Høgskulen på Vestlandet

BACHELOROPPGAVE

Mobilapplikasjon for automatisert måling av bevegelighet ved bruk av Kunstig Intelligens

Mobile application for automatic measurement of mobility using Artificial Intelligence

Ole August Solem

Henrik Vallestad

Dat191

Informasjonsteknologi

Fakultet for teknologi, miljø- og samfunnsvitskap

Institutt for datateknologi, elektroteknologi og realfag

Veileder: Per Christian Engdal
Innleveringsdato: 13.05.2024

Høgskulen på Vestlandet

Fakultet for teknologi, miljø- og samfunnsvitenskap

Institutt for datateknologi, elektroteknologi og realfag

TITTELSIDE FOR HOVEDPROSJEKT

Rapportens tittel: Mobilapplikasjon for automatisert måling av bevegelighet ved bruk av Kunstig Intelligens	*Dato: 13.05.2024*
Forfatter(e): Ole August Solem, Henrik Vallestad	*Antall sider u/vedlegg: 52*
	Antall sider vedlegg: 57
Studieretning: *Informasjonsteknologi*	*Antall disketter/CD-er: 0*
Kontaktperson ved studieretning: *Per Christian Engdal*	*Gradering: Ingen*
Merknader:	

Oppdragsgiver: Ytir	*Oppdragsgivers referanse:*
Oppdragsgivers kontaktperson: Asle Jonny Førde og Per Christian Engdal	*Telefon:*

Sammendrag:

Måling av bevegelighet er tradisjonelt en manuell prosess utført av fysioterapeuter og er utsatt for subjektive vurderinger. Prosjektet tar sikte på å utvikle et system fra grunnen av for å adressere dette problemet. Resultatet som presenteres er en mobilapplikasjon som benytter Human Pose Estimation, slik at pasienter selv kan utføre bevegelighetstester med høy nøyaktighet.

Summary:

Range of motion testing is traditionally a manual process, susceptible to subjective assessments. To address this issue, the project aims to develop a system from the ground up. The solution presented is a mobile application that utilizes Human Pose Estimation, allowing patients to perform mobility tests with high accuracy and precision.

Stikkord:

Range of motion	Human pose estimation	MoveNet	React Native
Azure	PostgreSQL	ASP .NET	Kunstig intelligens

Høgskulen på Vestlandet, Fakultet for teknologi, miljø- og samfunnsvitenskap
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

Takk til Ytir for deres bidrag og innspill.

Takk til vår veileder, Per Christian Engdal, for god veiledning med systemutviklingen og rapportskrivingen.

Videre ønsker vi å takke alle de som har hjulpet med brukertesting og gitt gode tilbakemeldinger.

Gruppen er svært fornøyd med å ha arbeidet med et så spennende og lærerikt prosjekt. Vi håper prosjektet kommer til å hjelpe Ytir eller andre som ønsker å jobbe med lignende systemer.

INNHOLDSFORTEGNELSE

ORDLISTE

ORD	FORKLARING / DEFINISJON
Array/Matrise	Data struktur som inneholder en gruppe med dataelementer av lik type.
Aspect ratio	Forholdet mellom skjermens bredde og høyde
Backend	Programvare på en server som ligger nærmest databasen.
Belastningsprofil	Profil som viser pasientens belastning ved ulike ledd.
Flexbox	En design stil som dynamisk skalerer komponenter.
Frontend	Delen av systemet som brukeren interagerer med.
Goniometer	Et verktøy som brukes av fysioteraperuter for å måle vinklutslag i kroppsdeler.
Inndata	Data som går inn til et system, eller modul
Klient	Dataprogram som kjører lokalt hos brukeren.
MacOS	Operativsystemet som er på PCer fra Apple.
Mock Database	En simulert database med samme struktur som eksisterende database.
Output	Informasjon som går ut ifra et system, eller modul
React	Et bibliotek for utvikling av web og mobile brukergrensesnitt.
Server	Programvare som tilbyr flere tjenester til andre datamaskiner.
Tensorflow	Et rammeverk for å utvikle KI Modeller.
Use case	Brukstilfelle
Vertex	Toppunkt / Vinkelspiss der to stråler møtes.
Wireframe	Visuell representasjon av tenkt design.

AKRONYMER

Akronym	Forklaring
FPS	Frames Per Second
HPE	Human Pose Estimation
HVL	Høgskulen på Vestlandet
IOS	Iphone Operating System
JSON	JavaScript Object Notation
KI	Kunstig Intelligens
MVP	Minimum Viable Product
ROM	Range Of Motion / Bevegelsesområde
UI	User interface / Brukergrensesnitt

1 INNLEDNING

1.1 Kontekst

Ytir er en oppstartsbedrift som utvikler en applikasjon for å redusere sykefraværet hos ansatte i Norge. Selskapet er en oppstartsbedrift med 1 ansatt, og ble etablert i 2020. Ytir har utviklet en metode og et digitalt verktøy for å redusere sykefraværet i bedrifter. Løsningen til Ytir har vært prøvd ut over en periode på 12 måneder i 2 ulike organisasjoner, med gode resultat.

Ved noen skader eller plager, blir pasienter rådet til å konsultere en fysioterapeut for få sin bevegelighet vurdert. Under funksjons- og bevegelighetsvurdering utføres manuelle tester, der man for eksempel måler leddutslag ved bruk av goniometer. Utslagene blir notert i en pasientjournal, og ved oppfølgende vurderinger, kan en sammenligne tidligere målinger for å vurdere en positiv eller negativ utvikling av bevegelighet.

Sentral i Ytir sin løsning, står etablering og bruk av belastningsprofiler for ansatte i sykefravær, for en nøyaktig utvelgelse av egnede arbeidsoppgaver som del av en tilrettelegging. Etableringen av belastningsprofiler gjøres i dag manuelt, og Ytir ser etter måter å automatisere denne prosessen.

1.2 Motivasjon

Måling av bevegelighet i ledd gjøres i dag av en fysioterapeut eller annen behandler. Målinger registreres under en pasients utførelse av ulike øvelser. En kategori av øvelser, er Range Of Motion (ROM), hvor behandler visuelt og ved bruk av goniometer måler oppnådd leddutslag under utførelsen. Denne metoden har sine utfordringer ved at målingene ofte blir unøyaktige og basert på subjektive vurderinger fra behandler. ROM testing krever også fysisk tilstedeværelse mellom pasient og behandler. Ved å automatisere denne prosessen vil flere ha mulighet til å testes, pasientene kan testes oftere, og det vil ikke være et krav om fysisk oppmøte.

1.3 Problembeskrivelse og mål

Tilnærmingen til bevegelighetstesting ved bruk av ROM er i stor grad manuell og subjektiv. På Fysioterapeuten.no står det dette om behandlingsprosessen for pasienter: "*Det er tidkrevende, og stiller krav både til presisjon og formuleringsevne*." (Eng-Galåen, 2024). Behandlere kan ha noe partisk oppfatning om hva som defineres som normal og lav bevegelighet. Behandlere utfører tester manuelt, noe som fører til ulik grad av presisjon og pålitelighet. I tillegg må pasientene reise fysisk til behandler for å utføre testene, noe kan være både tidkrevende og utfordrende for de med begrenset mobilitet.

Ny teknologi basert på Kunstig intelligens (KI) og Human Pose Estimation (HPE), benyttes i ulike applikasjoner innenfor sport/trening, underholdning og helse. Med HPE kan en i sanntid, ved bruk av kamera, simulere en virtuell modell av kroppen og dens bevegelser ved å plassere punkter ved ulike ledd i et koordinatsystem.

Ytir har sett inn ulike områder for om mulig å automatisere prosessen med etablering av belastningsprofiler. Et av områdene er bruk av KI og HPE. Hypotesen er at ved bruk av HPE og måling av vinkelutslag for ulike deler av kroppen, kan det indikeres en mulig profil for bevegelighet og dermed belastning. Selskapet ser også andre mulige anvendelsesområder for automatisert måling av bevegelighet, blant annet under opptrening av slagpasienter.

Dette prosjektet har som mål å utvikle en mobilapplikasjon, basert på HPE, hvor pasienter selv effektivt og med god nøyaktighet kan utføre tester og målinger av egen bevegelighet.

Applikasjonen skal ha en visuell oversikt over brukerens fremgang eller eventuelle tilbakefall. I tillegg skal applikasjonen ha samme teknologivalg som Ytirs eksisterende applikasjon, ha et brukervennlig UI-design, og følge samme systemarkitektur som det Ytir har brukt.

1.4 Oppbygging av rapporten

Kapittel 1: Introduserer det helhetlige prosjektet og gir kontekst til problemstillingen.

Kapittel 2: Introduserer leseren for prosjektet.

Kapittel 3: Diskusjon av prosjektets design.

Kapittel 4: Detaljert løsning.

Kapittel 5: Resultatene blir evaluert og presentert.

Kapittel 6: Resultatene fra tidligere kapittel blir diskutert.

Kapittel 7: Konklusjon av prosjektet og det gir nyttige tips til videre utvikling av prosjektet.

Kapittel 8: Referanser.

Kapittel 9: Vedlegg.

2 PROSJEKBESKRIVELSE

2.1 Praktisk bakgrunn

2.1.1 Tidligere arbeid

Ytir sin nåværende løsning er en mobilapplikasjon der bedrifter kan følge opp ansatte med sykefravær. Applikasjonen gir bedriftsledere mulighet til å definere mulige arbeidsoppgaver for ansatte, basert på belastningsprofiler. Oppgaver tildeles med utgangspunkt i den ansatte sin helsesituasjon.

2.1.2 Initielle krav

De initielle kravene til oppgaven er at mobilapplikasjonen skal være tilgjengelig på både iOS og Android. Ved å være tilgjengelig på begge plattformene vil applikasjonen nå ut til en større brukerbase.

Videre er det krav om at brukeren har mulighet for å måle og visualisere utslag for kroppsdeler under utførelse av ROM øvelsen, i sanntid.

Det er også krav om å visualisere resultat fra målinger av utslag etter gjennomføring av aktivitet/program. Visualiseringen av testresultatene vil gi brukeren verdifull informasjon om egen bevegelighet. Denne informasjonen kan benyttes for å identifisere behovet for behandling og opptrening.

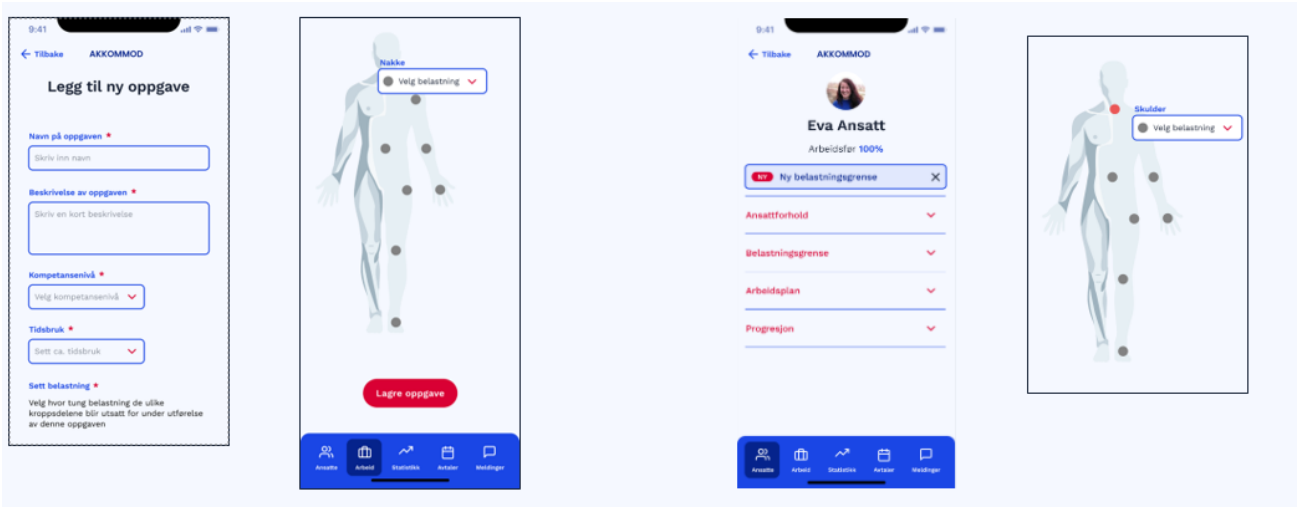
Deretter er det krav om å ha mulighet for å visualisere utvikling over tid, og se progresjon eller tilbakefall. Denne visualiseringen, kalt en *Belastningsprofil*, vil gi brukeren mulighet til å velge forskjellige tidsperioder for å enkelt følge utviklingen over tid.

Det siste kravet er at applikasjonen skal ha akseptabel responstid. Når brukere har ubegrenset tilgang til apper, er det viktig å ha en app som er rask, responsiv og feilfri. Applikasjoner med dårlig ytelse kan føre til frustrasjon, tapte brukere og negative anmeldelser (Korvyan, 2024)).

2.1.3 Initiell løsnings-idé

Den initielle ideen bak prosjektet er å bruke kunstig intelligens i form av en HPE modell for å utføre automatiserte bevegelighetstester. Applikasjonen utvikles som et eget prosjekt og bygges fra grunnen av. Dette inkluderer en database, REST API, en mobil applikasjon, oppkobling til en kunstig intelligens modell og kommunikasjonen mellom alle lagene.

Det er lagt opp til at gruppen selv kan utforske og utvikle et design. Samtidig ble det utdelt skisser av Ytir sine UI-design. I Figur 2-1 står det AKKOMOD, dette var bedriften sitt navn før de ble Ytir, her vises designstilen til Ytir.



Figur 2-1: Ytir Design

2.2 Avgrensninger

Dette prosjektet har avgrensninger som er viktig å ta hensyn til. Den første avgrensningen er at utviklingen av et innloggingssystem utelukkes. Ytir har et eksisterende innloggingssystem oppkoblet til BankID.

Begrensninger i tilgjengelig tid og en endelig sluttdato gjør til at det skal utvikles en frittstående applikasjon, i form av en prototype som demonstrerer løsningen. Løsningen skal være basert på Ytir sin nåværende arkitektur og teknologi. Fokus skal være på kjernefunksjonalitet, identifisert gjennom brukseksemplene.

Det skal ikke lagres personsensitive data i løsningen, slik at det ikke settes spesielle krav til sikkerhet i form av pålogging eller kryptering av data. Sikkerhet vil i dette prosjektet ikke bli prioritert fremfor utviklingen av et Minimum Viable Product (MVP).

2.3 Ressurser

Deltakere i gruppen har hver sin PC med Windows operativsystem, pc-ene vil bli brukt til å utvikle applikasjonen, drive testing av applikasjonen, tilegne oss nyttig informasjon, designe skisser, og skrive rapport.

Via Høgskulen på Vestlandet (HVL) er det muligheter for sitteplasser med eksterne skjermer. Dette benyttes ved faste tidspunkter slik at gruppen holder en jevn flyt over utvikling- og skrive prosessen. Ved behov for opptrening av egen KI modell er et grafikkort av typen RTX4070 tilgjengelig. Til slutt vil kilder på internett brukes for å skaffe relevant kunnskap til prosjektet.

Gruppens veileder, Per Christian Engdal, er åpen for å gi tilbakemeldinger eller foreslå endringer for å forbedre rapporten og systemets struktur. Oppdragsgiver Asle Jonny Førde er tilgjengelig for faglige spørsmål angående fysioterapi.

2.4 Litteratur om problemstillingen

2.4.1 Beveglighet

Store norske leksikon definerer bevegelighet som å ha muligheten til å utføre bevegelsesutslag i ledd. De skiller deretter mellom to typer bevegelighet, aktiv og passiv. Aktiv beskrives som leddutslaget pasienten klarer å utføre ved bruk av egen muskulatur. Imens passiv beskrives som leddutslaget pasienten kan utføre ved hjelp av ytre kraft (Alvær & Straume-Næsheim, 2023).

2.4.2 Range Of Motion

ROM brukes for å måle hvor stort utslag en pasient har i leddene og er en fellesbetegnelse på ulike tester for å vurdere bevegelighetsevnen til en pasient. ROM kan måles aktivt ved at pasienten utfører øvelser på egenhånd, eller passivt ved hjelp av en behandler. ROM-øvelser gir målinger basert på pasientens utførelse i forhold til normal bevegelighet (Sears, 2023).

