

Applikation til rehabilitering af patienter med kronisk obstruktiv lungesygdom

Bachelorprojekt 6. semester



Skrevet af
Gruppe 17gr6403

6. Semester

School of Medicine and Health

Sundhedsteknologi

Fredrik Bajers Vej 7A
9220 Aalborg

Titel:

Applikation til rehabilitering af
patienter med kronisk obstruktiv
lungesygdom

Tema:

Design af sundhedsteknologiske
systemer

Projektperiode:

P6, Foråret 2017

Projektgruppe:

17gr6403

Synopsis:



Medvirkende:

Birgithe Kleemann Rasmussen
Linette Helena Poulsen
Maria Kaalund Kroustrup
Mads Kristensen

Vejleder:

Hovedvejleder: Lars Pilegaard Thomsen

Sider:

Bilag:

Afsluttet: XX/05/2017

Offentliggørelse af rapportens indhold, med kildeangivelse, må kun ske efter aftale med forfatterne.

Forord og læsevejledning

Forord

Dette bachelorprojekt er udarbejdet af gruppe 17gr6403 på 6. semester Sundhedsteknologi på Aalborg Universitet i perioden 1. februar til 30. maj år 2017. Projektet tager udgangspunkt i det overordnede tema "Design af sundhedsteknologiske systemer" og projektforslaget "Udvikling af KOL patientens nye bedste ven - den smarte KOL trænings-app!", som er stillet af Lars Pilegaard Thomsen. Læringsmålet for dette projekt er ifølge studieordningen følgende: "Bachelorprojektet er afslutningen på bacheloruddannelsen og den studerende skal kunne demonstrere evner, som er relevante for arbejdsmarkedet og for en videre videnskabelig uddannelse [1]."

Vi vil gerne takke hovedevejleder Lars Pilegaard Thomsen for vejledning og feedback gennem hele projektperioden.

Læsevejledning

Projektet er delt op i to dele, herunder problemanalyse og en problemløsning. I problemanalysen analyseres den opstillede problemstilling, hvor problemløsningen omhandler analyse, design, implementering og test af et system. Der er udarbejdet et metodeafsnit til hver del, som beskriver den anvendte metode i det pågældende afsnit. De to dele afsluttes med en syntese, der omfatter diskussion, konklusion samt perspektivering. Dette efterfølges af litteraturliste samt bilag.

I dette projekt anvendes Vancouver-metoden til håndtering af kilder. De anvendte kilder nummereres fortløbende i kantede parenteser. Er kilderne angivet før punktum i en sætning henvender denne sig til den pågældende sætning. Er kilden angivet efter punktum henvender denne sig til det foregående afsnit. I litteraturlisten ses kilderne, der er angivet med forfatter, titel og årstal. Forkortelser i rapporten er første gang skrevet ud, efterfulgt af forkortelsen angivet i parentes. Herefter anvendes forkortelsen fremadrettet i rapporten.

Rapporten er udarbejdet i L^AT_EX, og app'en er udviklet i Android Studio version 2.3.

Indholdsfortegnelse

Kapitel 1 Indledning	1
1.1 Initierende problemstilling	1
Kapitel 2 Metode	2
2.1 Opbygning af rapporten	2
2.2 Vidensindsamling	3
Kapitel 3 Problemanalyse	4
3.1 Kronisk obstruktiv lungesygdom	4
3.1.1 Symptomer	5
3.1.2 Diagnose	5
3.1.3 Behandling	9
3.1.4 Prognose	10
3.2 Rehabilitering af KOL-patienter	10
3.2.1 Rehabiliteringsforløb	10
3.3 Efter rehabiliteringsforløb	11
3.4 Projektafgrænsning	11
3.5 Problemformulering	12
Kapitel 4 Metode	13
4.1 Objektorienteret programmering	13
4.1.1 Unified Modellig Language	14
4.2 Unified Process	16
Kapitel 5 Systemanalyse	18
5.1 Systembeskrivelse	18
5.2 Kravspecifikationer	19
5.2.1 Use case	19
5.3 Funktionalitet	21
5.4 Analyseklasser	34
Kapitel 6 Systemdesign	36
6.1 Designafgrænsning	36
6.2 Objektorienteret design	36
6.3 Design af database	56
6.3.1 ER-diagram	56
6.3.2 Schema	57
Kapitel 7 Implementering	59
7.1 Android Studio	59
7.2 Implementering af grænseflader	59

7.3	Implementering af model og controller klasser	60
7.3.1	Implementering af database	61
7.3.2	Implementering af tilpasning af træningsniveau	62
7.3.3	Implementering af resultater	63
Kapitel 8 Test		64
8.1	Database	64
8.2	Log ind	65
8.3	Kategorisering	66
8.4	Tilpasning af træningsniveau	67
8.5	Træning	70
8.6	Resultater	73
8.7	Venneliste	74
8.8	Redigering	75
8.9	Log ud	76
Kapitel 9 Syntese		78
9.1	Diskussion	78
9.1.1	Kravsspecifikationer	78
9.1.2	Design	78
9.1.3	Implementering	80
9.1.4	I praksis	80
9.2	Konklusion	81
9.3	Perspektivering	82
Litteratur		83
Bilag A Bilag		86
A.1	Bilag A	86

Kapitel 1

Indledning

Kronisk obstruktiv lungesygdom (KOL) er en kronisk inflammatorisk lungesygdom, der ødelægger bronkiernes vægge og/eller danner forsnævringer i luftvejene. Dette forårsager, at lungefunktionen gradvist nedsættes.[2] Bronkiernes ødelagte vægge reducerer lungernes overflade, som mindsker luftudvekslingen. Forsnævringerne i luftvejene blokerer, hvorfor luft ikke længere kan passere frit igennem. Det kræver derfor mere arbejde ved ventilation end normalt.[3]

I Danmark er der ca. 430.000 mennesker med KOL, hvortil der årligt kommer 10.000 nye tilfælde [4]. Den årlige mortalitet er 3.500, hvilket gør KOL til den fjerde hyppigste dødsårsag i Danmark.[2] På verdensplan er KOL på nuværende tidspunkt den tredje hyppigste dødsårsag [5].

KOL opstår af skadelige partikler samt gasser og miljøpåvirkninger. Tobaksrygning samt passiv rygningen udgør 85 – 90% af tilfældene, hvilket gør disse til den hyppigste årsag til KOL.[2, 6, 7, 4] Miljøpåvirkninger kan blandt andet være dårligt arbejdsmiljø, som eksempelvis arbejde med asbest eller opvækst i dårligt miljø, hvilket kan påvirke barnets lunger til ikke at udvikle sig ordentligt. Miljøpåvirkninger kan derved resultere i en accelererende reduktion i lungefunktionen.[7]

Lungefunktionen nedsættes gradvist over mange år, hvilket gør, at KOL først kommer til udtryk sent i sygeforløbet. Dette kan resultere i, at patienter først opsøger deres læge, når lungefunktionen er halveret.[6] Symptomer forbundet med KOL opleves som åndenød samt hoste ved fysisk aktivitet, derudover er der en tendens til hyppige eksacerbationer. Eksacerbationer er akut forværring af patienters tilstand, hvilket kræver behandling.[2, 6] Derudover er der en række komorbiditeter, der kan være forårsaget af åndenød samt svage perifere muskler, som opleves ved KOL. Disse fremtræder som kardiovaskulære sygdomme, type-2 diabetes, osteoporose, lungecancer og muskelsvækkelse. Foruden de nævnte komorbiditeter, kan patienterne ligeledes opleve psykiske komorbiditeter, såsom depression og angst, da patienterne ofte isolerer sig på grund af generne ved KOL.[6]

KOL kan ikke helbredes, og det er dertil ikke muligt at genvinde den tabte lungefunktion. Dog er det muligt at forhindre yderligere tab af lungefunktionen forårsaget af KOL samt lindre patienters symptomer.[2] Dette leder op til følgende initierende problemstilling.

1.1 Initierende problemstilling

Hvordan er nuværende diagnosticering og behandling af patienter med kronisk obstruktiv lungesygdom, og hvilke rehabiliteringsmuligheder kan tilbydes?

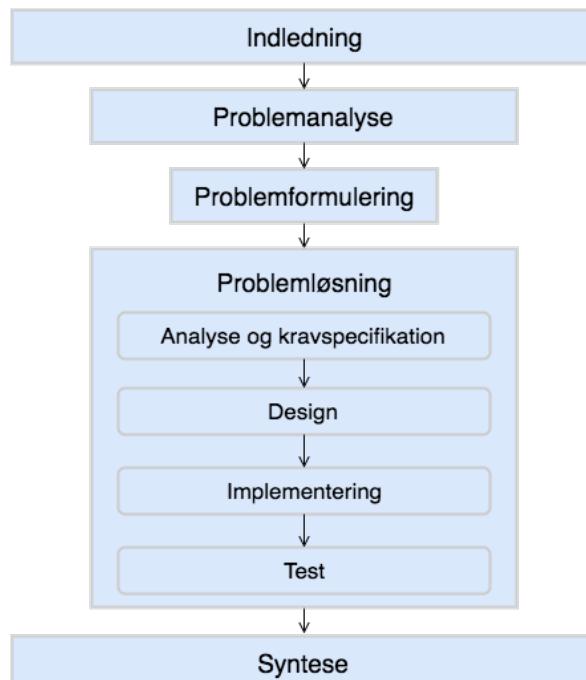
Kapitel 2

Metode

I dette kapitel beskrives metoden anvendt til opbygning af rapporten med henblik på at opnå struktur. Herudover beskrives, hvordan litteratur er indsamlet for at opnå tilstrækkelig viden om KOL.

2.1 Opbygning af rapporten

Denne rapport er opbygget efter AAU-modellen, der tager udgangspunkt i en problembaseret tilgang. AAU-modellen fremgår af figur 2.1. Rapporten indledes med en bred litteratursøgning, hvor det initierende problem opstilles. Dette problem undersøges i problemanalysen, hvor en afgrænsning til problemformuleringen forekommer. Problemformuleringen forsøges endvidere besvaret i problemløsningen, der udformes efter Unified Proces, jf. afsnit 4.2. Herunder vil løsningen analyseres, hvorved der ligeledes opstilles kravspecifikationer. Løsningen vil derudover designes, implementeres og testes. Efterfølgende vil en diskussion af problemanalysen og problemløsningen lede op til besvarelse af problemformuleringen, der forekommer i en samlet konklusion for projektet. Til sidst afsluttes projektet med en perspektivering.



Figur 2.1: Opbygning af rapport ud fra AAU-modellen.

2.2 Vidensindsamling

Der er anvendt ustruktureret og struktureret søgning for at opnå tilstrækkelig viden. Den ustrukturerede søgning er anvendt for at skabe en grundlæggende viden før påbegyndelse af projektskrivning. Denne søgning foregik på Google og AUB, hvor mindre artikler samt medicinske begreber har skabt en grundlæggende viden og forståelse om KOL. Den strukturerede søgning er anvendt til at besvare projektets problemstilling. I denne søgning er der anvendt AUB, PubMed med flere. Derudover er der udarbejdet en model for søgning for, at få en fast struktur over denne. Et eksempel på dette fremgår af tabel 2.1.

Emne	Søgeord
Kronisk obstruktiv lungesygdom	KOL, Chronic Obstructive Pulmonary Disease, COPD, Diagnose, Behandling, Treatment, Incidens, Prävalens.

Tabel 2.1: Eksempel på anvendte søgeord for KOL

Kapitel 3

Problemanalyse

I dette kapitel beskrives kronisk obstruktiv lungesygdom og de tilhørende symptomer. Yderligere undersøges det, hvordan KOL diagnosticeres samt, hvilke behandlingsmuligheder KOL-patienter tilbydes. Heraf analyseres KOL-patienters resultater efter gennemgået rehabiliteringsforløb.

3.1 Kronisk obstruktiv lungesygdom

KOL er en kronisk inflammatorisk sygdom, der resulterer i gradvist nedsat lungefunktion. Inflammationen opstår i luftvejene og lungevævet, hvilket forårsager, at bronkiernes vægge ødelægges og/eller luftvejene forsnævres. Dette medfører, at lungernes overflade reduceres samt en blokering i luftvejene kan forekomme, hvilket forværre ventilationen [3]. På nuværende tidspunkt er KOL den tredje hyppigste dødsårsag på verdensplan [5]. I Danmark er der ca. 430.000 patienter diagnosticeret med KOL, hvortil der årligt kommer 10.000 nye tilfælde [4]. Den årlig mortalitet er på 3.500 patienter, hvilket gør KOL til den fjerde hyppigste dødsårsag i Danmark. [2] KOL rammer i større grad den ældre del af befolkningen [8]. I år 2014 var over 90% af KOL-patienterne over 50 år samt halvdelen af patienterne over 70 år [8].

KOL er beslægtet med to patologier, herunder kronisk bronkitis og emfysem. KOL-patienter oplever ofte begge patologier, men omfanget af disse varierer fra patient til patient.[2] Kronisk bronkitis er luftvejsinflammation, hvor bronkierne i slimhinden er beskadiget, hvilket medfører en øget slimproduktion. Derudover er antallet af cilia mindsket, hvormed transport af slim og støvparkikler fra bronkierne til svælget begrænses, hvorfor der opstår bakterielle infektioner.[9, 10] KOL-patienter med overvejende kronisk bronkitis betegnes blue bloater. Disse patienter har ofte lungeinfektioner, cor pulmonale, hvilket betegner en trykbelastet og med tiden udvidet hypertrofisk samt dårlig fungerende højre ventrikkel. Derudover oplever patienter ofte type 2 respirationssvigt, hvor iltniveauet er lavt og indhold af kuldioxid højt. Den dårlige iltilførsel til ekstremiteter, huden samt læber vil medvirke til, at huden bliver blålig, hvorfor disse patienter omtales blue bloater.[11]

Emfysem skyldes, at lungernes volumen er øget grundet beskadiget lungevæv, herunder destruktion af elastiske fibre og nedbrydning af væggene i de små lungeblærer. Dette medfører, at overfladen som lungerne har til rådighed ved luftudvekslingen mindskes, hvormed små bronkier kan klappe sammen og derved lukke under ventilation.[12, 13] KOL-patienter med overvejende emfysem betegnes pink puffer. Disse patienter lider ofte af alvorlig afmagring eller vægtab med tydelige tegn på nedbrydning af muskelmasse og fedtvæv. Deres brystkasse er tøndeformet og de oplever type 1 respirationssvigt. Type 1 respirationssvigt betegner et lavt iltniveau og normalt indhold af kuldioxid. Disse patienter omtales pink puffer, da deres kroppe ved vejrtrækning pustes op og huden bliver rødlig.[11]

KOL bestemmes ved ratioen mellem forceret eksspiratorisk volumen (FEV1) og forceret

vitalkapacitet (FVC). FEV1 måles ud fra, hvad der udåndes i det første sekund efter en maksimal indånding. FVC er lungevolumen målt i liter. Ved tilfælde af KOL er FEV1/FVC under 70 % af den forventede lungekapacitet.[2]

Der er flere disponerende faktorer til KOL heriblandt skadelige partikler samt gasser, miljøpåvirkninger og genetiske faktorer. Den hyppigste årsag til KOL er tobaksrygning samt passiv rygning, der udgør 85 – 90% af KOL tilfælde.[6, 2, 7, 4] Dertil har undersøgelser vist, at 35 – 40% af tobaksrygere udvikler KOL [14]. Foruden tobaksrygning kan miljøpåvirkninger have betydning for udviklingen af KOL. Opvækst i et dårligt miljø vil blandt andet kunne påvirke barnets lunger til ikke at udvikle sig ordentligt, hvilket kan resultere i en lavere FEV1. Derudover vil et dårligt arbejdsmiljø, som eksempelvis arbejde med asbest, kunne medvirke til en accelererende reduktion i FEV1, der ligeledes kan øge risikoen for KOL.[7]

3.1.1 Symptomer

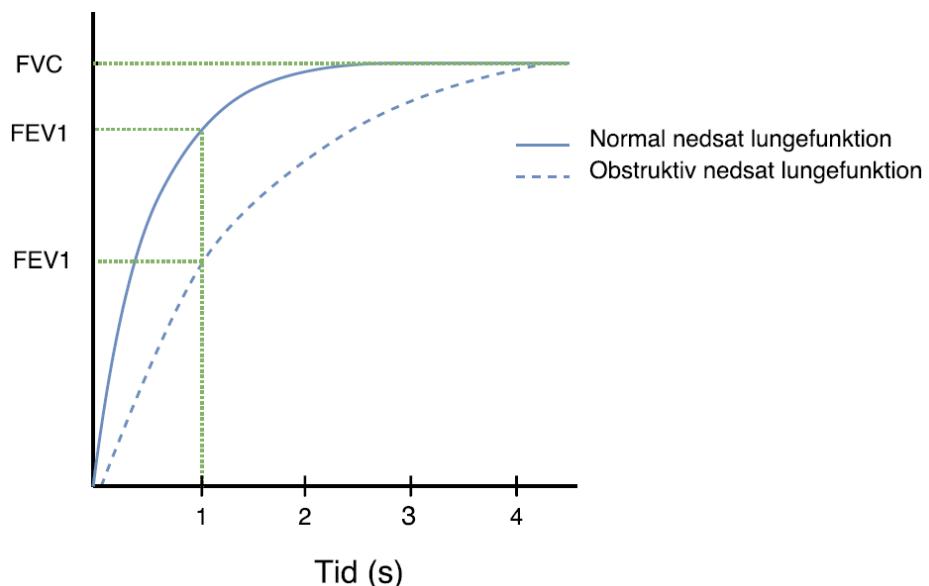
KOL udvikles over mange år, dog bemærkes sygdommen ofte ikke før lungefunktionen er markant nedsat. Dette betyder, at KOL og dens symptomer som regel først kommer til udtryk efter 50 årsalderen [15]. Dette kan i praksis betyde, at patienter først op søger en læge, når deres lungefunktion er halveret [6].

Symptomer på KOL opleves som åndenød og hoste ved fysisk aktivitet. Hosten er ofte med ekspektoration, som hos de fleste patienter er klart eller hvidt.[2] Derudover er der en tendens til hyppig eksacerbationer, hvilket er tilfælde, hvor KOL-patienters tilstand akut forværres og kræver behandling. Eksacerbationer forekommer hyppigere som KOL udvikler sig, og kan tage op til flere uger før patienten ikke længere er påvirket af eksacerbationen [16]. KOL-patienter klassificeret med moderat KOL af tabel 3.2 oplever i gennemsnit 2,68 eksacerbationer pr. år, mens patienter med svær KOL oplever 3,43 eksacerbationer pr. år [16]. Symptomerne i forhold til eksacerbationer opleves som øget åndenød, hoste samt grønt eller gulligt ekspektoration og øget purulens. Denne tilstand skyldes ofte bakterielle infektioner, hvilket udgør ca. halvdelen af tilfældene.[6, 2]

Der er en række komorbiditeter, som hyppigt ses hos KOL-patienter, der kan have en negativ påvirkning på patienters livskvalitet og prognose. Derfor bør patienter regelmæssigt tjekkes for de hyppigste komormiditeter, såsom kardiovaskulære sygdomme, type-2 diabetes, osteoporose, lungecancer, muskelsvekkelse samt angst og depression. Nogle af komorbiditeterne kan skyldes, at åndenød har medført et nedsat fysisk aktivitetsniveau og dermed svage perifere muskler samt vægtab [6]. Desuden har tobaksrygning og generelt dårlig livsstil betydning for udviklingen af disse komorbiditeter.[6, 17] Psykiske komorbiditeter, ofte i form af depression og angst, har en øget forekomst hos patienter med en FEV1 værdi på under 50 % af den forventede værdi. Den øgede risiko for psykiske lidelser skyldes, at KOL kan medføre social isolation og tab af sociale relationer, skyldfølelse og usikkerhed i forhold til fremtiden.[6]

3.1.2 Diagnose

Ved mistanke om KOL undersøges lungefunktionen ved spirogrammålinger, hvor FEV1 og FVC måles. Af figur 3.1 ses spirogrammålinger for henholdsvis patienter med normal og obstruktiv nedsat lungefunktion samt en kombination af disse.[2, 18]



Figur 3.1: Spirometrimålinger for patienter med normal og obstruktiv nedsat lungefunktion. Revideret[2].

Det fremgår af figur 3.1, at der ved obstruktivt nedsat lungefunktion er et fald i FEV1 samt FVC. Der udføres ligeledes en reversibilitetstest for at sikre, at patienter ikke lider af differentialdiagnosen astma. Disse patienter gives broncodilatorer, som hos astmapatienter vil forbedre spirogrammetsningen, mens lungefunktionen for KOL-patienter forbliver uændret.[2, 18] For at undersøge KOL og patienters komorbiditeter undersøges foruden lungefunktionsundersøgelser også BMI, røntgen af thorax, EKG-målinger og blodprøver [18].

Klassifikation af KOLs sværhedsgrad

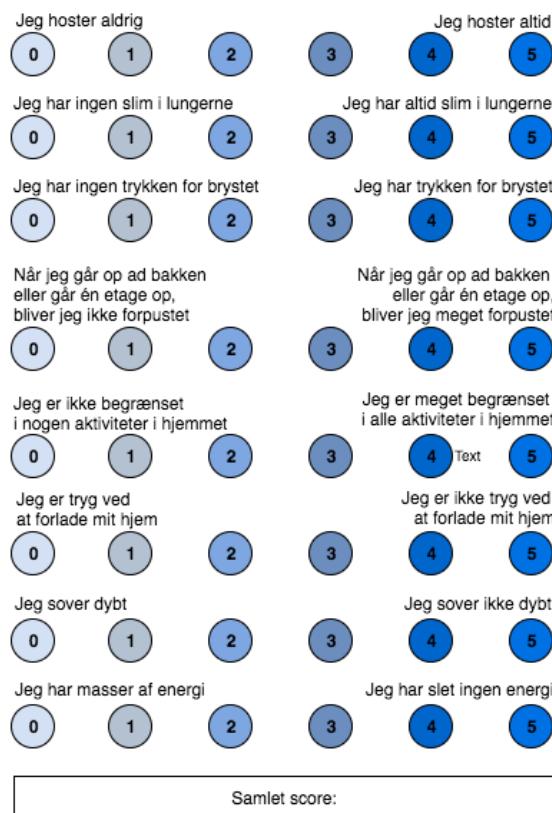
Sværhedsgraden af KOL vurderes på baggrund af patienters symptomer, egne erfaringer og livskvalitet. Denne vurderes ud fra Medical Research Council åndenødsskala (MRC) eller Chronic obstructive pulmonary disease Assessment Test (CAT). Patienter kan efterfølgende inddeltes i klassifikationer med udgangspunkt i MRC, CAT eller ved spirogrammetsninger.[2]

MRC-skalaen er en skala fra 1 til 5, hvor patienter vurderer mængden af aktivitet, som de kan udføre i forhold til åndenød. Skalaen fremgår af tabel 3.1, hvor 1 svarer til, at patienter først oplever åndenød ved meget anstrengelse, og 5 svarer til, at patienter oplever åndenød ved meget lav fysisk aktivitet.[2]

MRC					
1	Jeg får kun åndenød, når jeg anstrenger mig meget.				
2	Jeg får kun åndenød, når jeg skynder mig meget eller går op ad en lille bakke.				
3	Jeg går langsommere end andre på min egen alder, og jeg er nødt til at stoppe op for at få vejret, når jeg går frem og tilbage.				
4	Jeg stopper op for at få vejret efter ca. 100 m eller efter få minutters gang på stedet.				
5	Jeg har for megen åndenød til at forlade mit hjem, eller jeg får åndenød, når jeg tager mit tøj på eller af.				

Tabel 3.1: MRC er en skala fra 1 til 5. Patienter, der oplever åndenød ved meget anstrengelse vurderes til 1, mens patienter, der oplever åndenød ved lav aktivitet vurderes til 5 på MRC-skalaen. Revideret[2].

En anden metode til at vurdere symptomerne ved KOL er ved hjælp af CAT-spørgeskema. Her vurderes otte udsagn fra en skala fra 0 til 5, hvor ingen symptomer angives 0 og mange symptomer angives 5. Ud fra de otte udsagn opnås en samlede score, jo højere den samlede score er, desto værre opleves patienters symptomer. Af figur 3.2 ses CAT-spørgeskema til vurdering af symptomer.[6, 2]



Figur 3.2: CAT er et spørgeskema, hvor patienter vurderer graden af deres symptomer ud fra otte udsagn på en skala fra 0 til 5. Ved ingen symptomer angives karakteren 0, mens ved mange symptomer angives karakteren 5. Patienter opnår en samlede score, jo højere den samlede score er, desto værre opleves patienters symptomer. Revideret[2].

Ud fra MRC-skalaen eller CAT-spørgeskemaet samt lungefunktionstest, antallet af indlægser eller eksacerbationer det seneste år kan KOL-patienter kategoriseres. Patienterne kategoriseres i A, B, C eller D, hvor D er patienter i høj risiko og med mange symptomer. Kategoriseringen fremgår af figur 3.3.



Figur 3.3: KOL-patienter kategoriseres i fire kategorier herunder A, B, C og D. A og B inddeltes i lav risiko, mens C og D er i høj risiko. Revideret[2].

Udover ABCD-kategoriseringen kan sværhedsgraden af KOL udelukkende bestemmes ud fra spirometrimålinger. Sværhedsgraden er klassificeret ud fra retningslinjer opstillet af the Global Initiative for Chronic Obstructive Lung Disease (GOLD).[6] Lungefunktionen vurderes på baggrund af FEV1 i % af den forventede lungekapacitet, hvorfaf det inddeltes i fire stadier. Disse fremgår af tabel 3.2.

GOLD	
SVÆRHEDSGRAD	FEV1 VÆRDI I % AF FORVENTET
1 GOLD Mild	$\geq 80\%$
2 GOLD Moderat	$50\% \leq FEV1 < 80\%$
3 GOLD Svær	$30\% \leq FEV1 < 50\%$
4 GOLD Meget svær	FEV1 < 30% eller FEV1 < 50% og respirationssvigt

Tabel 3.2: GOLD er inddelt efter sværhedsgraderne 1 til 4 herunder mild, moderat, svær og meget svær. Patienter, der har over 80 % af forventet lungekapacitet klassificeres som 1 GOLD mild, mens patienter med under 30 % eller over 50 % af forventet lungekapacitet samt respirationssvigt klassificeres som 4 GOLD meget svær. Revideret[2].

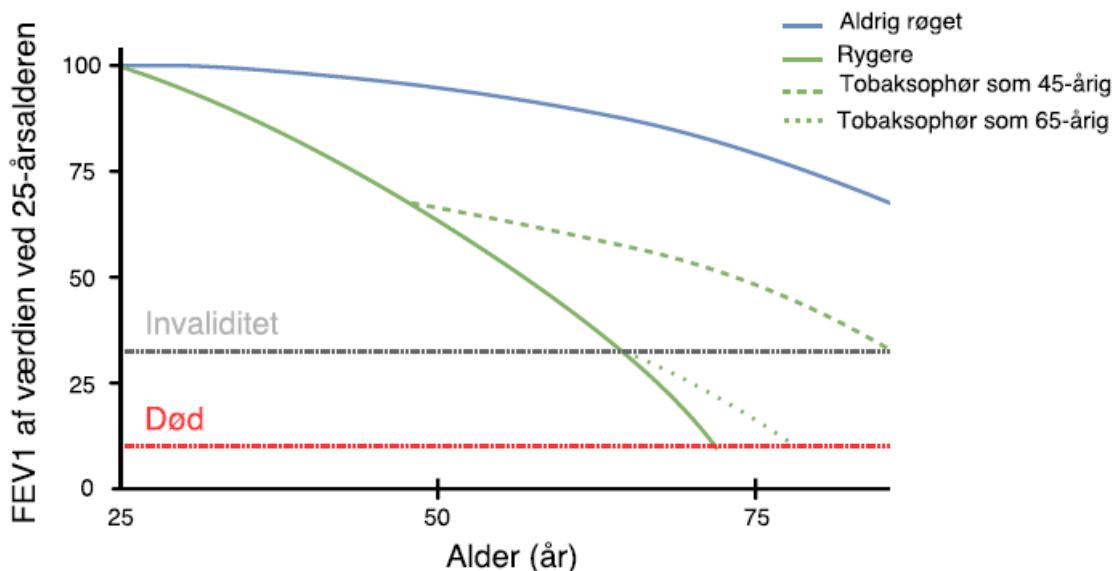
3.1.3 Behandling

Det er ikke muligt at helbrede patienter med KOL, da KOL er en kronisk lungesygdom. Dog er det muligt at forhindre udviklingen af KOL samt lindre symptomerne, hvilket kan opnås ved tobaksafvænning, fysisk aktivitet, kostvejledning og medicin.[2]

En medicinsk behandling består som ofte af langtidsvirkende luftvejsudvidende medicin såsom LABA og/eller LAMA. Patienter med svær KOL samt tendens til mange eksacerbationer kan modtage medicinsk behandling med inhalationssteroid. Derudover kan en kombinationsbehandling af disse også forekomme.[19] KOL-patienter med sekretproblemer tilbydes continuous positive airway pressure (CPAP) eller positive expiratory pressure (PEP-fløjte) [2].

Sammenlignes KOL-patienter over 30 år med den resterende befolkning i Danmark, der ligeledes er over 30 år, har KOL-patienter et højere ressourceræk på sundhedsvæsenet. Dette indebærer eksempelvis kontakter til sundhedsvæsnet og medicin. Udgiften hertil var fem gange højere for KOL-patienter end pr. borger generelt. KOL-patienter har i gennemsnit omkring tre gange så mange hospitalsindlæggelser samt et tre gange så højt medicinforbrug end den resterende befolkning.[8]

Da den tabte lungefunktion ikke kan genvindes, rådes patienterne til ophøre tobaksrygning eller det, der kan være årsagen til KOL eksempelvis dårligt arbejdsmiljø, hurtigst muligt for således at bibeholde den tilbageværende lungefunktion [2]. Det fremgår af figur 3.4, hvordan tobaksrygning kan påvirke lungefunktionen over tid.



Figur 3.4: Fletcher-kurve, som viser faldet af FEV1 over tid for henholdsvis rygere, ikke-rygere og rygere med tobaksophør i 45- og 65-årsalderen. Revideret[2].

Det ses af figur 3.4, at tobaksrygning medvirker til et accelererende tab af FEV1, og dermed udsigt til kortere levetid. På trods af tobaksophør genoprettes FEV1 ikke, dog bremses det accelererende tab af FEV1 til det normale aftag.[6]

3.1.4 Prognose

Dødsfald hos KOL-patienter ses især efter 65-års alderen og udgør 90 % af alle dødsfald citeFolkesundhed2007. KOL-patienter med eksacerbationer har efter indlæggelse en dødelighed på næsten 10 % i løbet af den første måned. Dødeligheden ligger på omkring 64 per 100.000 per år for mænd og 54 per 100.000 per år for kvinder. Udviklingen, hvormed sygdommen progredierer for KOL-patienter er specielt afhængig af, hvorvidt patienter ophører eksponering til den udløsende faktor for eksempel tobaksophør. Det er derfor vigtigt at få en tidlig diagnosticering således, at patienter hurtigt kan få hjælp.[6]

3.2 Rehabilitering af KOL-patienter

Da KOL er en kronisk lungesygdom kan KOL-patienter tilbydes rehabilitering med henblik på at mindske deres symptomer, eksacerbationer samt hospitalindlæggelser [20, 21].

