
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
Kapitel 6 Systemdesign	36
6.1 Designafgrænsning	36
6.2 Objektorienteret design	36
6.3 Design af database	61
6.3.1 ER-diagram	61
6.3.2 Schema	62
Kapitel 7 Implementering	63
7.0.1 Implementering af objektorienteret design	63
7.0.2 Implementering af database	63
Kapitel 8 Test	64

8.1 Database	64
8.2 Log ind	65
8.3 Kategorisering	65
Kapitel 9 Syntese	72
Litteratur	73
Bilag A Bilag	77
A.1 Bilag A	77

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?

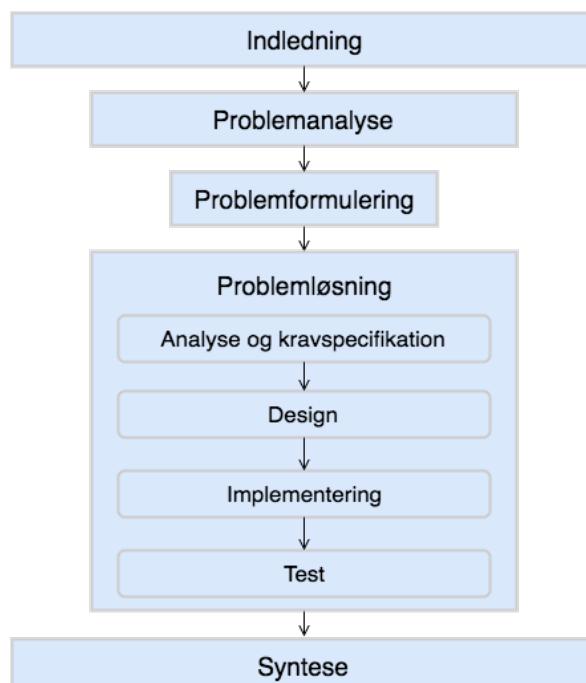
Kapitel 2

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

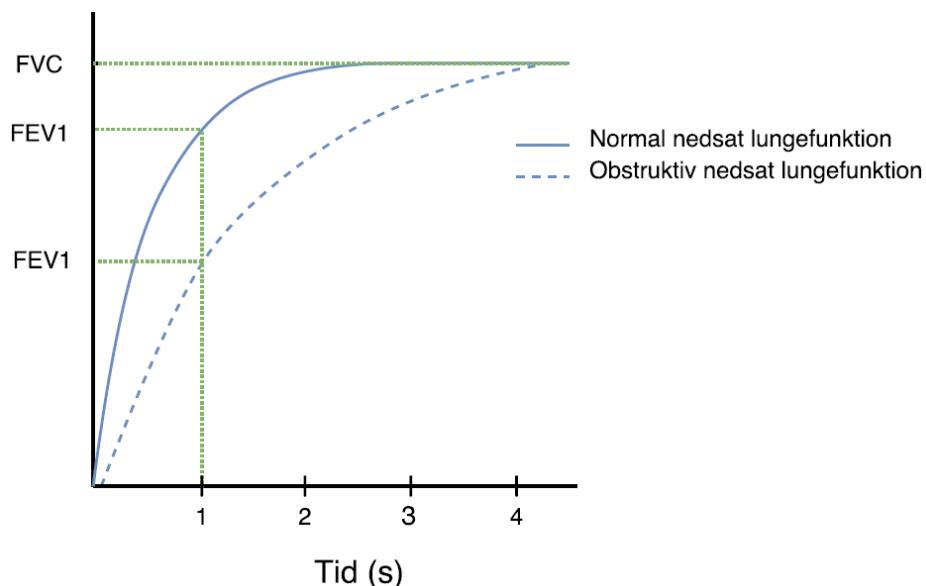
KOL udvikles over mange år, dog bemærkes sygdommen ofte ikke før lungefunktionen er markant nedsat. Dette betyder, at KOL og dens symptomer som regel først kommer til udtryk efter 50 årsalderen [15]. Dette kan i praksis betyde, at patienter først op søger en læge, når deres lungefunktion er halveret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

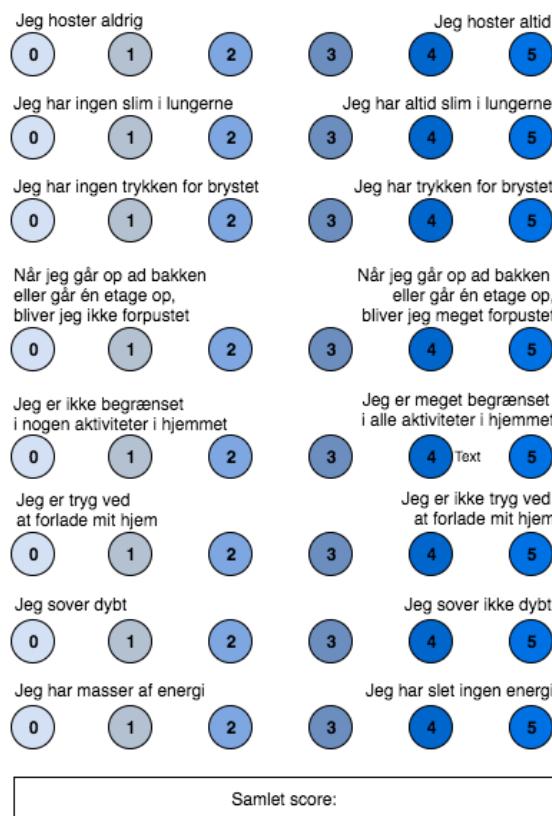
Sværhedsgraden af KOL vurderes på baggrund af patienters symptomer, egne erfaringer og livskvalitet. Denne vurderes ud fra Medical Research Council åndenødsskala (MRC) eller Chronic obstructive pulmonary disease Assessment Test (CAT). Patienter kan efterfølgende inddeltes i klassifikationer med udgangspunkt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

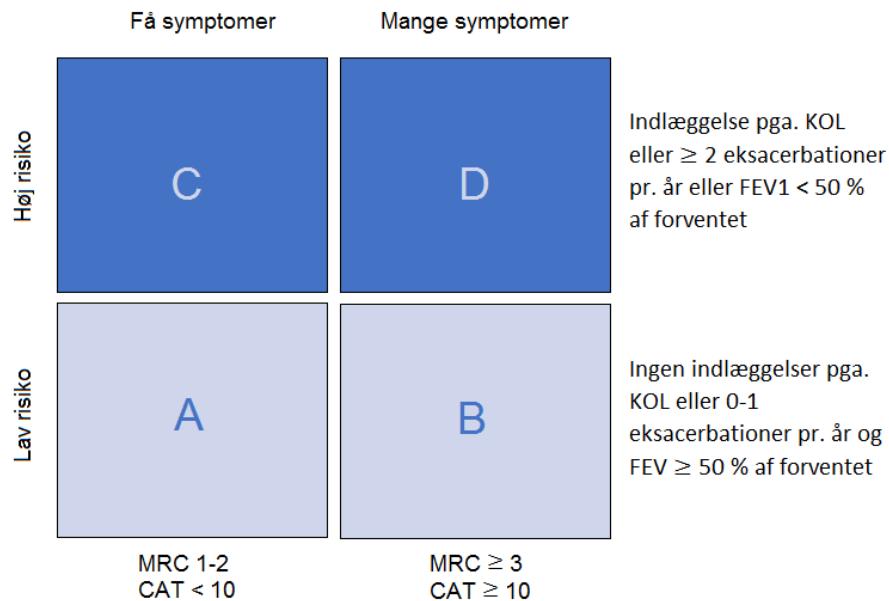
Tabel 3.1: MRC er en skala fra 1 til 5. Patienter, der oplever åndenød ved meget anstrengelse vurderes til 1, mens patienter, der oplever åndenød ved lav aktivitet vurderes til 5 på MRC-skalaen. Revideret[2].

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].

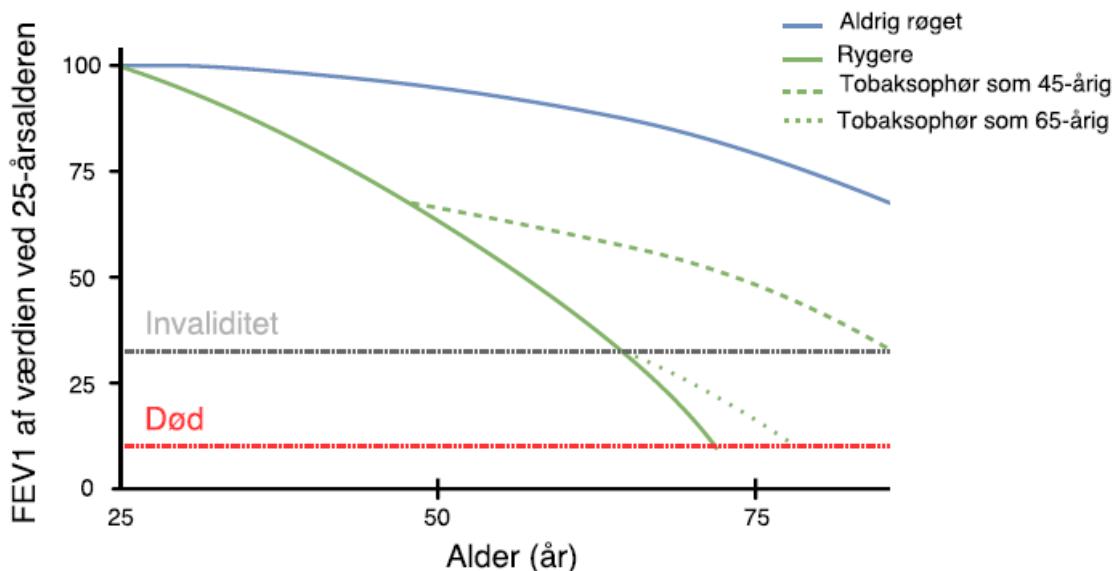
3.1.3 Behandling

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]. Fordel 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.

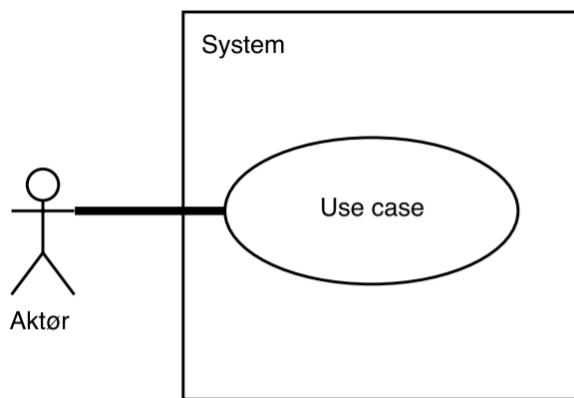
Objektorienteret programmering består af tre grundprincipper, herunder indkapsling, nedarvning og polymorfi. Indkapsling er en illustration af, at objekter både indeholder egenskaber og metoder. Egenskaber opbevarer data, mens metoder anvendes til at behandle data. Indkapsling kan både have synlige og 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.



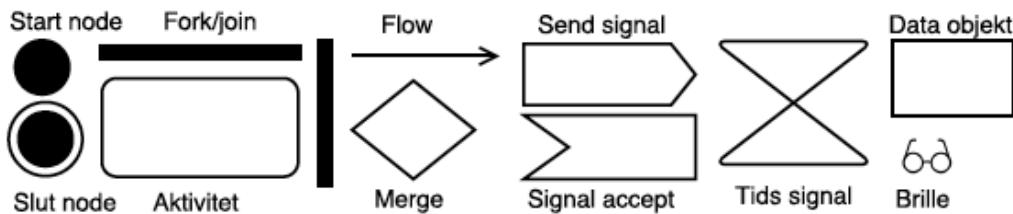
Figur 4.1: Simpelt use case diagram.

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

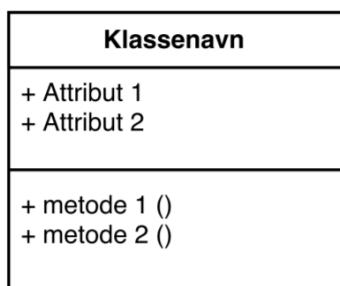
beskrives yderligere i et nyt aktivitetsdiagram.[34] Symboler, der kan anvendes inden for aktivitetsdiagrammer, fremgår af figur 4.2.



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

Klassediagrammer

Klassediagrammer anvendes som redskab til at beskrive strukturen i et givent 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.

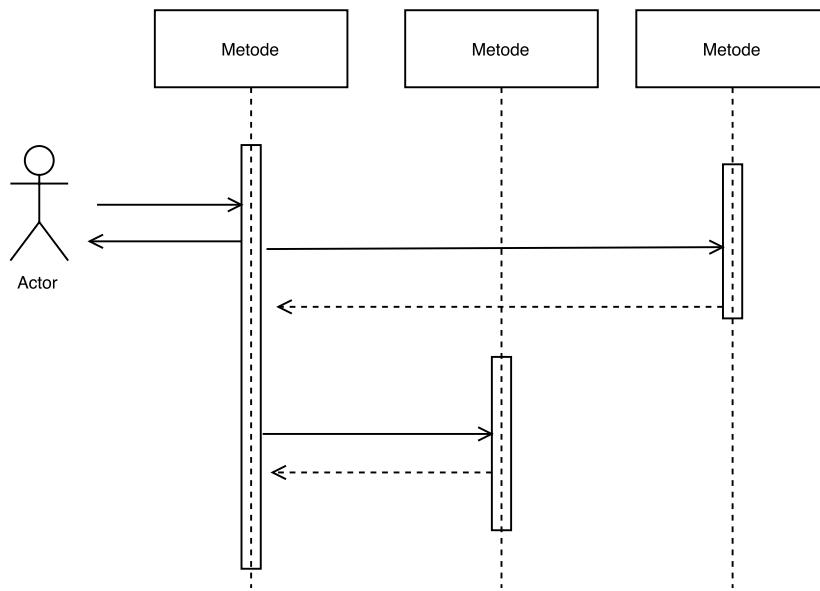
Klassediagrammer kan inddeltes i tre stereotyper, herunder entity, boundary og control, hvilket anvendes til at identificere klasser i analyse og tidlig designfase [?].

- *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].

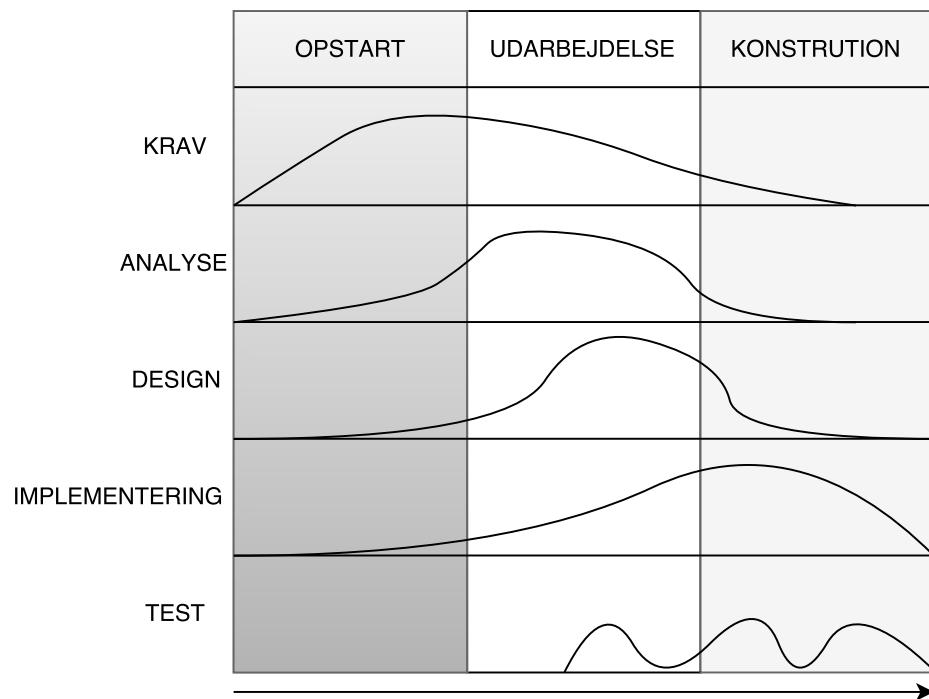
Af figur 4.4 fremgår det, at hver entitet i et use case diagram har en kolonne i sekvensdiagrammet. Den vertikale retning beskriver tiden, og de horisontale linjer illustrerer funktionaliteter. For enden af de horisontale linjer er der navngivet metoden på entiteten.[33]

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 kalde 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 vejentrækningsøvelser, hvilket alle har symptomreducerende effekt, jf. afsnit 3.2.

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 og distance 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 tidligere evalueringer fra lignende træning. 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, derudover kan app'en tilkobles kompatible måleenheder. Monitoreringen er med til at vejlede patienten til at følge det valgte træningsniveau. Kompatible måleenheder er for eksempel et pulsur, som kan støtte patienten i forhold til at kontrollere, at patienten ikke overanstrenges.