2.4.3 Human Pose Estimation

HPE er en metode for å predikere og visualisere bevegelsene til mennesker ved bruk av et kamera. Systemet som utfører HPE, analyserer bilder fra kameraet og identifiserer nøkkelpunkter på kroppen, som ledd, bryst, mage og rygg.

HPE kan utføres i sanntid ved at en videostrøm sendes til modellen samtidig som den predikterer posituren. En annen metode er å laste opp en video til en server for analyse. De to ulike måtene å drive HPE på har fordeler og ulemper. Ved HPE i sanntid får man se direkte hvordan modellen estimerer og man unngår komplikasjoner som medfølger opplastning av video, men dette kommer på bekostning av nøyaktighet (Odemakinde, 2024). Servere som håndterer HPE kan benytte en kraftigere modell som gir mer presise resultater.

2.4.4 KI på mobile enheter

KI er et samlebegrep for ulike teknologier og metoder som gir maskiner mulighet til å simulere menneskelig intelligens. Denne simuleringen er basert på tidligere informasjon og gjør det mulig for maskiner å utføre oppgaver av omfattende kompleksitet som tidligere ville krevet menneskelig intelligens (Cao, et al., 2019).

Det finnes ulike verktøy som er designet for å implementere KI-modeller til bruk på mobile enheter. Ved bruk av slike verktøy blir det mer intuitivt for utviklere å integrere KI i sine applikasjoner. Oppgaver som kan utføres ved bruk at KI er eksempelvis bildegjenkjenning, talegjenkjenning og HPE.

Den største forskjellen ved bruk av KI på mobile enheter og tradisjonelle datamaskiner, er at mobile enheter har lavere prosesseringsevne enn andre datamaskiner. Datamaskiner brukt til KI har ofte kraftige grafikkort som gjør det mulig å benytte modeller som håndterer store mengder data uten at dette påvirker systemet i stor grad. Mobile enheter har varierende prosesseringsevne, enkelte med lav, og andre høyere prosesseringskraft. Det er derfor viktig å bruke modeller som tar hensyn til mobilens nedsatte regnekraft, spesielt i tilfeller der det er avgjørende med lav latens.

3 DESIGN AV PROSJEKTET

3.1 Forslag til løsning

Som alternativer til modeller har gruppen valg å utforske: MoveNet Lightning, MoveNet Thunder og OpenPose. Dette er tre HPE modeller med ulik variasjon av kompleksitet der to er utviklet for mobile enheter.

3.1.1 Alternativ løsning 1 – MoveNet Lightning

Det første alternativet er HPE modellen MoveNet Lightning. MoveNet Lightning er en modell basert på MoveNet arkitekturen utviklet av Google. Modellen har hastighet som hovedfokus og oppnår en bildefrekvens på over 50FPS på de fleste mobile moderne mobiltelefoner (Tensorflow, 2022). Lightning har et presisjonsnivå på 75.1% (Jo & Kim, 2022) og plasserer 17 punkter på kroppen ved prediksjon av menneskelig positur.

3.1.2 Alternativ løsning 2 – MoveNet Thunder

Det andre alternative er MoveNet Thunder. Movenet Thunder er en modell basert på MoveNet arkitekturen, og har presisjon som hovedfokus. Thunder oppnår en bildefrekvens på rundt 30FPS på de fleste mobile moderne mobiltelefoner (Tensorflow, 2022). Modellen har et presisjonsnivå på 80.6% (Jo & Kim, 2022) og plasserer 17 punkter på kroppen i ved prediksjon av menneskelig positur.

3.1.3 Alternativ løsning 3 - OpenPose

Det tredje alternativet er OpenPose, utviklet av forskere hos Facebook (Cao, et al., 2019). OpenPose er en modell som har stort fokus på presisjon og detaljert deteksjon av menneskelige positurer. OpenPose har presisjonsnivå på 86.2% (Jo & Kim, 2022). OpenPose er en detaljert modell med 137 punkter og har mulighet for deteksjon av flere personer.

3.1.4 Diskusjon av alternativene

Verken MoveNet Lightning eller MoveNet Thunder har mulighet for HPE på mer enn én person om gangen i sin opprinnelige form, mens OpenPose støtter HPE på flere personer samtidig.

I en studie utført av BeomJun Jo og SeongKi Kim ved Sangmyung University, sammenligner de MoveNet Thunder, Lightning og OpenPose sin ytelse på bilder. Resultatet viste at OpenPose bruker 12,04 ganger lenger tid enn MoveNet Lightning på å prediktere 1000 bilder, og 4,60 ganger mer enn Thunder (Jo & Kim, 2022). MoveNet løsningene predikerer 17 punkter på kroppen i motsetning til OpenPose med 137 punkter, OpenPose vil da gi mer detaljert informasjon per bilde, men dette vil også føre til nedsatt ytelse for systemet.

3.2 Valgt løsning

Basert på sammenligningene mellom de tre HPE modellene står valget mellom, Lightning som vil gi best ytelse, men dårligst presisjon. Thunder som vil gi middels ytelse, men bedre presisjon enn Lightning, og OpenPose som vil være den mest presise modellen, men på bekostning av ytelse.

Gruppen valgte å gå for HPE modellen MoveNet Lightning. Med dagens teknologi veier hastigheten til MoveNet opp for presisjonen til OpenPose. Lightning er mer responsiv enn Thunder, noe som er avgjørende for å oppnå gode brukeropplevelser og en responsiv applikasjon på tvers av ulike mobile enheter. Presisjonen til Lightning er lavere enn de to andre, men den vil være adekvat for vårt bruksområde. OpenPose sin mulighet til å estimere flere personer samtidig vil ikke være relevant for prosjektet. Dette sammen med sine 137 punkter vil skape unødvendig kompleksitet i systemet.

3.3 Valg av verktøy

3.3.1 .Net

.Net er en utviklerplattform utviklet av Microsoft (Microsoft, 2024:b). .Net bruker C# som programmeringsspråk og benyttes i dette prosjektet til å utvikle et backend REST API som er et kommunikasjonsledd med databasen og frontend.

3.3.2 Android Studio

Android studio er et offisielt utviklingsverktøy utviklet av Google (Google, 2024), for utvikling av Android-applikasjoner. Android Studio har muligheten for å simulere en Android telefon. Videre gir Android Studio muligheten til å koble opp personlige Android telefoner, som muliggjør testing med en mobiltelefon. Android Studio sine testverktøy benyttes til testing av applikasjonen.

3.3.3 Azure PostgreSQL Database

Azure er en skyplattform fra Microsoft som tilbyr en rekke tjenester for å bygge, distribuere og administrere applikasjoner på tvers av flere rammeverk (Microsoft, 2024:a). Systemet tar i bruk Azure sin databaseløsning for å opprette en PostgreSQL database.

3.3.4 Chat GPT 3.5

Chat GPT 3.5 er en språkmodell utviklet av OpenAI (OpenAI, 2024:a). Modellen benyttes i dette prosjektet til å generere SQL kode til databasen basert på gitt tekst prompt (OpenAI, 2024:b).

3.3.5 Figma

Figma er et designverktøy som benyttes for å designe brukergrensesnittet til applikasjonen. Teknologien er utviklet av Figma Inc. (Figma, Inc., 2024) og gjør det mulig å samarbeide med å utvikle skisser, som reduserer utviklingstiden av designet.

3.3.6 Github

Github er et system for lagring og administrasjon av programkode (Github , 2024). Verktøyet bygger på Git som er et versjonskontrollsystem som lar brukere behandle og dele kildekode. Github gjør det lettvint å sette opp en felles kodebase der utviklerne kan jobbe parallelt.

3.3.7 Lucidchart

Lucidchart er et design verktøy der man kan utvikle ulike diagrammer for å organisere systemer (Lucid Software Inc., 2024). I denne oppgaven benyttes Lucidchart for å strukturere system og database arkitekturen.

3.3.8 Tensorflow Lite

Tensorflow Lite er en samling med verktøy utviklet av Google for å overføre og optimalisere Tensorflow modeller på mobile enheter (Tensorflow, 2022). Tensorflow Lite benyttes i denne oppgaven for å integrere AI modellen opp mot mobilapplikasjonen.

3.3.9 MoveNet

MoveNet er en ferdigtrent HPE modell utviklet av Google AI. MoveNet er bygget på Tensorflow og er spesielt utviklet for å være responsiv på web og mobile enheter (Tensorflow, 2021). MoveNet implementeres ved bruk av Tensorflow Lite.

3.3.10 React Native

React Native er et rammeverk for utvikling av applikasjoner på web og mobile enheter ved bruk av React (Meta Platforms, Inc., 2024). Dette prosjektet benytter React Native for å utvikle brukergrensesnittet til mobilapplikasjonen.

3.3.11 Visual Studio Code

Visual Studio Code er en kodeeditor utviklet av Microsoft (Microsoft, 2024:c). Ved å installere utvidelser kan systemet utvides til å utvikle applikasjoner med ulike programmeringsspråk. All kode i dette prosjektet skrives ved bruk av Visual Studio Code.

3.3.12 Office Pakken

Office pakken er en samling med programvareapplikasjoner utviklet av Microsoft (Microsoft, 2024:d). Verktøyene som benyttes i prosjektet, er Microsoft Word, Microsoft Excel og Microsoft PowerPoint som brukes til å samarbeide i sanntid med dokumenter tilknyttet rapporten.

3.4 Prosjektmetodikk

3.4.1 Utviklingsmetodikk

Utviklingsmetodene i dette prosjektet er en iterativ og inkrementell tilnærming, som har grunnlag i flere agile utviklings metodikker. For at prosjektet ikke skal få en fossefall-struktur, en metodikk som bryter utviklingen ned i faser som ikke gjenbesøkets (Store Norske Leksikon, 2024), er det viktig å følge agile prinsipper. Prosessene som ble valgt til dette prosjektet var en kombinasjon av utvalgte egenskaper hos Scrum.

Scrum er en metodikk som er best egnet til små team med færre enn 10 personer. Et prosjekt som følger alle stegene til Scrum-metodikken starter med en prioritert liste over funksjoner som skal utvikles, kalt product backlog. I neste steg, kalt sprintplanleggingen, velges de oppgavene fra backlogen som skal utføres. I Scrum-metodikken avholdes daglige møter der teammedlemmene oppdaterer hverandre om utført arbeid og eventuelle problemer med utviklingsprosessen. En sprint er en tidsbasert iterasjon, ofte på 2 til 4 uker, der en gruppe arbeider med definerte mål for å levere et ferdig inkrement av et fullverdig produkt. Ved faste intervall, på slutten av en sprint, møtes Scrum teamet til en gjennomgang av utført arbeid med produkteier og andre interessenter for tilbakemelding. I slutten av et utført møte, reflekterer teamet over sprinten og diskuterer forbedringspotensialer, deretter iverksettes en ny sprint (Schwaber & Sutherland, 2020).

Utviklingsmetodikken i dette prosjektet tar inspirasjon fra Scrum prosessen. Utviklingsprosessen vår er basert på sprinter med 2 ukers mellomrom. I slutten av iterasjonen utføres en sprintgjennomgang med veileder og det diskuteres fremgang og muligheter for forbedring.

Parprogrammering er en utviklingsteknikk der to utviklere samarbeider på en datamaskin. En navigatør og en fører. Navigatøren reflekterer over den eksisterende koden, gir forslag til forbedring, og planlegger den fremtidige utviklingen av prosjektet. Føreren fokuserer på programmeringen, og håndterer den aktuelle problemstillingen, uten å tenke over det helhetlige perspektivet. Parprogrammering er vist å redusere sannsynlighet for feilkode (Paula, 2024).

Gruppen benyttet parprogrammering da det skulle utforskes ny kode, eller i sitasjoner hvor ulike versjoner i Github skulle synkroniseres.

3.4.2 Prosjektplan

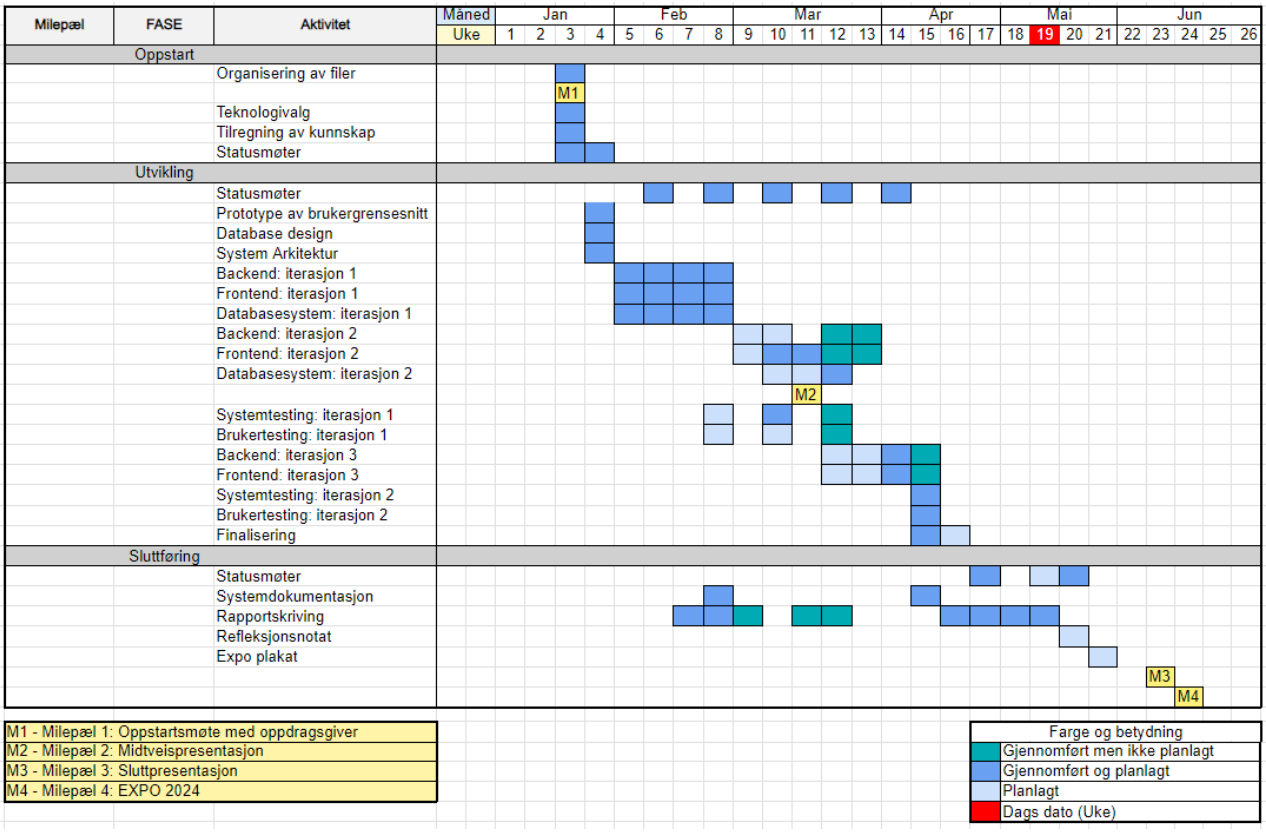
Gruppen utviklet en prosjektplan i Microsoft Excel i form av et Gantt diagram. Gantt diagram er et diagram som viser ulike aktiviteter og iterasjoner, samt faser som må fullføres i et prosjekt. Figur 3-1 viser diagrammet med aktivitetene til venstre og søylediagrammet til høyre. Formålet med diagrammet er å strukturere en overordnet plan for prosjektet. Planen skal gi indikasjoner på fremdrift og identifisere nødvendig fremtidig arbeid. Fargene og deres betydning illustreres i tabellen under Gantt diagrammet.

Oppstart, er første fase og indikerer oppstartsfasen i prosjektet. Denne fasen omfatter strukturering av prosjektet, og danner grunnlaget for utvikling av applikasjonen.

Utvikling, er andre fase og indikerer utviklingsfasen i prosjektet. Fasen dekker design og utvikling av systemet, samt ulike former for testing. I denne fasen er det stort fokus på system og applikasjonsutvikling.

Siste fase i Gantt diagrammet er Sluttføring. Denne fasen omfatter utvikling av rapporten. I denne fasen er det stort fokus på rapportskriving og sluttføring av dokumentasjon.

