Roskilde frie børnehave rapport

Vi har fået tildelt den opgave at udarbejde et program til børnehaven Roskilde frie børnehave.

Dette program skal gøre livet lettere for de ansatte der dagligt arbejder inde på institutionen. Dette vil vi gøre ved at få papirarbejdet digitaliseret. Dette papirarbejde består primært af at indskrive nye børn, ændre børneoplysninger, vagtplaner mm. Vi har i vores organisation derfor udarbejdet et JAVA-program med forbindelse til en MySQL-Database, hvor dataen bliver skrevet ind.

Denne proces vil blive gennemført ved hjælp af Unified Process model (UP), hvor vi vil forsøge at følge modellen iht. dens faser og iterationer. UP består af fire faser (Inception, elaboration, construction, transition) som er med til at skabe et relativt godt projekt, da man i høj grad dykker helt ned i projektet og får et overblik over projektets kerne elementer. Modellen er primær god, da den indeholder flere iterationer hvilket betyder at man har en deadline og nogle bestemte opgaver der skal udføres inden man når til iterationens deadline.

ITO:

På nuværende tidspunkt befinder vi os i inception fasen, hvor vi begynder at undersøge de forskellige krav der eksisterer til projektet.
I faget IT-organisation har vi valgt at arbejde med feasibility study, hvilket er et relativt godt sted at starte med sit projekt, da man hurtigt kan få et overblik over om projektet kommer til at blive en succes eller ikke.
Feasibility study består af 4 dele:

Tekniske forhold
De hovedteknologier og værktøjer der bliver brugt i systemet, er følgende:

- Visual paradigm
- IntelliJ
- Java
- PowerPoint
- Word
- MySQL

Hver af teknologierne og værktøjet er til rådighed 24 timer i døgnet, og kræver ingen internetforbindelse, hvilket gør det endnu mere produktivt.

Økonomiske og finansielle forhold

- Programmet kommer til at køre på børnehavens egen server, derfor er der ikke brug for at leje en server fra f.eks. AWS eller andre tjenester umiddelbart. Men dette kan ske på et andet tidspunkt.
- Alt værktøj som bliver brugt til at udvikle systemet, er gratis, da dette projekt bliver sponsoreret af KEA.
- Vi har programmører/designere til at udvikle systemet gratis, uden nogle omkostninger.
- Ud fra de ovenstående punkter kan vi se at, projektet er velegnet til at blive gennemført ift. Økonomiske og finansielle rammer.

Organisatorisk og operationelle forhold

- Vi har en del konkurrenter.
- Projektet vil påvirke organisationen positivt, hvis projektet bliver en succes, dette vil medføre til at flere kunder henvender sig organisationen.
- Det nye system vil være et godt redskab for ledelsen og medarbejderne i børnehaven, da det vil simplificere deres arbejde i en høj grad dvs. tidsbesparende.
- Brugerne er klar til at give feedback til forbedringer.
- Brugerne vil få lettere adgang til de diverse informationer.

Retlige og juridiske forhold

- Systemet bliver udviklet med stærke sikkerhedsovervejelser
- Dermed sørger vi også for, at vi overholder persondataloven
- Alle teknologier og værktøjer er lovligt at anvende
- Medarbejderne har gode kontraktforhold og overenskomster

Planmæssige forhold

- Det er ekstremt vigtigt at tidsplanen og tidsfristerne bliver overholdt for at projektet kan gennemføres og leveres til kunden
- Vi forsøger at følge de iterationer som er udarbejdet ud fra UP modellen

Børnehavens mission og vision:

Mission

- Gennem et udviklende miljø skaber vi glade og robuste børn, der kan mestre eget liv.

Vision

- At være en innovativ og kvalitetsbevidst børnehave, der giver børn omsorg, og en professionel faglig dagligdag.