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 følge sin egen udvikling via app'en. Desuden skal app'en informere, hvis KOL-patienter ikke har udført træning med app'en i længere tid. For at øge motivationen hos KOL-patienter skal de derudover kunne opnå virtuelle belønninger ved at udføre gentagne eller forskellige træningsformer.[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

- Sundhedspersonalet skal kunne oprette brugere i en database
Dette er nødvendigt for, at brugere kan anvende app'en
- Brugere skal kunne log ind med et medlemsID og adgangskode
Dette er nødvendigt for at tilgå og sikre, at brugere har deltaget i et rehabiliteringsforløb samt adskille brugerens data
- Systemet skal kunne kategorisere brugere i ABCD
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 håndtere målinger fra kompatible måleenheder
Dette er nødvendigt for at muliggøre måling af biologiske målinger under træning
- Brugere skal kunne evaluere hver træning
Dette er nødvendigt for at tilpasse træningen efter den enkelte bruger
- Systemet skal kunne gemme og hente data i en database
Dette er nødvendigt for, at brugere kan tilgå brugerdata
- Systemet skal kunne sende notifikationer og 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 brugerens motivation
- Brugere skal kunne redigere deres adgangskode
Dette er nødvendigt for, at brugere kan ændre adgangskode
- Brugere skal kunne log ud
Dette er nødvendigt for sikre brugerens individuelle data

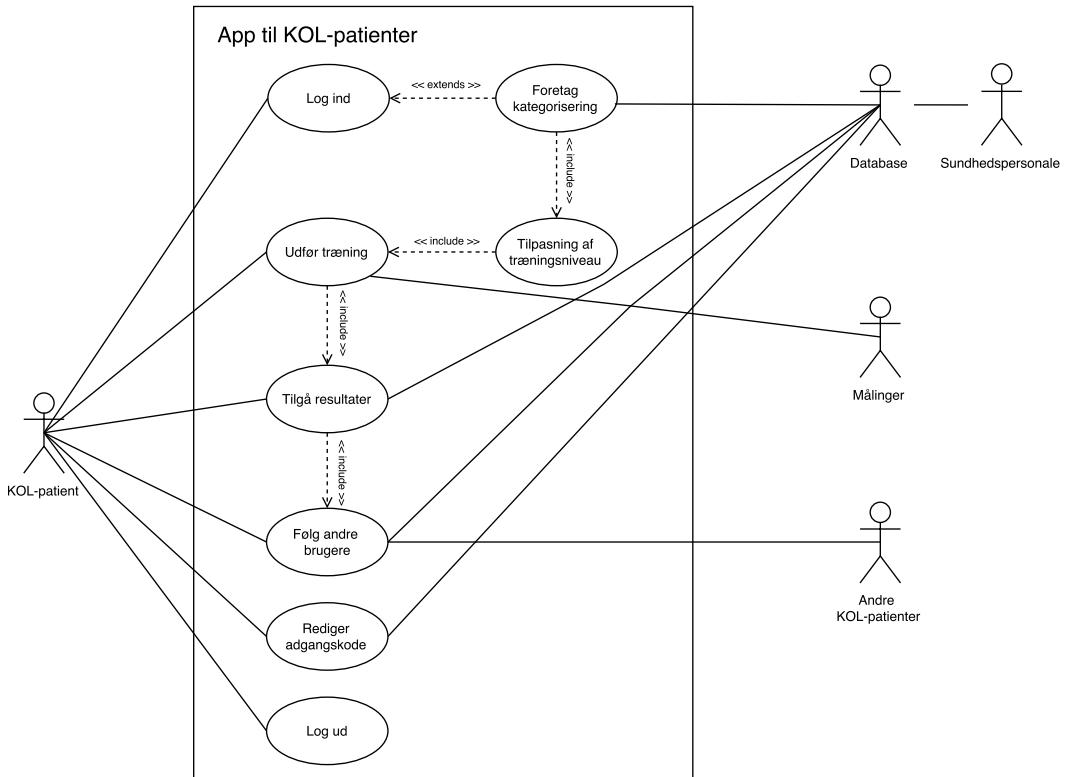
Non-funktionelle krav

- Systemet skal visualiseres på en smartphone eller tablet med android og bluetooth
- 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. Målinger, 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, følge andre brugere, 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 træning. Under træningen kan kompatible måleenheder tilkobles systemet, således målinger kan opsamles.

Efter træningen skal brugere evaluere træningen, og de kan efterfølgende se deres resultater. Evaluering og resultater fra træningen gemmes efterfølgende i en database. Brugere kan *Tilgå resultater*, der vises som daglige og ugentlige træningsresultater samt individuelle belønninger.

Brugere kan via *Følg andre brugere* tilgå andres KOL-patienters 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, idet de ved registrering får en randomiseret adgangskode udleveret. Hvis der foretages ændringer gemmes disse efterfølgende i databasen.

Brugeren kan *Log ud* af app'en.

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, træning, tilpasning af træningsniveau, resultater, venneliste, redigering af adgangskode og log ud. Nogle af funktionaliteterne er beskrevet i aktivitetsdiagrammer, som er opdelt i bruger, systemet og database. Hver gang et aktivitetsdiagram starter antages det at brugeren har trykket på den gældende aktivitet. Efter hver endt aktivitet vender brugeren tilbage til en hovedmenu. Brugeren kan altid gå tilbage i systemet.

Registrering

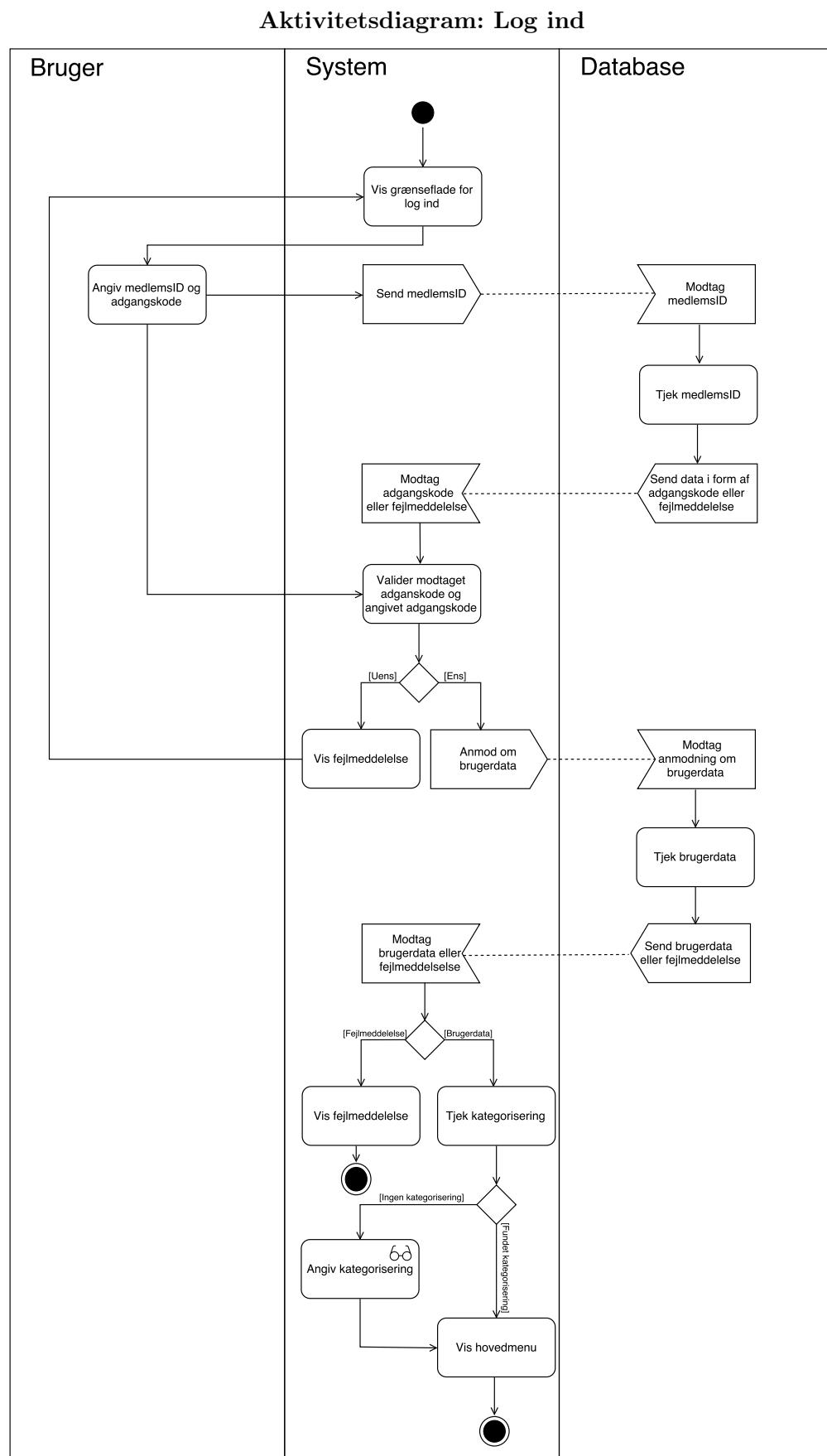
Inden KOL-patienter kan anvende app'en skal de registreres som brugere af systemet. Dette skal foregå i forbindelse med rehabiliteringsforløbet, hvor sundhedspersonale opretter patienterne i databasen. Patienterne får tilknyttet et medlemsID og en randomiseret adgangskode. Dette er vigtigt for, at reducere risikoen for misbrug af personlige oplysninger, da uvedkommende ellers kan have mulighed for at tilgå informationerne via netadgang eller enheden [39]. MedlemsID'et skal bestå af tal, eksempelvis *01170301*, 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 foreslægt 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 vil her angive log ind-information, der vil tillade adgang til information i form af private oplysninger og tidligere resultater, tilknyttet den givne bruger. Aktiviteterne for log ind fremgår af figur 5.2.

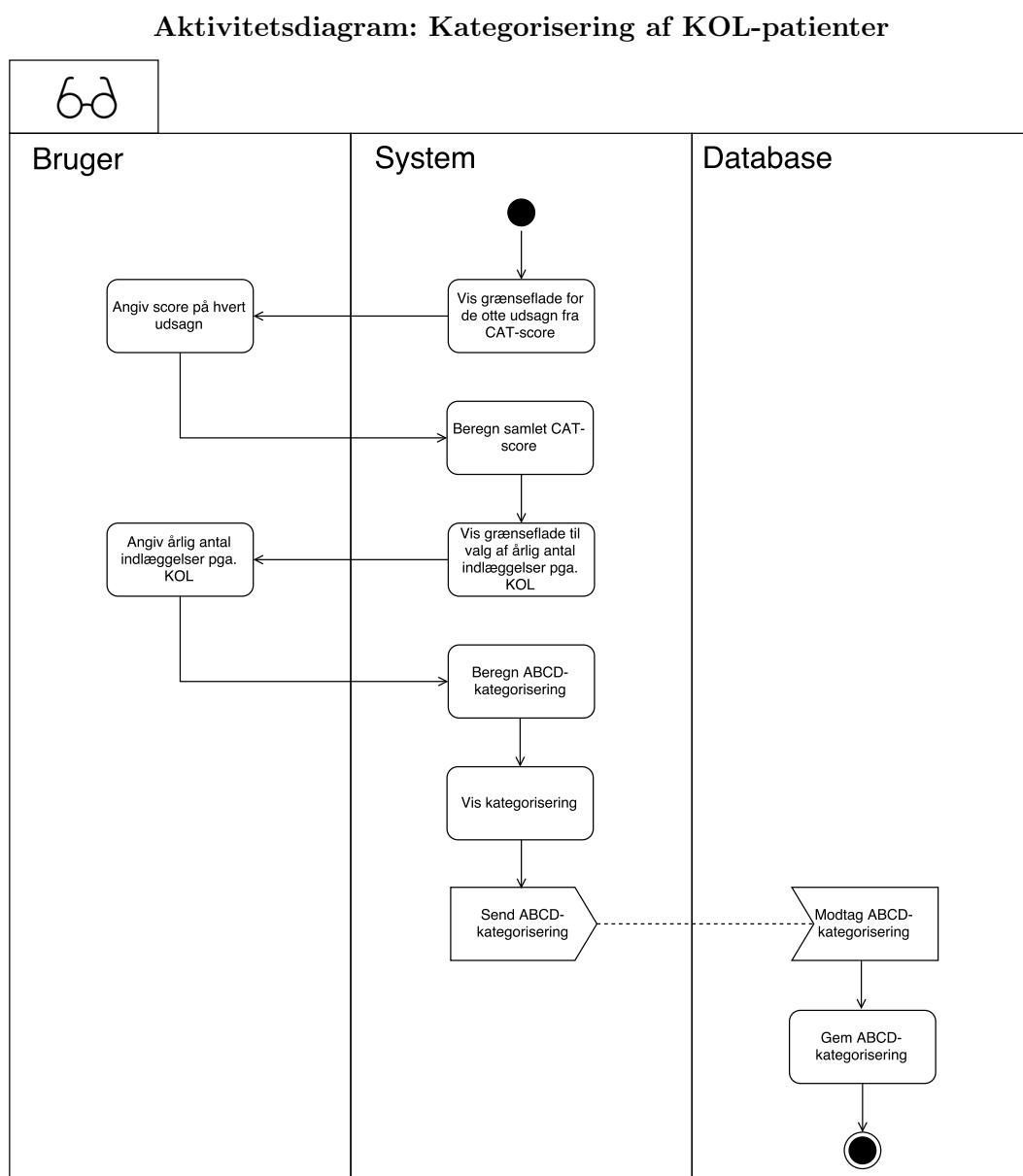


Figur 5.2: Aktivitetsdiagram for log ind. Kategorisering af KOL uddybes af figur 5.3

En grænseflade for log ind er det første brugeren henvises til, når app'en åbnes. Brugeren kan i denne grænseflade angive sit medlemsID samt adgangskode, hvorefter systemet tilsender det indtastede medlemsID til databasen. Databasen returnerer hertil den passende adgangskode, hvis medlemsID'et findes i databasen. Genkendes medlemsID'et ikke, returneres en fejlmeddeelse, hvorefter brugeren henvises tilbage til at indtaste log ind-informationerne igen. Hvis databasen returnerer den tilhørende adgangskode, valideres den indtastede samt den modtaget adgangskode. Stemmer disse adgangskoder ikke overens, sendes igen en fejlmeddeelse, hvorefter brugeren bedes indtaste informationerne igen. Har brugeren glemt sin adgangskode bedes de kontakte sundhedspersonalet. Er de to adgangskoder derimod ens, anmoder systemet databasen om brugerdata, der omfatter brugeroplysninger, resultater samt vennelise. Mislykkedes dette, sendes og vises en fejlmeddeelse. Hvis brugerdataen derimod hentes, tjekker systemet om brugeren har fået sin kategorisering beregnet. 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 således at sikre patienter får en træning tilpasset til deres niveau. Kategoriseringen inddeler brugerne i A, B, C eller D, som beskrevet i afsnit 3.1.2. Af figur 5.3 ses aktivitetsdiagrammet for kategoriseringen.

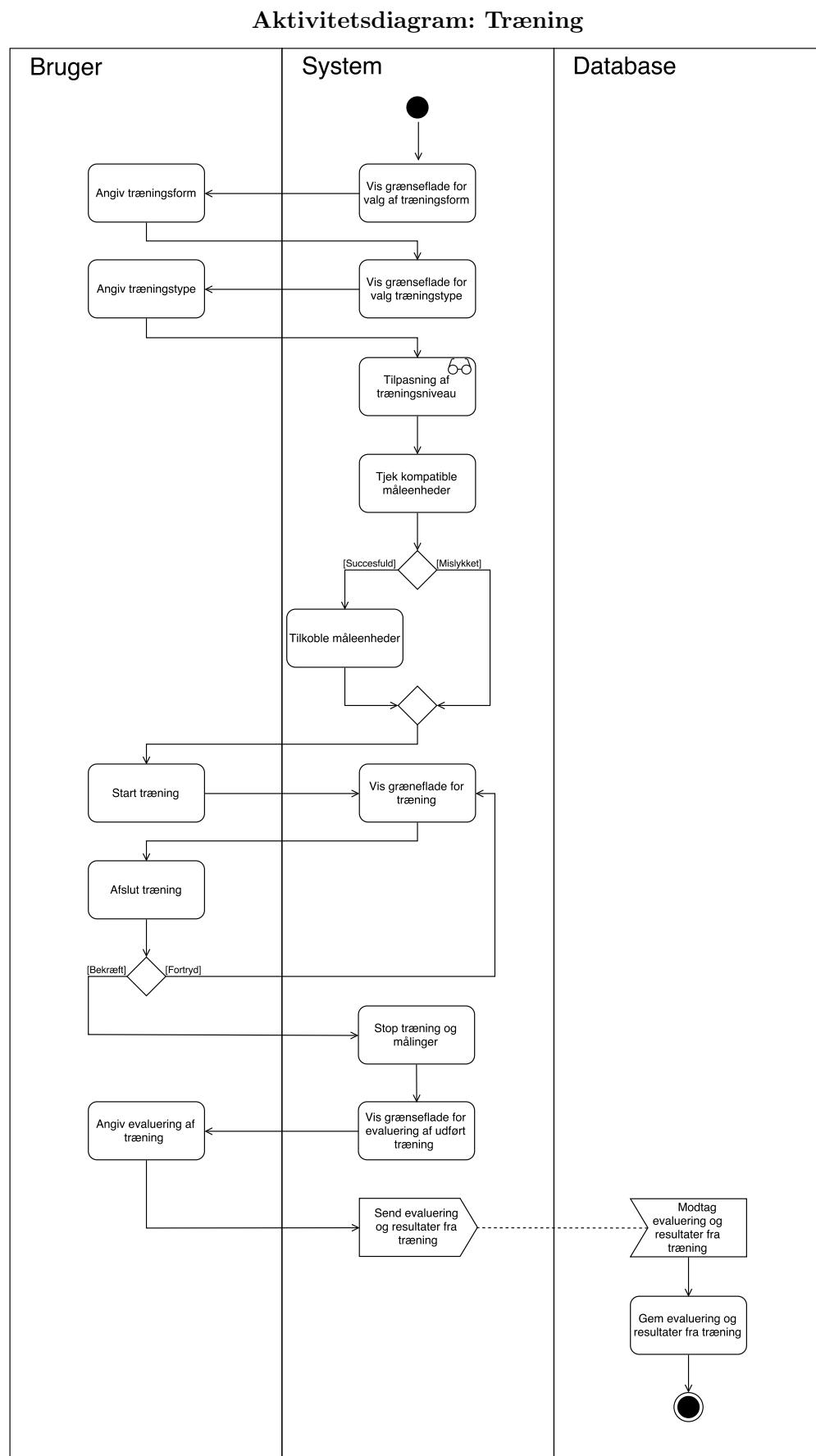


Figur 5.3: Aktivitetsdiagram for kategorisering af KOL-patienter.

Systemet starter med at vise en grænseflade for de otte udsagn der udgør CAT, jf. figur 3.2. Til hvert af udsagnene angiver brugeren således 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, og systemet viser efterfølgende brugerens kategorisering som A, B, C eller D. Kategoriseringen sendes og gemmes i en database.

Træning

Brugeren har mulighed for at foretage træninger baseret på forskellige træningsformer og træningstyper. Derudover skal træningen kunne tilpasses individuelt under hver enkelt træningssession, samt tilkoble kompatible måleenheder for at opnå en vejledende træning. Aktivitetsdiagrammet over træningen fremgår af figur 5.4.



