

Applikation til opretholdelse af regelmæssig træning  
efter endt rehabilitering for 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 Øst

Synopsis:

Titel:

Applikation til opretholdelse af regelmæssig træning efter endt rehabilitering for patienter med kronisk obstruktiv lungesygdom

Tema:

Design af sundhedsteknologiske systemer

Projektperiode:

P6, Foråret 2017

Projektgruppe:

17gr6403

Medvirkende:

Birgithe Kleemann Rasmussen  
Linette Helena Poulsen  
Mads Kristensen  
Maria Kaalund Kroustrup

Vejleder:

Hovedvejleder:  
Lars Pilegaard Thomsen

Sider:

Bilag:

Afsluttet: XX/05/2017

Chronic obstructive pulmonary disease (COPD) is the fourth most common cause of death in Denmark, where up to 430,000 people suffers from it. COPD is a disease that causes the patient to gradually lose lung function over time, to where there is no possibility of restoring it again. Since there is no treatment, COPD patients is offered to join a pulmonary rehabilitation program, where they get the chance to learn about COPD, smoking discontinuation, diet, medication and exercise. Pulmonary rehabilitation allows the patients to achieve a relief in symptoms, through regular exercise, in which they get a better utilization of the remaining lung function and a better physical function level. However, studies show that some of these patients are not able to maintain the symptom relieving effects as they fall back to previous habits and routines after 6 to 12 months post-rehabilitation. In this project an app has been developed as a means of addressing this issue. The intention of the app is to guide and motivate COPD patients to maintain regular exercise during post-rehabilitation. Based on the individual COPD patient's condition, the app adapts the timeline of the recommended exercise, so it will be best suitable for the patient. The app possess different functionalities, as a way of motivating the patients and remind them of regular exercise, like daily notifications and the ability to obtain achievements based on different aspects of exercise and joint interaction in which users of the app can follow each other's progress. However the app is currently a prototype, in which changes must be made before it can be implemented and tested in practice.

# Forord og læsevejledning

---

## Forord

Dette bachelorprojekt er udarbejdet af gruppe 17gr6403 på ingeniøruddannelsen Sundhedsteknologi på Aalborg Universitet i perioden 1. februar til 30. maj 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: *Bachelorprojektet er afslutningen på bacheloruddannelsen og den studerende skal kunne demonstrere evner, som er relevante for arbejdsmarkedet og for en videre videnskabelig uddannelse [1]*.

Projektgruppen retter tak til hovedevejleder Lars Pilegaard Thomsen for vejledning og feedback gennem projektperioden.

## Læsevejledning

Projektet er delt op i tre dele, herunder problemanalyse, problemløsning og synopsis. I problemanalysen analyseres den opstillede problemstilling, hvor problemløsningen omhandler analyse, design, implementering og test af et system. Der er udarbejdet to metodeafsnit, hvoraf det første beskriver strukturen af rapporten samt vidensindsamling. Det andet metodeafsnit omfatter metoden anvendt i problemløsningen. Projektet 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. Hvis centrale elementer fra figurer yderligere er beskrevet markeres dette med kursiv.

Rapporten er udarbejdet i L<sup>A</sup>T<sub>E</sub>X, og app'en er udviklet i Android Studio version 2.3.1. Af nedenstående link forekommer en demonstrationsvideo af den udarbejdede app.

[https://drive.google.com/drive/folders/0B3f\\_ZB7s3Su3VHRzYk5MODAzdzA](https://drive.google.com/drive/folders/0B3f_ZB7s3Su3VHRzYk5MODAzdzA)

# Indholdsfortegnelse

---

<b>Kapitel 1 Indledning</b>	<b>1</b>
1.1 Initierende problemstilling . . . . .	1
<b>Kapitel 2 Metode I</b>	<b>2</b>
2.1 Opbygning af rapporten . . . . .	2
2.2 Vidensindsamling . . . . .	3
 <b>I Problemanalyse</b>	 <b>4</b>
<b>Kapitel 3 Analyse</b>	<b>5</b>
3.1 Kronisk obstruktiv lungesygdom . . . . .	5
3.1.1 Symptomer . . . . .	6
3.1.2 Diagnose . . . . .	6
3.1.3 Behandling . . . . .	9
3.1.4 Prognose . . . . .	11
3.2 Rehabilitering af KOL-patienter . . . . .	11
3.2.1 Rehabiliteringsforløb . . . . .	11
3.3 Efter rehabiliteringsforløb . . . . .	12
3.4 Projektgrænsning . . . . .	13
3.5 Problemformulering . . . . .	13
 <b>II Problemløsning</b>	 <b>14</b>
<b>Kapitel 4 Metode II</b>	<b>15</b>
4.1 Objektorienteret programmering . . . . .	15
4.1.1 Unified Modellig Language . . . . .	16
4.2 Unified Process . . . . .	18
<b>Kapitel 5 Systemanalyse</b>	<b>20</b>
5.1 Systembeskrivelse . . . . .	20
5.2 Kravspecifikationer . . . . .	21
5.2.1 Use case . . . . .	22
5.3 Funktionalitet . . . . .	23
5.4 Analyseklasser . . . . .	35
<b>Kapitel 6 Systemdesign</b>	<b>37</b>
6.1 Designafgrænsning . . . . .	37
6.2 Objektorienteret design . . . . .	37
6.3 Design af database . . . . .	59

<b>Kapitel 7 Implementering</b>	<b>61</b>
7.1 Platform . . . . .	61
7.2 Database . . . . .	62
7.3 Grænseflader . . . . .	64
7.4 Model- og controllerklasser . . . . .	65
7.5 Tilpasning af træningsniveau . . . . .	65
7.6 Træning . . . . .	66
7.7 Resultater . . . . .	69
7.8 Venneliste . . . . .	70
<b>Kapitel 8 Test</b>	<b>72</b>
8.1 Database . . . . .	72
8.2 Log ind . . . . .	73
8.3 Kategorisering . . . . .	74
8.4 Tilpasning af træningsniveau . . . . .	76
8.5 Træning . . . . .	79
8.6 Resultater . . . . .	81
8.7 Venneliste . . . . .	83
8.8 Redigering af adgangskode . . . . .	84
8.9 Log ud . . . . .	85
<b>III Syntese</b>	<b>87</b>
<b>Kapitel 9 Syntese</b>	<b>88</b>
9.1 Diskussion . . . . .	88
9.1.1 Kravsspecifikationer . . . . .	88
9.1.2 Design . . . . .	89
9.1.3 Implementering . . . . .	90
9.2 Konklusion . . . . .	91
9.3 Perspektivering . . . . .	92
<b>Litteratur</b>	<b>93</b>
<b>Bilag A Relation mellem designklasser</b>	<b>96</b>

# Kapitel 1

## Indledning

---

Kronisk obstruktiv lungesygdom (KOL) er en inflammatorisk sygdom, 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, hvorved 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 er 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 de hyppigste årsager til KOL.[2, 4, 6, 7] Miljøpåvirkninger kan blandt andet være dårligt arbejdsmiljø eller opvækst i dårligt miljø. Dårligt arbejdsmiljø er eksempelvis arbejde med asbest, hvorimod dårligt miljø kan medvirke til, at barnets lunger ikke udvikler sig ordentligt. Disse faktorer kan derved resultere i en accelererende reduktion af lungefunktionen.[7]

Lungefunktionen nedsættes gradvist over mange år, hvilket gør, at KOL først kommer til udtryk sent i sygdomsforløbet. Dette kan resultere i, at patienter først opsøger 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] KOL-patienter oplever åndenød, hvilket kan medføre svage perifere muskler. Dette kan lede til en række komorbiditeter. Disse fremtræder som kardiovaskulære sygdomme, type-2 diabetes, osteoporose, lungecancer og muskelsvækkelæse. 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

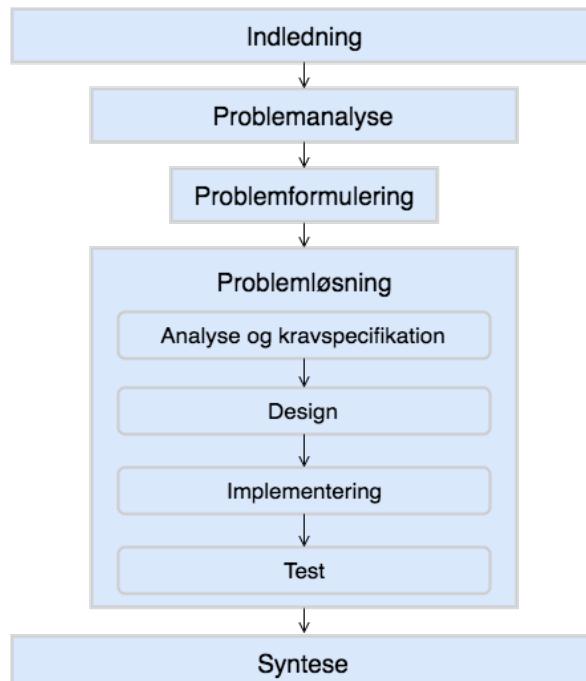
## Metode I

---

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. Metode til problemløsning forekommer senere af et andet metodeafsnit.

### 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, hvortil der opstilles kravspecifikationer. Et løsningsforslag vil herefter 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 projektkskrivning. Denne søgning foregik på Google og Primo, hvor 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 Primo, 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. Disse søgeord er anvendt alene samt i kombination.

# Del I

## Problemanalyse

# Kapitel 3

## Analyse

---

I dette kapitel beskrives KOL og de tilhørende symptomer. Yderligere undersøges det, hvordan KOL diagnosticeres samt, hvilke behandlingsmuligheder KOL-patienter tilbydes. Heraf analyseres KOL-patienters udbytte af gennemført 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 resulterer i bronkiernes vægge ødelægges og/eller luftvejene forsnævres. Dette medfører, at lungernes overflade reduceres samt blokeringer i luftvejene kan forekomme, hvilket forværret 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 støvpartikler 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 og 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, at huden bliver blålig, hvorfor disse patienter omtales blue bloater.[11]

Emfysem skyldes, at lungernes volumen er øget grundet beskadiget lungevæv, herunder sker en 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 falde 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]

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.[2, 4, 6, 7] Dertil har undersøgelser vist, at 35 – 40 % af tobaksrygere udvikler KOL [14]. Miljøpåvirkninger indebærer blandt andet opvækst i et dårligt miljø, der kan påvirke barnets lunger til ikke at udvikle sig ordentligt, hvilket kan resultere i en lavere lungefunktion. Derudover vil et dårligt arbejdsmiljø, som eksempelvis arbejde med asbest, kunne medvirke til en accelererende reduktion i lungefunktion, der ligeledes kan øge risikoen for KOL.[7]

### 3.1.1 Symptomer

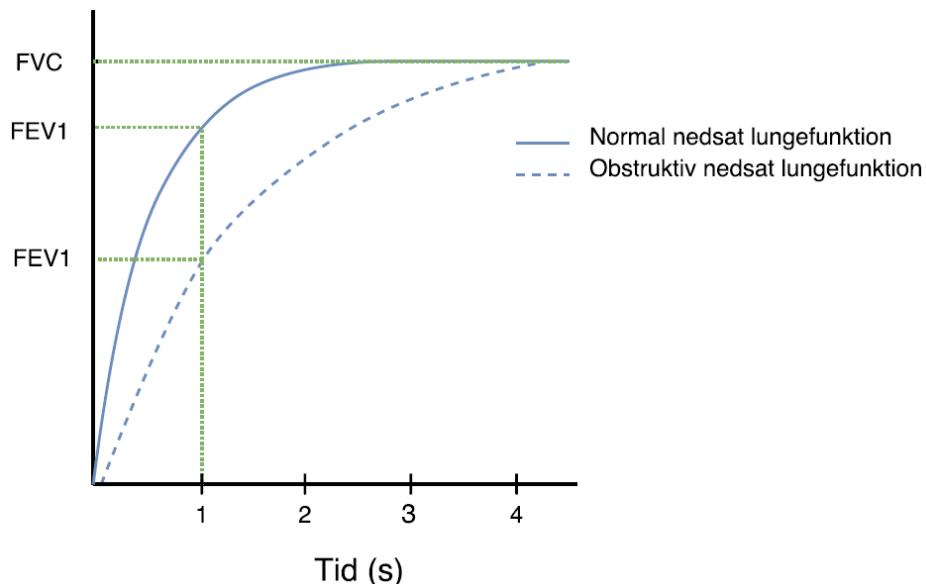
KOL udvikles over mange år, dog bemærkes sygdommen ofte ikke før lungefunktionen er markant nedsat. Dette betyder, at KOL og dens symptomer som regel først kommer til udtryk efter 50 årsalderen [15]. Dette kan i praksis betyde, at patienter først op søger læge, når deres lungefunktion er halveret [6].

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 i takt med udviklingen af KOL og kan tage op til flere uger, før patienten ikke længere er påvirket af eksacerbationen [16]. KOL-patienter klassificeret med moderat KOL oplever i gennemsnit 2,68 eksacerbationer per år, mens patienter med svær KOL oplever 3,43 eksacerbationer per år [16]. Symptomerne i forhold til eksacerbationer opleves som øget åndenød, hoste samt grønt eller gulligt ekspektoration og øget purulens. Halvdelen af disse tilfælde skyldes bakterielle infektioner.[2, 6]

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 komormiditer, 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 forceret eksspiratorisk volumen (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. KOL diagnosticeres ved ratioen mellem 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] Af figur 3.1 ses spirogrammålinger for henholdsvis patienter med normal og obstruktiv nedsat lungefunktion.[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 sammenlignet med normal lungefunktion. Dertil tager det længere tid for en KOL-patient at opnå FVC. For at sikre, at patienter ikke lider af differentialdiagnosen, asthma, udføres ligeledes en reversibilitetstest. Disse patienter gives broncodilatorer, som hos astmapatienter vil forbedre spirogrammålingen, 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 og sværhedsgrad af KOL

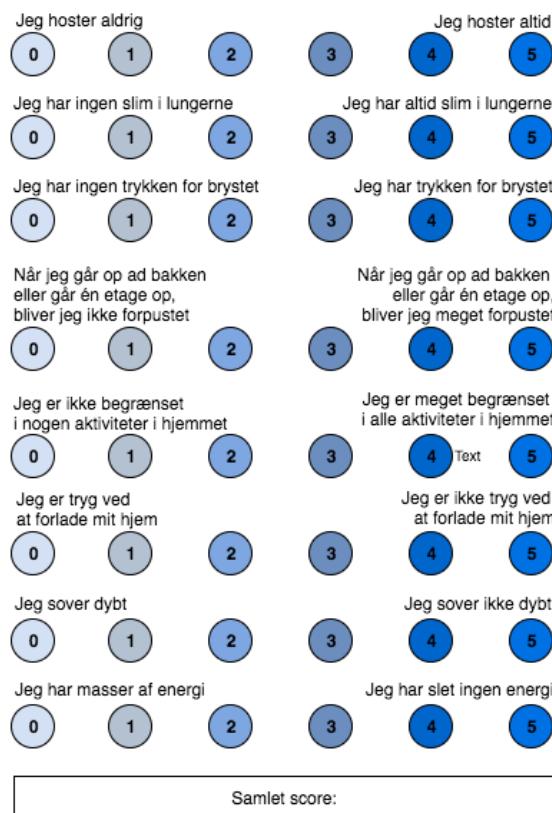
Sværhedsgraden af KOL vurderes på baggrund af patienters symptomer, egne erfaringer og livskvalitet. Denne vurderes ud fra Medical Research Council åndenødsskala (MRC) eller Chronic obstructive pulmonary disease Assessment Test (CAT). Patienter kan efterfølgende inddeltes i klassifikationer med udgangspunkt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

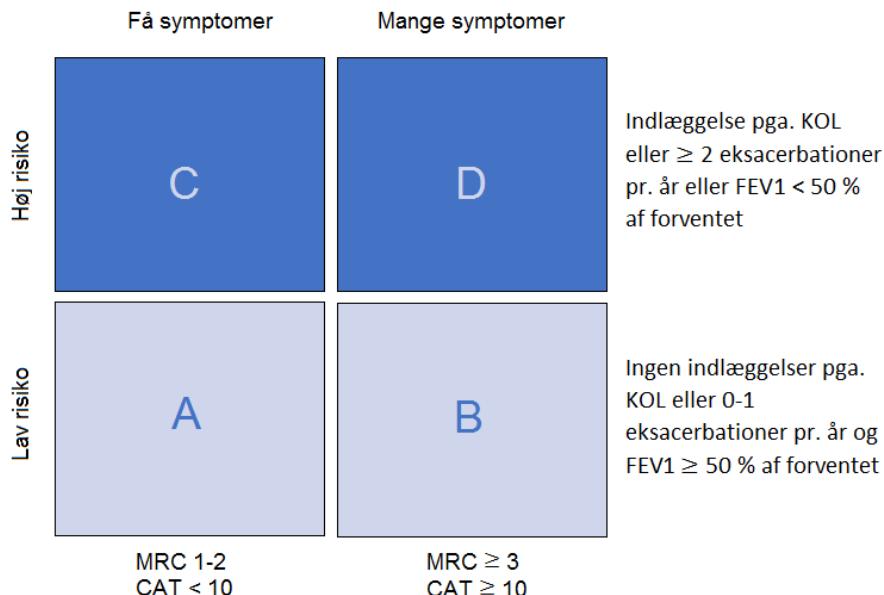
**Tabel 3.1:** MRC er en skala fra 1 til 5. Patienter, der oplever åndenød ved meget anstrengelse vurderes til 1, mens patienter, der oplever åndenød ved lav aktivitet vurderes til 5 på MRC-skalaen. Revideret [2].

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 samlet score, jo højere den samlede score er, desto værre opleves patienters symptomer. Af figur 3.2 ses CAT-spørgeskema til vurdering af symptomer.[2, 6]



**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 samlet 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 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 procent 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	$50\% \leq \text{FEV1} < 80\%$
3 GOLD Svær	$30\% \leq \text{FEV1} < 50\%$
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 under 50 % af forventet lungekapacitet samt respirationssvigt klassificeres som 4 GOLD meget svær. Revideret [2].

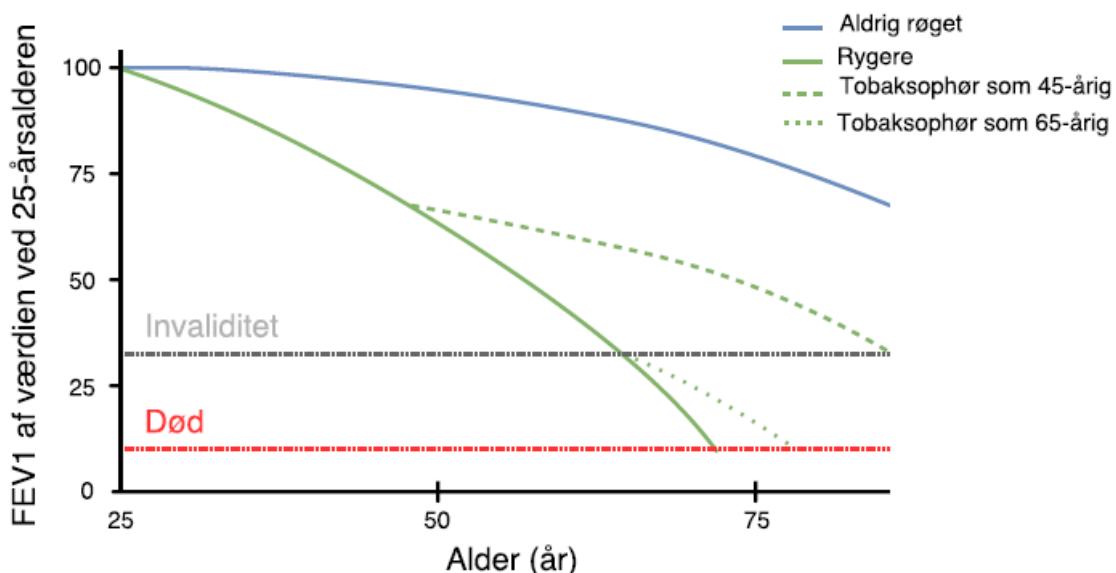
### 3.1.3 Behandling

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 [19]. 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].

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]

### Behandlingsomkostning

Sammenlignes KOL-patienter over 30 år med den resterende befolkning i Danmark, der ligeledes er over 30 år, er KOL-patienter mere ressourcekrævende for sundhedsvæsenet. Dette indebærer eksempelvis kontakter til sundhedsvæsenet og medicin. Udgiften hertil var fem gange højere for KOL-patienter end per borger generelt i år 2014. KOL-patienter havde i samme år i gennemsnit omkring tre gange så mange hospitalsindlæggelser samt et tre gange så højt medicinforbrug end den resterende befolkning.[8]

### 3.1.4 Prognose

Dødsfald hos KOL-patienter ses især efter 65-års alderen og udgør 90 % af alle dødsfald [14]. KOL-patienter med eksacerbationer har efter indlæggelse en dodelighed på næsten 10 % i løbet af den første måned. Dodeligheden ligger årligt på omkring 64 per 100.000 for mænd og 54 per 100.000 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 tilbydes KOL-patienter rehabilitering med henblik på at lindre deres symptomer, eksacerbationer samt hospitalindlæggelser [20, 21].

I Danmark henvises KOL-patienter til rehabilitering af 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 én til to gange om ugen, hvortil patienterne de resterende dage kan 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, hvorefter 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 relevant 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

Rehabiliteringsforløb 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

Til trods for, at rehabilitering viser positive resultater for de deltagende KOL-patienter, ses det gennem diverse studier og undersøgelser, at ikke alle patienter er i stand til at opretholde resultaterne. Efter et halvt til et helt år efter endt rehabiliteringsforløb falder deres fysiske tilstand 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]. Dertil har Lungeforeningen i Danmark forskellige lokalafdelinger, hvor der et par gange årligt afholdes arrangementer for patienter samt pårørende [3]. Fordelen 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 [27, 30]. Denne form for rehabilitering viser, at KOL-patienter oplever 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 lindre 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.[3, 17, 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 vedligeholdesesprogrammer, 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 ønskes det at udvikle en app med fokus på social interaktion og motivation til vedligeholdelse af resultaterne fra rehabiliteringsforløb.

### 3.5 Problemformulering

*Hvordan udvikles en app med henblik på at vejlede og motivere KOL-patienter til regelmæssig træning i forlængelse af rehabiliteringsforløb med ønsket om at lindre symptomer forbundet med KOL?*

# Del II

# Problemløsning

# Kapitel 4

## Metode II

---

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 én eller flere instanser kan være defineret for den samme klasse. De forskellige objekter repræsenterer hver sin del af app'en og indeholder data samt logik. Derudover har objekterne mulighed for at kommunikere mellem hinanden. Objekter er karakteriseret ud fra deres attributter, og deres funktioner er beskrevet ved metoder.[32, 33] Eksempler på attributter og metoder fremgår af tabel 4.1.

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

*Tabel 4.1: Objekter karakteriseres ud fra deres attributter 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 illustrerer, at objekter både indeholder attributter og metoder. Attributter 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

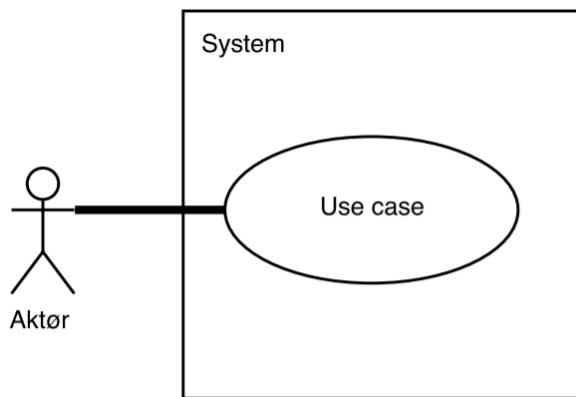
funktioner. Polymorfi giver mulighed for, at to klasser kan have samme grænseflade og er defineret ved nedarvningen.[32]

#### 4.1.1 Unified Modellig Language

En af de anvendte standarder indenfor objektorienteret programmering er 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.



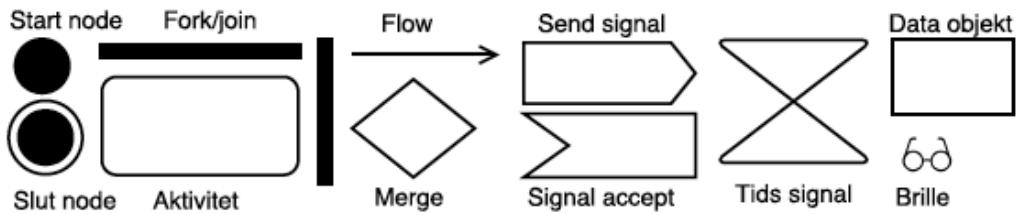
*Figur 4.1: Simpelt use case diagram.*

Af figur 4.1 ses aktørens interaktion med use case visualiseret som en streg mellem de to. I et use case diagram vil aktøren kunne tilgå systemets funktionaliteter. Aktøreren kan eksempelvis være en person, rolle, objekt eller 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 kanstå alene og derfor er nødt til at arve noget fra en anden use case er denne include. Modsat kan extend anvendes, hvis use casen kanstå 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. Aktivitetsdiagrammer fortæller ikke hvem, der udfører aktiviteten, hertil kan der anvendes skillevægge, som viser, hvilken aktivitet en klasse tilhører. For at holde et ak-

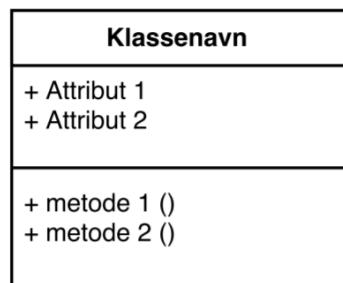
tivitetsdiagram enkelt kan der anvendes et brillesymbol i en aktivitet. Denne aktivitet vil efterfølgende kunne beskrives yderligere i et nyt aktivitetsdiagram.[34] Symboler, der kan anvendes inden for aktivitetsdiagrammer, fremgår af figur 4.2.



*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. Parenteserne angiver en metode, hvori der kan defineres inputsparametre.

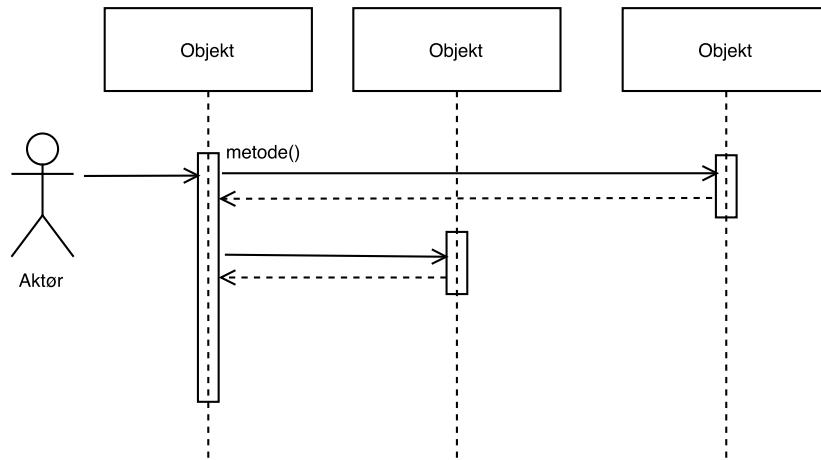
Relationerne mellem klasserne illustreres ved brug af forskellige pile, og disse kan navngives for at tydeliggøre forholdet mellem klasserne.

Klassediagrammer kan inddeltes i tre typer, herunder entity, boundary og control, hvilket anvendes til at identificere klasser i analyse og tidlig designfase [36].

- *Entity* anvendes til at lagre og opdatere informationer om objekter. Dens attributværdier gives ofte af en aktør.[36]
- *Boundary* anvendes til interaktion mellem bruger og system og sikrer, at ændringer i boundary ikke påvirker resten af systemet [36].
- *Control* anvendes til at kontrollere handlinger [36].

## Sekvensdiagrammer

Sekvensdiagrammer anvendes til at beskrive detaljer om, hvilke og hvornår forskellige operationer udføres. Disse diagrammer er organiseret efter flow. 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].