Risikoanalyse:

Derudover har vi til projektet udarbejdet en risikoanalyse for at sikre at alt går som planlagt og hvis der skulle opstå nogle problemer undervejs, så ved vi hvordan problemerne skal løses ud fra den udvidet risiko tabel som vi har udarbejdet og gjort klar til brug. Risikotabellen viser bla. de risikomomenter vi kan løbe ind i, hvordan de skal behandles og hvor stor en indflydelse det har på projektet. Gruppen kunne sagtens havde undladt at tage risikoanalysen med, da projektet ikke er så stort og derfor ikke har de store risikomomenter. Men for at være på den sikre side har vi alligevel valgt at tage den med, hvis vi nu skulle løbe ind i de små problemer. Risiko tabel kan i ses på *figur 1*, hvor vi har valgt de mest hyppig forekommende risikomomenter. På *figur 2* kan vi se den udvidet version er risikotabellen, hvor redningsplanen også forekommer.

Design:

Design delen er helt klart en af de vigtigste elementer i et software projekt, da det visualiserer og skaber en god forståelse for projektets hovedtræk. I denne her del af rapporten befinder vi os i både i den første og anden del af UP processen, nemlig inception og elaboration fasen som primær går ud på at lave blueprints for softwarekonstruktionen og at se om det overhovedet kan lad sig gør. En fordel ved at bruge UP her er at der er flere iterationer som gør det muligt at designet bliver mere optimalt og brugbart. En ulempe her kan være at det hurtigt kan gå hen og blive kompleks og uorganiseret, især hvis projektet er stort. I vores tilfælde er projektet ikke så stort, hvilket gør det nemt og overskueligt at bruge UP uden at

det bliver kompleks og uorganiseret. I tilfælde af større projekter vil vi helt klart anvende Agile modellen, da den er mere fleksibel og tilpasningsdygtig.

FURPS+:

Gruppen synes at det var oplagt lave en FURPS+ som kan ses på *figur 3*, da gruppen også gerne vil have et overblik over de non funktionelle krav ved siden af de funktionelle krav som er defineret i use casen. De non funktionelle krav fortæller os, hvordan vi gør programmet mere brugervenligt og robust. Det stiller også krav til hvor hurtig programmet skal kører og hvor lang tid der skal gå før alt data er blevet hentet. Derudover kan der være krav om hvor mange antal brugere der kan anvende programmet ad gangen. Derfor er FURPS+ en rigtig god måde at komme i gang med sit projekt på, da det viser hvilken kvalifikationer programmet skal have før man starter med at kode.

Use case model:

Gruppen har valgt at udarbejde en use case model, som kommer til at indeholde en use case diagram, use case tekst, System sequence diagram og Sequence diagram. Det vil vi komme nærmere ind på i dette afsnit.

Use case diagram er en god måde at starte et projekt med, da det giver en god forståelse for hvad systemet skal indebære og hvilken funktionalitet den skal udføre.
Vi har udarbejdet vores use cases på baggrund af de informationer og krav kunden stiller. Vi kan ud fra *figur 4* se de 11 use cases som kunden forventer at systemet skal udøve. På figuren ser vi også at der befinder sig to primære aktører som systemet skal kunne give adgang til. Manager har mulighed for at anvende alle use cases, hvorimod pædagogen har begrænset adgang. I opgavebeskrivelsen bliver der ikke beskrevet om hvorvidt pædagogen har adgang til hele system, derfor har vi valgt at tildele pædagogen de mindre funktionelle use cases, da manageren står for de større opgaver.

Vi har derudover valgt at lave en brief, casual og fully dressed, som er med til at give os en dybere forståelse af de pågældende use cases og hvordan system skal virke ud fra forskellige scenarier. Uden disse tre elementer er det svært at få et indblik i systemet, derfor synes gruppen at det helt klart skulle med i projektet.

I det følgende kan i se brief og casual tekster. Fully dressed kan ses på figur 5.