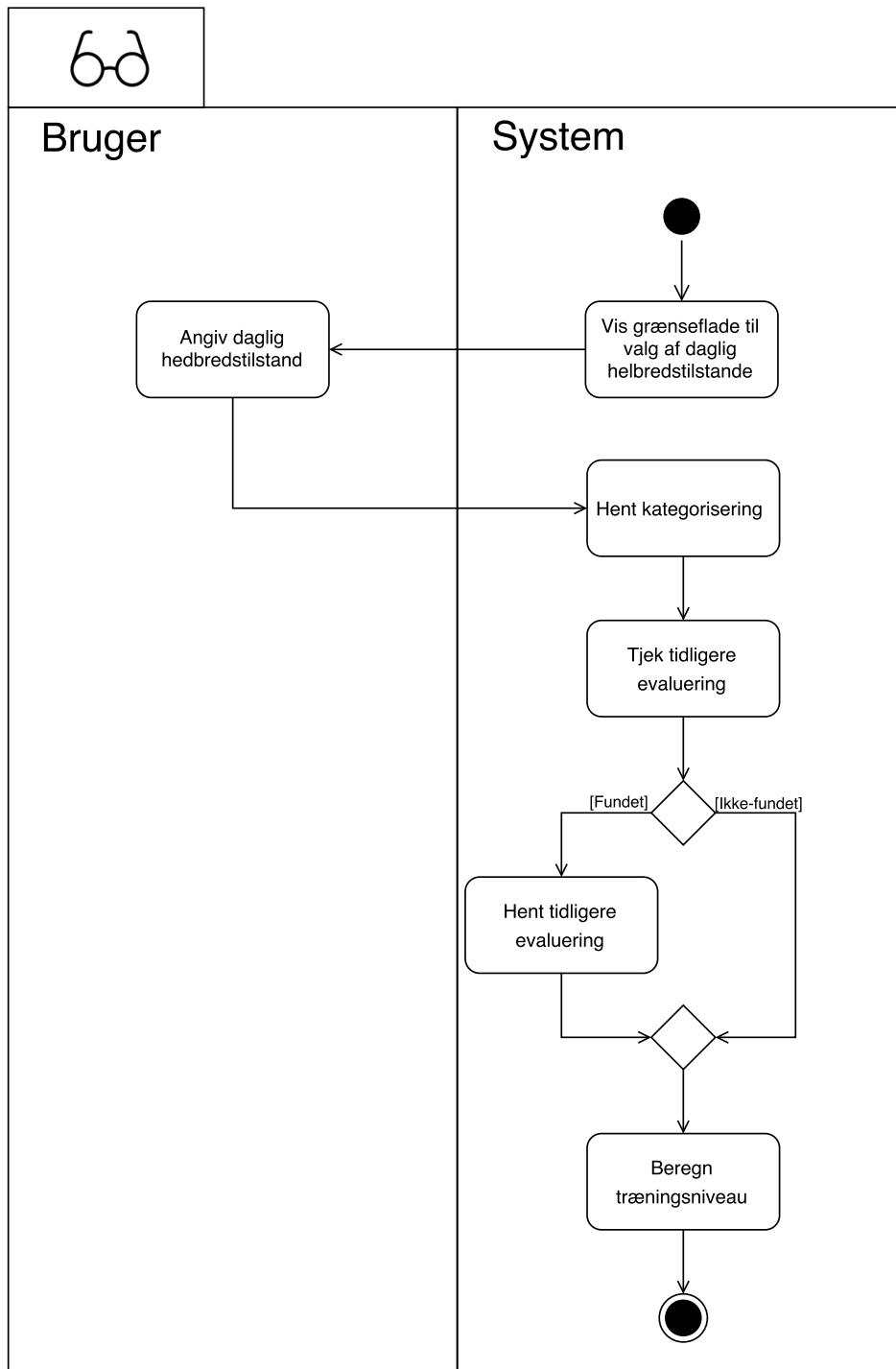
Figur 5.4: Aktivitetsdiagram over træning. Tilpasning af træningsniveau uddybes af figur 5.5.

Før selve træningen påbegyndes, skal brugeren angive den ønskede træningsform, herunder konditions-, styrketræning eller vejrtrækningsøvelser. Ud fra den valgte træningsform skal brugeren angive træningstype, eksempelvis kan der ved valg af konditionstræning vælges gå, løbe eller cykle. Træningsniveau skal efterfølgende tilpasses. Tilpasning af træningsniveauet er yderligere beskrevet af figur 5.5. Når systemet har tilpasset træning, tjekker systemet automatisk om der er kompatible måleenheder, hvis dette er tilfældet tilkobles disse. Ellers kan træningen påbegyndes uden. Brugeren starter herefter træningen og grænsefladen for denne vises. Under træningen vil systemet kontinuert vise træningen og målinger, der foretages. Brugeren kan til en hver tid vælge at afslutte træningen, dog skal denne handling bekræftes i tilfælde af, at brugeren ved en fejl angiver, at træningen skal stoppes. Ved bekræftelse stopper systemet træningen og afventer, at brugeren giver en evaluering. Efter evalueringen sendes evaluering og træningsresultater til en database, hvor det gemmes.

Tilpasning af træningsniveau

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.

For at systemet kan tilpasse træningsniveauet vises grænsefladen for valg af daglig helbredstilstand, hvortil brugeren skal angive sin helbredstilstand. Systemet henter kategorisering og tjekker om der er tidligere evalueringer for den valgte træningsform og -type. Hvis brugeren ikke har angivet tidligere evalueringer bestemmes niveauet ud fra resterende parametre. Hvis de findes hentes de tidligere evalueringer. Tilpasningen af træningsniveau beregnes efterfølgende. Et simpel eksempel på denne beregning fremgår af tabel 5.1. Tabellen beskriver hvordan

en algoritme, ville regulere træningsniveauet, således der tages højde for den enkelte bruger.

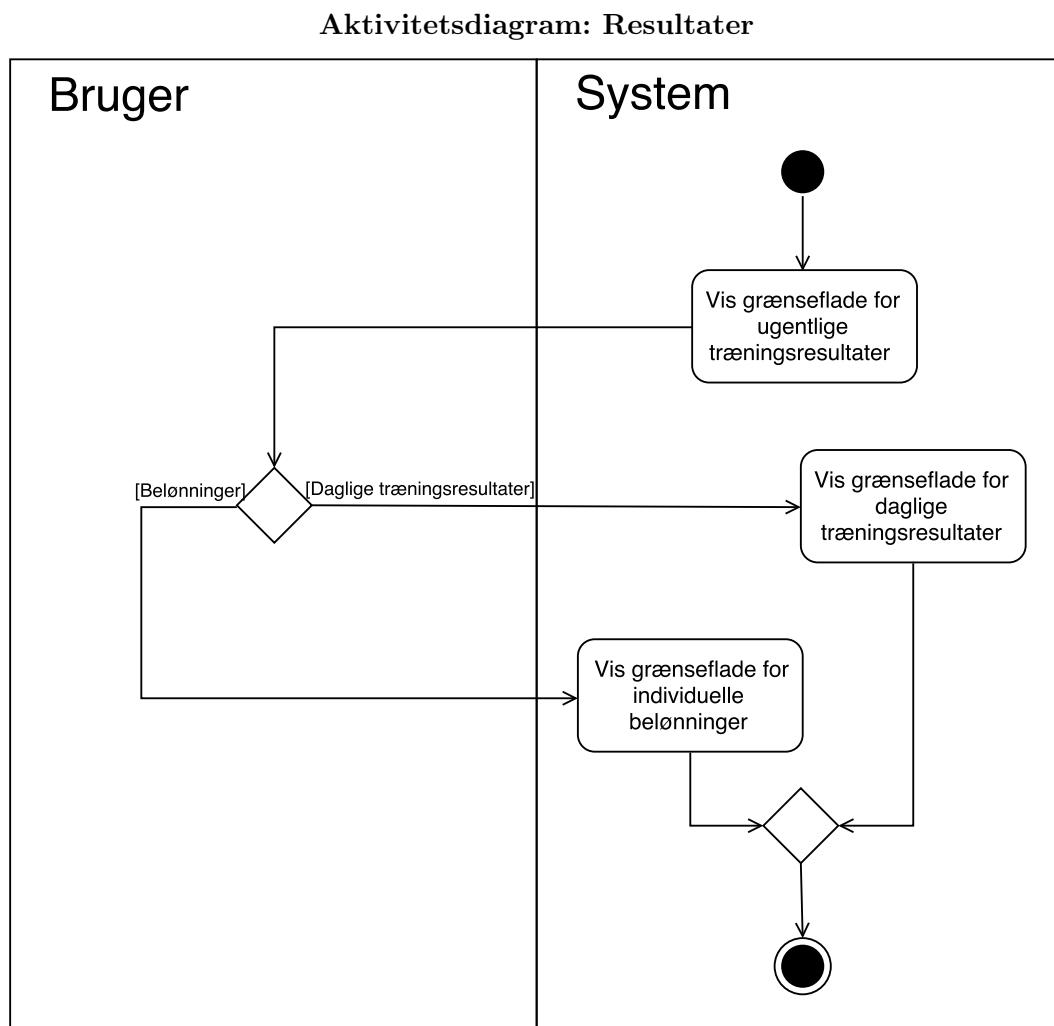
Simpel beslutningstabel											
Kategorisering	A			B			C		D		
Træningsform	Konditionstræning			Styrketræning			Vejtrækningsøvelser				
Træningstype	Gå	Løbe	Cykle	Type 1	Type 2	Type 3	Type 1	Type 2	Type 3		
Daglig Helbredstilstand	1: Meget dårligt		2: Dårligt		3: Moderat		4: Godt		5: Meget godt		
Træningsniveau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

Tabel 5.1: Beslutningstabel 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 tilpasset et træningsniveau. Hvis brugeren tidligere har evaluert en træningen inden for konditration 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 beslutningstabel for, hvorledes et træningssæt 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. På denne måde er det muligt for brugeren at få et overblik over udviklingen samt udførte træninger. Aktivitetsdiagrammet over resultater fremgår af figur 5.6.

*Figur 5.6: Aktivitetsdiagram over resultater.*

Under resultater er det muligt for brugeren at følge sine ugentlige træningsresultater. Derfra har brugeren mulighed for at vælge daglige træningsresultater og tilgå 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						
	★	★★	★★★	★★★★	★★★★★	★★★★★★
Afstand (km)	2	4	6	8	9	10
Tid (min)	5	10	15	20	25	30
Træning (antal)	5	10	15	20	25	30
Uger med træning (antal)	2	5	10	15	20	25
Konditionstræning (antal)	2	4	6	8	9	10
Styrketræning (antal)	2	4	6	8	9	10
Vejrtrækningsovelse (antal)	2	4	6	8	9	10

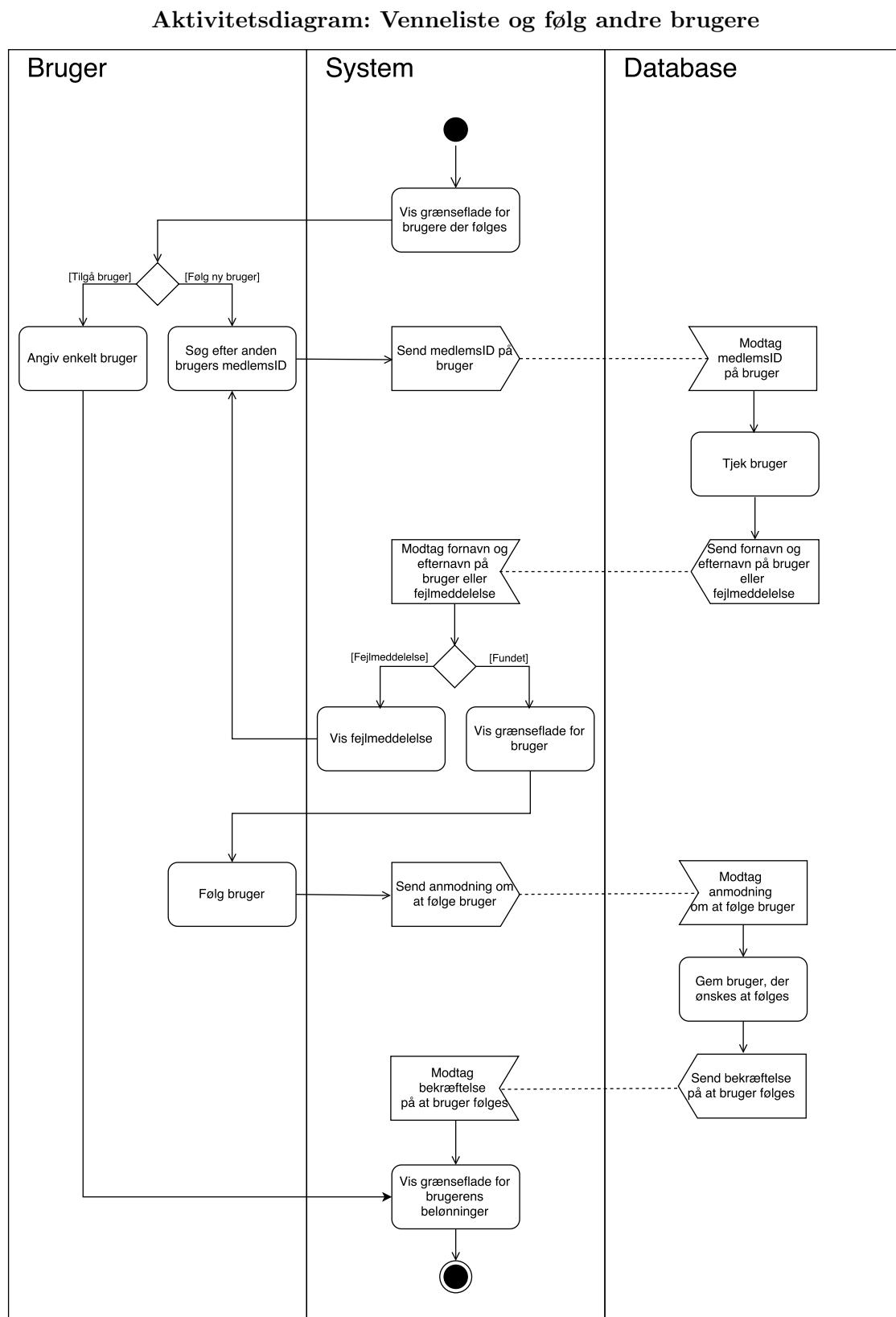
Tabel 5.2: Eksempel på belønninger opnået ved træning inden for forskellige kategorier.

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 og følg andre brugere

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 venneliste og følg andre brugere .



Figur 5.7: Aktivitetsdiagram for venneliste og følg andre brugere.

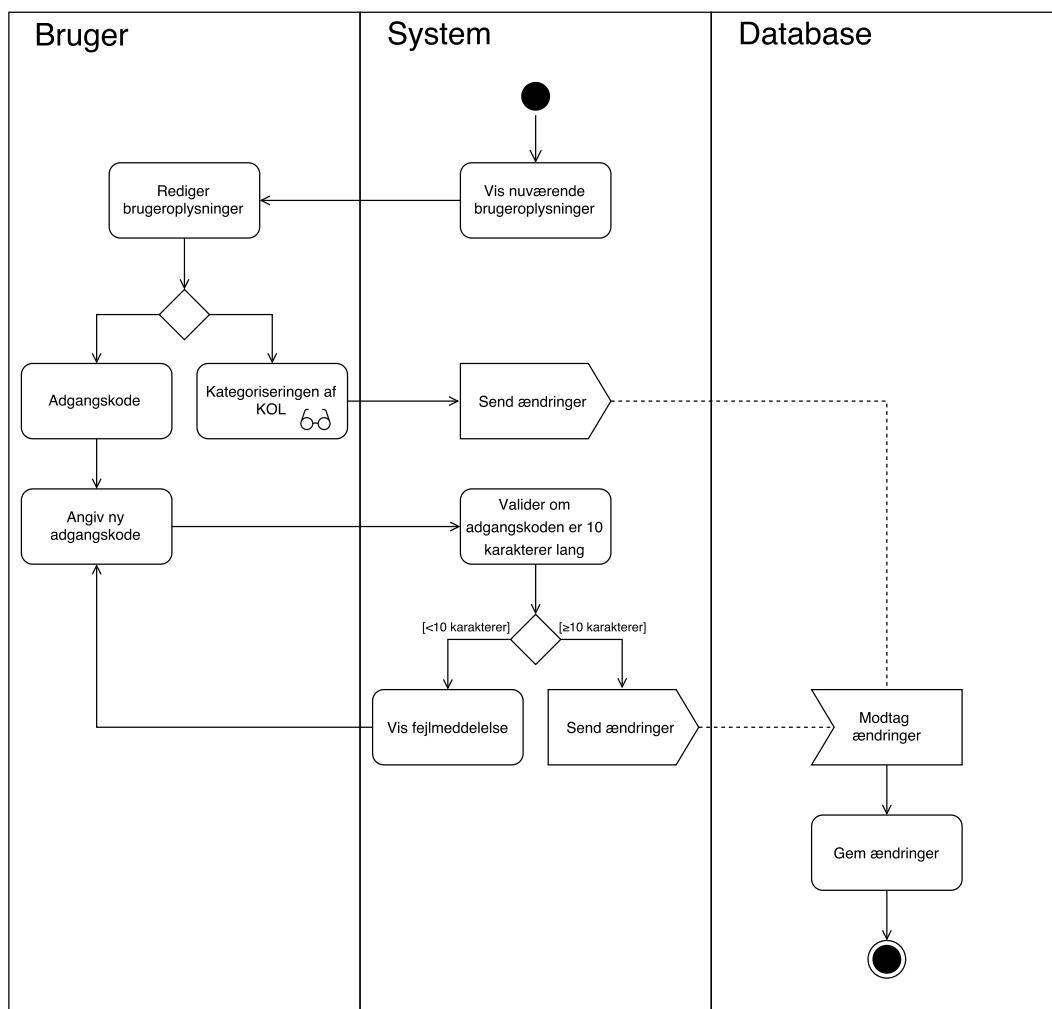
Systemet viser en grænseflade for vennelisten, der indeholder en oversigt over andre brugere, som den individuelle bruger følger. Brugeren kan vælge at følge en ny bruger eller tilgå en

bruger den allerede følger. Ønsker brugeren at følge en ny, skal brugeren angive MedlemsID på denne. Systemet sender det indtastede MedlemsID til databasen, som tjekker 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, hvorefter en ny bruger kan følges. Ønskes dette, sender systemet en anmodning om at følge brugeren. Databasen gemmer brugeren, der ønskes at følges, og sender en bekræftelse eller ved angivet ven viser systemet grænsefladen for den valgte eller tilføjede 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 brugeroplysninger



Figur 5.8: Aktivitetsdiagram for redigering af adgangskode.