En milepæl er en viktig hendelse som markerer fremskritt i prosjektet. Figur 3-1 viser en oversikt over alle milepælene til prosjektet. Milepælene representerer blant annet sprinter og utkast av viktige dokumenter, milepælene er definert under Gantt diagrammet.



Figur 3-1: Gantt Diagram

3.4.3 Risikovurdering

Figur 3-2 viser risikoanalysen for prosjektet, i risikoanalysen gjør gruppen en vurdering over situasjoner eller hendelser som kan påvirke prosjektet negativt. Hendelsene blir tildelt et tall fra 1 til 5 på hvor sannsynlig og hvor stor konsekvens en hendelse har på prosjektet. I tillegg til dette oppgis det hva en mulig årsak til en hendelse kan være, og det presenteres forslag til forebyggende tiltak.

Risikomatrisen i Figur 3-3 brukes til å beregne risiko-produktet for hver risiko. Risiko-produkt er et produkt av sannsynlighet og konsekvens. I tilfeller der risiko-produktet har høy verdi er det viktig at gruppen utvikler en forbyggende plan for risikoen. Formelen som benyttes er: $S \cdot K = R$, hvor S representerer sannsynlighet, K konsekvens og R risiko-produkt.

Risikoen *Applikasjonen er ikke brukervennlig* har risikoprodukt på 15 og er en av de to største risikoene. Manglende brukervennlighet kan ha negative effekter på produktet, for eksempel økt frustrasjon hos brukere, og vil dermed påvirke brukeropplevelsen. Konsekvensen av denne risikoen er vurdert til å være 5. Gruppen har lite tidligere erfaringer med utvikling av brukervennlige applikasjoner, som øker sannsynligheten for risikoen til 3. Gitt det høye risikoproduktet, er det viktig å ha samtale med brukere og opprette et oversiktlig brukergrensesnitt for å forbygge at risikoen blir reell.

Modellens målinger er utilstrekkelige er den andre betydelige risikoen, med risikoprodukt på 15. Dersom applikasjonens modell feiler i å gi presise prediksjoner, vil applikasjonen miste kjernefunksjonaliteten. Konsekvensen til risikoen blir derfor vurdert til 5. Gruppen mangler erfaring med implementasjon av KI på mobile enheter, og sannsynligheten for risikoen blir satt til 3. Forbyggende tiltak til denne risikoen vil være å starte utforsking av ulike modeller for KI på mobile enheter tidlig i prosjektet, for å unngå tidspress og uoppnådd kjernefunksjonalitet.

Hendelse / Risiko	Årsak	Sannsynlighet	Konsekvens	Risiko- produkt		Forebyggende tiltak
Oppdragsgiver/veileder er ikke fornøyd med løsning	Mangelfullt arbeid, utilstrekkelig kommunikasjon med oppdragsgiver/veileder, ikke brukervennlig design.	2	2	Lav	4	Kontinuerlig forbedre løsningens brukervennlighet, funksjonalitet og praktiske egenskaper basert på tilbakemeldinger fra bruker.
Applikasjonen er ikke brukervennlig	Mangelfullt arbeid. Utilstrekkelig kommunikasjon med brukere. Mangelfull brukertesting.	3	5	Middels	15	Opprettholde ett universelt og brukervennlig design på applikasjonen. Inkluder en veiledningsfunksjon i applikasjonen for å hjelpe brukeren. Opprett veiledningsvideoer og forklarende lydklipp. Utfør brukertesting for validering av resultat.
Feilestimere utviklingstid for oppgaven	Ustrukturert arbeidsplan. Uforutsette hendelser. Uvant teknologi	2	5	Middels	10	Implementering av et gant-diagram for å strukturere fremdrift. Start utviklingsprosessen tidlig. Kontinuerlig arbeid med prosjektet.
Feilestimere brukerbehov	Mangelfull brukertesting. Utilstrekkelig kommunikasjon med sluttbruker. Veileder og oppdragsgiver.	2	5	Middels	10	Gjevnlig samtale med prosjektleder og oppgavegiver. Utfør brukertesting for validering av resultat. Implementer interessentanalyse.
Modellens målinger er utilstrekkelige.	Feil valg av ML modell. Mangelfull testing.	3	5	Middels	15	Utforsk ulike modeller som gir bedre resultat. Gjennomfør grudige tester før modellen tas i bruk.
Sykdom i gruppen	Smittebølger. Pandemi. Temperatur. Klima.	4	2	Middels	8	Bruk varme klær. Ta vare på helsen. Ta pauser ved behov.
Ineffektivt gruppearbeid	Mangelfull strukturering av oppgaver.	1	4	Lav	4	Opprett en strukturert plan. Arbeid kontinuerlig. Tett dialog med prosjektleder
Utilstrekkelig ytelse på telefon	Valg av feil ML modell. Mangelfull applikasjonstesting.	2	4	Middels	8	Velg en enklere modell. Utfør applikasjonstesting.

Figur 3-2: Risikoanalyse

Sannsynlighet	Svært Høy (5)	5	10	15	20	25
	Høy (4)	4	8	12	16	20
	Middels (3)	3	6	9	12	15
	Lav (2)	2	4	6	8	10
	Svært Lav (1)	1	2	3	4	5
Konsekvens		Svært Lav (1)	Lav (2)	Middels (3)	Høy (4)	Svært Høy (5)

Figur 3-3: Risikomatrise

3.5 Evalueringsplan

3.5.1 Evaluering av HPE modell

Evaluering av HPE modellen blir en del av en større generell systemevaluering. Siden gruppen har valgt modell basert på tidligere forskning, vil det ikke utføres videre analyser av presisjonen, ytelsen og nøyaktigheten til den valgte HPE modellen.

3.5.2 Evaluering av applikasjonen

Evaluering av applikasjonen vil skje i flere omganger. Møter vil holdes underveis i utviklingen med oppdragsgiver, der oppdragsgiver kommer med innspill og tilbakemeldinger på applikasjonen, og det vil utføres brukertesting av systemet som del av sluttfasen. En brukertest i vårt tilfelle vil være at en bruker gir tilbakemeldinger basert på om applikasjonen er brukervennlig, intuitiv og forståelig.

Funksjoner evalueres ved bruk av enhetstester. Dette er enkeltkomponenter som er designet for å teste at funksjonene i programmet opererer som planlagt, og for å avdekke feil i koden. Enhetstester blir brukt iterativt gjennom utviklingen av systemet og skal kjøres etter nye implementasjoner av kode.

4 DETALJERT LØSNING

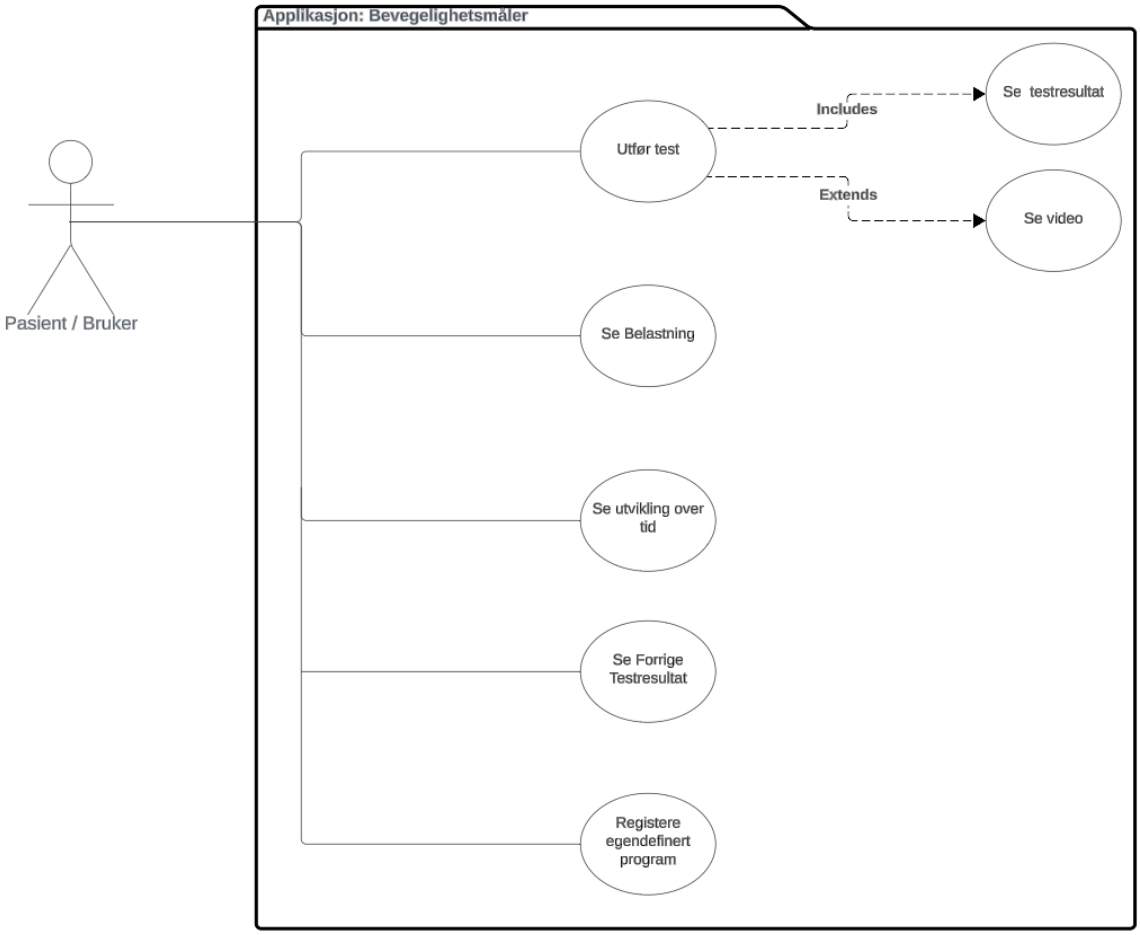
4.1 Brukstilfellediagram

Use-case diagrammet i Figur 4-1 viser brukstilfellene basert på de funksjonelle kravene gitt av oppdragsgiver. Applikasjonen har en interessent, *Pasient*, som har tilgang til alle funksjonalitetene, og går under betegnelsen bruker. Pasient representerer de som oppsøker behandling.

Utfør test er kjernefunksjonaliteten i applikasjonen, det er her brukeren av applikasjonen utfører øvelsene som behandles av en HPE-modell. Utfør test inneholder to ekstra brukstilfeller, *se test resultat* og *se video.* Det vil si at en test skal ha funksjonaliteten til å vise resultatet, og en video som viser hvordan øvelsen utføres.

Se belastning, *Se utvikling over tid*, og *Se forrige testresultat* er brukstilfeller som er tett knyttet sammen. Brukstilfellene omhandler sitasjoner der det er gunstig med en visuell representasjon over en brukers utvikling og nåværende bevegelighet.

Det siste brukstilfellet er *Registrere egendefinert program*, som gir brukeren mulighet til å opprette egne program og velge hvilke øvelser de ønsker å inkludere i programmet.



Figur 4-1: Use-case Diagram

4.2 Wireframes

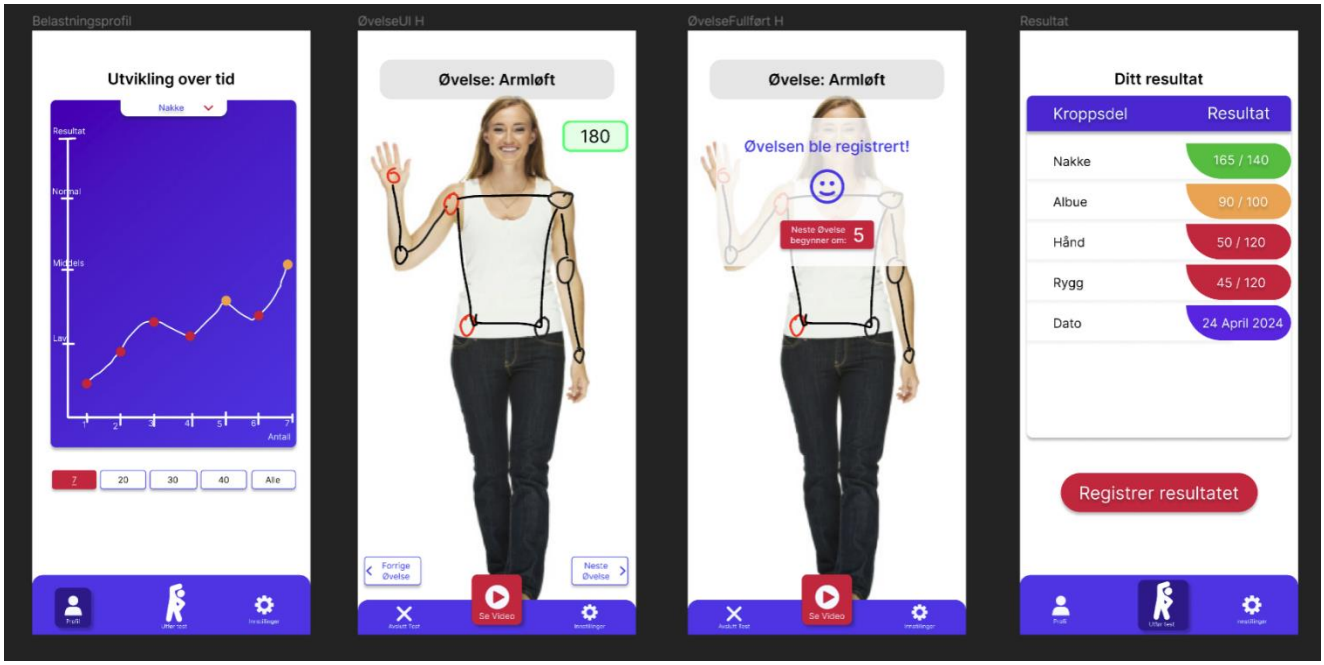
Wireframes er iterativt utviklet i Figma og er ment å visualisere brukergrensesnittet. De representerer ikke det ferdige produktet, men er ment som et hjelpemiddel under utviklingen av applikasjonen. Fire viktige skisser er inkludert i Figur 4-2, for en oversikt over alle wireframes og tidligere iterasjoner se kapittel 4 i (Vedlegg IV: Kravspesifikasjon).

Den første skissen fra venstre, viser brukerens *utvikling over tid*. Brukeren velger selv kroppsdel fra en nedtrekksmeny og antall punkter som vises på grafen. Fargen på punktene baseres på scoren til øvelsen i forhold til normal bevegelighet, hvor rødt representerer lav, gul moderat og grønn normal bevegelighet. Denne skjermen er ment for å løse tre av brukstilfellene, *Se utvikling over tid*, *Se belastning* og *Se forrige resultat*.

Den andre skissen viser designet til utfør test-siden. Vinkelutslaget til brukeren vises øverst til høyere og kalkuleres basert på vinkelen mellom de røde punktene i skissen. Utfør test har to ulike moduser, manuell og automatisk, modusen kan endres i innstillinger. Under en manuell test utfører brukeren øvelsene for så å manuelt navigere til neste øvelse.

Med automatisk modus aktivert, som vises i tredje skisse, er øvelsene tidsbasert. Brukeren får 15 sekunder på å utføre øvelsen. Deretter vises meldingen *Øvelsen ble registrert* og brukeren får 5 sekunder før neste øvelse begynner. Etter den siste øvelsen navigeres brukeren til resultatsiden.

Den siste skissen viser resultatet for øvelsene. Siden oppsummerer resultatene for det utførte programmet. Vinkelutslagene brukeren oppnådde illustreres på siden og benytter samme fargekoder som graf siden. Brukeren kan så trykke på *registrer resultatet* knappen for å registrere resultatet og bli navigert videre til *velg øvelse* skjermen som illustreres i (Vedlegg IV: Kravspesifikasjon).