Use case - Brief:

- *Redigere vagtplan:*

Når aktør gør brug af denne funktion, så redigerer aktøren i børnehavens vagtplan, ved at indtaste de nye oplysninger for den kommende vagtplan.

- *Redigere børneoplysninger:*

Når aktør gør brug af denne funktion, så redigerer aktøren i det enkelte barns oplysninger. Oplysningerne kan bestå af barnets kontaktoplysninger, alder, navn osv. Man kan derfor rette i disse oplysninger, samt slette og oprette nye oplysninger.

- *Indskrive nye børn:*

Når aktør gør brug af denne funktion, så indskriver aktør nye børn ind til børnehaven.

- *Se børneoplysninger:*

Når aktør gør brug af denne funktion, så kan aktør se det enkelte barns oplysninger. Oplysninger kan bestå af barnets kontaktoplysninger, alder, navn osv.

- *Se liste over børn:*

Når aktør gør brug af denne funktion, så kan aktør se en liste over alle børn i institutionen.

- *Se vagtplan*:

Når aktør gør brug af denne funktion, så kan aktør se vagtplanen for den gældende arbejdsplads.

- *Slette børn:*

Når aktør gør brug af denne funktion, så kan aktør slette et barn fra nuværende liste.

- *Se telefonliste:*

Når aktør gør brug af denne funktion, så kan aktør finde det enkelte barns kontaktoplysninger.

- *Tilføj børn til venteliste*:

Når aktør gør brug af denne funktion, så indskriver man et nyt barn til børnehavens venteliste.

- *Se venteliste:*

Når aktør gør brug af denne funktion, så kan aktøren se hvilke børn der er skrevet op på venteliste.

Use case Casual:

Main succes Scenario:

Brugeren logger ind i system, hvor aktøren enten kan vælge at redigere i vagtplanen, børnenes oplysninger eller indskrive nye børn ind i databasen, samt slette børn fra børnehavens database. Aktøren kan også vælge at se vagtplanen, samt børnenes oplysninger, hvilket vil sige at hun kan finde forældrenes oplysninger frem.

Alternate Scenarios:

Hvis børnenes oplysninger bliver skrevet forkert ind eller at de bliver opdateret, så ved hjælp af systemet har brugeren mulighed for at redigere i diverse oplysninger, såsom børnenes oplysninger eller vagtplanen.

System sequence diagram:

I Dette system sequence diagram har gruppen taget udgangspunkt use casen "redigere oplysninger" som går ud på at redigerer børnenes oplysninger. Her ser vi kort hvilke metoder der bliver brugt i koden for at redigerer oplysninger, hvilket giver os en god forståelse på hvordan brugeren tilgår system og hvad der i virkeligheden sker i backend. Diagrammet kan ses på *figur 6.*

Sequence diagram:

Gruppen har også lavet en sequence diagram som er et mere dybdegående diagram end system sequence diagram, da man får lov til at se hvilken klasser de pågældende metoder bliver kaldt fra. Hvilket giver gruppen en god forståelse for hvordan koden skal bygges op. Sequence diagrammet kan ses på *figur 7*, hvor gruppen tager udgangspunkt i "slette børn" use casen.

Konceptuel model:

En konceptuel (idemæssig) model er et system som beskriver de forskellige enheder som er i systemet og beskriver deres relation til hinanden. Der mange gode grunde til at lave en konceptuel model, først og fremmest giver det et overblik over alle de mulig klasser systemet kan indeholde. Derudover kan man fastlægge alle de mulige attributter som den pågældende klasse kan have. Den konceptuelle model kan være en stor hjælp senere, når man skal i gang med at lave den rigtige software klassediagram. Den konceptuelle model kan ses på *figur 8*.