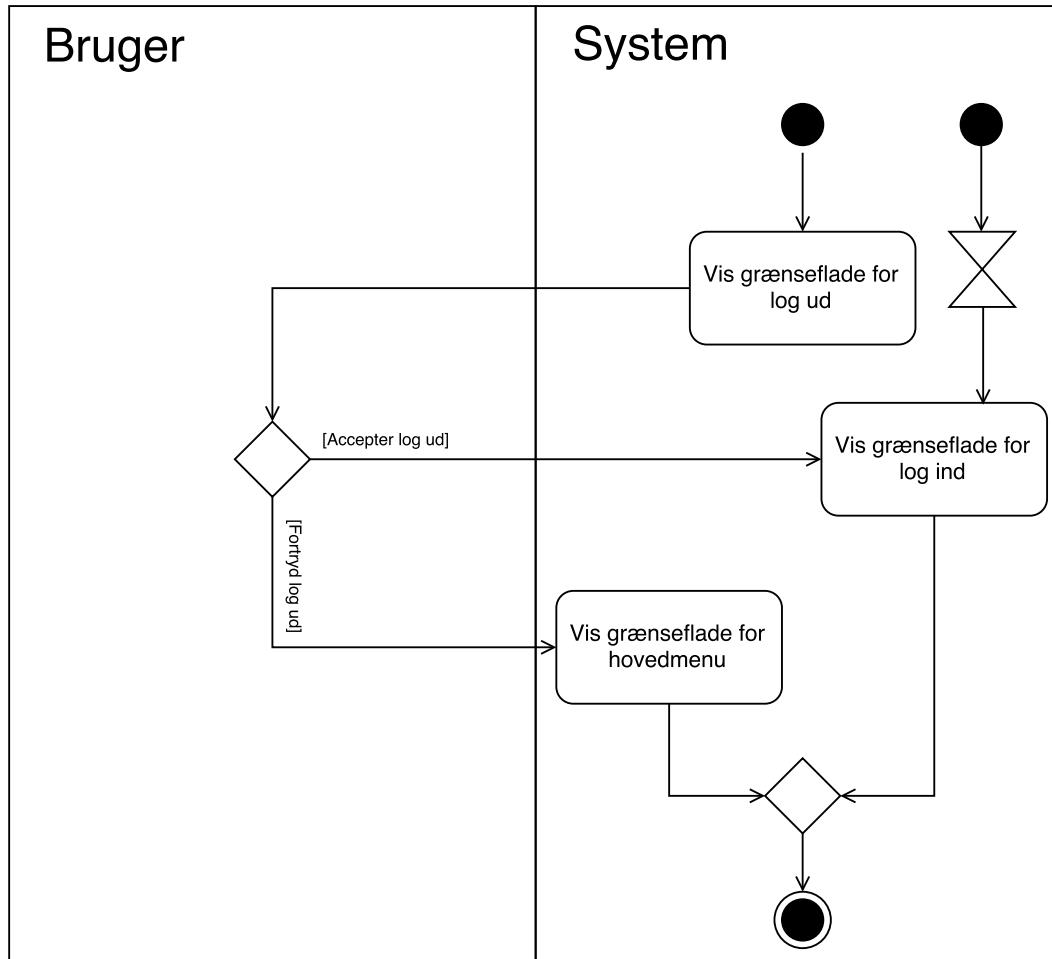
Det skal være muligt for brugeren at ændre adgangskode, da brugeren ved oprettelse får tildelt en randomiseret adgangskode. Dertil kan adgangskoden blive personlig for brugeren, hvilket angivet vil gøre det nemmere for brugeren at huske. Brugeren kan ændre adgangskoden

vælge brugeroplysninger fra hovedmenuen, hvorfra muligheden for at ændre adgangskode fremkommer. Dette opstiller en ny grænseflade hvor brugeren kan angive en ny adgangskode. 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 anbefalder 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 sig 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. Har brugeren været inaktiv i længere tid logger systemet ud automatisk og log ind skærmen vises.

Analyseklasser

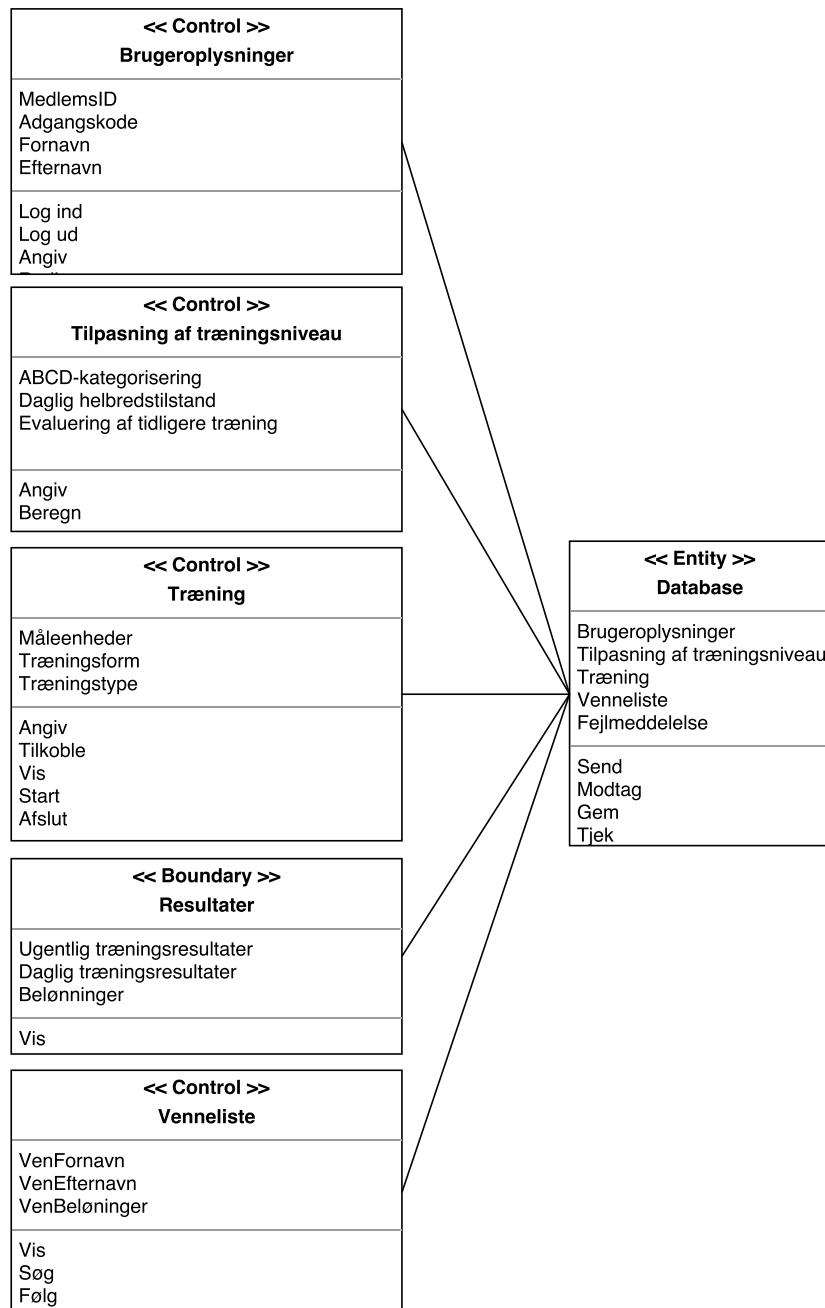
Inden analyseklasserne udarbejdes, foretages en analyse ud fra systembeskrivelsen, use case og funktionaliteter til at identificere substantiver og verber. Dette gøres for at sikre, at alle funktionaliteter indgår i klassediagrammet. Substantiver og verber fremgår af tabel 5.3.

Substantiver	Verber
Brugeroplysninger	
MedlemsID	Log ind
Adgangskode	Log ud
Fornavn	Validér
Efternavn	Angiv
Tilpasning af træningsniveau	Beregn
ABCD-kategorisering	Tilkoble
Daglig helbredstilstand	Vis
Evaluering af tidligere træning	Start
Træning	Afslut
Træningsform	Søg
Træningstype	Følg
Måleenheder	Send
Resultater	Modtag
Ugentlig træningsresultater	Gem
Daglig træningsresultater	Tjek
Belønninger	
Venneliste	
VenMedlemsID	
VenFornavn	
VenEfternavn	
VenBeløninger	
Database	
Fejlmeddeelse	

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 analyseklasserne er identificeres i tabel 5.3 inddeltes disse i analyseklasser og opdeles i stereotyper. 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 entityklasser, da de skal lagre og opdatere informationer. *Resultater* er defineret som boundaryklasse, da den skal vise resultater på 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. Under design af app'en tages udgangspunkt i de analyseklasser, der blev identificeret i kapitel 5, hvor disse analyseklasser omdannes til designklasser. Herefter visualiseres sammenspillet mellem de forskellige designklasser i sekvensdiagrammer. Til sidst i kapitlet vil designet af databasen beskrives.

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 nogle afgrænsninger for at gøre design og efterfølgende implementering af systemet mere konkret. I henhold til kravet om, at systemet skal kunne håndtere målinger fra kompatible enheder, vælges det at fokusere på målinger fra en sensor til optisk oximetri. 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, prægnans¹, 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
- Fejrlaten 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.

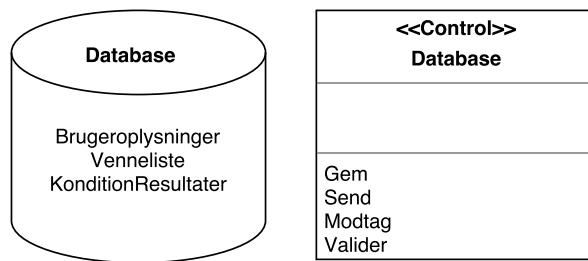
¹FiXme Note: Figur-grund eller prægnans omhandler det psykologiske fænomen, at man ikke kan opleve en figur uden samtidig at opleve dens baggrund

I sekvensdiagrammerne er systemets klasser opdelt efter MVC, og livlinerne farvekodes afhængig af klassens MVC-type. Model-klasser er røde, view-klasser er blå, og controller-klasser er grønne. Desuden findes i sekvensdiagrammer også database og kompatible enheder, som er illustreret med henholdsvis lilla og grå. 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 og log ud.

Database

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



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

Databasen indeholder Brugeroplysninger, Venneliste og KonditionResultater. 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 har brugerens resultater for konditionstræning. Sundhedspersonalet og brugeren har mulighed for at tilgå denne.

Designet af databasen er beskrevet i et ER-diagram og vil fremgå af afsnit 6.3. *Database* controlleren indeholder metoderne **Gem**, **Send** og **Modtag**. Disse har til formål at kommunikere mellem databasen og de forskellige controllere. Der oprettes forbindelse til denne ved log ind, hvor Brugeroplysninger, Venneliste og KonditionResultater sendes og gemmes i forskellige entitys.

Lagring af data

Når der oprettes forbindelse til databasen i forbindelse med, at brugeren logger ind på app'en, sendes og gemmes data i tre forskellige entity, 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>> Venneliste	<<Entity>> Resultater
#MedlemsID #Adgangskode #Fornavn #Efternavn #Kategorisering	#VenMedlemsID #VenFornavn #VenEfternavn #VenBelønninger	#Dato/tid #TræningsType #TræningsForm #DagligHelbredstilstand #Evaluering #Tid #Afstand #KompatibleMålinger #Belønninger

Figur 6.2: Designklasser for Entitys, herunder Brugeroplysninger, Venneliste og KonditionResultater. Attributterne er markeret med #, hvilket indikerer, at disse er beskyttede.²

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. *Venneliste* har informationer om andre brugere, der følges, såsom VenMedlemsID, VenFornavn, VenEfternavn og VenBelønninger. *Resultater* indeholder brugerens resultater, som er defineret ud fra DatoTid, TræningsType, TræningsForm, DagligHelbredstilstand, Evaluering, Tid, Afstand, KompatibleMåleenheder og Belønninger.

Fejlmeddelelse

Til at vise fejlmeddelelser er der opstillet en fælles grænseflade, hvori teksten ændres alt efter, hvilken fejl, der forekommer. Designklassen ses af figur 6.3.

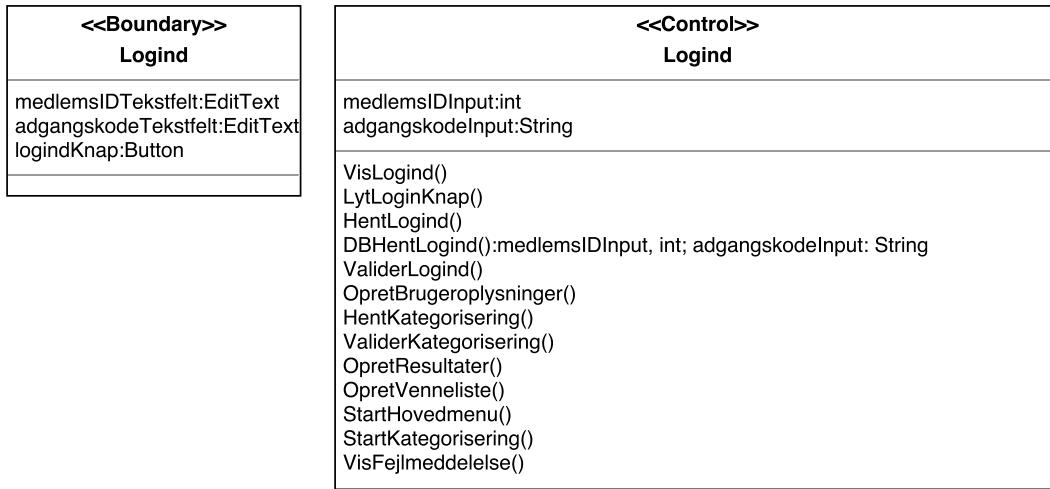
<<Boundary>> Fejlmeddelelse
Fejlmeddelelse:Tekstfelt:TextView OKKnap:Button

Figur 6.3: Designklasse for fejlmeddelelse.

Grænsefladen indeholder, foruden et tekstfelt, en OKKnap, der ved tryk henviser systemet til den forhenværende grænseflade. Fejlmeddelesesgrænsefladen håndteres af den pågældende controller, hvori fejlen opstår.

Log ind

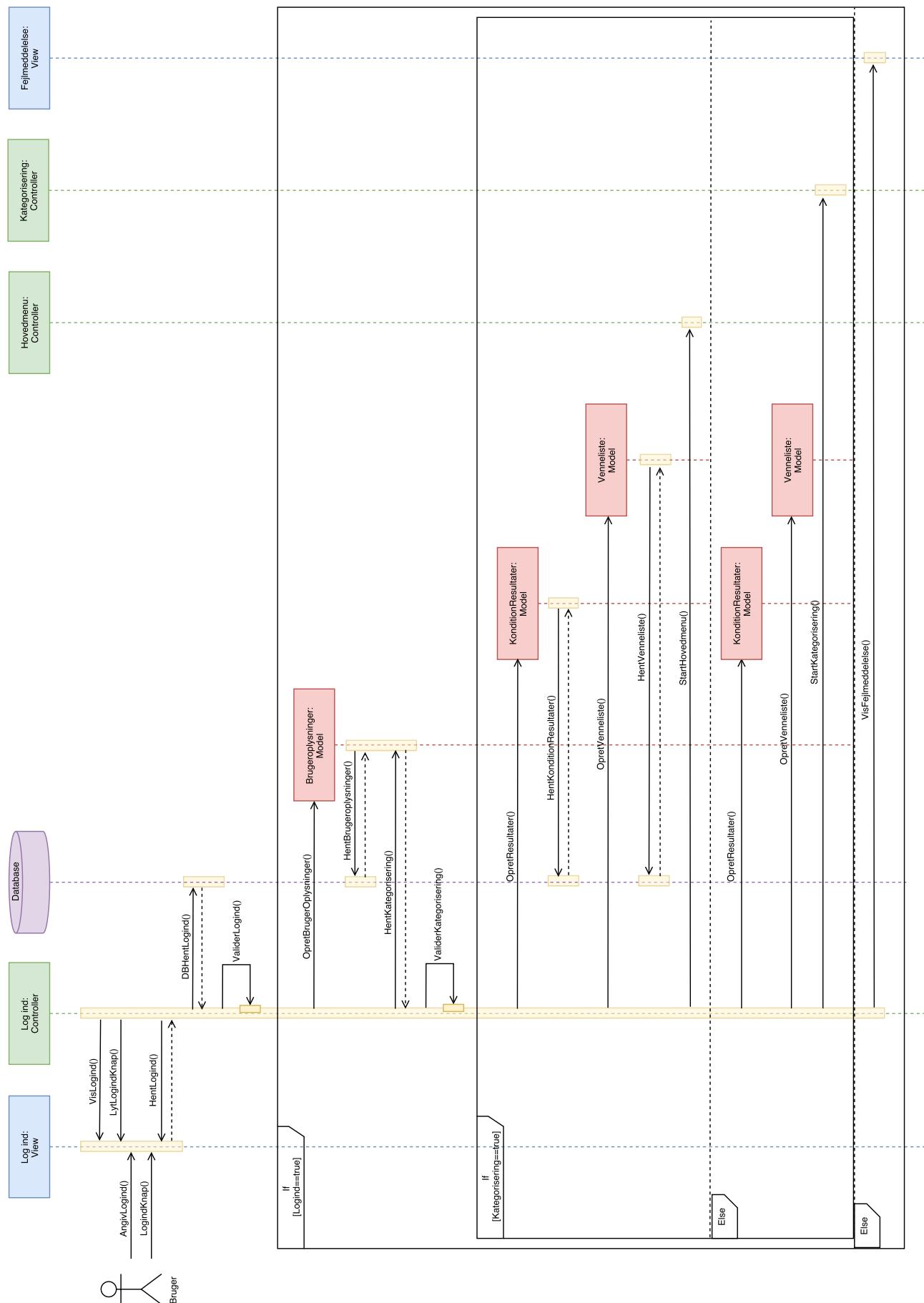
Log ind-funktionen inddeltes i klasser af typen boundary samt en tilhørende controller. Disse fremgår af figur 6.4.