Figur 4-2: Wireframes

4.3 Arkitektur

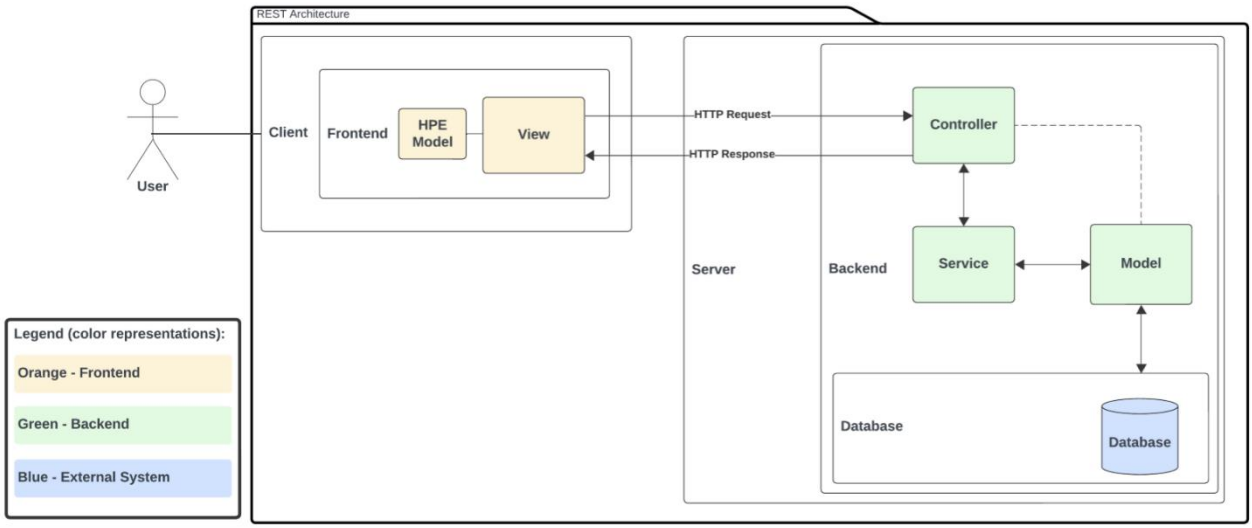
4.3.1 Overordnet Arkitektur

Den overordnede arkitekturen for applikasjonen er illustrert i Figur 4-3.

Som beskrevet i (Vedlegg I: Systemdokumentasjon), gir arkitekturen en oversikt over hvilke elementer løsningen består av, hva de ulike elementene inneholder, og hvordan de kommuniserer med hverandre for å løse oppgavene. Vi ser at subsystemene i klientsiden, markert med oransje, kommuniserer med klassene på serversiden som er markert med grønn. Backend opererer som et REST API, der frontend kan utføre kall på tilgjengelige funksjoner via HTTP.

Videre ser vi den generelle samhørigheten mellom de ulike typer av subsystemer og hvordan informasjonsflyten formidles. Bruker interagerer med grensesnittet på klient siden, klienten kommuniserer så med serveren. Deretter ser vi at backend kommuniserer med databasen, et eksternt system markert i blått.

REST (REpresentational State Transfer), er en arkitektonisk stil som gir en fundamentalistisk standard for kommunikasjon mellom nettverksapplikasjoner (CodeAcademy, 2024). Stilen er tilstandsløs, som vil si at komponentene og klassene i backend er uavhengige av frontend (Fielding, 2000, p. 78). I et tilstandsløst system behøver ikke server-siden å lagre informasjon om frontend. Når et API-kall utføres, inkluderes all informasjonen nødvendig for å utføre en funksjon. Prinsippet om tilstandsløshet gjør applikasjonen mer skalerbar og administrerbar.



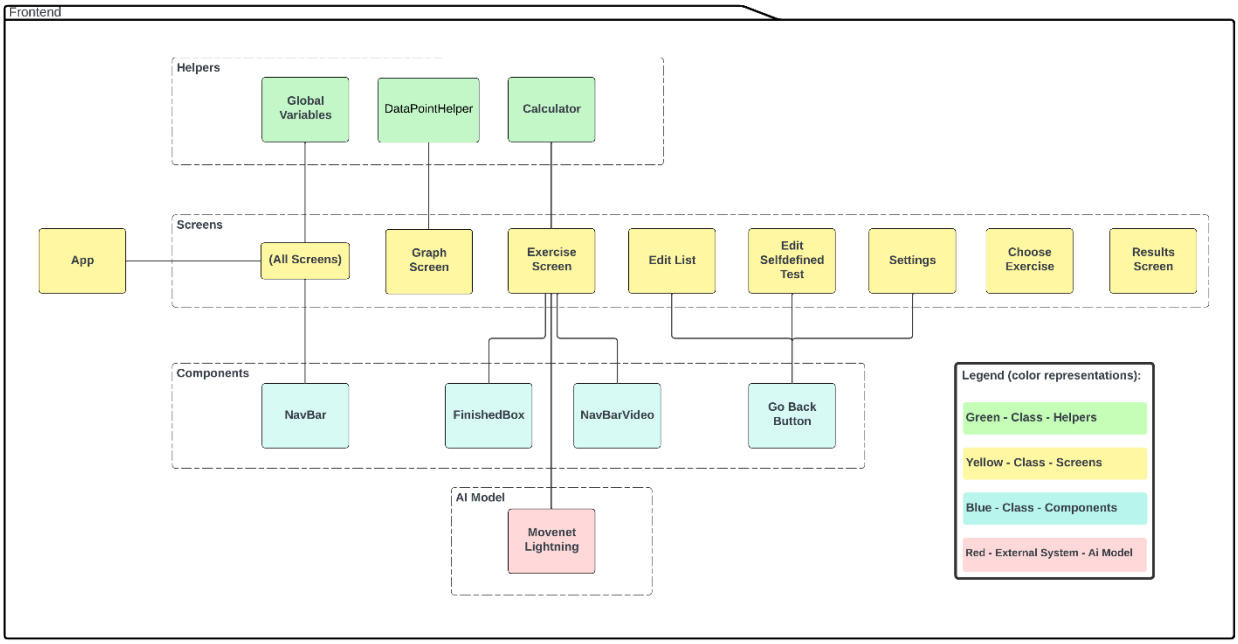
Figur 4-3: Overordnet Arkitektur

4.3.2 Frontend

Frontend-arkitekturen for applikasjonen er illustrert i Figur 4-4. Systemet håndterer all klientside logikk og fungerer som en intermediær mellom brukere og backend-serveren.

Som beskrevet i (Vedlegg I: Systemdokumentasjon), er frontend-arkitekturen strukturert i fire distinkte klasser: *Helpers, Screens, Components* og *AI Model*. Klassen *Helpers* inneholder hjelpemetoder med formål å abstrahere bort kode fra *Screens* klassene, samt forbedre systemets lesbarhet. Denne klassen omfatter også globale variabler som benyttes på tvers av skjermene. *Screens* representerer brukergrensesnittene som brukeren interagerer med. *Components* består av mindre, gjenbrukere kodeenheter som benyttes av *Screens* for å fremme modulær utvikling av brukergrensesnittkomponenter. *App*-klassen håndterer navigasjonslogikk mellom ulike skjermer og definerer gyldige input-verdier mellom dem. *AI Model* representerer *MoveNet Lightning* modellen som har ansvar for å prediktere punkter basert på input fra mobilkameraet. *AI Model* er en del av *Exercise Screen* skjermen.

For å forbedre visuell klarhet og lesbarhet i dokumentasjonen, er klassene i modellen fargekodet: *Helpers* er markert i grønt, *Screens* i gult, *Components* i blått, og *AI Model* i rødt, som representerer eksterne systemer.



Figur 4-4: Frontend Arkitektur

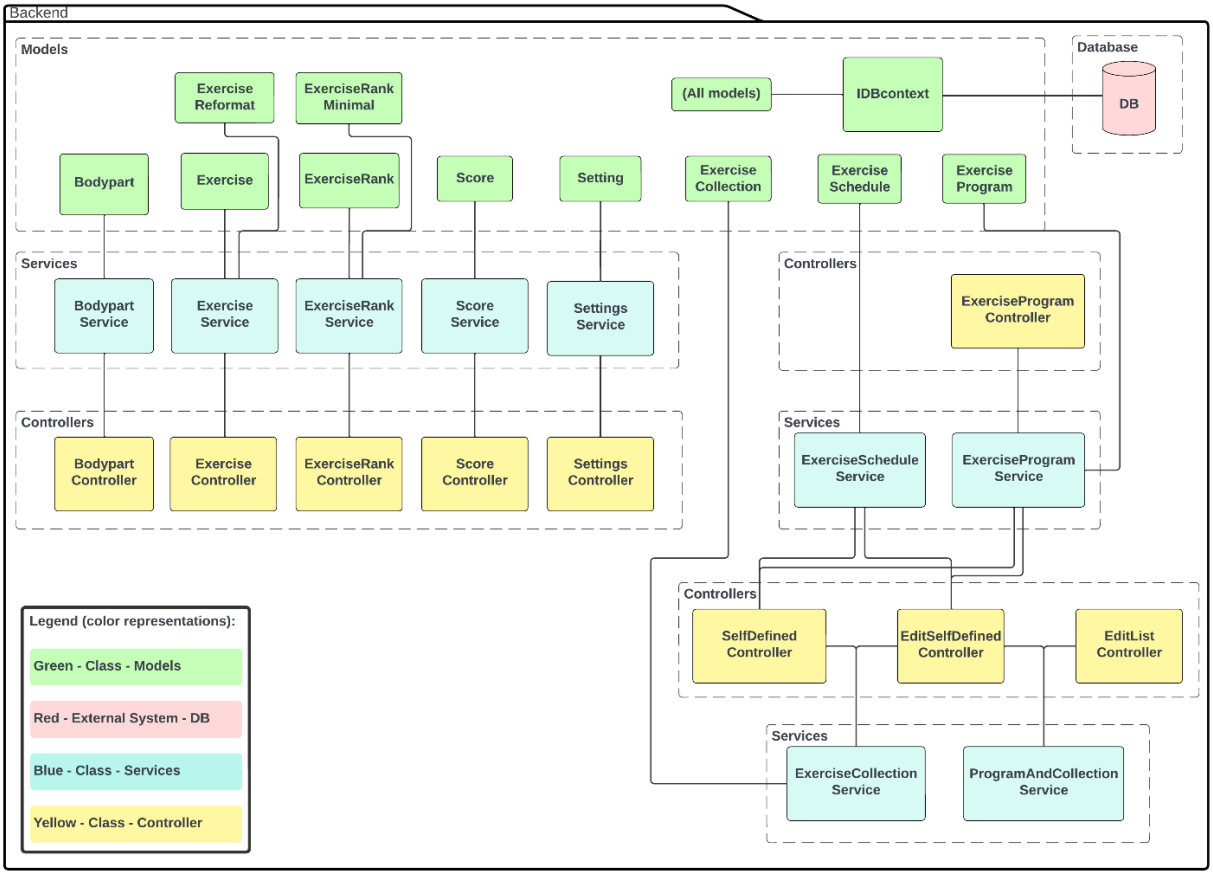
4.3.3 Backend

Backend-arkitekturen håndterer prosessene som utføres på serversiden av applikasjonen og illustreres i Figur 4-5.

Backend fungerer som et bindeledd mellom frontend og databasen, som administrerer dataflyt, logikk og tilgangskontroll. Denne delen av systemet mottar forespørsler fra klienten, utfører nødvendige operasjoner, for deretter å returnere data til klienten.

Det er implementert et fargesystem for å visualisere ulike aspekter av systemet, som bidrar til økt lesbarhet og organisering. Modeller, markert med grønt, representerer klasser som korresponderer med databasestrukturen og håndterer datastruktur og assosiasjoner. Databasen, et eksternt system hostet på en Azure-server, markeres med rødt. Blått representerer tjenestelaget, som inneholder klasser dedikert til å utføre forretningslogikk og interaksjoner mellom modeller og kontrollere. Kontrollere, indikert med gult, styrer dataflyten mellom modeller, visninger og brukerforespørsler.

For mer informasjon om arkitekturen og den overordnede arkitekturen henviser vi til systemdokumentasjon i (Vedlegg I: Systemdokumentasjon).



Figur 4-5: Backend Arkitektur

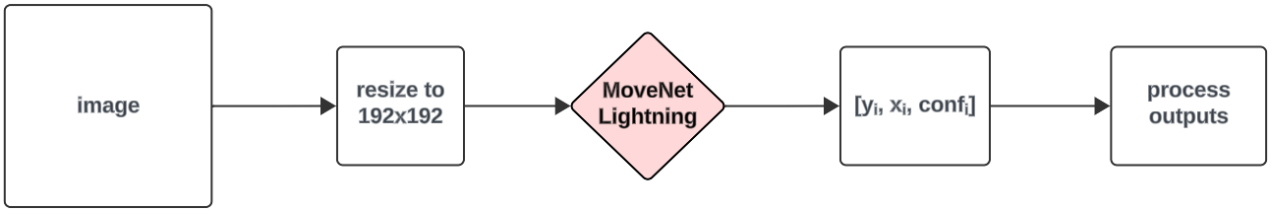
4.4 HPE modell

4.4.1 MoveNet Lightning

Systemet implementerer en ferdig trent HPE modell av typen Movenet Lightning, utviklet av Google for bruk med TensorFlow. Modellen er utformet for mobile enheter og er ideell for systemer hvor lav latens er nødvendig, samtidig som det kreves høy presisjon og nøyaktighet (Eng-Galåen, 2024). Modellen sin inndata er bilder med oppløsning på 192x192 piksler og produserer posisjonsdata i form av x, y koordinater, og konfidensnivå for hver prediksjon. Output fra modellen består av en array med 51 elementer som følger mønsteret: $[Y_0, X_0, C_0, Y_1, X_1, C_1, \dots, Y_{16}, X_{16}, C_{16}]$ hvor $(Xi,\ Yi,\ Ci)$ representerer koordinater og konfidensnivå for det i-te punktet. Modellen gir totalt 17 punkter som tilsvarer ulike kroppsdeler (Tensorflow, 2024).

4.4.2 Intialisering

Initialisering og bruk av MoveNet Lightning-modellen håndteres i klassen *Exercise Screen*. Denne klassen tar flere bilder i sekundet, transformerer de for å tilpasse formatet som kreves av AI-modellen, overfører dem til modellen, og håndterer deretter modellens output. Denne prosessen sikrer at systemet kan reagere dynamisk på brukerens bevegelser i sanntid og gi brukeren en interaktiv opplevelse, visualisert i Figur 4-6.



Figur 4-6: Dataflyt - HPE Modell

4.4.3 Validering av utslagsvinkler

Figur 4-7 illustrerer to skisser: en ugyldig utførelse av armløft, og en gyldig utførelse av samme øvelse.

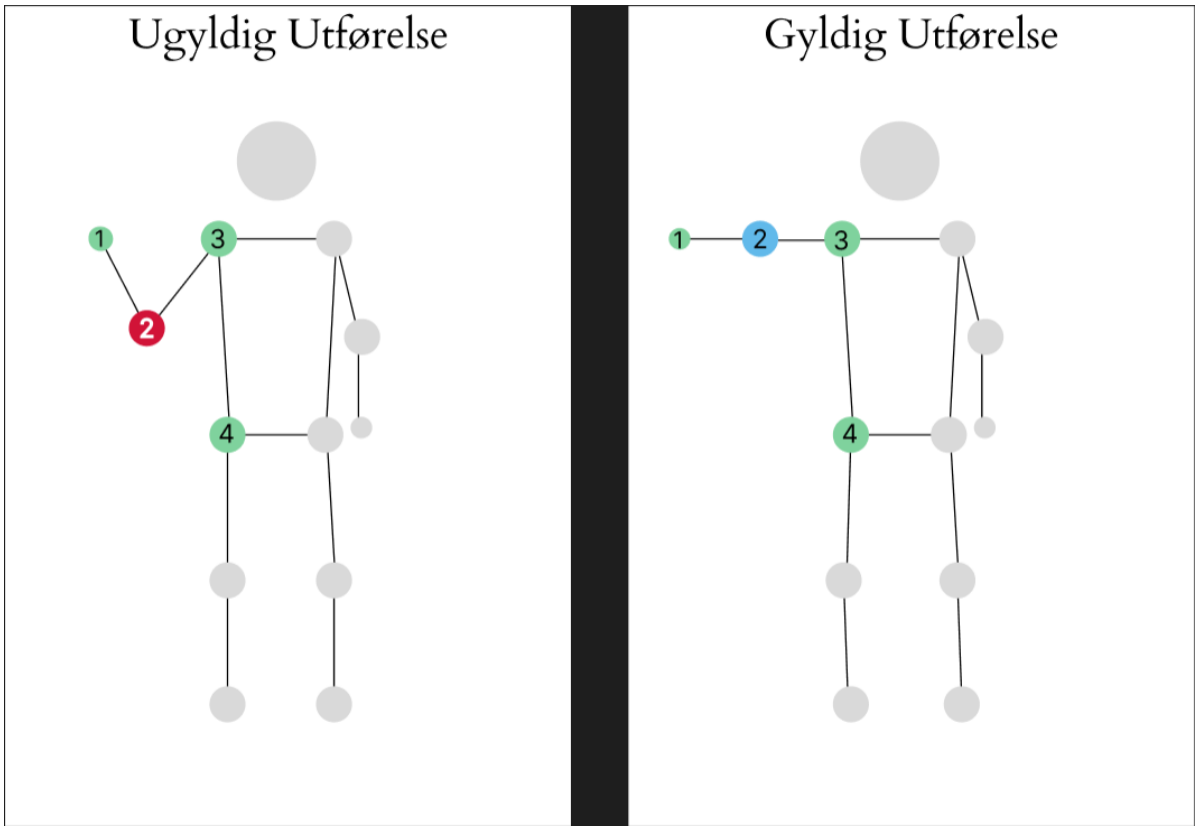
Det er viktig å vite at resultatet fra MoveNet Lightning-modellen ikke direkte gir vinkelutslag, men posisjonerer punkter innenfor et kartesisk koordinatsystem. Basert på punktene kan vinkelutslaget beregnes ved å bruke formelen:

$$\arccos\left(\frac{(a^2 + b^2 - c^2)}{(2 * a * b)}\right)$$

Hvor a, b og c er avstander mellom tre punkter, x, y og z. Avstanden a er mellom punkt x og punkt y, b er avstanden mellom punkt x og punkt z, og c er avstanden mellom punkt z og punkt y.