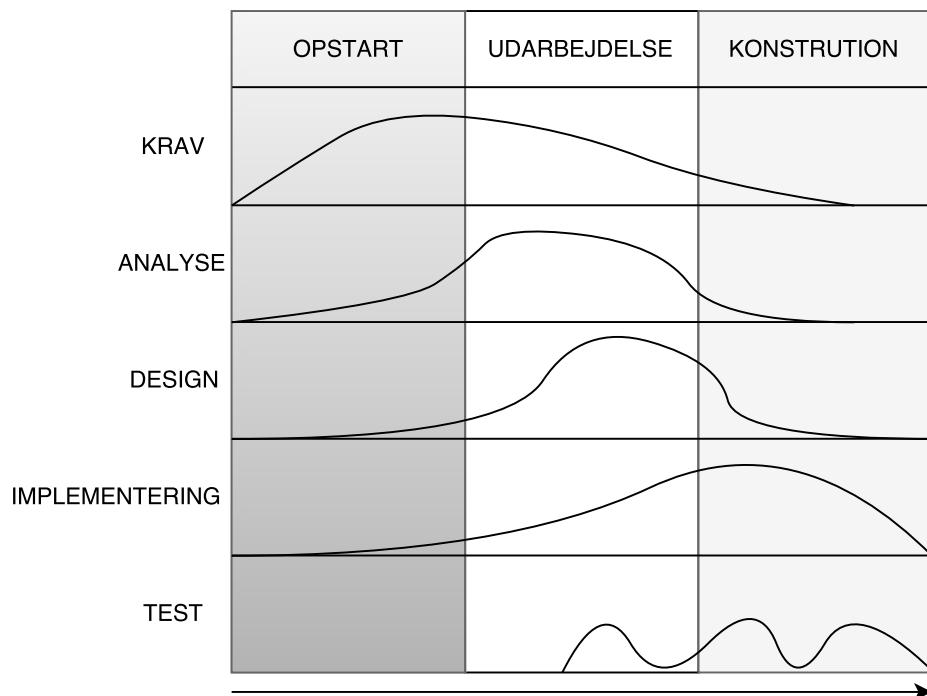
Den vertikale retning af figur 4.4 beskriver flowet, og de horisontale linjer illustrerer funktionaliteter. Over de horisontale pile er der navngivet metoden på objektet. Hver objekt har en livslinje, hvorpå der kan illustreres en aflang boks, som beskriver, hvornår objektet er aktivt. [33]

Sekvensdiagrammer kan opdeles i Model-View-Controller (MVC) arkitektur, der har ligheder med de definerede typer 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 op 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.[37] Den sidste fase, overgang, er udeladt af dette projekt, da der kun udvikles en prototype. 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 fordelingen af arbejdsmængden. Revideret [37].

Af figur 4.5 fremgår softwareprocesudviklingen i dette projekt. Opstart og udarbejdelse anvendes med henblik på den senere implementering i konstruktionsfasen. I forbindelse med analyse, design og implementering er der opstået iterationer, hvilket har medført ændringer i app-udviklingen og derved arbejdsmængden i de forskellige faser.

# 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 yderligere beskrevet ved et use case diagram, hvortil de enkelte funktionaliteter er beskrevet. Til sidst analyseres substantiver og verber for at udarbejde analyseklasser.

### 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 symptomlindrende effekt, jf. afsnit 3.2. Ligeledes skal app'en kunne håndtere forskellige træningstyper, som for eksempel gå, løb og cykel.

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

Der er forskel på, hvor meget fysisk aktivitet KOL-patienter kan udføre, og der skal derved være forskel i varighed af den træning som app'en foreslår. Dertil skal app'en kunne tilpasse træningsniveau ud fra den enkelte patients parametre. Disse parametre består af kategoriseringen af KOL-patienter efter ABCD, jf. afsnit 3.1.2, daglige helbredstilstande, der skal tage højde for dag til dag variationer og tidlige evalueringer fra lignende træning. Dette medvirker til, at app'en henvender sig specifikt til den enkelte KOL-patient, idet træningsniveauet tilpasses alt efter, hvordan patienten har det den pågældende dag.

Under selve træningen monitoreres træningen ved brug af timer og GPS. Timeren har til formål at oplyse brugeren om træningstiden, hvortil GPS'en oplyser brugeren om den tilbagelagte afstand. Dette er med henblik på at vejlede patienter til at følge det anbefalede træningsniveau.

For at hjælpe KOL-patienter med vedligeholdelse af regelmæssig træning skal app'en virke motiverende for patienterne. Dette gøres blandt andet ved, at KOL-patienter kan se sin ugentlige træning samt opnå virtuelle belønninger ved at udføre gentagne eller forskellige træningsformer. Desuden skal app'en påminde brugeren om daglig træning.[38, 39]

Som nævnt i afsnit 3.3, er det sociale fællesskab en væsentlig faktor for at opretholde resultaterne fra rehabiliteringsforløb. For at inddrage dette aspekt i app'en, skal KOL-patienter være i stand til at kunne følge andre KOL-patienter og tilgå deres virtuelle belønninger. 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.[38, 39]

## 5.2 Kravspecifikationer

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

### Funktionelle krav

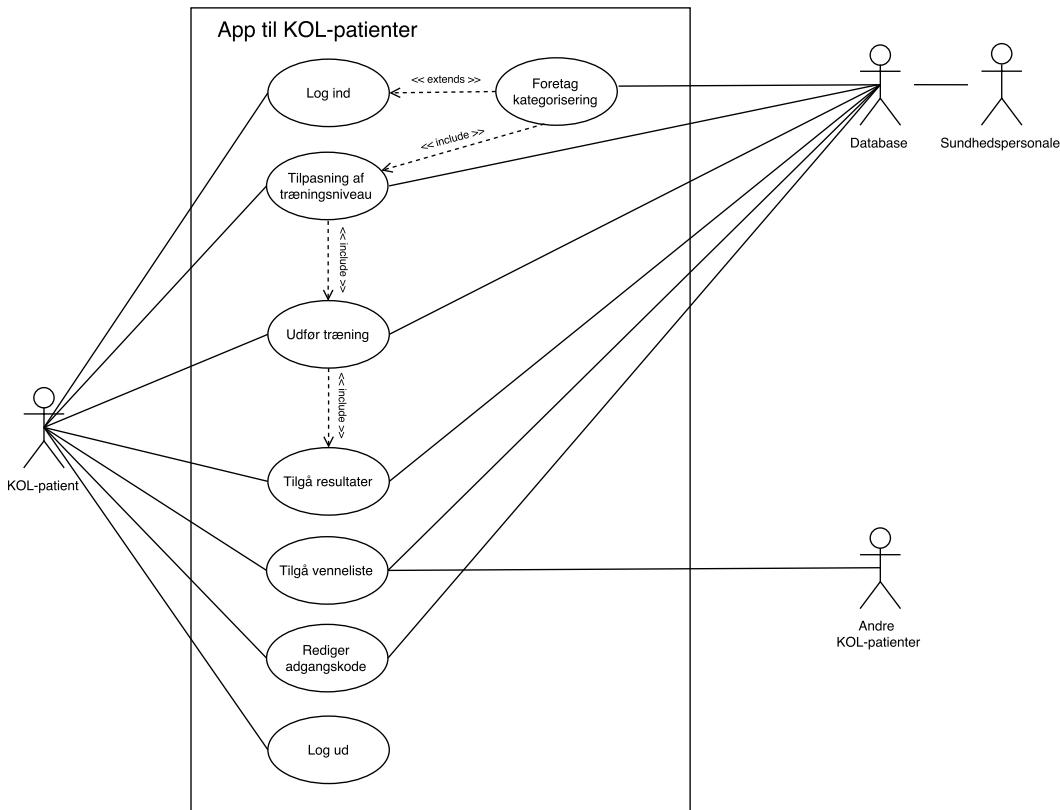
- Brugere skal kunne oprettes i en database  
*Dette er nødvendigt for, at brugere kan anvende app'en*
- Systemet skal kunne gemme og hente data i en database  
*Dette er nødvendigt for, at brugere kan tilgå brugerdata*
- Brugere skal kunne logge ind med et personligt medlemsID og adgangskode  
*Dette er nødvendigt for at tilgå og sikre, at brugere har deltaget i et rehabiliteringsforløb samt adskille brugernes data*
- Systemet skal kunne kategorisere brugere i ABCD på baggrund af CAT-score og antallet af indlæggelser på grund af KOL  
*Dette er nødvendigt for at kunne tilpasse træningen efter den enkelte bruger*
- Brugere skal kunne angive deres daglige helbredstilstand  
*Dette er nødvendigt for tage højde for daglige variationer og derved tilpasse træningen for den enkelte bruger*
- Systemet skal kunne måle tid via timer og afstand via GPS  
*Dette er nødvendigt for at monitorere træningen*
- Brugere skal kunne evaluere hver træning  
*Dette er nødvendigt for at tilpasse træningen yderligere efter den enkelte bruger*
- Systemet skal kunne sende en daglig notifikation for således at påminde brugeren om træning  
*Dette er nødvendigt for, at kunne motivere brugere til at udføre træning*
- Systemet skal kunne vise brugerens ugentlige træningsudvikling  
*Dette er nødvendigt for at brugere kan følge sin ugentlige udvikling samt motivere brugeren*
- Systemet skal kunne give virtuelle belønninger  
*Dette er nødvendigt for at kunne motivere brugere til at udføre træning*
- Brugere skal kunne følge andre brugere  
*Dette er nødvendigt for at skabe fællesskab samt gøre det muligt for brugere at tilgå hinandens virtuelle belønninger, hvilket skal øge brugerens motivation*
- Brugere skal kunne redigere deres adgangskode  
*Dette er nødvendigt for, at brugere skal kunne gøre deres adgangskode personlig*
- Brugere skal kunne logge ud af app'en  
*Dette er nødvendigt for at sikre brugerens individuelle data og give brugeren mulighed for at logge ud*

### Non-funktionelle krav

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

### 5.2.1 Use case

På baggrund af systembeskrivelsen samt opstillede krav er der udarbejdet et use case diagram, der beskriver app'ens funktioner. Af use case diagrammet på figur 5.1 ses systemet, *app til KOL-patienter*, samt de forskellige use cases og aktører, der interagerer med systemet. KOL-patienten er den primære aktør, som kan tilgå alle use cases. Database og andre KOL-patienter er sekundære aktører og kan kun tilgå enkelte use cases. Sundhedspersonalet har kun adgang til data via en database.



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

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

Førend brugeren kan udføre træning foretages *Tilpasning af træningsniveau*, der vurderes på baggrund af kategoriseringen, daglig helbredstilstand samt tidligere evaluering baseret ud fra samme træning samt helbredstilstand. Efter træningsniveauet er tilpasset kan brugeren *Udfør træning*. Efter træningen skal brugere evaluere træningen. Evaluering og resultater fra træningen gemmes efterfølgende i databasen. Brugere kan *Tilgå resultater*, der viser brugerens umentlige udvikling samt virtuelle belønninger, som opnås på baggrund af udførte træninger.

Brugere kan via *Tilgå venneliste* tilgå andre KOL-patienters virtuelle belønninger, hvilket medvirker til, at brugere kan motivere hinanden. Brugere kan *Rediger adgangskode*, da brugeren skal kunne ændre denne til en personlig adgangskode. Hvis der foretages ændringer gemmes disse efterfølgende i databasen. Brugeren kan *Log ud* af app'en, hvis dette ønskes.

### 5.3 Funktionalitet

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

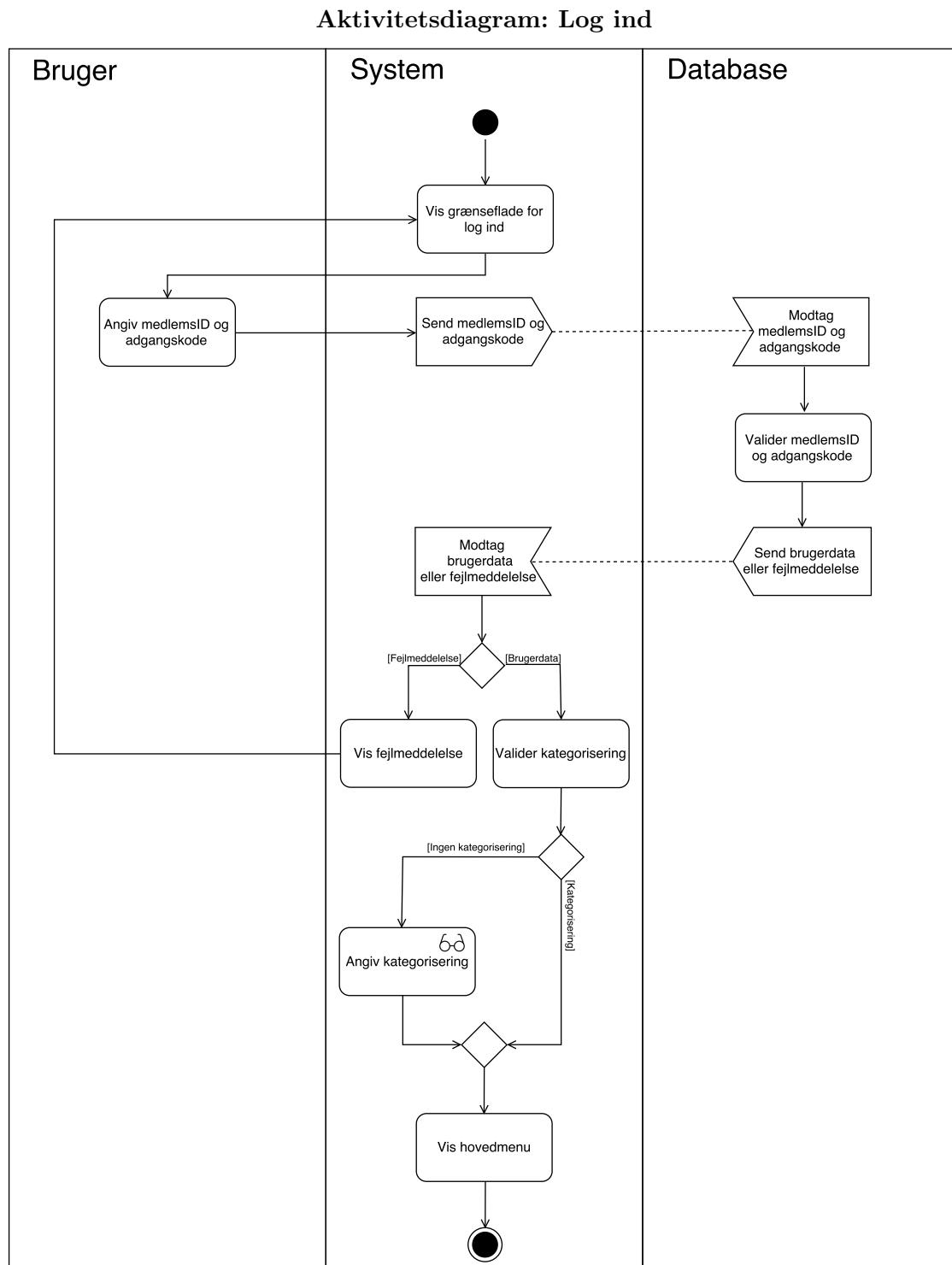
#### Registrering

Inden KOL-patienter kan anvende app'en, skal de registreres som brugere af systemet. Dette skal foregå i forbindelse med rehabiliteringsforløbet, hvor sundhedspersonalet opretter patienterne i databasen. Patienterne får tildelt et personligt medlemsID og en randomiseret adgangskode. Dette er vigtigt for at reducere risikoen for misbrug af personlige oplysninger, da uvedkommende kan have mulighed for at tilgå informationerne via netadgang eller enheden [40]. MedlemsID'et skal bestå af tal, som er sammensat ud fra lokalisering, årstal og måned for påbegyndt rehabiliteringsforløb samt nummerering af den enkelte KOL-patient, eksempelvis **11170301**.

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-patienter med at kategorisere patienters 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 anbefalet af app'en. Der er desuden mulighed for at kunne få besvaret eventuelle tvivlsspørgsmål, der kan opstå første gang app'en anvendes.

#### Log ind

I systemet benyttes en log ind-funktion til at beskytte og identificere den enkelte bruger. Brugeren skal her angive log ind-information, der vil tillade adgang til brugerinformation i form af private oplysninger og 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.*

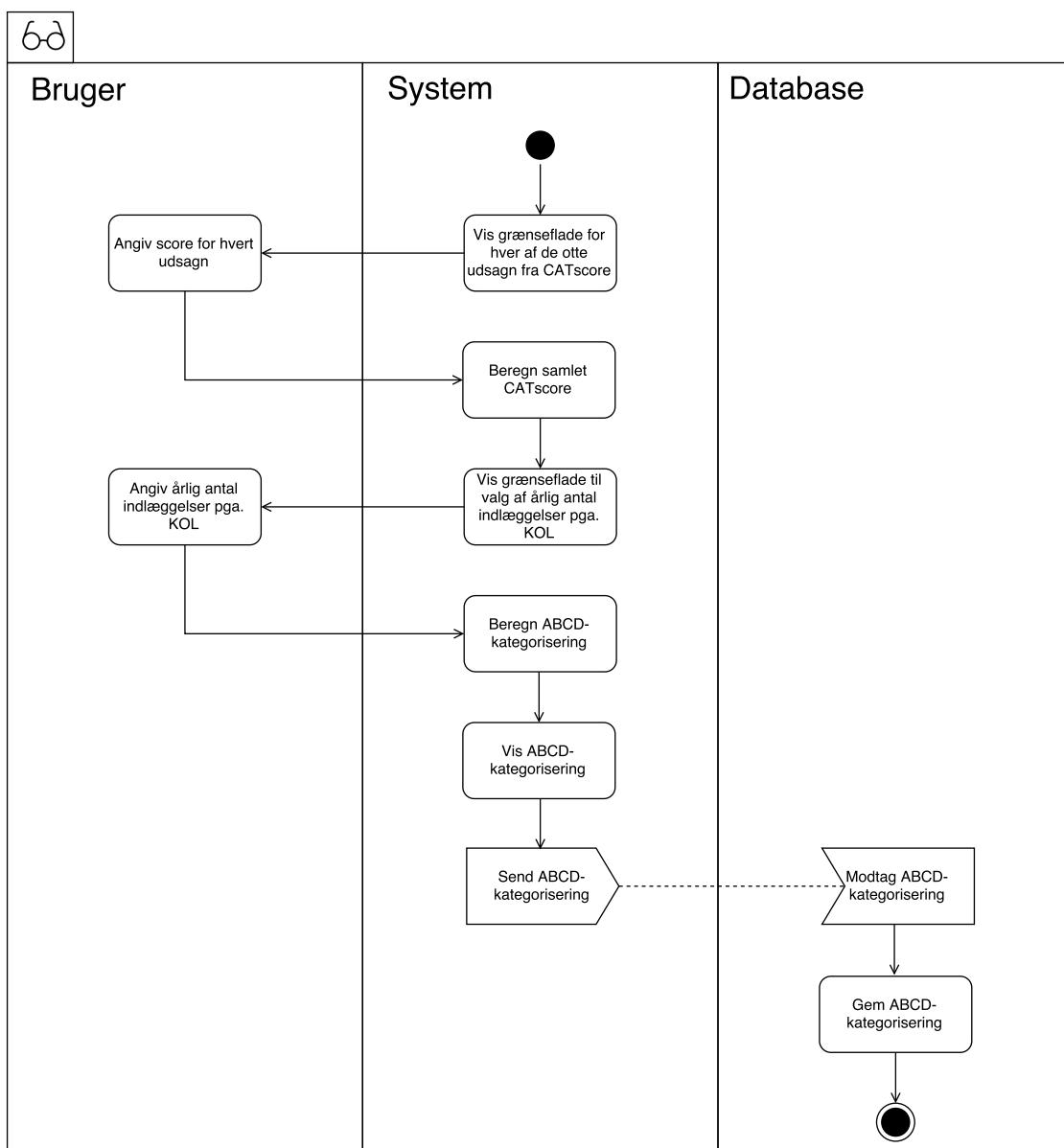
En grænseflade for log ind er det første der vises, når brugeren første gang åbner app'en. Brugeren kan i denne grænseflade angive sit medlemsID samt adgangskode, hvorefter systemet sender de indtastede log ind-informationer til databasen. Databasen validerer log ind-informationer og sender brugerdata eller en fejlmeldelse, hvis medlemsID'et med tilhørende adgangskode ikke eksisterer i databasen. Sendes en fejlmeldelse, hvorefter brugeren igen har mulighed for at indtaste sit medlemsID samt adgangskode. Har brugeren glemt sine log ind-

informationer, bedes brugeren kontakte sundhedspersonalet. Er de indtastede informationer ens med informationerne i databasen, logges brugeren ind, hvormed brugeroplysninger hentes fra databasen. Når brugerdata er hentet, validerer systemet om brugeren har en kategorisering. Har brugeren ingen kategorisering, skal brugeren foretage en kategorisering, der fremgår af figur 5.3, ellers vises hovedmenuen.

### Kategorisering

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

Aktivitetsdiagram: Kategorisering af KOL-patienter



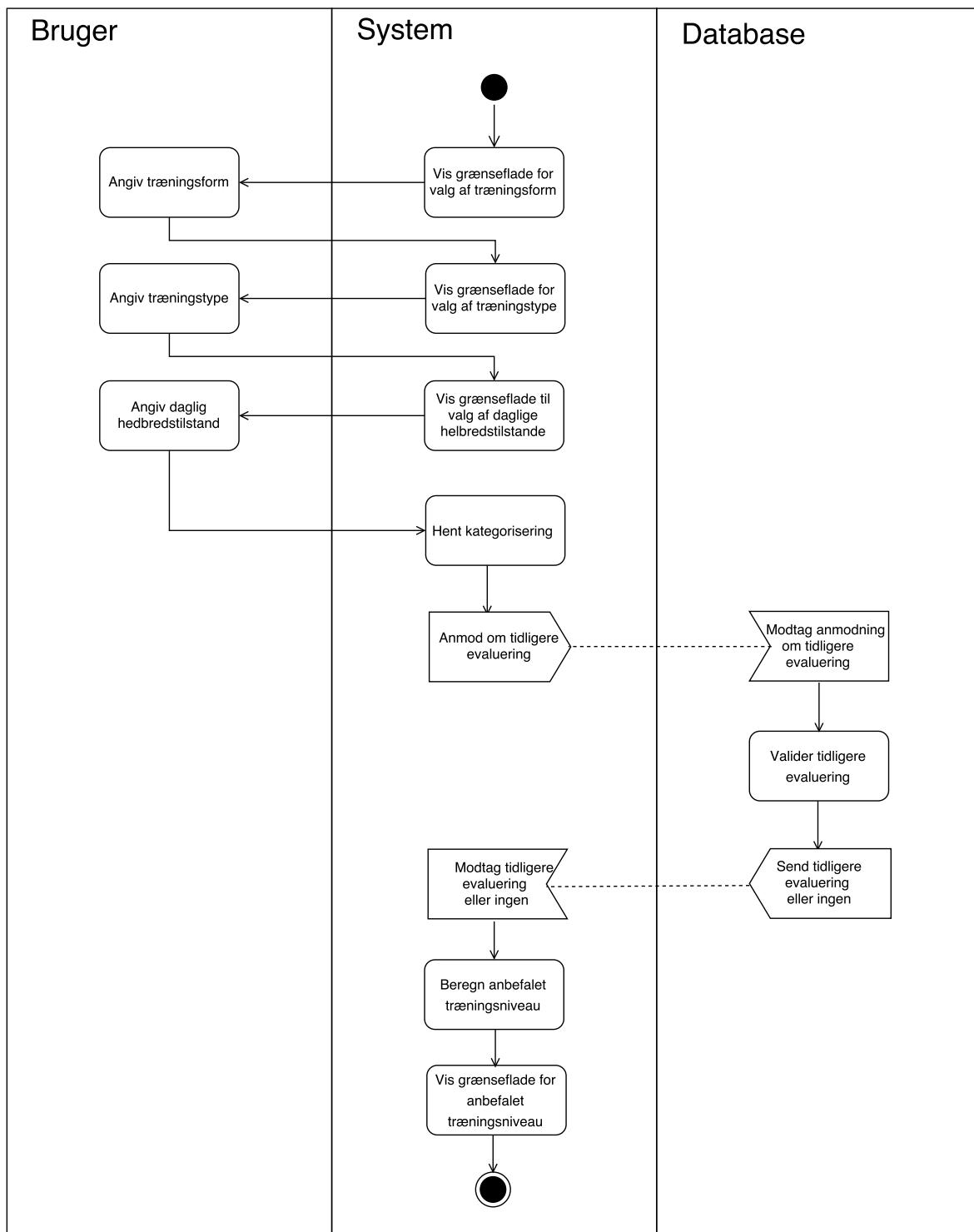
**Figur 5.3:** Aktivitetsdiagram for kategorisering af KOL-patienter.

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

### Tilpasning af træningsniveau

Træningen tilgås fra hovedmenuen, dog skal træningsniveauet tilpasses brugeren først træningen kan påbegyndes. Tilpasning af træningsniveau er en funktionalitet, der skal tage højde for daglige variationer ved at anbefale et træningsniveau ud fra brugerens kategorisering, daglig helbredstilstand og tidligere evaluering. Aktivitetsdiagrammet over tilpasning af træningsniveau fremgår af figur 5.4.

### Aktivitetsdiagram: Tilpasning af træning



*Figur 5.4: Aktivitetsdiagram over tilpasning af træningsniveau.*

Systemet viser grænsefladen for valg af træningsform, hvor brugeren skal angive den ønskede træningsform, herunder konditions-, styrketræning eller vejrrækningsøvelser. Ud fra den valgte træningsform skal brugeren angive træningstype, eksempelvis kan der ved valg af konditionstræning vælges gå, løbe eller cykle. Herefter vises grænsefladen for valg af daglige helbredstilstande, hvortil brugeren skal angive sin helbredstilstand. Systemet

henter kategorisering, hvorefter den anmoder om tidligere evaluering i databasen. Databasen validerer, om der er en tidligere evaluering passende til træningen samt helbredstilstanden. Hvis brugeren ikke har angivet tidligere evaluering, bestemmes niveauet ud fra de resterende parametre. Et simpelt eksempel på denne beregning fremgår af tabel 5.1. Tabellen beskriver, hvordan en algoritme vil kunne tilpasse træningsniveauet, således der tages højde for den enkelte bruger.

Simpel beslutningstabell (uden evaluering)									
Kategorisering	A		B			C			D
Træningsform	Konditionstræning			Styrketræning			Vejtrækningssøvelser		
Træningstype	Gå	Løb	Cykel	Type 1	Type 2	Type 3	Type 1	Type 2	Type 3
Daglig Helbredstilstand	1: Meget dårligt		2: Dårligt		3: Moderat		4: Godt		5: Meget godt
Træningsniveau									

**Tabel 5.1:** En simpel beslutningstabell for tilpasning af træningsniveau uden tidligere evaluering.

Af tabel 5.1 fremgår en simpel beslutningstabell for, hvorledes et træningsniveau tilpasses den enkelte bruger. Beslutningstabellen tager udgangspunkt i brugerens kategorisering, valgt træningsform og -type samt daglig helbredstilstand. Brugeren er i dette tilfælde kategoriseret til B og har valgt at udføre konditionstræning, herunder løb. Helbredstilstanden angives fra 1 til 5, hvortil brugerens helbredstilstand i dette eksempel angives som 3. Den simple beslutningstabell udvælger dertil et træningsniveau. Dette træningsniveau kan yderligere tilpasses, hvis en passende evaluering tidligere er foretaget. En simpel beslutningstabell med en tidligere evaluering fremgår af tabel 5.2.

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

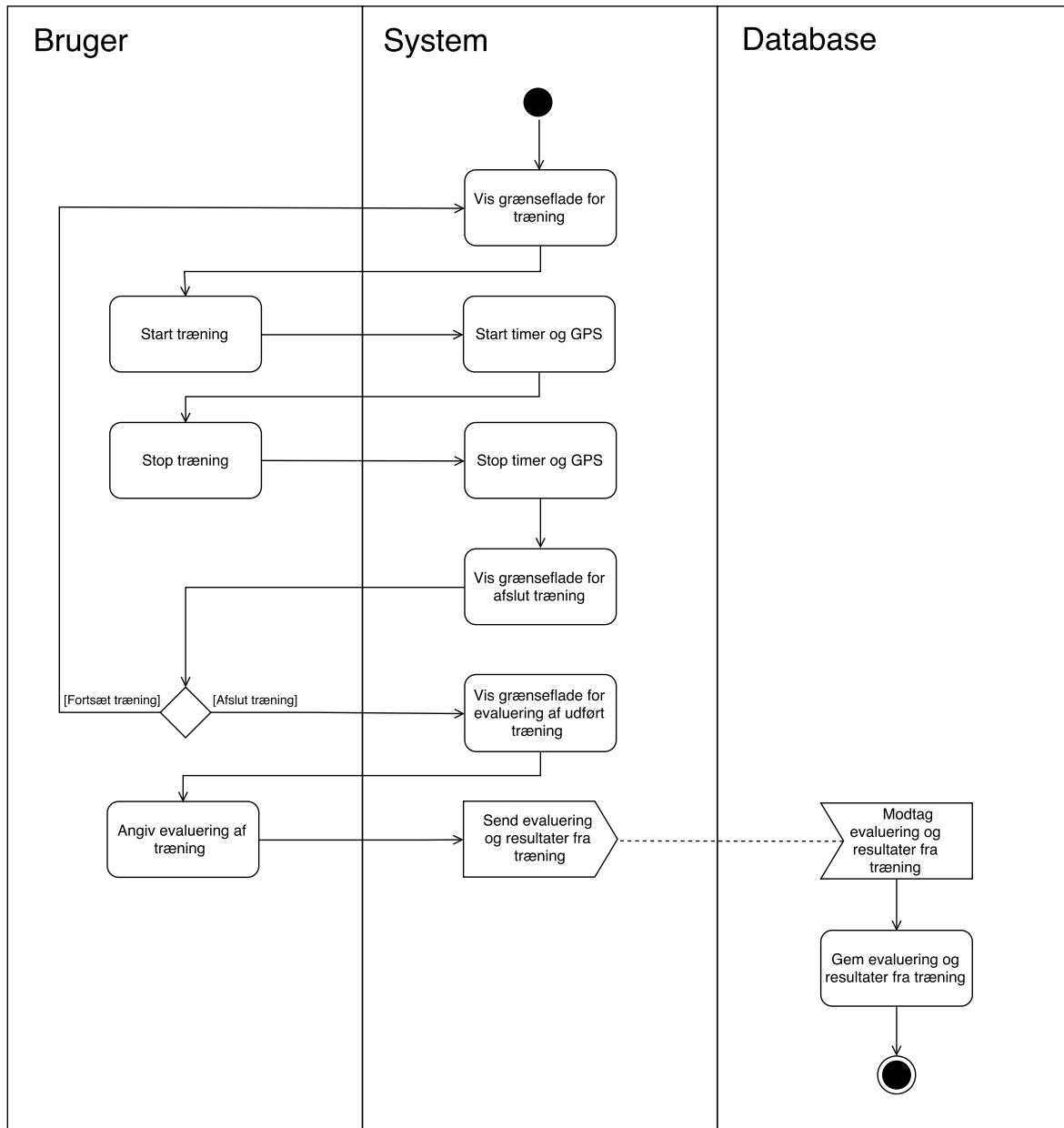
**Tabel 5.2:** En simpel beslutningstabell for tilpasning af træningsniveau med tidligere evaluering.