I Danmark henvises KOL-patienter til rehabilitering fra praktiserende læge eller hospital, hvor rehabiliteringen typisk forløber over en otte ugers periode på et sundhedscenter eller hospital. Under dette forløb tilbydes KOL-patienter træning en til to gange om ugen, de resterende dage vil patienter kunne udføre fremviste øvelser hjemme.[17, 22] Som tidligere nævnt kan den tabte lungefunktion ikke genoprettes, dog kan motion ned sætte symptomerne som følge af KOL. Motion styrker patienters muskler samt forbedrer deres kondition, herved vil vejrtrækningen forbedres, da lungerne fremover belastes mindre ved fysisk aktivitet.[3]

Individuel rehabilitering ses som værende fundamental for KOL-patienter, hvor forløbet tilpasses patienters behov med henblik på at opnå det bedste udbytte af rehabiliteringen [17, 23, 24]. Derudover vurderes rehabiliteringen på baggrund af graden af KOL, da KOL fremkommer i flere grader samt med varierende progression [17]. Dertil anses den individuelle rehabilitering ligeledes relavant i forhold til, at KOL-patienter oplever dag til dag variationer i deres tilstand [20].

Rehabiliteringen kan give patienter bedre mulighed for deltagelse i hverdagen, såfremt patienters tilstand tillader det [17, 23, 24]. Opfølgninger kan foretages efter rehabiliteringsforløbet er afsluttet, for således at undersøge om patienter opretholder de gavnlige effekter [22].

3.2.1 Rehabiliteringsforløb

Rehabiliteringsforløbet fokuserer på tobaksafvænning, fysisk træning, kendskab til sygdommen samt ernæringsvejledning [17, 23, 24].

Tobaksafvænning er, som beskrevet i afsnit 3.1.3, et relevant element i forhold til at begrænse udviklingen af sygdommen og bevare mest mulig lungefunktion. Den fysiske træning, der udføres under rehabiliteringen, medvirker til, at patienter kan opnå et bedre udbytte af den resterende lungefunktion samt opnå et bedre fysisk funktionsniveau.[24] Træningen kan ligeledes modvirke eventuelle følger ved KOL, da fysisk træning øger muskelfunktionen samt udsætter træthed, hvilket medfører øget aktivitetstolerance [17]. En problematik kan dog ses ved, at fysisk træning kan resultere i åndenød hos KOL-patienter, der kan forstærkes, hvis patienter påvirkes af angst som følge af åndenød. Dette kan betyde, at KOL-patienter afholder sig fra fysisk træning på grund af frygten for angst.[17, 24]

Et led i rehabiliteringen er ligeledes, at patienter opnår viden indenfor sygdomshåndtering, der omhandler kendskab til og forebyggelse af sygdommen, livsstilsændringer samt håndtering

af eksacerbationer. Her fokuseres blandt andet på de gavnlige effekter ved tobaksophør og regelmæssig fysisk aktivitet, samt hvornår og hvordan eventuel medicin skal indtages. Patienter vil yderligere blive introduceret til energibesparende strategier og vejotrækningsøvelser.[17, 24]

3.3 Efter rehabiliteringsforløb

Gennem studier er det oplyst, at ikke alle patienter er i stand til at opretholde resultaterne efter et halvt til et år, og deres fysiske tilstand falder tilbage til niveauet før rehabiliteringsforløbet [25, 26, 27, 28]. Årsagerne til dette tilbagefald kan blandt andet være som følge af, at rehabiliteringen ikke er med til at gøre patienter mere aktive i hjemmet efter afsluttet forløb, da de falder tilbage til deres tidligere vaner og rutiner [25]. Ligeledes ses det hos patienter, der fortsat træner, at intensiteten og hyppigheden af træningen falder [28]. Dansk Selskab for Almen Medicin (DSAM) anbefaler dertil KOL-patienter at følge et vedligeholdelsesprogram bestående af fire til fem træningssessioner om ugen efter afsluttet rehabiliteringsforløb [6]. Derudover tilbydes KOL-patienter at deltage i forskellige træningssessioner og fællesskaber, hvor de har mulighed for at danne træningsgrupper og afholde arrangementer [24]. Derudover har Lungeforeningen i Danmark forskellige lokalafdelinger, hvor der et par gange årligt afholdes arrangementer for patienter samt pårørende [3]. Fordele ved de forskellige gruppeaktiviteter er, at KOL-patienter kan undgå social isolation samtidig med, at de lærer af hinandens erfaringer i forhold til, hvordan de hver især oplever og håndterer sygdommen. Herved kan sociale fællesskaber være en medhjælpende faktor til vedligeholdelse af effekten ved rehabiliteringen.[6]

Det ses i stigende grad, at telehealth anvendes i sundhedsrelateret sammenhæng for at skabe en forbindelse mellem professionel behandling og self-management uden for sundhedspleje faciliteter [21, 29]. Herunder viser studier positiv anvendelse af telerehabilitering for KOL-patienter [27]. Telerehabiliteringsteknologier inkluderer mobiltelefoner, video og telekonferencer og trådløst udstyr til dataopsamling [30, 27]. Denne form for rehabilitering viste, at KOL-patienter oplevede øget sundhedsrelateret livskvalitet, fysisk aktivitet samt træningskapacitet [27]. I Danmark ses app'en HomeRehab, der har til formål at gøre KOL-patienter i stand til at varetage sig selv ved at opretholde effekterne af rehabiliteringen gennem motivering til daglig træning. Denne app er udviklet af Firmaet Aidcube til anvendelse under og efter et rehabiliteringsforløb. Data fra HomeRehab app'en hjælper også sundhedspersonale, der kan tilgå data via en webportal, med at identificere tegn på sygdomsforværring, hvilket anvendes til at reducere risikoen for hospitalsindlæggelse. HomeRehab testes på nuværende tidspunkt i samarbejde med blandt andet Hvidovre Hospital, Frederiksberg Hospital og Silkeborg Kommune.[31]

3.4 Projektafgrænsning

I dette projekt fokuseres der på KOL-patienter samt deres formåen til at reducere deres symptomer. KOL-patienter tilbydes rehabiliteringsforløb for at få viden om sygdommen, hjælp til tobaksophør samt ernæring og motion. Rehabiliteringsforløb har til formål at nedsætte symptomerne, således en bedre livskvalitet kan opnås.[17, 3, 23, 24] Studier viser dog, at KOL-patienter har svært ved at opretholde resultaterne efter et afsluttet rehabiliteringsforløb [25, 26, 27, 28]. I Danmark ses forskellige værktøjer til at forsøge at opretholde resultaterne, blandt andet vedligeholdelsesprogrammer, sociale fællesskaber samt forskellige app's [24, 31].

De sociale fælleskaber viser positive resultater i forhold til motivation til opretholdelse af den forbedrede livsstil [6]. På baggrund af dette udvikles en app med fokus på social interaktion og motivation til vedligeholdelse af resultaterne fra rehabiliteringsforløb.

3.5 Problemformulering

Hvordan udvikles en app til at vejlede og motivere KOL-patienter til hjemmetræning i forlængelse af rehabiliteringsforløb med henblik på at mindske symptomer forbundet med KOL?

Kapitel 4

Metode

I dette kapitel beskrives de metoder, der anvendes i problemløsningen, herunder de grundlæggende principper inden for objektorienteret programmering samt forskellige diagrammer, der anvendes inden for dette. Derudover beskrives modeller, der kan anvendes til udviklingen af app's.

4.1 Objektorienteret programmering

Objektorienteret programmering er et programmeringsparadigme, som anvendes til at analysere, designe, implementere samt udvikle app's. Hyppige termer inden for objektorienteret programmering er blandt andet objekter, klasser, indkapsling, nedarvning og polymorfi.[32, 33]

I objektorienteret programmering opdeles programmeringskoden i klasser, hvor hver klasse fungerer som en opskrift for et objekt. Hvert objekt er en instans af en bestemt klasse, hvor en klasse kan være bygget op omkring en eller flere instanser. De forskellige objekter repræsenterer hver sin del af app'en og indeholder data og logik. Derudover har objekterne mulighed for at kommunikere mellem hinanden. Objekter er karakteriseret ud fra deres egenskaber, og deres funktioner er beskrevet ved metoder.[32, 33] Eksempler på egenskaber og metoder fremgår af tabel 4.1.

Egenskaber	Metoder
Navn	Gå
Køn	Løbe
Alder	Hoppe
Højde	Sove
Vægt	Tale

Tabel 4.1: Objekter karakteriseres ud fra deres egenskaber som for eksempel navn, mens metoder beskriver deres funktion som for eksempel sove.

Objektorienteret programmering består af tre grundprincipper, herunder indkapsling, nedarvning og polymorfi. Indkapsling er en illustration af, at objekter både indeholder egenskaber og metoder. Egenskaber opbevarer data, mens metoder anvendes til at behandle data. Indkapsling kan både have synlige og skjulte informationer. Synlig information udgør ofte grænsefladen, såsom knapper og display, mens skjult information kan være implementeringen af grænsefladen. Dette gør sig også gældende for objekter, hvilket defineres som public eller private. Ved public har alle objekter adgang til metoderne, mens private kun er metoder med samme objekt, der kan tilgå denne. Nedarvning betyder, at et objekt kan arve data og funktioner fra et andet objekt. Dette muliggør, at objektet kan udvides med ekstra data og

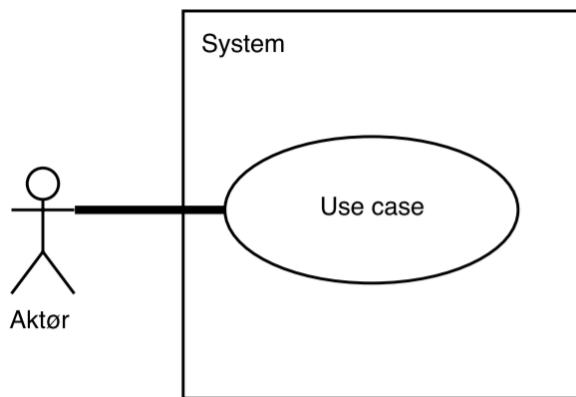
funktioner. Polymorfi giver mulighed for, at to klasser kan have samme grænseflade. Denne er defineret ved nedarvningen.[32]

4.1.1 Unified Modellig Language

En af de anvendte sprog indenfor objektorienteret programmering er standarden Unified Modelling Language (UML). Ud fra denne standard anvendes modeller til at visualisere struktur og egenskaber af systemet. Derudover relaterer metoderne til analyse og design af systemet. Til visualiseringen anvendes forskellige UML diagrammer, som kan opdeles i tre kategorier, herunder adfærds-, struktur- og interaktiondiagrammer. Adfærdsdiagrammer er for eksempel use case- og aktivitetsdiagrammer. Strukturdiagrammer kan være klassediagrammer, mens interaktionsdiagrammer kan være sekvensdiagrammer. [34, 35].

Use case diagrammer

Use case diagrammer benyttes til at illustrere aktørernes interaktion med et system samt, hvordan forskellige use cases interagerer mellem hinanden. Dertil er use case diagrammer med til at repræsentere funktionelle krav for systemet. [35] Et eksempel på et use case diagram ses af figur 4.1.



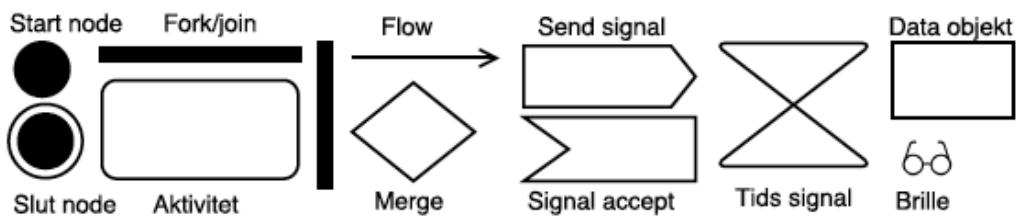
Figur 4.1: Simpelt use case diagram.

Af figur 4.1 ses aktørens interaktion med use case visualiseret som en streg mellem de to. I et use case diagram vil aktøren definere en person, der kan tilgå systemets funktionaliteter. Dette kan eksempelvis være en person, rolle, objekt eller en anden given genstand. Hertil vil den enkelte use case beskrive en handling eller funktionalitet i systemet. Ved anvendelse af flere use cases kan der opstå et forhold mellem de enkelte use cases. Dette forhold visualiseres med en stiblet pil mellem use casene og kan enten være include eller extend. Hvis en use case ikke kan stå alene og derfor er nødt til at arve noget fra en anden use case er denne include. Modsat kan extend anvendes, hvis use casen kan stå alene. [34, 35]

Aktivitetsdiagrammer

Aktivitetsdiagrammer anvendes til at beskrive, hvad der sker i programmet, herunder proceduremæssig logik, business processer og arbejdsflow. Aktiviteter kan opdeles i subaktiviteter eller metoder. Subaktiviteter vil fremgå af diagrammet ved et rivesymbol, mens metoder vil fremgå ved syntaksen klasse-navn::metode-navn. Aktivitetsdiagrammer

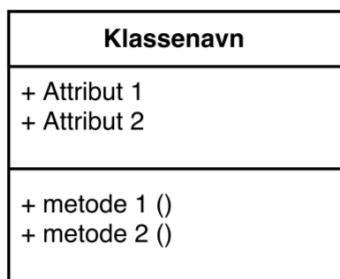
fortæller ikke hvem, der udfører aktiviteten, hertil kan der anvendes skillevægge, som viser, hvilken aktivitet en klasse eller organisation tilhører. For at holde et aktivitetsdiagram enkelt kan der anvendes et brillesymbol i en aktivitet. Denne aktivitet vil efterfølgende kunne beskrives yderligere i et nyt aktivitetsdiagram.[34] Symboler, der kan anvendes inden for aktivitetsdiagrammer, fremgår af figur 4.2.



Figur 4.2: Symboler der kan anvendes i aktivitetsdiagrammer. Revideret [34].

Klassediagrammer

Klassediagrammer anvendes som redskab til at beskrive strukturen i et givet system og dermed skabe overblik over forskellige klasser og relationer, der indgår i systemet [34]. Det fremgår af figur 4.3, at hver klasse identificeres ud fra et unikt klassenavn, hvor der yderligere kan tildeles attributter og metoder til klassen.



Figur 4.3: I klassediagrammer identificeres klasser ud fra et klassenavn, og dertilhørende attributter og metoder tilføjes nedenfor navnet. Revideret [34].

Attributter og metoder kan markeres med symbolerne; +, - eller #, som symboliserer, at de henholdsvis er public, private eller beskyttede, jf. afsnit 4.1.

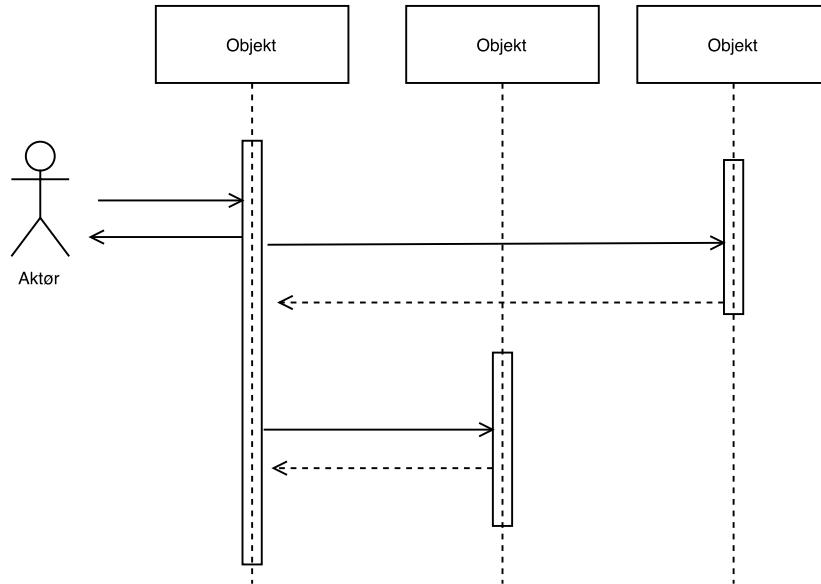
Relationerne mellem klasserne illustreres ved brug af forskellige pile, og disse kan navngives for at tydeliggøre forholdet mellem klasserne. Yderligere kan multipliciteten angives ved at tilføje symbolet *, der angiver "mange", eller specifikke værdier i pilenes ender.

Klassediagrammer kan inddeltes i tre stereotyper, herunder entity, boundary og control, hvilket anvendes til at identificere klasser i analyse og tidlig designfase [?].

- *Entity* anvendes til at lagre og opdatere informationer om objekter. Dens attributværdier gives ofte af en aktør.[?]
- *Boundary* anvendes til interaktion mellem bruger og system og sikre at ændringer i boundary ikke påvirker resten af systemet [?].
- *Control* anvendes til at kontrollere handlinger [?].

Sekvensdiagrammer

Sekvensdiagrammer anvendes til at beskrive detaljer om, hvilke og hvornår forskellige operationer udføres. Disse diagrammer er organiseret efter tid. De forskellige objekter, som anvendes i operationerne er angivet fra venstre mod højre. Sekvensdiagrammer anvendes i design og er udarbejdet ud fra use case diagrammer.[33] Et simpelt sekvensdiagram fremgår af figur 4.4.



Figur 4.4: Simpelt sekvensdiagram. Revideret [33].

Af figur 4.4 fremgår det, at hver objekt i et use case diagram har en kolonne i sekvensdiagrammet. Den vertikale retning beskriver tiden, og de horisontale linjer illustrerer funktionaliteter. For enden af de horisontale pile er der navngivet metoden på objektet. Hver objekt har en livslinje, hvor på der kan illustreres en aflang boks, som beskriver, hvornår objektet er aktivt. [33]

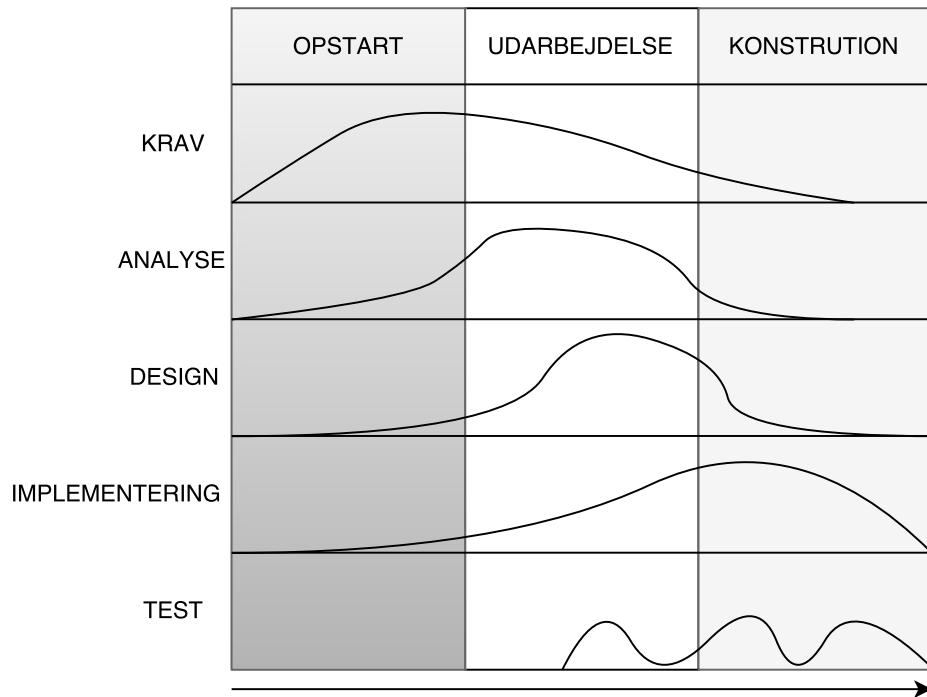
Sekvensdiagrammer kan opdeles i Model-View-Controller (MVC) arkitektur, der har ligheder med de definerede stereotyper i klassediagrammer. MVC anvendes til at organisere systemet og giver større fleksibilitet [33].

- *Model* anvendes til at lagre data [33].
- *View* anvendes til interaktion mellem bruger og system [33].
- *Controller* anvendes til at kontrollere brugerinput og kaldet metoder [33].

4.2 Unified Process

En objektorienteret softwareudviklingsproces er Unified Process (UP), som definerer hvem, hvad, hvornår og hvordan softwaren udvikles. UP er bygget om omkring iterationer, hvor softwareudviklingsprocessen deles op i mindre projekter. Dette gøres da det forventes, at fejl opdages hurtigere og er lettere at løse, hvilket ofte medfører, at projekter gennemføres med succes. Hvert projekt er en iteration og opdeles i arbejdsmængder, såsom krav, analyse, design, implementering og test. Denne arbejdsmængde opdeles yderligere i fire faser, herunder opstart, udarbejdelse, konstruktion og overgang, hvor hver fase afsluttes med en milepæl. Hver fase har en eller flere iterationer. Antallet af iterationer afhænger af projektets størrelse. De forskellige

faser overlappes i forbindelse med projektets fremskreden og arbejdsmængden ændres.[36] Opdeling af projektet og arbejdsmængden ud fra UP fremgår af figur 4.5



Figur 4.5: UP struktur. X-aksen viser tiden over projektet opdelt i opstart, udarbejdelse og konstruktion. Y-aksen viser projektets faser, herunder krav, analyse, design, implementering og test. Kurverne viser arbejdsmængden. Revideret [36]

Af figur 4.5 fremgår softwareprocesudviklingen i dette projekt. Opstart og udarbejdelse anvendes med henblik på den senere implementering i konstruktionsfasen. Den sidste fase, overgang, er udeladt af dette projekt, da der kun udvikles en prototype.¹

¹FiXme Note: Skal ændres lidt til hvis vi oplever fejl

Kapitel 5

Systemanalyse

I dette kapitel beskrives funktionaliteten af den ønskede app. På baggrund af dette opstilles funktionelle samt non-funktionelle krav. Herefter er systemet beskrevet ved hjælp af et use case diagram, hvortil de enkelte funktionaliteter er beskrevet yderligere.

5.1 Systembeskrivelse

I dette projekt udvikles en app, der har til formål at hjælpe KOL-patienter til at opretholde regelmæssig motion efter et endt rehabiliteringsforløb. App'en skal kunne håndtere forskellige træningsformer herunder konditions- samt styrketræning og vejrtækningssøvelser, hvilket alle har symptomreducerende effekt, jf. afsnit 3.2. Ligeledes skal app'en kunne håndtere forskellige træningstyper, som for eksempel gå, løbe eller cykle.

KOL-patienter introduceres samt registreres i app'en i forbindelse med deres rehabiliteringsforløb. Dette skal sikre, at det kun er KOL-patienter, der er tilmeldt rehabiliteringshold, som kan anvende app'en til træning. Ved registrering oprettes KOL-patienter med medlemsID, fornavn, efternavn samt adgangskode.

Der er forskel på, hvor meget fysisk aktivitet KOL-patienter kan udføre, og der skal derved være forskel i varighed af den træning som app'en foreslår. Dertil skal app'en kunne tilpasse træningsniveau ud fra den enkelte patients parametre. Disse parametre består af kategoriseringen af KOL-patienter efter ABCD, jf. kapitel 3, daglige helbredstilstande, der skal tage højde for dag til dag variationer samt tidlige evalueringer fra lignende træningstype. Dette medvirker til, at app'en er henvendt specifik til KOL-patienter i modsætning til andre træningsapp's, der henvender sig til hele befolkningen.

Under selve træningen monitoreres træningen ved brug af timer og GPS. Monitoreringen er med til at vejlede patienten til at følge det valgte træningsniveau.

For at hjælpe KOL-patienter med vedligeholdelse af den daglige træning skal app'en virke motiverende for patienterne. Dette gøres blandt andet ved, at KOL-patienter kan opnå virtuelle belønninger ved at udføre gentagne eller forskellige træningsformer. Desuden skal app'en informere, hvis KOL-patienter ikke har udført træning med app'en i længere tid. [37, 38]

Som nævnt i afsnit 3.3, er det sociale fællesskab en væsentlig faktor for at opretholde resultaterne fra rehabiliteringsforløb. Dette gøres ved, at KOL-patienter kan følge og tilgå andre KOL-patienters virtuelle belønninger, hvormed dette motiverer patienterne til vedligeholdelse af den forbedrede livsstil. Derudover skal sundhedsfagligt personale kunne tilgå KOL-patienters resultater i en database, så de kan følge med i patienters udvikling. De har herved mulighed for at informere patienter om deres indsats, hvilket ligeledes kan have en motiverende effekt.[37, 38]

5.2 Kravspecifikationer

På baggrund af systembeskrivelsen er funktionelle og non-funktionelle krav til app'en opstillet. De funktionelle krav beskriver, hvilke funktionaliteter app'en skal have. De non-funktionelle krav er opstillet ud fra overbevisningen om, at det ikke er krav til systemets funktionalitet, men er relevant i relation til brugervenlighed og brugeroplevelse.

Funktionelle krav

- Brugere skal kunne oprettes i en database
Dette er nødvendigt for, at brugere kan anvende app'en
- Systemet skal kunne gemme og hente data i en database
Dette er nødvendigt for, at brugere kan tilgå brugerdata
- Brugere skal kunne logge ind med et personligt medlemsID og adgangskode
Dette er nødvendigt for at tilgå og sikre, at brugere har deltaget i et rehabiliteringsforløb samt adskille brugernes data
- Systemet skal kunne kategorisere brugere i ABCD på baggrund af CATscore og antallet af indlæggelser på grund af KOL.
Dette er nødvendigt for at kunne tilpasse træningen efter den enkelte bruger
- Brugere skal kunne angive deres daglige helbredstilstand
Dette er nødvendigt for tage højde for daglige variationer og derved tilpasse træningen for den enkelte bruger
- Systemet skal kunne måle tid og afstand
Dette er nødvendigt for at monitorere træningen
- Brugere skal kunne evaluere hver træning
Dette er nødvendigt for at tilpasse træningen efter den enkelte bruger
- Systemet skal kunne sende en notifikation, hvis brugere ikke har trænet før klokken 15.
Dette er nødvendigt for at kunne motivere brugere til at udføre træning
- Systemet skal kunne give virtuelle belønninger
Dette er nødvendigt for at kunne motivere brugere til at udføre træning
- Brugere skal kunne følge andre brugere
Dette er nødvendigt for at skabe fællesskab samt gøre det muligt for brugere at tilgå hinandens virtuelle belønninger, hvilket skal øge brugeres motivation
- Brugere skal kunne redigere deres adgangskode
Dette er nødvendigt for, at brugere skal kunne gøre deres adgangskode personlig
- Brugere skal kunne logge ud af app'en
Dette er nødvendigt for at sikre brugeres individuelle data

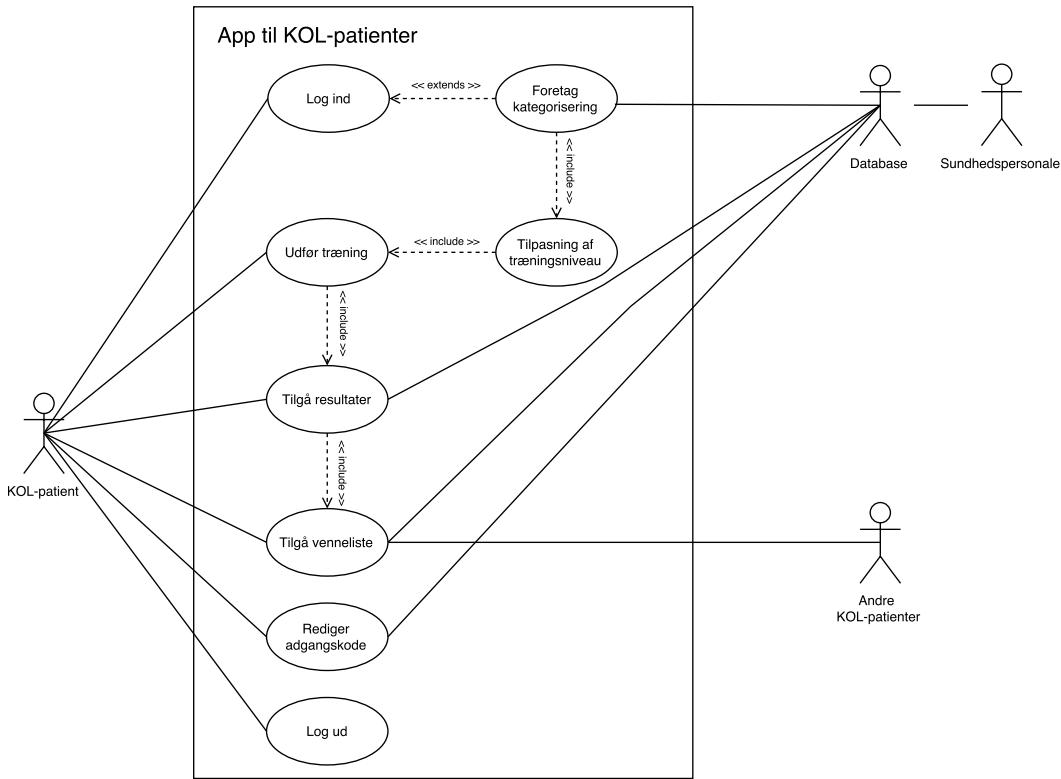
Non-funktionelle krav

- Systemet skal visualiseres på en smartphone eller tablet med android
- Systemet skal være brugervenligt
Dette er nødvendigt for at sikre let orientering i app'en

5.2.1 Use case

På baggrund af systembeskrivelsen samt opstillede krav er der udarbejdet et use case diagram, der beskriver app'ens funktioner. Af use case diagrammet på figur 5.1 ses systemet, app til

KOL-patienter, samt de forskellige use cases og aktører, der interagerer med systemet. KOL-patienten er den primære aktør, som kan tilgå alle use cases. Database og andre KOL-patienter er sekundære aktører og kan kun tilgå enkelte use cases. Sundhedspersonalet har kun adgang til data via en database.



Figur 5.1: Use case diagram for app til KOL-patienter

KOL-patienter skal *Log ind* i app'en via medlemsID og adgangskode. Første gang de logger ind skal de *Foretag kategorisering*. Denne kategorisering gemmes efterfølgende i en database. Hvis kategoriseringen er foretaget har brugere adgang til en hovedmenu, hvorfra de kan vælge at udføre træning, tilgå resultater, tilgå venneliste, redigere adgangskode og logge ud.

Vælger brugeren, at *Udføre træning* tilpasses træningsniveauet før selve træningen kan påbegyndes ud fra *Tilpasning af træningsniveau*, der vurderes på baggrund af kategoriseringen, daglig helbredstilstand samt evaluering af tidligere træninger.

Efter træningen skal brugere evaluere træningen, og de kan efterfølgende tilgå deres resultater. Evaluering og resultater fra træningen gemmes efterfølgende i en database. Brugere kan *Tilgå resultater*, der vises som virtuelle belønninger, som opnås på baggrund af udførte træninger.

Brugere kan via *Tilgå venneliste* tilgå andre KOL-patienters virtuelle belønninger, hvilket medvirker til, at brugere kan motivere hinanden til at udføre træning.

Brugere kan *Redigere adgangskode*, da det skal være muligt for brugeren at ændre adgangskoden til en personlig adgangskode. Hvis der foretages ændringer gemmes disse efterfølgende i databasen. Brugeren kan *Log ud* af app'en, hvis dette ønskes.

5.3 Funktionalitet

I dette afsnit beskrives funktionaliteterne, der er udarbejdet ud fra systembeskrivelsen samt use case diagrammet. De enkelte funktionaliteter er opdelt efter registrering, log ind, kategorisering af KOL, tilpasning af træningsniveau, træning, resultater, venneliste, redigering af adgangskode og log ud. Nogle af funktionaliteterne er beskrevet i aktivitetsdiagrammer, som er opdelt i bruger, systemet og database. Der er i nogle af aktivitetsdiagrammerne angivet et billesymbol, hvilket betyder at aktiviteten vil uddybes i et andet aktivitetsdiagram. Når et aktivitetsdiagram starter antages det, at brugeren har trykket på den gældende aktivitet. Efter hver endt aktivitet eller ved at trykke tilbage vises en hovedmenu.