Som demonstrert i den første skissen i Figur 4-7, er det nødvendig å implementere unntakshåndtering for tilfeller der øvelsen utføres feil. For eksempel indikerer Figur 4-7 at når albuen, markert som punkt 2, er bøyd, betraktes øvelsen som ugyldig. For å adressere dette, definerer gruppen konseptene utslagsvinkel og valideringsvinkel. Utslagsvinkelen bergenes med et vertex-punkt og to referansepunkter, eksempelvis punkt 3, 4 og 1. Valideringsvinkelen i figuren benytter det samme vertex-punktet, punkt 3, et referansepunkt som tilsvarer et fra utslagsvinkelen, punkt 4, i tillegg til et annet referansepunkt som ikke er i utslagsvinkelen, punkt 2. Dette muliggjør beregning av forskjellen mellom de to vinklene.

For å oppnå pålitelige resultater på målingene, beregnes absoluttverdien av differansen mellom vinkelutslagene på utslagsvinkelen og valideringsvinkelen. I tilfeller der forskjellen mellom vinklene er lav, vil øvelsen være korrekt utført og utslagsvinkelen oppdateres. dersom differansen er høy, vil utslaget ignoreres.



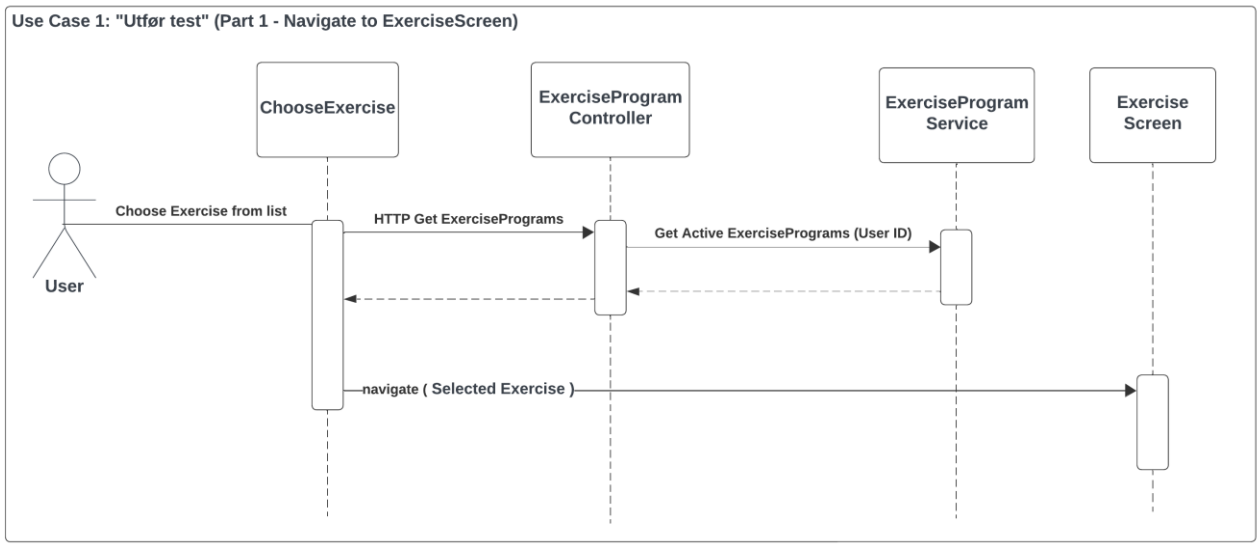
Figur 4-7: Validering Av Utslagsvinkler

4.5 Sekvensdiagram

Denne seksjonen presenterer sekvensdiagrammer for de to viktigste brukstilfellene i applikasjonen: *Utfør test* og *Se utvikling over tid*. Sekvensdiagrammene illustrerer flyten gjennom de ulike komponentene i applikasjonen. *Utfør test* inneholder omfattende logikk og det er hensiktsmessig å dele den opp i to separate deler. Den første delen dekker logikken bak menyen der brukeren velger et oppfølgingsprogram. Den andre delen omhandler logikken som utføres under øvelsene.

4.5.1 Use Case 1: Utfør test (Del 1)

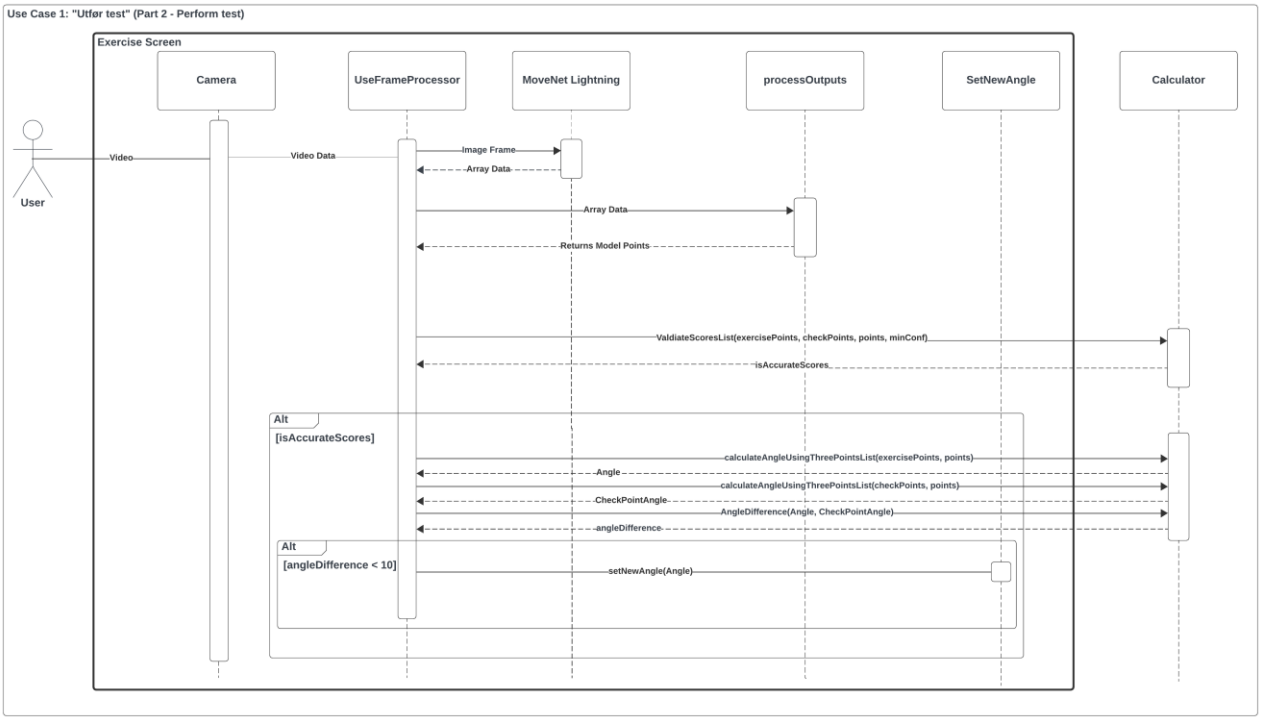
Det første sekvensdiagrammet se Figur 4-8 illustrerer interaksjonene mellom brukeren, skjermen *ChooseExercise* og backend serveren for å navigere til *ExerciseScreen*. Brukeren velger et program fra listen over relevante programmer. Deretter sendes en GET forespørsel til *ExerciseProgramController*. Kontrolleren kaller på *ExerciseProgramService* som henter informasjon om de korresponderende øvelsene fra databasen. Øvelsene blir så sendt med HTTP til klientsiden. Brukeren navigerer så til *ExerciseScreen* der øvelsene kan utføres.



Figur 4-8: Sekvensdiagram - Utfør test

4.5.2 Use Case 1: Utfør test (Del 2)

Det andre sekvensdiagrammet se Figur 4-9, illustrerer interaksjonene mellom *Bruker, Camera*, funksjoner i *Exercise Screen* Klassen, KI modellen og *Calculator* Klassen. *Camera* komponenten sender video data i sanntid til *UseFrameProcessor* som kaller på *MoveNet Lightning* modellen der bildene prosesseres. Modellen returnerer data i form av en array, som deretter sendes til *processOutputs* funksjonen. *ProcessOutputs* tar listen med punkter og behandler dem slik at de plasseres korrekt i forhold til personen i bildet. Punktene som benyttes for å beregne vinkelutslag valideres så av *ValidateScoresList* funksjonen i *Calculator* før vinkelutslaget mellom kroppsdelene og validerings punktene beregnes og oppdateres med *SetNewAngle*.



Figur 4-9: Sekvensdiagram - Utfør Test Del 2

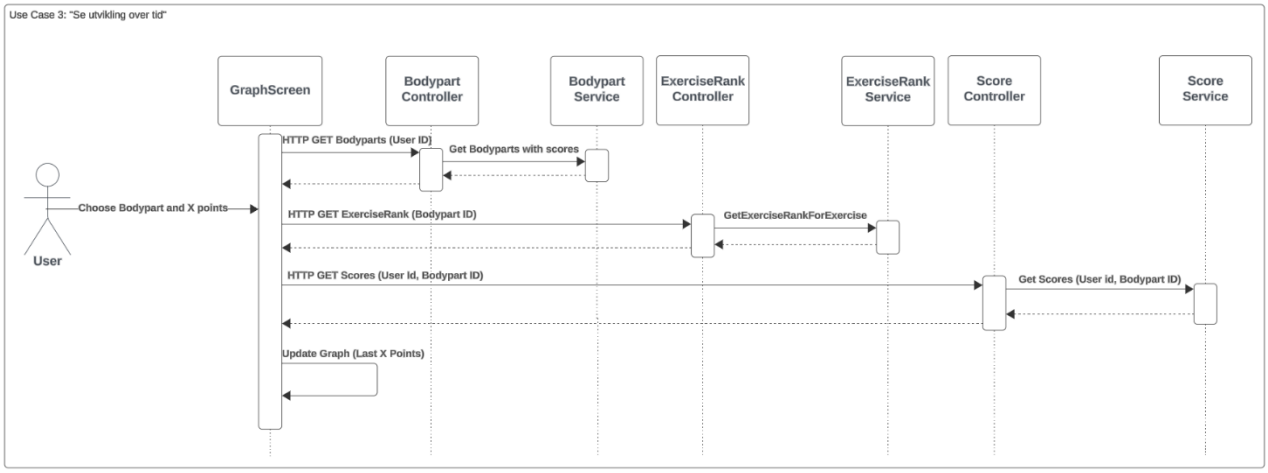
4.5.3 Use Case 2: Se utvikling over tid

Sekvensdiagrammet i Figur 4-10 illustrerer utviklingen over tid og starter med at *GraphScreen* initierer en HTTP GET-forespørsel for å hente alle kroppsdelene som brukeren har utført øvelser for. Brukeren velger så en spesifikk kroppsdel og angir tidsperioden de ønsker å analysere.

Etter brukerens valg, sender *GraphScreen* en ny HTTP GET-forespørsel til *ExerciseRank Controller*, den returnerer en *exerciseRank* som består av tre nivå: lav, middels og normal i tillegg til scoren som kreves for å nå de ulike rangeringene. *ExerciseRank* benyttes for å fastslå hvordan en bruker ligger an i forhold til normal bevegelighet.

Deretter gjør *GraphScreen* en ytterligere HTTP GET-forspørsel til *Score Controller*, som henter ut alle resultatene til brukeren for den valgte kroppsdelen.

GraphScreen kaller så *UpdateGraph* funksjonen, som oppdaterer grafen som vises på skjermen, slik at den viser valgt antall punkter for en gitt kroppsdel.



Figur 4-10: Sekvensdiagram - Utvikling Over Tid

5 RESULTATER

5.1 Evalueringsmetode

5.1.1 Brukertesting

Formålet med brukertesting er å innhente subjektive tilbakemeldinger fra potensielle brukere om deres tilfredshet med applikasjonen, samt eventuelle elementer de er mindre fornøyde med. Slike tilbakemeldinger er nødvendige for å videreutvikle applikasjonen slik at den best mulig tilfredsstiller brukernes behov.

Deltakerne i brukertesten er et utvalg av syv tilfeldig frivillige, som potensielt vil kunne være pasienter i en virkelig brukssituasjon. Dette sikrer en upartisk forståelse av brukeropplevelsen.

Før testingen starter, får deltakerne en kort introduksjon til applikasjonen for å lettere forstå hensikten og funksjonaliteten. Deltakeren får deretter utforske applikasjonen på egenhånd.

Etter gjennomføringen blir deltakerne bedt om å svare på spørsmål knyttet til deres opplevelse med applikasjonen. Spørsmålene er basert på System Usability Scale (SUS) diagrammet, utviklet av ingeniør John Brooke i 1986. SUS er en systematisk metode for å strukturere brukertesting. Ved å alternere mellom positive og negative spørsmål tenker brukeren mer grundig over spørsmålene (Gallavin, 2014).

SUS diagrammet som er utviklet av gruppen er basert på de 10 spørsmålene utarbeidet av John Brooke i tillegg til et spørsmål rettet mot brukergrensesnittet. Det er i tillegg lagt på fargekode på negativt og positivt ladet spørsmål, noe som muliggjør beregning av gjennomsnittlig brukertilfredshet. For å se det originale SUS skjemaet se (Vedlegg VI: SUS skjema utviklet av John Brooke).

Videre blir deltakerne spurt om det er noen ønskede funksjoner applikasjonen manglet og om det er aspekter ved applikasjonen som de ville endret.

5.1.2 Enhetstester

Testing er en sentral praksis i systemutvikling som har som formål å minimere eller eliminere uønskede eller defekte komponenter i et system. For å validere systemkoden i prosjektet, og sørge for programvaren utfører operasjonene den er spesifisert til er det utviklet ulike tester.

En enhetstest er et isolert program som tester individuelle komponenter i koden. Enhetstesting har som formål å avdekke feil i enkeltkomponenter og forbedre systemets stabilitet og pålitelighet. Videre bidrar enhetstester til å styrke systemets skalerbarhet siden de fungerer som en automatisert metode for å teste systemets eksisterende funksjoner.

For testing av databasekall er det utviklet en Mock database. En Mock database er en simulert database utviklet for å imitere atferden og virkemåten til den eksisterende databasen. Formålet med en Mock database er å avdekke eventuelle feil med funksjonskall til databasen, uten å benytte den reelle. Dette reduserer risikoen for feil som kan oppstå i databasesystemet og fungerer som en måte å isolere testmiljøet fra databasesystemet.

5.1.3 Systemtest

Systemtester er en overordnet form for testing som utføres over et helhetlig system. Formålet med systemtester er å avdekke defekte komponenter og feil mellom funksjonskall.

Systemtestingen utføres ved bruk av ad hoc testing, der alle funksjonaliteter, systemkall og brukstilfeller testes. Ad hoc testing er ofte utført av utviklere som kjenner systemet godt, der ulike funksjoner av systemet blir testet uten behov for dokumentasjon om hvordan testingen skal utføres. Denne typen testing utføres etter enhetstester (Devaraj, 2024).