Af tabel 5.2 er træningsniveauet yderligere reguleret ud fra en tidligere evaluering passende til samme træning samt helbredstilstand. Brugeren har hertil samme kategorisering, træningsform samt -type og helbredstilstand som i eksemplet fra tabel 5.1. Dog er der hertil angivet en tidligere evaluering som værende hård. Dette medvirker til, at algoritmen regulerer træningsniveauet for denne træning, for således at give brugeren en bedre og mere tilpasset træning.

## Træning

Efter tilpasning af træningsniveauet kan brugeren påbegynde en træning. Aktivitetsdiagrammet over træningen fremgår af figur 5.5.

Aktivitetsdiagram: Træning



Figur 5.5: Aktivitetsdiagram over træning.

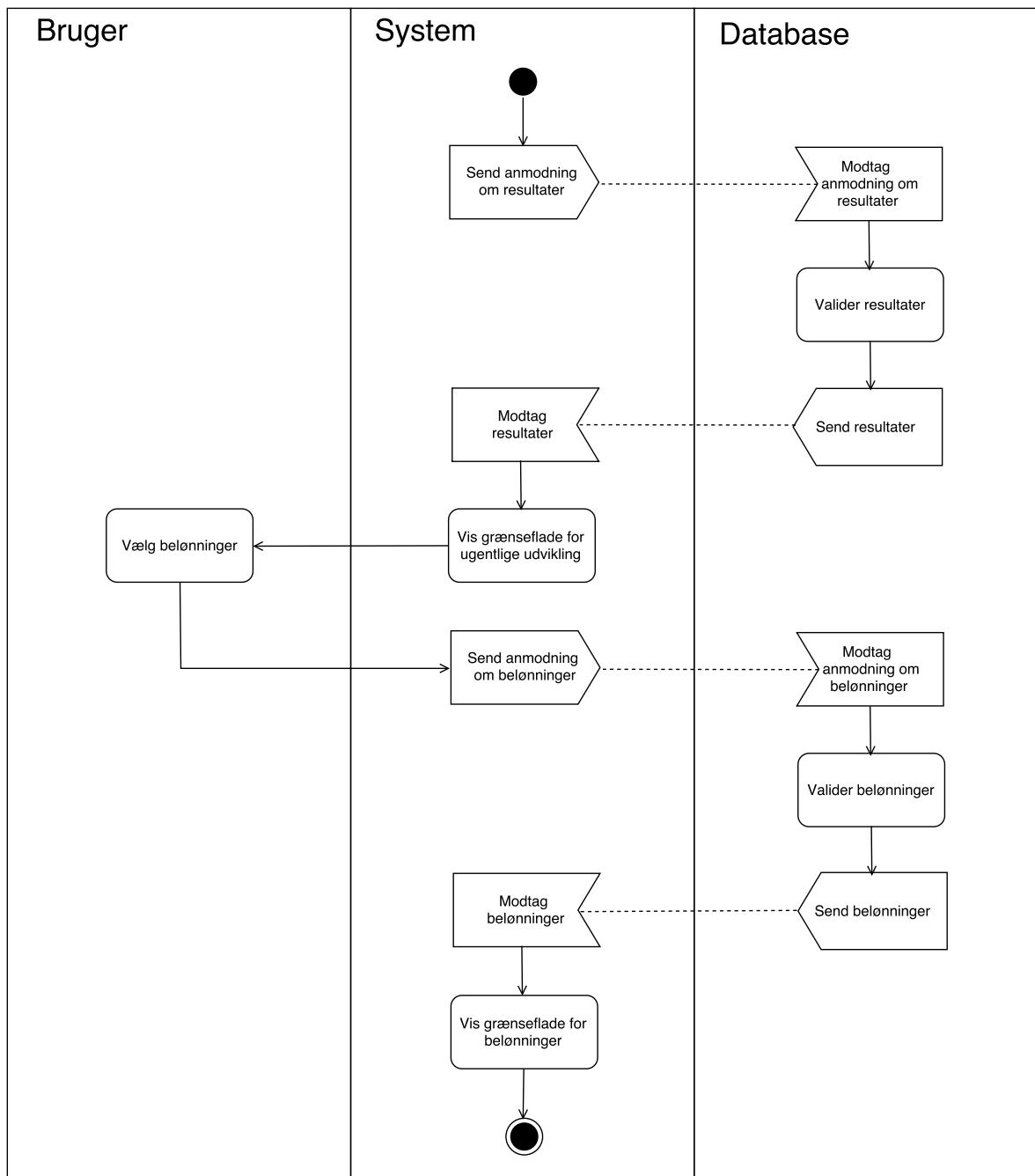
Når systemet har tilpasset træning niveau, vises grænsefladen for træning. Brugeren kan påbegynde træningen ved at trykke start, hvorefter timer og GPS starter. Under træningen vil tid og afstand kontinuert fremgå af grænsefladen. Brugeren kan til enhver tid vælge at afslutte træningen ved at trykke på stop, hvilket medfører, at timer og GPS stoppes og grænsefladen for afslut træning vises. Denne handling skal bekræftes i tilfælde af, at brugeren ved en fejl angiver, at træningen skal stoppes. Er dette tilfældet afsluttet træningen ikke og grænsefladen for træning vises igen. Ved bekræftelse af stop træning, vises grænsefladen for evaluering af

udført træning, hvor brugeren skal angive en evaluering. Efterfølgende sendes evalueringen og træningsresultater til databasen, hvor det gemmes.

## Resultater

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

**Aktivitetsdiagram: Resultater**



*Figur 5.6: Aktivitetsdiagram over resultater.*

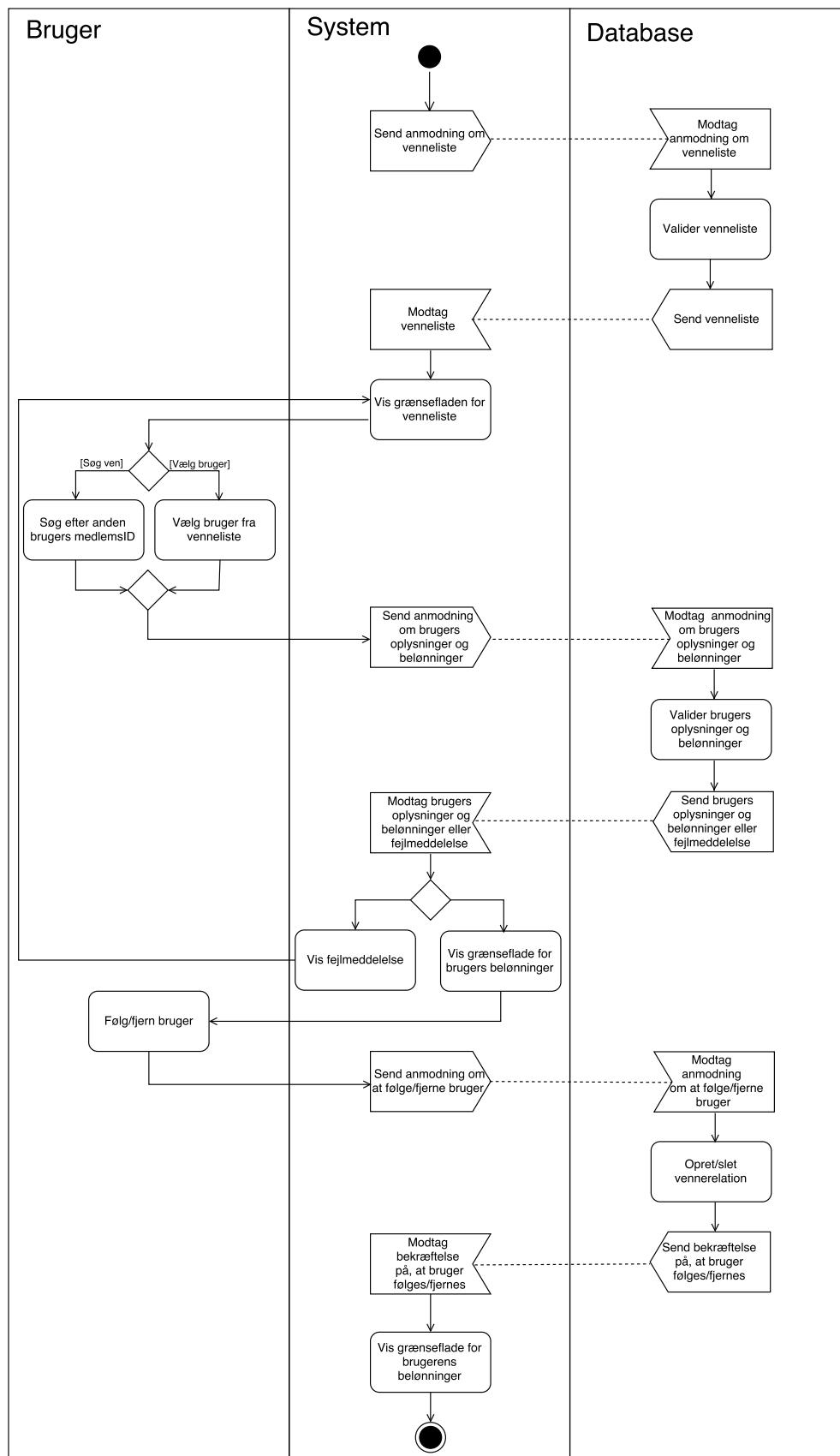
Under resultater er det muligt for brugeren at se sin ugentlige træning samt virtuelle belønninger. Idet brugeren tilgår resultater fra hovedmenuen, hentes resultater, der vises

som en grafisk udvikling, fra databasen. Brugeren har fra grænsefladen for grafisk udvikling mulighed for at tilgå sine belønninger. Ønskes dette, hentes brugerens belønninger fra databasen, hvorefter de visualiseres i grænsefladen for belønninger. Belønningerne varierer afhængig af træningsform, og der kan opnås forskellige belønninger inden for forskellige kategorier.

### Venneliste

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

### Aktivitetsdiagram: Venneliste



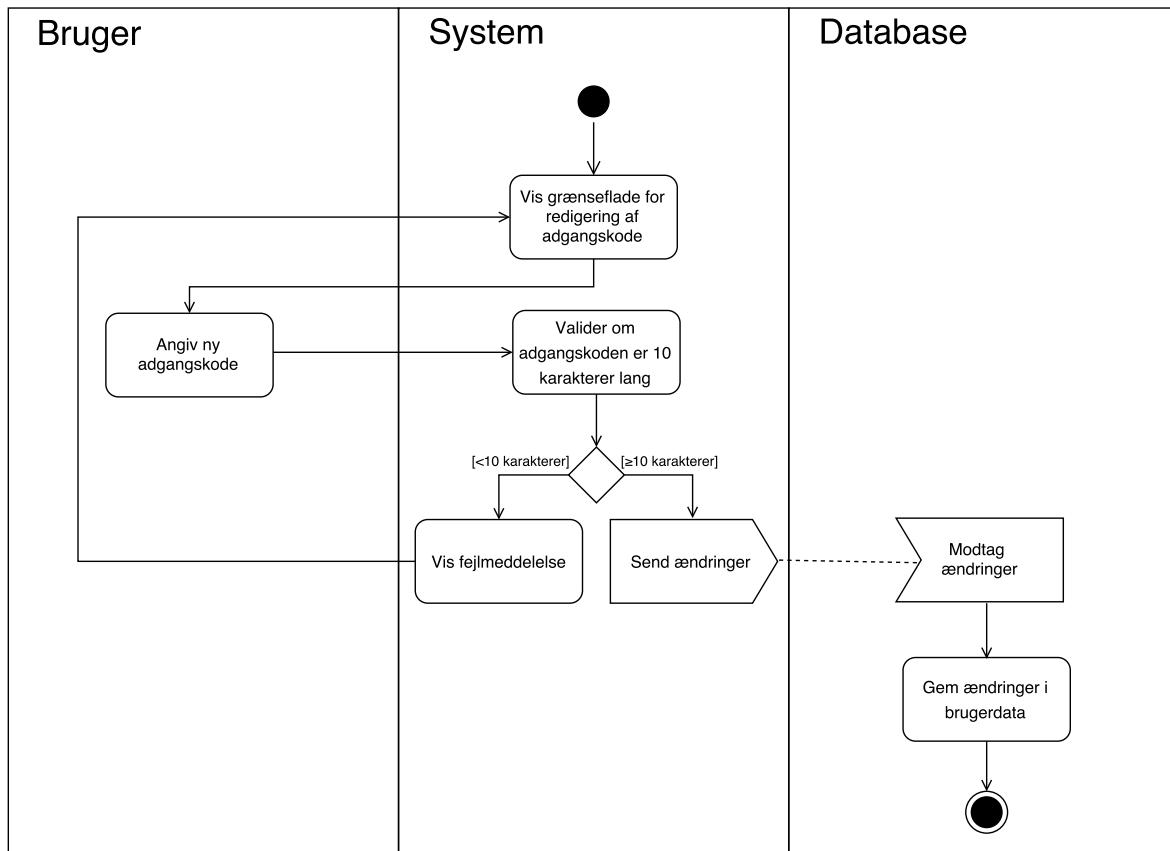
Figur 5.7: Aktivitetsdiagram for venneliste.

Tilgår brugeren sin venneliste fra hovedmenuen, hentes brugerens venner fra databasen, hvorefter de opstilles i en liste. Systemet viser hertil en grænseflade for vennelisten, der indeholder en oversigt over de brugere, som den individuelle bruger følger. Brugeren kan vælge at søge efter en ny bruger eller vælge en bruger fra vennelisten. Ønsker brugeren at søge efter en ny, skal brugeren angive medlemsID på denne. Systemet sender det indtastede medlemsID til databasen, som validerer om brugeren findes i databasen. Findes brugeren ikke i databasen, sendes en fejlmeddeelse, og brugeren har igen mulighed for at angive medlemsID på en anden bruger. Er brugeren i databasen, hentes brugerens oplysninger, herunder medlemsID og navn, samt brugerens belønninger. Vælger brugeren en ven fra vennelisten, hentes ligeledes brugerens oplysninger samt belønninger. Når brugerens data er hentet fra databasen, vises grænsefladen for brugers belønninger. Det er hertil muligt at følge brugeren, hvis vedkommende ikke allerede følges samt fjerne brugeren, hvis det ønskes ikke at følge brugeren mere. Ønskes det at følge eller fjerne en bruger, sender systemet en anmodning, hvorefter en vennerelation oprettes eller slettes i databasen og en bekræftelse sendes.

### Redigering af adgangskode

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

Aktivitetsdiagram: Redigering af adgangskode



Figur 5.8: Aktivitetsdiagram for redigering af adgangskode.

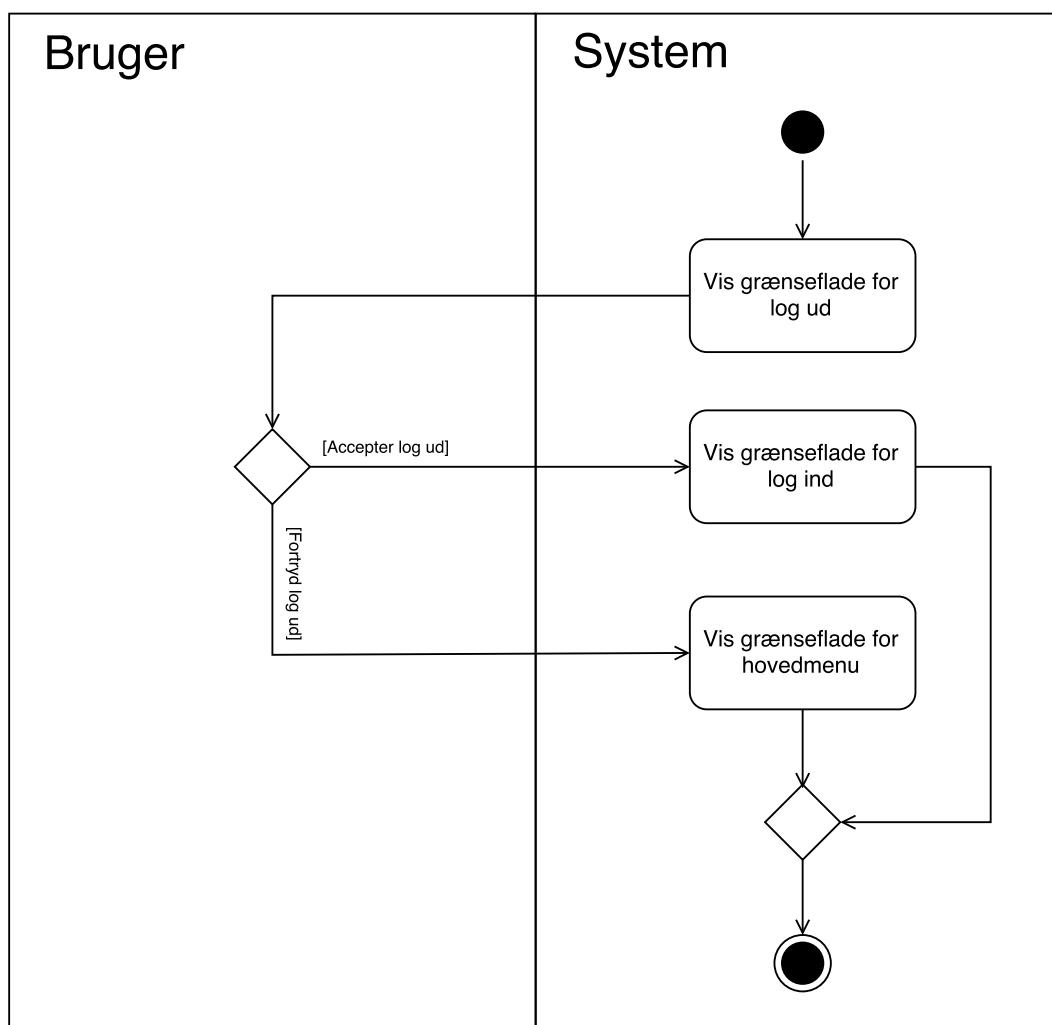
Det skal være muligt for brugeren at ændre adgangskode, da brugeren ved oprettelse får tildelt

en randomiseret adgangskode. Dertil kan adgangskoden blive personlig for brugeren, hvilket vil gøre det nemmere for brugeren at huske. For at den nye adgangskode kan benyttes, skal den minimum være 10 karakterer lang. Rådet for Digital Sikkerhed anbefalder, at adgangskoder bør være minimum 10 karakterer lang, dog er dette ikke et krav [41]. Hvis ønsket om minimum 10 karakterer ikke opfyldes, sendes en fejlmeldelse tilbage til brugeren, hvortil en ny adgangskode kan indtastes. Lykkedes ændring af adgangskoden sendes denne til databasen, hvorefter den gemmes.

### Log ud

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

Aktivitetsdiagram: Log ud



Figur 5.9: Aktivitetsdiagram over log ud.

Vælger brugeren at logge ud via hovedmenuen, vises grænsefladen for log ud. Brugeren skal efterfølgende bekræfte, at de ønsker at logge ud. Ønskes dette, logges brugeren ud og grænsefladen for log ind vises. Fortryder brugeren at logge ud, vil systemet returnere til hovedmenuen og grænsefladen for denne vises.

## 5.4 Analyseklasser

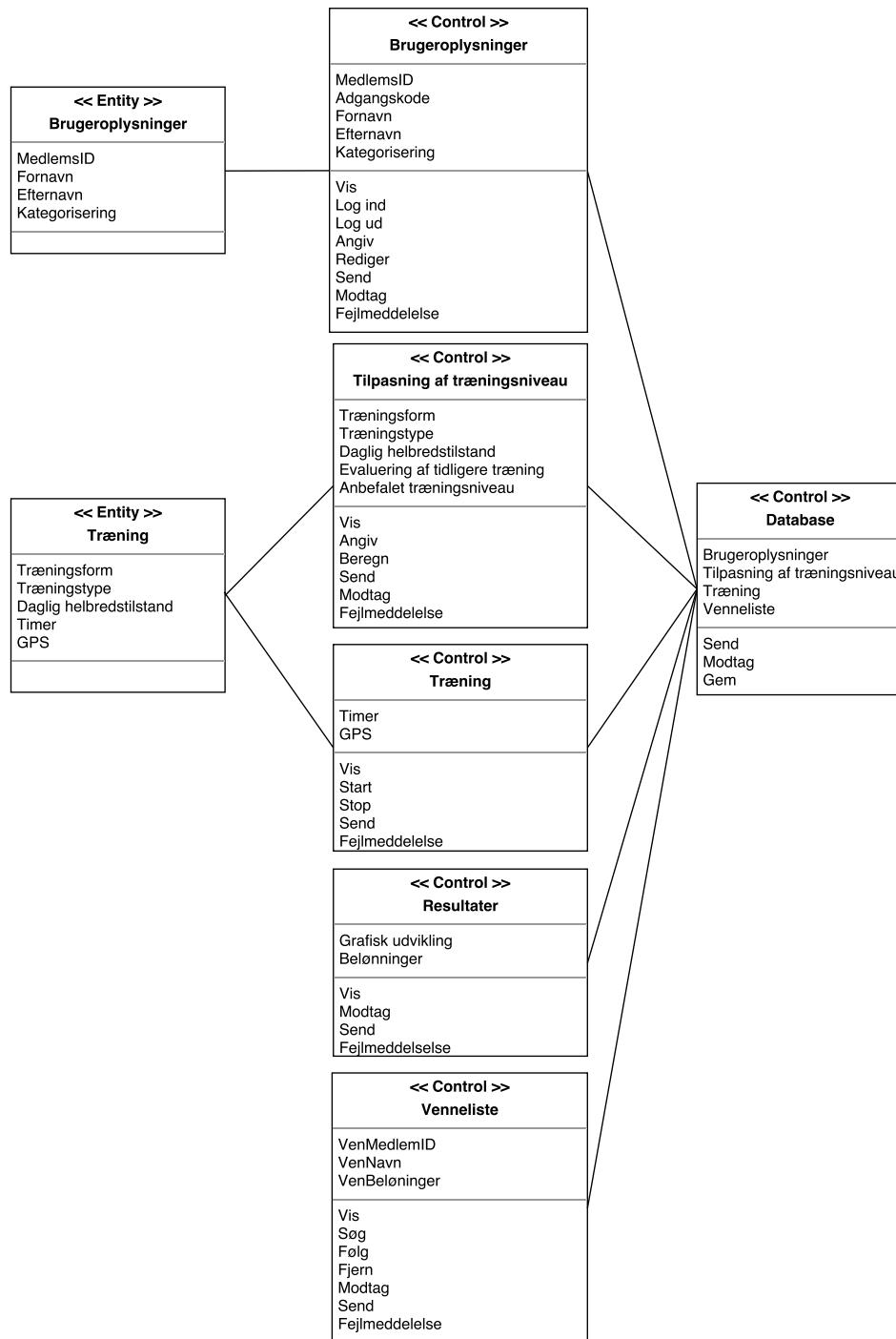
For at analyseklasserne kan udarbejdes, foretages en analyse ud fra systembeskrivelsen, use case og funktionaliteter til at identificere substantiver og verber. Dette gøres for at sikre, at alle funktionaliteter indgår i design af klassediagrammer. Substantiver og verber fra analysen fremgår af tabel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

Substantiver	Verber
<b>Brugeroplysninger</b>	
MedlemsID	Log ind
Adgangskode	Log ud
Fornavn	Valider
Efternavn	Angiv
Kategorisering	Beregn
Fejlmeddeelse	Vis
<b>Tilpasning af træningsniveau</b>	
Træningsform	Start
Træningstype	Stop
Daglig helbredstilstand	Søg
Evaluering af tidligere træning	Følg
Anbefalet træningsniveau	Fjern
Fejlmeddeelse	Send
<b>Træning</b>	Modtag
Timer	Gem
GPS	
Fejlmeddeelse	
<b>Resultater</b>	
Grafisk udvikling	
Belønninger	
Fejlmeddeelse	
<b>Venneliste</b>	
VenMedlemsID	
VenNavn	
VenBelønninger	
Fejlmeddeelse	
<b>Database</b>	

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

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

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



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

Af figur 5.10 fremgår relationen mellem klasserne og deres tilhørende attributter samt metoder. Controlklasserne indeholder metoderne, der skal udføres når denne tilgås, hvorimod entityklasserne har til formål at lagre data. *Brugeroplysninger* og *Traening* er defineret som entityklasser med tilhørende controlklasser. Entityklassen *Traening* har yderligere en relation til controlklassen *Tilpasning af træningsniveau*, da informationer fra denne ligeledes gemmes i denne entity. Derudover er opstillet tre controlklasser, herunder *Resultater*, *Venneliste* og *Database*. Klassen, *Database*, tilgås fra samtlige controlklasser, da formålet med denne klasse er at kommunikere med databasen.

# Kapitel 6

## Systemdesign

---

I dette kapitel beskrives design af app'en samt design af database. I design af app'en tages der udgangspunkt i de analyseklasser, der blev identificeret i afsnit 5.4, hvoraf disse omdannes til designklasser. Herefter visualiseres sammenspillet mellem de forskellige designklasser i sekvensdiagrammer. Design af database, der vil fremgå til sidst af kapitlet, vil visualiseres i et Entity-Relationship (ER)-diagram samt tilhørende schema.

### 6.1 Designafgrænsning

Systemet designes med henblik på at opfylde kravsspecifikationerne, der blev udarbejdet i systemanalysen jf. afsnit 5.2. I forbindelse med de opstillede funktionelle krav foretages der afgrænsninger for at gøre design og efterfølgende implementering af systemet mere konkret.

I relation til træningsaspektet i app'en ønskes det at håndtere forskellige træningsformer samt -typer. Dog afgrænses der til at fokusere på konditionstræning, da det ikke er det primære formål at udforme et træningssæt til KOL-patienter, men at udvikle en prototype, hvortil et reelt træningssæt senere kan inkorporeres. Dog forventes det, at implementeringen af styrketræning og vejrtrækningssøvelser 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.

Gestaltlovene bygger på en række principper, 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 gestaltlove. De mest anvendte til design af grænseflader er; symmetri, regelmæssighed, lukkethed, prægnans, fokuspunkt, ensartethed, nærhed, lighed og harmoni.[42] Udover gestaltlovene har følgende egenskaber betydning for brugervenligheden [43]:

- Systemets funktioner skal være lette at forstå og anvende for uerfarne brugere [43].
- Systemet skal være effektivt at anvende i forhold til tiden, det tager at fuldføre en aktivitet [43].
- Det skal være let at huske, hvordan systemet anvendes efter længere perioder uden brug af systemet [43].
- Fejrlaten ved brug af systemet skal være så lav som muligt [43].
- Systemet skal være tilfredsstillende for brugeren at anvende [43].

