. Microsoft (20. juli 2015) Delegates - C# Programming Guide [Internett] tilgjengelig fra: <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/delegates/> [Lest 23. mai 2019].
19. Microsoft (31. mai 2018) DirectShow - Windows applications [Internett] tilgjengelig fra: <https://docs.microsoft.com/en-us/windows/desktop/directshow/directshow> [Lest 23. mai 2019].
20. NHI (u. å) Spirometri [Internet] tilgjengelig fra: < https://nhi.no/sykdommer/lunger/undersokelser/spirometri/> [Lest 20. mai 2019].
21. Statpead (09. oktober 2017) EILO (Excercised induced laryngeal obstruction VCD (Vocal cord dysfuntion) [internett] tilgjengelig fra: <http://www.statped.no/fagomrader-og-laringsressurser/sprak-og-tale/stemmevansker/stemmevansker/ulike-stemmediagnoser/eilo-exercised-induced-laryngeal-obstruction-vcd-vocal-cord-dysfunction/> [Lest 15. mai 2019].
22. Store Norske Leksikon (28. april 2014) codec [internett] tilgjengelig fra <https://snl.no/codec> [Lest 14. mai 2019].
23. Store Norske Leksikon (28. oktober 2016) Frontend [Internett] tilgjengelig fra: <https://snl.no/Frontend> [Lest 23. mai 2019].
24. Unosquare (u. å) ffmediaelement/README at master [internett] tilgjengelig fra: <https://github.com/unosquare/ffmediaelement/blob/master/README.md> [Lest 05. mai 2019].
25. VIS innovasjon (u. å) *Om oss* [internett] tilgjengelig fra: <https://www.visinnovasjon.no/om-oss/> [Lest 02. april 2019].
26. w3schools (u. å) XML introduction [internett] tilgjengelig fra: <https://www.w3schools.com/xml/xml_whatis.asp> [Lest 15. mai 2019].
27. w3schools (u. å) Java Wrapper Classes [Internett] tilgjengelig fra: <https://www.w3schools.com/java/java_wrapper_classes.asp> [Lest 23. mai 2019].
28. West Pead Research (u. å) ABOUT US [internett] tilgjengelig fra: <https://www.westpaed.com/about-us> [Lest 03. mai 2019].

9 APPENDIX

9.1 Ordliste

Codec - programvare og/eller elektronikk brukt for konvertering av lyd, tale og video mellom analoge og digitale signaler. (Store Norske Leksikon, 2014)

Branch - Brukes i versjonskontrollsystemet Git og er et øyeblikksbilde av endringene i prosjektet. (Git, u. å)

.Net - Et rammeverk utviklet av Microsoft for utvikling av applikasjoner på kryss av plattformer. (Microsoft, u. å)

Mediaelement - her refererer vi til FFME (FF Media Element), som er et GUI-element brukt til strømming av video.

Relasjonsdatabase - En database hvor data er organisert som rader og tabeller. Det er relasjoner mellom data som lagres i tabellene.

Fullstack - begrep som omhandler hele teknologihierarkiet i en programvare.

Frontend - Delen av programvaren som ligger nærmest brukeren, ofte kalt brukergrensesnitt (Store Norske Leksikon, 28. oktober 2016).

Delegatfunksjon - En delegatfunksjon er en referanse til en metode med en gitt parameterliste og returtype. Delegater blir brukt for å sende metoder som parametere til andre metoder (Microsoft, 20. juli 2015).

DirectShow - En API fra Microsoft brukt for inn- og avspilling av video og lyd (Microsoft, 31. mai 2018).

Rammeverk - En ferdiglaget plattform med generisk funksjonalitet som brukeren selektivt kan implementere etter eget ønske (GeeksForGeeks, u. å).

Kodebibliotek - En samling av klasser og metoder som tilbyr ferdigdefinerte hjelpefunksjoner klare til bruk (GeeksForGeeks, u. å).

Wrapper-klasse - En klasse som muliggjør bruken av primitive datatyper som objekter (w3schools, u. å).