5.2 Evalueringsresultat

5.2.1 Brukertesting

			10	7,5	5	2,5	0
			0	2,5	5	7,5	10
			Helt Enig	Delvis Enig	Nøytral	Delvis Uenig	Helt Uenig
	1	Jeg tror jeg ville brukt denne applikasjonen jevnlig					
	2	Jeg fant applikasjonen unødvendig kompleks					
	3	Jeg synes applikasjonen var lett å bruke					
	4	Jeg tror jeg ville trengt hjelp fra en teknisk anlagt person for å forstå systemet					
	5	Jeg fant at funksjonene i applikasjonen fungerer godt sammen					
	6	Jeg følte det var for mye inkonsistensitet i applikasjonen					
	7	Jeg kan tenke meg at folk flest vil lære seg bruken av applikasjonen fort					
	8	Jeg følte at applikasjonen var vanskelig å forstå					
	9	Jeg følte at jeg forstod applikasjonen godt					
	10	Jeg må anskaffe meg mye kunnskap før jeg kan bruke denne applikasjonen.					
	11	Er du fornøyd med brukergrensesnittet (designet på siden)					

Figur 5-1: Brukertest

Undersøkelsen vises i Figur 5-1, der grønne farger indikerer positivt ladet spørsmål og rød indikerer et negativt ladet spørsmål. Resultatet er kalkulert ved at svaralternativet blir konvertert til et tall, helt enig blir 10, delvis enig blir 7,5 og helt uenig blir 0.

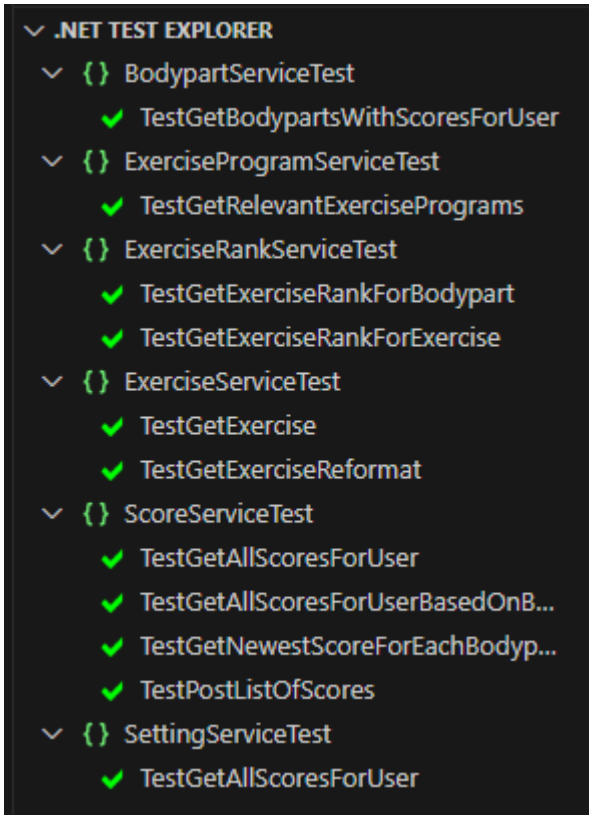
Svar fra negativt ladde spørsmål blir invertert, det vil si hvis en deltaker er helt enig med et negativt spørsmål blir resultatet til det korresponderende spørsmålet satt til null. Dette resulterer i at store tall er positive svar og lave tall er negative svar.

Brukertesten gav god konstruktiv kritikk til applikasjonen, brukerene formidlet både om positive og negative aspekter med applikasjonen. Gjennomsnittlig brukertilfredshet beregnes basert på resultatene til brukerundersøkelsen.

Den gjennomsnittlige brukertilfredsheten er 76,30 %. Tallene som gav dette resultatet, kan leses i (Vedlegg VII: Svar på brukerundersøkelsen). Det aspektet deltakerne gav høyest rangering var *Jeg synes applikasjonen var lett å bruke* med brukertilfredshet på 89,79%. Aspektet deltakerne gav lavest rangering var *Jeg tror jeg ville brukt denne applikasjonen jevnlig* med brukertilfredshet på 53,57%.

Deltakerne foreslo endringer og nye funksjoner. Et av ønskene var en funksjon der instruksjonsvideoene starter når brukeren starter en ny øvelse. Et annet ønske var at brukeren av applikasjonen kan få opplest navnet på en øvelse når den startes. Det ble også observert at brukere fant det vanskelig å lese teksten på skjermen når de sto med lang avstand fra mobilskjermen. Det siste som ble foreslått var en kort introduksjon i starten av appen, slik at nye brukerene får en kjapp gjennomgang av funksjonene.

5.2.2 Unit Testing



Figur 5-2: Service Tester

Figur 5-2 viser et utklipp av service tester, der alle testene returnerte forventet verdi fra Mock databasen. Programkoden endres til den består testene.

5.2.3 Systemtest

Systemtest ble utført manuelt for hver skjerm og for hver funksjon. Det ble ikke registrert noen avvik i funksjonalitet eller feil på skjermene. Evaluering av HPE modellen er en del av systemtesten, gruppen bemerket at MoveNet Lightning modellen var responsiv og gav nøyaktige og presise prediksjoner, noe som støtter under resultatene fra forskningsartikkelen (Jo & Kim, 2022).

5.3 Prosjektresultat

5.3.1 Funksjonelle og ikke funksjonelle krav

For å evaluere de funksjonelle og ikke funksjonelle kravene for utvikling av applikasjonen er det utviklet to diagrammer. Diagrammene illustrerer oversikter over de planlagte og fullførte kravene.

Tabellen i Figur 5-3 viser de funksjonelle kravene som er satt til applikasjonen. Det er et avvik på krav nummer 1: *Applikasjonen skal være kompatibel med IOS og Android*.

Tabellen i Figur 5-4 viser de ikke-funksjonelle kravene som er satt til applikasjonen. Her ser vi at alle kravene er oppfylt. For mer informasjon angående de funksjonelle og ikke funksjonelle kravene se (Vedlegg II: Visjonsdokument).

NR	Funksjonelle krav	Fullført Ja / Nei
1	Applikasjonen skal være kompatibel med IOS og Android	Nei
2	Utføre bevegelighetstest for måling av bevegelighet	Ja
3	Vise vinkelutslag for kroppsdel under utførelse av øvelser	Ja
4	Brukeren skal ha muligheten til å utføre tester for måling av beveglighet	Ja
5	Etter endt oppfølgning skal testresultatet vises.	Ja
6	Brukeren skal ha muligheten til å visualisere progresjon av vinkelutslag for individuelle kroppsdeler	Ja
7	Brukeren skal ha muligheten til å se utførelsen av ROM testen gjennom en video.	Ja

Figur 5-3: Funksjonelle Krav

NR	Ikke-funksjonelle krav	Fullført Ja / Nei
1	Applikasjonen skal ha god ytelse	Ja
2	Målingene til AI modellen skal være presise	Ja
3	Applikasjonen skal benytte rammeverket ASP.NET	Ja
4	Applikasjonen skal benytte rammeverket React Native	Ja
5	Applikasjonen skal benytte en Postgre SQL database	Ja

Figur 5-4: Ikke-Funksjonelle Krav

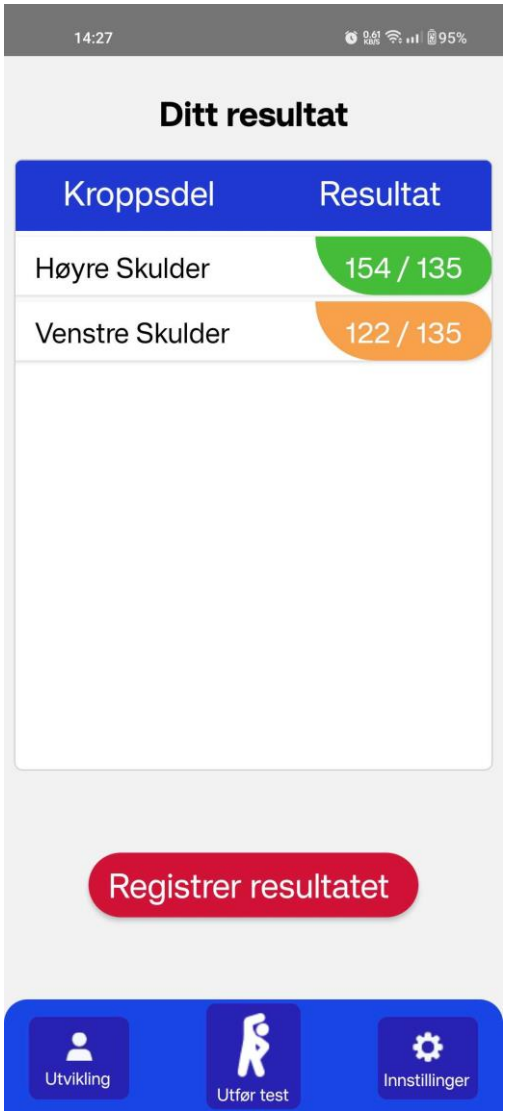
5.3.2 Applikasjonen

Under vises det skjermbilder av brukergrensesnittet som benyttes i applikasjon. Alle sidene av applikasjonen vises i (Vedlegg VIII: Skjermbilder fra Applikasjonen), kildekoden for applikasjonen finnes i vedlegg (Vedlegg V).

Brukergrensesnittet i Figur 5-5 viser et utklipp av det ferdige grensesnittet på *utfør test*-siden med innstillingene *tidsbasert øvelse* og *vis linjer* aktivert. Brukeren kan trykke på *Video* knappen for å få visualisert utførelse av øvelsen. Her utfører brukeren øvelsen som er indikert øverst på skjermen. Det største vinkelutslaget blir målt og vist i en boks med en farge som indikerer om det er lav, moderat eller høy bevegelighet. Under ser brukeren tiden som gjenstår før neste øvelse begynner.

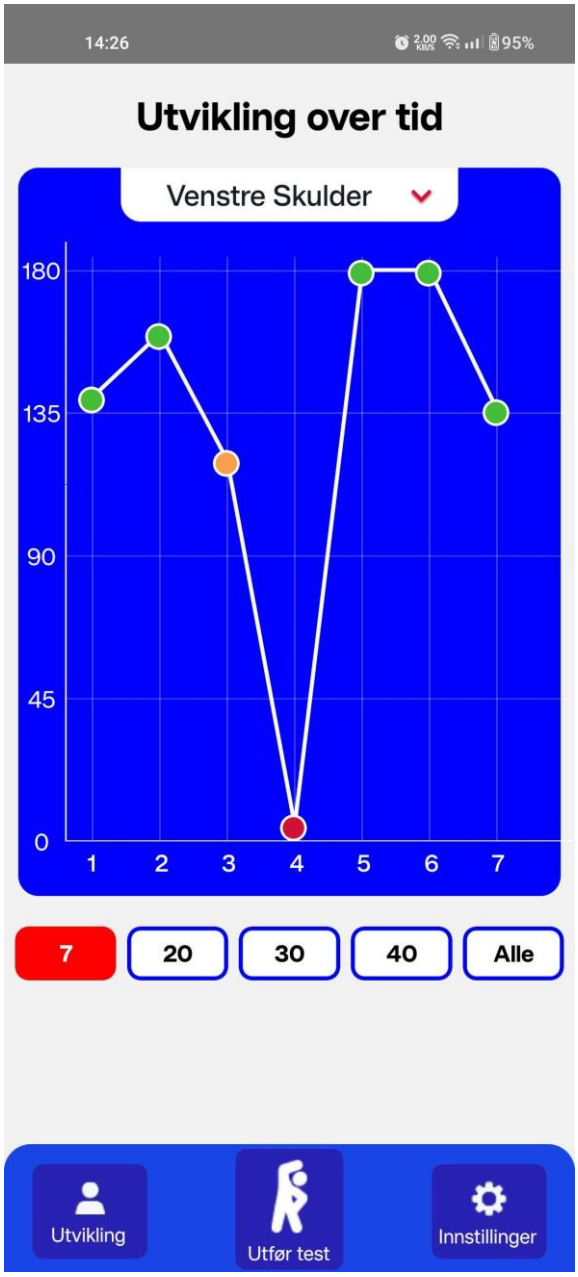
Figur 5-5: Brukergrensesnitt - Utfør Test

Figur 5-6 viser et skjermbilde av resultatet til brukeren etter utført test. Etter at en bruker har utført en test, vil man bli videresendt til denne skjermen som gir brukeren informasjon om hvor stor belastning de har på de ulike kroppsdelene. Brukeren velger så Registrer resultatet når de er ferdig å se på sitt resultat. Denne skjermen oppfyller brukstilfellene *Se belastning* og *Se forrige testresultat*.



Figur 5-6. Brukergrensesnitt - Ditt Resultat

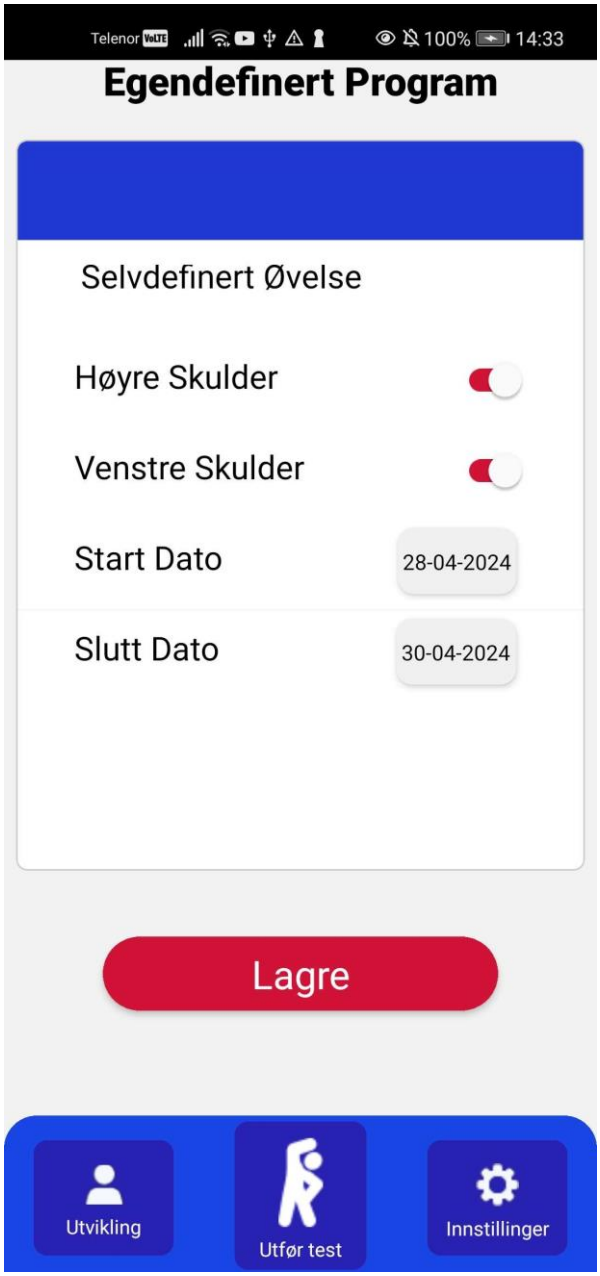
Skjermbilde av graf-skjermen på applikasjonen er vist i Figur 5-7. Denne siden viser en brukers siste syv punkter for sin venstre skulder. De punktene der brukeren oppnådde høy bevegelighet er markert i grønn. Punkter med middels bevegelighet er gul, og lav bevegelighet er markert med rød. I tillegg kan brukeren trykke på punktene for å få en mer detaljert forklaring på hva resultatet ble. Denne skjermen oppfyller brukstilfellene *Se utvikling over tid* og *Se forrige testresultat*.



Figur 5-7: Brukergrensesnitt – Utvikling Over Tid

Et program består av en eller flere øvelser, har en start og slutt dato og benyttes av brukere til å teste individuell bevegelighet. Programmer kan være egendefinert eller utviklet av veileder.

Figur 5-8 viser et skjermbilde av egendefinert program siden. Denne siden benyttes av brukeren til å definere egne programmer. Brukeren kan definere navn på programmet, hvilke øvelser programmet består av, samt start og sluttdato. Denne siden oppfyller brukstilfellet *Registrere egendefinert program*.