### 6.2 Objektorienteret design

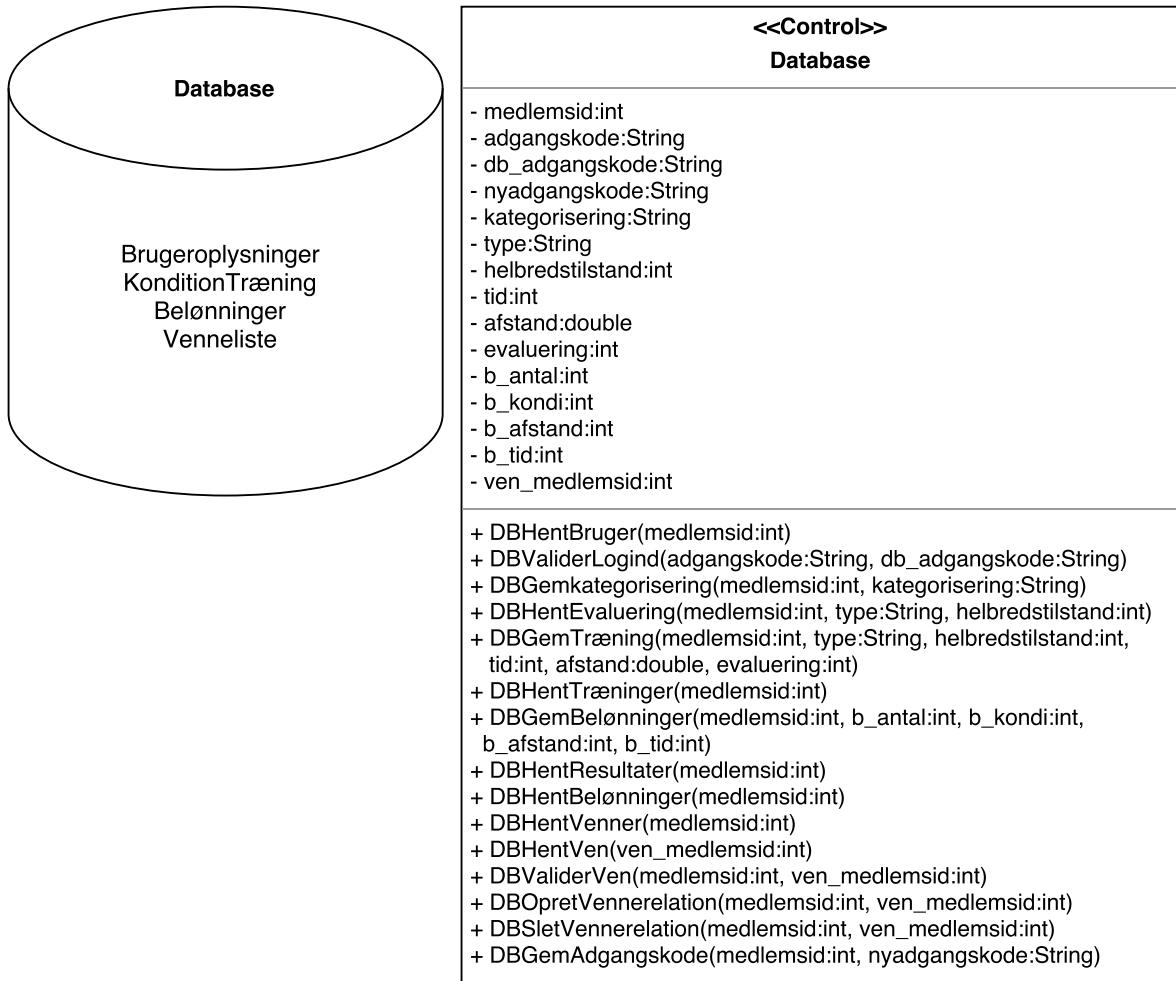
Der vælges at opdele analyseklasserne, som tidligere er defineret, til mindre designklasser. Dette gøres for at specificere de definerede attributter og metoder, hvilket gør det lettere

at modifcere systemet. Ud fra de opstillede designklasser udarbejdes sekvensdiagrammer. I sekvensdiagrammerne er systemets klasser opdelt efter MVC. Objekter og livslinjer farvekodes afhængig af klassens MVC-type. Modelklasser fremstår som røde, viewklasser som blå, og controllerklasser som grønne. I sekvensdiagrammer findes desuden også database samt tilhørende controller, som er illustreret med lilla. I sekvensdiagrammerne findes metoder, der henter data fra en klasse til en anden. Idet det ikke anses som en reel metode, der sender data, illustreres overførslen med en stiplet pil, når data sendes til en klasse efter en hent-metode er kaldt. Metoderne, der fremgår af designklasserne, navngives efter deres funktion.

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

## Database

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



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

Databasen indeholder de oprettede tabeller Brugeroplysninger, KonditionTræning, Belønnin-

ger og Venneliste. Designet af databasen og tilhørende tabeller er yderligere beskrevet i et ER-diagram, der fremgår af afsnit 6.3.

*Database*-controlleren indeholder attributter og metoder. De tilhørende attributter er private og metoder er public. Attributterne er private, da det ikke ønskes at ændre eller tilgå disse direkte fra andre controller. Dertil skaber det en større kontrol samt mindske fejl i forhold til uhensigtsmæssige ændringer. Samtlige attributter er private i de resterende klasser. Attributtens type vil fremgå efter navngivningen af den tilhørende attribut. Metoderne er public, da disse skal kunne tilgås fra andre klasser. De tilhørende metoder for denne controller er henholdsvis Hent, Valider, Gem, Opret og Slet. Disse metoder udføres i forhold til definerede inputsparametre, hvilket er angivet i parenteser. Metoderne har til formål at kommunikere mellem databasen og de forskellige controller. Der oprettes forbindelse til databasen ved samtlige controller.

### Lagring af data

Når der oprettes forbindelse til databasen i forbindelse med, at brugeren logger ind, tilgår resultater eller redigering på app'en, sendes og gemmes data i forskellige entitys, som fremgår af figur 6.2. Der er valgt at udarbejde entitys for at lagre data midlertidig.

<b>&lt;&lt;Entity&gt;&gt;</b> <b>Brugeroplysninger</b>	<b>&lt;&lt;Entity&gt;&gt;</b> <b>KonditionResultater</b>
<ul style="list-style-type: none"> <li>- MedlemID</li> <li>- Fornavn</li> <li>- Efternavn</li> <li>- Kategorisering</li> </ul>	<ul style="list-style-type: none"> <li>- Dato/tid</li> <li>- TræningsType</li> <li>- TræningsForm</li> <li>- DagligHelbredstilstand</li> <li>- Evaluering</li> <li>- Tid</li> <li>- Afstand</li> </ul>
<ul style="list-style-type: none"> <li>+ Get()</li> <li>+ Set()</li> </ul>	<ul style="list-style-type: none"> <li>+ Get()</li> <li>+ Set()</li> </ul>

**Figur 6.2:** Designklasser for entitys, herunder Brugeroplysninger og KonditionResultater.

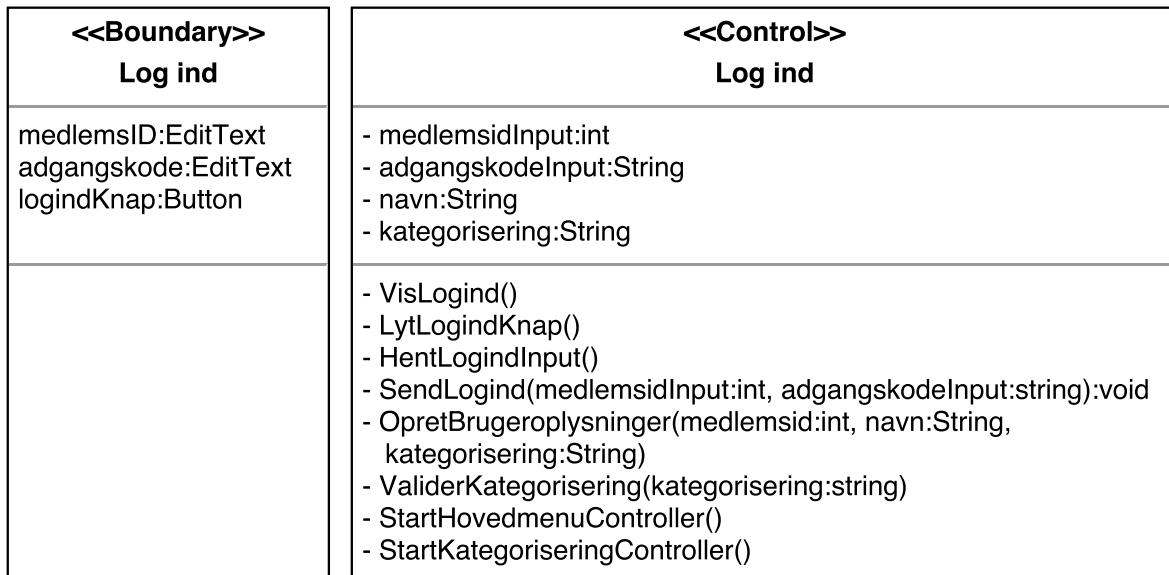
På figur 6.2 fremgår det, at de forskellige attributter er private. Dette er gjort, da det ønskes, at attributterne kun kan tilgås indenfor den samme klasse. Dertil er der opsat public get- og setmetoder, der har til formål at hente og ændre attributternes værdier. *Brugeroplysninger* indeholder informationer om brugeren, herunder MedlemsID, Fornavn, Efternavn og Kategorisering. *KonditionResultater* indeholder brugers resultater, som er defineret ud fra DatoTid, TræningsType, TræningsForm, DagligHelbredstilstand, Evaluering, Tid og Afstand.

## Meddelelser

Der er opstillet en metode for meddelelser, der anvendes i samtlige controllere. Disse meddelelser indebærer både en fejlmeldelse samt bekræftelse, der vises i den tilhørende grænseflade, hvor enten en fejl eller bekræftelse skal forekomme. Disse anvendes til at oplyse brugeren om, hvis for eksempel indtastede værdier ikke stemmer overens med oplyste værdier i databasen, eller parametre er succesfuldt gemt i databasen.

## Log ind

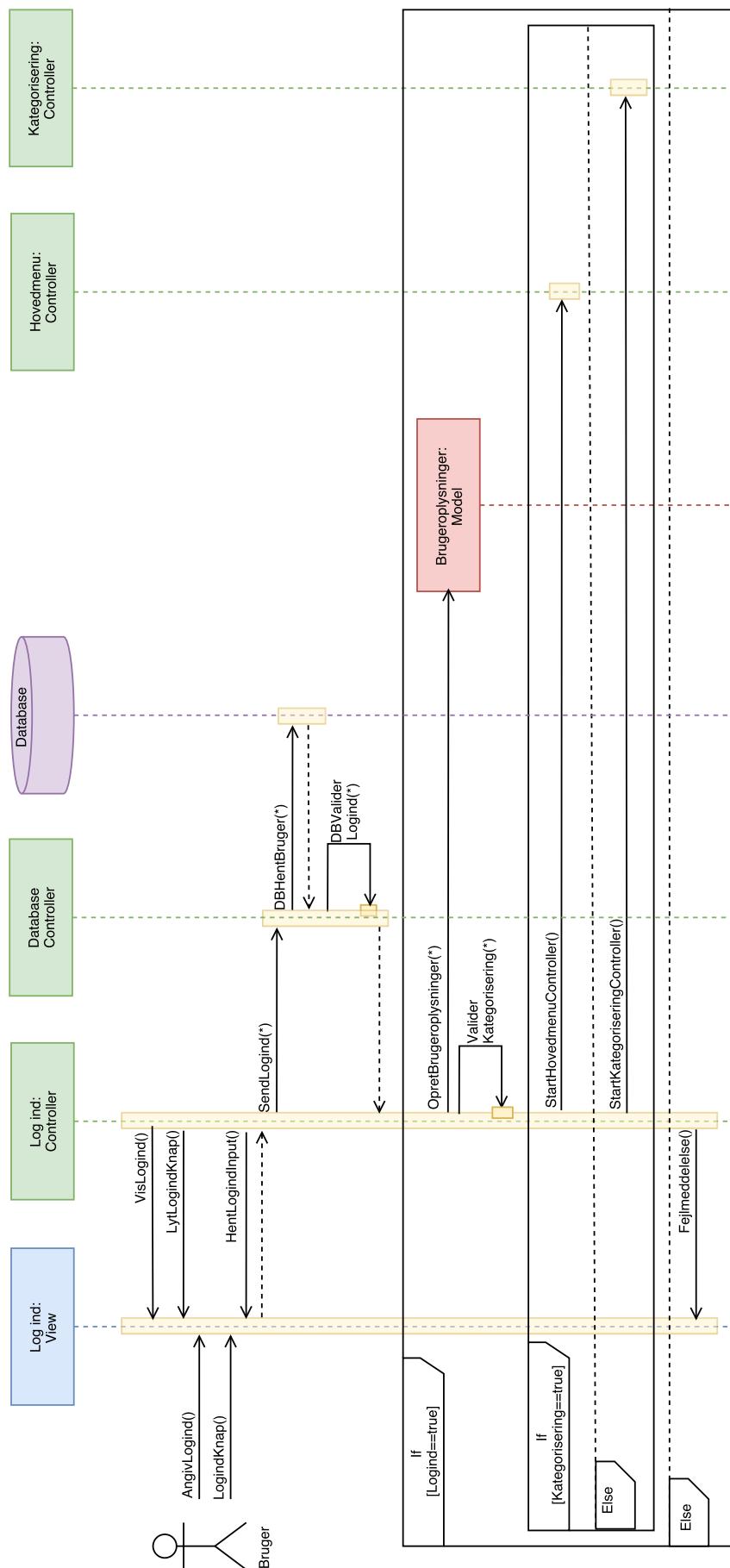
Log ind-funktionen inddeltes i en klasse af typen boundary samt en tilhørende controller. Disse fremgår af figur 6.3.



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

I grænsefladen for *Logind* opstilles tekstmængder 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. Der er opstillet attributter og metoder, der angives som private. Dette gør sig gældende for de resterende klasser. Metoderne er Vis, Lyt, Send, Opret, Valider og Start. Controlleren sender log ind i forhold til inputsparametrene, medlemsidInput og adgangskodeInput, hvorefter brugerens oplysninger hentes via *Database*-controlleren. Denne metode er angivet som en void, hvorfor en værdi ikke returneres. Dertil oprettes en entity, brugeroplysninger, hvori oplysninger senere kan lagres. Ligeledes håndterer controlleren, hvilken grænseflade systemet efterfølgende skal henvises til. Der er i sammenspil med disse designklasser udarbejdet et sekvensdiagram, der fremgår af figur 6.4.



Figur 6.4: Sekvensdiagram for log ind.

Det fremgår af sekvensdiagrammet, at *Log ind*-controlleren starter med at vise grænsefladen for *Log ind*. Dertil lytter controlleren på LogindKnap, der ved tryk henter brugerens angivne log ind-informationer. Dertil sendes disse informationer til *Database*-controlleren, der henter brugeren fra databasen og validerer denne. Er der angivet et asterisk symbol (\*) i metoderne, henviser dette sig til, at der i metoden indgår inputsparametre. Disse parametre fremgår af den tilhørende designklasse for controlleren. Dette gør sig ligeledes gældende for de resterende sekvensdiagrammer.

Bekræftes log ind, oprettes en entity af *Brugeroplysninger* med informationerne hentet på brugeren. Herefter validerer *Log ind*-controlleren, hvorvidt en kategorisering er foretaget. Forefindes en kategorisering i de henteede brugeroplysninger, startes *Hovedmenu*-controlleren. Eksisterer kategoriseringen ikke, startes *Kategorisering*-controlleren. Mislykkes log ind, vises en fejlmeldelse, der forsvinder efter kort tid, hvorefter brugeren igen har mulighed for at indtaste log ind-oplysninger.

## Kategorisering

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

<b>&lt;&lt;Boundary&gt;&gt; Introduktion</b>	<b>&lt;&lt;Boundary&gt;&gt; Indlæggelser</b>	<b>&lt;&lt;Control&gt;&gt; Kategorisering</b>
Intro:TextView VidereKnap:Button	Indlæggelser:TextView IndlæggelserKnap:Button VidereKnap:Button	<ul style="list-style-type: none"> <li>- medlemsid:int</li> <li>- CATscore:int</li> <li>- CATscore:ArrayList&lt;&gt;</li> <li>- antallIndlæggelser:int</li> <li>- kategorisering:String</li> </ul> <ul style="list-style-type: none"> <li>- VisIntroduktion()</li> <li>- LytVidereKnap()</li> </ul> <ul style="list-style-type: none"> <li>- VisUdsagn()</li> <li>- Lyt0til5Knap()</li> <li>- BeregnCATscore(CATscore):int</li> </ul> <ul style="list-style-type: none"> <li>- VisIndlæggelser()</li> <li>- LytIndlæggelserKnap()</li> </ul> <ul style="list-style-type: none"> <li>- BeregnKategorisering(CATscore:int, antallIndlæggelser:int):String</li> <li>- VisKategorisering()</li> <li>- SendKategorisering(medlemsid:int, kategorisering:String):void</li> </ul> <ul style="list-style-type: none"> <li>- StartHovedmenuController()</li> </ul>
<b>&lt;&lt;Boundary&gt;&gt; CATScore</b>	<b>&lt;&lt;Boundary&gt;&gt; Kategorisering</b>	
Udsagn:TextView 0til5Knap:Button VidereKnap:Button	ABCD:TextView VidereKnap:Button	

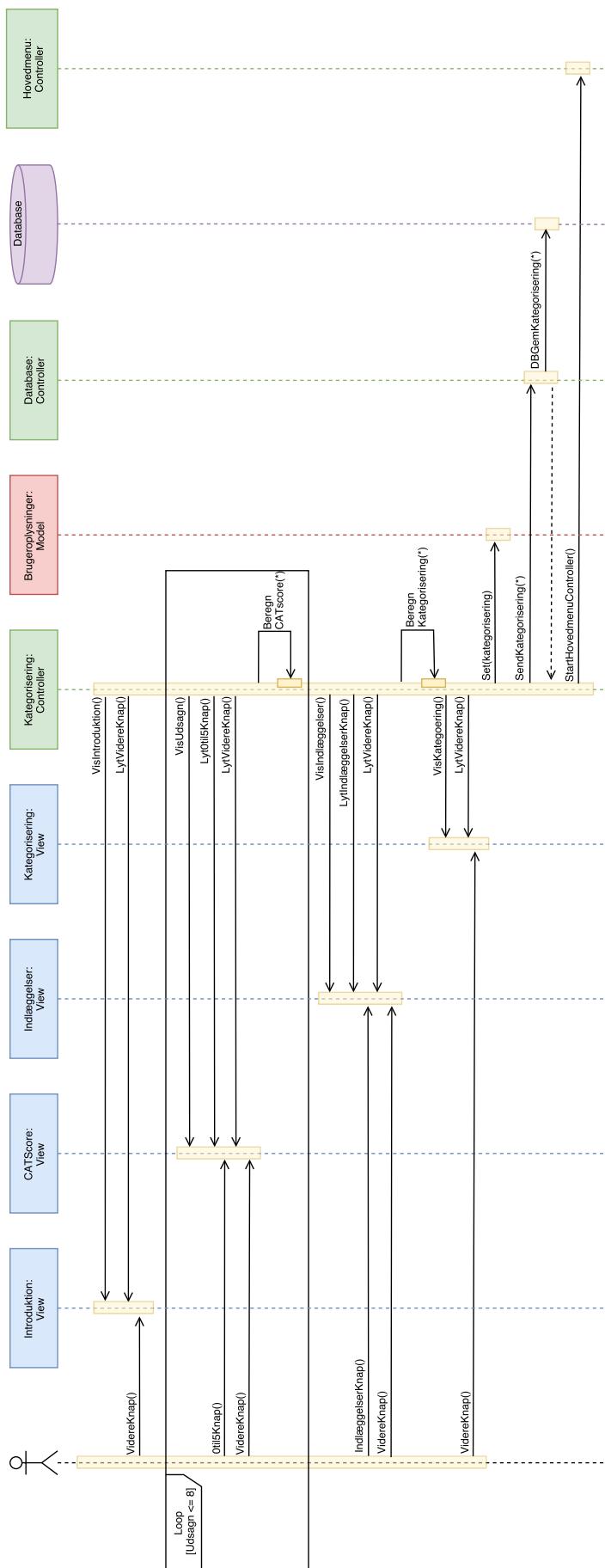
**Figur 6.5:** Designklasser for kategorisering. Til venstre ses de forskellige grænseflader, henholdsvis *Introduktion*, *CATscore*, *Indlæggelser* og *Kategorisering*. Til højre fremgår den dertilhørende controller.

Kategoriseringen inddeltes i fire grænseflader, herunder *Introduktion*, *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 brugervenlig, hvilket kan opnås ved at tydeliggøre adskillelsen mellem *CATscore* og besvarelse af antal årlige indlæggelser, jf. gestaltloven om lighed i afsnit 6.1. Hertil skal brugeren foretage et begrænset antal valg på hver grænseflade, således brugeren ikke eksponeres til for mange valg på samme tid samt informationer på én grænseflade.

De fire grænseflader indeholder tekstmærker af typen TextView, der informerer brugeren om den følgende handling, brugeren skal foretage. Dertil er der ligeledes opsat knapper af typen

Button, således brugeren kan besvare udsagn. I boundaryklassen *CATscore* ses 0til5Knap, som repræsenterer seks forskellige knapper, der skal implementeres i grænsefladen. Disse seks knapper vil gøre det muligt for brugeren at vælge en værdi mellem 0 og 5 for hvert udsagn. Ligeledes ses der i boundaryklassen, *Indlæggelser*, en 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 ingen eller flere indlæggelser.

Den dertilhørende *Kategorisering*-controller håndterer visningen af de fire boundarys samt respektive knapper og tekst. En af attributterne for denne controller fremgår som et ArrayList, der er et array, som anvendes til beregning af CATscore. Controlleren indeholder ligeledes metoderne Vis, Lyt, Beregn, Send og Start, som handler i forhold til de definerede inputsparametre. BeregnKategorisering returnerer dertil en kategorisering af typen string, således denne senere kan sendes og gemmes i databasen. Der er ligeledes udarbejdet et sekvensdiagram for kategorisering, hvilket ses af figur 6.6.



Figur 6.6: Sekvensdiagram for kategorisering.

Det fremgår af sekvensdiagrammet, at grænsefladen for *Introduktion* vises som det første. Denne grænseflade skal informere brugeren om kategoriseringen, der skal foretages for således, at træningsniveau kan tilpasses. Idet brugeren trykker på VidereKnap vises de otte udsagn, der er opstillet i et loop. Udsagnene besvares ved hjælp af knapper fra 0 til 5. Idet brugeren trykker videre fra hvert udsagn, adderes CATscoren. Efter de otte udsagn er besvaret, og den samlede CATscore er beregnet, vises grænsefladen for *Indlæggelser*.

Controlleren lytter dertil til IndlæggelserKnap, hvori brugeren besvarer antal årlige indlæggelser forårsaget af KOL. Besvarelsen bekræftes ved at benytte VidereKnap. Kategoriseringen kan derved beregnes og vises i *Kategorisering*-grænsefladen. Denne kategorisering gemmes i entityen *Brugeroplysninger* ved kalde set-metoden fra entityen og sendes til *database*-controlleren, hvorved denne gemmer kategoriseringen i *database*. Herefter startes *hovedmenu*-controlleren.

## Hovedmenu

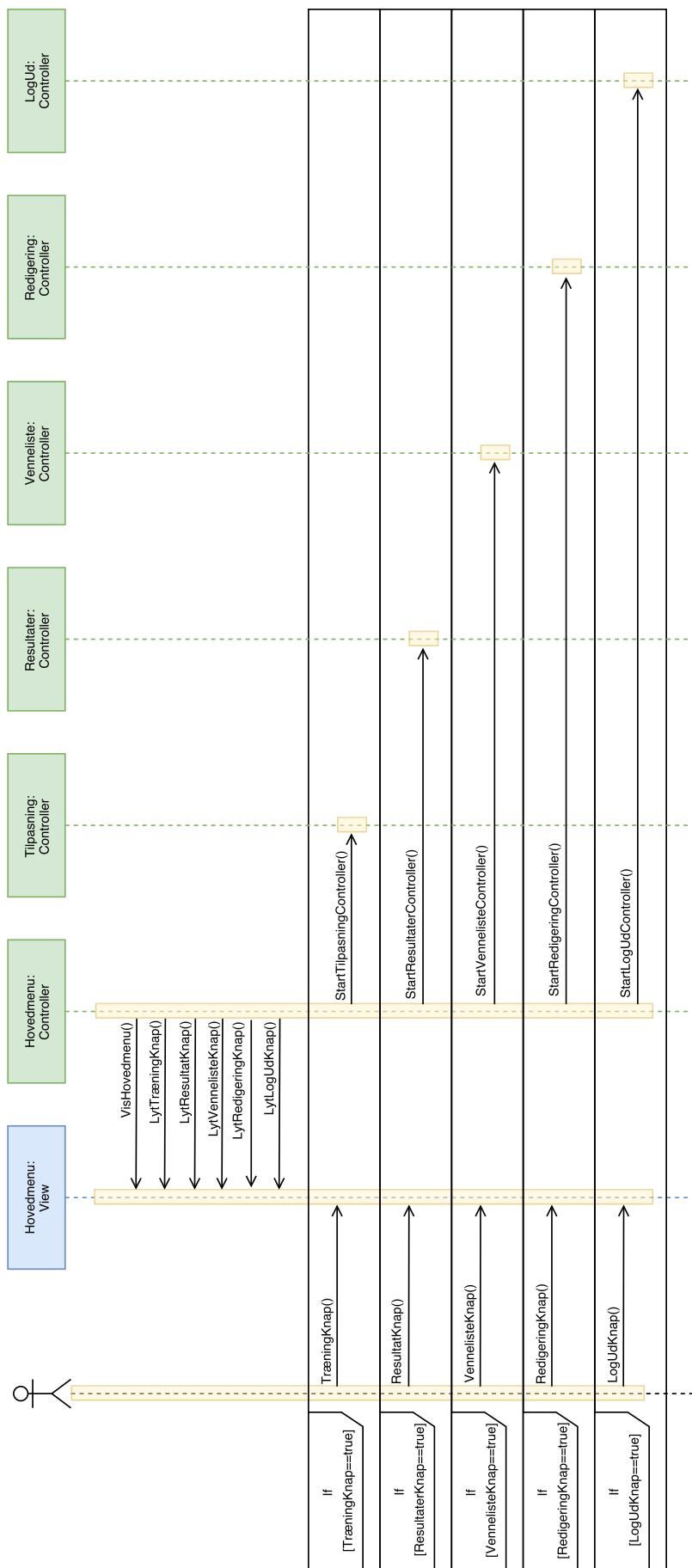
Hovedmenuen er den primære grænseflade, hvor andre grænseflader kan tilgås. Hovedmenuen indeles i en boundary samt en dertilhørende controller. Disse fremgår af figur 6.7.

<b>&lt;&lt;Boundary&gt;&gt;</b>	<b>&lt;&lt;Control&gt;&gt;</b>
<b>Hovedmenu</b>	<b>Hovedmenu</b>
TræningKnap:Button ResultaterKnap:Button VennelisteKnap:Button RedigeringKnap:Button LogudKnap:Button	<ul style="list-style-type: none"> <li>- VisHovedmenu()</li> <li>- LytTræningKnap()</li> <li>- LytResultaterKnap()</li> <li>- LytVennelisteKnap()</li> <li>- LytRedigeringKnap()</li> <li>- LytLogudKnap()</li> </ul> <ul style="list-style-type: none"> <li>- StartTilpasningController()</li> <li>- StartResultaterController()</li> <li>- StartVennelisteController()</li> <li>- StartRedigeringController()</li> <li>- StartLogudController()</li> </ul>

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

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

Controlleren indeholder metoderne Vis, Lyt og Start. Et sekvensdiagram hertil er udarbejdet, hvilket fremgår af figur 6.8.