Spirometri – Spirometri er en undersøkelse som måler to forskjellige egenskaper ved

lungene: Mengden (volum) luft som pustes ut og strømningshastigheten til denne luften (flow) (NHI, u. å).

9.2 Akronymer

EILO - Exercised Induced Laryngeal Obstruction er pustevansker som gir problemer på innpust. (Stadpead, 2017)

CLE - Continuous Laryngoscopy exercise er en standard tredemølle belastningstest der strupen kan observeres kontinuerlig. (Helseforskning, u. å)

HVL - Høgskulen på Vestlandet

XAML - Extensible application markup language er et markeringsspråk utviklet for Microsofts .Net plattform (Microsoft 03. mars 2017)

FFME - Fast Forward Media Element er en avansert erstatning for Microsoft sitt MediaElement (Unosquare, u. å).

FFMPEG - Fast forward moving picture experts group er et rammeverk som brukes for håndtering av multimedia. (FFMPEG, u. å)

XML - Extensible markup language er et markeringsspråk laget for å lagre og transportere data. (w3schools, u. å)

WPF - Windows Presentation Foundation er et rammeverk utviklet av Microsoft for å lage brukergrensesnitt. (Microsoft, 11. april 2016)

IDE - Integrated development environment er en programvare brukt til å lage annen programvare.

BiPAP - Bi-level Positive Airway Pressure er en maskin som presser luft ned i lungene ved kronisk pustesvikt. (Helsekompetanse, u.å).

RDP - Remote Desktop Protocol er en protokoll laget av Microsoft for tilkobling til en annen, ekstern datamaskin over Internett.

GUI - Graphical User Interface er et brukergrensesnitt som lar brukeren interagere med datamaskinen.

UDP – User Datagram Protocol er en meldingsorientert nettverksprotokoll. (Geeks for Geeks, u. å)

9.3 De 12 prinsippene for Agile programutvikling

Agile programvareutvikling er basert på disse 12 prinsippene (Agile Alliance, u. å):

1. At kunden er fornøyd er høyeste prioritet. Dette skal oppnås gjennom tidlig og kontinuerlig leveranse av verdifull programvare.
2. Vær åpen for endringer i kravspesifikasjonen, selv sent i utviklingen.
3. Lever fungerende programvare ofte. Tidsperioden kan varier fra et par uker opp til et par måned. Kortere perioder er å foretrekke.
4. Forretningsfolk og utviklere må jobbe sammen daglig gjennom prosjektets levetid.
5. Bygg prosjektet rundt motiverte individer. Gi de miljøet og motivasjonen de trenger, og stol på at de får jobben gjort.
6. Den mest effektive måten å formidle informasjon på i en prosjektgruppe er gjennom beskjeder ansikt til ansikt.
7. Fungerende programvare er beste måten å måle fremgang på.
8. Agile oppfordrer til bærekraftig utvikling. Alle parter som er involvert bør kunne opprettholde et konstant tempo.
9. Kontinuerlig fokus på teknisk ekspertise og godt design fremmer smidighet.
10. Enkelhet, kunsten å maksimere mengden av arbeid som ikke blir gjort, er essensiell.
11. De beste arkitekturene, spesifikasjonene og designet kommer fra selvorganiserte prosjektgrupper.
12. Prosjektgruppen søker kontinuerlig etter måter og bli mer effektive på, og endrer arbeidsmønsteret etter dette.

9.4 Risikoliste

Risikoanalyse:

S: Sannsynlighet for at hendelsen inntreffer.

K: Konsekvenser ved at hendelsen inntreffer.

RF: Risikofaktor. Sannsynlighet multiplisert med konsekvens.

1-5: 1 er liten sannsynlighet og konsekvens av uønsket hendelse inntreffer. 5 er stor sannsynlighet og konsekvens av uønsket hendelse inntreffer.