Figur 6.4: 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, Valider, Opret og Start. Controlleren validerer log ind og opretter tre entitys, hvori oplysninger senere kan lagres. Disse entitys 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 6.5.



Figur 6.5: 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 til det indtastede medlemsID i databasen, hvorefter log ind valideres. Bekræftes log ind oprettes en entity af *Brugeroplysninger*, der henter og lagrer brugeroplysninger fra databasen. Forefindes en kategorisering i disse oplysninger, oprettes to yderligere entitys, der omfatter *KonditionResultater* og *Venneliste*.³ Disse entitys henter og lagrer ligeledes deres respektive data fra databasen. Systemet henvises hertil til hovedmenu, der fremgår af figur 6.9. Forekommer ingen kategorisering i *Brugeroplysninger*, oprettes de to entitys, *KonditionResultater* og *Venneliste*, dog uden at hente og lagre data fra databasen. Systemet henvises herefter til kategoriseringen, hvilket er illustreret af ???. Mislykkes log ind, vises en fejlmeldelse til *Log ind* grænsefladen, hvorved brugeren igen kan indtaste sit log ind.

Kategorisering

Kategorisering inddeltes i boundarys og en dertilhørende controlklasse, som det fremgår af figur 6.6.

<table border="1"> <thead> <tr> <th style="text-align: center;"><<Boundary>></th></tr> <tr> <th style="text-align: center;">CATScore</th></tr> </thead> <tbody> <tr> <td>UdsagnTekstfelt:TextView 0til5Knap:Button VidereKnap:Button</td></tr> </tbody> </table> <table border="1"> <thead> <tr> <th style="text-align: center;"><<Boundary>></th></tr> <tr> <th style="text-align: center;">Indlæggelser</th></tr> </thead> <tbody> <tr> <td>IndlæggelsesTekstfelt:TextView IndlæggelserKnap:Button VidereKnap:Button</td></tr> </tbody> </table> <table border="1"> <thead> <tr> <th style="text-align: center;"><<Boundary>></th></tr> <tr> <th style="text-align: center;">Kategorisering</th></tr> </thead> <tbody> <tr> <td>ABCDKategoriseringTekstfelt:TextView VidereKnap:Button</td></tr> </tbody> </table>	<<Boundary>>	CATScore	UdsagnTekstfelt:TextView 0til5Knap:Button VidereKnap:Button	<<Boundary>>	Indlæggelser	IndlæggelsesTekstfelt:TextView IndlæggelserKnap:Button VidereKnap:Button	<<Boundary>>	Kategorisering	ABCDKategoriseringTekstfelt:TextView VidereKnap:Button	<table border="1"> <thead> <tr> <th style="text-align: center;"><<Control>></th></tr> <tr> <th style="text-align: center;">Kategorisering</th></tr> </thead> <tbody> <tr> <td>CATscore: int AntallIndlæggelser: int ABCDkategorisering: char</td></tr> <tr> <td>VisCATscore() VisUdsagn() Lyt0til5Knap() BeregnCATscore()</td></tr> <tr> <td>VisIndlæggelser() LytIndlæggelserKnap()</td></tr> <tr> <td>BeregnABCDKategorisering() VisABCDKategorisering()</td></tr> <tr> <td>LytVidereKnap() GemABCDkategorisering() StartHovedmenu()</td></tr> </tbody> </table>	<<Control>>	Kategorisering	CATscore: int AntallIndlæggelser: int ABCDkategorisering: char	VisCATscore() VisUdsagn() Lyt0til5Knap() BeregnCATscore()	VisIndlæggelser() LytIndlæggelserKnap()	BeregnABCDKategorisering() VisABCDKategorisering()	LytVidereKnap() GemABCDkategorisering() StartHovedmenu()
<<Boundary>>																	
CATScore																	
UdsagnTekstfelt:TextView 0til5Knap:Button VidereKnap:Button																	
<<Boundary>>																	
Indlæggelser																	
IndlæggelsesTekstfelt:TextView IndlæggelserKnap:Button VidereKnap:Button																	
<<Boundary>>																	
Kategorisering																	
ABCDKategoriseringTekstfelt:TextView VidereKnap:Button																	
<<Control>>																	
Kategorisering																	
CATscore: int AntallIndlæggelser: int ABCDkategorisering: char																	
VisCATscore() VisUdsagn() Lyt0til5Knap() BeregnCATscore()																	
VisIndlæggelser() LytIndlæggelserKnap()																	
BeregnABCDKategorisering() VisABCDKategorisering()																	
LytVidereKnap() GemABCDkategorisering() StartHovedmenu()																	

Figur 6.6: Designklasser for kategorisering. Til venstre ses de forskellige grænseflader, henholdsvis CATScore, Indlæggelser og Kategorisering. Til højre fremgår den dertilhørende controller.

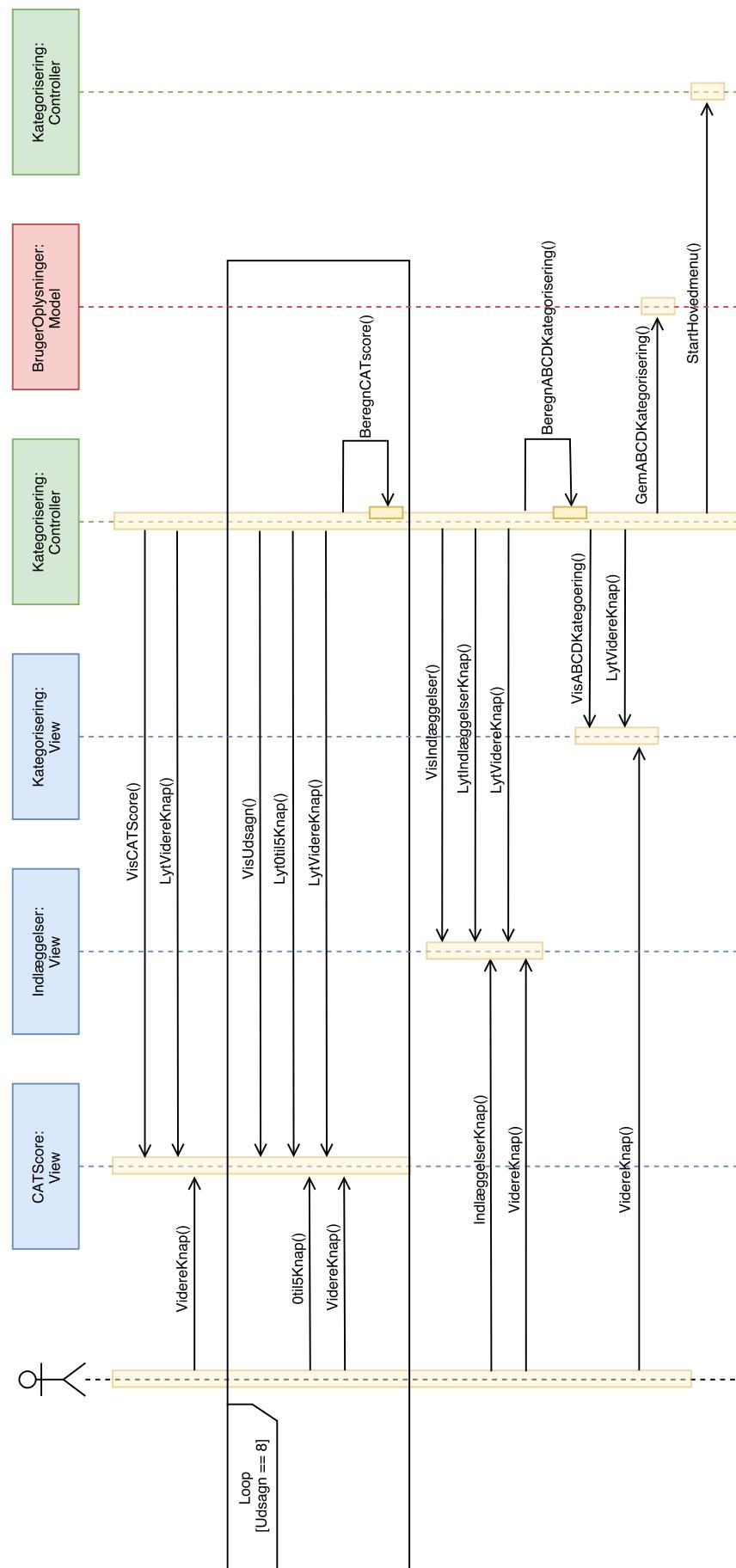
Kategoriseringen inddeltes i tre grænseflader, herunder *CATScore*, *Indlæggelser* samt *Kategorisering*. Dette er valgt, idet grænsefladerne skal have forskellige layouts, og en opdeling vil gøre implementeringen af grænsefladerne mere overskuelig. Desuden ønskes det at gøre app'en overskuelig og brugervenlig, hvilket kan opnås ved at tydeliggøre adskillelsen mellem CAT-score og besvarelse af antal årlige indlæggelser, j.f. gestaltloven om lighed i ???. Hertil skal

³FiXme Note: Måske et dårligt designvalg, der vil stadig skulle hentes meget information

brugeren foretage minimale valg på hver brugergrænseflade, 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 spørgsmålene. 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 CAT-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 6.7.

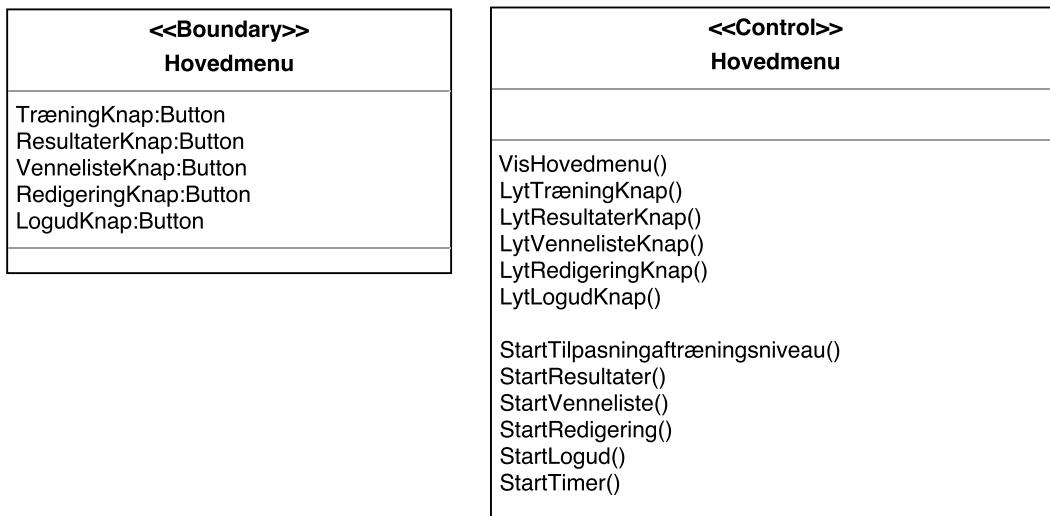


Figur 6.7: Sekvensdiagram for kategorisering.

Det fremgår af sekvensdiagrammet, at grænsefladen for *CATScore* vises som det første. Brugeren introduceres her til CAT-score, 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, *Otil5Knap*. Idet brugeren trykker videre fra hvert spørgsmål, adderes CAT-scoren. Efter de otte udsagn er besvaret, og den samlede CAT-score 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*, når brugeren trykker videre. Herefter startes hovedmenuen.

Hovedmenu

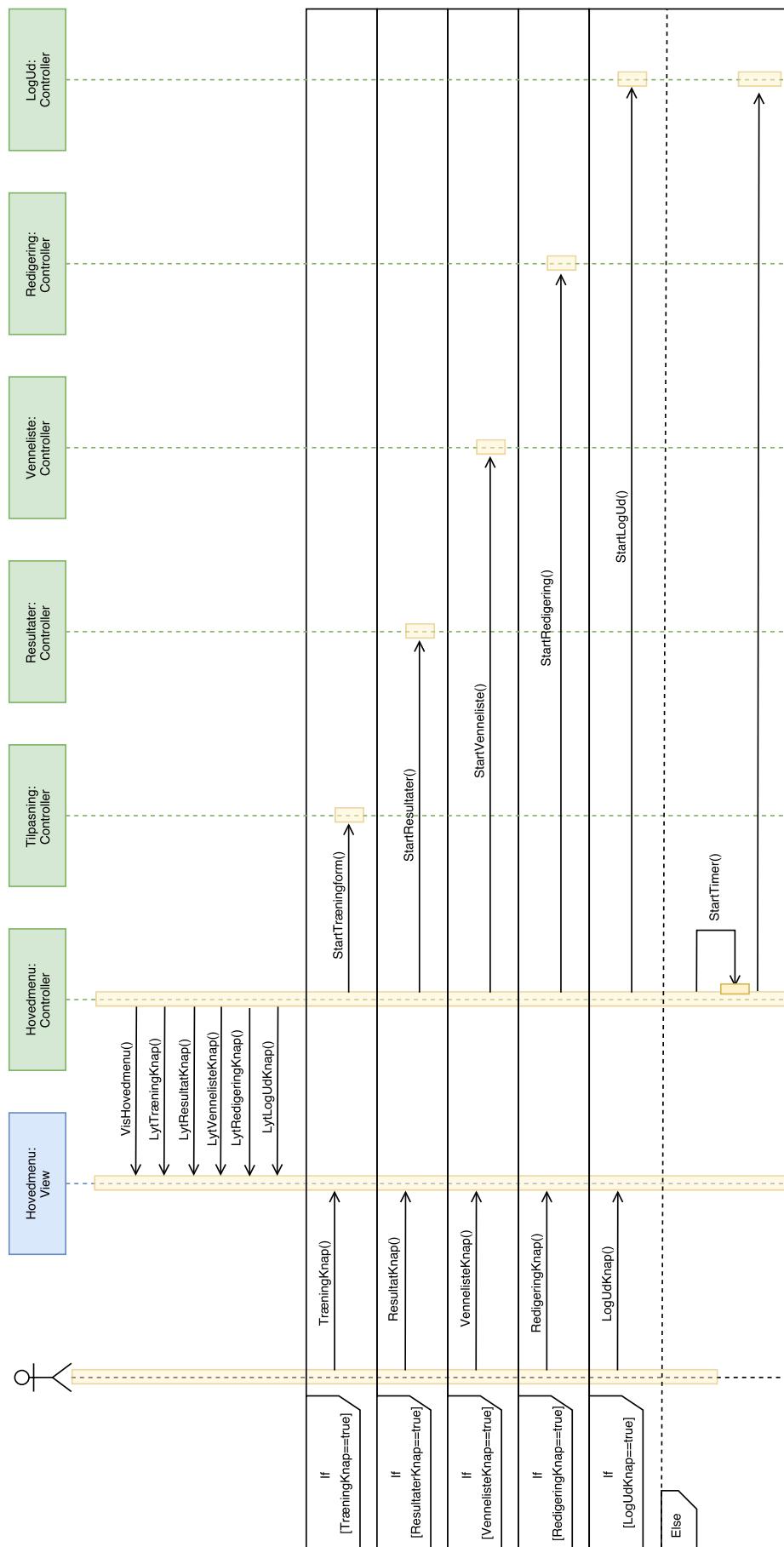
Hovedmenuen er den primære grænseflade, hvor andre grænseflader kan tilgås fra via deres controller. Hovedmenuen inddeltes i en boundary samt en dertilhørende controller. Disse fremgår af figur 6.8.



Figur 6.8: 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 6.9.



Figur 6.9: 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 *Hovedmenuen*, hvorefter en timer starter.

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 6.10.

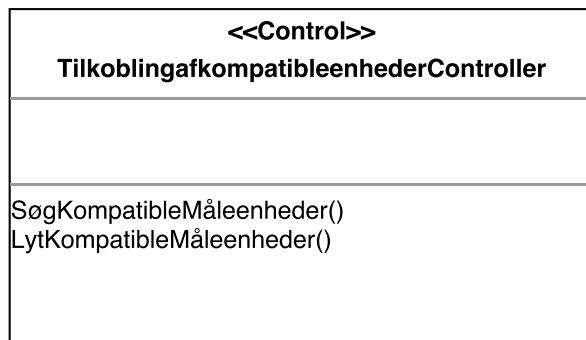
<<Boundary>> Træningsform	<<Boundary>> Helbredstilstand	<<Control>> Tilpasning
TræningsformTekstfelt:TextView TræningsformKnap:Button VidereKnap: Button TilbageKnap:Button	Helbredstilstand:Tekstfelt:Textview HelbredstilstandKnap:Button VidereKnap:Button TilbageKnap:Button	Træningsform:int Træningstype:int Helbredstilstand:int LytVidereKnap() VisTræningsform() LytTræningsformKnap() VisTræningstype() LytTræningstypeKnap() VisHelbredstilstand() LytHelbredstilstandKnap() HentKategorisering() HentEvaluering() LytKompatibleEnheder() BeregnAnbefaletTræning() VisAnbefaletTræning() LytStartKnap() VisTræning()
<<Boundary>> Træningstype	<<Boundary>> AnbefaletTræning	
Træningstype:Tekstfelt:TextView TræningstypeKnap:Button VidereKnap:Button TilbageKnap:Button	AnbefaletTræningTekstfelt:TextView StartKnap:Button TilbageKnap:Button	

Figur 6.10: 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 og Beregn-metoder til denne controller.