Figur 5-8: Brukergrensesnitt – Egendefinert Program

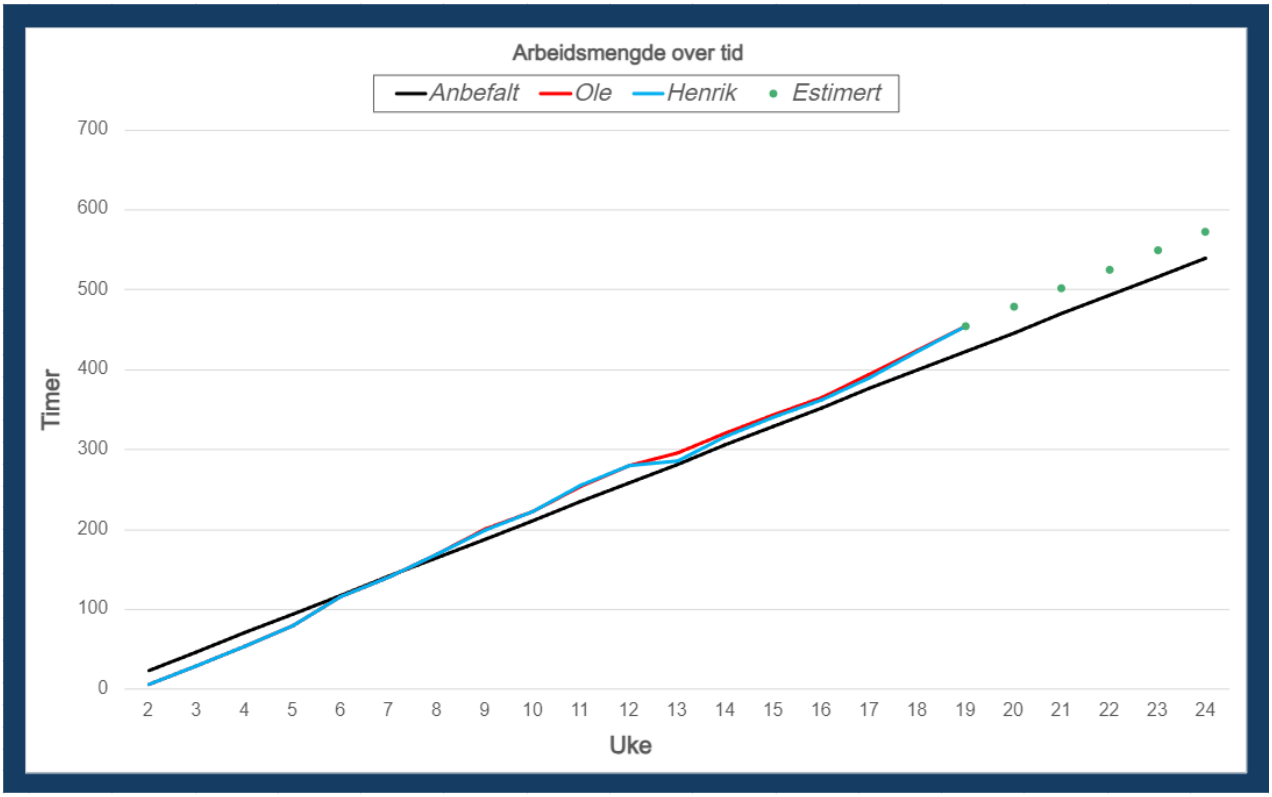
5.4 Prosjektgjennomføring

5.4.1 Tidsbruk

Gruppen har arbeidet inkrementelt gjennom prosjektets levetid, der det har blitt ført timer etter hver arbeidsøkt. Timelisten illustrert i Figur 5-9 viser tidsbruken per gruppemedlem og anbefalt mengde timer mellom uke 2 og 24. Gruppemedlemmene markeres med rød og blå farge, imens anbefalt mengde timer illustreres med svart. Anbefalt mengde timer per uke er 23,5 timer som totalt over prosjektet skal bli 540 timer per gruppemedlem fra uke 2 til uke 24. I tillegg til å vise gruppemedlems timer, viser den en stipulert linje basert på estimert fremdrift i prosjektet. Dersom gruppen følger denne fremdriften, vil medlemmene opparbeide 572 timer hver.

Gruppen har holdt en jevn og god fremdrift gjennom hele prosjektet. Litt lavere enn anbefalt de første 5 ukene, for deretter å jevnt ligge litt over anbefalt timeforbruk gjennom hele prosjektet.

Totalt har medlemmene i gruppen opparbeidet 910 timer.
For en mer detaljert timeliste som grafen er basert på, se (Vedlegg III: Prosjekthandbok).
Total tidsbruk, basert på normal tidsbruk etter endt prosjekt er 1145 timer



Figur 5-9: Tidsbruk

6 DISKUSJON

6.1 Brukertesting

Resultatene fra brukertesten viser at brukerne er jevnt over fornøyd med applikasjonen. Det spørsmålet som gav høyest gjennomsnittlig rangering, var *Jeg synes applikasjonen var lett å bruke* med gjennomsnittlig brukertilfredshet på 89,29%. Og det som gav lavest gjennomsnittlig rangering var *Jeg tror jeg ville brukt denne applikasjonen jevnlig* med gjennomsnittlig brukertilfredshet på 53,57%. Resultatene fra brukertesten viser at brukeren stort sett er fornøyd med applikasjonen, de følte den var lett å bruke og forstå. Spørsmålet med lavest score, kan tolkes på flere måter, det kan bety at deltakerne ikke så behovet for å teste bevegeligheten sin jevnlig. Det kan også være at de ikke ville brukt applikasjonen fordi den har manglende funksjonalitet.

Brukerene kom med ønsker til nye funksjonaliteter. Et forslag var en funksjon der videoene som viser utførelsen av øvelsen starter automatisk når det navigeres til neste øvelse. For å forbedre systemets brukervennlighet kunne dette vært implementert som en innstilling der brukeren kan slå dette av eller på.

En annen tilbakemelding dreier seg om synligheten til tekstene på utfør test siden. Dette kan adresseres ved å øke kontrasten mellom teksten og bakgrunnsfargen eller øke tekststørrelsen. En annen løsning til denne utfordringen, er å implementere en funksjon der brukeren får opplest navnet på øvelsen, dette kan forbedre brukeropplevelsen hos brukere med svakt eller nedsatt syn.

Enkelte brukere ønsket også flere øvelser. Noen øvelser kan raskt implementeres med dagens system, andre krever ny implementasjon. Dagens modell er noe begrenset med antall punkter, for enkelte øvelser vil det kreve oppkobling til en mer avansert HPE modell som viser punkter for flere ulike kroppsdeler.

6.2 Brukergrensesnitt

Brukergrensesnittet ble slik som planlagt i skissene med få unntak. I design skissen vises sidene brukeren er på med en mørkere farge i navigasjonsmenyen, men i applikasjonen er dette ikke implementert. Et annet unntak fra designskissene er at i innstillinger skal det være ikoner ved siden av teksten som indikerer hvilken innstilling som skal justeres.

Et problem som ble oppdaget, var at på visse skjermstørrelser kunne enkelte komponenter i applikasjonen fremstå med feil proporsjoner. For å oppnå et konsistent og responsivt design, burde flexbox og prosentuelle verdier benyttes på komponentene i en høyere grad. For økt robusthet i designet på tvers av ulike mobile enheter, burde brukergrensesnittet testes på flere enheter av variert størrelse.

6.3 Funksjonelle og ikke funksjonelle krav

Applikasjonen oppfyller de fleste funksjonelle og ikke funksjonelle kravene, bortsett fra *Applikasjonen skal være kompatibel med IOS og Android.* Kravet om kompatibilitet er et viktig krav å oppfylle. I 2020 ble det estimert at 51,89% av det mobile markedet i Skandinavia er IOS brukere (mytrendyphone, u.å). Som vil si at uten kompatibilitet med IOS vil Ytir miste store deler av sin brukerbase. Grunnen til at dette kravet ikke ble oppfylt er at gruppen manglet tilgang på en mobil med IOS og en pc med Mac OS. For å kunne utvikle en mobil applikasjon til IOS er dette et krav. Applikasjonen er designet med React Native som i teorien er kompatibel med IOS og Android, men grunnet mangel på utstyr, blir ikke dette testet.

6.4 HPE modell

Gruppen valgte å implementere HPE-modellen MoveNet Lightning i applikasjonen. Denne modellen benyttes fordi den har høy ytelse, noe som er viktig for brukeropplevelsen. Et kompromiss med MoveNet Lightning er at presisjonen er lavere enn med en større og kraftigere modell, dette resulterer i at utregningen av vinkler ikke blir like nøyaktig som med andre modeller. Generelt sett fungerer Lightning modellen best på øvelser med større vinkelutslag, siden modellens unøyaktigheter vil påvirke målingene i mindre grad.

6.5 Validering av utslag

Utslagene blir, som forklart i kapittel 4, validert av en matematisk metode for utregning av vinkler, dette fungerer bra for noen øvelser, men kan gi avvik i presisjon for mer komplekse øvelser. Rotasjon av håndledd og øvelser som innebærer å bøye ryggen er to eksempler som blir vanskelig å implementere med den nåværende metoden for validering. Dagens system håndterer ikke rotasjon av kroppsdeler, og kan ikke utføre målinger av kroppsdeler som ikke dekkes av modellen. For å adressere slike øvelser må en ny HPE modell med evnen til å detektere mer detaljerte kroppsdeler implementeres, samt algoritmer som håndterer beregning av rotasjon.

Et alternativ til nåværende valideringsmetode, er å validere vinkelutslag ved bruk av en klassifiseringsmodell, som avgjør om øvelsen blir korrekt utført. Et klassifiserings-system har mulighet til å oppfatte mindre feil som den nåværende matematiske modellen ikke klarer å oppdage, fordi den sjekker øvelsen i sin helhet i stedet for 3 punkter. Dette ville krevet å trene en modell på hver øvelse, som vil øke belastningen på brukerens mobile enhet og øke kompleksiteten i programmet. En annen utfordring med en slik løsning er mangelen på treningsdata for modellen. Innsamling av treningsdata til et slikt program er tidkrevende, og det vil innebære manuelle vurderinger om hvilke data som skal klassifisere hvilke øvelser.

6.6 Arbeidsprosessen

Arbeidsflyten og samarbeidet gjennom prosjektets levetid har vert en problemfri prosess. Det ble tidlig konsensus innen gruppen om å gjøre en god innsats i prosjektet. Gruppen har jobbet jevnt og iterativt fra prosjektstart til prosjektslutt. Som har ført til at tidsfrister ble møtt og alle målene, med ett unntak, ble nådd.

Gruppen har benyttet minst 3 dager i uken for å arbeide med bachelor prosjektet, og noen ganger i helgen ved behov. Store deler av arbeidet har foregått digitalt. Det har også vært jevnlige møter med veileder annen hver uke.

En stor del av den iterative arbeidsprosessen, var å gjøre endringer basert på oppdragsgivers tilbakemeldinger og spesifikasjoner. Problemløsningen for å implementere endringene ble håndtert så tidlig som mulig.

7 KONKLUSJON OG VIDERE ARBEID

7.1 Konklusjon

Som beskrevet i 1.4 har dette prosjektet som mål å utvikle en mobilapplikasjon med HPE, hvor pasienter kan utføre tester som gir målinger av bevegelighet, effektivt og med god nøyaktighet.

Sluttproduktet ble en helhetlig mobilapplikasjon for Android. Applikasjonen oppnår god ytelse og med unntak av oppkobling til IOS, ble kravene for oppgaven oppfylt. Gruppen er godt fornøyd med resultatet, det har vært mange komponenter og prosesser som måtte utvikles for å få systemet til å fungere slik som planlagt, fra gruppens perspektiv er målet oppnådd.

Ved prosjektstart konsentrerte gruppen seg om å strukturere samt organisere filer og verktøy som var nødvendige for utviklingen av applikasjonen og utformingen av rapporten med de tilhørende vedleggene. I oppstartsfasen ble brukstilfellene for applikasjonen definert gjennom et brukstilfellediagram. Brukergrensesnittet ble deretter utviklet gjennom flere iterasjoner, hvor kvaliteten på designskissene forbedret seg for hver iterasjon. Databasen ble designet ved hjelp av et ER-diagram og implementert med SQL-kode. Frontend og backend ble utviklet parallelt og senere integrert opp mot hverandre gjennom bruk av REST API. Til slutt ble applikasjonen brukertestet for å innhente anonyme deltakeres subjektive tilbakemeldinger om både ønsket og eksisterende funksjonalitet, som kan implementeres i påfølgende iterasjoner.

Basert på tidligere forskning valgte gruppen å implementere HPE modellen MoveNet Lightning, per dags dato er denne vist å gi best ytelse på mobile applikasjoner, men på bekostning av presisjon. Modellens ytelse er tilpasset dagens mobile enheter og har vist tilstrekkelig presisjon for applikasjonens behov.

7.2 Videre arbeid

Applikasjonen har forbedringspotensialer, og det er funksjoner som burde implementeres for å ferdigstille og forbedre systemet. Endringer og funksjonaliteter som gruppen anbefaler for å videreutvikle systemet, er funksjonaliteter som ble oppdaget under utviklingen og under brukerundersøkelsen.

Første prioritet burde være å koble applikasjonen opp mot IOS, siden dette var et funksjonelt krav som gruppen ikke oppfylte. Oppkobling til IOS vil tilgjengeliggjøre applikasjonen for en større brukerbase, som betyr at flere kan få nytte av applikasjonen.

Videre anbefales det å legge til flere øvelser i databasen, slik at brukere kan få mer relevante program og utføre ROM målinger for kroppsdeler de ønsker å få vurdert. Enkelte øvelser kan legges til ved eksisterende system, andre øvelser krever mer spesifikk implementasjon. MoveNet Lightning markerer 17 punkter på kroppen for de største leddene og er ikke trent på mindre ledd. Dette betyr at øvelser med håndledd, fingre og nakke ikke kan implementeres med dagens løsning. Men øvelser der man bøyer større kroppsdeler som albue og kne kan implementeres.

Det er også en rekke med mindre endringer og funksjoner som kan implementeres for at brukeren skal få en mer helhetlig opplevelse av applikasjonen. Eksempelvis, dobbeltrykk på kamera-skjermen for å flippe kameraet, med en slik funksjon gjøre det lette for en bruker å få hjelp med å utføre øvelsene.

Tilbakemeldingene fra brukertesten hinter til at det trengs flere innstillinger for ulike funksjoner i applikasjonen. Det ble etterspurt en mulighet for automatisk start av instruksjonsvideo ved ny øvelse. Det ble også etterspurt en funksjon som gir brukere opplæring av applikasjonen.

Vår undersøkelse ble noe kortfattet med få deltakere, så en brukertest med flere deltakere er å anbefale. Brukertester lar ekte brukere gi en direkte input på hva de synes om applikasjonen, som vil være verdifullt for videre arbeid.

Det er viktig å notere seg at gruppen ikke har tatt hensyn til sikkerhet under utvikling av applikasjonen. For å oppnå et produkt som kan benyttes i en produksjonssammenheng er det avgjørende å ta hensyn til sikkerhet. En viktig del av videreutviklingen burde være å utforske krav til sikkerhet ved denne typen mobil applikasjon og implementere nødvendige sikkerhetstiltak for at kravene oppfylles.

8 Referanser

Alvær, A. L. & Straume-Næsheim, T. M., 2023. *bevegelighet.* [Internett]
Available at: https://sml.snl.no/bevegelighet
[Funnet 20 Februar 2024].

Beletti, F., Yu-Hui Chen, A. O. & Votel, R., u.å. *MoveNet.SinglePose,* s.l.: s.n.

Cao, Z. et al., 2019. *OpenPose: Realtime Multi-Person 2D Pose,* s.l.: arXiv:1812.08008v2.

CodeAcademy, 2024 . *What is REST?.* [Internett]
Available at: https://www.codecademy.com/article/what-is-rest
[Funnet 26 April 2024].

Datatilsynet, 2023. *Hva er personvern?.* [Internett]
Available at: https://www.datatilsynet.no/rettigheter-og-plikter/hva-er-personvern/
[Funnet 20 Februar 2024].