Registrering

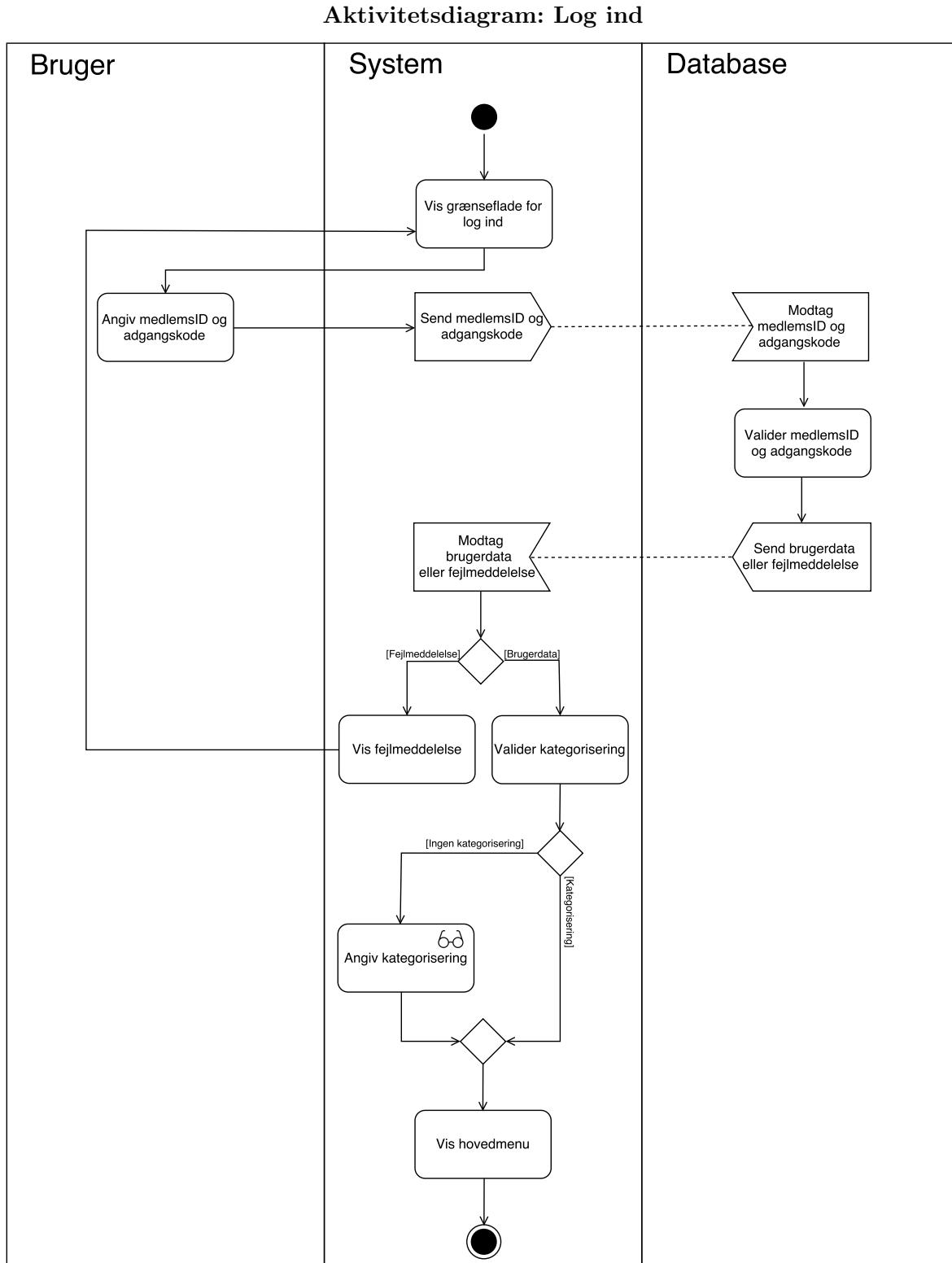
Inden KOL-patienter kan anvende app'en skal de registreres som brugere af systemet. Dette skal foregå i forbindelse med rehabiliteringsforløbet, hvor sundhedspersonalet opretter patienterne i databasen. Patienterne får tilknyttet et medlemsID og en randomiseret adgangskode. Dette er vigtigt for at reducere risikoen for misbrug af personlige oplysninger, da uvedkommende kan have mulighed for at tilgå informationerne via netadgang eller enheden [39]. MedlemsID'et skal bestå af tal, eksempelvis *11170301*, som er sammensat ud fra lokalisering, årstal og måned for påbegyndt rehabiliteringsforløb samt nummerering af den enkelte KOL-patient.

Under registrering oprettes KOL-patienter ligeledes med fornavn og efternavn, der skal gøre dem identificerbare, således andre brugere kan følge dem.

I forbindelse med registrering skal KOL-patienterne logge ind, hvortil sundhedspersonalet introducerer KOL-patienter til brugen af app'en. Herunder skal de hjælpe KOL-patienterne med at kategorisere patientens sygdom før app'en anvendes til træning i hjemmet. Dette skal gøres i et forsøg på at skabe tryghed hos patienterne, da denne kategorisering har betydning for, hvilket træningsniveau patienten senere får foreslået af app'en. Der er desuden mulighed for at kunne få besvaret eventuelle tvivlsspørgsmål, der kan opstå første gang app'en anvendes.

Log ind

I systemet benyttes en log ind-funktion til at beskytte og identificere den enkelte bruger. Brugeren skal her angive log ind-information, der vil tillade adgang til brugerinformation i form af private oplysninger og tidlige resultater, tilknyttet den givne bruger. Aktiviteterne for log ind fremgår af figur 5.2.



Figur 5.2: Aktivitetsdiagram for log ind. Kategorisering af KOL uddybes af figur 5.3

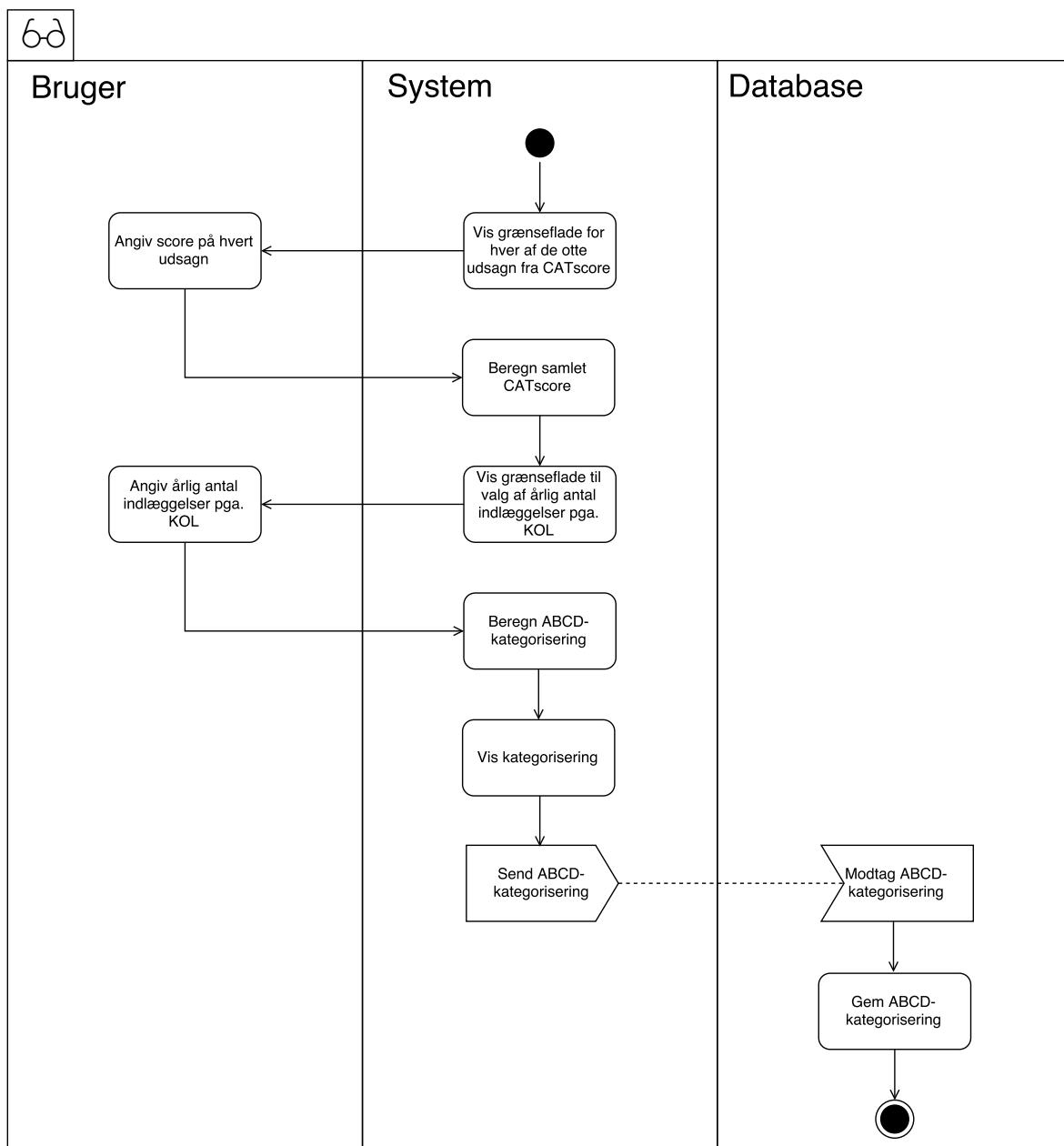
En grænseflade for log ind er det første der vises, når brugeren første gang åbner app'en. Brugeren kan i denne grænseflade angive sit medlemsID samt adgangskode, hvorefter systemet tilsender de indtastede log ind-informationer til databasen. Databasen validerer log ind-informationer og sender brugerdata eller en fejlmeldelse, hvis medlemsID'et eller adgangskoden ikke findes i databasen.

Sendes en fejlmeddeelse vises grænsefladen for log ind, hvor brugeren kan indtaste log ind-informationerne igen. Har brugeren glemt sin log ind-informationer, bedes de kontakte sundhedspersonalet. Findes log ind-informationerne i databasen sendes brugerdata, der omfatter brugeroplysninger, resultater samt venneliste. Når brugerdata er hentet valider systemet om brugeren har fået en kategorisering. Har brugeren ingen kategorisering, henvises brugeren til kategoriseringen, der fremgår af figur 5.3, ellers henvises brugeren videre til hovedmenuen.

Kategorisering af KOL-patienter

Første gang KOL-patienter logger ind i app'en skal de have en individuel kategorisering. Dette er nødvendigt for at sikre, at brugeren får anbefalet et træningsniveau, som passer til deres helbred. Kategoriseringen inddeler brugerne i A, B, C eller D, som beskrevet i afsnit 3.1.2. Af figur 5.3 ses aktivitetsdiagrammet for kategoriseringen.

Aktivitetsdiagram: Kategorisering af KOL-patienter



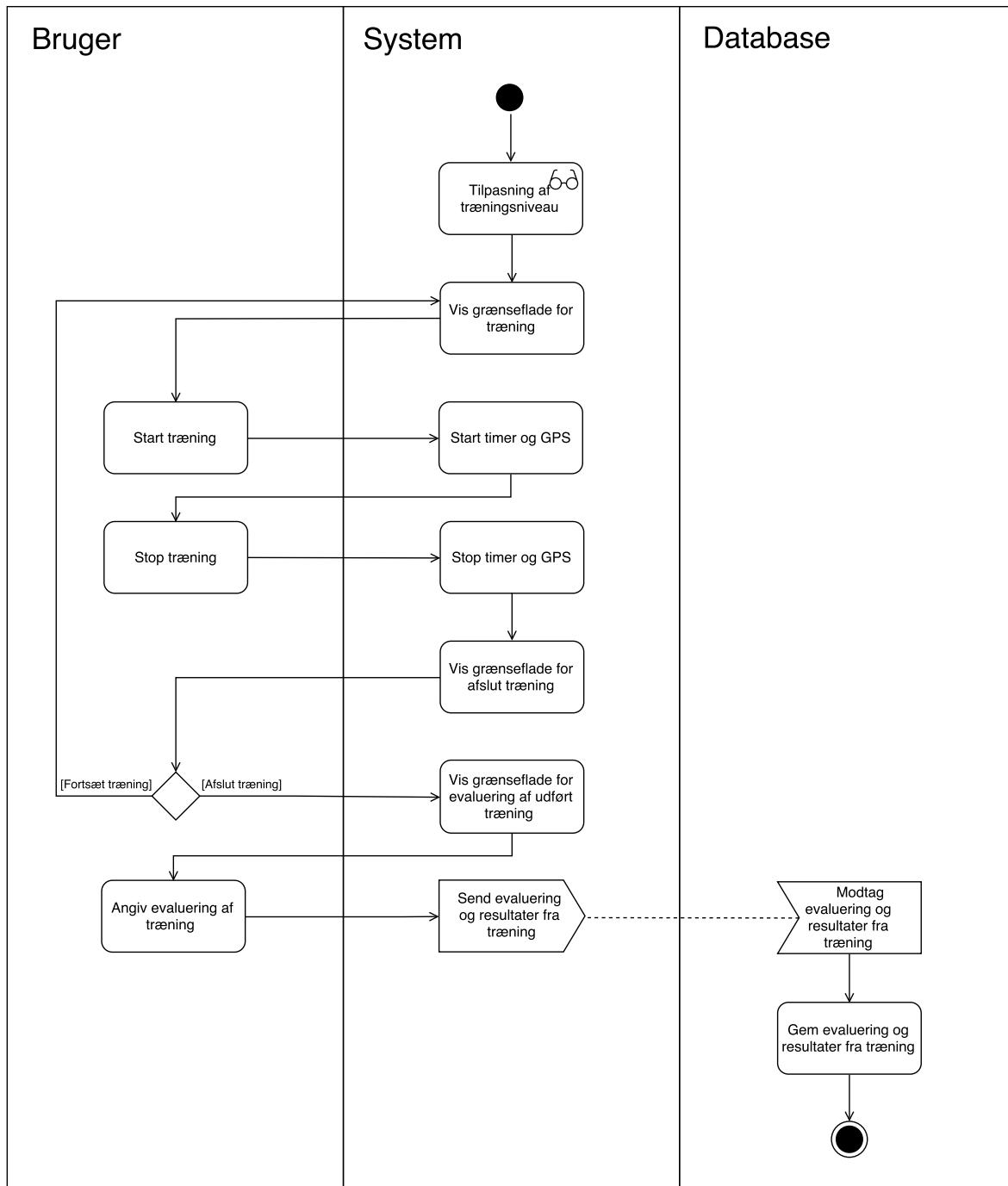
Figur 5.3: Aktivitetsdiagram for kategorisering af KOL-patienter.

Systemet starter med at vise en grænseflade for hver af de otte udsagn, der udgør CATscoren, jf. figur 3.2. Til hvert af udsagnene angiver brugeren en score passende til deres sygdomstilstand, hvor systemet ud fra de individuelle score beregner en samlet CAT-score. Dernæst vises grænsefladen for årlig antal indlæggelser på grund af KOL, hvor brugeren skal angive antal indlæggelser årligt på grund af KOL. Ud fra den samlede CAT-score og antal indlæggelser, beregner systemet brugerens kategorisering af KOL. Efterfølgende viser systemet brugerens kategorisering som A, B, C eller D. Kategoriseringen sendes og gemmes i en database.

Træning

Inden brugeren kan træne skal træningen tilpasses den enkelte bruger. Tilpasning af træningsniveauet er yderligere beskrevet af figur 5.5. Aktivitetsdiagrammet over træningen fremgår af figur 5.4.

Aktivitetsdiagram: Træning



Figur 5.4: Aktivitetsdiagram over træning. Tilpasning af træningsniveau uddybes af figur 5.5.

Når systemet har tilpasset træning vises grænsefladen for træning. Brugeren kan påbegynde træningen ved at trykke start, hvorefter timer og GPS starter. Under træningen vil tid og afstand kontinuert fremgå af grænsefladen. Brugeren kan til en hver tid vælge at afslutte

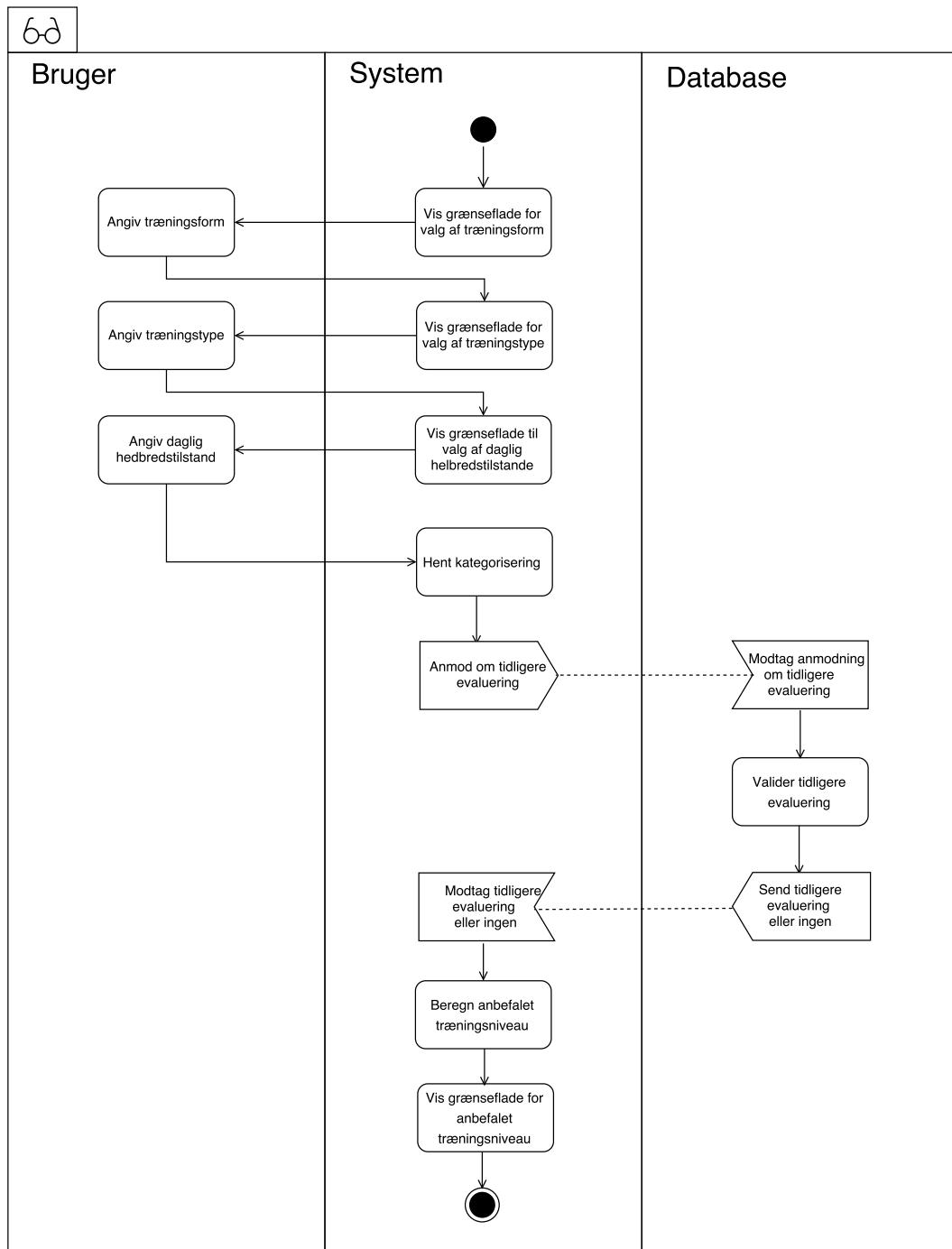
træningen ved at trykke på stop træningen, hvilket medfører, at timer og GPS stoppes og grænsefladen for afslut træning vises. Denne handling skal bekræftes i tilfælde af, at brugeren ved en fejl angiver, at træningen skal stoppes. Ved en fejl vises grænsefladen for træning. Ved bekræftelse af stop træning, vises grænsefladen for evaluering af udført træning, hvor brugeren skal angive en evaluering. Efterfølgende sendes evalueringen og træningsresultater til en database, hvor det gemmes.

Tilpasning af træningsniveau

Før selve træningen påbegyndes, skal træningsniveauet tilpasses brugeren.

Tilpasning af træningsniveau er en funktion der skal tage højde for daglige variationer ved at anbefale et træningsniveau ud fra brugeres kategorisering, daglige helbredstilstand og tidlige evalueringer af træninger. Aktivitetsdiagrammet over tilpasning af træningsniveau fremgår af figur 5.4.

Aktivitetsdiagram: Tilpasning af træning



Figur 5.5: Aktivitetsdiagram over tilpasning af træningsniveau.

Systemet viser grænsefladen for valg af træningsform, hvor brugeren skal angive den ønskede træningsform, herunder konditions-, styrketræning eller vejrtærkningsøvelser. Ud fra den valgte træningsform skal brugeren angive træningstype, eksempelvis kan der ved valg af konditionstræning vælges gå, løbe eller cykle. Herefter vises grænsefladen for valg af daglig helbredstilstande, hvortil brugeren skal angive sin helbredstilstand. Systemet henter kategorisering, hvorefter den anmelder om tidligere evaluering i databasen. Databasen tjekker om der er tidligere evalueringer for den valgte træningsform og -type. Hvis brugeren ikke har angivet tidligere evalueringer bestemmes niveauet ud fra de resterende parametre. Hvis der

findes tidligere evalueringer hentes disse og medregnes som en parameter. Et simpel eksempel på denne beregning fremgår af tabel 5.1. Tabellen beskriver, hvordan en algoritme vil kunne regulere træningsniveauet, således der tages højde for den enkelte bruger.

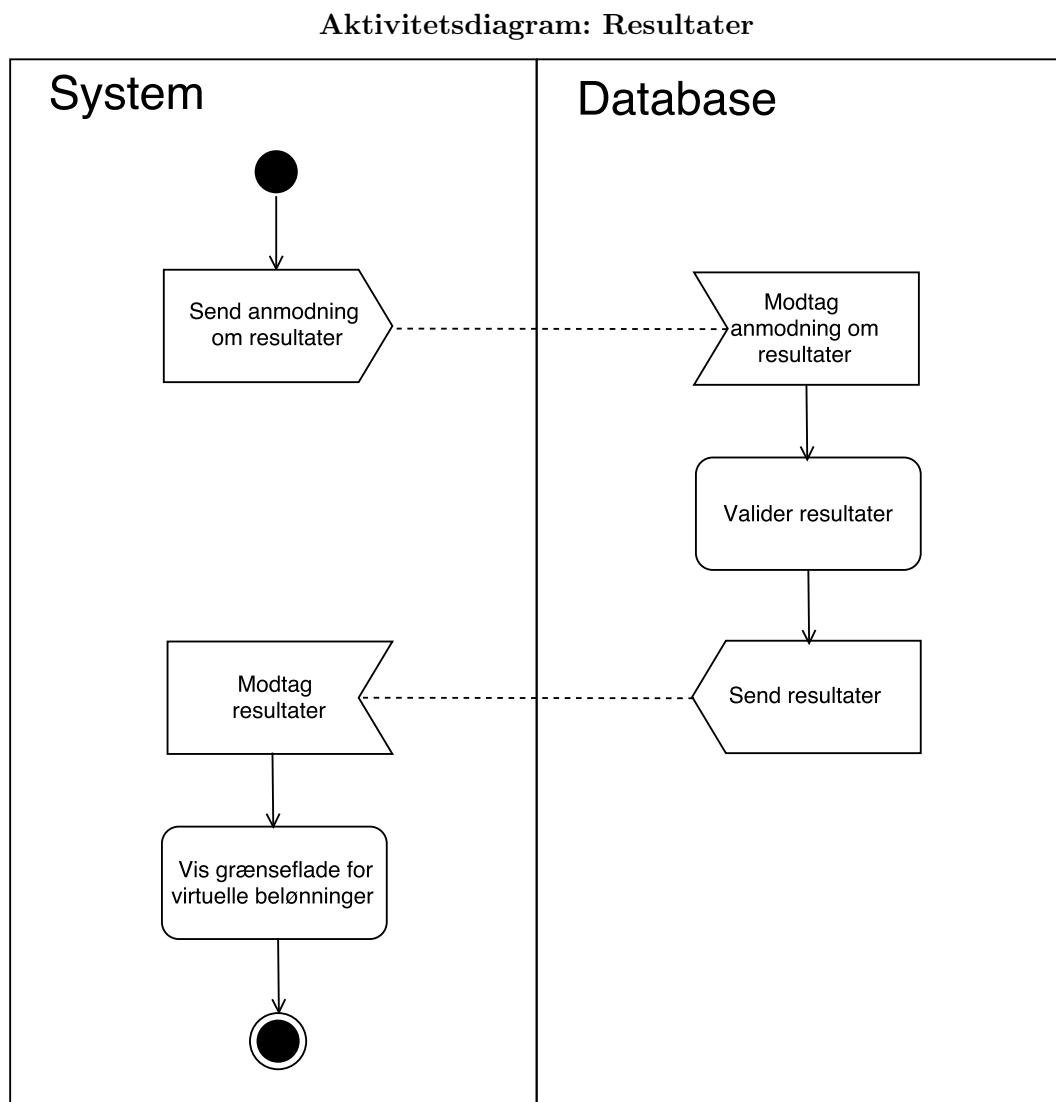
Simpel beslutningstabell									
Kategorisering	A		B			C			D
Træningsform	Konditionstræning			Styrketræning			Vejrtrækningssøvelser		
Træningstype	Gå	Løb	Cykel	Type 1	Type 2	Type 3	Type 1	Type 2	Type 3
Daglig Helbredstilstand	1: Meget dårligt		2: Dårligt	3: Moderat	4: Godt	5: Meget godt			
Træningsniveau									
Træningsform	Konditionstræning			Styrketræning			Vejrtrækningssøvelser		
Træningstype	Gå	Løb	Cykel	Type 1	Type 2	Type 3	Type 1	Type 2	Type 3
Evaluering	Let (+) Træningsniveau			Moderat			Hård (-) Træningsniveau		
Reguleret træningsniveau									

Tabel 5.1: Beslutningstabell for træningsniveau. Kategorisering, daglig helbredstilsand samt eventuel evaluering anvendes til at bestemme træningsniveauet til den enkelte bruger. Af dette eksempel er brugeren kategoriseret B, har valgt konditionstræning, herunder løb. Derudover har brugeren angivet sin helbredstilstand til moderat. Ud fra dette kan der tilpasses et træningsniveau. Hvis brugeren tidligere har evaluert en træningen inden for konditræning og løb anvendes denne evaluering til regulere træningsniveauet. I dette tilfælde har brugeren tidligere evaluert træningen til hård, hvorfor niveauet af træningen sænkes.

Af tabel 5.1 fremgår en simpel beslutningstabell for, hvorledes et træningsniveau tilpasses den enkelte bruger. Beslutningstabellen tager udgangspunkt i brugerens kategorisering, valgt træningsform og -type samt daglig helbredstilstand og en eventuel evaluering. Brugeren er i dette tilfælde kategoriseret til B. Valgt at udføre konditionstræning herunder løb. Helbredstilstanden angives førend en træning påbegyndes, for således at tilpasse niveauet til den pågældende dag. Helbredstilstanden angives efter 1: *Meget dårligt*, 2: *Dårligt*, 3: *Moderat*, 4: *Godt* eller 5: *Meget godt*, hvortil brugerens helbredstilstand her angives som moderat. Træningsniveauet vurderes dermed ud fra brugerens kategorisering, valg af træningsform og -type samt helbredstilstand. For at have mulighed for at kunne regulere træningsniveauet yderligere, medregnes den forhenværende evaluering, der er forbundet med samme helbredstilstand, træningsform og type. I dette tilfælde har brugeren før haft samme helbredstilstand, træningsform samt -type og dertil evalueres denne træning til værende hård. Algoritmen regulerer hertil træningsniveauet for denne træning ned, for således at give brugeren en bedre og mere tilpasset træningsoplevelse.

Resultater

Fra app'ens hovedmenu kan brugeren tilgå sine resultater, som visualiseres ved virtuelle belønninger. Aktivitetsdiagrammet over resultater fremgår af figur 5.6.



Figur 5.6: Aktivitetsdiagram over resultater.

Under resultater er det muligt for brugeren at se sine virtuelle belønninger. Belønningerne varierer afhængig af træningsform og der kan opnås forskellige belønninger inden for forskellige kategorier. Et eksempel på fordeling af belønninger i forskellige kategorier fremgår af tabel 5.2.

Belønninger						
	★	★★	★★★	★★★★	★★★★★	★★★★★★
Tid (min)	10	20	30	40	50	60
Afstand (km)	1	3	5	7	9	10
Træning (antal)	1	5	10	15	20	25
Konditionstræning (antal)	1	5	10	15	20	25
Styrketræning (antal)	1	5	10	15	20	25
Vejrtrækningsøvelse (antal)	1	5	10	15	20	25

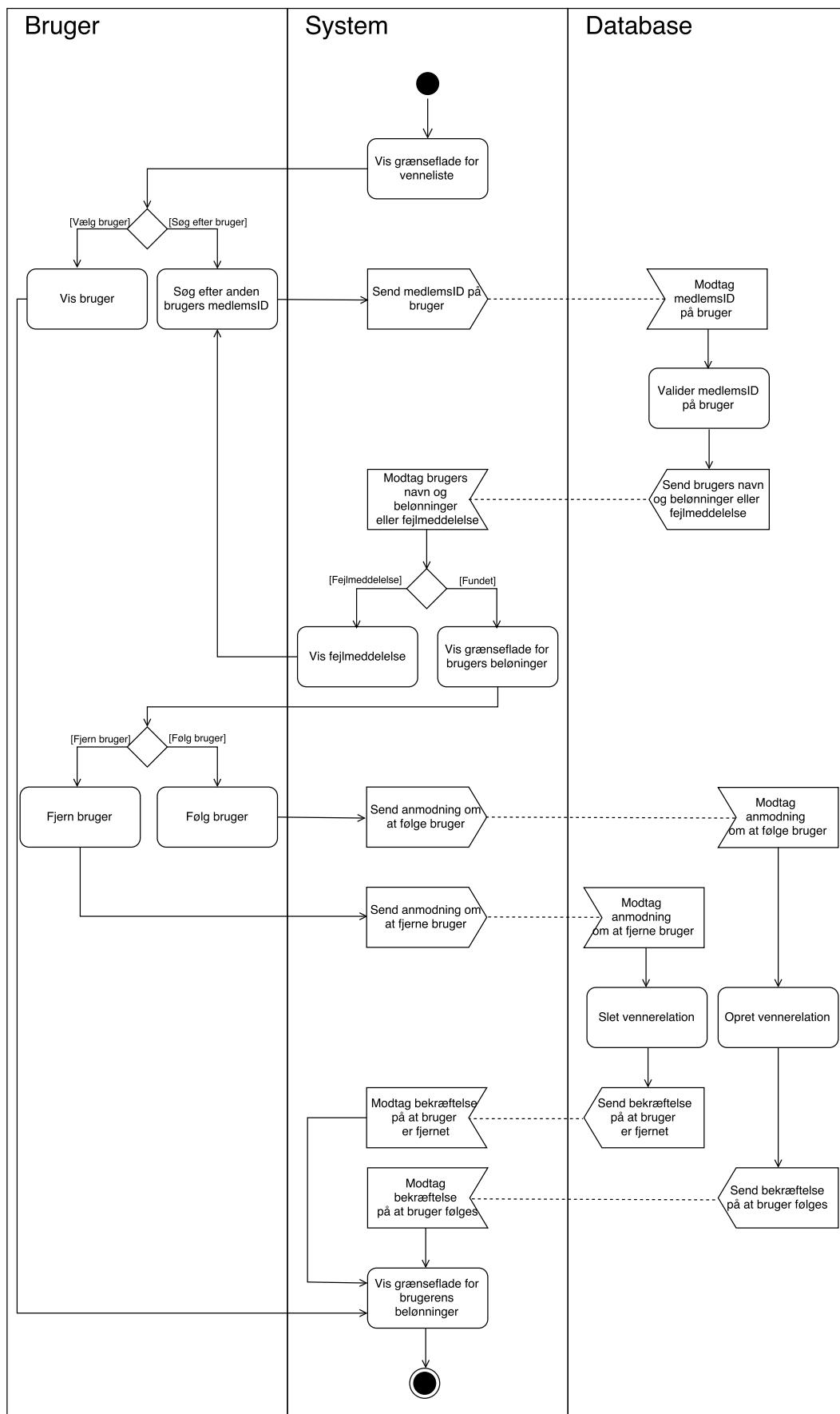
Tabel 5.2: Eksempel på belønninger opnået ved træning inden for forskellige kategorier.