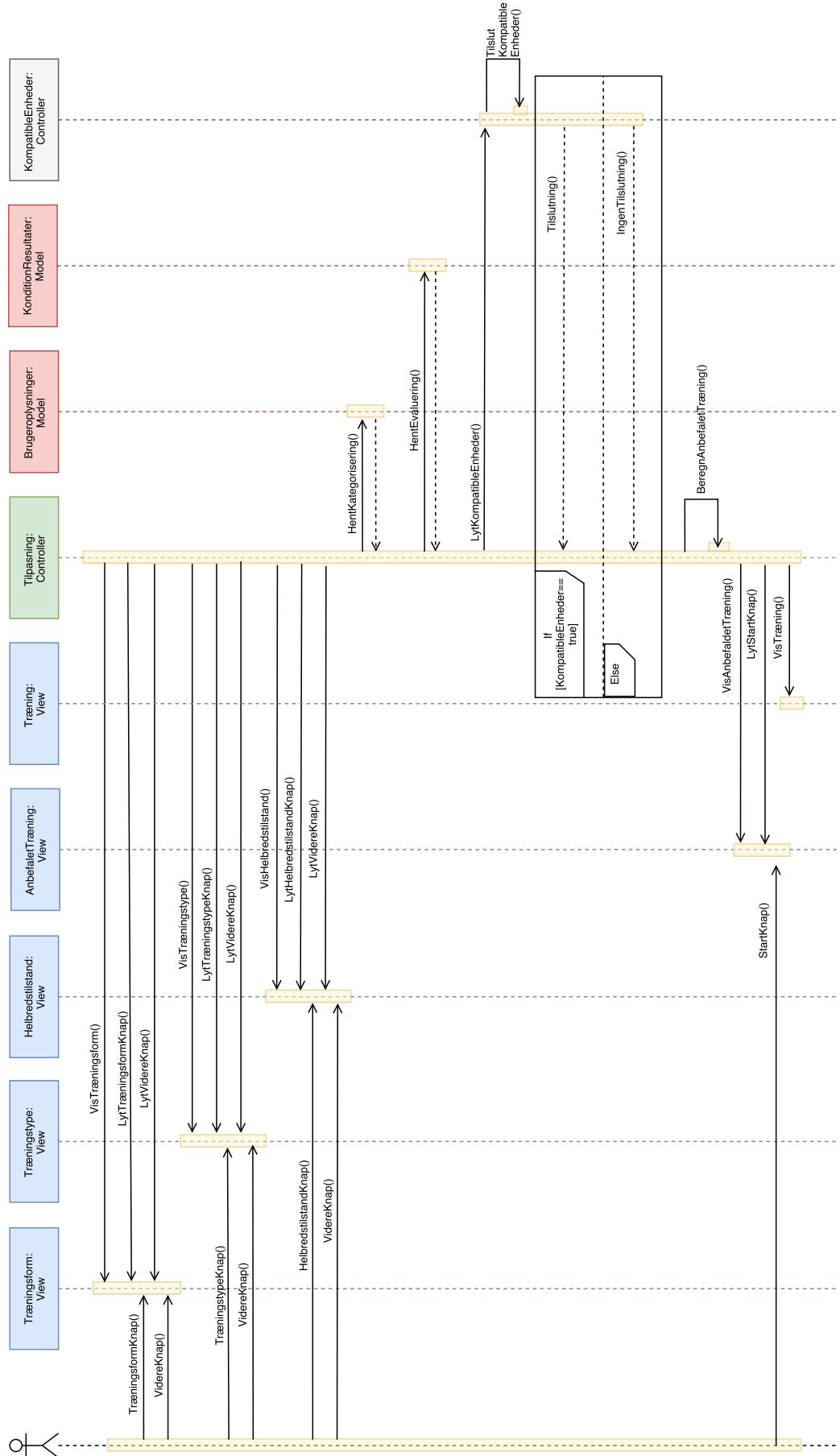
Da det for brugeren skal være muligt at tilkoble kompatible enheder, forekommer endnu en controller. Denne skal håndtere kommunikation mellem systemet og kompatible måleenheder. Designklassen for kompatible måleenheder ses af figur 6.11.



Figur 6.11: Designklasse for kompatible måleenheder. Denne er en controller.

Denne controller tilgås ikke af brugeren, da det ønskes, at denne er en autonom funktion, der lytter efter og eventuelt tilslutter kompatible måleenheder førend en træning påbegyndes. Controlleren har en LYT og TILSLUT metode.

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

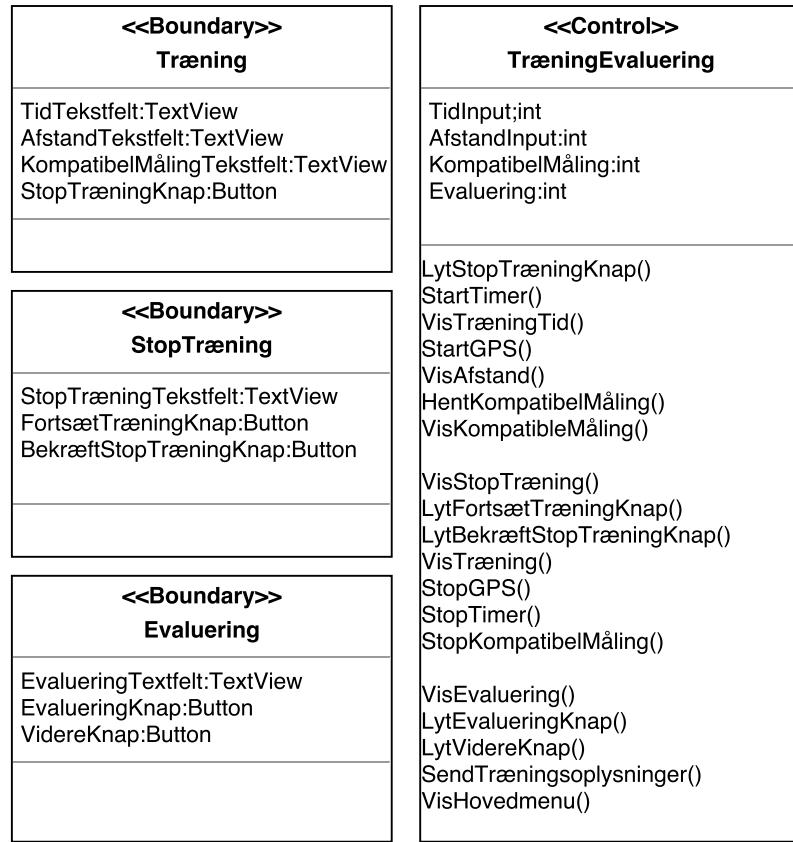
**Figur 6.12:** Sekvensdiagram for tilpasning af træningsniveau.

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 vejrrækningsøvelser. 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 evaluering i *KonditionResultater*. Controlleren lytter efter kompatible enheder fra den anden controller *KompatibleEnheder*.⁴ Hvis controlleren, *KompatibleEnheder*, prøver at tilslutte kompatible enheder sender denne en besked til *Tilpasning* controlleren om, hvorvidt en kompatible måleenhed er tilsluttet. Hvis kompatible måleenheder tilsluttes, vil dette fremgå i grænsefladen, *AnbefaletTræning*. Modsat vil det fremgå, hvis ingen kompatible måleenheder er tilsluttet. Efterfølgende beregner controlleren anbefalet træningsniveau og viser dette på grænsefladen for *AnbefaletTræning*. Brugeren kan herefter trykke på StartKnap, hvis denne trykkes på viser controlleren grænsefladen for *Træning*.

Træning og evaluering

Træningen er opdelt i tre boundarys, hvor de to er aktive under træningen samt ved stop træning. Den sidste boundary er aktiv efter træningen, idet træningen skal evalueres. Der er til disse tre boundarys opstillet en fælles controller. Boundarys og controller fremgår af figur 6.13.

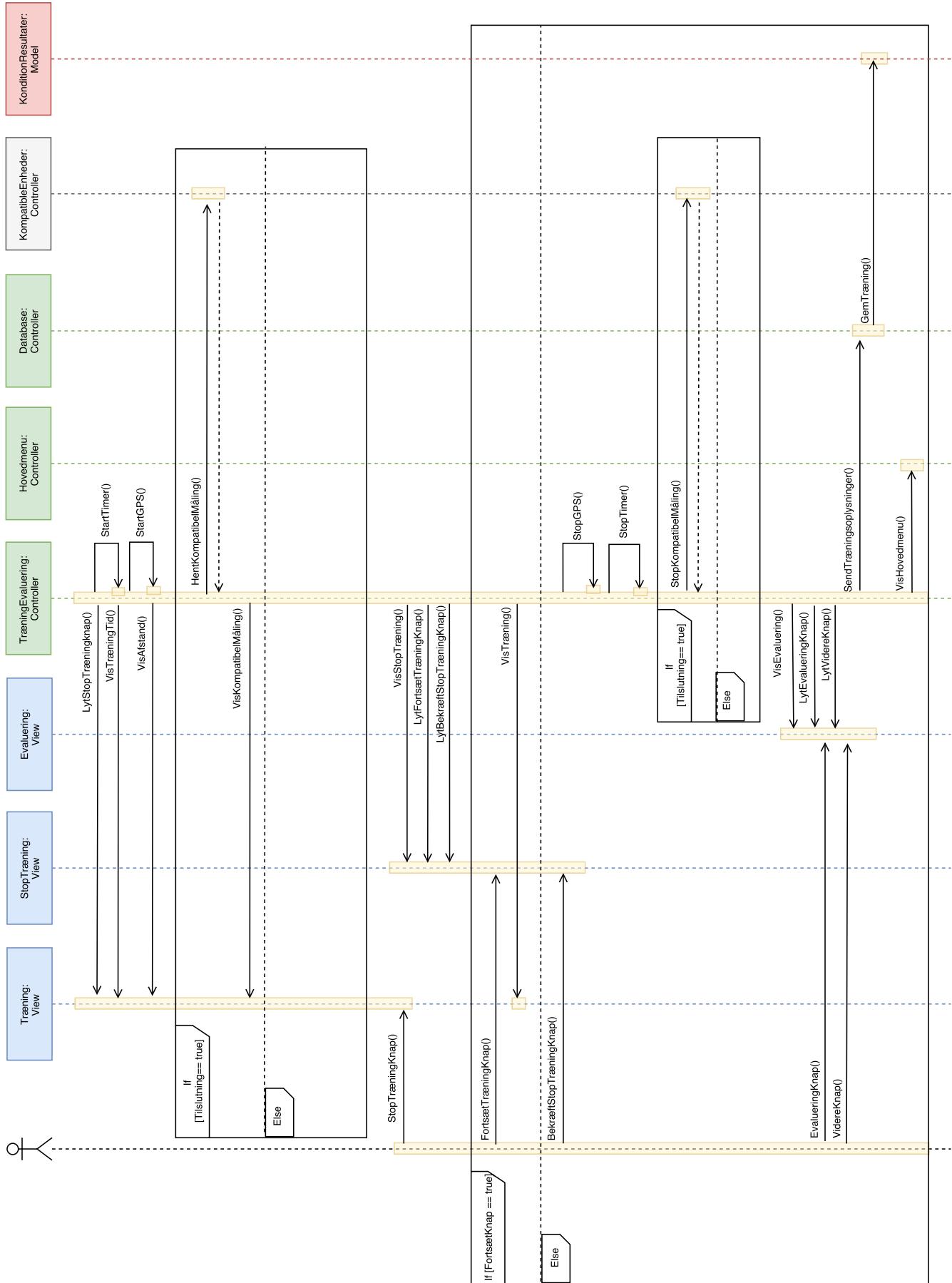
⁴FiXme Note: OBS! Vi er stadig lidt usikre på denne del



Figur 6.13: 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 tekstmeldinger og knapper af typen TextView og Button. Controlleren *TræningEvaluering* indeholder metoderne Lyt, Vis, Start, Hent, Stop og Send.

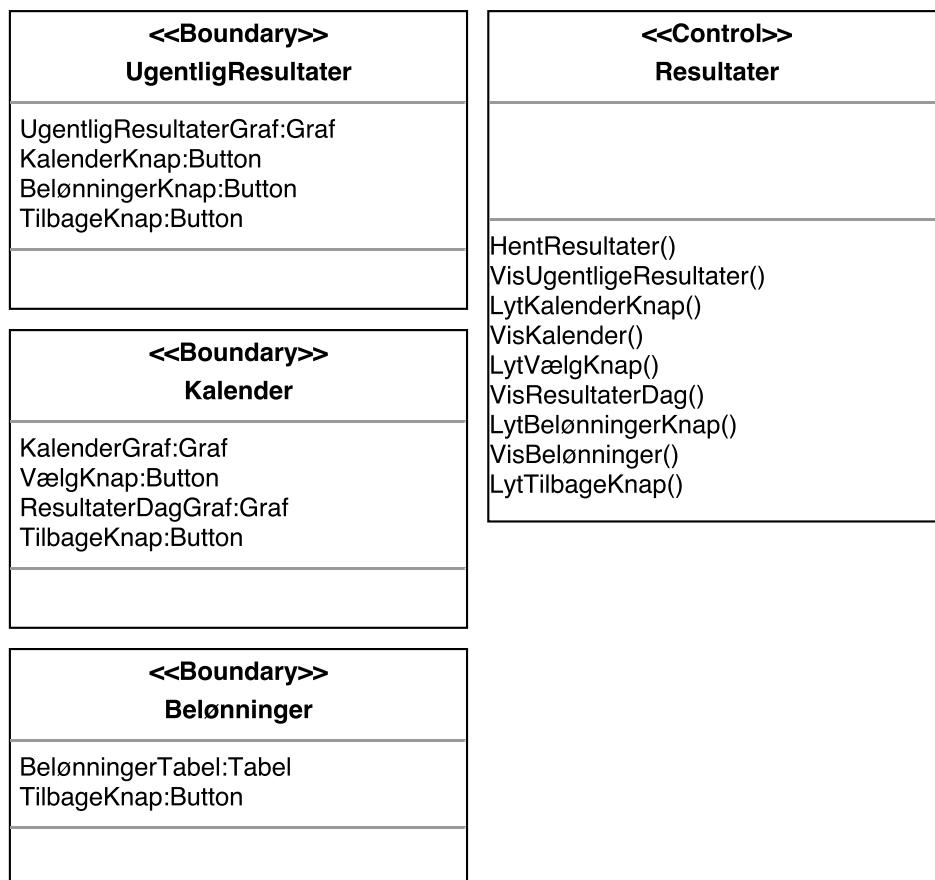
I sammenspil med designklasserne er der udarbejdet et sekvensdiagram, hvilket fremgår af figur 6.14.

Figur 6.14: Sekvensdiagram for træning.⁵

Træning er den første grænseflade, der vises ved en igangværende træning. Controlleren *TræningEvaluering* starter timer og GPS. Herefter vises tiden og afstanden i grænsefladen. Controlleren henter målinger fra kompatible enheder, hvis disse er tilsluttet og viser dem.⁶. 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* igen og brugeren kan fortsætte træningen. Vælges BekræftStopTræningKnap stopper controlleren med at lagre data fra GPS, timer og eventuelle kompatible måleenheder. 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*⁷. Herefter gives der besked til *Hovedmenu* controlleren om at vise hovedmenuen.

Resultater

Resultater har tre boundarys, hvor til der er opstillet en kontrolklasse. Boundarys og tilhørende controller fremgår af figur 6.15.



Figur 6.15: Designklasser for resultater. Til venstre ses de tre boundarys, *UgentligResultater*, *Kalender* og *Belønninger*. Til højre fremgår den tilhørende controller.

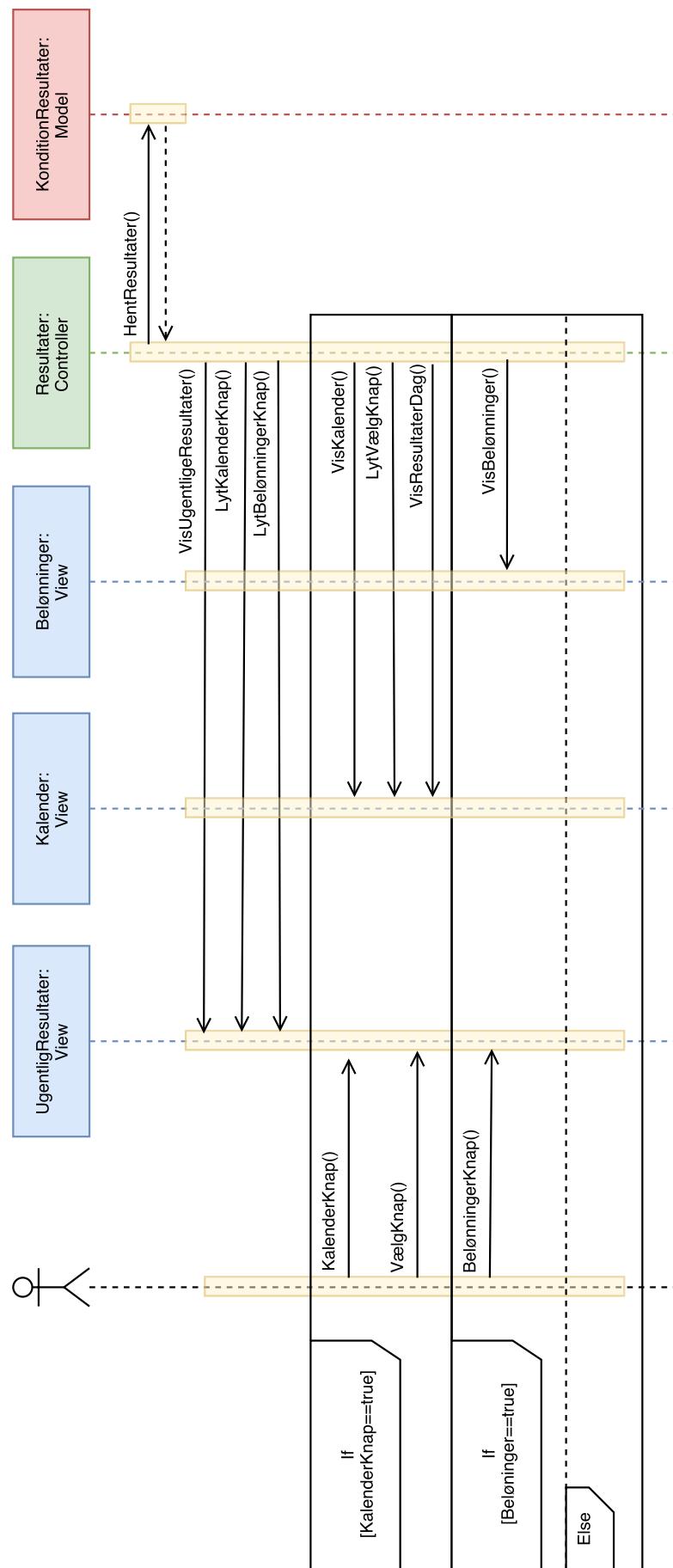
⁶FiXme Note: OBS!! Dette er stadig lidt uklart

⁷FiXme Note: Hvad gør vi i forhold til dette?