Devaraj, K., 2024. *What is Adhoc Testing?.* [Internett]
Available at: https://testsigma.com/blog/adhoc-testing/
[Funnet 8 Mai 2024].

Eng-Galåen, T. E., 2024. *Den viktige journalen.* [Internett]
Available at: https://www.fysioterapeuten.no/epikriser-fysioterapeut-fysioterapeuter/den-viktige-journalen/152969
[Funnet 16 Februar 2024].

Fielding, R. T., 2000. *Architectural Styles and the Design of Network-based Software Architectures,* s.l.: Doctoral dissertation, University of California, Irvine, 2000. .

Figma, Inc., 2024. *How you design, align and build. Do it together with Figma.* [Internett]
Available at: https://www.figma.com/
[Funnet 20 Februar 2024].

Gallavin, G., 2014. *System Usability Scale (SUS): Improving Products Since 1986.* [Internett]
Available at: https://digital.gov/2014/08/29/system-usability-scale-improving-products-since-1986/
[Funnet 28 April 2024].

Github , 2024. *Let's build from here.* [Internett]
Available at: https://github.com/about
[Funnet 20 Februar 2024].

Google, 2021. *movenet.* [Internett]
Available at: https://www.kaggle.com/models/google/movenet/tfJs/singlepose-lightning
[Funnet 23 April 2024].

Google, 2024. *Meet Android Studio.* [Internett]
Available at: https://developer.android.com/studio/intro
[Funnet 20 February 2024].

Jo, B. & Kim, S., 2022. *Comparative Analysis of OpenPose, PoseNet, and MoveNet Models for Pose Estimation in Mobile Devices,* Seoul: iieta.

Korvyan, U. C., 2024. *The Importance of Mobile App Performance.* [Internett]
Available at: https://virgosol.com/en/blog/detail/the-importance-of-mobile-app-performance
[Funnet 17 Februar 2024].

Llanasas, R., 2020. *How AI is Transforming Mobile Technology.* [Internett]
Available at: https://www.greenbook.org/insights/insights-technology/how-ai-is-transforming-mobile-technology
[Funnet 6 Mars 2024].

Lucid Software Inc., 2024. *Where seeing becomes doing..* [Internett]
Available at: https://www.lucidchart.com/pages/?
[Funnet 20 Februar 2024].

Meta Platforms, Inc., 2024 . *Create native apps for Android, iOS, and more using React.* [Internett]
Available at: https://reactnative.dev/
[Funnet 20 Februar 2024].

Microsoft , 2024. *What is Azure?.* [Internett]
Available at: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-azure
[Funnet 20 February 2024].

Microsoft, 2024. *Download .Net For Windows.* [Internett]
Available at: https://dotnet.microsoft.com/en-us/download
[Funnet 20 Februar 2024].

Microsoft, 2024. *GitHub Copilot and Visual Studio 2022.* [Internett]
Available at: https://visualstudio.microsoft.com/#vscode-section
[Funnet 20 Februar 2024].

Microsoft, 2024. *Øk produktiviteten med kunstig intelligens.* [Internett]
Available at: https://www.microsoft.com/nb-no/microsoft-365
[Funnet 1 Mars 2024].

mytrendyphone, u.å. *Hvor mange mennesker har smarttelefoner i Norden?.* [Internett]
Available at: https://www.mytrendyphone.no/shop/cms-rapport-smarttelefonbruk-i-norden.html#:~:text=Mens%20iOS-enheter%20har%20popularitet,det%20totale%20eierskapet%20i%20Finland
[Funnet 30 April 2024].

Odemakinde, E., 2024. *Human Pose Estimation with Deep Learning – Ultimate Overview in 2024.* [Internett]
Available at: https://viso.ai/deep-learning/pose-estimation-ultimate-overview/#:~:text=Human%20pose%20estimation%20and%20tracking,of%20objects%20such%20as%20cars
[Funnet 20 Februar 2024].

OpenAI, 2024. *ChatGPT.* [Internett]
Available at: https://openai.com/chatgpt
[Funnet 20 Februar 2024].

OpenAI, 2024. *PostgreSQL Query Examples.* [Internett]
Available at: https://chat.openai.com/share/3dc0a760-141c-4989-9668-aa7cbdddec10
[Funnet 19 Januar 2024].

Paula, 2024. *PremierAgile.* [Internett]
Available at: https://premieragile.com/pair-programming-pros-cons/
[Funnet 21 Februar 2024].

Schwaber, K. & Sutherland, J., 2020 . *The 2020 Scrum GuideTM.* [Internett]
Available at: https://scrumguides.org/scrum-guide.html#scrum-team
[Funnet 21 Februar 2024].

Sears, B., 2023. *What Is Range of Motion?.* [Internett]
Available at: https://www.verywellhealth.com/overview-range-of-motion-2696650
[Funnet 20 Februar 2024].

Store Norske Leksikon, 2024. *fossefallsmetode.* [Internett]
Available at: https://snl.no/fossefallsmetode
[Funnet 21 Februar 2024].

Tensorflow, 2021. *Next-Generation Pose Detection with MoveNet and TensorFlow.js.* [Internett]
Available at: https://blog.tensorflow.org/2021/05/next-generation-pose-detection-with-movenet-and-tensorflowjs.html
[Funnet 20 Februar 2024].

Tensorflow, 2022. *TensorFlow Lite.* [Internett]
Available at: https://www.tensorflow.org/lite/guide
[Funnet 1 Mars 2024].

Tensorflow, 2024. *MoveNet: Ultra fast and accurate pose detection model..* [Internett]
Available at: https://www.tensorflow.org/hub/tutorials/movenet
[Funnet 23 Mars 2024].

9 VEDLEGG

Vedlegg I: Systemdokumentasjon

Vedlegg II: Visjonsdokument

Vedlegg III: Prosjekthandbok

Vedlegg IV: Kravspesifikasjon

Vedlegg V: Ytir_kildekode

Vedlegg VI: SUS skjema utviklet av John Brooke

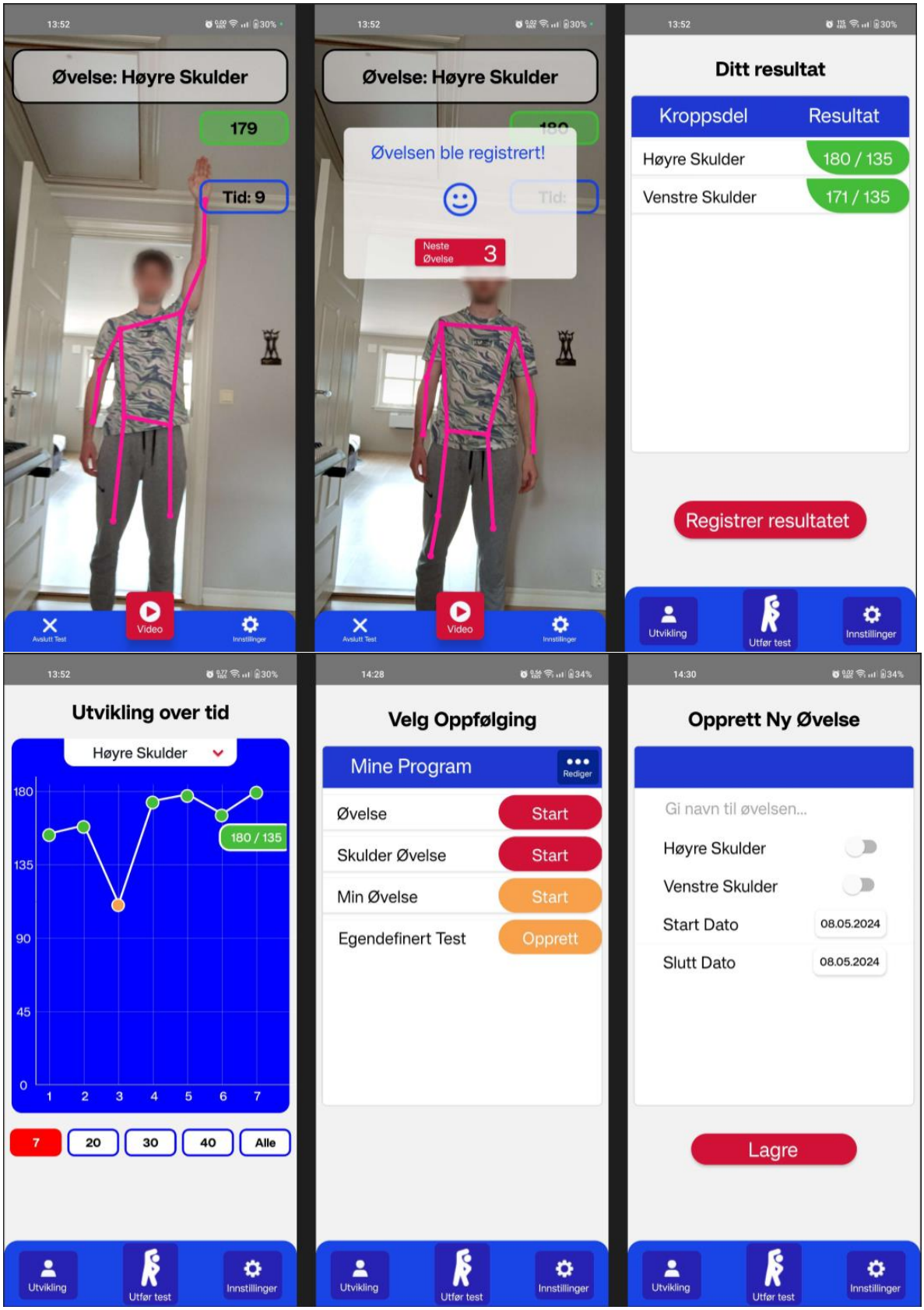
	Strongly Disagree	Somewhat Disagree	Neutral	Somewhat Agree	Strongly Agree
1. I think I would like to use this tool frequently.	☐	☐	☐	☐	☐
2. I found the tool unnecessarily complex.	☐	☐	☐	☐	☐
3. I thought the tool was easy to use.	☐	☐	☐	☐	☐
4. I think that I would need the support of a technical person to be able to use this system.	☐	☐	☐	☐	☐
5. I found the various functions in this tool worked well together.	☐	☐	☐	☐	☐
6. I thought there was too much inconsistency in this tool.	☐	☐	☐	☐	☐
7. I would imagine that most people would learn to use this tool very quickly.	☐	☐	☐	☐	☐
8. I found the tool very difficult to use.	☐	☐	☐	☐	☐
9. I felt very confident using the tool.	☐	☐	☐	☐	☐
10. I needed to learn a lot of things before I could get going with this tool.	☐	☐	☐	☐	☐

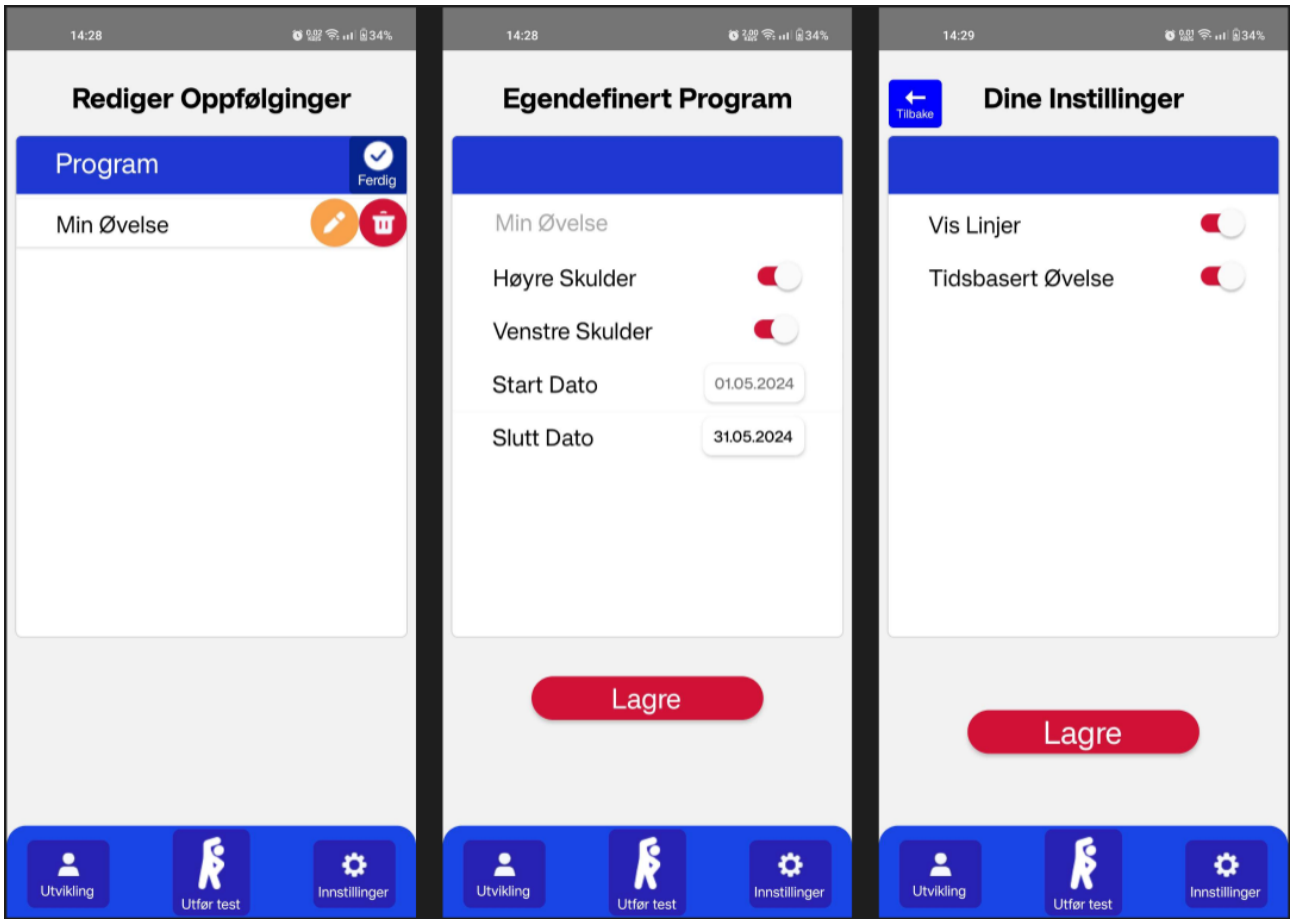
Vedlegg VII: Svar på brukerundersøkelsen

	B 1	B 2	B 3	B 4	B 5	B 6	B 7	AVG S
	7,5	7,5	5	7,5	2,5	5	2,5	53,57
	7,5	5	10	10	5	7,5	7,5	75,00
	10	7,5	10	10	10	7,5	7,5	89,29
	7,5	5	10	7,5	7,5	10	10	82,14
	7,5	7,5	10	10	7,5	7,5	5	78,57
	5	5	10	10	2,5	7,5	7,5	67,86
	10	7,5	10	10	7,5	5	7,5	82,14
	10	5	10	10	2,5	7,5	10	78,57
	10	5	10	10	7,5	7,5	7,5	82,14
	10	7,5	10	10	7,5	7,5	10	89,29
	7,5	5	10	7,5	2,5	5	5	60,71
AVG B	84,09	61,36	95,45	93,18	56,82	70,45	72,73	
TOTAL AVG	76,30							

B1, B2 … står for Bruker1, Bruker2 … og kolonnene under representerer svarene til de ulike deltakerene. Svarene til deltakerne har samme rekkefølge som spørsmålene utviklet av gruppen. Til høyre for kolonnene under AVG S ser vi en gjennomsnittlig score per spørsmål. Under svarene markert med AVG B, ser vi gjennomsnittlig hvor fornøyd en deltaker var med applikasjonen. Til slutt ser vi gjennomsnittlig tilfredshet ved TOTAL AVG.

Vedlegg VIII: Skjermbilder fra Applikasjonen




14:28
34%
Rediger Oppfølginger
Program
Ferdig
Min Øvelse
Utvikling
Utfør test
Innstillinger
14:28
34%
Egendefinert Program
Min Øvelse
Høyre Skulder
Venstre Skulder
Start Dato
01.05.2024
Slutt Dato
31.05.2024
Lagre
Utvikling
Utfør test
Innstillinger
14:29
34%
Tilbake
Dine Instillinger
Vis Linjer
Tidsbasert Øvelse
Lagre
Utvikling
Utfør test
Innstillinger