Ud fra tabel 5.2 fremgår et eksempel på fordeling af virtuelle belønninger, der er opdelt efter afstand, tid og antallet af gennemførte træninger.

Venneliste

For at motivere brugere til regelmæssig træning, vælges at integrere muligheden for sociale relationer i app'en. Dette muliggør, at brugere kan følge hinanden og derved se, hvilke belønninger andre brugere har opnået. Hertil skal det være muligt for brugeren at tilføje nye brugere til vennelisten. Af figur 5.7 fremgår et aktivitetsdiagram for vennelisten.

Aktivitetsdiagram: Venneliste



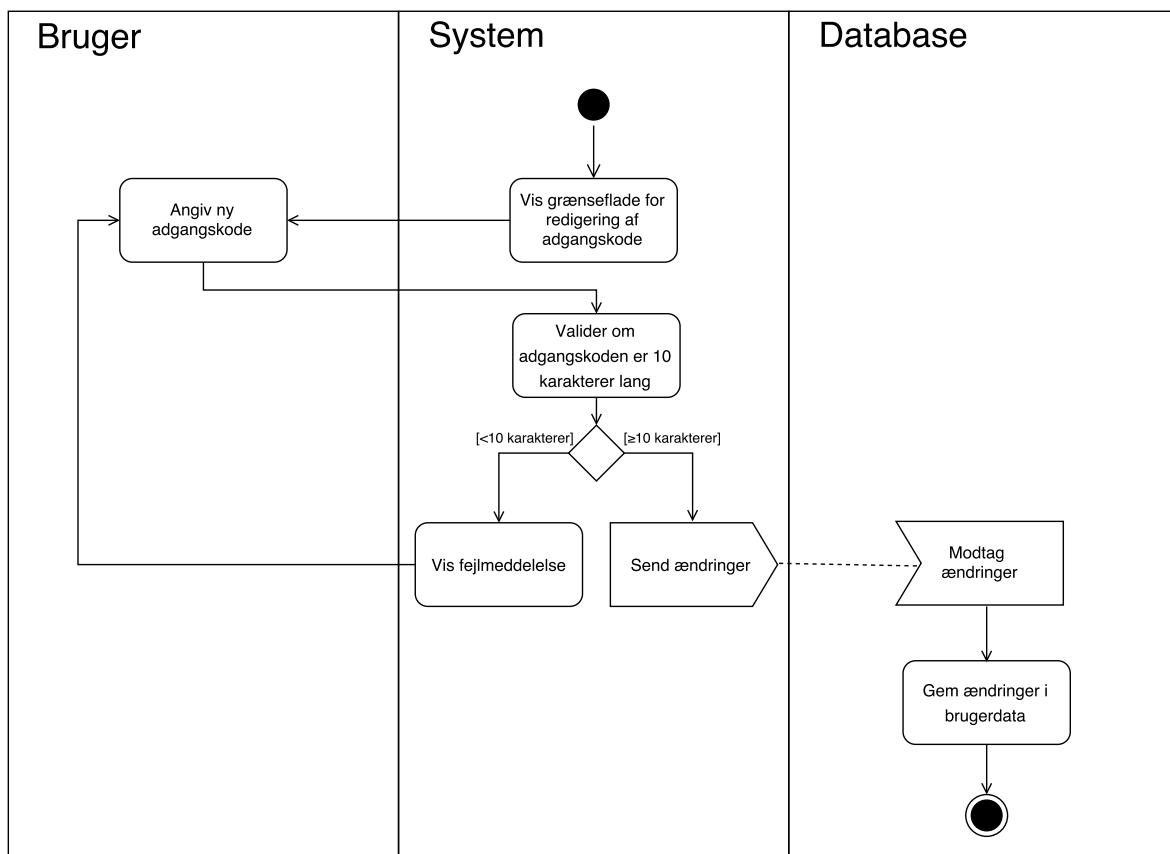
Figur 5.7: Aktivitetsdiagram for venneliste.

Systemet viser en grænseflade for vennelisten, der indeholder en oversigt over andre brugere, som den individuelle bruger følger. Brugeren kan vælge at søge efter en ny bruger eller vælge en bruger fra vennelisten. Ønsker brugeren at søge efter en ny, skal brugeren angive medlemsID på denne. Systemet sender det indtastede medlemsID til databasen, som validerer om brugeren findes i databasen. Findes brugeren ikke i databasen, sendes en fejlmeddeelse og brugeren har ingen mulighed for at angive medlemsID på en anden bruger. Er brugeren i databasen, vises grænsefladen for brugeren, hvor det er muligt at følge brugeren, hvis den ikke allerede følges samt fjerne brugere, hvis det ønskes ikke at følge brugeren mere. Ønskes det at følge brugeren, sender systemet en anmodning om at følge brugeren. Databasen opretter vennnerelation, hvorefter systemet sender en bekræftelse. Ligeledes sender systemet en anmodning om at fjerne brugeren, hvis dette ønskes, hvorefter databasen sletter vennnerelationen og systemet sender en beskæftigelse. Efter bekræftelse eller ved at vælge en ven fra vennelisten viser systemet grænsefladen for den valgte, tilføjede eller fjernede brugers belønninger.

Redigering af adgangskode

Ud fra app'ens hovedmenu har brugeren mulighed for at tilgå og få vist brugeroplysninger samt redigere sin adgangskode. Af figur 5.8 illustreres aktivitetsdiagrammet for redigering af adgangskode.

Aktivitetsdiagram: Redigering af adgangskode



Figur 5.8: Aktivitetsdiagram for redigering af adgangskode.

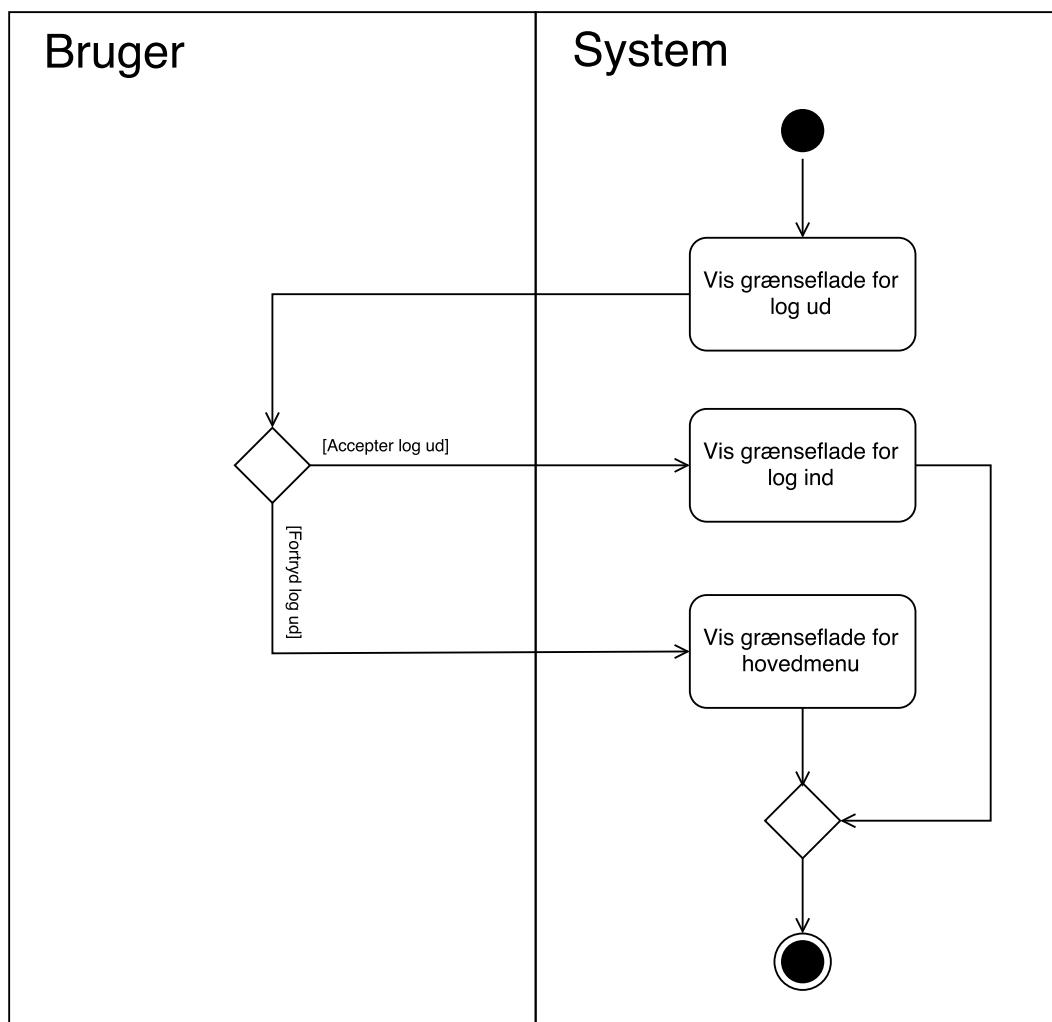
Det skal være muligt for brugeren at ændre adgangskode, da brugeren ved oprettelse får tildelt en randomiseret adgangskode. Dertil kan adgangskoden blive personlig for brugeren, hvilket

vil gøre det nemmere for brugeren at huske. Brugeren kan ændre adgangskoden ved at vælge rediger adgangskode fra hovedmenuen, hvorefter grænsefladen for redigering af adgangskode vises. For at den nye adgangskode kan benyttes, skal den minimum være 10 karakterer lang. Grunden til dette er, at der ved log ind sendes en fejlmeddelelse, hvis indtastede informationer ikke findes i databasen, dertil skal adgangskoden ikke kunne forveksles med fejlmeddelelsen. Desuden anbefaler Rådet for Digital Sikkerhed, at adgangskoder bør være minimum 10 karakterer lang, dog er dette ikke et krav [40]. Hvis kravet om minimum 10 karakterer ikke opfyldes, sendes en fejlmeddelelse tilbage til brugeren, hvortil en ny adgangskode kan indtastes. Ændres adgangskoden sendes ændringen til databasen, hvorefter den gemmes i databasen.

Log ud

Log ud-funktion kan tilgås fra hovedmenuen og tillader brugeren at logge ud af systemet. Dette medvirker til, at brugeren ikke forbliver logget ind og tillader at andre kan logge ind på samme app. Aktiviteterne for log ud fremgår af figur 5.9.

Aktivitetsdiagram: Log ud



Figur 5.9: Aktivitetsdiagram over log ud.

Vælger brugen at logge ud, vises grænsefladen for log ud. Brugeren skal efterfølgende bekræfte, at de ønsker at logge ud. Ønskes dette, logges brugeren ud og grænsefladen for log ind vises.

Fortryder brugeren at logge ud, vil systemet returnere til hovedmenuen og grænsefladen for denne vises.

5.4 Analyseklasser

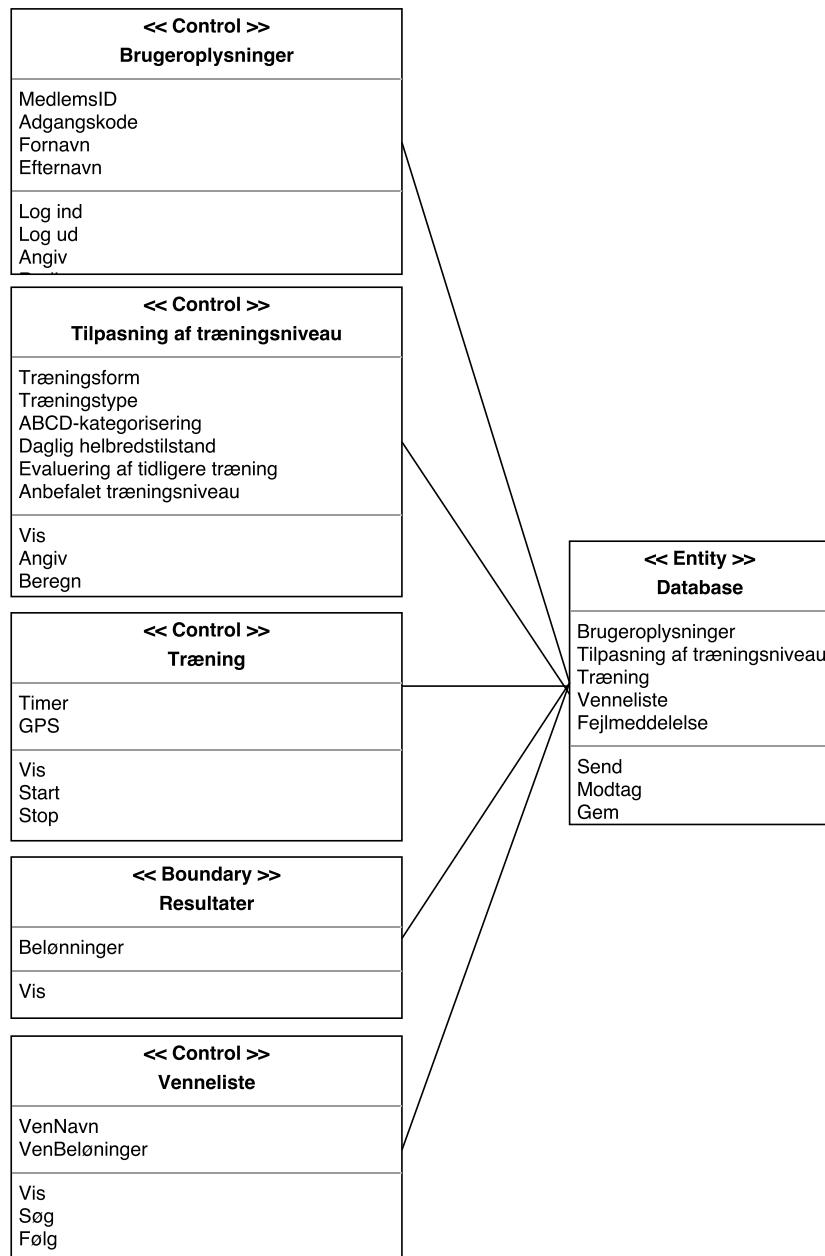
Inden analyseklasserne udarbejdes, foretages en analyse ud fra systembeskrivelsen, use case og funktionaliteter til at identificere substantiver og verber. Dette gøres for at sikre, at alle funktionaliteter indgår i design af klassediagrammer. Substantiver og verber fra analysen fremgår af tabel 5.3.

Substantiver	Verber
Brugeroplysninger MedlemsID Adgangskode Fornavn Efternavn Tilpasning af træningsniveau Træningsform Træningstype ABCD-kategorisering Daglig helbredstilstand Evaluering af tidligere træning Anbefalet træningsniveau Træning Timer GPS Resultater Belønninger Venneliste VenMedlemsID VenNavn VenBeløninger Database Fejlmeddeelse	Log ind Log ud Valider Angiv Beregn Vis Start Stop Søg Følg Send Modtag Gem

Tabel 5.3: Substantiver og verber identificeret ved analyse af systembeskrivelse, use case samt funktionaliteter.

De fremhævede substantiver, brugeroplysninger, træning, resultater, venneliste og database, identificeres som klasser. Under hver klasse fremgår deres tilhørende attributter, der beskriver den overordnede klasse. Verberne betegner de metoder, der kan tilgås i de forskellige klasser.

Efter substantiver og verber er identificeret inddeltes disse i analyseklasser og opdeles i stereotyperne, entity, boundary og control. Dette kan ses af figur 5.10.



Figur 5.10: Analyseklasser udarbejdet ud fra de identificerede substantiver og verber.

Af figur 5.10 fremgår relationen mellem klasserne og deres tilhørende attributter samt metoder. *Database* er defineret som entityklasse, da den skal lagre og opdatere informationer. *Resultater* er defineret som boundaryklasse, da den skal vise resultater på en grænseflade. *Brugeroplysninger*, *Tilpasning af træningsniveau*, *Træning* og *Venneliste* defineres som controlklasser, idet disse anvendes til at kontrollere handlinger.

Kapitel 6

Systemdesign

I dette kapitel beskrives design af app'en samt design af database. I design af app'en tages der udgangspunkt i de analyseklasser, der blev identificeret i kapitel 5, hvoraf disse omdannes til designklasser. Herefter visualiseres sammenspillet mellem de forskellige designklasser i sekvensdiagrammer. Design af database, der vil fremgå til sidst af kapitlet, vil visualiseres i et ER-diagram samt tilhørende schema.

6.1 Designafgrænsning

Systemet designes med henblik på at opfylde kravsspecifikationerne, der blev udarbejdet i systemanalysen jf. afsnit 5.2. I forbindelse med de opstillede funktionelle krav foretages der afgrænsninger for at gøre design og efterfølgende implementering af systemet mere konkret. Der afgrænses til konditionstræning, da app'en formål ikke er at udforme et træningssæt til KOL-patienter. Dog forventes det, at implementeringen af styrketræning og vejrtrækningsøvelser er lignende.

Der er opstillet et non-funktionelt krav om et brugervenligt system. For at opfylde dette krav, tages der udgangspunkt i gestaltlovene samt generelle egenskaber indenfor design af brugergrænseflader, der kan have indflydelse på brugervenligheden.

Gestalt principperne bygger på en række love, der er opstillet af forskellige psykologer og anvendes til at designe visuelle elementer med henblik på forbedring af læring eller effektivisering af visuelle resultater. Der findes mange gestalt love. De mest anvendte til design af grænseflader er; symmetri, regelmæssighed, lukkethed, prægnans, fokuspunkt, ensartethed, nærhed, lighed og harmoni.[41] Udover gestaltlovene har følgende egenskaber betydning for brugervenligheden [42]:

- Systemets funktioner skal være lette at lære at anvende for uerfarne brugere.
- Systemet skal være effektivt at anvende i forhold til tiden, det tager at fuldføre en opgave.
- Det skal være let at huske, hvordan systemet anvendes efter længere perioder uden brug af systemet.
- Fejlraten ved brug af systemet skal være så lav som muligt.
- Systemet skal være tilfredsstillende for brugeren at anvende.

6.2 Objektorienteret design

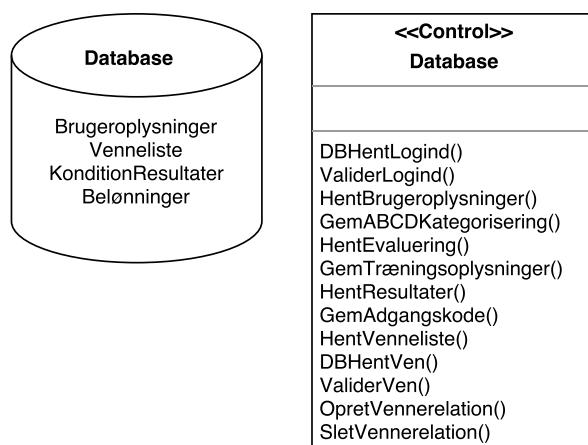
Der vælges at opdele analyseklasserne, som tidligere er defineret, til mindre designklasser. Dette gøres for at specificere de definerede attributter og metoder, hvilket gør det lettere at modificere. Ud fra de opstillede designklasser udarbejdes sekvensdiagrammer. I sekvensdiagrammerne er systemets klasser opdelt efter MVC. Objektet og livslinjer farvekodes

afhængig af klassens MVC-type. Model-klasser er røde, view-klasser er blå, og controller-klasser er grønne. I sekvensdiagrammer findes desuden også database, som er illustreret med henholdsvis lilla. I sekvensdiagrammerne findes metoder, der henter data fra en klasse til en anden. Idet det ikke er en egentlig metode, der sender data, illustreres overførslen med en stiplet pil, når data sendes til en klasse efter en hent-metode er kaldt.

Der er udarbejdet et samlet diagram over designklasser og relationerne i mellem, hvilket vil fremgå af afsnit A.1. De enkelte designklasser og sekvensdiagrammer er opdelt i database, lagring af data, log ind, kategorisering, hovedmenu, tilpasning af træningsniveau, træning, resultater, venneliste, redigering af adgangskode og log ud.

Database

Systemet skal som nævnt i kravsspecifikationer have forbindelse til en database. Denne kan tilgås fra den tilhørende controller. Klasserne for databasen og controlleren fremgår af figur 6.1.



Figur 6.1: Designklasse for Database. Til venstre ses databasen og til højre ses den tilhørende controller.

Databasen indeholder Brugeroplysninger, Venneliste, KonditionResultater og Belønninger. Brugeroplysninger indeholder alle brugerens oplysninger. Disse registreres i forbindelse med rehabiliteringsforløbet af sundhedspersonalet, som senere kan tilgå disse og redigere i brugeroplysningerne. Vennelisten indeholder andre bruger som brugeren følger, derudover har brugeren mulighed for at tilføje flere brugere til sin venneliste. KonditionResultater indeholder brugerens resultater for konditionstræning. Sundhedspersonalet har mulighed for at tilgå alle resultater. Belønninger indeholder de belønninger som brugeren har opnået i forbindelse med træning. Brugeren og de brugere der følger denne har adgang til disse.

Designet af databasen er yderligere beskrevet i et ER-diagram, der vil fremgå af afsnit 6.3. *Database* controlleren indeholder metoderne Hent, Valider, Gem, Opret og Slet. Disse har til formål at kommunikere mellem databasen og de forskellige controllerne. Der oprettes forbindelse til databasen ved log ind, træning, resultater, venneliste samt redigering.

Lagring af data

Når der oprettes forbindelse til databasen i forbindelse med, at brugeren logger ind eller tilgår resultater på app'en, sendes og gemmes data i forskellige entitys, som fremgår af figur 6.2. Der er valgt at udarbejde entitys for ikke at skulle tilgå databasen hver gang data skal hentes.

Dertil er det også muligt at opdele data, som har med en specifik controller at gøre, så irrelevant data undgås.

<<Entity>> Brugeroplysninger	<<Entity>> KonditionResultater
#MedlemsID #Adgangskode #Fornavn #Efternavn #Kategorisering	#Dato/tid #TræningsType #TræningsForm #DagligHelbredstilstand #TidligereEvaluering #Tid #Afstand #Belønninger

Figur 6.2: Designklasser for Entitys, herunder Brugeroplysninger og Resultater. Attributterne er markeret med #, hvilket indikerer, at disse er beskyttede.

På figur 6.2 fremgår det, at de forskellige attributter er beskyttede. Dette er gjort, da det ønskes at attributterne kun kan tilgås indenfor det samme projekt. *Brugeroplysninger* indeholder informationer om brugeren, herunder MedlemsID, Adgangskode, Fornavn, Efternavn og Kategorisering. *Resultater* indeholder brugers resultater, som er defineret ud fra DatoTid, TræningsType, TræningsForm, DagligHelbredstilstand, TidligereEvaluering, Tid, Afstand og Belønninger.

Fejlmeddeelse

Fejlmeddeelser vil fremgå som en metode, der anvendes i flere af de opstillede controllere, herunder log ind, venneliste og redigering af adgangskode. Disse anvendes til at oplyse brugeren om, hvis for eksempel indtaste værdier ikke stemmer overens med oplyste værdier i databasen eller ikke eksisterer i databasen. Til fejlmeddelelser anvendes metoden makeText.

Log ind

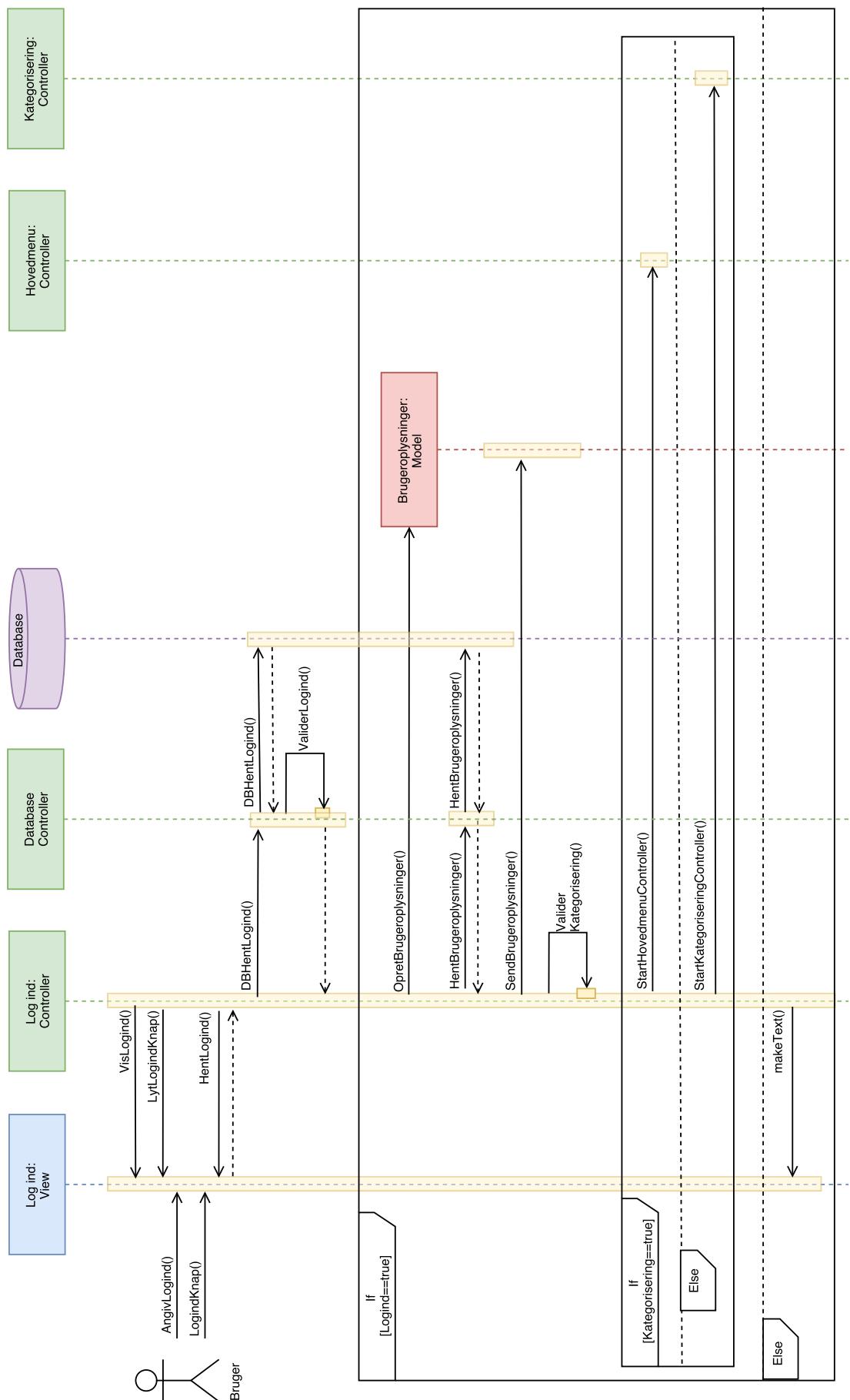
Log ind-funktionen inddeltes i en klasse af typen boundary samt en tilhørende controller. Disse fremgår af figur 6.3.

<<Boundary>> Log ind	<<Control>> Log ind
medlemsIDTekstfelt:EditText adgangskodeTekstfelt:EditText loginKnap:Button	medlemsIDInput:int adgangskodeInput:String VisLogind() LytLoginKnap() HentLogind() DBHentLogind() OpretBrugeroplysninger() HentBrugeroplysninger() SendBrugeroplysninger() ValiderKategorisering() StartHovedmenuController() StartKategoriseringController() makeText()

Figur 6.3: Designklasser for log ind. Til venstre ses grænsefladen og til højre den tilhørende controller.

I grænsefladen for *Logind* opstilles tekstmærker af typen EditText, hvor brugeren kan angive medlemsID samt adgangskode. Dertil opsættes en LogindKnap, af typen Button, der ved tryk indikerer, at brugerens informationer er angivet og klar til at logge ind.

Til denne grænseflade er der opstillet en *Logind* controller. Denne controller har metoderne Vis, Lyt, Hent, Opret, Send, Valider, Start og makeText. Metoden makeText anvendes til at fejlmeddelelser. Controlleren validerer log ind og opretter en entity, hvori oplysninger senere kan lagres. Denne entity beskrives af afsnit 6.2. Ligeledes håndterer controlleren, hvilken grænseflade systemet efterfølgende skal henvises til. Der er i sammenspil med disse designklasser udarbejdet et sekvensdiagram, der fremgår af figur 6.4.

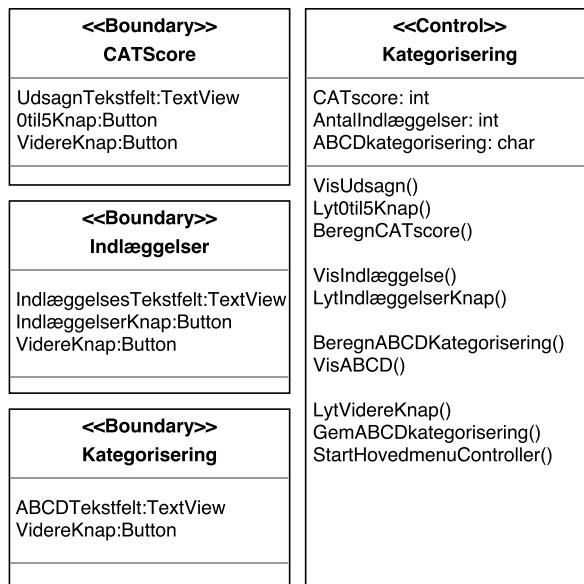


Figur 6.4: Sekvensdiagram for log ind.

Det fremgår af sekvensdiagrammet, at controlleren for *Log ind* starter med at vise grænsefladen for *Log ind*. Dertil lytter controlleren på LogindKnap, der ved tryk henter brugerens angivne log ind-informationer. Controlleren henter ligeledes log ind-informationer passende og validerer disse i controlleren for *Database*. Bekræftes log ind oprettes en entity af *Brugeroplysninger*. Controlleren for *Logind* henter oplysninger i databasen og sender disse til den oprettede entity, der lagrer brugeroplysninger fra databasen. Forefindes en kategorisering i de hentede brugeroplysninger startes controlleren for *Hovedmenu*. Eksistere kategoriseringen ikke startes *Kategorisering* controlleren. Mislykkes log ind vises en fejlmeddeelse, hvorved brugeren igen har mulighed for at indtaste log ind-oplysninger.

Kategorisering

Kategorisering inddeltes i tre boundarys og en dertilhørende controlklasse, som det fremgår af figur 6.5.