Suksessfaktorer	S	K	RF	Tiltak
Applikasjonen kan ikke brukes av oppdragsgiver	2	5	10	Opprettholde god kommunikasjon med oppdragsgiver og drive kontinuerlig testing.
Misforstått krav og spesifikasjoner	2	5	10	Skrive ned alt som blir sagt på møter og spørre hvis det er noe som ikke blir forstått.
Skjev arbeidsfordeling innad i gruppen	3	4	12	Opprettholde god kommunikasjon. Viktig å oppdatere seg på hva som jobbes på så man ikke gjør dobbelt arbeid.
Fravær/sykdom	2	3	6	Passe på å gi god beskjed hvis man ikke kan møte på avtalte tidspunkter.
Applikasjonen er ikke brukervennlig	2	4	8	Oppdaterer oppdragsgiver ofte der programmet blir vist.
Dårlig samarbeid	1	5	5	Gi gode og ærlige tilbakemeldinger på hvordan arbeidet går.
Mangel på kompetanse	3	4	12	Fokusere på å lære seg de aktuelle rammeverkene og teknologiene.
Valg av feil rammeverk eller utviklingsmiljø	2	5	10	Lese seg god opp på teknologien som skal brukes så man ikke blir overrasket over eventuelle begrensninger.
Dårlig håndtering av tid	3	5	15	Ikke la fritid gå utover arbeidstiden og ikke utsette ting til siste liten.
Utvikling av applikasjon går utover oppgaveskriving	3	4	12	Ikke ta på seg for stor arbeidsmengde og følge de oppsatte tidsfristene som er satt.

9.5 Oppgavebeskrivelsen

P4 – Lungediagnostikk (HVL / Haukeland Universitetssykehus)

Bakgrunn

På Energisenteret for Barn og Unge ved Haukeland Universitetssykehus har vi en hjerte lunge testlab hvor vi mellom annet er patenthavere og verdensledende i gjennomføring ev CLE-testen, en test for å påvise eller avkrefte «Exercise Induced Laryngeal Obstruction» (EILO). Testen går ut på å løpe en maksbelastningstest på tredemølle, med kontinuerlig overvåkning av ulike parameter fra ulike kilder. Gjennom forsknings-gruppen WestPaed Research er vi tildelt samarbeidsmidler med Olympus for softwareutvikling til innsamling og analyse av disse parametere.

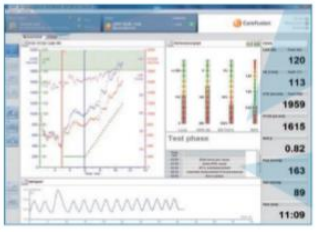
Oppgave

I dag har vi et oppsett som gjør av skjermbildene fra maskin 1-3 samles på et skjermbilde som vi tar kontinuerlig videoopptak av gjennom hele testsekvensen (15-25 minutt), og eksporterer i .avi. Deres oppgave blir å lage et nytt og mer stabilt innsamlingsoppsett enn det vi benytter i dag. Vi ønsker å trekker inn rådata fra alle aktuelle kilder og ha mulighet til å manipulere data gjennom de ulike vinduene i tenkt programvare.

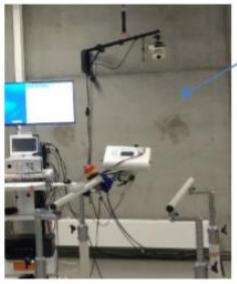
Aktuelle parameter:

- Lungekapasitet og hjerteaktivitet (EKG) (PC1)
- Videoopptak av pusterørets innside (laryngs) (PC2)
- Videoopptak av pasienten (PC3)
- Trykkmålinger (Ikke på bilder)

Gjennom dette prosjektet får dere ta et steg inn i helsevesenet og vil få god innblikk i krysningspunkt mellom klinikere, leverandører og driftsteknikk personell. Dersom resultatet av prosjektarbeidet blir vellykket vil det gjennomføres utprøving og demonstrasjon i Bergen og London, og Olympus (www.olympus.com) bidrar med promoteringsmateriell av EILO og CLE-testing, der programvaren kan få en viktig rolle i pakken



PC1



Kontaktpersoner

Lars Peder Bovim, lpb@hvl.no
Energisenteret for Barn og Unge

Haukeland Universitetssjukehus

Ola Drange Røksund
Fakultet for Helse og Sosial

Høgskulen på Vestlandet

7