*Figur 6.8: Sekvensdiagram for hovedmenu.*

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

## Tilpasning af træningsniveau

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

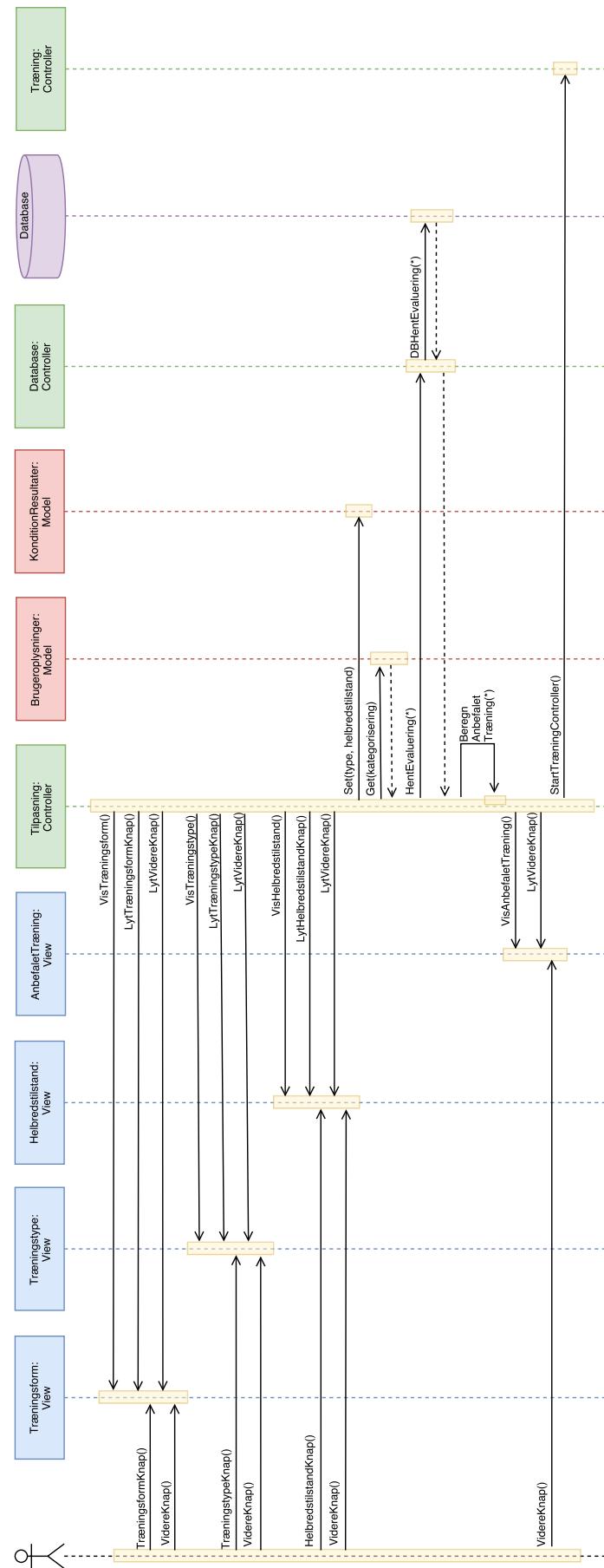
<b>&lt;&lt;Boundary&gt;&gt; Træningsform</b>	<b>&lt;&lt;Boundary&gt;&gt; Helbredstilstand</b>	<b>&lt;&lt;Control&gt;&gt; Tilpasning</b>
Træningsform: TextView TræningsformKnap: Button VidereKnap: Button	Helbredstilstand: TextView HelbredstilstandKnap: Button VidereKnap: Button	<ul style="list-style-type: none"> <li>- medlemsid:int</li> <li>- type:String</li> <li>- helbredstilstand:int</li> <li>- kategorisering:String</li> <li>- evaluering:int</li> </ul> <ul style="list-style-type: none"> <li>- VisTræningsform()</li> <li>- LytTræningsformKnap()</li> <li>- LytVidereKnap()</li> </ul>
<b>&lt;&lt;Boundary&gt;&gt; Træningstype</b>	<b>&lt;&lt;Boundary&gt;&gt; AnbefaletTræning</b>	
Træningstype: TextView TræningstypeKnap: Button VidereKnap: Button	AnbefaletTræningt: TextView VidereKnap: Button	<ul style="list-style-type: none"> <li>- VisTræningstype()</li> <li>- LytTræningstypeKnap()</li> </ul> <ul style="list-style-type: none"> <li>- VisHelbredstilstand()</li> <li>- LytHelbredstilstandKnap()</li> </ul> <ul style="list-style-type: none"> <li>- HentEvaluering(medlemsid:int, type:String, helbredstilstand:int): void</li> </ul> <ul style="list-style-type: none"> <li>- BeregnAnbefaletTræning(kategorisering:String, helbredstilstand:int, evaluering:int)</li> <li>- VisAnbefaletTræning()</li> </ul> <ul style="list-style-type: none"> <li>- StartTræningController()</li> </ul>

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

Der er opstillet grænseflader for tilpasning af træning, hvilket omfatter *Træningsform*, *Træningstype*, *Helbredstilstand* og *AnbefaletTræning*. Der er til hver grænseflade opstillet tekstmelte af typen TextView og knapper af typen Button.

Den tilhørende controller til de ovenstående boundarys lagrer den angivne træningsform, træningstype samt helbredstilstand i entityen, KonditionResultater, således disse senere kan sendes til databasen efter endt træning. Der er opstillet tilhørende attributter og metoder, herunder Vis, Lyt, Hent, Beregn og Start. Metoderne henter og beregner i forhold til definerede inputsparametre. Efter inputsparametre er der angivet, hvorvidt der returneres en værdi.

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



*Figur 6.10: Sekvensdiagram for tilpasning af træningsniveau.*

Den første grænseflade, som vises, er *Træningsform*. Denne grænseflade indeholder et tekstfelt, der beskriver, hvad brugeren skal angive. Dertil er der opstillet en TræningsformKnap, hvor brugeren kan vælge mellem konditionstræning, styrketræning eller vejrrækningsøvelser. Når der er angivet træningsform, trykker brugeren på VidereKnap, hvorefter *Tilpasning*-controlleren 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 kaldes set-metoden fra modellen *KonditionResultater*, hvori type og helbredstilstand gemmes. Controlleren henter kategoriseringen ved brug af get-metoden fra modellen *Brugeroplysninger* og henter efterfølgende tidligere evaluering fra *Database* via *Database*-controlleren, hvis der eksisterer en tilhørende evaluering. Efterfølgende beregner *Tilpasning*-controlleren anbefalet træningsniveau og viser dette på grænsefladen for *AnbefaletTræning*. Brugeren kan herefter trykke på VidereKnap, hvis denne trykkes på, startes *Træning*-controlleren.

## Træning

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

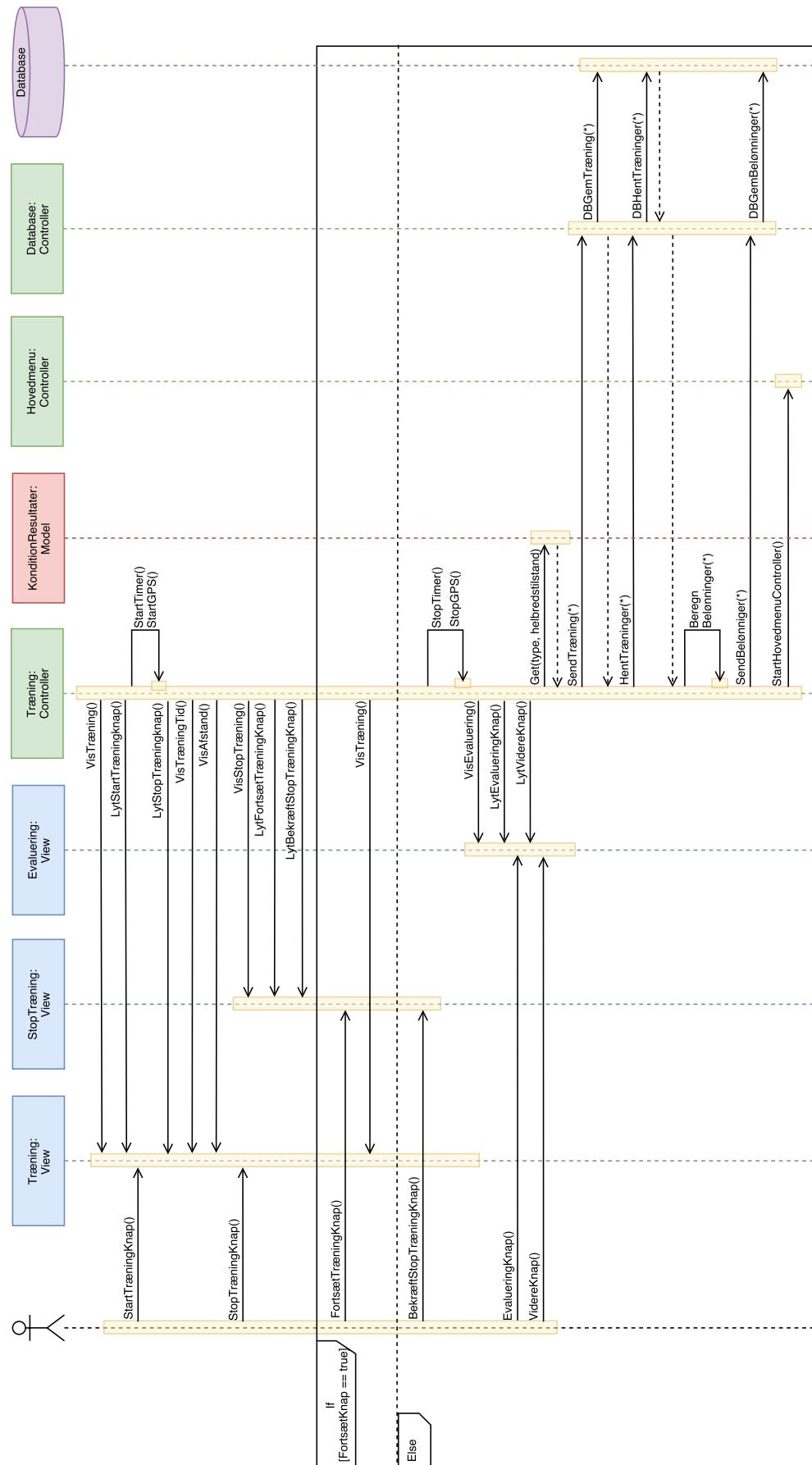
<p><b>&lt;&lt;Boundary&gt;&gt;</b></p> <p><b>Træning</b></p> <hr/> <p>Tid:TextView Afstand:TextView StartTræningKnap:Button StopTræningKnap:Button</p>	<p><b>&lt;&lt;Control&gt;&gt;</b></p> <p><b>Træning</b></p> <hr/> <ul style="list-style-type: none"> <li>- medlemsid:int</li> <li>- tid:int</li> <li>- afstand:double</li> <li>- type:String</li> <li>- helbredstilstand:int</li> <li>- evaluering:int</li> <li>- b_antal:int</li> <li>- b_kondi:int</li> <li>- b_afstand:int</li> <li>- b_tid:int</li> </ul>
<p><b>&lt;&lt;Boundary&gt;&gt;</b></p> <p><b>StopTræning</b></p> <hr/> <p>StopTræning:TextView FortsætTræningKnap:Button BekræftStopTræningKnap:Button</p>	<ul style="list-style-type: none"> <li>- VisTræning()</li> <li>- LytStartTræningKnap()</li> <li>- LytStopTræningKnap()</li> <li>- StartTimer()</li> <li>- VisTræningTid()</li> <li>- StopTimer()</li> <li>- StartGPS()</li> <li>- VisAfstand()</li> <li>- StopGPS()</li> </ul> <ul style="list-style-type: none"> <li>- VisStopTræning()</li> <li>- LytFortsætTræningKnap()</li> <li>- LytBekræftStopTræningKnap()</li> </ul>
<p><b>&lt;&lt;Boundary&gt;&gt;</b></p> <p><b>Evaluering</b></p> <hr/> <p>EvalueringTextfelt:TextView EvalueringKnap:Button VidereKnap:Button</p>	<ul style="list-style-type: none"> <li>- VisEvaluering()</li> <li>- LytEvalueringKnap()</li> <li>- LytVidereKnap()</li> </ul> <ul style="list-style-type: none"> <li>- SendTræning(medlemsid:int, type:String, helbredstilstand:int, tid:int, afstand:double, evaluering:int):void</li> <li>- HentTræninger(medlemsid:int):void</li> <li>- BeregnBelønninger(medlemsid:int, b_antal:int, b_kondi:int, b_afstand:int, b_tid:int):void</li> <li>- SendBelønninger(medlemsid:int, b_antal:int, b_kondi:int, b_afstand:int, b_tid:int):void</li> </ul> <ul style="list-style-type: none"> <li>- StartHovedmenuController()</li> </ul>

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

Grænsefladen for *Træning*, *StopTræning* og *Evaluering* indeholder tekstfelter og knapper af typen TextView og Button.

*Træning*-controlleren indeholder attributter samt metoder, hvoraf metoderne omfatter Vis, Lyt, Start, Stop, Send, Hent og Beregn. Metoderne Send, Hent, og Beregn handler i forhold til definerede inputsparametre.

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

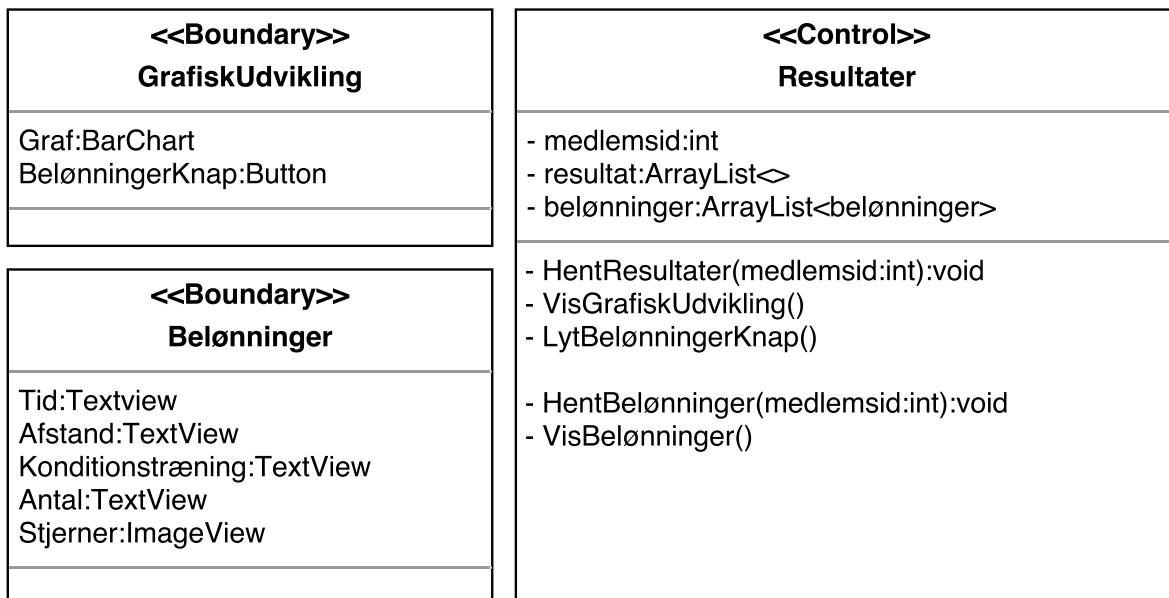


**Figur 6.12:** Sekvensdiagram for træning.

*Træning* er den første grænseflade, der vises efter tilpasningen af træningsniveauet. *Træning*-controlleren starter timer og GPS, når brugeren har trykket på StartTræningKnap. Herefter vises tiden og afstanden i grænsefladen. Brugeren kan under træningen, eller når træningen er fuldført, afslutte ved at trykke på StopTræningKnap, hvorefter grænsefladen for *Stop Træning* vises. I denne grænseflade har brugeren mulighed for at trykke på FortsætTræningKnap eller BekræftStopTræningKnap. Vælges FortsætTræningKnap vises grænsefladen for *Træning*, og brugeren skal igen trykke på StartTræningKnap for at starte træning og måleenheder. Vælges BekræftStopTræningKnap, stopper controlleren med at måle data fra GPS og timer. Herefter vises *Evaluering*-grænsefladen, hvor brugeren skal angive en evaluering af dagens træning. Evaluering bekræftes ved VidereKnap. Herefter henter controlleren typen samt helbredstilstanden ved brug af get-metode fra modellen, *KonditionResultater*, hvorefter controlleren sender oplysninger fra den samlede træning til *Database*-controlleren, som gemmer træningen i *Database*. Efterfølgende hentes tidligere træninger fra *Database* via dens controller. Disse træninger benyttes for at beregnes brugerens samlede belønninger i *Træning*-controlleren. Derefter sendes disse til *Database*-controlleren, der gemmer disse belønninger i *Database*, hvorefter der gives besked til *Hovedmenu*-controlleren om at vise hovedmenuen.

## Resultater

Resultater har to boundarys, hvortil der er opstillet en controlklasse. Boundarys og tilhørende controller fremgår af figur 6.13.

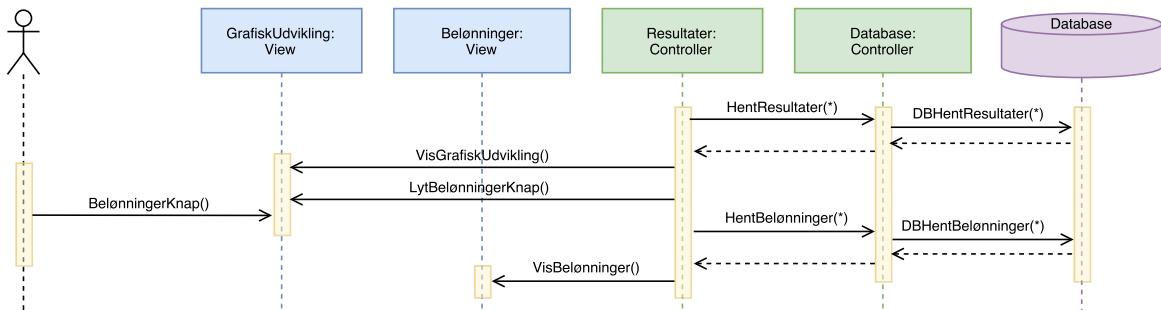


*Figur 6.13: Designklasser for resultater. Til venstre ses boundarys for den grafiske udvikling og belønninger. Til højre fremgår den tilhørende controller.*

I grænsefladen for *GrafiskUdvikling* er der opstillet en graf af typen BarChart samt en knap af typen Button, for at give mulighed for at tilgå belønninger. Boundary for *Belønninger* indeholder billeder af typen ImageView og tekstfelter af typen TextView. Billederne viser antallet af stjerner, der repræsenterer belønninger, brugeren har opnået. *Resultater*-controlleren indeholder attributter og metoder. Disse metoder omfatter Hent, Vis, Lyt og Start. Der er opstillet attributter, hvorfaf to af disse er af typen ArrayList, der

benyttes til at lagre resultater samt belønninger. Hent-metoder handler ud fra definerede inputsparametre og er angivet void.

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



**Figur 6.14:** Sekvensdiagram for Resultater.

Når brugeren tilgår resultater, henter *Resultat-controller*, gennem *Database-controller*, brugerens resultater fra *Database*. Resultater vises i et BarChart i grænsefladen for *GrafiskUdvikling*. Dertil er der opstillet en *BeloenningerKnap*, der videresender brugeren til grænsefladen for *Beloenninger* ved tryk. Idet brugeren trykker på knappen, henter *Resultat-controller*, gennem *Database-controller*, brugerens belønninger, der vises i grænsefladen. Denne viser stjerner opnået indenfor tid, afstand, konditonstræning og antal træninger.

## Venneliste

Venneliste inddeltes i to boundaryklasser og en fælles controlklasse, som fremgår af figur 6.15.

<table border="1"> <thead> <tr> <th style="text-align: center;"><b>&lt;&lt;Boundary&gt;&gt;</b></th></tr> <tr> <th style="text-align: center;"><b>Venneliste</b></th></tr> </thead> <tbody> <tr> <td>         VenNavn:ListView          VælgVenKnap:Button          VenMedlemsID:EditText          SøgVenKnap:Button       </td></tr> </tbody> </table> <table border="1"> <thead> <tr> <th style="text-align: center;"><b>&lt;&lt;Boundary&gt;&gt;</b></th></tr> <tr> <th style="text-align: center;"><b>Ven</b></th></tr> </thead> <tbody> <tr> <td>         VenNavn:TextView          VenMedlemsID:TextView          VenBelønning:ImageView          FølgVenKnap:Button          FjernVenKnap:Button       </td></tr> </tbody> </table>	<b>&lt;&lt;Boundary&gt;&gt;</b>	<b>Venneliste</b>	VenNavn:ListView VælgVenKnap:Button VenMedlemsID:EditText SøgVenKnap:Button	<b>&lt;&lt;Boundary&gt;&gt;</b>	<b>Ven</b>	VenNavn:TextView VenMedlemsID:TextView VenBelønning:ImageView FølgVenKnap:Button FjernVenKnap:Button	<table border="1"> <thead> <tr> <th style="text-align: center;"><b>&lt;&lt;Control&gt;&gt;</b></th></tr> <tr> <th style="text-align: center;"><b>Venneliste</b></th></tr> </thead> <tbody> <tr> <td> <ul style="list-style-type: none"> <li>- medlemsid:int</li> <li>- venneliste:ArrayList&lt;&gt;</li> <li>- ven_medlemsidInput:int</li> </ul>   <ul style="list-style-type: none"> <li>- HentVenner(medlemsid:int):void</li> <li>- VisVenneliste()</li> <li>- LytVælgVenKnap()</li> <li>- LytSøgVenKnap()</li> <li>- SendVenMedlemsid(ven_medlemsidInput:int)</li> <li>- HentVen(ven_medlemsidInput:int):void</li> </ul> </td></tr> </tbody> </table> <table border="1"> <thead> <tr> <th style="text-align: center;"><b>&lt;&lt;Control&gt;&gt;</b></th></tr> <tr> <th style="text-align: center;"><b>Ven</b></th></tr> </thead> <tbody> <tr> <td> <ul style="list-style-type: none"> <li>- VisVen()</li> <li>- LytFølgVenKnap()</li> <li>- LytFjernVenKnap()</li> <li>- OpretVennrerelation(medlemsid:int, ven_medlemsidInput:int):void</li> <li>- SletVennrerelation(medlemsid:int, ven_medlemsidInput:int):void</li> </ul> </td></tr> </tbody> </table>	<b>&lt;&lt;Control&gt;&gt;</b>	<b>Venneliste</b>	<ul style="list-style-type: none"> <li>- medlemsid:int</li> <li>- venneliste:ArrayList&lt;&gt;</li> <li>- ven_medlemsidInput:int</li> </ul> <ul style="list-style-type: none"> <li>- HentVenner(medlemsid:int):void</li> <li>- VisVenneliste()</li> <li>- LytVælgVenKnap()</li> <li>- LytSøgVenKnap()</li> <li>- SendVenMedlemsid(ven_medlemsidInput:int)</li> <li>- HentVen(ven_medlemsidInput:int):void</li> </ul>	<b>&lt;&lt;Control&gt;&gt;</b>	<b>Ven</b>	<ul style="list-style-type: none"> <li>- VisVen()</li> <li>- LytFølgVenKnap()</li> <li>- LytFjernVenKnap()</li> <li>- OpretVennrerelation(medlemsid:int, ven_medlemsidInput:int):void</li> <li>- SletVennrerelation(medlemsid:int, ven_medlemsidInput:int):void</li> </ul>
<b>&lt;&lt;Boundary&gt;&gt;</b>													
<b>Venneliste</b>													
VenNavn:ListView VælgVenKnap:Button VenMedlemsID:EditText SøgVenKnap:Button													
<b>&lt;&lt;Boundary&gt;&gt;</b>													
<b>Ven</b>													
VenNavn:TextView VenMedlemsID:TextView VenBelønning:ImageView FølgVenKnap:Button FjernVenKnap:Button													
<b>&lt;&lt;Control&gt;&gt;</b>													
<b>Venneliste</b>													
<ul style="list-style-type: none"> <li>- medlemsid:int</li> <li>- venneliste:ArrayList&lt;&gt;</li> <li>- ven_medlemsidInput:int</li> </ul> <ul style="list-style-type: none"> <li>- HentVenner(medlemsid:int):void</li> <li>- VisVenneliste()</li> <li>- LytVælgVenKnap()</li> <li>- LytSøgVenKnap()</li> <li>- SendVenMedlemsid(ven_medlemsidInput:int)</li> <li>- HentVen(ven_medlemsidInput:int):void</li> </ul>													
<b>&lt;&lt;Control&gt;&gt;</b>													
<b>Ven</b>													
<ul style="list-style-type: none"> <li>- VisVen()</li> <li>- LytFølgVenKnap()</li> <li>- LytFjernVenKnap()</li> <li>- OpretVennrerelation(medlemsid:int, ven_medlemsidInput:int):void</li> <li>- SletVennrerelation(medlemsid:int, ven_medlemsidInput:int):void</li> </ul>													

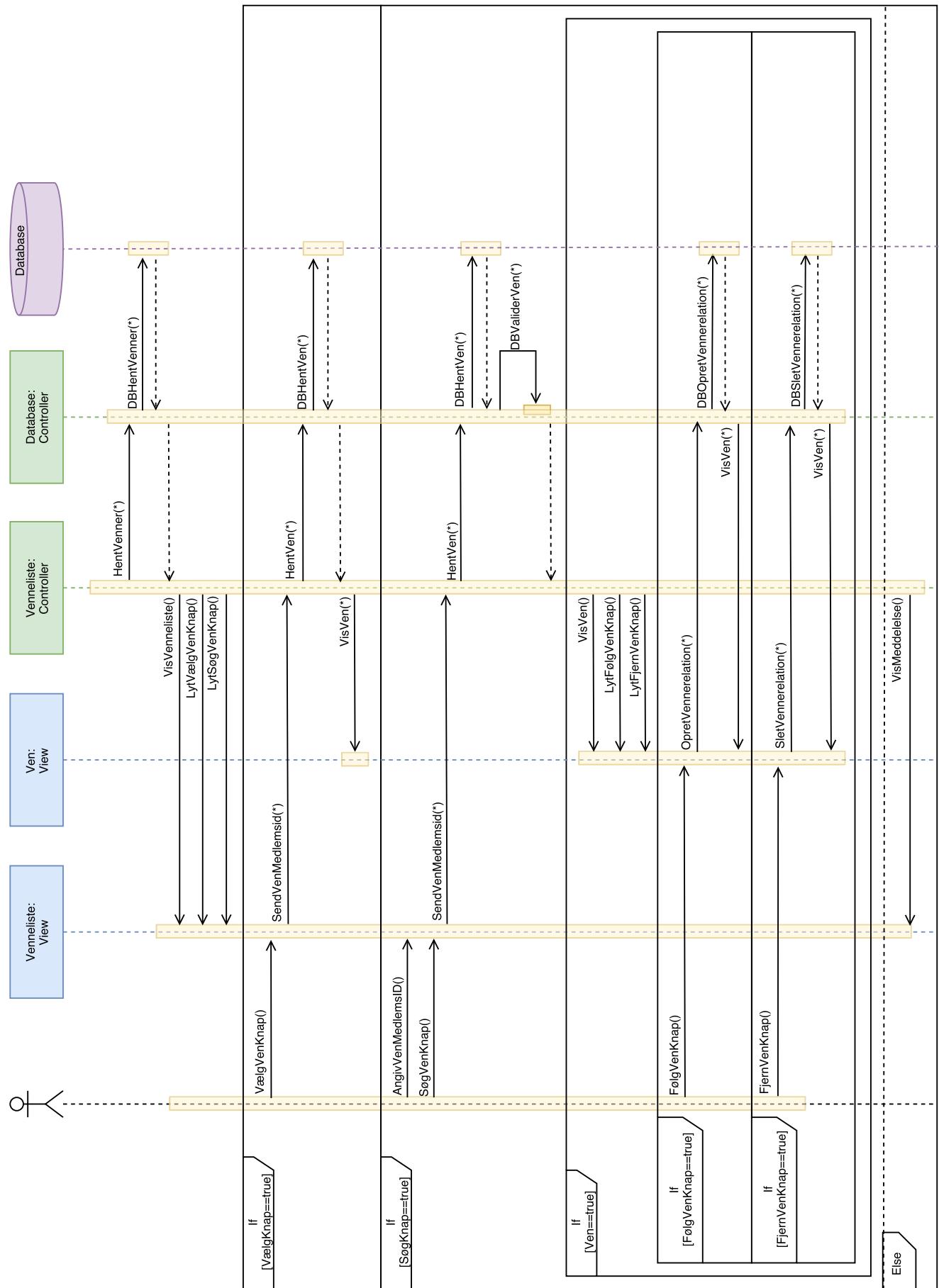
**Figur 6.15:** Designklasser for venneliste. Til venstre ses boundary for Venneliste og Ven, hvor der til højre ses tilhørende controller.

I *Venneliste*-grænsefladen findes en liste, af typen *ListView*, med navn på de brugere, der følges. Hver bruger, der vises af den givne grænseflade, kan tilgås ved at trykke på brugeren. Derudover er der i denne grænseflade et søgerfelt, af typen *EditText*, med en

tilhørende SøgVenKnap, af typen Button, som muliggør, at brugeren kan søge på andre brugere ved at angive deres medlemsID. Ven-grænsefladen indeholder tekstfelter af typen TextView, som viser brugerens navn og medlemsID. Derudover er der anvendt billede af typen ImageView, som viser brugerens belønninger. Hvis brugeren ikke følges, er der her opstillet en FølgVenKnap. Følges brugeren allerede er der opstillet en FjernVenKnap.

Controller for *Venneliste* indeholder attributter, herunder attributter af typen int og ArrayList. Arrayet for venneliste har til formål at lagre brugerens venner. Dertil har controlleren ligeledes metoder, herunder Hent, Vis, Lyt, Søg, Valider, Opret og Slet. Metoderne handler ud fra inputsparametre angivet i parenteserne.