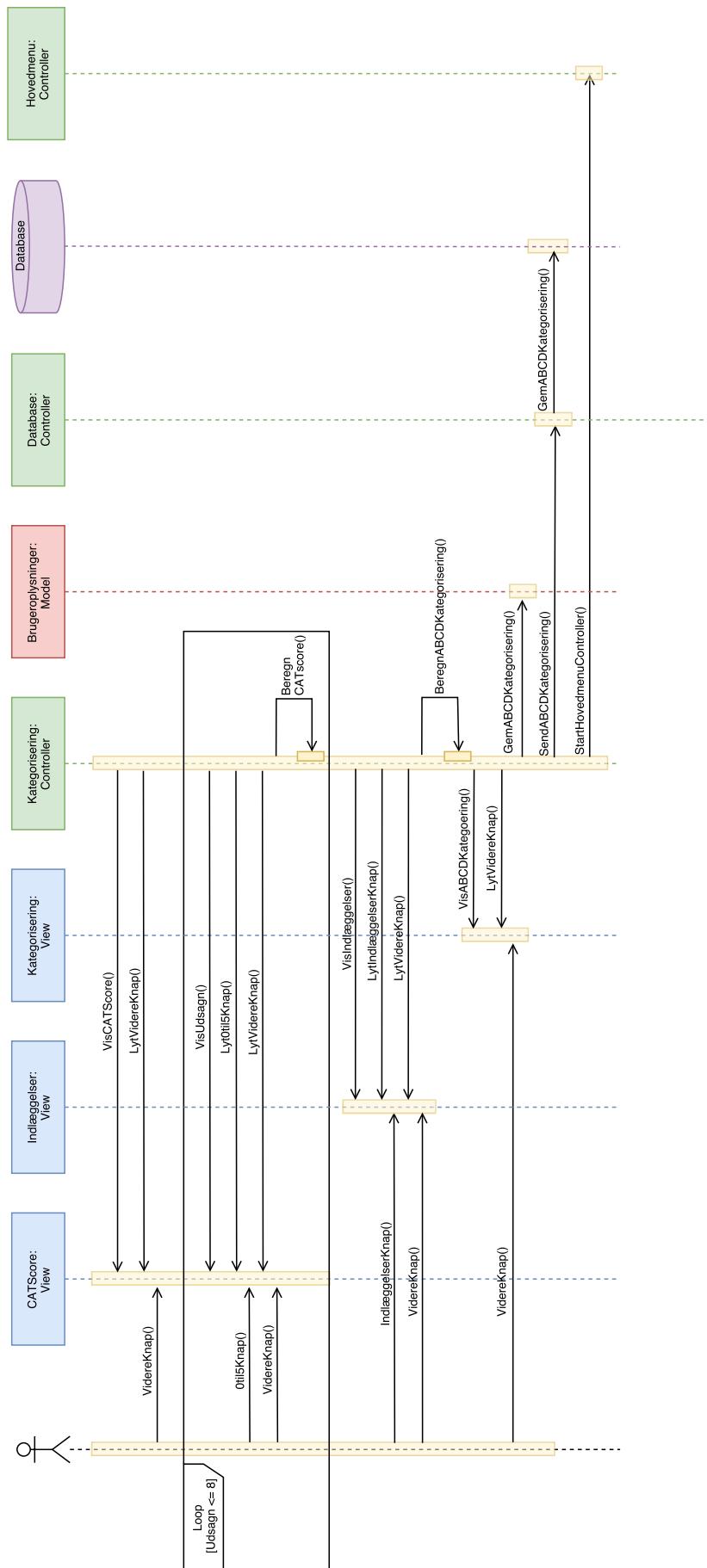
Figur 6.5: Designklasser for kategorisering. Til venstre ses de forskellige grænseflader, henholdsvis *CATScore*, *Indlæggelser* og *Kategorisering*. Til højre fremgår den dertilhørende controller.

Kategoriseringen inddeltes i tre grænseflader, herunder *CATScore*, *Indlæggelser* samt *Kategorisering*. Dette er valgt, idet grænsefladerne skal have forskellige layouts, og en opdeling vil gøre implementeringen af grænsefladerne mere overskuelig. Desuden ønskes det at gøre app'en overskuelig og brugervenlig, hvilket kan opnås ved at tydeliggøre adskillelsen mellem *CATscore* og besvarelse af antal årlige indlæggelser, j.f. gestaltloven om lighed i afsnit 6.1. Hertil skal brugeren foretage minimale valg på hver grænseflade, således brugeren ikke eksponeres til for mange valg på samme tid samt informationer på en grænseflade.

De tre grænseflader indeholder tekstfelter af typen *TextView*, der informerer brugeren om den følgende handling. Dertil er der ligeledes opsat knapper af typen *Button*, således brugeren kan besvare udsagn. I boundaryklassen *CATScore* ses *0til5Knap*, som repræsenterer seks forskellige knapper, der skal implementeres i grænsefladen. Disse seks knapper vil gøre det muligt for brugeren at vælge en værdi mellem nul og fem for hvert udsagn. Ligeledes ses der i boundaryklassen *Indlæggelser* en knap, *IndlæggelserKnap*, som repræsenterer to knapper

i grænsefladen. Brugeren skal på denne grænseflade angive antallet af årlige indlæggelser på grund af KOL, hvor knapperne gør det muligt at vælge mellem nul og en eller flere indlæggelser.

Den dertilhørende controller, *Kategorisering*, håndterer visningen af de tre boundarys samt respektive knapper og tekst. Controlleren indeholder metoderne Vis, Lyt, Beregn, Gem og Start. Der er ligeledes udarbejdet et sekvensdiagram for kategorisering, hvilket ses af figur 6.6.

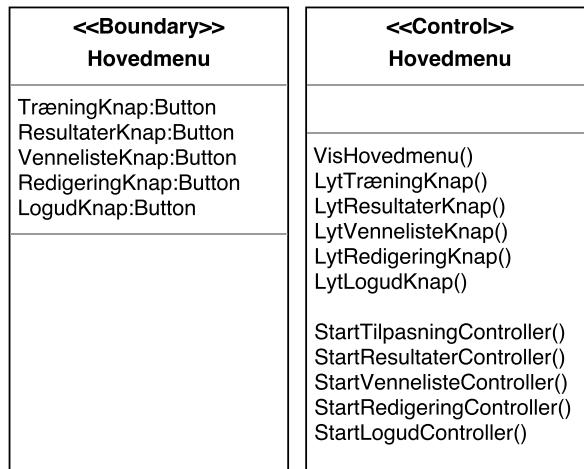


Figur 6.6: Sekvensdiagram for kategorisering.

Det fremgår af sekvensdiagrammet, at grænsefladen for *CATScore* vises som det første. Brugeren introduceres her til CATscore, hvorefter brugeren har mulighed for at trykke på VidereKnap. Herefter er der opstillet en loop, som har til formål at stille otte spørgsmål, som brugeren skal besvare ved hjælp af knapper, *0til5Knap*. Idet brugeren trykker videre fra hvert spørgsmål, adderes CATscoren. Efter de otte udsagn er besvaret, og den samlede CATscore er beregnet, vises grænsefladen for *Indlæggelser*. Controlleren lytter dertil til *IndlæggelserKnap*, hvori brugeren besvarer antal årlige indlæggelser forårsaget af KOL. Besvarelserne bekræftes ved at benytte *VidereKnappen*. ABCD-kategoriseringen kan derved beregnes og vises i *Kategorisering* grænsefladen. Denne kategorisering gemmes i entityen *Brugeroplysninger* og databasen, når brugeren trykker videre. Herefter startes hovedmenuen.

Hovedmenu

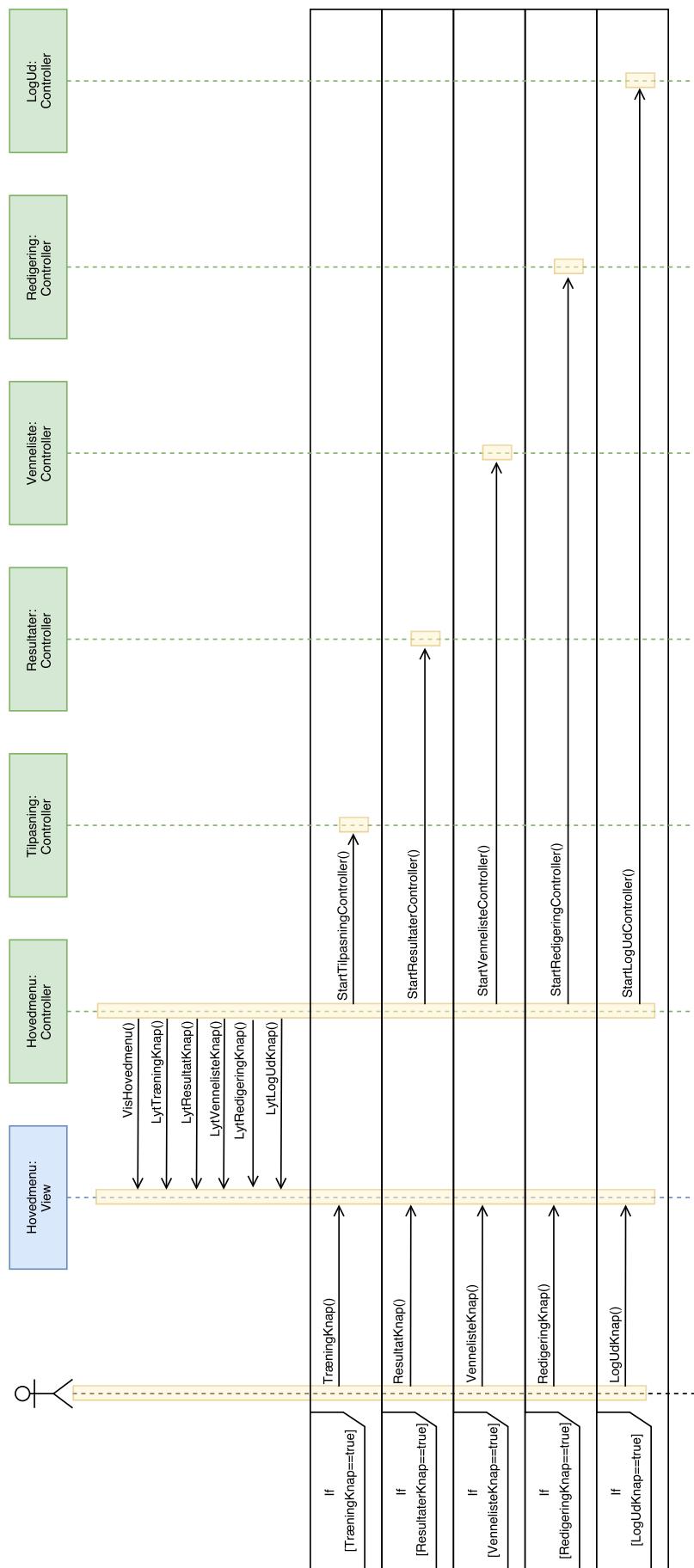
Hovedmenuen er den primære grænseflade, hvor andre grænseflader kan tilgås fra via deres controller. Hovedmenuen inddeltes i en boundary samt en dertilhørende controller. Disse fremgår af figur 6.7.



Figur 6.7: Designklasser for hovedmenu. Til venstre ses grænsefladen og til højre tilhørende controller.

Grænsefladen for *Hovedmenu* skal tillade adgang til app'ens forskellige funktionaliteter, herunder skal det være muligt at tilgå træning, resultater, venneliste, redigering samt log ud. Funktionaliteterne tilgås via tilhørende knapper af typen Button.

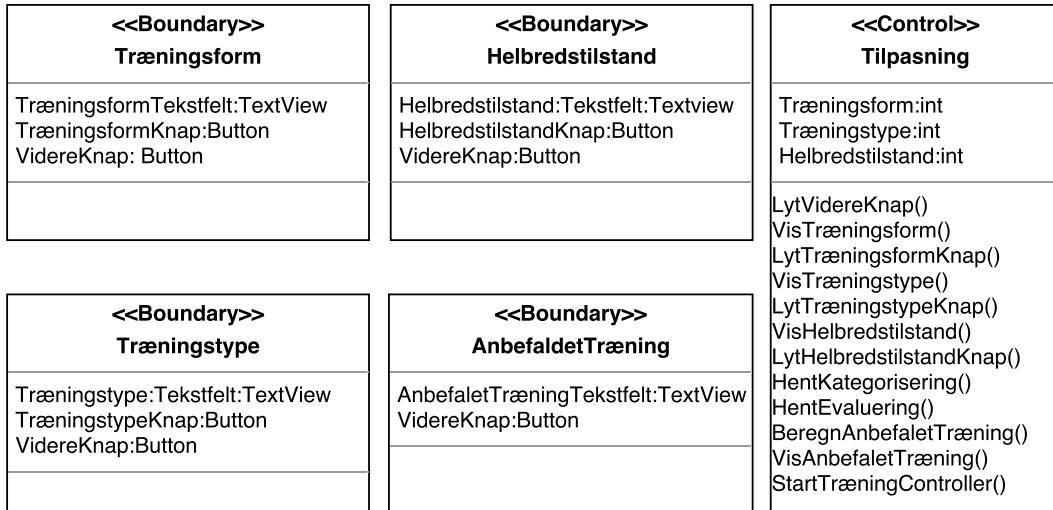
Controlleren indeholder metoderne Vis, Lyt og Start. Den viser grænsefladen for layoutet for hovedmenuen og lytter til de forskellige knapper. Et sekvensdiagram hertil er udarbejdet, hvilket fremgår af figur 6.8.

*Figur 6.8: Sekvensdiagram for hovedmenu.*

Det fremgår af sekvensdiagrammet, at der er opstillet forskellige if-loops, som viser sekvensen for hver sin knap. Når brugeren angiver sin ønskede funktion, henvises systemet til den dertilhørende controller. Hvis ingen af de opsatte knapper angives, forbliver brugeren på grænsefladen for *Hovedmenu*.

Tilpasning af træningsniveau

Før end træningen kan påbegyndes, tilpasses træningsniveauet den enkelte bruger. Tilpasning af træningsniveauet er inddelt i fire boundarys. Disse håndteres af en samlet controller, hvilket fremgår af figur 6.9.

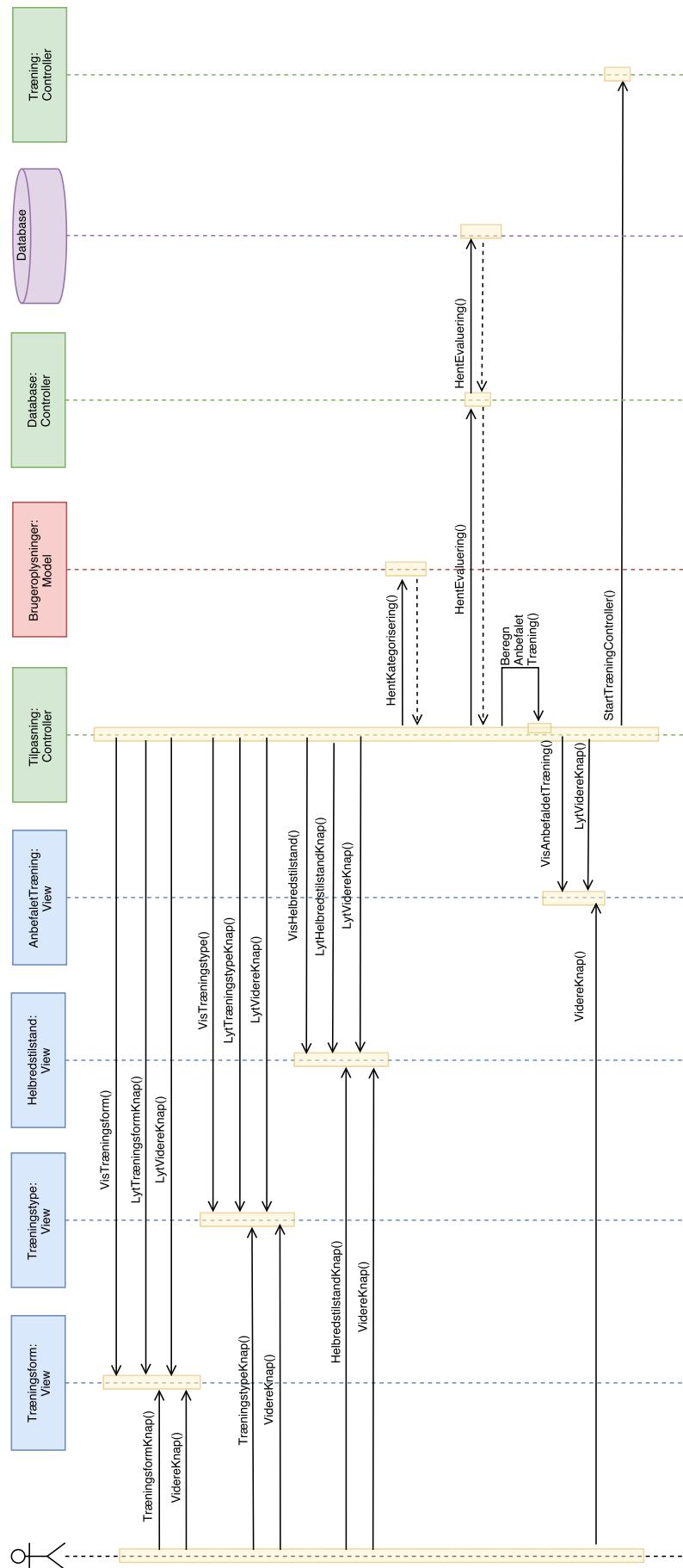


Figur 6.9: Designklasser for tilpasning af træningsniveau. Til venstre ses de fire boundarys for henholdsvis Træningsform, Træningstype, Helbredstilstand og AnbefaletTræning. Til højre fremgår den tilhørende controller.

Der er opstillet grænseflader for tilpasning af træning, hvilket omfatter *Træningsform*, *Træningstype*, *Helbredstilstand* og *AnbefaletTræning*. Tilpasningen af træningsniveau er delt for således at gøre app'en overskuelig samt brugervenlig. Der er til hver grænseflade opstillet tekstmærker af typen TextView og knapper af typen Button.

Den tilhørende controller til de ovenstående boundarys lagrer den angivne træningsform, træningstype samt helbredstilstand, således disse senere kan benyttes til at beregne et passende træningsniveau. Der er dertil opstillet Vis, Lyt, Hent, Beregn og Start-metoder til denne controller.

I sammenspil med designklasserne er der udarbejdet et sekvensdiagram, hvilket fremgår af figur 6.10.

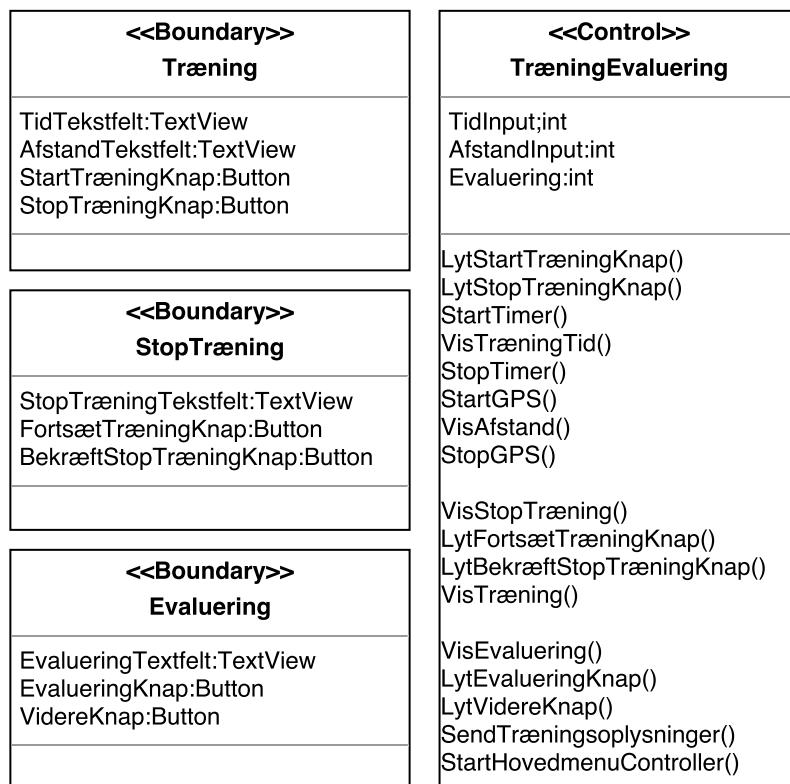


Figur 6.10: Sekvensdiagram for tilpasning af træningsniveau.

Den første grænseflade, som vises, er *Træningsform*. Denne grænseflade indeholder et tekstfelt, der beskriver, hvad brugeren skal angive. Dertil er der opstillet en TræningsformKnap, hvor brugeren kan vælge mellem konditionstræning, styrketræning eller vejrtrækningsøvelser. Når der er angivet træningsform, trykker brugeren på VidereKnap. Hvorefte controlleren, *Tilpasning*, viser grænsefladen for valg af *Træningstype*. Brugeren skal her angive træningstype ud fra tre forskellige muligheder. Dette vil for eksempel ved konditionstræning være gå, løbe eller cykle. Brugeren bekræfter valget ved at trykke på VidereKnap, hvorefter controlleren viser grænsefladen for *Helbredstilstand*. Helbredstilstanden angives og bekræftes ved at trykke på VidereKnap. Herefter henter controlleren kategoriseringen i modellen for *Brugeroplysninger* og efterfølgende tidligere evaluering via controlleren for *Database* i *Database*, hvis der eksisterer en tilhørende evaluering. Efterfølgende beregner controlleren *Tilpasning* anbefalet træningsniveau og viser dette på grænsefladen for *AnbefaletTræning*. Brugeren kan herefter trykke på VidereKnap, hvis denne trykkes på, viser controlleren grænsefladen for *Træning*.

Træning og evaluering

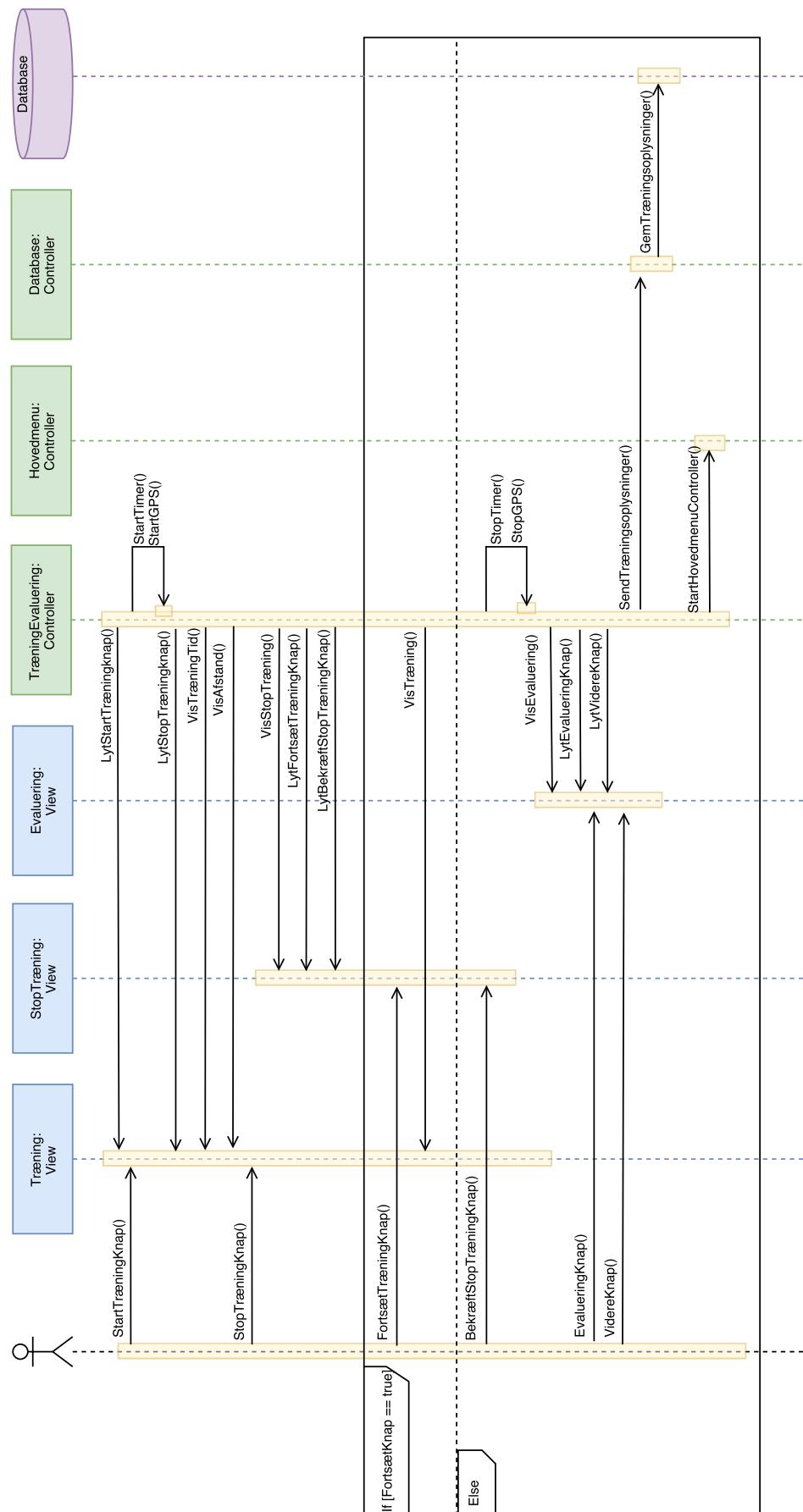
Træningen er opdelt i tre boundarys. Boundaryen *Evaluering* er først aktiv efter træningen, idet træningen skal evalueres. Der er til disse tre boundarys opstillet en fælles controller. Boundary og controller fremgår af figur 6.11.



Figur 6.11: Designklasser for træning samt evaluering. Til venstre fremgår boundarys for Træning, StopTræning og Evaluering. Den tilhørende controller ses til højre.

Grænsefladen for *Træning*, *StopTræning* og *Evaluering* indeholder tekster og knapper af typen TextView og Button. Controlleren *TræningEvaluering* indeholder metoderne Lyt, Vis, Start, Hent, Stop og Send.

I sammenspil med designklasserne er der udarbejdet et sekvensdiagram, hvilket fremgår af figur 6.12.

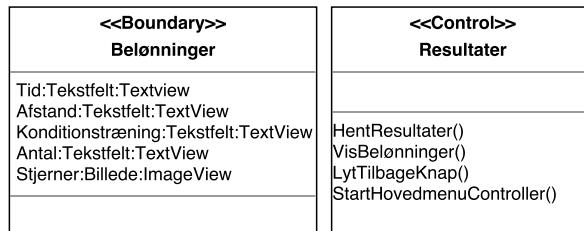


Figur 6.12: Sekvensdiagram for træning.

Træning er den første grænseflade, der vises efter tilpasningen af træningsniveauet. Controlleren *TræningEvaluering* starter timer og GPS, når brugeren har trykket på StartTræningKnap. Herefter vises tiden og afstanden i grænsefladen. Brugeren kan under træningen, eller når træningen er fuldført, afslutte ved at trykke på StopTræningKnap, hvorefter grænsefladen for *StopTræning* vises. I denne grænseflade har brugeren mulighed for at trykke på FortsætTræningKnap eller BekræftStopTræningKnap. Vælges FortsætTræningKnap vises grænsefladen for *Træning* og brugeren skal igen trykke på StartTræningKnap for at starte træning og måleenheder. Vælges BekræftStopTræningKnap stopper controlleren med at lagre data fra GPS og timer. Herefter vises grænsefladen *Evaluering*, hvor brugeren skal angive en evaluering af dagens træning. Evaluering bekræftes ved VidereKnap. Herefter sender controlleren træningsoplysninger til controlleren for *Database*, som gemmer træningen i *Database*. Herefter gives der besked til *Hovedmenu* controlleren om at vise hovedmenuen.

Resultater

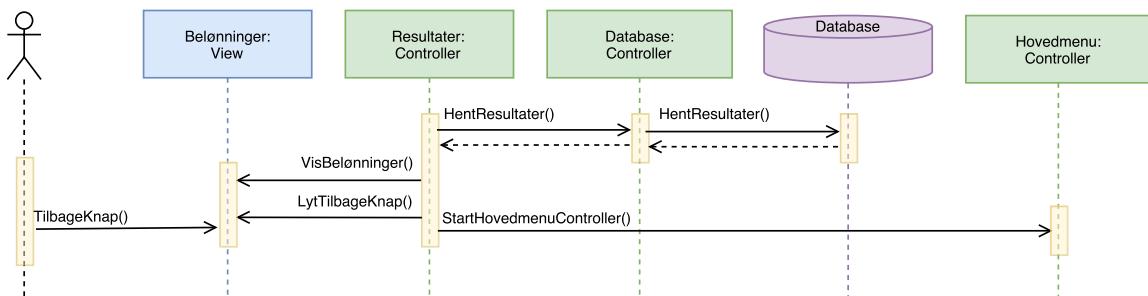
Resultater har en boundary, hvortil der er opstillet en controlklasse. Boundary og tilhørende controller fremgår af figur 6.13.



Figur 6.13: Designklasser for resultater. Til venstre ses boundary for Belønninger. Til højre fremgår den tilhørende controller.

I grænsefladen billede af typen ImageView og tekstmærker af typen TextView. Billederne viser antallet af stjerner, brugeren har opnået. Controlleren for Resultater indeholder Hent, Vis, Lyt og Start-metoder.

I sammenspil med designklasserne for resultater er der udarbejdet et sekvensdiagram, hvilket fremgår af figur 6.14



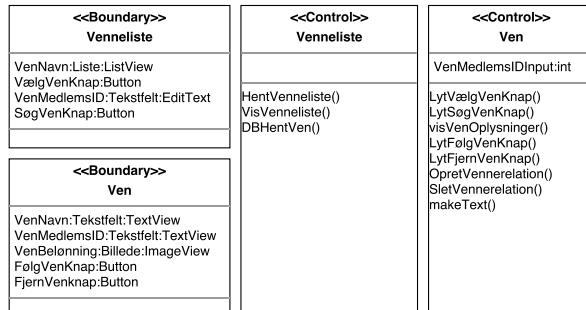
Figur 6.14: Sekvensdiagram for Resultater.

Når brugeren er tilgået resultater henter controlleren, *KonditionResultater* resultater fra *Databasen* via dens controller. Hvis brugeren endnu ikke har nogle resultater, vil disse ikke hentes, hvorved disse ikke vil vises i grænsefladen for *Belønninger*. Grænsefladen viser stjerner

opnået indenfor tid, afstand, konditonstræning og antal træninger. Trykker brugeren ikke tilbage, forbliver brugeren på grænsefladen for *Belønninger*

Venneliste

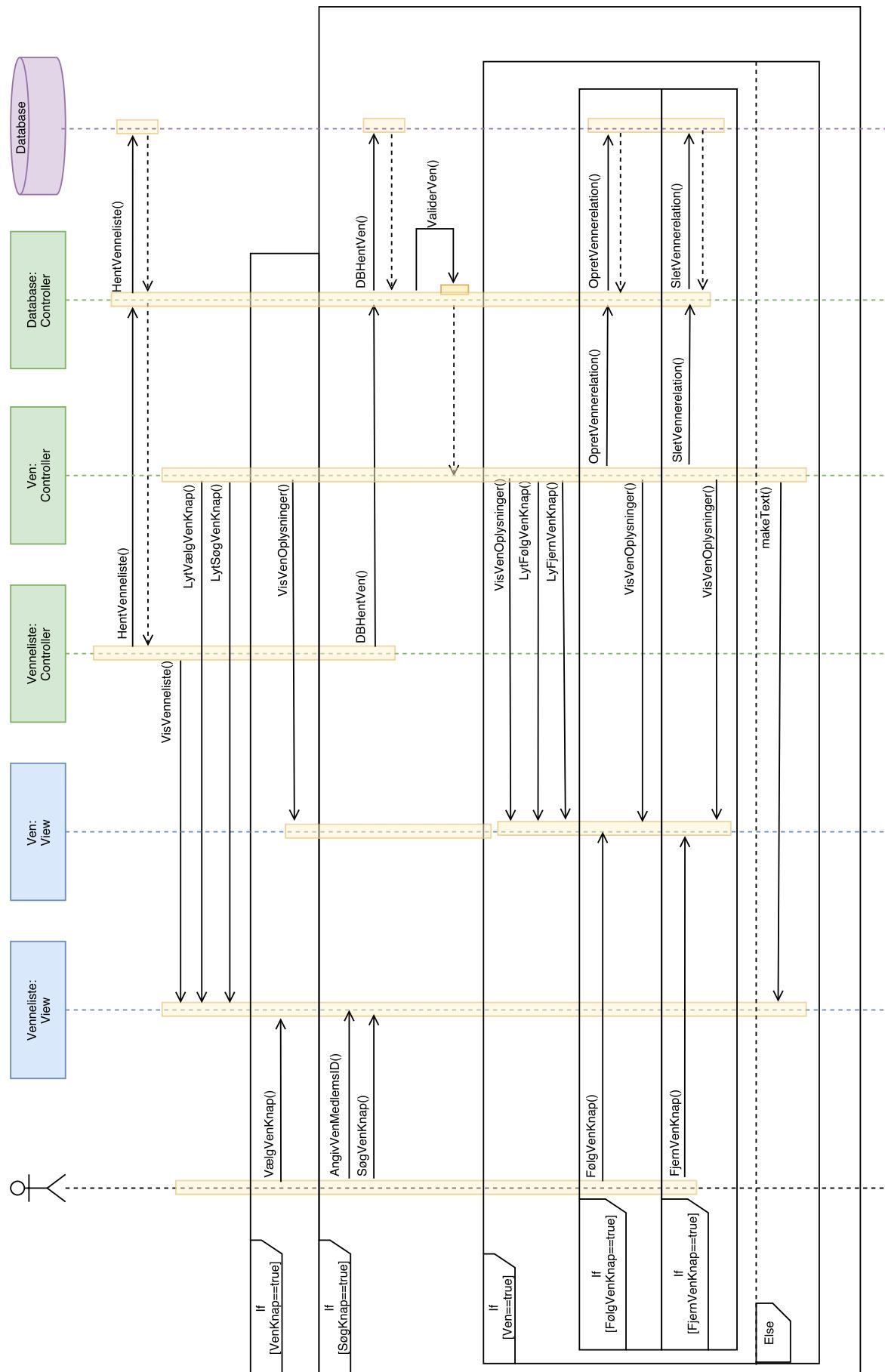
Venneliste inddeltes i to boundaryklasser og to controlklasser, som fremgår af figur 6.15.