I grænsefladerne er der grafer, tabeller og knapper. Disse er af typen Graf, Tabel og Button.
⁸. Controlleren for *Resultater* indeholder Hent, Vis og Lyt metoder.

I sammenspil med designklasserne for resultater er der udarbejdet et sekvensdiagram, hvilket fremgår af figur 6.16

⁸**FiXme** Note: OBS!! Dette er stadig lidt uklart



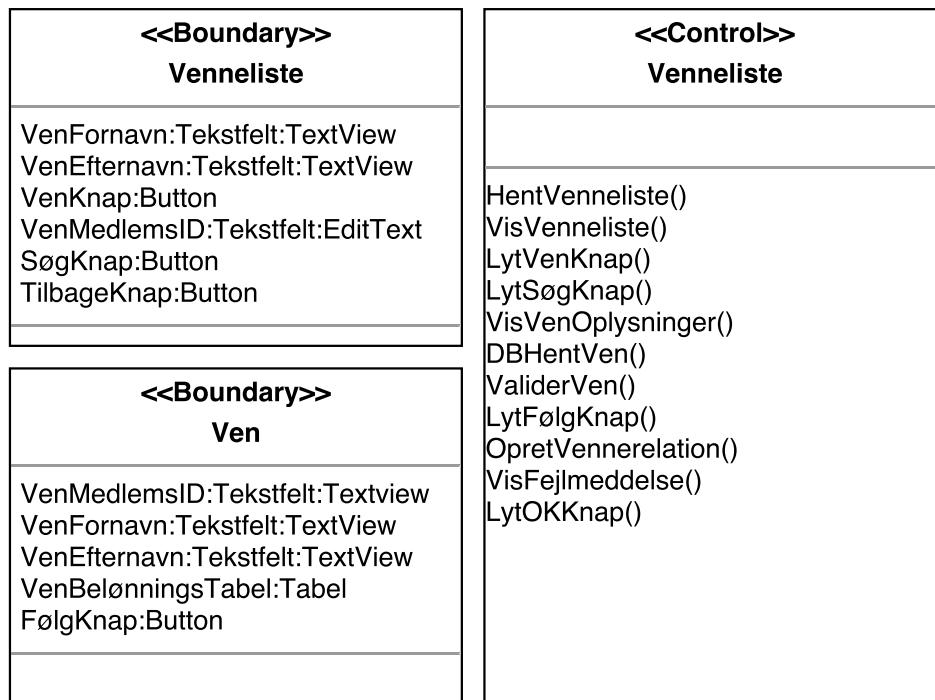
Figur 6.16: Sekvensdiagram for Resultater.

Når brugeren er tilgået resultater henter controlleren, *KonditionResultater* resultater fra modellen *KonditionResultater*. Hvis brugeren endnu ikke har nogle resultater, vil disse ikke hentes, hvorved disse ikke vil vises i grænsefladerne.

Grænsefladen *UgentligResultater* viser ugentlige resultater i en graf. Brugeren har mulighed for at trykke KalenderKnap eller BeløningerKnap. Trykker brugeren på KalenderKnap, vises grænsefladen for *Kalender*, hvor brugeren har mulighed for at vælge en dag. Vælges dette, vises resultater i en graf for den valgte dag. Vælger brugeren at trykke på BelønningerKnap, viser grænsefladen *Belønninger* en tabel med belønninger. Trykker brugeren ikke på en af knapperne, forbliver brugeren på grænsefladen for *UgentligeResultater*

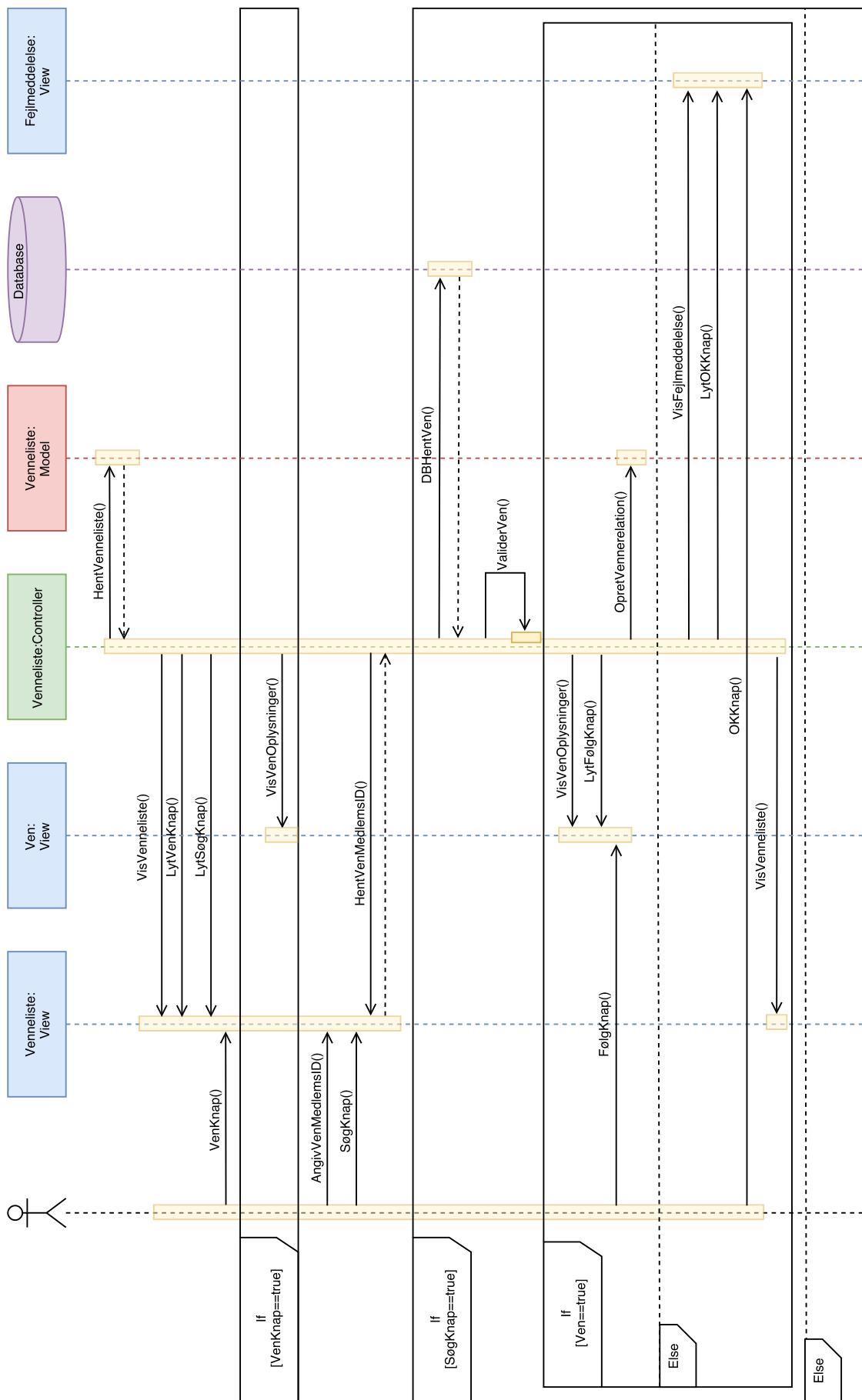
Venneliste

Venneliste inddeltes i to boundaryklasser og en dertilhørende controlklasse, som fremgår af figur 6.17.



Figur 6.17: Designklasser for venneliste. Til venstre ses boundary for Venneliste og Ven, hvor der til højre ses tilhørende controller.

I grænsefladen *Venneliste* findes tekstfelter, af typen TextView, med fornavn og efternavn 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 SøgKnap, af typen Button, som muliggør, at brugeren kan søge på andre brugere med deres medlemsID. Når en bruger tilgås, enten via vennelisten eller ved at søge på en bruger, vises grænsefladen for den pågældende ven. Hertil vises tekstfelter, af typen TextView, med brugerens medlemsID, fornavn, efternavn samt vedkommendes belønninger. Hvis brugeren ikke følges, er der her opstillet en FølgKnap. Den tilhørende controller har metoderne Vis, Hent, Lyt, Valider og Opret. Ud fra de nævnte designklasser er udarbejdet et sekvensdiagram, der fremgår af figur 6.18.



Figur 6.18: Sekvensdiagram for venneliste.

Controlleren henter brugerens venneliste fra vennelistemodellen, hvorefter grænsefladen for venneliste vises. Dertil lytter controlleren på VenKnap samt SøgKnap. Vælger brugeren at tilgå en ven fra vennelisten, vises grænsefladen for den valgte 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 findes i databasen, vises VenOplysninger, hvorimod en grænseflade for fejlmeddelelse vises, hvis brugeren ikke findes. Brugeren skal herved benytte OKKnap på grænsefladen for at bekræfte, at fejlen er set, hvorefter systemet returneres til brugerens venneliste. Grænsefladen for den søgte ven tillader ligeledes muligheden for at følge brugeren, hvis vedkommende ikke følges. Ønskes det at følge brugeren, oprettes en vennerelation i vennelistemodellen.

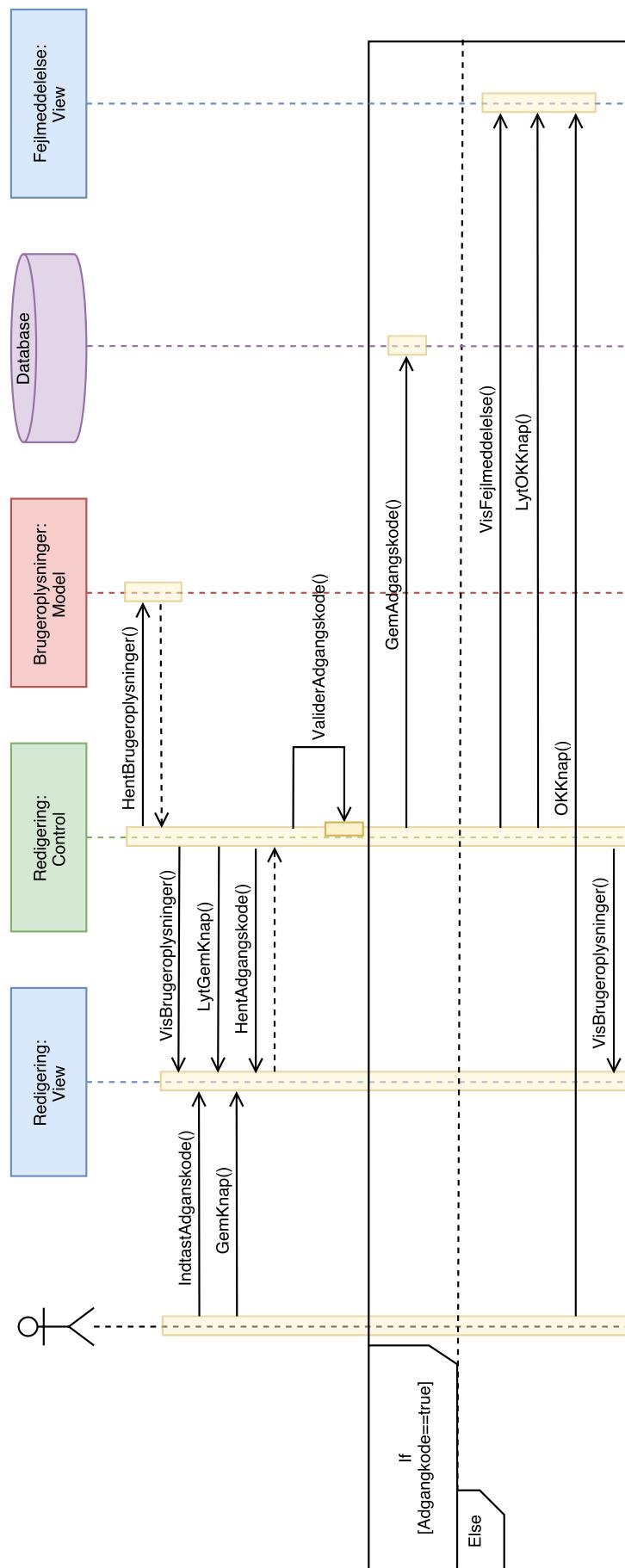
Redigering

Redigering inddeltes i en boundary og en tilhørende controller, som det fremgår af figur 6.19.

<<Boundary>>	<<Control>>
Redigering	Redigering
MedlemsID:Tekstfelt:TextView Fornavn:Tekstfelt:TextView Efternavn:Tekstfelt:TextView Kategorisering:Tekstfelt:TextView NyAdgangskode:Tekstfelt:EditText GentagAdgangskode:Tekstfelt:EditText GemKnap:Button TilbageKnap:Button	NyadgangskodeInput:String GentagAdgangskodeInput:String HentBrugeroplysninger() VisBrugeroplysninger() LytGemKnap() HentAdgangskode() ValiderAdgangskode() GemAdgangskode() VisFejlmeddelse() LytOKKnap() LytTilbageKnap()

Figur 6.19: Designklasser for Redigering. Til venstre fremgår boundary og til højre controller for redigering.

I grænsefladen for *redigering* opstilles tekstfelter af typen TextView for medlemsID, fornavn, efternavn og kategorisering. Derudover opstilles tekstfelter, af typen EditText, for ny adgangskode og gentag adgangskode, hvor brugeren kan ændre adgangskoden. Dertil er der en GemKnap og en TilbageKnap, af typen Button. Gemknappen indikerer ved tryk, at brugeren ønsker at gemme den nye adgangskode. Den tilhørende controller har metoderne Hent, Vis, Lyt, Valider og Gem. Til disse klasser er der opstillet et sekvensdiagram, hvilket fremgår af figur 6.20.

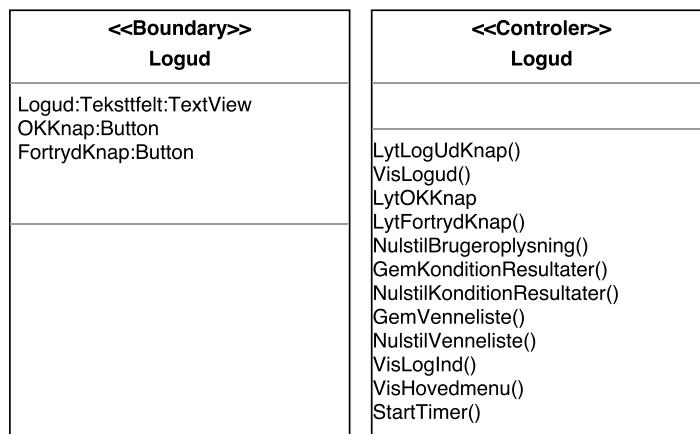


Figur 6.20: Sekvensdiagram for redigering.

Controlleren *Redigering* henter brugeroplysninger fra modellen *Brugeroplysninger*. Disse oplysninger vises i grænsefladen *Redigering*, hvortil brugeren kan se deres 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 databasen. Dette gøres af sikkerhedsmæssige årsager, således koden ikke først gemmes, idet brugeren logger ud af app'en. Overholder adgangskoden derimod ikke kriterierne, vises en grænseflade for fejlmeddelelse. På denne grænseflade er der opstillet en OKKnap, der ved tryk henviser systemet tilbage til grænsefladen for redigering.

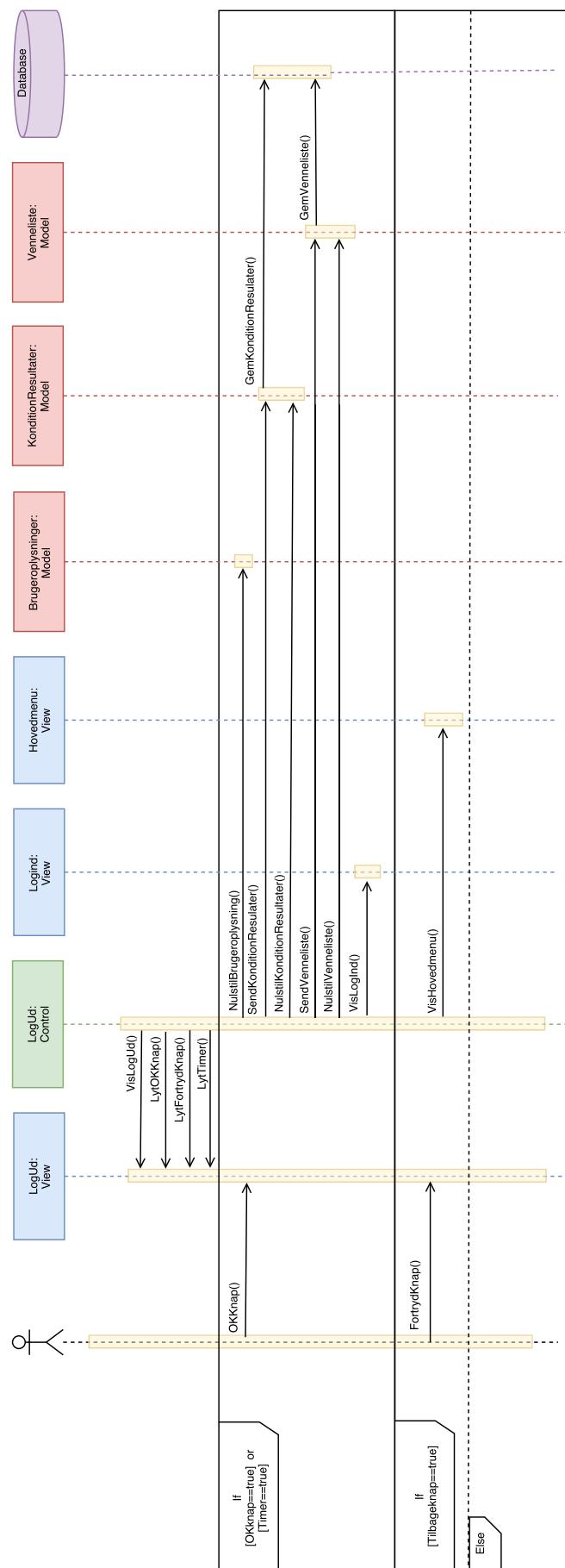
Log ud

Log ud inddeltes i en boundary og en dertilhørende controller, som det fremgår af figur 6.21.