Ud fra de nævnte designklasser er udarbejdet et sekvensdiagram, der fremgår af figur 6.16.



Figur 6.16: Sekvensdiagram for venneliste.

*Venneliste*-controlleren henter venner fra *Database* via *Database*-controlleren, hvorefter grænsefladen for venneliste vises. Dertil lytter controlleren på VælgVenKnap samt SøgVenKnap. Vælger brugeren at tilgå en ven fra vennelisten, sendes vennens medlemsID til *Venneliste*-controlleren, hvorefter vennen og vedkommendes belønninger hentes i databasen. Dertil vises grænsefladen for *Ven*. Vælger brugeren derimod at søge efter en anden bruger, sendes det indtastede medlemsID, således den søgte bruger kan findes i databasen, hvortil den valideres. Såfremt brugeren ikke findes i databasen, vises en fejlmeddeelse i *Venneliste*-grænsefladen. Findes brugeren i databasen, vises grænsefladen for *Ven*, hvor brugeren har mulighed for at følge eller fjerne en bruger. Hvis brugeren trykker på FølgVenKnap oprettes der en vennrelation i databasen, hvor vennrelationen modsat vil slettes ved tryk på FjernVenKnap. Efterfølgende sendes en bekræftelse på, at vennen er tilføjet eller fjernet.

## Redigering

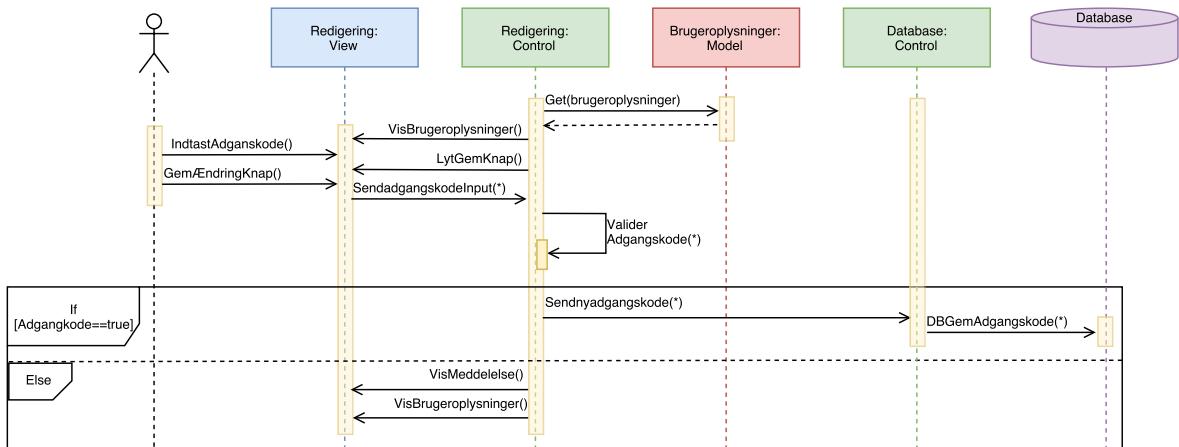
Redigering inddeltes i en boundary og en tilhørende controller, som det fremgår af figur 6.17.

<<Boundary>> <b>Redigering</b>	<<Control>> <b>Redigering</b>
MedlemsID:TextView Navn:TextView Kategorisering:TextView NyAfgangskode:EditText GentagAfgangskode:EditText GemÆndringKnap:Button	<ul style="list-style-type: none"> <li>- medlemsid:int</li> <li>- navn:String</li> <li>- kategorisering:String</li> <li>- nyadgangskodelInput:String</li> <li>- gentagadgangskodelInput:String</li> </ul> <ul style="list-style-type: none"> <li>- VisBrugeroplysninger()</li> <li>- LytGemÆndringKnap()</li> <li>- SendadgangskodelInput(nyadgangskode:String, gentagadgangskode:String):void</li> <li>- ValiderAfgangskode(nyadgangskodelInput:String, gentagadgangskodelInput:String)</li> <li>- Sendnyadgangskode(medlemsid:int, nyadgangskode:String):void</li> </ul>

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

I grænsefladen for *Redigering* opstilles tekstmærker af typen TextView for medlemsID, navn og kategorisering. Derudover er der tekstmærker, af typen EditText, for ny adgangskode og gentag adgangskode, hvor brugeren kan angive en ny adgangskode. Dertil er der en GemÆndringKnap, af typen Button. GemÆndringKnap indikerer ved tryk, at brugeren ønsker at gemme den nye adgangskode. Den tilhørende controller har attributter samt metoder, der omfatter Vis, Lyt, Valider og Send. Valider og Send handler ud fra opstillede inputsparametre.

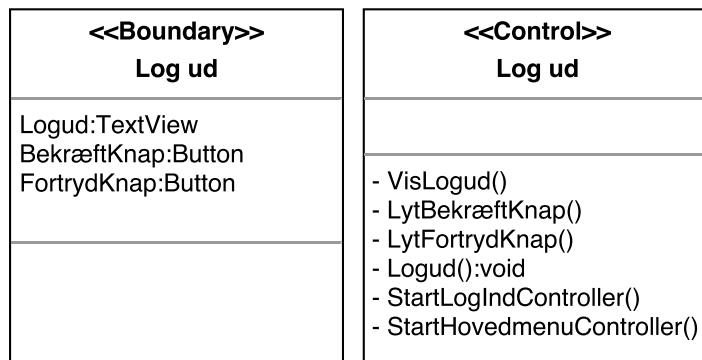
Til disse klasser er der opstillet et sekvensdiagram, hvilket fremgår af figur 6.18.

**Figur 6.18:** Sekvensdiagram for redigering.

Redigering-controlleren henter brugeroplysninger, herunder medlemsID, fornavn, efternavn samt kategorisering ved hjælp af get-metoden fra modellen *Brugeroplysninger*. Disse oplysninger vises i grænsefladen for *Redigering*, hvortil brugeren kan se sin information. Fra denne grænseflade er det ligeledes muligt for brugeren at ændre sin adgangskode. Adgangskoden skal indtastes to gange for at sikre, at adgangskoden er identisk. Dertil skal adgangskoden være minimum 10 karakterer lang. Adgangskoden sendes og valideres i *Redigering*-controlleren, hvorefter den sendes til *Database*-controlleren, der gemmer den i *Database*. Overholder adgangskoden ikke de opstillede kriterier, vises en fejlmeddelelse i grænsefladen. Denne forsvinder efter kort tid, hvortil grænsefladen for redigering vises igen.

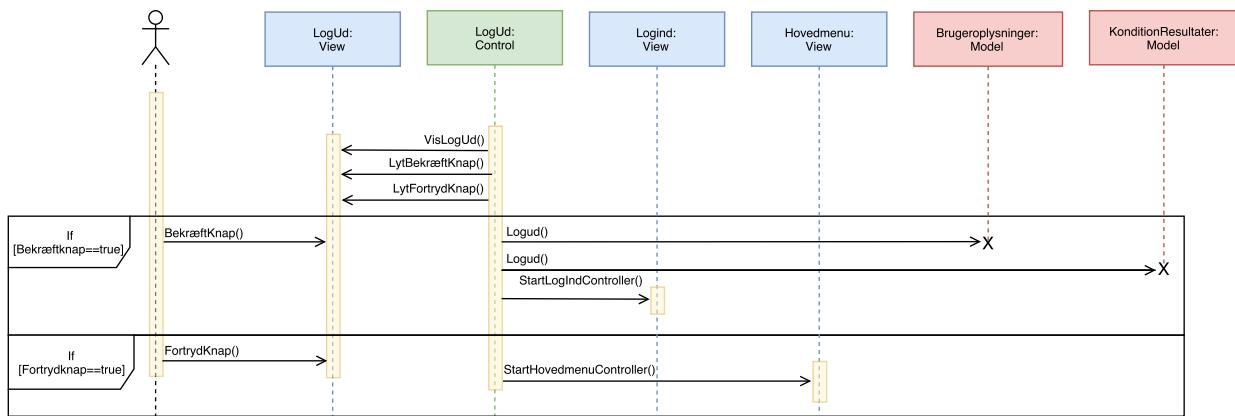
## Log ud

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

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

Der opstilles i log ud et tekstfelt, af typen TextView. Dertil opstilles en BekræftKnap og FortrydKnap af typen Button. Den tilhørende controller indeholder metoder Vis, Lyt, Logud og Start. Logud-metoden har til formål at slette data på app'en, hvorefter denne kan oprettes på ny, idet en bruger logger ind.

I sammenspil med designklasserne er der opstillet et sekvensdiagram, som fremgår af figur 6.20.



Figur 6.20: Sekvensdiagram for log ud.

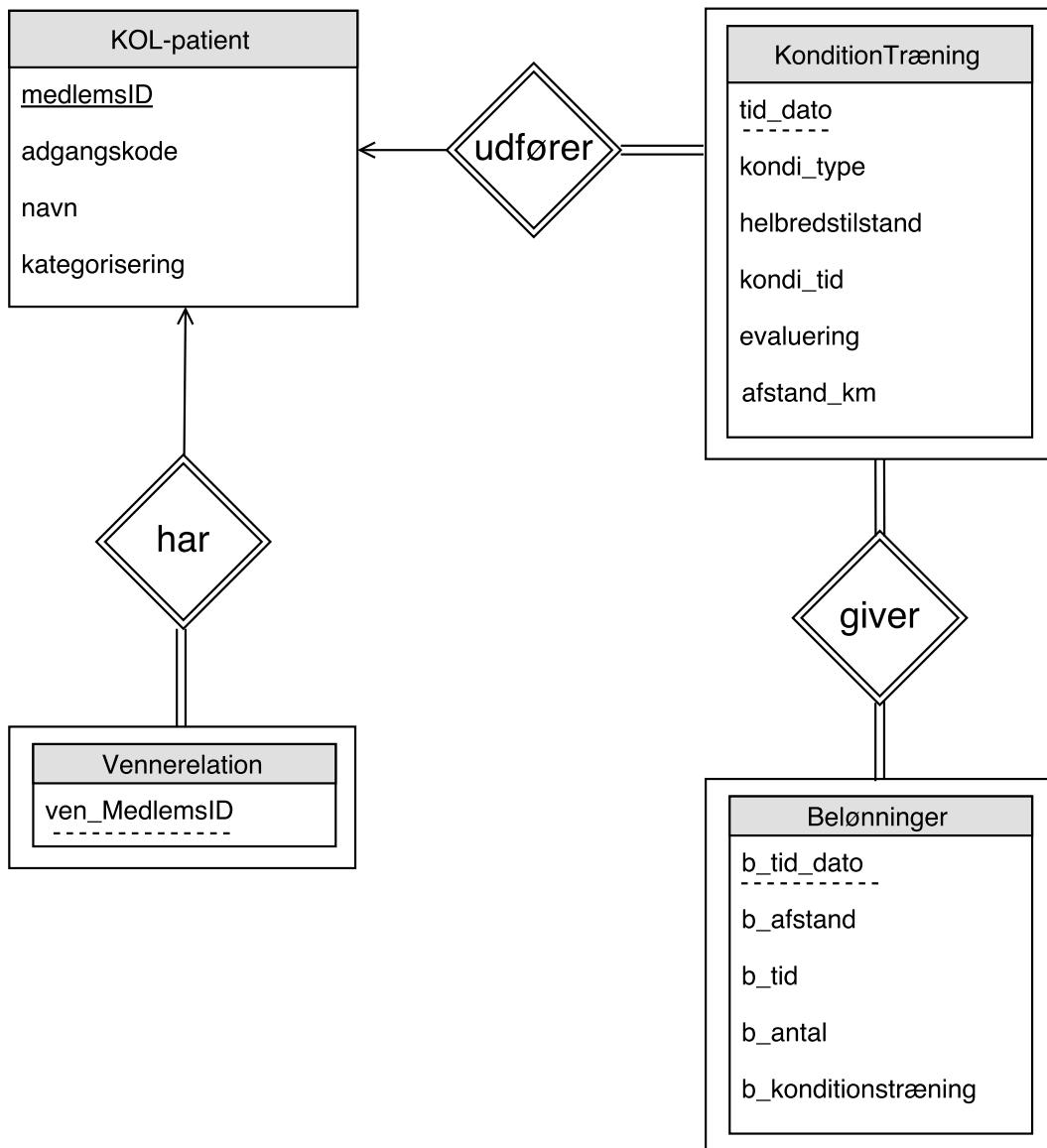
Når brugeren via hovedmenuen trykker på knappen for log ud, vises grænsefladen for *Log ud*. Her har brugeren mulighed for at trykke på BekræftKnap eller FortrydKnap. Hvis brugeren trykker på BekræftKnap, logges brugeren ud ved, at *Brugeroplysninger* og *KonditionResultater* destrueres, hvorefter grænsefladen for *Log ind* vises. Trykker brugeren på FortrydKnap, vises grænsefladen for *Hovedmenu* igen.

### 6.3 Design af database

Det ønskes, at brugerdata er knyttet til den enkelte bruger i databasen. Dette er med henblik på, at app'en ikke skal lagre større mængder data på den mobile enhed samt sikre data i tilfælde af uforudsete hændelser, som eksempelvis tab af mobil enhed.

#### 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 KOL-patient, vennerelation, konditionstræning og belønninger. Disse entiteter har tilhørende attributter, der ses af ER-diagrammet for databasen i figur 6.21.



Figur 6.21: ER-diagram for database.

Af figur 6.21 ses ER-diagrammet over databasen, hvori KOL-patienter oprettes og informationer om patienterne samt deres resultater lagres. Den enkelte KOL-patient er en stærk entitet, som registreres med primærnøglen, *medlemsID*. Derudover fremgår de svage entiteter, herunder *Vennrerelation*, *KonditionTræning* og *Belønninger*. Én KOL-patient har mange vennrerelationer, som kan identificeres ved KOL-patientens *medlemsID* og *ven\_MedlemsID*. Det samme gør sig gældende for konditionstræninger, hvor én KOL-patient kan udføre mange konditionstræninger, der identificeres ved *medlemsID* og *tid\_dato*. Ligeledes kan belønninger, som er en mange til mange relation, identificeres ved *medlemsID* og *b\_tid\_dato*.

## Schema

ER-diagrammet omskrives til schema for at kunne normalisere og implementere databasen. Normaliseringen anvendes med henblik på at reducere redundans og inkonsistens. Første normalform opnås ved at gøre alle attributter atomiske. Navn, fra entiteten, KOL-patient, opdeles i fornavn og efternavn for at opnå første normalform. Når første normalform er opnået, er det muligt at opnå anden normalform ved at gøre alle attributter i en tabel afhængige af en primærnøgle. Herefter kan tredje normalform opnås. For tredje normalform må en attribut ikke være funktionel afhængig af en anden attribut, der er funktionel afhængig af primærnøglen. Schemaet på tredje normalform ses på tabel 6.1.

<b>Stærke entiteter</b>	KOL-patient = ( <i>medlemsID</i> , adgangskode, fornavn, efternavn, kategorisering)
<b>Svage entiteter</b>	Træning = ( <i>medlemsID</i> , <i>tid_dato</i> , <i>kondi_type</i> , helbredstilstand, <i>kondi_tid</i> , <i>afstand_km</i> , evaluering) <i>Vennrerelation</i> = ( <i>medlemsID</i> , <i>venMedlemsID</i> ) <i>Belønninger</i> = ( <i>medlemsID</i> , <i>b_tid_dato</i> , <i>b_afstand</i> , <i>b_tid</i> , <i>b_antal</i> , <i>b_konditionstræning</i> )

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

# Kapitel 7

## Implementering

I dette kapitel beskrives implementeringen af app'en, der omhandler transformationen fra design til kode. Det vil beskrives, hvilken platform koden implementeres gennem samt den tilhørende database. Derudover beskrives implementeringen af designklasserne, der opdeles i grænseflader samt model- og controllerklasser. Ydermere vil de centrale elementer i app'en beskrives. Disse elementer omhandler tilpasning af træningsniveau, træning, resultater samt brugerens venneliste. Overordnet vil beskrivelserne tage udgangspunkt i udpluk af den implementerede kode.

I kapitel 6 er analyse- og designklasser samt funktionsnavne navngivet på dansk, hvorfor bogstaverne æ, ø og å forekommer. Disse symboler anvendes ikke under implementeringen. Metoderne er tidligere, i kapitel 6, defineret som private, nogle af disse er dog implementeret som public, da det ønskes at tilgå disse fra andre klasser.

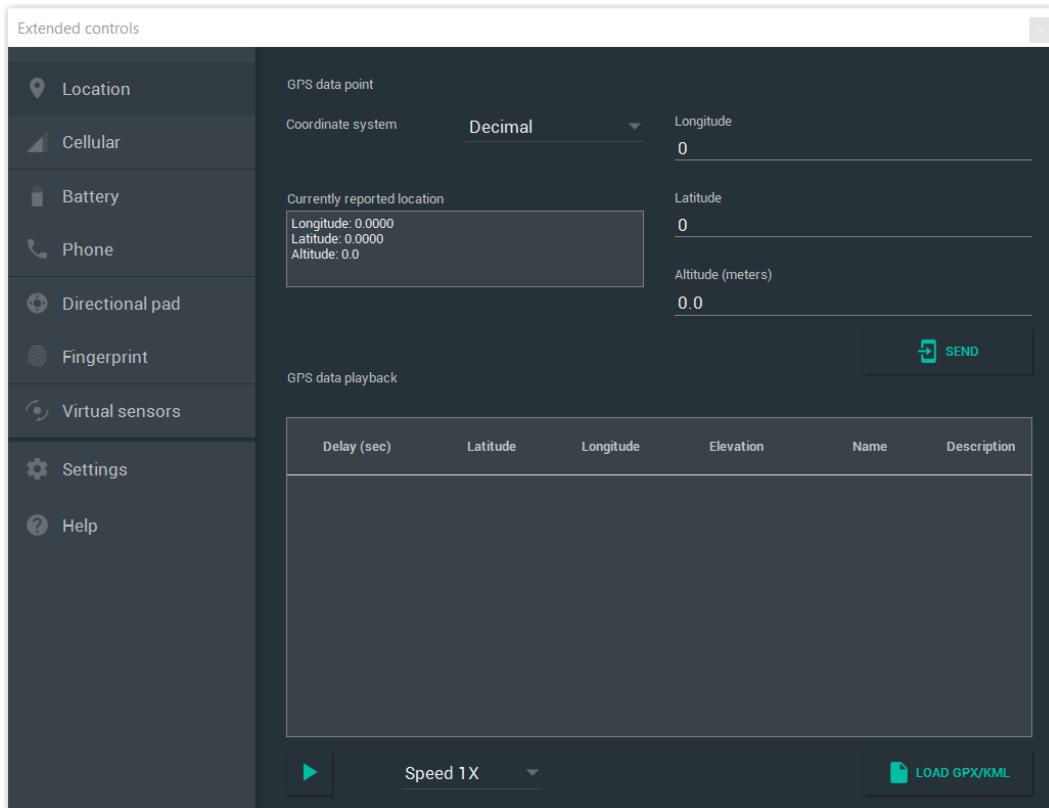
### 7.1 Platform

Det er valgt at implementere app'en i platformen Android Studio version 2.3.1 og programmere i Java, da dette er et objektorienteret programmeringssprog [33]. Android Studio er et officielt Integrated Development Environment (IDE) for udvikling af android app's [44], hvilket er passende for udviklingen af dette projekts problemløsning. Dertil tilbyder Android Studio at teste app's på Android Emulator, som kan simulere en android smartphone. Denne kan ses af figur 7.1, hvoraf der til højre for androidskærmen fremgår et kontrolpanel, hvor funktionaliteter for emulatoren kan tilgås.



Figur 7.1: Android Emulator.

I emulatorens kontrolpanel findes Extended Controls, som fremgår af figur 7.2. Hertil er det muligt at simulere en android smartphones egenskaber, såsom lokalisering.



*Figur 7.2: Android Emulator.*

## Struktur af Android Studio

Strukturen i Android Studio opdeles i *manifests*, *res* samt *java* mapper. *Manifests* mappen indeholder en *AndroidManifest.xml* fil, der er essentiel for, at app'en kan køre og indeholder de aktiviter, der implementeres. Derudover er der i denne fil defineret tilladelser for den givne app. For udvikling af app'en har det været nødvendigt at tillade brug af internet samt GPS-lokalisation, for således at kunne tilgå databasen og beregne afstand. *Res* mappen indeholder ikke-kode ressourcer, der eksempelvis er de forskellige layouts, som udgør grænsefladerne. Javaklasserne, der er oprettet i forbindelse med udviklingen af app'en er placeret i *java* mappen.[44]

## 7.2 Database

Til implementering af databasen samt kommunikation med denne benyttes programmet XAMPP. Programmet opretter en lokal apache webserver samt database på en given computer, der fungerer som et Localhost miljø. Den lokale webserver simulerer en ekstern webserver, hvilket giver et passende miljø til at teste og udvikle prototypen. Dertil anses dette ikke som værende fjernt fra et virkelighedsnært scenarie, hvor systemet benyttes med ekstern server og database. Ved direkte håndtering af databasen benyttes phpMyAdmin, der er et online databaseadministrationssystem, som understøtter Structured Query Language

(SQL) [45]. Databasen er implementeret under navnet *db\_KOL*, hvori tabeller oprettes på baggrund af design jf. afsnit 6.3 i relation til attributter og datatyper.

## Kommunikation med databasen

Kommunikation mellem Android Studio og databasen kan ikke forekomme direkte, hvorfor php: Hypertext preprocessor scripts benyttes. Php er et Server-Side Scripting Language, hvilket køres på serveren og muliggør, at systemet kan tilgå databasen [45]. Dertil kan de oprettede php-scripts betragtes som controlleren for database, der er designet i afsnit 6.2. Der er oprettet et php-script, *config.php*, der indeholder informationerne host, bruger, adgangskode samt navn på den oprettede database. Dette script inkluderes i et separat script, *DB\_connect.php*, til at etablere forbindelsen til databasen. Der er opstillet forskellige php-scripts, som javakoden refererer til via URL links. Af disse links fremgår ip-adressen for serveren samt filplaceringen af det givne script.

De forskellige scripts får information fra app'en for således at kunne udføre SQL-kommandoer. Informationen sendes fra app'en som en JavaScript Object Notation (JSON)-repræsentation. JSON benyttes ved dataoverførsel, der gør det muligt at pakke data som et objekt eller array.[45] Php-scripts brugt i dette projekt er sat op til at tjekke, hvorvidt der bliver sendt en værdi i det respektive navn. Er værdierne sat og ikke NULL, ligges værdierne i en ny variabel. Et eksempel af dette ses af figur 7.3, hvor et udklip af php-scriptet for log ind er visualiseret.

```

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

```

*Figur 7.3:* Udklip af log ind php-script, hvor medlemsid samt adgangskode modtages og indsættes i tilhørende variabler, der benyttes i et funktionskald.

Dette udklip viser, hvordan koden modtages og gemmes i nye variabler. Værdierne gemmes i et associativt array, der ved hjælp af en POST-metode gør det muligt at overføre data til et andet script, *db\_functions.php*, hvori SQL-kommandoer udføres [46]. Linje 21 viser, hvordan variablerne benyttes i et funktionskald, der eksekveres i *db\_functitons.php*. Af figur 7.4 ses et udklip af denne funktion.

```

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

```

*Figur 7.4:* Udklip af php-scripts for funktionen log ind. Herunder ses SQL-kommandoen for denne funktion.

Det ses af figur 7.4, hvordan en SQL-kommando udføres. Tabellen *users* er tidligere i afsnit 6.3 defineret som KOL-patient. Den viste SQL-kommando henter alt fra tabellen *users* tilhørende medlemsid'et i databasen, der er lig spørgsmåltegn. Spørgsmåltegnet markerer et

bindingspunkt, hvorpå en parameter kan tilknyttes. Parameteren, der bindes på ved brug af `bind_param`, fremgår af linje 176, hvoraf `"s"` indikerer, at medlemsid'et er af typen string. I dette tilfælde valideres log ind-informationerne, førend user returneres til log ind php-scriptet, der håndterer user, hvilket fremgår af figur 7.5.

```

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

```

**Figur 7.5:** Udklip af php-scripts, hvori user håndteres og sendes til app'en.

Som det ses af figur 7.5 opstilles if/else-statement, der håndterer, hvorvidt `user` returneres. Hvis `user` returneres bliver brugerdata gemt i `response`, der sendes tilbage til app'en som en JSON-repræsentation. Returneres `user` ikke, gemmes en fejlmeldelse i `response`, der ligeledes sendes til app'en som en JSON-repræsentation.

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

### 7.3 Grænseflader

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

Elementer, såsom `TextView` og `Button`, opstilles i layouts med type, størrelse samt orientering. Forekommer det, at elementerne skal benyttes i javakoden, defineres disse ligeledes med et id. Et eksempel af layoutkode fra log ind ses af figur 7.6.

```

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

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

```

**Figur 7.6:** Udklip af kode fra log ind-layout. Udkippet viser et `EditText`, hvori brugeren angiver adgangskode samt en `Button` for log ind.

Af dette udklip ses koden for tekstfeltet, hvori brugeren angiver adgangskode samt koden for layoutet af log ind-knappen. Feltet for adgangskoden er her af typen *EditText*, der tillader, at brugeren kan angive tekst. Dette felt har ligeledes et id, der muliggør at referere til fletet og hente den angivne adgangskode til validering af log ind. Knappen er opstillet af typen *Button* og har ligeledes et id, hvortil en listener er opstillet i javakoden. Dette ses yderligere af figur 7.8.

## 7.4 Model- og controllerklasser

Model- og controllerklasser er implementeret i Java Class-filer. Heri er klasser defineret med navn samt tilhørende attributter og metoder. Af figur 7.7 ses et eksempel på, hvordan en controller er implementeret.

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

**Figur 7.7:** Udpluk af javaklassen for *LoginController*. Punktummerne symboliserer den resterende kode, der ikke anses nødvendig for forklaring af opsætning af javaklasser.

Det fremgår af dette udklip, at klassen, *LoginController*, er af typen *public* og nedarver *activity*. Dette gør sig gældende for samtlige klasser implementeret. Denne klasse for *LoginController* har attributten *btnLogin*, der refererer til log ind-knappen på grænsefladen. Idet app'en skal reagere, når brugeren trykker på log ind-knappen, opsættes en listener på knappen, hvilket ses af figur 7.8.

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

**Figur 7.8:** Udpluk af koden for listeneren opsat for log ind-knappen. Punktummerne symboliserer den resterende kode, der ikke anses nødvendig for forklaring af opsætning af listener.

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

Modelklasserne er implementeret på samme vis som controllerklasserne, de har dog til formål at lagre data midlertidigt. I disse klasser er der ligeledes opstillet attributter med tilhørende get- og set-metoder.

## 7.5 Tilpasning af træningsniveau

I tilpasningscontrolleren beregnes den anbefalede træningstid. Dette gøres ud fra kategorisering, den daglige helbredstilstand og en tidligere evaluering, foretaget efter samme træningsform,

-type og heldbredstilstand. Beregningen for den anbefalede træning er implementeret ved brug af if/else-statement. Et udpluk af denne beregning fremgår af figur 7.9.

```

242     public void beregnanbefaling(final int heldbredstilstand, final int evaluering, final String kategorisering) {
243         TextView TVAnbefaling = (TextView) findViewById(R.id.TVAnbefaling);
244
245         if (kategorisering.equals("A")) {
246             if (heldbredstilstand == 2 && evaluering == 1) {
247                 TVAnbefaling.setText("Din anbefalede træningstid er 15 minutter");
248                 setMinutter(15);
249             } else if (heldbredstilstand == 2 && evaluering == 0 || heldbredstilstand == 2 && evaluering == 2) {
250                 TVAnbefaling.setText("Din anbefalede træningstid er 20 minutter");
251                 setMinutter(20);
252             } else if (heldbredstilstand == 2 && evaluering == 3) {
253                 TVAnbefaling.setText("Din anbefalede træningstid er 25 minutter");
254                 setMinutter(25);
255         }

```

**Figur 7.9:** Utpluk af koden for anbefalet træning. If/else-loops afgør, hvilket træningsniveau brugeren får anbefalet.

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

## 7.6 Træning

Træningscontrolleren har til formål at håndtere GPS-lokalisation for at beregne tilbagelagt afstand under træning. Ligeledes håndterer controlleren en timer, der viser tiden brugt til den givne træning. Efter træningen er udført, vil træningscontrolleren opsætte en notifikation som efterfølgende vises dagligt.

### Afstandsberegning

For at implementere GPS skal app'en have tilladelse til at anvende den mobile enheds lokalisation. Dette er en engangstilladelse, der skal gives første gang brugeren vil foretage en træning.