Figur 6.15: Designklasser for venneliste. Til venstre ses boundary for Venneliste og Ven, hvor der til højre ses tilhørende controllere.

I grænsefladen *Venneliste* findes en liste, af typen ListView, med navn på de brugere, der følges. Hver bruger, der vises af den givne grænseflade, kan tilgås ved at trykke på brugeren. Derudover er der i denne grænseflade et søgefelt, af typen EditText, med en tilhørende SøgVenKnap, af typen Button, som muliggør, at brugeren kan søge på andre brugere med deres medlemsID. Grænsefladen, *Ven*, indeholder tekstfelter af typen TextView som viser brugerens navn og medlemsID. Derudover er der anvendt billede af typen ImageView, som viser brugerens belønninger. Hvis brugeren ikke følges, er der her opstillet en FølgVenKnap. Følges brugeren allerede er der opstillet en FjernVenKnap.

Controlleren for *Venneliste* har metoderne Hent og Vis, mens *Ven* indeholder Lyt, Opret, Slet og makeText. Metoden makeText anvendes i forbindelse med fejlmeddelelser. Ud fra de nævnte designklasser er udarbejdet et sekvensdiagram, der fremgår af figur 6.16.

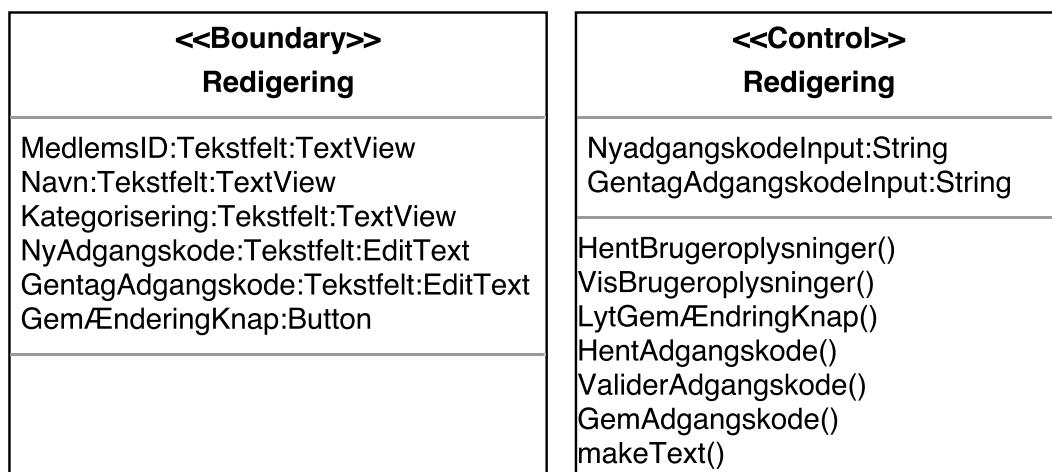


Figur 6.16: Sekvensdiagram for venneliste.

Controlleren *Venneliste* henter venneliste fra *Database* via dens tilhørende controller, hvorefter vises grænsefladen for venneliste. Dertil lytter controlleren for *Ven* på VælgVenKnap samt SøgVenKnap. Vælger brugeren at tilgå en ven fra vennelisten, vises grænsefladen *Ven*. Vælger brugeren derimod at søge efter en anden bruger, hentes det indskrevne medlemsID, således den søgte bruger kan findes i databasen, hvortil den valideres. Såfremt brugeren ikke findes i databasen vises en fejlmeddeelse i *Venneliste* grænsefladen. Findes brugeren i databasen vises grænsefladen *Ven*, hvor brugeren har mulighed for at følge eller fjerne en bruger, hvis denne allerede følges. Hvis brugeren trykker på FølgVenKnap oprettes der en vannerelation i databasen, hvor vannerelationen modsat vil slettes ved tryk på FjernVenKnap.

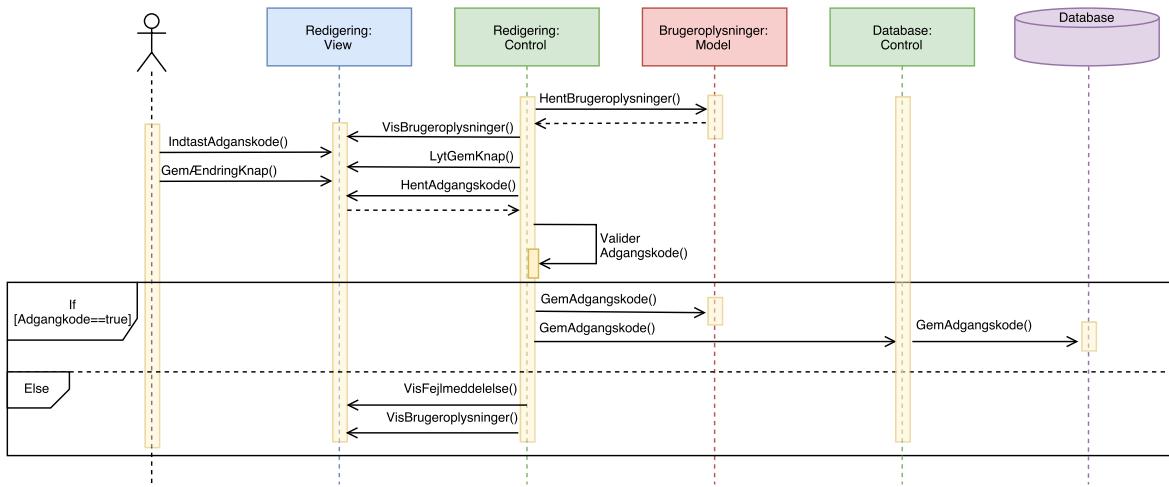
Redigering

Redigering inddeltes i en boundary og en tilhørende controller, som det fremgår af figur 6.17.



Figur 6.17: Designklasser for Redigering. Til venstre fremgår boundary og til højre controller for redigering.

I grænsefladen for *redigering* opstilles tekster af typen TextView for medlemsID, navn og kategorisering. Derudover opstilles tekster, af typen EditText, for ny adgangskode og gentag adgangskode, hvor brugeren kan angive en ny adgangskode. Dertil er der en GemÆndringKnap, af typen Button. GemÆndringKnap indikerer ved tryk, at brugeren ønsker at gemme den nye adgangskode. Den tilhørende controller har metoderne Hent, Vis, Lyt, Valider, Gem og makeText. Den sidste nævnte anvendes til fejlmeddelelse. Til disse klasser er der opstillet et sekvensdiagram, hvilket fremgår af figur 6.18.



Figur 6.18: Sekvensdiagram for redigering.

Controlleren *Redigering* henter brugeroplysninger fra modellen *Brugeroplysninger*. Disse oplysninger vises i grænsefladen *Redigering*, hvortil brugeren kan se sin information. Fra denne grænseflade er det ligeledes muligt for brugeren at ændre sin adgangskode. Adgangskoden skal indtastes to gange for at sikre, at adgangskoden er identisk. Dertil skal adgangskoden være minimum 10 karakterer lang. Adgangskoden valideres, hvis adgangskoden overholder de fornævnte kriterier og gemmes direkte i *Database* via dens controller og i modellen, *Brugeroplysninger*. Dette gøres af sikkerhedsmæssige årsager, således koden ikke først gemmes, idet brugeren logger ud af app'en. Overholder adgangskoden derimod ikke kriterierne, en fejlmeddelelse i grænsefladen. Denne grænseflade forsvinder efter kort tid, hvortil grænsefladen for redigering vises igen.

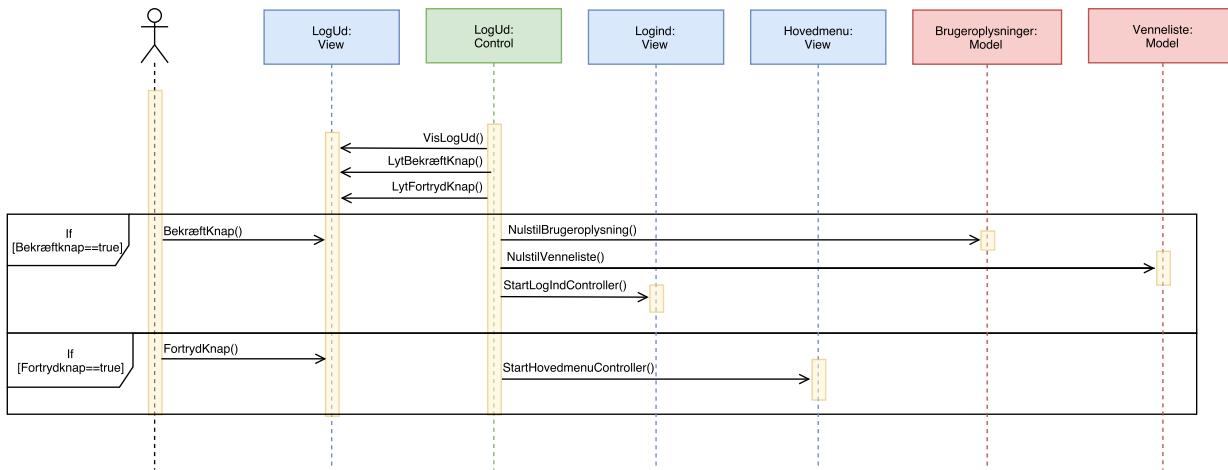
Log ud

Log ud inddeltes i en boundary og en dertilhørende controller, som det fremgår af figur 6.19.

<<Boundary>> Logud	<<Control>> Logud
Logud:Tekstfelt:TextView BekræftKnap:Button FortsætKnap:Button	VisLogud() LytBekræftKnap() LytFortsætKnap() NulstilBrugeroplysning() NulstilVenneliste() StartLogIndController() StartHovedmenuController()

Figur 6.19: Designklasser for log ud. Til venstre ses boundary og til højre controller.

Der opstilles i log ud et tekstfelt, af typen TextView. Dertil opstilles en BekræftKnap og FortsætKnap af typen Button. Den tilhørende controller indeholder metoderne, Vis, Lyt, Nulstil og Start. I sammenspil med designklasserne er der opstillet et sekvensdiagram, som fremgår af figur 6.20.



Figur 6.20: Sekvensdiagram for log ud.

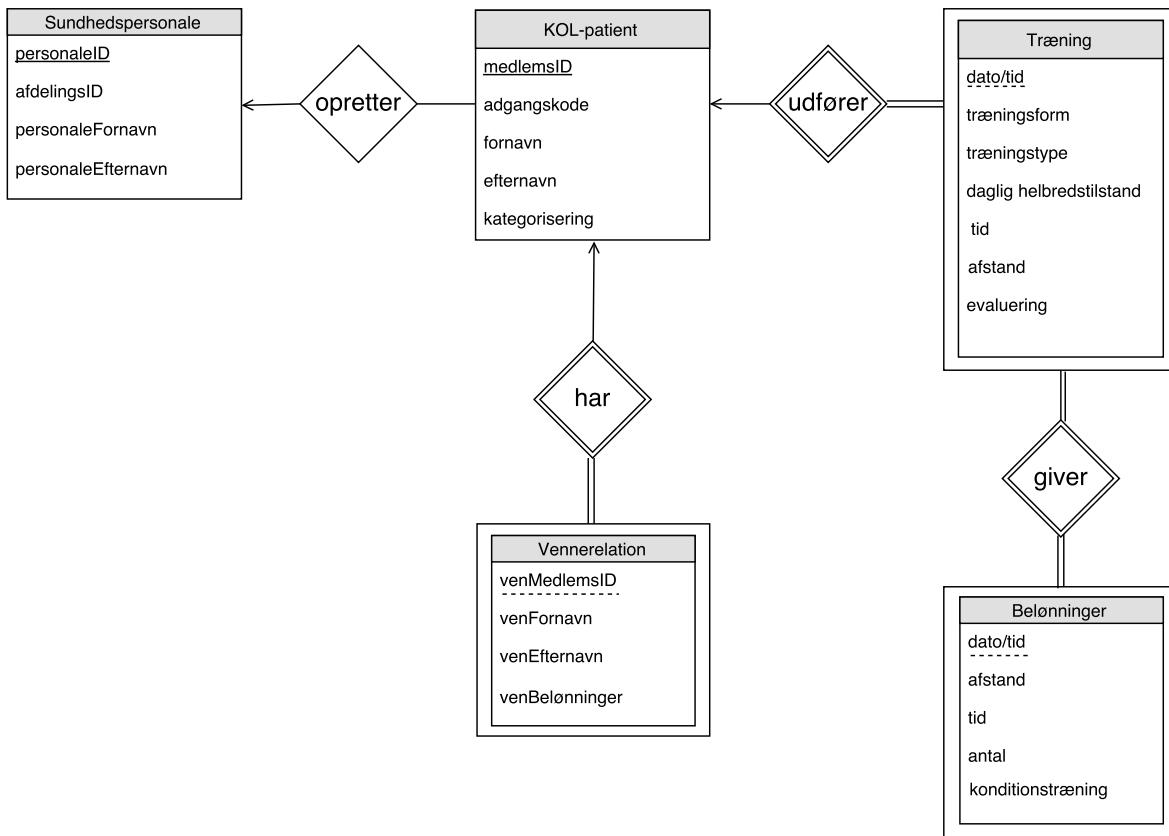
Når brugeren via grænsefladen for hovedmenuen trykker på knappen for log ud vises grænsefladen for *Log ud*. Her har brugeren mulighed for at trykke på BekræftKnap eller Fortryd-Knap. Hvis brugeren trykker på BekræftKnap, nulstilles modellen for *Brugeroplysninger* og *Venneliste*, hvorefter grænsefladen for *Log ind* vises. Trykker brugeren på FortrydKnap, vises grænsefladen for *Hovedmenu* igen.

6.3 Design af database

Det ønskes, at brugerdata er knyttet til den enkelte bruger i databasen. Dette er med henblik på, at app'en ikke skal lagre større mængder data på den mobile enhed samt sikre data i tilfælde af uforudsete hændelser, som eksempelvis tab af mobil enhed. Databasen skal indeholde oplysninger om de enkelte KOL-patienter, herunder resultater opnået ved træning og vennrerelation.

6.3.1 ER-diagram

Modellering af databasen udarbejdes ud fra et ER-diagram. ER-diagrammet relaterer sig til én KOL-patient i databasen. Databasen tager udgangspunkt i entiteter, som sundhedspersonale, KOL-patient, vennrerelation, træning og belønninger. ER-diagrammet for databasen fremgår af figur 6.21.

**Figur 6.21:** ER-diagram for database.

Af figur 6.21 ses ER-diagrammet over databasen, hvori KOL-patienter oprettes og informationer om patienterne samt deres resultater lagres. Sundhedspersonalet fremgår som en stærk entitet, der opretter alle KOL-patienter, hvor én KOL-patient oprettes i databasen én gang. Sundhedspersonalet defineres med et personaleID, afdelingID, personalefornavn og personaleefternavn, hvor *personaleID* er primærnøglen, der kan identificere medarbejderen. Den enkelte KOL-patient er en stærk entitet, som registreres med primærnøglen, *medlemsID*, samt adgangskode, fornavn, efternavn og kategorisering. Derudover fremgår de svage entiteter, herunder vennrelation, træning og belønninger. Én KOL-patient har mange vennrelationer, som kan identificeres ved KOL-patientens *medlemsID* og *venMedlemsID*. Det samme gør sig gældende for træninger, hvor én KOL-patient kan udføre mange træninger, der identificeres ved *dato/tid* og *medlemsID*. Ligeledes kan belønninger, som er en mange til mange relation, identificeres ved *dato/tid* og *medlemsID*.

6.3.2 Schema

ER-diagrammet omskrives til schema for at kunne normalisere og implementere databasen. Normaliseringen anvendes med henblik på at reducere redundans og inkonsistens. Schema er i anden normalform og fremgår af tabel 6.1.

Stærke entiteter	Sundhedspersonale = (<u>personaleID</u> , afdelingsID, personaleFornavn, personaleEfternavn) KOL-patient = (<u>medlemsID</u> , adgangskode, fornavn, efternavn, kategorisering)
Svage entiteter	Træning = (<u>medlemsID</u> , <u>tid/dato</u> , træningsform, træningstype, daglig helbredstilstand, tid, afstand, evaluering) Vennerelation = (<u>medlemsID</u> , <u>venMedlemsID</u> , venFornavn, VenEfternavn venBelønninger) Belønninger = (<u>medlemsID</u> , <u>tid/dato</u> , afstand, tid, antal, konditionstræning)

Tabel 6.1: ER-diagram for databasen omskrevet til schema på anden normalform.

Schemaet på anden normalform optimeres til tredje normalform, da det giver bedre muligheder ved implementering af databasen. For at komme på tredje normalform fjernes dimensioner, der ikke har en direkte tilgang til primærnøglen. Tredje normalform ses af tabel 6.2.

Stærke entiteter	Sundhedspersonale = (<u>personaleID</u> , afdelingsID) KOL-patient = (<u>medlemsID</u> , adgangskode, kategorisering)
Svage entiteter	Træning = (<u>medlemsID</u> , <u>tid/dato</u> , træningsform, træningstype, daglig helbredstilstand, tid, afstand, evaluering) Vennerelation = (<u>medlemsID</u> , <u>venMedlemsID</u> , venBelønninger) Belønninger = (<u>medlemsID</u> , <u>tid/dato</u> , afstand, tid, antal, konditionstræning)

Tabel 6.2: ER-diagram for databasen omskrevet til schema på tredje normalform.

Kapitel 7

Implementering

I dette kapitel beskrives implementeringen af app'en, der omhandler transformationen fra design til kode. Dette vil være med udgangspunkt i designklasserne for henholdsvis grænseflade, model og controller. Hertil vil databasen samt kommunikation med denne indrages for således at give et indblik i, hvordan dette er implementeret. Ydermere vil de centrale elementer i app'en beskrives. Disse elementer omhandler tilpasning af træningsniveau samt resultater for brugeren. Overordnet vil beskrivelse tage udgangspunkt i udpluk af den implementeret kode.

I kapitel 6 er analyse- og designklasser samt funktionsnavne navngivet på dansk, hvorfor bogstaverne æ, ø og å forekommer. Disse symboler anvendes ikke under implementeringen for at undgå fejl.

7.1 Android Studio

Det er valgt at implementere app'en i Android Studio version 2.3.1 og programmere i Java, da dette er et objektorienteret programmeringssprog [33]. Android Studio er et officelt Integrated Development Environment (IDE) for udvikling af android app's [?].

Strukturen i Android Studio opdeles i Manifests, res samt java mappe. Manifests mappen indeholder en *AndroidManifest.xml* fil, der er essentiel for, at app'en kan køre og indeholder aktiviter, der er implementeret. I manifestet er der derudover defineret tilladelser for den givne app. Dette kan eksempelvis være tilladelse til brug af internet samt GPS, der har været nødvendigt for at kunne tilgå databasen og beregne afstand i relation til konditionstræningen. Res mappen indeholder ikke-kode ressourcer, der eksempelvis er de forskellige layouts. Javaklasserne, der er oprettet i forbindelse med udviklingen af app'en er placeret i java mappen.[?]

7.2 Implementering af grænseflader

Til at implementere boundary klasserne i systemet benyttes XML-filer, der håndteres i deres respektive controllore. I disse filer er det muligt at definere, hvilke type elementer, der skal indgå i layoutet. Dertil kan typen af layout defineres alt efter, hvordan layoutet skal opstilles. Layouts brugt i dette system er af type linear og relativ layout.

Elementer opstilles i layouts med type, størrelse samt orientering. Forekommer det, at elementerne skal benyttes i javakoden, defineres disse ligeledes med et id. Et eksempel af layoutkode fra log ind ses af figur 7.1.

```

<!-- user adgangskode -->
<EditText
    android:id="@+id/adgangskode"
    android:layout_width="200dp"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:inputType="textPassword" />

<!-- Login Button -->
<Button
    android:id="@+id	btnLogin"
    android:layout_width="100dp"
    android:layout_height="50dp"
    android:layout_marginTop="10dp"
    android:layout_gravity="center"
    android:text="LOG IN" />

```

Figur 7.1: Udklip af kode fra log ind layout. Udkippet viser et *EditText*, hvori brugeren angiver adgangskode samt en button for log ind.

Af dette udklip ses koden for tekstfeltet, hvori brugeren angiver adgangskode samt koden for layoutet af log ind-knappen. Feltet for adgangskoden er her af typen *EditText*, der tillader, at brugeren kan angive tekst. Dette felt har ligeledes et id, der muliggører at referere til fletet og hente den angivet adgangskode til validering af log ind. Knappen er opstillet af typen *button*, og har ligeledes et id, hvortil en listener er opstillet i javakoden, hvilket beskrives af afsnit 7.3

7.3 Implementering af model og controller klasser

Model og controllerklasser er implementeret i *Java Class* filer. Heri er klasser defineret med navn samt tilhørende attributter og metoder. Af figur 7.2 ses et eksempel på, hvordan en controller er implementeret.

```

40  public class LoginController extends Activity {
41      private Button btnLogin;
42      ...
43  }

```

Figur 7.2: Et udpluk af javaklassen for *LoginController*.

Det fremgår af dette udklip, at klassen er af typen *public* og nedarver *activity*. Dette gør sig gældende for samtlige klasser implementeret. Denne klasse for *LoginController* har attributten *btnLogin*, der referer til log ind knappen på grænsefladen. Idet app'en skal reagerer, når brugeren trykker på *btnLogin* opsættes en listener på knappen, hvilket ses af figur 7.3.

```

82      // Log ind knap listener
83      btnLogin.setOnClickListener(new View.OnClickListener() {
84          public void onClick(View view) {
85              ...
86          }
87      });

```

Figur 7.3: Et udpluk af listeneren opsat for log ind knappen.

Af dette udklip ses den opsatte listener, der har til formål at lytte på knappen. Indenfor listeneren er der opstillet kode, der køres idet der trykkes på knappen. Denne listener er implementeret for samtlige knapper i controllerklasserne, da det er disse klasser, der håndterer input for tilhørende layouts.

Modelklasserne er implementeret på samme vis som controller klasserne, de har dog til formål at lagre data forinden det sendes til databasen. Der er ligeledes opstillet attributter i modelklasserne med tilhørende get og set metoder.

7.3.1 Implementering af database

Til implementering af databasen samt kommunikation med denne benyttes programmet XAMPP. Programmet opretter en lokal apache webserver samt database på en given computer, der fungerer som et Localhost miljø. Den lokale webserver simulerer en ekstern webserver, hvilket giver et passende miljø til at teste og udvikle prototypen. Dertil anses dette ikke som værende fjernt fra et virkelighedsnært senarie, hvor systemet benyttes med ekstern server og database. Ved direkte håndtering af databasen benyttes phpMyAdmin, der er et online databaseadministrationssystem, som understøtter Structured Query Language (SQL) ???. Databasen er implementeret under navnet *db_KOL*, hvori tabeller oprettes på baggrund designet jf. afsnit 6.3 i relation til attributter og datatyper.

Kommunikation med databasen

Kommunikation mellem Android Studio og databasen kan ikke forekomme direkte, hvorfor PHP: Hypertext preprocessor scripts benyttes. PHP er et Server-Side Scripting Language, hvilket køres på serveren og muliggøre systemet kan tilgå databasen [?]. Der er oprettet et php-script, *config.php*, der indeholder informationerne host, bruger, adgangskode samt navn på den oprettede database. Dette script inkluderes i et separat script, *DB_connect.php*, til at etablerer forbindelsen til databasen. Der er opstillet forskellige PHP-scripts, som javakoden refererer til via URL links. Af disse links fremgår ip-adressen for serveren samt filplaceringen af det givne script.

De forskellige scripts får information fra app'en for således at kunne udføre SQL-kommandoer. Informationen sendes fra app'en som en JavaScript Object Notation (JSON) repræsentation. JSON er et dataoverførselsformat, der gør det muligt at pakke data som et objekt eller array. PHP-scripts brugt i dette projekt er sat op til at tjekke, hvorvidt der bliver sendt en værdi i det respektive navn. Er værdierne sat og ikke NULL, ligges værdierne i en ny variabel. Et eksempel af dette ses af figur 7.4, hvor et udklip af PHP-scriptet for log ind er visualiseret.

```

14 if (isset($_POST['medlemsid']) && isset($_POST['adgangskode'])) {
15
16     // modtager POST parametre: medlemsid og adgangskode
17     $medlemsid = $_POST['medlemsid'];
18     $adgangskode = $_POST['adgangskode'];
19
20     // hent brugeroplysninger med medlemsid og adgangskode
21     $user = $db->getUserByMedlemsidAndAdgangskode($medlemsid, $adgangskode);

```

Figur 7.4: Udklip af log ind PHP-script, hvor medlemsid samt adgangskode modtages og indsættes i tilhørende variabler, der benyttes i et funktionskald.

Dette udklip viser, hvordan koden modtages og gemmes i nye variabler. Værdierne gemmes i et associativt array, der ved hjælp af en POST-metode gør det muligt at overføre data til et andet script, *db_functions.php*, hvori SQL-kommandoer udføres. Linje 21 viser, hvordan variablerne benyttes i et funktionskald, der eksekveres i *db_functitons.php*. Af figur 7.5 ses et udklip af denne funktion.

```
173 public function getUserByMedlemsidAndAdgangskode ($medlemsid, $adgangskode) {
174
175     $stmt = $this->conn->prepare ("SELECT * FROM users WHERE medlemsid = ?");
176     $stmt->bind_param ("s", $medlemsid);
```

Figur 7.5: SQL-kommando for funktionen log ind.

Det ses af figur 7.5, hvordan en SQL-kommando udføres. Den viste SQL-kommando henter alt fra tabellen *users* tilhørende medlemsid'et i databasen, der er lig spørgsmålsteget. Spørgsmålsteget markerer et bindingspunkt, hvorpå en parametre kan tilknyttes. Parametren, der bindes på ved brug af *bind_param*, fremgår af linje 176, hvoraf "s" indikerer, at medlemsid'et er af typen string. I dette tilfælde valideres log ind informationerne førend user returneres til log ind PHP-scriptet, der håndterer user, hvilket fremgår af figur 7.6.

```
23 if ($user != false) {
24     // brugeren er fundet
25     $response["error"] = FALSE;
26     $response["user"]["navn"] = $user["navn"];
27     $response["user"]["medlemsid"] = $user["medlemsid"];
28     $response["user"]["kategorisering"] = $user["kategorisering"];
29     echo json_encode($response);
30 } else {
31     // brugeren kunne ikke findes
32     $response["error"] = TRUE;
33     $response["error_msg"] = "Det indtastede medlemsid eller adgangskode er forkert.";
34     echo json_encode($response);
35 }
```

Figur 7.6: Datahåndtering i PHP-scriptet førend respons til app'en.

Som det ses af figur 7.6 opstilles if/else statement, der håndterer, hvorvidt *user* returneres. Hvis *user* returneres bliver brugerdata gemt i *response*, der sendes tilbage til app'en som en JSON repræsentation. Returneres *user* ikke, gemmes en fejlmeldelse i *response*, der ligeledes sendes til app'en som en JSON repræsentation.

Der er i javakoden opsat en *Response.listener*, der lytter efter respons. Ved respons oprettes et nyt JSON objekt, hvori responset lagres, således dette kan benyttes i app'en.

7.3.2 Implementering af tilpasning af træningsniveau

I tilpasningcontrolleren beregnes den anbefalede træningstid. Dette gøres ud fra kategorisering, den daglige helbredstilstand og en tidligere evaluering fortaget efter samme træningsform, -type og heldbredstilstand. Beregningen for den anbefalede træning er implementeret ved brug af if/else statement. Et udpluk heraf fremgår af figur 7.7.

```

242     public void beregnanbefaling(final int helbredstilstand, final int evaluering, final String kategorisering) {
243         TextView TVAnbefaling = (TextView) findViewById(R.id.TVAnbefaling);
244
245         if (kategorisering.equals("A")) {
246             if (helbredstilstand == 2 && evaluering == 1) {
247                 TVAnbefaling.setText("Din anbefalede træningstid er 15 minutter");
248                 setMinutter(15);
249             } else if (helbredstilstand == 2 && evaluering == 0 || helbredstilstand == 2 && evaluering == 2) {
250                 TVAnbefaling.setText("Din anbefalede træningstid er 20 minutter");
251                 setMinutter(20);
252             } else if (helbredstilstand == 2 && evaluering == 3) {
253                 TVAnbefaling.setText("Din anbefalede træningstid er 25 minutter");
254                 setMinutter(25);

```

Figur 7.7: Udpluk af kodden for anbefalet træning. If/else løkker afgør, hvilket træningsniveau brugeren får anbefalet.

Af figuren ses beregningen af den anbefalede træningstid for brugeren med kategorisering "A" og heldbredstilstanden på "2". Variationen for anbefalingerne afhænger af evalueringen og kan i dette eksempel variere mellem "15", "20" og "25 minutter". Metodekalderet *setMinutter* refererer til, at der sættes en bemærkning idet den anbefalte træningstid er opnået.

7.3.3 Implementering af resultater

Belønninger gives udfra samlet tid, afstand, antal træninger samt antal konditionstræninger og beregnes efter endt træning. Da disse belønninger er baseret ud fra den samlede træning, hentes den samlagte tid, afstand samt totale antal træninger og omregnes til et tal mellem 1-6, der repræsenterer antallet af stjerner brugeren har opnået. Et udklip af javakoden for omregningen for bruger med en kategorisering "A" ses af ??.

```

258     // AFSTAND
259     if (afstand_t >= 5 && afstand_t < 25) {
260         b_afstand = 1;
261     } else if (afstand_t >= 25 && afstand_t < 50) {
262         b_afstand = 2;
263     } else if (afstand_t >= 50 && afstand_t < 100) {
264         b_afstand = 3;

```

Figur 7.8: Udklip af javakode for omregning af samlet afstand til optjente belønninger for bruger med kategorisering "A".

Det fremgår af dette javaudklip at omregningen foregår ved hjælpe af if/else løkker. Hvis afstanden ligger indenfor et interval af 5 og 25 sættes *b_afstand* lig 1, hvilket repræsenterer én stjerne. Af fremgår kriterierne for at opnå stjerner for bruger med kategoriseringen "A".