Figur 6.21: Designklasser for log ud. Til venstre ses boundary og til højre controller.

Der opstilles i log ud et tekstfelt, af typen TextView, for log ud. Dertil opstilles en OKKnap og FortrydKnap af typen Button. Den tilhørende controller indeholder metoderne, Lyt, Vis, Nulstil, Send, Gem og Start. I sammenspil med designklasserne er der opstillet et sekvensdiagram, som fremgår af figur 6.22.



Figur 6.22: Sekvensdiagram for log ud.

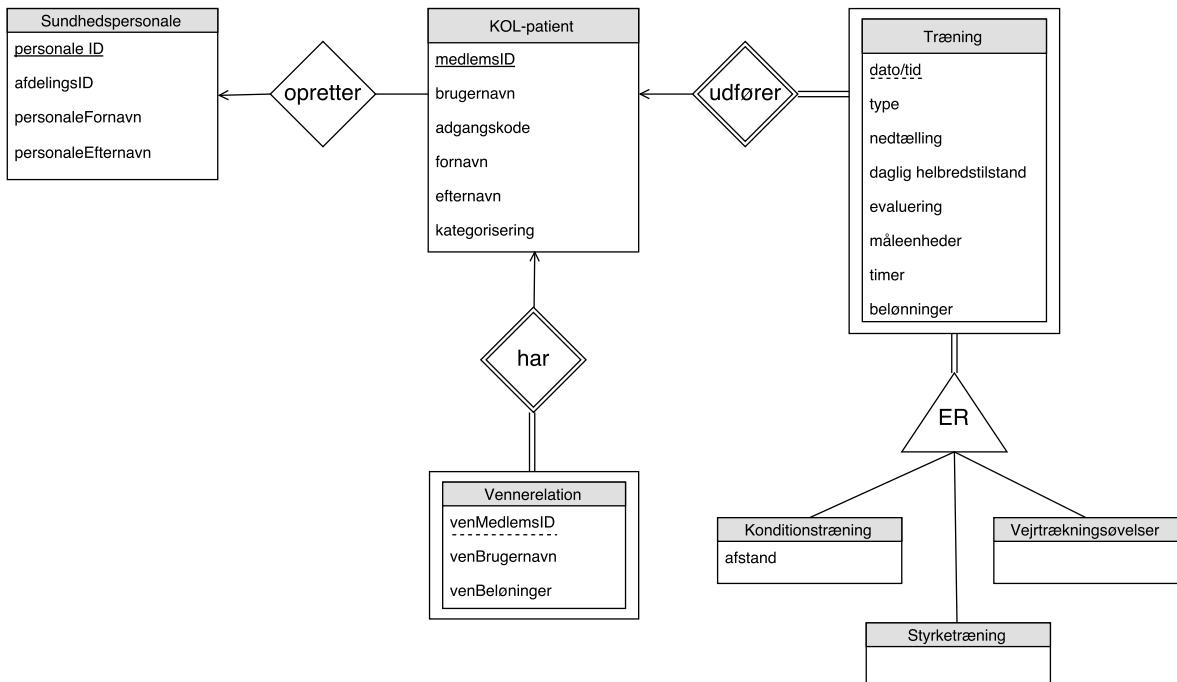
Når brugeren via grænsefladen for hovedmenuen trykker på knappen for log ud vises grænsefladen for *Log ud*. Her har brugeren mulighed for at trykke på OKKnap eller FortrydKnap. Hvis brugeren trykker på OKKnap eller timer udløst nulstiller controlleren, *Log ud*, modellen for Brugeroplysninger. Herefter sendes besked til modellerne KonditionResultater og Venneliste om at gemme oplysningerne i databasen, hvorefter modellerne nulstilles. Grænsefladen for *Log ind* vises efterfølgende, så brugeren har mulighed for at logge ind igen. Trykker brugeren på FortrydKnap, vises grænsefladen for hovedmenu igen. Hvis brugeren ikke angiver noget vil den forblive på grænsefladen for *Log ud*.

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 vennerelation.

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, vennerelation, træning, konditionstræning, styrketræning og vejrtækningsøvelser. ER-diagrammet for databasen fremgår af figur 6.23.



Figur 6.23: ER-diagram for database.

Af figur 6.23 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*, fornavn, efternavn og afdelingsID, hvor *personaleID* er primærnøglen, der kan identificere personen. Den enkelte KOL-patient er en stærk entitet, som registreres med primærnøglen, *medlemsID*, samt brugernavn, fornavn, efternavn, adgangskode og kategorisering. Derudover fremgår de svage entiteter, herunder vennerelation, træning, konditionstræning, styrketræning og vejrtækkningsøvelser. Én KOL-patient har mange vennerelationer, 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*. Af figur 6.23 fremgår det, at konditionstræning, styrketraening og vejrtækkningsøvelser nedarver flere attributter fra træningen, da entiteterne har ligheder og forskelligheder. Konditionstræning har foruden de nedarvede attributter afstand.

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> , brugernavn, adgangskode, fornavn, efternavn, kategorisering)
Svage entiteter	Træning = (<u>medlemsID</u> , <u>tid/dato</u> , type, nedtælling, daglig helbredstilstand, evaluering, måleenheder, timer, belønninger) Vennerelation = (<u>medlemsID</u> , <u>venMedlemsID</u> , venBrugernavn, venBelønninger)

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> , type, nedtælling, daglig helbredstilstand, evaluering, måleenheder, timer, belønninger) Vennerelation = (<u>medlemsID</u> , <u>venMedlemsID</u> , venBelønninger)

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 omsætningen af design til kode. Det er valgt at implementere app'en i Android Studio version 2.3 og programmere i Java, da dette er et objektorienteret programmeringssprog. De forskellige controllerer og views, der blev defineret i sekvensdiagrammerne implementeres i henholdsvis Java-klasser og XML-filer. Database samt modeller er implementeret i MySQL ved brug af phpMyAdmin, der er et online databaseadministrationssystem og understøtter Structured Query Language (SQL) ¹....

I ?? 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.

7.0.1 Implementering af objektorienteret design

I afsnittet vil der fremgå udsnit af Java kode og i nogle tilfælde figurer af grænseflader. Dette er valgt, da nogle funktioner er kompliceret at beskrive.

Log ind

7.0.2 Implementering af database

¹FiXme Note: Her skal muligvis tilføjes noget

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:

- Sundhedspersonalet **kan vi teste dette** skal kunne oprette brugere 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 ("INTO users (, , navn, medlemsid, db_adgangskode,) VALUES (, , Jens Jensen, 01170301, adgangskode,);"). ✓ 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"). ✓ Kategorisering er gemt i databasenTryk på "REDIGER ADGANGSKODE" via hovedmenuen. ✓ Kategorisering er hentet til rediger adgangskode.
Resultat	

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 registeret i databasen. Der er opstillet funktionelle krav til log ind:

- Brugere skal kunne log ind med et 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 6.4.

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"> 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. Indtast et MedlemsID som eksisterer i databasen ("01170301") og en adgangskode, som ikke tilhører medlemsID'et ("forkertadgangskode") og tryk på log ind knappen. <ul style="list-style-type: none"> ✓ Forventet log ind mislykkedes. Fejlmeddeelse vises i grænsefladen for log ind. Indtast et medlemsID ("01170301") og et adgangskode ("adgangskode") som eksisterer i databasen. <ul style="list-style-type: none"> ✓ Forventet log ind lykkes.
Resultat	

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 **efter brugeren har angivet svar på udsagn fra CATscore og antallet af årlige indlæggelser på grund af KOL. Og kun første gang de logger ind**

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 svarende til A, B, C, eller D.
Main flow:	<p>1. 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 (“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	

Tabel 8.3: Test af kategorisering

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 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 "KONDITIONSTRÆNING" og tryk videre. Vælg herefter "GÅ" og tryk videre. Vælg "MODERAT" og tryk videre.</p> <ul style="list-style-type: none"> ✓ Forventet anbefaling af træningstid er 30 min <p>2. Vælg "KONDITIONSTRÆNING" og tryk videre. Vælg herefter "LØB" og tryk videre. Vælg "MEGET DÅRLIGT" og tryk videre.</p> <ul style="list-style-type: none"> ✓ Forventet anbefaling af træningstid er 10 min <p>3. Vælg "KONDITIONSTRÆNING" og tryk videre. Vælg herefter "CYKEL" og tryk videre. Vælg "MEGET GODT" og tryk videre.</p> <ul style="list-style-type: none"> ✓ Forventet anbefaling af træningstid er 50 min
Resultat	

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 tidlige 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 Tidlige evaluering af samme træning er ":)". Vælg "KONDITIONSTRÆNING" og tryk videre. Vælg herefter "GÅ" og tryk videre. Vælg "MODERAT" og tryk videre.</p> <ul style="list-style-type: none"> ✓ Forventet anbefaling af træningstid er 30 min <p>2. Tidlige evaluering af samme træning er ":D". Vælg "KONDITIONSTRÆNING" og tryk videre. Vælg herefter "LØB" og tryk videre. Vælg "MEGET DÅRLIGT" og tryk videre.</p> <ul style="list-style-type: none"> ✓ Forventet anbefaling af træningstid er 20 min <p>3. Tidlige evaluering af samme træning er ":-(". Vælg "KONDITIONSTRÆNING" og tryk videre. Vælg herefter "CYKEL" og tryk videre. Vælg "MEGET GODT" og tryk videre.</p> <ul style="list-style-type: none"> ✓ Forventet anbefaling af træningstid er 40 min
Resultat	

Tabel 8.5: Test af tilpasning af træningsniveau med evaluering

Træning

Træning skal muliggøre måling af biologiske målinger 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 samt** håndtere målinger fra kompatible måleenheder

*Dette er nødvendigt for at muliggøre måling af biologiske målinger under træning **Det er vel ikke biologiske målinger længere***

- Brugere skal kunne evaluere hver træning

Dette er nødvendigt for at tilpasse træningen efter den enkelte bruger

For at test, 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, afstand og sensor samt evaluere træningen efterfølgende.
Main flow:	<ol style="list-style-type: none"> 1. 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 2. 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 1.572 km. ✓ Forventet afstand ved 0.02 er 3.145 km. 3. 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.
Resultat	

Tabel 8.6: Test af træning

Resultater

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

- **Systemet skal kunne sende notifikationer - synes ikke rigtig det passer ind længere og 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 5 minutter og tryk “STOP TRÆNING” evaluer træningen til “:D”. Tryk herefter på “RESULTATER” via hovedmenuen og vælg “BELØNNINGER”.</p> <ul style="list-style-type: none"> ✓ Forventet stjerner under tid er en. <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 0.2. Tryk “STOP TRÆNING” evaluer træningen til “:;)”. Tryk herefter på “RESULTATER” via hovedmenuen og vælg “BELØNNINGER”.</p> <ul style="list-style-type: none"> ✓ Forventet stjerner under afstand er fire. <p>3. Tryk på “RESULTATER” via hovedmenuen og vælg “BELØNNINGER”</p> <ul style="list-style-type: none"> ✓ Forventet stjerner under antal træninger er to. ✓ Forventet stjerner under konditionstræning.
Resultat	

Tabel 8.7: Test af resultater

Venneliste

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

- Brugere skal kunne følge andre brugere og tilgå hinandens belønninger
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 eksistere, hvorefter denne bruger følges.
Main flow:	<p>1. Tryk "VENNELISTE" via hovedmenuen og indtast medlemsID "1234567890" og tryk søg.</p> <ul style="list-style-type: none"> ✓ Forventet søgning mislykket. Fejlmeddeelse vises i grænsefladen for venneliste. <p>2. Tryk "VENNELISTE" via hovedmenuen og indtast medlemsID "1234567891" og tryk søg.</p> <ul style="list-style-type: none"> ✓ Forventet søgning lykkes. Grænsefladen for søgte vens belønninger vises. <p>3. Tryk "VENNELISTE" via hovedmenuen og indtast medlemsID "1234567891" og tryk søg. Herefter trykkes følg.</p> <ul style="list-style-type: none"> ✓ Forventet følg knap forsvinder i grænsefalden for vennelisten. <p>4. Tryk "VENNELISTE" via hovedmenuen</p> <ul style="list-style-type: none"> • Forventet bruger med medlemsID "1234567891" vises i grænsefladen forvenne-listen.
Resultat	

Tabel 8.8: Test af venneliste

Redigering

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

- Brugere skal kunne redigere deres adgangskode
Dette er nødvendigt for, at brugere kan ændre adgangskode

Til 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 adgangskode og to ens adgangskoder.
Main flow:	<p>1. Tryk "REDIGER ADGANGSKODE" via hovedmenuen. Indtast "nyadgangskode" i ny adgangskode og indtast "adgangskode" i gentag adgangskoden . Tryk "GEM ÆNDRINGER".</p> <ul style="list-style-type: none"> ✓ Forventet ændring mislykkes. Fejlmeddeelse vises i grænsefladen for redigering. <p>2. Tryk "REDIGER ADGANGSKODE" via hovedmenuen. Indtast "nyadgangskode" i ny adgangskode og gentag adgangskode. Tryk "GEM ÆNDRINGER".</p> <ul style="list-style-type: none"> ✓ Forventet ændring lykkes. Meddeelse viser at adgangskoden er gemt.
Resultat	

Tabel 8.9: Test af redigering

Log ud

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

- Brugere skal kunne log ud af app'en

Dette er nødvendigt for sikre brugerens 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	

Tabel 8.10: Test af log ud

Kapitel 9

Syntese

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.

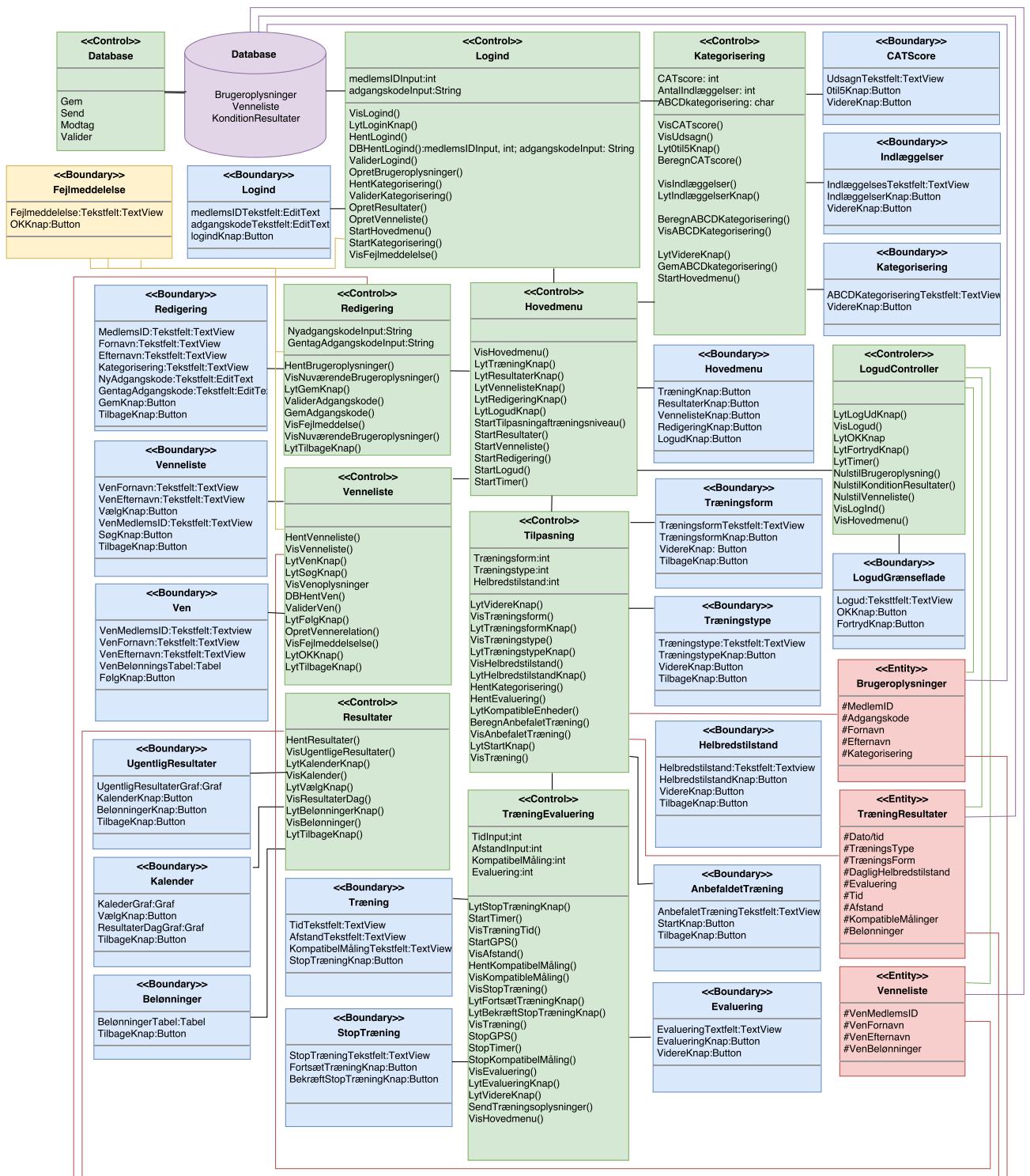
- [17] McCarthy B. et al. Pulmonary rehabilitation for chronic obstructive pulmonary disease. *Cochrane Library*, 2015.
- [18] De specialeansvarlige lungemedicinere i Storstrømmens Sygehus. KOL, behandling, udredning. *Sundhed.dk*, 2013. URL <https://www.sundhed.dk/sundhedsfaglig/information-til-praksis/sjaelland/patientforloeb/forloebsbeskrivelser/r-luftveje/kol/>.
- [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