Der instansieres en LocationManager og LocationListener. LocationManager tillader app'en at opnå enhedens lokalisation, hvor LocationListener modtager lokalisationsopdateringer. Disse opdateringer forekommer, idet enheden ændrer lokalisation.[47, 48] Af koden i figur 7.10 fremgår det, at systemet indeholder en metode, *onLocationChanged*, der kaldes, når lokalisationen ændres.

```

194     locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);
195     locationListener = new LocationListener() {
196         @Override
197         public void onLocationChanged(Location location) {
198
199             lat2 = location.getLatitude();
200             lng2 = location.getLongitude();
201
202             setLat2(lat2);
203             setLng2(lng2);
204     }

```

**Figur 7.10:** Metoden, *onLocationChanged*, der kaldes, når lokalisation ændres.

Når lokalisationen ændres, hentes enhedens longitude og latitude, hvortil disse værdier benyttes i set-metoder. Afstanden beregnes ud fra følgende formel:

$$d = \arccos(\sin(lat_1) * \sin(lat_2) + \cos(lat_1) * \cos(lat_2) * \cos(lon_2 - lon_1)) * R \quad (7.1)$$

Formlen, der ses af ligning 7.1, benyttes til at udregne afstand ud fra to longitude- og latitudepunkter. Af formlen betegner  $d$  den udregnede afstand,  $lat_1$ ,  $lon_1$  samt  $lat_2$ ,  $lon_2$  er latitude- og longitudekoordinaterne for henholdsvis den første og anden lokalisation.  $R$  betegner jordens radius.[49]

Ligning 7.1 implementeres i javakode, hvortil denne benytter longitude- og latitudekoordinaterne, der hentes fra koden, som ses af figur 7.10. Implementeringen af formlen ses af figur 7.11.

```

301     private double distance(double lat1, double lon1, double lat2, double lon2) {
302         double gammelDist = getnyDist();
303         double theta = lon1 - lon2;
304         double dist = Math.sin(deg2rad(lat1))
305             * Math.sin(deg2rad(lat2))
306             + Math.cos(deg2rad(lat1))
307             * Math.cos(deg2rad(lat2))
308             * Math.cos(deg2rad(theta));
309         dist = Math.acos(dist);
310         dist = (dist * 6371);
311         double nyDist = gammelDist + dist;
312         textViewKm.setText(String.format("%.2f", nyDist) + " km");
313         setnyDist(nyDist);
314         return (dist);
315     }

```

**Figur 7.11:** Metode i træningscontrolleren, der beregner afstanden i kilometer ud fra lokalisationer.

Metoden, *distance*, kaldes hver gang, der sker en lokalisationsændring og beregner afstanden mellem enhedens pågældende lokalisation samt forrige lokalisationsopdatering. Af koden fremgår det, at latitude- og longitudeværdierne konverteres fra grader til radianer for at kunne anvende cosinus- og sinusfunktionerne. Afstanden ønskes udregnet i kilometer, hvortil *dist* ganges med jordens radius i kilometer. Efterfølgende sættes den udregnede totale afstand *nyDist*, hvortil denne værdi hentes og gemmes i variablen *gammelDist*.

## Timer

I træningscontrolleren startes en timer for at kunne monitorere træning. Denne er implementeret ved at instansiere klassen *Runnable*, hvori en metode, *run*, defineres. Koden for dette kan ses i figur 7.12.

```

73
74     public Runnable runnable = new Runnable() {
75         public void run() {
76             MillisecondTime = SystemClock.uptimeMillis() - StartTime;
77             UpdateTime = TimeBuff + MillisecondTime;
78             Seconds = (int) (UpdateTime / 1000);
79             Minutes = Seconds / 60;
80             Hours = Minutes / 60;
81             Seconds = Seconds % 60;
82             Minutes = Minutes % 60;
83
84             textViewTimer.setText(" " + Hours + ":" + "" + Minutes + ":" + String.format("%02d", Seconds));
85         }
86     }

```

**Figur 7.12:** Udklip af kode fra træningscontrolleren. Udkippet viser metoden for timeren.

Metoden, *run*, indeholder variablen *MillisecondTime*, der tæller tiden, som er gået siden brugeren har trykket start træning. Vælger brugeren at trykke stop træning, stopper *MillisecondTime* med at tælle op, hvortil denne værdi gemmes i variablen *UpdateTime*. Værdien gemmes for, at timeren vil kunne fortsætte, hvis brugeren ønsker at fortsætte træningen. *Seconds* defineres ud fra *UpdateTime*, der måles i millisekunder. Det defineres yderligere, at *Seconds* kun kan tælle op til 60.

Under tilpasning af træningsniveau er der blevet anbefalet en træningstid. Opnår træningstiden denne værdi under træning, vil app'en gøre brugeren opmærksom på dette ved angivelse af en lyd. Dette er implementeret ved en if-loop, hvori klassen MediaPlayer instansieres, hvis den anbefalede tid opnås. Hertil kaldes en metode, der afspiller en lydfil.

## Notifikation

Ifølge de opstillede kravspecifikationer skal app'en sende en daglig notifikation. Denne er implementeret i træningscontrolleren, hvor notifikationen aktiveres første gang brugeren har evalueret en træning. Hertil er det implementeret, ved brug af klassen *Calendar* og metoden *getInstance*, at den daglige notifikation sendes klokken 15. Af figur 7.13 fremgår et udklip af koden, som definerer tidspunktet for notifikationen.

```

475     Calendar calendar = Calendar.getInstance();
476     Calendar now = Calendar.getInstance();
477     calendar.set(Calendar.HOUR_OF_DAY, 15);
478     calendar.set(Calendar.MINUTE, 0);
479     calendar.set(Calendar.SECOND, 0);
480     if(now.after(calendar)) {
481         calendar.add(Calendar.DATE, 1);
482     }

```

**Figur 7.13:** Udklip af kode fra træningscontrolleren, som viser notifikationens starttidspunkt.

## 7.7 Resultater

For at brugeren kan følge sin egen udvikling er der implementeret et BarChart. Denne graf viser tiden, brugeren har trænet for den givne uge. Af denne graf plottes en træningstid for hver af ugens dage. Af figur 7.14 fremgår et udpluk af implementeringen for den opstillede graf.

```

130     ArrayList<BarEntry> barEntries = new ArrayList<>();
131     barEntries.add(new BarEntry(data[0]/60,0));
132     barEntries.add(new BarEntry(data[1]/60,1));
133     barEntries.add(new BarEntry(data[2]/60,2));

```

**Figur 7.14:** Udklip af javakode for BarChart, der senere har til formål at illustrere brugerens ugentlige træningstid.

Der ses af dette udklip, at der oprettes et ArrayList under navnet *barEntries*. Dette array anvendes til at lagre mængden af data, der ønskes plottet. Dertil ses det, hvorledes nye elementer tilføjes dette array ud fra en værdi og index. Værdien, der instættes tages fra et andet array af navnet *data*, hvori der lagres en træningstid for ugens dage. Plads nummer 0 refererer til den første dag i ugen og tæller således op efter ugens dage. *Data* er divideret med 60, da træningstiden er gemt som sekunder og ønskes plottet i minutter.

Belønninger gives udfra samlet tid, afstand, antal træninger samt antal konditionstræninger og beregnes efter endt træning. Da disse belønninger er baseret ud fra den samlede træning, hentes den sammenlagte tid, afstand samt totale antal træninger fra databasen og omregnes til et tal mellem 1 til 6, der repræsenterer antallet af stjerner brugeren har opnået. Et udklip af javakoden for omregningen for brugere med en kategorisering *A* ses af figur 7.15.

```

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

```

**Figur 7.15:** Udklip af javakode for omregning af samlet afstand til optjente belønninger for brugere med kategorisering *A*.

Det fremgår af dette javaudklip, at omregningen foregår ved if/else-loops. Hvis afstanden ligger indenfor et interval af 5 og 25, sættes *b\_afstand* lig 1, hvilket repræsenterer én stjerne. Af tabel 7.1 fremgår kriterierne for at opnå stjerner for brugere med kategoriseringen *A*.

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

**Tabell 7.1:** Kriterier for at opnå belønninger inden for henholdsvis samlet tid, afstand og total antal træninger samt konditionstræninger for brugere med kategorisering A.

## 7.8 Venneliste

I vennelistecontrolleren er der implementeret en venneliste, som indeholder brugere, der følges. For at app'en kan hente venner som brugeren følger, sendes brugerens medlemsID til databasen. Et php-script benytter medlemsid i en SQL-kommando, der tilgår tabellen for vennerelationer, som returnerer *ven\_medlemsid* med tilhørende navn for de brugere, der følges. Dette modtages i app'en som et JSON-objekt, hvorfra et JSON-array hentes og ligges i et array. Dette kan ses af udkippet i figur 7.16.

```

209     venner_array = json.getJSONArray("venner");
210     if(!error){
211         for(int i = 0; i < venner_array.length(); i++){
212             JSONObject c = venner_array.getJSONObject(i);
213             // Vi lagrer det i en variabel
214             String navn = c.getString("navn");
215             String ven_medlemsid = c.getString("ven_medlemsid");
216             //Hashmap
217             HashMap<String, String> map = new HashMap<String, String>();
218             // Værdier i Hashmap
219             map.put("navn", navn);
220             map.put("ven_medlemsid", ven_medlemsid);
221             // Tilføjer HashList to ArrayList VenneListe
222             VenneListe.add(map);

```

**Figur 7.16:** Utpluk af koden for venneliste. Koden viser et udpluk af, hvordan venners navn og medlemsid gemmes i arraylisten, VenneListe.

Af koden i figuren er der opsat en for-loop, hvori JSON-objekter hentes fra alle placeringerne i et array kaldet *venner\_array*. Hvert af disse objekter repræsenterer én af brugerens venner, hvor værdier for *navn* og *ven\_medlemsid* er hentet. Af for-loopen ses det, at objekterne gemmes i et nyt array af navnet *VenneListe*, som senere benyttes til at opstille vennerne i ListView for grænsefladen. Grundet bruget af *HashMap* til at håndtere *navn* og *ven\_medlemsid*, har det været nødvendigt at håndtere medlemsid som en string. Dette skyldes, at ved instansiering af Hashmappet defineres inputstyperne som string, idet navnet er af typen string.

Der er i app'en ligeledes mulighed for at tilføje og fjerne brugere fra vennelisten. Muligheden for at tilføje en bruger til vennelisten er implementeret ved at benytte SQL-kommandoen, der ses af figur 7.17.

```

105     public function OpretVennerelation($medlemsid, $ven_medlemsid) {
106         $stmt = $this->conn->prepare(
107             "INSERT INTO vennerelation(medlemsid, ven_medlemsid) VALUES (?, ?)" );
108         $stmt->bind_param("ss", $medlemsid, $ven_medlemsid);
109         $result = $stmt->execute();
110         $stmt->close();

```

*Figur 7.17: SQL-kommando for tilføj bruger til venneliste.*

I ovenstående kode ses det, at medlemsid og ven\_medlemsid indsættes i tabellen, Vennerelation, hvorved der dannes en vennerelation. Fjernes en bruger fra vennelisten, anvendes SQL-kommandoen, der fremgår af figur 7.18. Dertil er forskellen fra at oprette en vennerelation, at SQL-kommandoen anvender delete, og dermed fjerner vennerelationen fra tabellen i databasen. Hertil fjernes relationen kun, hvis både medlemsID for bruger og ven indgår.

```

126     public function SletVennerelation($medlemsid, $ven_medlemsid) {
127         $stmt = $this->conn->prepare(
128             "DELETE FROM vennerelation WHERE medlemsid = ? AND ven_medlemsid = ?" );
129         $stmt->bind_param("ss", $medlemsid, $ven_medlemsid);
130         $result = $stmt->execute();
131         $stmt->close();

```

*Figur 7.18: SQL-kommando for fjern bruger til venneliste.*

# Kapitel 8

## Test

---

I dette kapitel beskrives test af app'en. Der tages udgangspunkt i at teste kravspecifikationer, som blev opstillet i afsnit 5.2, hvorefter der blev opstillet et use case diagram 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. Handlingen, der sker i main flowet, er markeret med kursiv. Efter hvert main flow beskrives det forventede output, hvorefter det samlede resultat af testen vil fremgå af resultat. De forskellige tests er foretaget ved brug af Android Emulator, da den lokale database kan tilgå fra denne.

### 8.1 Database

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

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

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

<b>Test:</b>	Database
<b>Formål:</b>	Formålet er at oprette brugere i databasen samt hente og sende data. Dette gøres ved at oprette en bruger i databasen. Efterfølgende vil en kategorisering foretages, og derefter tjekkes om denne 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:	<p>1. Indtast SQL-forespørgsel <code>INSERT INTO 'users' ('navn', 'medlemsid', 'db_adgangskode', 'kategorisering') VALUES ('Jens Jensen', '11170301', 'adgangskode', 'F')</code> i databasen.</p> <ul style="list-style-type: none"> <li>• Bruger er oprettet i databasen</li> </ul> <p>2. Åben app'en og log ind med medlemsID, 11170301, og adgangskode, <i>adgangskode</i>. Udfør kategorisering og få en samlet CAT-score 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 <i>INDLÆGGELSER</i>.</p> <ul style="list-style-type: none"> <li>• Kategorisering A er gemt i databasen (<b>Figur 1 og 2</b>)</li> </ul> <p>3. Tryk på <i>REDIGER ADGANGSKODE</i> via hovedmenuen.</p> <ul style="list-style-type: none"> <li>• MedlemsID og navn samt opdaterede kategorisering, A, vises under rediger adgangskode (<b>Figur 3</b>)</li> </ul>
Resultat:	 <p><b>Figur 1</b></p>  <p><b>Figur 2</b></p>  <p><b>Figur 3</b></p>

Resultater for test af database fremgår af ovenstående figurer. Til venstre er der opnået en kategorisering svarende til A, som på figuren i midten viser, at kategoriseringen er gemt i databasen. Til højre vises redigering af adgangskode, hvor brugeroplysninger om den oprettede bruger, Jens Jensen, vises. Disse værdier er hentet fra databasen, idet brugeren logger ind. Dertil er kategoriseringen opdateret i forhold til den nye kategorisering.

- ✓ På baggrund af denne test er kravene for databasen opfyldt

*Tabel 8.1: Test af database.*

## 8.2 Log ind

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

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

Til at teste, hvorvidt kravene til log ind er opfyldt, udføres testen, der fremgår af tabel 8.2.

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

**Tabel 8.2:** Test af Log ind.

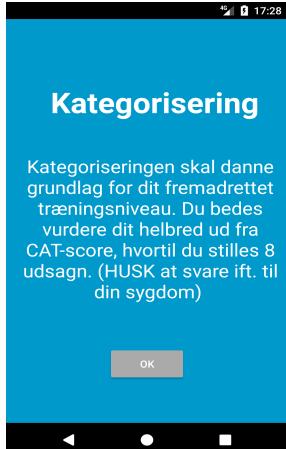
### 8.3 Kategorisering

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

- Systemet skal kunne kategorisere brugere i ABCD på baggrund af CAT-score og antallet af årlige indlæggelser på grund af KOL

*Dette er nødvendigt for at kunne tilpasse træningen efter den enkelte bruger*

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

Test:	Kategorisering
<b>Formål:</b>	Formålet er, at systemet skal kunne kategorisere brugere i ABCD efter, at brugeren har svaret på udsagn fra CAT-score og angivet antallet af indlæggelser forårsaget af KOL inden for det seneste år. Dette gøres ved at angive forskellige værdier, der giver kategorisering A, B, C, eller D.
<b>Main flow:</b>	<p>1. Få en samlet CAT-score under 10. Vælg <i>0,1,2,1,0,1,0,1</i>, tryk videre efter hver angivet værdi og vælg til sidst antal indlæggelser <i>0 INDLÆGGELSER</i> (<b>Figur 1, 2 og 3</b>).</p> <ul style="list-style-type: none"> <li>• Kategorisering er A (<b>Figur 4</b>)</li> </ul> <p>2. Få en samlet CAT-score over 10. Vælg <i>5,1,2,3,4,5,1,5</i>, tryk videre efter hver angivet værdi og vælg til sidst antal indlæggelser <i>0 INDLÆGGELSER</i>.</p> <ul style="list-style-type: none"> <li>• Kategorisering er B (<b>Figur 5</b>)</li> </ul> <p>3. Få en samlet CAT-score under 10. Vælg <i>0,1,2,1,0,1,0,1</i>, tryk videre efter hver angivet værdi og vælg til sidst antal indlæggelser <i>1 ELLER FLERE INDLÆGGELSER</i>.</p> <ul style="list-style-type: none"> <li>• Kategorisering er C (<b>Figur 6</b>)</li> </ul> <p>4. Få en samlet CAT-score over 10. Vælg <i>5,1,2,3,4,5,1,5</i>, tryk videre efter hver angivet værdi og vælg til sidst antal indlæggelser <i>1 ELLER FLERE INDLÆGGELSER</i>.</p> <ul style="list-style-type: none"> <li>• Kategorisering er D (<b>Figur 7</b>)</li> </ul>
<b>Resultat:</b>	 <p><b>Figur 1</b></p>  <p><b>Figur 2</b></p>  <p><b>Figur 3</b></p> <p>Ovenstående figurer viser grænsefladerne for introduktion til kategorisering, CAT-score og antal indlæggelser. Den midterste figur viser et af de otte udsagn, der udgør CAT-scoren. Disse grænseflader vises før grænsefladen for kategorisering.</p>

**Figur 4**      **Figur 5**      **Figur 6**      **Figur 7**

Resultater for test af kategorisering fremgår af ovenstående figurer. Fra venstre mod højre ses, at henholdsvis første til fjerde main flow viser de forventede kategoriseringer.

✓ På baggrund af denne test er kravet for kategorisering opfyldt

**Tabel 8.3:** Test af kategorisering.

## 8.4 Tilpasning af træningsniveau

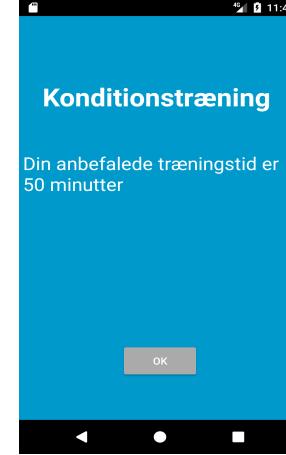
I tilpasning af træningsniveau skal brugeren oplyses et anbefalet træningsniveau, med henblik på at tage højde for daglige variationer, ved at anvende parametrene kategorisering, daglig helbredstilstand og eventuel tidligere evaluering. Følgende krav blev opstillet til tilpasning af træningsnivauet:

- Brugere skal kunne angive deres daglige helbredstilstand

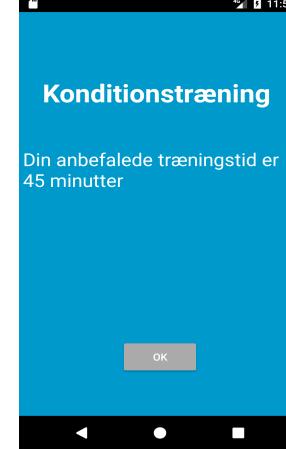
*Dette er nødvendigt for tage højde for daglige variationer og derved tilpasse træningen for den enkelte bruger*

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

<b>Test:</b>	Tilpasning af træningsniveau uden evaluering
<b>Formål:</b>	Formålet er, at brugeren skal kunne angive ønsket træningsform, træningstype samt daglig helbredstilstand, hvorefter systemet, på baggrund af dette samt brugerens kategorisering, anbefaler 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 <i>TRÆNING, KONDITIONSTRÆNING, LØB, MEGET DÅRLIGT</i> og tryk videre efter hver handling (<i>Figur 1, 2 og 3</i>).</p> <ul style="list-style-type: none"> <li>• Anbefaling af træningstid er 10 min (<i>Figur 4</i>)</li> </ul> <p>2. Vælg <i>TRÆNING, KONDITIONSTRÆNING, GÅ, MODERAT</i> og tryk videre efter hver handling.</p> <ul style="list-style-type: none"> <li>• Anbefaling af træningstid er 30 min (<i>Figur 5</i>)</li> </ul> <p>3. Vælg <i>TRÆNING, KONDITIONSTRÆNING, CYKEL, MEGET GODT</i> og tryk videre efter hver handling.</p> <ul style="list-style-type: none"> <li>• Anbefaling af træningstid er 50 min (<i>Figur 6</i>)</li> </ul>		
Resultat:	 <p><i>Figur 1</i></p>  <p><i>Figur 2</i></p>  <p><i>Figur 3</i></p>		
	<p>Ovenstående figurer viser grænsefladerne for valg af træningsform, trænings-type og daglig helbredstilstand. Disse grænseflader vises før grænsefladen for anbefalet træningstid.</p>  <p><i>Figur 4</i></p>  <p><i>Figur 5</i></p>  <p><i>Figur 6</i></p> <p>Resultater for test af tilpasning af træningsniveau uden evaluering fremgår af ovenstående figurer. Fra venstre mod højre ses, at henholdsvis første til tredje main flow viser de forventede anbefalinger af træningstider.</p>		

*Tabel 8.4: Test af tilpasning af træningsniveau uden evaluering.*

<b>Test:</b>	Tilpasning af træningsniveau med evaluering
<b>Formål:</b>	Formålet er, at brugeren skal kunne angive ønsket træningsform, træningstype samt daglig helbredstilstand, hvorefter systemet på baggrund af dette samt kategorisering og tidligere evaluering skal anbefale et træningsniveau. Dette gøres ved at angive samme træningsform, og vælge mellem de tre forskellige træningstyper og helbredstilstande. Brugeren er i kategoriseringen A og har forinden træningen angivet evaluering for samme træning samt helbredstilstand.
<b>Main flow:</b>	<p>1. Tidligere evaluering af samme træning er :-D og helbredstilstand meget dårligt. Vælg <i>TRÆNING, KONDITIONSTRÆNING, LØB, MEGET DÅRLIGT</i> og tryk videre efter hver handling.</p> <ul style="list-style-type: none"> <li>• Anbefaling af træningstid er 15 min (<i>Figur 1</i>)</li> </ul> <p>2. Tidligere evaluering af samme træning er :-) og helbredstilstand moderat. Vælg <i>TRÆNING, KONDITIONSTRÆNING, GÅ, MODERAT</i> og tryk videre efter hver handling.</p> <ul style="list-style-type: none"> <li>• Anbefaling af træningstid er 30 min (<i>Figur 2</i>)</li> </ul> <p>3. Tidligere evaluering af samme træning er :-( og helbredstilstand meget godt. Vælg <i>TRÆNING, KONDITIONSTRÆNING, CYKEL, MEGET GODT</i> og tryk videre efter hver handling.</p> <ul style="list-style-type: none"> <li>• Anbefaling af træningstid er 45 min (<i>Figur 3</i>)</li> </ul>
<b>Resultat:</b>	 <p><i>Figur 1</i></p>  <p><i>Figur 2</i></p>  <p><i>Figur 3</i></p> <p>Resultater for test af tilpasning af træningsniveau med evaluering fremgår af ovenstående figurer. Fra venstre mod højre ses, at henholdsvis første til tredje main flow viser de forventede anbefalinger af træningstider.</p> <p>✓ På baggrund af test af tilpasning af træningsniveau uden og med evaluering er kravet for tilpasning af træningsniveau opfyldt</p>

*Tabel 8.5: Test af tilpasning af træningsniveau med evaluering.*

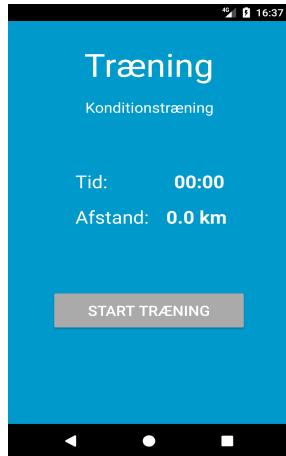
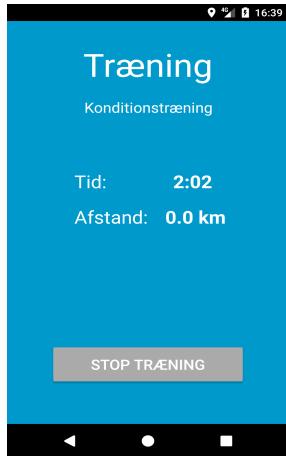
## 8.5 Træning

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

- Systemet skal kunne måle tid via timer og afstand via GPS  
*Dette er nødvendigt for at monitorere træningen*
- Brugere skal kunne evaluere hver træning  
*Dette er nødvendigt for at tilpasse træningen yderligere efter den enkelte bruger*
- Systemet skal kunne sende en daglig notifikation for således at påminde brugeren om træning  
*Dette er nødvendigt for at kunne motivere brugere til at udføre træning*

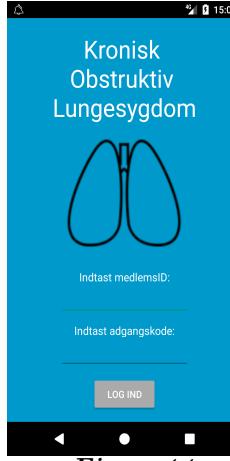
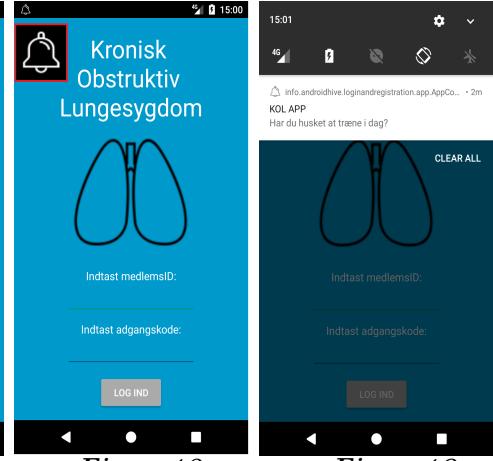
For at teste, hvorvidt de opstillede krav til træningen er opfyldt, udføres testen, som fremgår af tabel 8.6.

Test:	Træning
<b>Formål:</b>	Formålet er, at systemet skal kunne måle tid og afstand under træningen. Når brugeren har afsluttet træning, skal måleenhederne stoppe, og brugeren skal kunne angive evaluering. Dette gøres ved at måle tid og afstand samt evaluere træningen efterfølgende. Derudover skal en notifikation forekomme dagligt.
<b>Main flow:</b>	<ol style="list-style-type: none"> <li>1. Tryk <i>START TRÆNING</i> (<b>Figur 1</b>) og lad træningen forløbe i minimum 2 minutter. Tryk herefter <i>STOP TRÆNING</i>. (<b>Figur 3</b>)           <ul style="list-style-type: none"> <li>• Tiden er over 2 minutter (<b>Figur 2</b>)</li> </ul> </li> <li>2. Tryk <i>START TRÆNING</i> åben Extended controls, som fremgår af figur 7.2. 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 <i>STOP TRÆNING</i>. De forventede afstande er beregnet ved ligning 7.1.           <ul style="list-style-type: none"> <li>• Afstand ved 0 er 0 km (<b>Figur 4</b>)</li> <li>• Afstand ved 0.01 er 1.57 km (<b>Figur 5</b>)</li> <li>• Afstand ved 0.02 er 3.14 km (<b>Figur 6</b>)</li> </ul> </li> <li>3. Tryk <i>START TRÆNING</i> og tryk herefter <i>STOP TRÆNING</i> og bekræft. Angiv herefter evalueringen :-) (<b>Figur 7 og 8</b>) og tryk videre.           <ul style="list-style-type: none"> <li>• Evaluering er gemt i databasen (<b>Figur 9</b>)</li> </ul> </li> <li>4. Vent til klokken 15 med at udføre en træning           <ul style="list-style-type: none"> <li>• Notifikation starter klokken 15 (<b>Figur 11, 12 og 13</b>)</li> </ul> </li> </ol>

<b>Resultat main flow 1:</b>	 <p><b>Figur 1</b></p>	 <p><b>Figur 2</b></p>	 <p><b>Figur 3</b></p>
<b>Resultat main flow 2:</b>	 <p><b>Figur 4</b></p>	 <p><b>Figur 5</b></p>	 <p><b>Figur 6</b></p>

Resultater for det første main flow fremgår af ovenstående figurer. På venstre ses grænsefladen for træning før træningen er på begyndt. Når der trykkes på startknappen for træning, starter timer, som kan ses af den midterste figur. Til højre fremgår et popup-vindue, der indikerer, at træningen er stoppet.

Resultater for det andet main flow ses af ovenstående figurer. Til venstre ses grænsefladen for træningen, hvor det er simuleret, at brugeren har bevæget sig 0 km. I midten er det simuleret, at brugeren har bevæget sig 1.57 km. Til højre er det simuleret, at brugeren har bevæget sig 3.14 km.