Samlet belønninger opnået ved træning						
	★	★★	★★★	★★★★	★★★★★	★★★★★★
Tid (min)	60	90	120	300	400	500
Afstand (km)	5	25	50	100	300	500
Træning (antal)	3	30	90	210	300	450
Konditionstræning (antal)	1	10	30	70	100	150

Tabel 7.1: Kriterier for at opnå belønninger inden for henholdsvis samlet tid, afstand og total antal træninger samt konditionstræninger for bruger med kategorisering "A".

Kapitel 8

Test

I dette kapitel beskrives test af app'en. Der tages udgangspunkt i at teste kravspecifikationer der blev opstillet i afsnit 5.2, hvorefter der blev opstillet et use casediagram på baggrund af disse. Testene er opdelt i database og de forskellige use cases. Hver test vil indeholde navnet på testen samt en beskrivelse af formålet og main flowet for testen. Efter hvert main flow beskrives det forventede output, hvorefter det samlede resultat af testen vil fremgå af resultat.

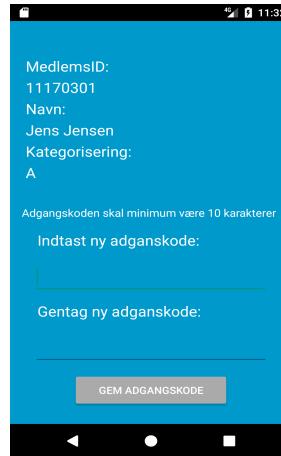
8.1 Database

Databasen anvendes i forbindelse med registrering af brugere. Derudover skal det være muligt for systemet at hente og gemme data i en database. Der blev opstillet følgende funktionelle krav til database:

- Brugere skal kunne oprettes i en database
Dette er nødvendigt for, at brugere kan anvende app'en
- Systemet skal kunne gemme og hente data i en database
Dette er nødvendigt for, at brugere kan tilgå brugerdata

For at teste om de opstillede krav til database er overholdt udføres testen, som fremgår af tabel 8.1.

Test:	Database
Formål:	Formålet er at oprette brugere i databasen samt hente og sende data i databasen. Dette gøres ved at oprette en bruger i databasen, efterfølgende udføre en kategorisering af brugeren i app'en, hvorefter der tjekkes om denne kategorisering er gemt i databasen. Til sidst skal brugeren gå til rediger adgangskode og tjekke om kategoriseringen er angivet og derved hentet fra databasen.
Main flow:	<ol style="list-style-type: none">Indtast SQL-forespørgsel INSERT INTO 'users' ('id' , 'unique_id' , 'navn', 'medlemsid', 'db_adgangskode' , 'kategorisering') VALUES ('" , " , 'Jens Jensen', '11170301', 'adgangskode', 'F').<ul style="list-style-type: none">✓ Bruger er oprettet i databasen.Åben app'en og log ind med medlemsID og adgangskode. Udfør kategorisering og få en samlet CATscore under 10 ved at vælge værdierne 0,1,2,1,0,1,0,1, tryk videre efter hver angivet værdi. Herefter vælges antallet af indlæggelser til 0 INDLÆGGELSER.<ul style="list-style-type: none">✓ Kategorisering er gemt i databasenTryk på REDIGER ADGANGSKODE via hovedmenuen.<ul style="list-style-type: none">✓ Kategorisering vises som A under rediger adgangskode.

Resultat:			
<p>Resultater for test af database fremgår af ovenstående figurer. Til venstre er der opnået en kategorisering svarende til A, som på figuren i midten bliver gemt i databasen. Til højre vises redigering af adgangskode, hvor brugeroplysninger om den oprettede bruger, Jens Jensen, vises. Derudover fremgår den opnåede kategorisering A, hvorfra denne er hentet fra databasen.</p> <p>✓ På baggrund af dette er kravene for databasen opfyldt.</p>			

Tabel 8.1: Test af database

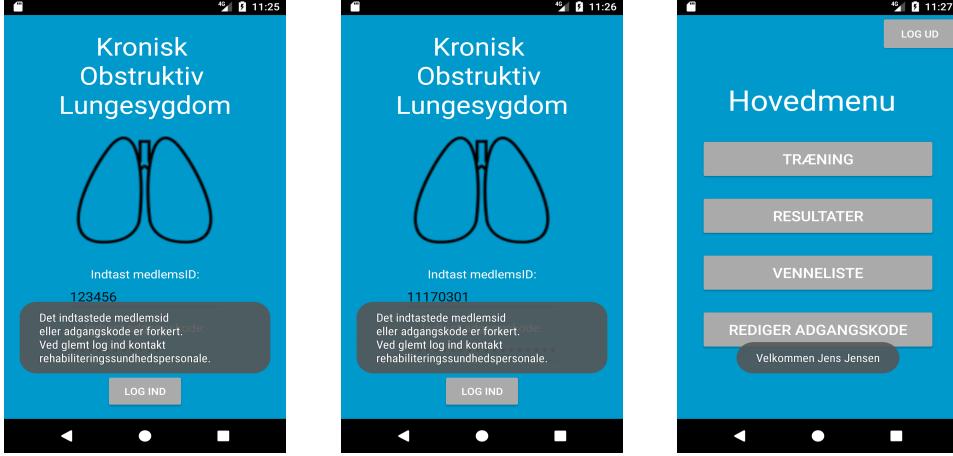
8.2 Log ind

Log ind anvendes for at adskille at brugere, som er registeret i databasen. Derudover er det med til at sikre, at brugeren har deltaget i et rehabiliteringsforløb, da det kun er disse brugere, der bliver registreret i databasen. Der er opstillet funktionelle krav til log ind:

- Brugere skal kunne logge ind med et personligt medlemsID og adgangskode, der er registreret i databasen
- Dette er nødvendigt for at tilgå og sikre, at brugere har deltaget i et rehabiliteringsforløb samt adskille brugeres data*

Til at teste, hvorvidt kravene til log ind er opfyldt, udføres testen, der fremgår af figur 6.3.

Test:	Log ind
Formål:	Formålet er at teste hvorvidt log ind-funktionen i systemet opfylder krav opstillet til log ind. Dette gøres ved at logge ind med en bruger, der findes i databasen og en bruger, som ikke findes i databasen.

Main flow:	<ol style="list-style-type: none"> 1. Indtast et medlemsID, som ikke eksisterer i databasen 123456 og en adgangskode, som eksisterer i databasen adgangskode og tryk på log ind knappen. <ul style="list-style-type: none"> ✓ Forventet log ind mislykkedes. Fejlmeddeelse vises i grænsefladen for log ind. 2. Indtast et MedlemsID, som eksisterer i databasen 11170301 og en adgangskode, som ikke tilhører medlemsID'et forkertadgangskode og tryk på log ind knappen. <ul style="list-style-type: none"> ✓ Forventet log ind mislykkes. Fejlmeddeelse vises i grænsefladen for log ind. 3. Indtast et medlemsID 11170301 og en adgangskode adgangskode, som eksisterer i databasen. <ul style="list-style-type: none"> ✓ Forventet log ind lykkes og hovedmenu vises.
Resultat:	 <p>Resultater for test af log ind fremgår af ovenstående figurer. Til venstre og i midten testes der for henholdsvis første og andet main flow. Hertil fremgår det, at log ind mislykkes, hvorfra fejlmeddelelsen vises. Til højre fremgår det, at log ind lykkes, og hovedmenuen vises.</p> <p>✓ På baggrund af dette er kravet for log ind opfyldt.</p>

Tabel 8.2: Test af Log ind

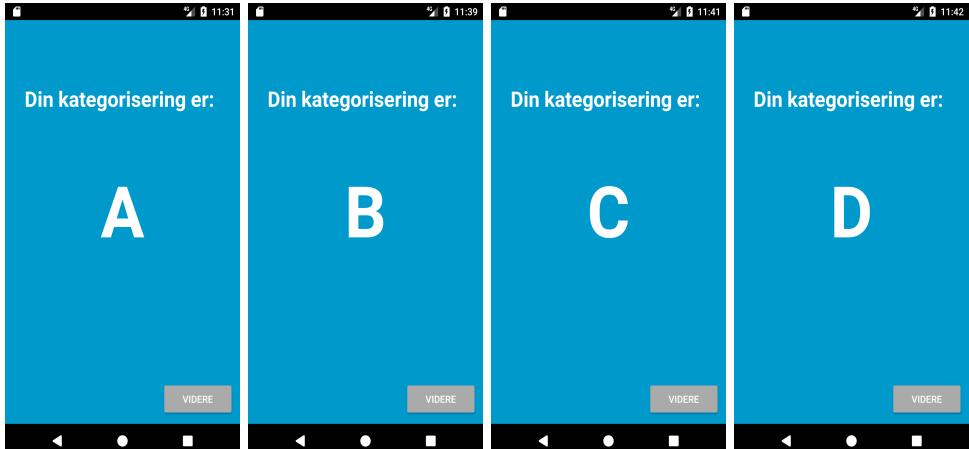
8.3 Kategorisering

Kategoriseringen skal foretages første gang brugeren logger ind i app'en og skal være en parameter i tilpasningen af et træningsniveau for den enkelte bruger. Følgende krav blev opstillet til kategoriseringen:

- Systemet skal kunne kategorisere brugere i ABCD på baggrund af CATscore og antallet af årlige indlæggelser på grund af KOL
Dette er nødvendigt for at kunne tilpasse træningen efter den enkelte bruger

Det testes, om de opstillede krav til kategorisering er overholdt. Testen fremgår af tabel 8.3.

Test:	Kategorisering
--------------	----------------

Formål:	Formålet er, at systemet skal kunne kategorisere brugere i ABCD efter, at brugeren har svaret på udsagn fra CATscore og angivet antallet af indlæggelser forårsaget af KOL inden for det seneste år. Dette gøres ved at angive forskellige værdier, der giver kategorisering A, B, C, eller D.
Main flow:	<p>1. Få en samlet CATscore under 10. Vælg 0,1,2,1,0,1,0,1, tryk videre efter hver angivet værdi og vælg til sidst antal indlæggelser 0 INDLÆGGELSER. ✓ Forventet kategorisering er A.</p> <p>2. Få en samlet CATscore over 10. 5,1,2,3,4,5,1,5, tryk videre efter hver angivet værdi og vælg til sidst antal indlæggelser 0 INDLÆGGELSER. ✓ Forventet kategorisering er B.</p> <p>3. Få en samlet CATscore under 10. 0,1,2,1,0,1,0,1, tryk videre efter hver angivet værdi og vælg til sidst antal indlæggelser 1 ELLER FLERE INDLÆGGELSER. ✓ Forventet kategorisering er C.</p> <p>4. Få en samlet CATscore over 10. 5,1,2,3,4,5,1,5, tryk videre efter hver angivet værdi og vælg til sidst antal indlæggelser 1 ELLER FLERE INDLÆGGELSER. ✓ Forventet kategorisering er D.</p>
Resultat:	 <p>Resultater for test af kategorisering fremgår af ovenstående figurer. Fra venstre mod højre ses, at henholdsvis første til fjerde main flow viser de forventede kategoriseringer.</p> <p>✓ På baggrund af dette er kravet for kategorisering er opfyldt.</p>

Tabel 8.3: Test af kategorisering

8.4 Tilpasning af træningsniveau

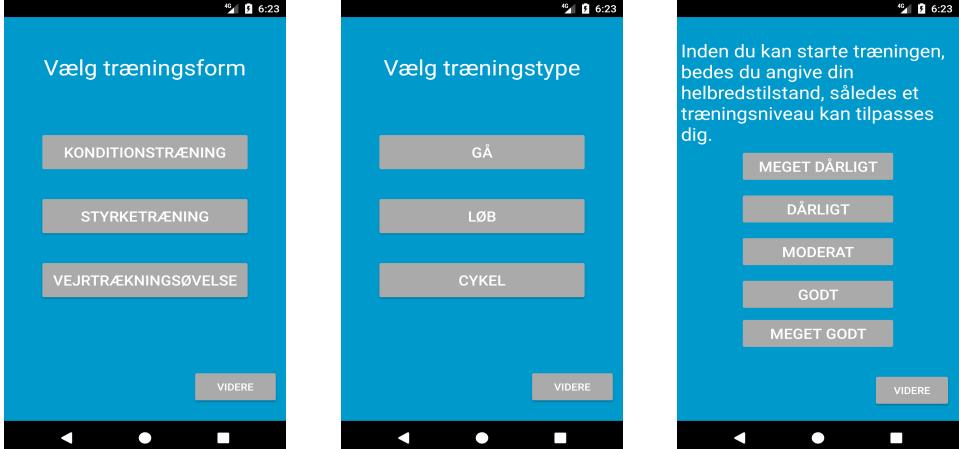
I tilpasningen af træningsniveau skal brugeren oplyses et anbefalet træningsniveau, med henblik på at tage højde for daglige variationer, ved at anvende parametre som kategorisering, daglig helbredstilstand og tidlige evalueringer af træninger. Følgende krav blev opstillet til tilpasningen af træningsniveauet:

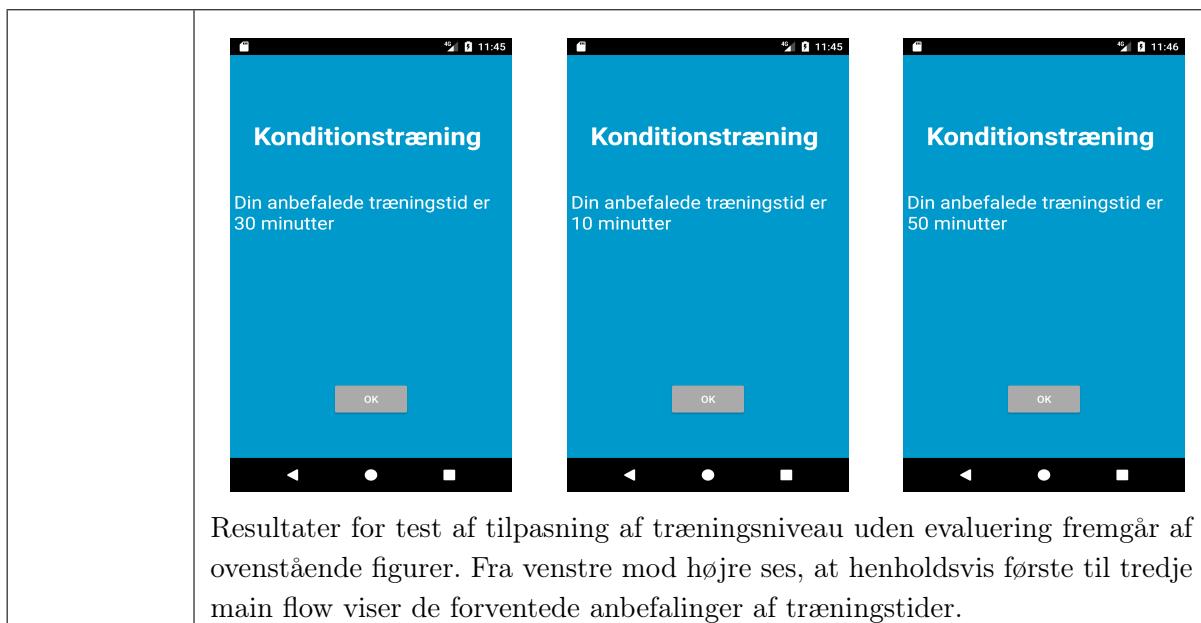
- Brugere skal kunne angive deres daglige helbredstilstand

Dette er nødvendigt for tage højde for daglige variationer og derved tilpasse træningen

for den enkelte bruger

Da der blev afgrænsset til konditionstræning er testen kun udført med konditionstræning. Derudover er testen opdelt i tilpasning af træningsniveau med og uden evaluering, da der ved første anvendelse ikke eksisterer en evaluering. Testen for tilpasning af træningsniveau uden evaluering fremgår af tabel 8.4 og med evaluering af tabel 8.5.

Test:	Tilpasning af træningsniveau uden evaluering
Formål:	Formålet er, at brugeren skal kunne angive ønsket træningsform, træningstype samt daglig helbredstilstand, hvorefter systemet på baggrund af dette samt kategorisering anbefale et træningsniveau. Dette gøres ved at angive samme træningsform, og vælge mellem de tre forskellige træningstyper og helbredstilsande. Brugeren er i kategoriseringen A.
Main flow:	<p>1 Vælg TRÆNING, KONDITIONSTRÆNING, GÅ, MODERAT og tryk videre efter hver.</p> <ul style="list-style-type: none"> ✓ Forventet anbefaling af træningstid er 30 min <p>2. Vælg TRÆNING, KONDITIONSTRÆNING, LØB, MEGET DÅRLIGT og tryk videre efter videre.</p> <ul style="list-style-type: none"> ✓ Forventet anbefaling af træningstid er 10 min <p>3. Vælg TRÆNING, KONDITIONSTRÆNING, CYKEL, MEGET GODT og tryk videre efter hver.</p> <ul style="list-style-type: none"> ✓ Forventet anbefaling af træningstid er 50 min
Resultat:	 <p>Ovenstående figurer viser grænsefladerne for valg af træningsform, træningstype og daglig helbredstilstand. Disse grænseflader vises før grænsefladen for anbefalingen af træningstid.</p>



Tabel 8.4: Test af tilpasning af træningsniveau uden evaluering

Test:	Tilpasning af træningsniveau med evaluering
Formål:	Formålet er, at brugeren skal kunne angive ønsket træningsform, træningstype samt daglig helbredstilstand, hvorefter systemet på baggrund af dette samt kategorisering og tidligere evalueringer skal anbefale et træningsniveau. Dette gøres ved at angive samme træningsform, og vælge mellem de tre forskellige træningstyper og helbredstilsande. Brugeren er i kategoriseringen A og har forinden træningen angivet evaluering for samme træning tidligere.
Main flow:	<p>1. Tidligere evaluering af samme træning er: :-). Vælg TRÆNING, KONDITIONSTRÆNING, GÅ, MODERAT og tryk videre efter hver.</p> <ul style="list-style-type: none"> ✓ Forventet anbefaling af træningstid er 30 min <p>2. Tidligere evaluering af samme træning er: :-D. Vælg TRÆNING, KONDITIONSTRÆNING, LØB, MEGET DÅRLIGT og tryk videre efter hver.</p> <ul style="list-style-type: none"> ✓ Forventet anbefaling af træningstid er 15 min <p>3. Tidligere evaluering af samme træning er: :-(. Vælg TRÆNING, KONDITIONSTRÆNING, CYKEL, MEGET GODT og tryk videre efter hver.</p> <ul style="list-style-type: none"> ✓ Forventet anbefaling af træningstid er 45 min

Resultat:			
Resultater for test af tilpasning af træningsniveau med evaluering fremgår af ovenstående figurer. Fra venstre mod højre ses, at henholdsvis første til tredje main flow viser de forventede anbefalinger af træningstider.			

✓ På baggrund af test af tilpasning af træningsniveau uden og med evaluering der kravet for tilpasning af træningsniveau opfyldt.

Tabel 8.5: Test af tilpasning af træningsniveau med evaluering

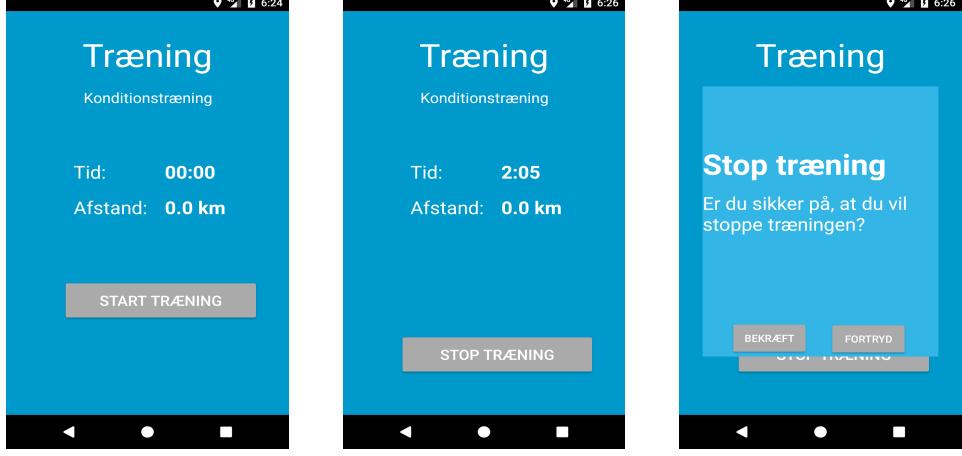
8.5 Træning

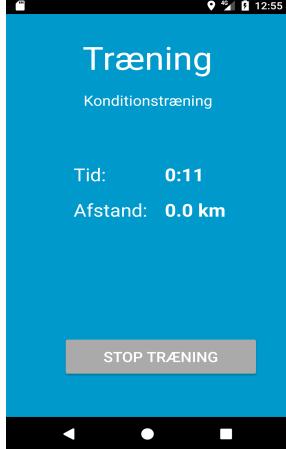
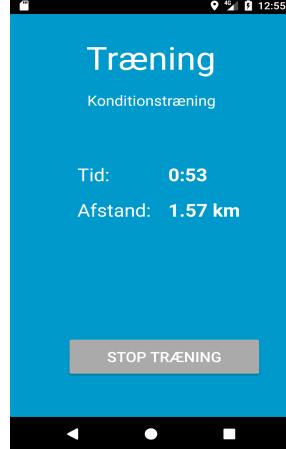
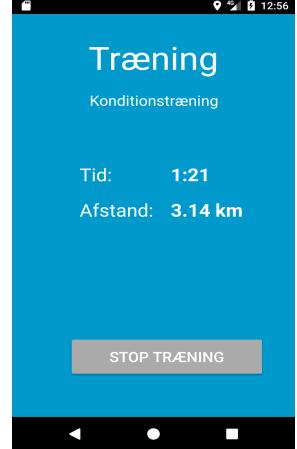
Træning skal muliggøre måling af tid og afstand under træning, derudover skal brugeren have mulig for at evaluere træningen efterfølgende. For træning er følgende krav opstillet:

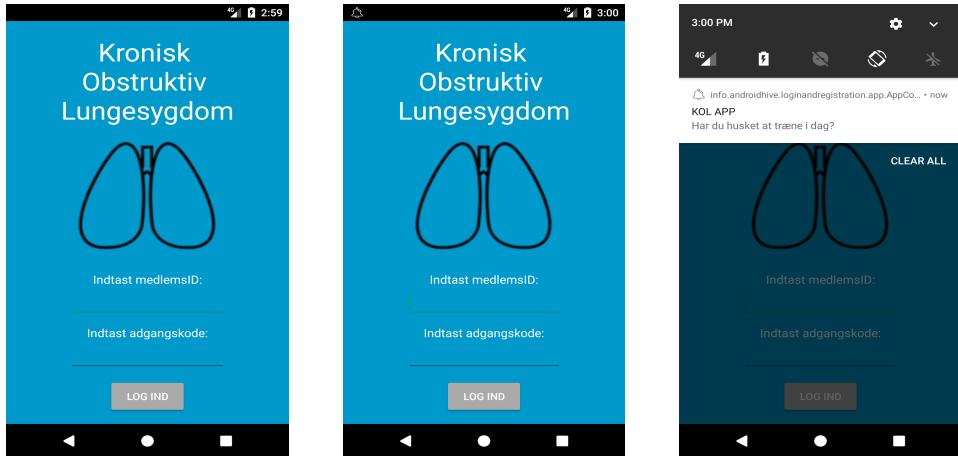
- Systemet skal kunne måle tid og afstand
Dette er nødvendigt for at monitorere træningen
- Brugere skal kunne evaluere hver træning
Dette er nødvendigt for at tilpasse træningen efter den enkelte bruger
- **Systemet skal kunne sende en notifikation, hvis brugere ikke har trænet før klokken 15**
Dette er nødvendigt for at tilpasse træningen efter den enkelte bruger

For at teste, hvorvidt de opstillede krav til træningen er opfyldt, udføres testen, som fremgår af tabel 8.6.

Test:	Træning
Formål:	Formålet er, at systemet skal kunne måle tid og afstand under træningen. Når brugeren har afsluttet træning skal måleenhederne stoppe, og brugeren skal kunne angive evaluering. Dette gøres ved at måle tid og afstand samt evaluere træningen efterfølgende. Derudover skal en notifikation forekomme, hvis brugeren ikke har trænet inden klokken 15.

Main flow:	<ol style="list-style-type: none"> Tryk START TRÆNING og vent til efter 2 minutter. Tryk herefter STOP TRÆNING. <ul style="list-style-type: none"> ✓ Forventet tid er over 2 minutter Tryk START TRÆNING åben Extended controls. Sæt herefter longitude samt latitude til 0 og tryk send. Ændre begge til 0.01 og tryk send. Ændre derefter begge til 0.02 og tryk send. Tryk herefter STOP TRÆNING. <ul style="list-style-type: none"> ✓ Forventet afstand ved 0 er 0 km. ✓ Forventet afstand ved 0.01 er 1.57 km. ✓ Forventet afstand ved 0.02 er 3.14 km. Tryk START TRÆNING og tryk herefter STOP TRÆNING og bekræft. Angiv herefter :-) og tryk videre. <ul style="list-style-type: none"> ✓ Forventet evaluering er gemt i databasen. Vent til klokken 15 med at udføre en træning <ul style="list-style-type: none"> ✓ Forventet notifikation klokken 15
Resultat main flow 1:	 <p>The figure consists of three screenshots of a mobile application interface for a training session. The first screenshot shows the start screen with the title 'Træning' and 'Konditionstræning'. It displays 'Tid: 00:00' and 'Afstand: 0.0 km'. A grey button labeled 'START TRÆNING' is visible. The second screenshot shows the same screen after 2 minutes, with 'Tid: 2:05' and 'Afstand: 0.0 km'. A grey button labeled 'STOP TRÆNING' is visible. The third screenshot is a pop-up window titled 'Stop træning' asking 'Er du sikker på, at du vil stoppe træningen?'. It has two buttons: 'BEKRÆFT' and 'FORTRYD'.</p> <p>Resultater for det første main flow fremgår af ovenstående figurer. På venstre ses grænsefladen for træning før træningen er på begyndt. Når der er trykkes på startknappen for træning starter timer, som kan ses af den midterste figur. Til højre fremgår et popup-vindue, der indikerer at træningen er stoppet.</p>

Resultat main flow 2:	 <p>Tid: 0:11 Afstand: 0.0 km</p> <p>STOP TRÆNING</p>	 <p>Tid: 0:53 Afstand: 1.57 km</p> <p>STOP TRÆNING</p>	 <p>Tid: 1:21 Afstand: 3.14 km</p> <p>STOP TRÆNING</p>
Resultater for det andet main flow ses af ovenstående figurer. På venstre ses grænsefladen for træningen, hvor det simuleret at brugeren har bevæget sig 0 km. I midten er det simuleret at brugeren har bevæget sig 1.57 km. Til højre er det simuleret at brugeren har bevæget sig 3.14 km.			

Resultat main flow 4:	
	<p>Resultater for fjerde main flow fremgår af ovenstående figurer. I midten ses det at notifikationen, i øverste venstre hjørne, er startet klokken 15. Notifikationen vises yderligere af figuren til højre.</p>

Resultat:

✓ På baggrund af resultater for main flow er kravene for træning opfyldt.

Tabel 8.6: Test af træning

8.6 Resultater

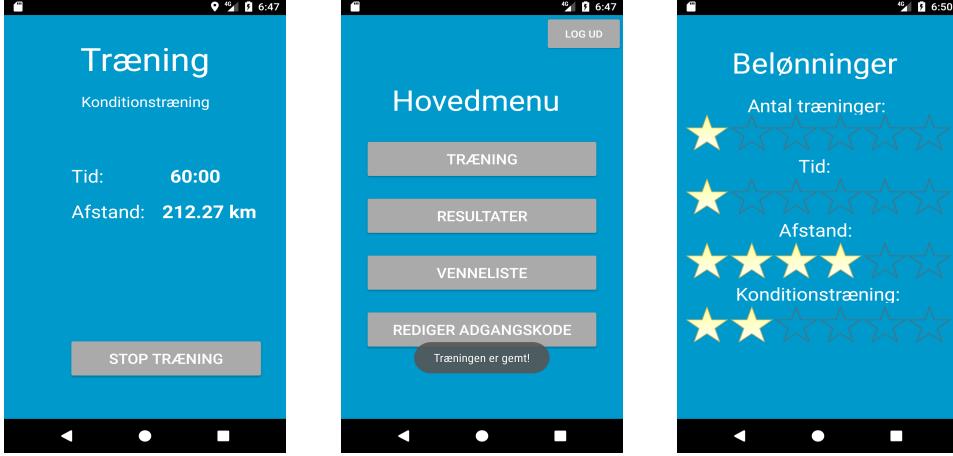
Resultater skal motivere brugeren til at udføre træningen. Følgende krav opstillet for resultater:

- Systemet skal kunne give virtuelle belønninger

Dette er nødvendigt for at kunne motivere brugere til at udføre træning

For at teste om de opstillede krav til resultater er overholdt, udføres testen, der fremgår af tabel 8.7.

Test:	Resultater
Formål:	Formålet er, at systemet skal kunne give virtuelle belønninger og sende en notifikation når brugeren har været inaktiv i 1 time. Dette gøres ved at træne imens der måles tid, afstand.

Main flow:	<p>1. Tryk på TRÆNING via hovedmenu. START TRÆNING, vent 60 minutter og tryk STOP TRÆNING og evaluér træningen til :-D. Tryk herefter på RESULTATER via hovedmenu og vælg BELØNNINGER.</p> <p>✓ Forventet stjerner under tid er en.</p> <p>2. Tryk på TRÆNING via hovedmenu. START TRÆNING og åben Extended controls. Sæt herefter longitude samt latitude til 0 og tryk send. Ændre herefter til 1.35. Tryk STOP TRÆNING og evaluér træningen til :-). Tryk herefter på RESULTATER via hovedmenu og vælg BELØNNINGER.</p> <p>✓ Forventet stjerner under afstand er fire.</p> <p>3. Tryk på RESULTATER via hovedmenuen og vælg BELØNNINGER</p> <p>✓ Forventet stjerner under antal træninger er én.</p> <p>✓ Forventet stjerner under konditionstræning er to.</p>
Resultat	 <p>På figurerne ovenfor vises resultaterne for test af resultater. Til venstre fremgår den ønskede tid og afstand. I midten gemmes resultaterne fra træningen, der af figuren til højre er gemt og belønninger er visualiseret som stjerner. De opnåede stjerner afspejler at brugeren har udført det antal træninger svarende til den opnåede stjerne.</p> <p>✓ På baggrund af dette er kravet for resultater opfyldt.</p>

Tabel 8.7: Test af resultater

8.7 Venneliste

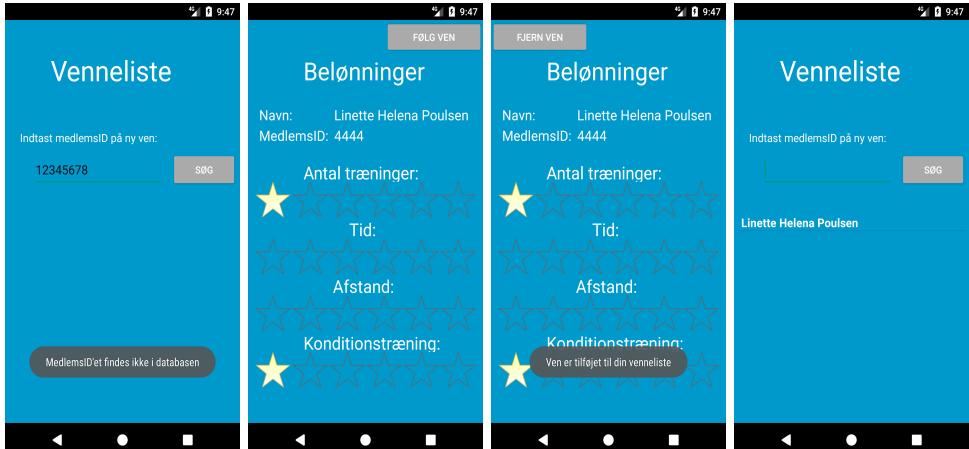
Vennelisten skal give en fællesskabsfølelse for brugeren ved at kunne følge og kunne tilgå andre brugeres virtuelle belønninger. Derudover skal vennelisten motivere brugeren til at udføre træning. Følgende krav er opstillet til vennelisten:

- Brugere skal kunne følge andre brugere

Dette er nødvendigt for at skabe fællesskab samt gøre det muligt for brugere at tilgå hinandens virtuelle belønninger, hvilket skal øge brugeres motivation

Der testes, om ovenstående krav til vennelisten er opfyldt. Testen fremgår af tabel 8.8.