<b>Resultat main flow 3:</b>			
<b>Resultat main flow 4:</b>			
<b>Resultat:</b>	<p>✓ På baggrund af test for main flow er kravene for træning opfyldt</p>		

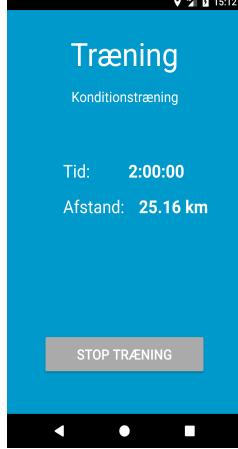
*Tabel 8.6: Test af træning.*

## 8.6 Resultater

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

- Systemet skal kunne vise brugerens ugentlige træningsudvikling  
*Dette er nødvendigt for, at brugere kan følge sin ugentlige udvikling samt motivere brugeren*
- Systemet skal kunne give virtuelle belønninger  
*Dette er nødvendigt for at kunne motivere brugere til at udføre træning*

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

Test:	Resultater
Formål:	Formålet er, at systemet skal kunne vise den ugentlige træningsudvikling grafisk samt give virtuelle belønninger efter tabel 7.1.
Main flow:	<p>1. Tryk på <i>TRÆNING</i> via hovedmenu. <i>START TRÆNING</i> og vent 2 timer. Imens åbnes Extended controls og sæt longitude samt latitude til 0 og tryk send. Ændre herefter longitude og latitude til 0.16 (<i>Figur 1</i>). Tryk <i>STOP TRÆNING</i> og evaluer træningen til <i>:-D</i>, hvorefter træningen gemmes (<i>Figur 2</i>). Tryk herefter på <i>RESULTATER</i> via hovedmenuen og vælg <i>BELØNNINGER</i>.</p> <ul style="list-style-type: none"> <li>• Grafen viser, at brugeren har trænet 120 minutter, svarende til 2 timer, på en enkelt dag (<i>Figur 3</i>)</li> <li>• Brugeren har opnået én stjerne i kategorien antal træninger (<i>Figur 4</i>)</li> <li>• Brugeren har opnået tre stjerner i kategorien tid (<i>Figur 4</i>)</li> <li>• Brugeren har opnået to stjerner i kategorien afstand (<i>Figur 4</i>)</li> <li>• Brugeren har opnået én stjerne i kategorien konditionstræning (<i>Figur 4</i>)</li> </ul>
	    <p><i>Figur 1</i>      <i>Figur 2</i>      <i>Figur 3</i>      <i>Figur 4</i></p> <p>På figurerne ovenfor vises resultaterne for test af resultater. Til venstre fremgår den ønskede tid og afstand. Anden figur viser, at resultaterne fra træningen gemmes, hvoraf figuren til højre for denne viser grafisk udvikling af den ugentlige træningstid. Den fjerde figur viser belønninger, der er visualiseret som stjerner. De opnåede stjerner afspejler, at brugeren har udført det antal træninger svarende til de opnåede stjerner.</p> <p>✓ På baggrund af denne test er kravet for resultater opfyldt</p>

*Tabel 8.7: Test af resultater.*

## 8.7 Venneliste

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

- 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*

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

Test:	Venneliste
<b>Formål:</b>	Formålet er, at brugeren kan følge andre brugere og tilgå deres belønninger. Dette gøres ved at indtaste et medlemsID på en bruger, som findes i databasen og et, som ikke eksisterer, hvorefter denne bruger følges.
<b>Main flow:</b>	<ol style="list-style-type: none"> <li>Tryk <b>VENNELISTE</b> via hovedmenuen og indtast medlemsID <b>12345678</b> og tryk <b>søg</b>.             <ul style="list-style-type: none"> <li>Søgning mislykkedes. Fejlmeddeelse vises i grænsefladen for venneliste (<b>Figur 1</b>)</li> </ul> </li> <li>Tryk <b>VENNELISTE</b> via hovedmenuen og indtast medlemsID <b>11170304</b> og tryk <b>søg</b>.             <ul style="list-style-type: none"> <li>Søgning lykkedes. Grænsefladen for søgte vens belønninger vises (<b>Figur 2</b>)</li> </ul> </li> <li>Tryk <b>VENNELISTE</b> via hovedmenuen og indtast medlemsID <b>11170304</b> og tryk <b>søg</b>. Herefter trykkes <b>følg</b>.             <ul style="list-style-type: none"> <li>Følg knap forsvinder i grænsefladen for vennelisten, og fjern knap synliggøres (<b>Figur 3</b>)</li> </ul> </li> <li>Tryk <b>VENNELISTE</b> via hovedmenuen             <ul style="list-style-type: none"> <li>Bruger med medlemsID <b>11170304</b> vises i grænsefladen for vennelisten (<b>Figur 4</b>)</li> </ul> </li> </ol>

Resultat				
	<i>Figur 1</i>	<i>Figur 2</i>	<i>Figur 3</i>	<i>Figur 4</i>

Ovenstående figurer viser resultater for test af venneliste. Til venstre ses figur for første main flow, hvor det indtastede medlemsID ikke eksisterer i databasen. Figuren ved siden af viser andet main flow, hvor medlemsID'et eksisterer i databasen, og belønninger for denne bruger fremgår. Brugeren har mulighed for at tilføje ven til vennelisten, hvilket fremgår af de to figurer til højre, som opfylder tredje og fjerde main flow.

- ✓ På baggrund af dette er kravet for venneliste opfyldt

**Tabel 8.8:** Test af venneliste.

## 8.8 Redigering af adgangskode

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

- Brugere skal kunne redigere deres adgangskode

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

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

<b>Test:</b>	Redigering
<b>Formål:</b>	Formålet er, at brugeren skal have mulighed for at redigere sin adgangskode. Dette gøres ved at indtaste to forskellige adgangskoder og to ens adgangskoder. Efterfølgende logges der ind med den ændrede adgangskode.

<b>Main flow:</b>	<p>1. Tryk <i>REDIGER ADGANGSKODE</i> via hovedmenuen. Indtast <i>nyadgangskode</i> i ny adgangskode og indtast <i>adgangskode</i> i gentag adgangskode. Tryk <i>GEM ÆNDRINGER</i>.</p> <ul style="list-style-type: none"> <li>Ændring mislykkedes. Fejlmeddeelse vises i grænsefladen for redigering (<b>Figur 1</b>)</li> </ul> <p>2. Tryk <i>REDIGER ADGANGSKODE</i> via hovedmenuen. Indtast <i>nyadgangskode</i> i ny adgangskode og gentag adgangskode. Tryk <i>GEM ÆNDRINGER</i>.</p> <ul style="list-style-type: none"> <li>Ændring lykkedes. Meddelelse viser, at adgangskoden er gemt (<b>Figur 2</b>)</li> </ul> <p>3. Log ud og log ind med medlemsID 11170301 og den nye adgangskode <i>nyadgangskode</i>.</p> <ul style="list-style-type: none"> <li>Log ind lykkes og hovedmenu vises (<b>Figur 3 og 4</b>)</li> </ul>
<b>Resultat</b>	    <p><b>Figur 1</b>      <b>Figur 2</b>      <b>Figur 3</b>      <b>Figur 4</b></p> <p>Ovenstående figurer viser resultater for test af redigering. Til venstre ses resultatet for det første main flow, hvor det fremgår, at de indtastede adgangskoder ikke er identiske. Til venstre for midten er adgangskoderne identiske, hvorved disse gemmes i databasen. Test for det tredje main flow fremgår af de to figurer til højre, hvor der logges ind med den nye adgangskode og hovedmenuen vises.</p> <p>✓ På baggrund af denne test er kravet for redigering opfyldt</p>

**Tabel 8.9:** Test af redigering.

## 8.9 Log ud

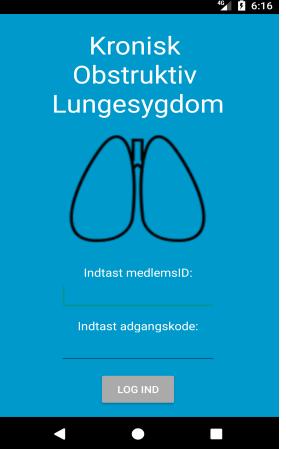
Log ud-funktionen skal sikre brugerens individuelle data. De opstillede krav til log ud er følgende:

- Brugere skal kunne logge ud af app'en

*Dette er nødvendigt for at sikre brugerens individuelle data og give brugeren mulighed for at logge ud*

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

<b>Test:</b>	Log ud
<b>Formål:</b>	Formålet er, at brugeren skal have mulighed for at logge ud af app'en.

Main flow:	1. Tryk <i>LOG UD</i> via hovedmenuen og tryk bekræft ( <i>Figur 1 og 2</i> ). • Grænseflade for log ind vises ( <i>Figur 3</i> )		
Resultat	 <p><i>Figur 1</i></p>  <p><i>Figur 2</i></p>  <p><i>Figur 3</i></p>		

Figurerne ovenfor viser resultater for test af log ud. Til venstre ses hovedmenuen, hvor brugeren har mulighed for at logge ud. I midten skal brugeren bekræfte log ud, hvis dette gøres vises grænsefladen for log ind, som ses af figuren til højre.

- ✓ Ud fra denne test er kravet for log ud opfyldt

*Tabel 8.10: Test af log ud.*

Del III

Syntese

# Kapitel 9

## Syntese

---

I syntesen diskuteses problemformuleringen i forhold til den udviklede app samt, hvorvidt der er alternative måder at designe og implementere app'en på. Diskussionen efterfølges af en konklusion, hvori problemformuleringen besvares samt en perspektivering, der tager udgangspunkt i yderligere undersøgelser, som skal foretages inden KOL-patienter ville kunne anvende app'en.

### 9.1 Diskussion

I dette kapitel diskuteses væsentlige problemstillinger forbundet med problemformuleringen. Formålet med projektet er at udvikle en app, der kan vejlede og motivere KOL-patienter til regelmæssig træning. Dette indebærer, hvad der er gjort for at besvare problemformuleringen samt mulige forbedringer i forhold til app'en. Det er dertil valgt at diskutere kravspecifikationer, design og implementering.

#### 9.1.1 Kravsspecifikationer

De opstillede funktionelle krav til app'en er testet og alle opfyldt. Det kan dog diskuteses, hvorvidt de opstillede krav skal genovervejes heriblandt, hvorvidt et fastlagt tidspunkt for notifikationen vil være hensigtsmæssigt for brugeren. Det formodes, at brugerne af app'en træner på forskellige tidspunkter i løbet af dagen, hvorved en notifikation kan virke forstyrrende for brugeren, hvis der allerede er udført en træning tidligere på dagen. En løsning på dette kunne være, at notifikationen annulleres, hvis brugeren har trænet inden klokken 15 eller, at notifikationen kommer 24 timer efter sidste udførte træning. Derudover er det en mulighed, at notifikationen tilpasses brugerens kategorisering og helbredstilstand, da de måske ikke er i stand til daglig motion, hvorfor notifikationen kan virke demotiverende og for eksempel først skal forekomme efter 48 timer fra sidste udførte træning. Et brugerdefineret tidspunkt for notifikationen er ligeledes en mulig løsning, så brugeren kan tilpasse tidspunktet ud fra, hvornår på dagen notifikationen ønskes. Yderligere kan det diskuteses, om den motiverende faktor kan optimieres ved brugen af notifikation på andre måder. For eksempel ved at give feedback i form af ros, hvis brugeren har udført en træning eller informere brugeren om, hvor mange kilometer brugeren mangler for at opnå en belønning. Dog kan for mange notifikationer have en modsatrettet effekt, hvorfor brugeren fravælger brug af app'en.

Af kravspecifikationerne fremgår det, at systemet skal bestemme brugerens kategorisering ud fra CAT-score samt antallet af årlige indlæggelser relateret til KOL. Ifølge afsnit 3.1.2 kan kategoriseringen bestemmes ud fra CAT-score eller MRC samt antal årlige indlæggelser på grund af KOL, antal årlige eksacerbationer eller ved FEV1-værdi. Det er valgt at anvende CAT-scoren, da denne baseres ud fra flere udsagn til forskel fra MRC, og det er dertil

vurderet, at denne kan give et bedre grundlag, idet udsagnene varetager flere aspekter i forhold til symptomer forbundet med KOL. Det er desuden fravalgt at anvende FEV1-værdien til kategorisering, idet det er mere ressourcekrævende, da der skal foretages spirometrimålinger. Antal årlige indlæggelser relateret til KOL er valgt fremfor antal årlige eksacerbationer, da det antages, at brugeren er bekendt med antal indlæggelser. Derimod kan et specifikt antal eksacerbationer være svært at vurdere.

Det har ikke været muligt at teste non-funktionelle krav, da app'en kun er en prototype. App'en er dog designet og implementeret ud fra gestaltlovene omfattende opsætning af grænseflader samt egenskaber, der har betydning for brugervenligheden, hvorfor det antages, at dette non-funktionelle krav er opfyldt. Dette er antaget på baggrund af, at app'ens forskellige grænseflader tager udgangspunkt i samme layout samt indeholder få knapper, der gentages løbende i app'en, som for eksempel videreknappen. Der skal dog foretages undersøgelser for at teste brugervenligheden af app'en. Der er foruden kravet om brugervenlighed opstillet et non-funktionelt krav om, at app'en skal kunne implementeres på en smartphone. Dette har ikke været muligt at teste, da databasen ligger på en lokal server, og derved ikke kan tilgås fra en smartphone. Det antages dog, at app'en senere vil kunne fungere på en reel smartphone, idet app'en er testet i Android Emulator.

### 9.1.2 Design

Projektets primære formål er ikke at udarbejde forskellige træningsprogrammer passende til KOL-patienters sværhedsgrad eller kategorisering, men at udvikle en app, hvortil et træningsprogram senere kan inkorporeres. Afgrænsningen til konditionstræning er valgt, da målinger, der vil foretages ved konditionstræning, herunder tid og afstand er et krav og derfor skal implementeres og testes. Det kan dertil diskuteres, hvorvidt andre målinger kan være mere hensigtsmæssige at implementere ved valg af styrketræning og vejrrækkningsøvelser, hvor afstand ikke anses som værende en essentiel måling. Det kunne for eksempel være mere hensigtsmæssig at implementere antallet af gentagelser.

Det er valgt at designe tilpasning af træningsniveauet ud fra en simpel beslutningstabell, hvilket ses af tabel 5.1 og tabel 5.2, som medregner parametre, såsom kategorisering, daglig helbredstilstand og tidlige evalueringer, der afhænger af tidlige helbredstilstande fra samme træningstype. Dette er gjort for at tage højde for daglige variationer samt brugerens evne til at vurdere sin daglige helbredstilstand i forhold til, hvad KOL-patienter helbredsmaessigt kan yde. De tidlige evalueringer, der afhænger af, at brugeren har valgt samme helbredstilstand og træningstype, er ikke tidsbegrænset, hvilket vil sige, at denne evaluering kan være forældet, hvis brugerens opfattelse af deres helbredstilstand ændres.

Det har ikke været muligt at finde litteratur eller teste i, hvilken grad parametrene har indflydelse på KOL-patienters helbred og fysiske egenskaber samt, hvordan den enkelte opfatter sin tilstand, hvis den ændres, og derved har en forældet evaluering. En mulig løsning til disse problemstillinger kan være at anvende bayesian læring, der lærer af indsamlet data. Dette kan forbedre tilpasningen af træningsniveau over tid på baggrund af de angivne parametre og derved yderligere tilpasse træningsniveauet til den enkelte bruger.

Et af projektets formål er at motivere brugeren til at udføre regelmæssig motion, hvilket er forsøgt opfyldt ved, at brugeren kan opnå belønnninger. I app'en fremgår det ikke, hvad der skal til for at opnå en belønning, og det kan diskuteres, hvorvidt dette skal være en mulighed for at øge motivation yderligere. En ulempe ved dette kan være, at brugeren bliver

for konkurrenceminded og derved fravælger træningstyper samt overanstrenger sig for at opnå en belønning. I app'en har brugeren ligeledes mulighed for at se en grafisk udvikling over sine ugentlig træning. I grafen vises den udførte træningstid for de enkelte dage i den foregående uge. Det kan diskuteres, hvorvidt det virker mere motiverende for brugeren, hvis der vises en graf over antal træninger eller tilbagelagt afstand. Et grafisk overblik over tilbagelagt afstand vil kun være relevant for konditionstræning, hvorfor der ved styrketræning og vejrtrækningsøvelser ville kunne implementeres oversigter over antal gentagelser. Dette vil medføre et øget antal grafer, hvilket kan virke overvældende eller uoverskueligt for en bruger.

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

### 9.1.3 Implementering

Det er valgt at implementere en database til lagring af data. Databasen er implementeret på en lokal server. En fordel ved dette er, at det simulerer en endelig database og foretrækkes at anvendes under udviklingsfasen, da eventuelle fejl er lettere at rette. Derudover begrænses en lokal server på nuværende tidspunkt, at app'en ikke kan anvendes på en smartphone. For at app'en endelig kan implementeres skal der foretages ændringer i forhold til at implementere databasen på en ekstern server.

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

På baggrund af designafsnittet blev der valgt at implementere en timer til at måle tiden under træning. Det er valgt at implementere denne som en optælling, da det ønskes, at brugere skal have mulighed for at fortsætte træningen efter den anbefalede træningstid er opnået. Det kan diskuteres, om dette er den mest hensigtsmæssige måde, idet tanken om at implementere et anbefalet træningsniveau er at sikre, at brugere ikke underpræsterer eller overpræsterer. For at undgå at brugeren ubevist træner i længere tid end, hvad der er anbefalet, er der i app'en implementeret en lyd, der afspilles, når den anbefalede træningstid er opnået. En vibration, blink eller et popup-vindue ville også kunne implementeres for at gøre brugeren opmærksom på tiden. Det er dog vurderet, at en lyd eller en vibration er den foretrukne løsning, da brugeren ikke nødvendigvis har mulighed for at se sin smartphone under træning. Dertil ville det være muligt at implementere flere af de nævnte funktioner for at øge sandsynligheden for, at brugeren bliver gjort opmærksom på den opnåede tid.

## 9.2 Konklusion

KOL er en kronisk sygdom, hvorfor det ikke er muligt at helbrede patienter. Derfor tilbydes patienter med KOL at deltage i et rehabiliteringsforløb med henblik på at lindre symptomer forbundet med sygdommen. Dette indebærer tobaksafvænning, fysisk træning, kendskab til sygdommen samt ernæringsvejledning. Studier viser, at resultaterne fra deltagelse på rehabiliteringsforløb har en positiv effekt, dog er KOL-patienterne ikke i stand til at opretholde resultaterne et halvt til et helt år efter endt rehabiliteringsforløb. Sociale fællesskaber kan have en positiv virkning på, at nogle KOL-patienter kan opretholde effekten af resultaterne hjemme. Derudover anvendes flere forskellige telerehabiliteringsteknologier, herunder app's, der har til formål at hjælpe patienter med at opretholde opnåede effekter udenfor sundhedsvæsnets faciliteter.

Det er på baggrund af dette valgt at udvikle en app til at vejlede og motivere KOL-patienter til hjemmetræning i forlængelse af rehabiliteringsforløb med henblik på at lindre symptomer forbundet med KOL.

Den udarbejdede app tager højde for daglige variationer ved at tilpasse træningsniveauet ud fra parametre, såsom kategorisering, daglig helbredstilstand samt tidligere evaluering af træning. Ud fra disse parametre anbefales et træningsniveau, som er en vejledning for brugeren. Under træningen anvendes en timer, så brugeren kan følge med i, hvornår den anbefalede træningstid er opnået, dertil afspilles en lydfil for at gøre brugere opmærksom på dette. For at motivere brugere kan brugeren se sin udvikling grafisk, samt opnå virtuelle belønninger på baggrund af udført træning. Derudover har brugere mulighed for at se andre brugeres virtuelle belønninger via en venneliste. Denne venneliste giver ligeledes mulighed for social interaktion, da brugere kan vælge at tilføje venner til vennelisten. Yderligere gøres brugeren opmærksom på træning hver dag ved en gentagende notifikation, med henblik på at informere samt motivere KOL-patienterne til at dyrke regelmæssig motion.

De centrale elementer, der indgår i app'en, herunder tilpasning af træningsniveau og motivering ved blandt andet social interaktion, er testet og på baggrund af disse tests opfyldt. Det har dog ikke været muligt at teste i, hvilket omfang app'en tilpasser træningsniveauet til den enkelte brugere samt i, hvor høj grad app'en motiverer til regelmæssig træning. Dertil har det ligeledes ikke været muligt at teste den sociale interaktions virkning på KOL-patienter. App'en giver dog muligheden for at tilpasse et træningsniveauet samt motivere ved at opnå belønninger og social interaktion.

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

Yderligere studier skal undersøges med henblik på at implementere træningsformer, -typer og anbefalede træningsniveauer, som er passende til KOL-patienter. Ligeledes skal studier udføres for at kunne bekraefte, hvilken effekt brug af app'en har på KOL-patienter i forhold til opretholdelse af resultater efter endt rehabiliteringsforløb herunder, hvorvidt dette vil lindre symptomer forbundet med KOL.

### 9.3 Perspektivering

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

På nuværende tidspunkt er serveren lokal, hvorved databasen kun kan tilgås via én computer. For at sundhedspersonalet fra flere rehabiliteringscentre i fremtiden skal kunne tilgå databasen, skal databasen implementeres som en ekstern server. Derudover skal det overvejes om sundhedspersonalet via app'en eller ved udvikling af en ny app, skal have mulighed for at registrere nye brugere i databasen samt kunne tilgå brugernes resultater via denne. Sundhedspersonalet har ikke mulighed for at se brugerens resultater visuelt, hvorfor en grafisk udvikling, lignende den brugerne kan tilgå, kan udarbejdes. Hertil vil det forventes, at sundhedspersonalet vil kunne motivere og give besked, hvis patienter har udviklet sig samt, hvis patienterne har været inaktiv i en længere periode.

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

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

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

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

# Litteratur

---

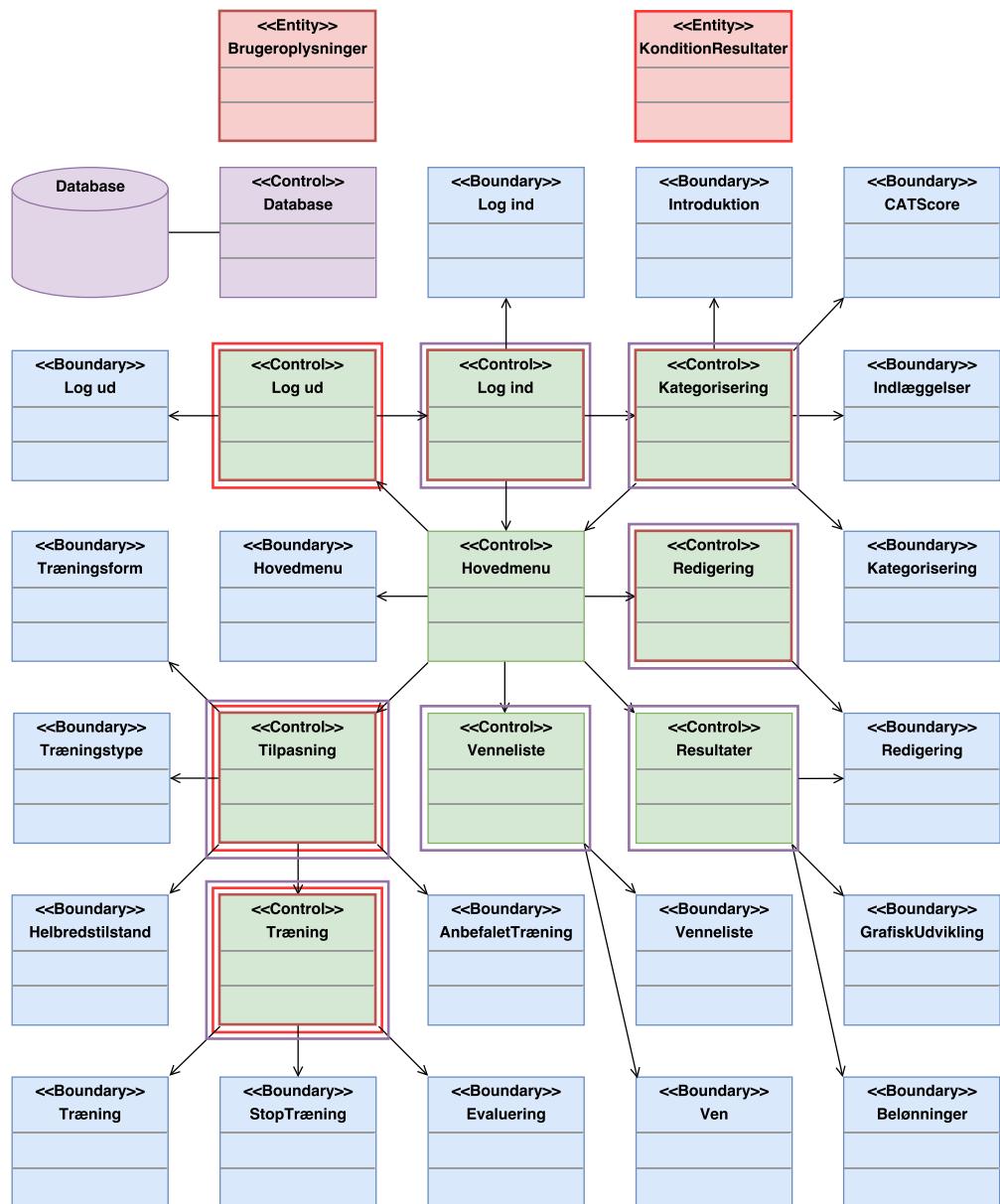
- [1] Det Sundhedsvidenskabelige Fakultet på AAU. Studieordningen for bacheloruddannelsen i Sundhedsteknologi. 2014.
- [2] Pernille Hauschildt and Jesper Ravn. *Basisbogen i Medicin og Kirurgi*. 2016.
- [3] Lungeforeningen. Lokalafdelinger og netværk. *Lungeforeningen*, 2016. URL <https://www.lunge.dk/lokalafdelinger-og-netvaerk>.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

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

- [32] Stoyan Stefanov and Kumar C. Sharma. Object-Oriented JavaScript. *Packt Publishing*, pages 32–38, 2013.
- [33] Dathan Brahma and Ramnath Sarnath. Object-Oriented Analysis, Design and Implementation. *Springer*, 2015. doi: 978-3-319-24280-4.
- [34] Martin Fowler. *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. Third edit edition, 2004. ISBN 978-0321193681.
- [35] Laurie Williams. An Introduction to the Unified Modeling Language. 2004. URL <http://agile.csc.ncsu.edu/SEMaterials/UMLOverview.pdf>.
- [36] Rational Software Corporation. Analysis Class Stereotypes. *Rational Software Corporation*, 2002.
- [37] Jim Arlow and Ila Neustadt. *UML and the Unified Process*. 2002. ISBN 0 201 77060 1.
- [38] Anders Gade. Motivation, belønning og afhængighed. *Københavns Universitet*, 2007.
- [39] Elizabeth and Tricomi and Samantha DePasque. *The Role of Feedback in Learning and Motivation*. 2016. ISBN 978-1-78635-474-7.
- [40] Sundhedsdatastyrelsen. Vejledning om informationssikkerhed i sundhedsvæsenet. *Sundhedsdatastyrelsen*, 2016.
- [41] Rådet for digital sikkerhed. Sikre adgangskoder. *Rådet for digital sikkerhed*, 2015. URL <http://www.digitalsikkerhed.dk/sikre-adgangskoder/>.
- [42] Dempsey Chang and Et.al. Gestalt Theory in Visual Screen Design – A New Look at an Old Subject. *Monash University*, 2002.
- [43] Xavier Ferré, Natalia Juristo, and Et.al. Usability Basics for Software Developers. 2001.
- [44] Andriod Studio. Android Studio. URL <https://developer.android.com/studio/intro/index.html>.
- [45] Abraham Silberschatz, Henry Korth, and S Surdanshan. *Database System Concepts*. sixth edition, 2011. ISBN 978-0-07-352332-3.
- [46] W3schools. w3schools.com. URL <https://www.w3schools.com/>.
- [47] AndroidDevelopers. LocationManager, . URL <https://developer.android.com/reference/android/location/LocationManager.html>.
- [48] AndroidDevelopers. LocationListener, . URL <https://developer.android.com/reference/android/location/LocationListener.html>.
- [49] Michel Deza and Elena Deza. *Encyclopedia of Distances*. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-00233-5. doi: 10.1007/978-3-642-00234-2\_1. URL [https://link.springer.com/chapter/10.1007/978-3-642-00234-2{\\_}1](https://link.springer.com/chapter/10.1007/978-3-642-00234-2{_}1).

# Bilag A

## Relation mellem designklasser



**Figur A.1:** Relationer mellem designklasser. Modelklasser er røde, viewklasser er blå, controllerklasser er grønne og Database samt databasecontroller er lilla. Indgår modellen Brugeroplysninger i en controller er disse markeret med en bordeaux kant. Er modellen KonditionResultater benyttet i en controller er dette markeret med en rød kant. Controllere tilknyttet til Databasen er markeret med lilla.