Test:	Venneliste
-------	------------

Formål:	Formålet er, at brugeren kan følge andre brugere og tilgå deres belønninger. Dette gøres ved at indtaste et medlemsID på en bruger, som findes i databasen og et, som ikke eksisterer, hvorefter denne bruger følges.
Main flow:	<ol style="list-style-type: none"> Tryk VENNELISTE via hovedmenuen og indtast medlemsID 12345678 og tryk søg. <ul style="list-style-type: none"> ✓ Forventet søgning mislykket. Fejlmeddeelse vises i grænsefladen for venneliste. Tryk VENNELISTE via hovedmenuen og indtast medlemsID 4444 og tryk søg. <ul style="list-style-type: none"> ✓ Forventet søgning lykkes. Grænsefladen for søgte vens belønninger vises. Tryk VENNELISTE via hovedmenuen og indtast medlemsID 4444 og tryk søg. Herefter trykkes følg. <ul style="list-style-type: none"> ✓ Forventet følg knap forsvinder i grænsefalden for vennelisten og fjern knap synliggøres. Tryk VENNELISTE via hovedmenuen <ul style="list-style-type: none"> • Forventet bruger med medlemsID 4444 vises i grænsefladen for vennelisten.
Resultat	 <p>Ovenstående figurere viser resultater for test af venneliste. Til venstre ses figur for første main flow, hvor det indtastede medlemsID ikke eksistere i databasen. Figuren ved siden af viser andet main flow, hvor medlemsID'et eksistere i databasen og belønninger for denne bruger fremgår. Brugeren har mulighed for at tilføje ven til vennelisten, hvilket fremgår af de to figurere til højre, som opfylder tredje og fjerde main flow.</p> <ul style="list-style-type: none"> ✓ På baggrund af dette er kravet for venneliste opfyldt.

Tabel 8.8: Test af venneliste

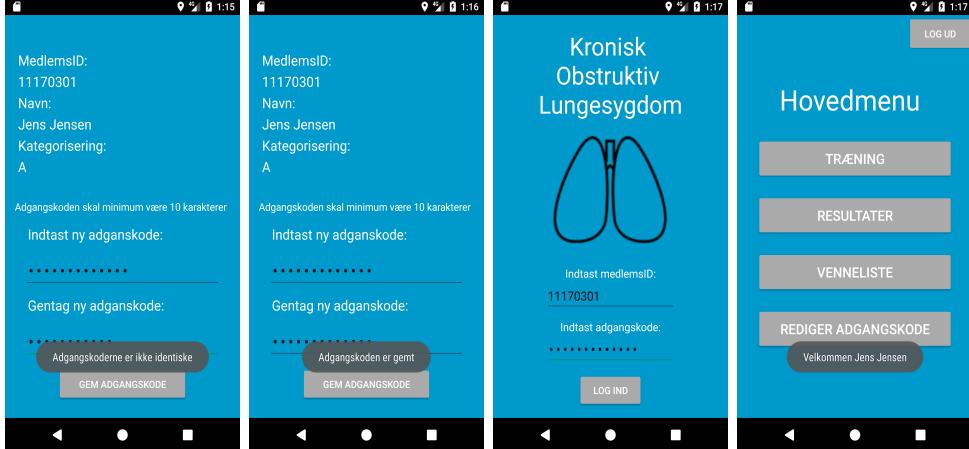
8.8 Redigering

Redigering skal give brugeren mulighed for at redigere adgangskoden til en personlig adgangskode, da brugeren får udleveret en ved registreringen. Der er opstillet følgende krav for redigering:

- Brugere skal kunne redigere deres adgangskode

Dette er nødvendigt for, at brugere skal kunne gøre deres adgangskode personlig

For at teste, hvorvidt kravene til redigering er overholdt, udføres testen, som fremgår af tabel 8.9.

Test:	Redigering
Formål:	Formålet er, at brugeren skal have mulighed for at redigere sin adgangskode. Dette gøres ved at indtaste to forskellige adgangskoder og to ens adgangskoder. Efterfølgende logges der ind med den ændrede adgangskode.
Main flow:	<p>1. Tryk REDIGER ADGANGSKODE via hovedmenuen. Indtast nyadgangskode i ny adgangskode og indtast adgangskode i gentag adgangskoden. Tryk GEM ÆNDRINGER.</p> <p>✓ Forventet ændring mislykkes. Fejlmeddeelse vises i grænsefladen for redigering.</p> <p>2. Tryk REDIGER ADGANGSKODE via hovedmenuen. Indtast nyadgangskode i ny adgangskode og gentag adgangskode. Tryk GEM ÆNDRINGER.</p> <p>✓ Forventet ændring lykkes. Meddeelse viser, at adgangskoden er gemt.</p> <p>3. Log ud og log ind med medlemsID 11170301 og den nye adgangskode nyadgangskode.</p> <p>✓ Forventet log ind lykkes og hovedmenu vises.</p>
Resultat	 <p>Ovenstående figurer viser resultater for test af redigering. Til venstre ses resultat for det første main flow, hvor det fremgår at de indtastede adgangskoder ikke er identiske. Til venstre for midten er adgangskoderne identiske, hvorved disse gemmes i databasen. Test for det tredje main flow fremgår af de to figurer til højre, hvor der logges ind med den nye adgangskode og hovedmenuen vises.</p> <p>✓ På baggrund af dette er kravet for redigering</p>

Tabel 8.9: Test af redigering

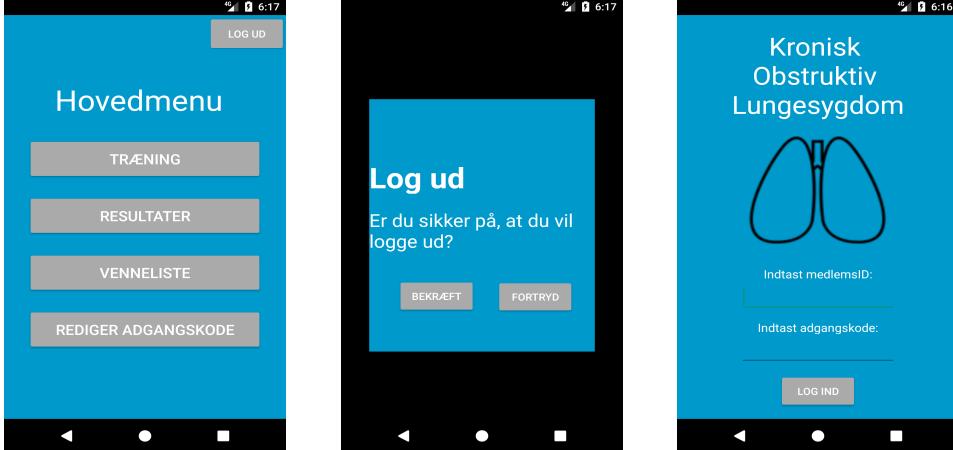
8.9 Log ud

Log ud-funktionen skal sikre brugerens individuelle data. De opstillede krav til log ud er følgende:

- Brugere skal kunne logge ud af app'en

Dette er nødvendigt for at sikre brugeres individuelle data

For at teste om ovenstående krav er opfyldt udføres testen, der fremgår af tabel 8.10.

Test:	Log ud
Formål:	Formålet er, at brugeren skal have mulighed for at logge ud af app'en.
Main flow:	1. Tryk LOG UD via hovedmenuen og tryk bekræft. ✓ Forventet grænseflade for log ind vises.
Resultat	 <p>Figurerne ovenfor viser resultater for test af log ud. Til venstre ses hovedmenuen, hvor brugeren har mulighed for at logge ud. I midten skal brugeren bekræfte log ud, hvis dette gøres vises grænsefladen for log ind, som ses af figuren til højre.</p> <p>✓ Ud fra denne test er kravet for log ud opfyldt.</p>

Tabel 8.10: Test af log ud

Kapitel 9

Syntese

9.1 Diskussion

I dette kapitel diskuteres væsentlige problemstillinger forbundet med problemformuleringen, herunder er formålet at udvikle en app, der kan vejlede og motivere KOL-patienter til regelmæssig træning. Dette indebærer hvad der er gjort for at besvare problemformuleringen samt mulige forbedringer i forhold til denne, hvorfor det er valgt at diskutere kravspecifikationer, design og implementering samt, hvilke ændringer og tilpasninger der muligvis skal foretages for at app'en kan anvendes i praksis.

9.1.1 Kravsspecifikationer

De opstillede funktionelle krav til app'en er testet og er alle overholdt. Det kan dog diskuteres, hvorvidt de kan specificeres yderligere i forhold til app'ens funktionaliteter. Hertil kunne det blandt andet diskuteres, hvorvidt et fastlagt tidspunktet for notifikation vil være hensigtsmæssigt for brugeren. Det formodes at brugerne af app'en træner på forskellige tidspunkter i løbet af dagen, hvorved en notifikation kan virke forstyrrende for brugeren, hvis den allerede har udført en træning tidligere på dagen. En løsning på dette kunne være, at notifikationen vil annulleres, hvis brugeren har trænet inden kl. 15:00 eller, at notifikationen først kommer 24 timer efter sidste udførte træning. Derudover kan det være, at notifikationen skal tilpasses brugerens kategorisering og helbredstilstand for ikke at virke demotiverende, da de måske ikke er i stand til daglig motion, hvorfor notifikationen for eksempel først skal forekomme efter 48 timer fra sidste udførte træning. Yderligere kan det diskuteres om den motiverende faktor kan optimeres ved brugen af notifikation. For eksempel ved at give feedback i form af ros, hvis brugeren har udført en træning eller informere brugeren om, hvor mange kilometer brugeren mangler for at opnå en belønning. Ved for mange notifikationer kan der muligvis opnås en modsatrettet effekt, hvor brugeren fravælger brug af app'en.

Det har ikke været muligt at teste non-funktionelle krav, da app'en kun er en prototype. Men da app'en er designet og derved også implementeret ud fra gestaltlovene omfattende opsætningen af grænseflader samt egenskaber for betydning af brugervenlighed, antages det, at de non-funktionelle krav er opfyldt. Dette er antaget på baggrund af, at grænsefladerne tager udgangspunkt i samme layout samt indeholder få knapper, der gentages løbende i app'en, som for eksempel knappen til at gå videre.

9.1.2 Design

I afsnittet for design er det valgt at afgrænse i forhold til træning. Projektets primære formål er ikke at udarbejde et træningsprogram, men at udvikle en app, hvorpå et træningsprogram senere kan inkorporeres. Afgrænsningen til konditionstræning er valgt, da målinger der vil

foretage ved konditionstræning, herunder tid og afstand er et krav og derfor skal implementeres og testes. Derudover skal det diskuteres, om andre målinger kan være mere hensigtsmæssige at implementere ved valg af styrketræning og vejrtrækningsøvelser, hvor afstand ikke er essentiel måling. Det kunne for eksempel være mere hensigtsmæssig at implementere gentagelser.

Det er valgt at designe valg af træningsniveauet ud fra en beslutningstabell, som medregner parametre, såsom kategorisering, daglig helbredstilstand og tidlige evalueringer, der afhænger af tidlige angivne helbredstilstande fra samme træningstype. Dette er gjort for at tage højde for daglige variationer samt brugerens evne til at vurdere sin daglige helbredstilstand i forhold til hvad KOL-patienter fysiske kan yde. De tidlige evalueringer, der afhænger af, at brugeren har valgt samme helbredstilstand og træningstype er ikke afhængig af tid, hvilket vil sige at denne evaluering kan være forældet, hvis brugerens opfattelse af deres helbredstilstand ændres. En bruger, der tidligere har angivet en daglig helbredstilstand til moderat vil for eksempel, hvis det generelle helbred forværres, angive den samme daglige helbredstilstand til god, da opfattelsen af helbredstilstanden er ændret. Den tilhørende evaluering vil derfor ikke være sigende.

Da det ikke har været muligt at finde litteratur eller teste i, hvilken grad parametrene har indflydelse på KOL-patienters helbred og fysiske egenskaber samt, hvordan den enkelte opfatter deres tilstand, hvis den ændres, og derved ikke angiver en sigende evaluering. En mulig løsning til disse problemstillinger kan være at anvende bayesian læring, som kan forbedre tilpasningen af træningsniveau over tid på baggrund af de angivne parametre og derved yderligere specialisere træningsniveauet til den enkelte bruger. Dette vil ligeledes tage højde for den ændring i opfattelse af helbredstilstand der muligvis kan forekomme.

Et af projektets formål er at motivere brugeren til at udføre regelmæssig motion, hvilket er forsøgt opfyldt ved, at brugeren kan opnå belønninger. I app'en fremgår det ikke, hvad der skal til for at opnå en belønning, og det kan diskuteres, hvorvidt dette skal være en mulighed for at øge motivation yderligere. En ulempe ved dette kan være, at brugeren bliver for konkurrenceminded og derved fravælger træningstyper samt overanstrenger sig for at opnå en belønning. Belønningerne opnås på samme måde for uafhængig af kategorisering, hvilket vil sige, at en bruger som er kategoriseret som D, vil skulle træne lige så meget som en der er kategoriseret i A. Det kan diskuteres, hvorvidt det vil virke demotiverende for brugerne, hvis belønningerne opleves for lette eller svære, da brugerne ikke kan præstere på samme vis. Derfor vil inddelingen af belønninger være mest hensigtsmæssigt at lave på baggrund af kategoriseringer.

Yderligere motivation er designet ved social interaktion, hvor brugere har mulighed for at se hinandens belønninger. Her vil der kunne opstå uligheder, hvis brugere med forskellige kategoriseringer vælger at følge hinanden, da belønningerne som tidligere nævnt ikke gives på baggrund af kategoriseringen. Det kan diskuteres, hvorvidt dette vil virke demotiverende eller motiverende for brugerne. Brugere med en kategorisering i A vil for eksempel kunne blive demotiverede, da niveauet ikke er højt nok eller motiveres idet de opnår mange belønninger i forhold til andre kategoriseringer. Modsat vil en brugere, der kategoriseres i D, have svært ved at opnå belønninger i forhold til andre kategoriseringer og måske anstrengte sig uhensigtsmæssigt for at opnå belønninger.

I forbindelse med social interaktion er app'en ikke designet så brugeren kan fravælge at andre brugere kan følge den. Dette kan muligvis være krænkende for nogle brugere, hvorved de kan fravælge brug af app'en. En løsning på dette kan være, at belønningerne er usynlige indtil brugeren har accepteret at en anden bruger må følge den. Modsat vil dette kunne demotivere

nogle brugere, hvis de ikke tillader adgang for andre brugere til at tilgå deres resultater, da konkurrenceelementet vil kunne svækkes. For eksempel vil brugeren ikke blive påvirket, hvis andre ikke kan se dens belønninger.

9.1.3 Implementering

Det er valgt at implementere en database til lagring af brugerdata. Denne blev implementeret på en lokal server. En fordel ved at gøre dette er, at det simulerer en endelig database, og foretrækkes at anvende under en udviklingsfase, da eventuelle fejl er lettere at rette. Derudover begrænser en lokal server på nuværende tidspunkt, at app'en ikke kan anvendes på en smartphone, som ifølge det opstillede non-funktionelle krav var hensigten. Det forventes dog at kunne anvendes på en smartphone, hvis databasen implementeres på en ekstern server, idet app'en på nuværende tidspunkt er simuleret i en android-emulator. For at app'en kan implementeres på en ekstern server, skal der foretages ændringer i forhold til den endelige implementering af app'en. En ulempe ved en ekstern server er, at det er mere omstændig at rette eventuelle fejl sammenlignet med en lokal server.

Da det er valgt at implementere databasen på en lokal server er det ikke valgt at kryptere data, da dette ikke ses lige så nødvendigt som, hvis app'en var implementeret på en ekstern server. Det kan dog diskuteres, hvorvidt data skal krypteres af sikkerhedsmæssige årsager. Det valgt at anvende medlemsID'er som identifikation fremfor personnumre i databasen af samme årsag. Det kan dog diskuteres, hvorvidt den enkelte vil anse data som værende personfølsomme, hvorfor disse muligvis burde krypteres.

På baggrund af designafsnittet blev der valgt at implementere en timer til at måle tiden under træning. Det er valgt at implementere denne som en optælling, da det ønskes at brugere skal have mulighed for at fortsætte træningen efter den anbefalede træningstid er opnået. Det kan diskuteres om dette er den mest hensigtsmæssige måde, idet tanken om at implementere et anbefalet træningsniveau er at sikre, at brugere ikke underpræstere eller overpræstere. På nuværende tidspunkt er der ikke implementeret en form for feedback, når brugeren har opnået det anbefalede træningsniveau, hvormed brugeren selv skal holde øje med om dette er opnået. En løsning på dette kunne være at implementere timeren som en nedtælling eller at indføre lyd, der gør brugeren opmærksom når det anbefalede træningsniveau er opnået.

9.1.4 I praksis

Da denne app er en prototype, skal der foretages ændringer og tilføjelser for at muliggøre implementering af denne i praksis. Dette indebærer blandt andet, at der implementeres øvelser passende til KOL-patienter, samt anbefalede træningsniveauer, som er realistiske i forhold til, hvad KOL-patienter med forskellige ABCD-kategoriseringer og helbredstilstande fysisk kan holde til. Dertil skal træningsformer og -typer ligeledes tilpasses efter øvelser og træninger der foretages i forbindelse rehabiliteringsforløbet, så det sikres at KOL-patienter har kendskab til de træningsformer samt -typer der implementeres i app'en.

9.2 Konklusion

9.3 Perspektivering

Da denne app er udviklet som en prototype, skal der foretages overvejelser i forhold til ændringer samt forbedringer inden app'en kan anvendes af KOL-patienter.

På nuværende tidspunkt er serveren lokal, hvorved databasen kun kan tilgås via én computer. For at sundhedspersonalet fra flere rehabiliteringscentre i fremtiden skal kunne tilgå databasen, skal det overvejes at implementere denne som en ekstern server. Derudover skal det overvejes om sundhedspersonalet via app'en eller ved udvikling af en ny app, skal have mulighed for at registrere nye brugere i databasen samt kunne tilgå brugernes resultater via denne.

I forhold til tilpasning af træningsniveau skal der undersøges flere studier om faktorer, der påvirker KOL-patienters helbredstilstand. Dette kunne eksempelvis være at inddrage komorbiditeter, blodtryk, vejrudsigter, om patienterne er blue bloater eller pink puffer. Dette ville medvirke til, at algoritmen for udregning af anbefalet træningsniveau muligvis vil kunne tilpasses den enkelte KOL-patient bedre end på nuværende tidspunkt. Det anbefalede træningsniveau illustreres ved tid, hvortil det skal overvejes, om det vil være mere hensigtsmæssigt at inddrage distance i stedet, eller om begge anbefalinger skal fremgå.

Træningsformer og -typer skal undersøges i separat studie for, hvilke der vil være mest hensigtsmæssig at udføre for patienter med KOL. Dertil skal det overvejes, om sundhedspersonalet skal have mulighed for at tilføje ekstra træningsformer eller øvelser, som patienterne har udført og har kendskab til i forbindelse med rehabiliteringsforløbet.

Til træningen kunne det overvejes at inddrage flere enheder til monitorering af træningen. Dette kunne for eksempel være eksterne måleenheder som pulsur og iltmåling til biologiske målinger, der kan være med til at vejlede patienten i forhold til at opnå mest ud af træningen samt advare patienten ved overanstrengelse. Derudover kunne det overvejes at gøre det muligt at tilkoble træningsmaskiner, såsom sofacykel, kondicykel og løbebånd, så patienter med adgang til disse kan anvende dem sammen med app'en. Dette vil dog kræve, at træningsmaskinerne skal kunne kommunikere med app'en for eksempel med bluetooth.

Resultaterne, opnået ved træning, er på nuværende tidspunkt vist som virtuelle belønninger. Dertil bør det overvejes, om grafisk udvikling eller anden visning vil være mere motiverende for brugeren. Ved grafisk udvikling vil sundhedspersonalet også få et visuelt indblik i, hvordan de enkelte brugeres udvikling forløber. Hertil vil det også forventes, at sundhedspersonalet vil kunne motivere og give besked, hvis patienter har udviklet sig samt, hvis patienterne har været inaktiv i en længere periode.

I forhold til venneliste skal opsætningen af denne overvejes. Det kunne være muligt at rangordne brugere, der følges, efter opnåede belønninger med henblik på motivering. Derudover skal det overvejes, om brugeren skal have notifikationer, når andre venner træner og har opnået belønninger.

Litteratur

- [1] Det Sundhedsvidenskabelige Fakultet på AAU. Studieordningen for bacheloruddannelsen i Sundhedsteknologi. 2014.
- [2] Pernille Hauschildt and Jesper Ravn. *Basisbogen i Medicin og Kirurgi*. 2016.
- [3] Lungeforeningen. Lokalafdelinger og netværk. *Lungeforeningen*, 2016. URL <https://www.lunge.dk/lokalafdelinger-og-netvaerk>.
- [4] Sundhedsstyrelsen. SYGDOMSBYRDEN I DANMARK. 2015.
- [5] WHO. The top 10 causes of death, . URL <http://www.who.int/mediacentre/factsheets/fs310/en/index3.html>.
- [6] Dansk Selskab for Almen Medicin. KOL. 2016. URL <http://vejledninger.dsam.dk/kol/?mode=visKapitel{&}cid=942{&}gotoChapter=942> <http://vejledninger.dsam.dk/kol/?mode=visKapitel{&}cid=951{&}gotoChapter=951>.
- [7] Fernando D. Martinez. Early-Life Origins of Chronic Obstructive Pulmonary Disease. *Asthma and Airway Disease Research Center, University of Arizona, Tucson.*, 2016.
- [8] Sundhedsdatastyrelsen. Borgere med KOL – kontaktforbrug i sundheds-væsenet og medicinforbrug. *Sundhedsdatastyrelsen*, 2016.
- [9] Ejvind Frausing. Kronisk bronkitis. *Lungeforeningen*, 2011. URL <https://www.lunge.dk/kronisk-bronkitis>.
- [10] The Editors of Encyclopædia Britannica. Bronchitis. *Encyclopædia Britannica*, 2016. URL <https://global.britannica.com/science/bronchitis>.
- [11] Healthguidances. Are You A Pink Puffer or A Blue Bloater. 2016. URL <http://www.healthguidances.com/pink-puffer-vs-blue-bloater/>.
- [12] Ejvind Frausing. Emfysem. *Lungeforeningen*, 2011. URL <https://www.lunge.dk/emfysem>.
- [13] John Flaschen-Hansen. Emphysema. *Encyclopædia Britannica*, 2008.
- [14] Statens Institut for Folkesundhed. Kronisk obstruktiv lungesygdom (KOL). *Folkesundhedsrapporten*, 2017.
- [15] Peter Lange. Kronisk obstruktiv lungesygdom. *Sygdomsleksikon*, 2015. URL <http://www.apoteket.dk/Sygdomsleksikon/SygdommeEgenproduktion/Kroniskbronkitis-KOLRygerlunger.aspx>.
- [16] A. Anzueto. Impact of exacerbations on COPD. *European Respiratory Review*, 2010. doi: 10.1183/09059180.00002610.

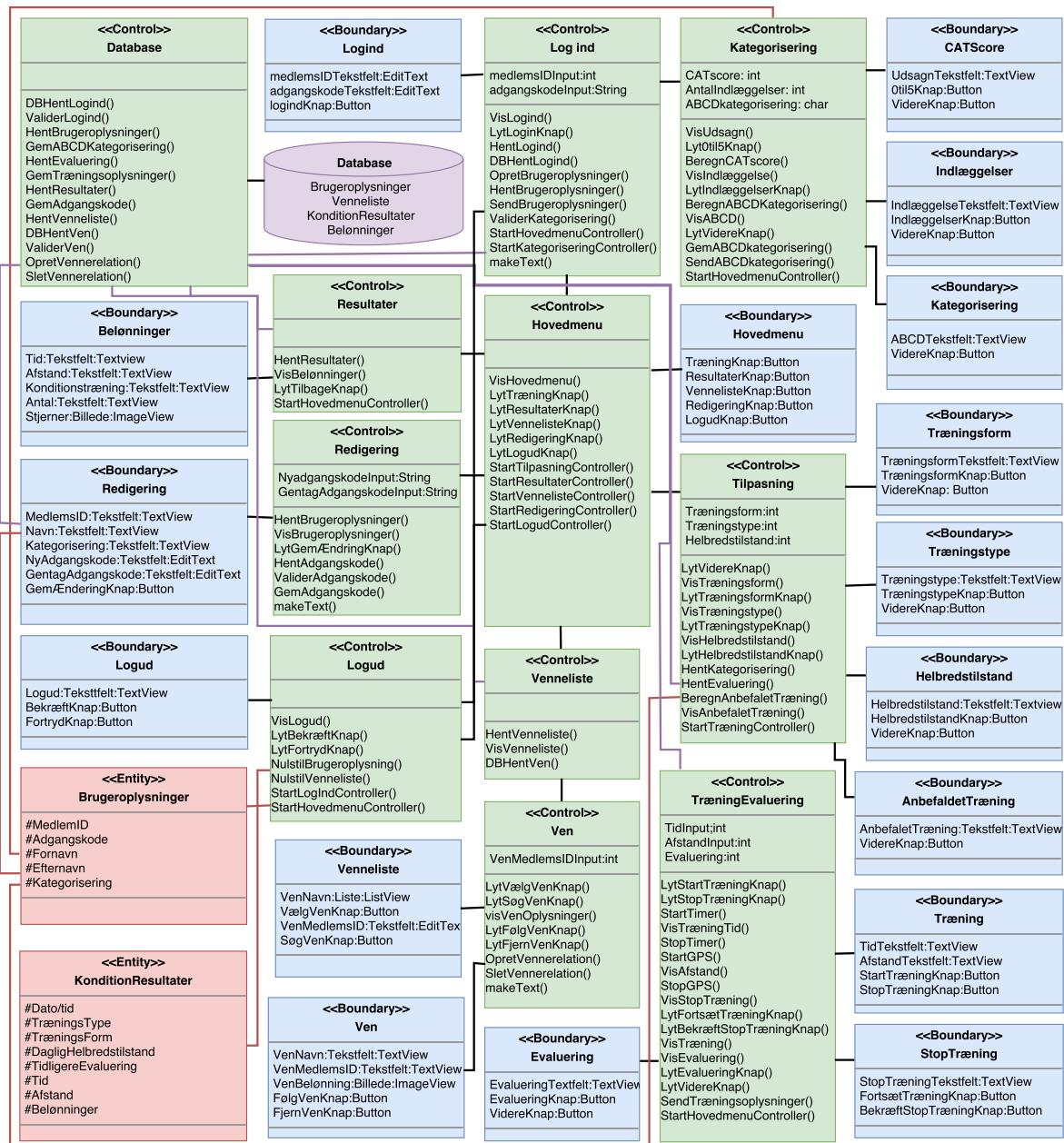
- [17] McCarthy B. et al. Pulmonary rehabilitation for chronic obstructive pulmonary disease. *Cochrane Library*, 2015.
- [18] De specialeansvarlige lungemedicinere i Storstrømmens Sygehus. KOL, behandling, udredning. *Sundhed.dk*, 2013. URL <https://www.sundhed.dk/sundhedsfaglig/information-til-praksis/sjaelland/patientforloeb/forloebsbeskrivelser/r-luftveje/kol/>.
- [19] Sundhedsdatastyrelsen. KOL Flere borgere med KOL i medicinsk behandling. 2015.
- [20] Elisabet Helle, Kari Bruusgaard, and Et. Al. Exercise maintenance: COPD patients' perception and perspectives on elements of success in sustaining long-term exercise. *Physiotherapy Theory and Practice*, pages 206–220, 2012. doi: 10.3109/09593985.2011.587502.
- [21] Veronika Williams, Jonathan Price, and Et.al. Using a mobile health application to support self-management in COPD. 2014. URL <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4073724/pdf/bjgpjul2014-64-624-e392.pdf>.
- [22] Ejvind Frausing. Rehabilitering. *Lungeforeningen*, 2011.
- [23] J. M. Habraken. Health-related quality of life and functional status in end-stage COPD: a longitudinal study. *European Respiratory journal*, 2011.
- [24] Sundhedsstyrelsen. Anbefalinger for tværsektorielle forløb for mennesker med KOL. *Sundhedsstyrelsen*, 2015. URL <https://www.sst.dk/da/udgivelser/2015/{~}/media/8365DCEC9BB240A0BD6387A81CBDDB49.ashx>.
- [25] Clarie and others Egan. Short term and long term effects of pulmonary rehabilitation on physical activity in COPD. *Respiratory Medicine*, 2012.
- [26] Maria K. et. al. Beachamp. A novel approach to long-term respiratory care: Results of a community-based post-rehabilitation maintenance program in COPD. *Respiratory Medicine*, 2013.
- [27] Paolo Zanaboni. Long-term exercise maintenance in COPD via telerehabilitation: a two-year pilot study. *Journal of Telemedicine and Telecare*, 2017. URL <http://journals.sagepub.com/doi/pdf/10.1177/1357633X15625545>.
- [28] T et. al. Ringbaek. Rehabilitation in COPD: the long-term effect of a supervised 7-week program succeeded by a self-monitored walking program. *Pulmonary Rehabilitation Research Group*, 2008. URL <https://www.ncbi.nlm.nih.gov/pubmed/18539720>.
- [29] WHO. Telehealth, . URL <http://www.who.int/sustainable-development/health-sector/strategies/telehealth/en/>.
- [30] WHO. WORLD REPORT ON DISABILITY. 2017. URL [http://www.who.int/disabilities/world{_\]report/2011/report.pdf?ua=1](http://www.who.int/disabilities/world{_]report/2011/report.pdf?ua=1).
- [31] Healthcare Denmark. Aidcube. *Healthcare Denmark*. URL <http://healthcaredenmark.dk/profiles/aidcube.aspx>.

- [32] Stoyan Stefanov and Kumar C. Sharma. Object-Oriented JavaScript. *Packt Publishing*, pages 32–38, 2013.
- [33] Dathan Brahma and Ramnath Sarnath. Object-Oriented Analysis, Design and Implementation. *Springer*, 2015. doi: 978-3-319-24280-4.
- [34] Martin Fowler. *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. Third edit edition, 2004. ISBN 978-0321193681.
- [35] Laurie Williams. An Introduction to the Unified Modeling Language. 2004. URL <http://agile.csc.ncsu.edu/SEMaterials/UMLOverview.pdf>.
- [36] Jim Arlow and Ila Neustadt. *UML and the Unified Process*. 2002. ISBN 0 201 77060 1.
- [37] Anders Gade. Motivation, belønning og afhængighed. *Københavns Universitet*, 2007.
- [38] Elizabeth and Tricomi and Samantha DePasque. *The Role of Feedback in Learning and Motivation*. 2016. ISBN 978-1-78635-474-7.
- [39] Sundhedsdatastyrelsen. Vejledning om informationssikkerhed i sundhedsvæsenet. *Sundhedsdatastyrelsen*, 2016.
- [40] Rådet for digital sikkerhed. Sikre adgangskoder. *Rådet for digital sikkerhed*, 2015. URL <http://www.digitalsikkerhed.dk/sikre-adgangskoder/>.
- [41] Dempsey Chang and Et.al. Gestalt Theory in Visual Screen Design – A New Look at an Old Subject. *Monash University*, 2002.
- [42] Xavier Ferré, Natalia Juristo, and Et.al. Usability Basics for Software Developers. 2001.

Bilag A

Bilag

A.1 Bilag A



Figur A.1: Designklasser