Prototype:

Vi har valgt at lave en prototype som vil visualisere hvordan system vil se ud, som også vil blive tilsendt til kunden. En fordel her kan være at man kan få feedback fra kunden om hvorvidt systemet skal forbedres eller ikke. *På figur 10* ser vi at systemet er blevet implementeret i en smartphone. Den første tegning til venstre ses en menu, hvor man kan vælge muligheden "opret barn". Den næste tegning ser vi at efter man har trykket på "opret barn", så kommer der en ny side op, hvor man skal indtaste barnet informationer. Den sidste tegning viser at efter man har udfyldt alle de mulige felter, så skal man trykke på den grønne knap for at udfører registreringen af barnet.

State machine diagram:

Et state machine diagram er et diagram der viser en status af et givent objekt, som undervejs kan ændre sig. I dette tilfælde viser diagrammet de forskellige stadier der kan forekomme, når et barn bliver tilmeldt i børnehaven. Diagrammet kan ses på *figur 9*.

Class diagram:

Det er nemlig her at gruppen identificerer alle klasser som deltager i softwareløsningen. Dette er nok den vigtigste del for overhovedet at komme i gang med at bygge et software. Uden dette blueprint bliver kodningsdelen uorganiseret, da man ikke ved hvilken klasser man skal oprette samt hvilke attributter og metoder klasserne skal indeholde. Diagrammet kan ses på figur 11.

Konstruktion:

For programmet har vi fået til opgave at lave en JAVA-prototype, der håndteres i tekstfiler. Det har vi dog set bort fra i vores program, da vi tænkte at anvende database (MySQL), vil være mere brugervenligt, og så ville vi også i gruppen udfordre os selv med at udvikle programmet med JDBC, så vi har fokus på MySQL, da det er vores primære fokus i fremtiden, så derfor har vi set bort fra den oprindelige opgave om at lave en tekstfil, og i stedet udviklet et mere brugervenligt program.

Vi har også lært at anvende Github til at dele koden op på, og har naturligvis anvendt dette igennem projektet forløb.

I Github har vi derudover også set bort fra den almindelige Github-tilgang, som man normalt bruger på IntelliJ, og fået downloadet en adapter (Github Desktop), som selvfølgelig gjorde det lettere for hvert medlem at pulle og pushe den nye version af vores kode.

Problemer med kode:

Vi har undervejs også stødt ind på nogle problemer med Github, hvor en del af gruppedeltagerne kunne se koden, samt skrive og redigere i koden, men dog ikke oprette nye klasser eller prøve programmet af. Vores løsning var dog, at et andet medlem skulle oprette et nyt repository inde på GitHub, og dele den igen. Efter flere forsøg gav dette pote, og alle medlemmer kunne selvfølgelig køre programmet.

Et andet problem vi også stødte på undervejs siden vi arbejdede med connector, det var at vi havde et problem med vores JAR-fil, som selvfølgelig var inde i vores Project-structure, men den gad ikke køre for alle gruppemedlemmerne. Løsningen på dette var, at det individuelle gruppemedlem skulle sætte sin egen JAR-fil ind under Project-structure, og det lykkedes derefter.
En alternativ løsning vi også lærte i tirsdagens undervisning (31-03-2019), var at man kunne oprette en maven fil, og hente en connector, som dependency, som gør alt det hårde arbejde for dig.

Koden:
Før vi gik i gang med koden, så skulle vi finde ud af hvad der havde størst relevans for projektets gang og skulle derudover også finde ud af hvilke use cases vi skulle udarbejde i

koden. Til koden besluttede vi os derfor at bruge de vigtigste use cases, som er *Create Child, Edit child, Delete Child, Create schedule, get Schedule samt Edit Schedule.*
Det var meget vigtigt for os, at vi i projektet at udarbejde nemlig disse Use cases, fordi det giver et overblik over et fuldkommet projekt, hvis det skulle lade sig gøre i fremtiden.

Da vi som sagt havde brugt database i stedet for tekstfiler, så har vi også oprettet en online server, så alt data er tilgængelig for alle brugere. Dette gjorde vi igennem GearHost, der tilbyder fri online servere. Serveren er som sagt brugt til at få alle brugere med til kun at oprette en specifik database, i stedet for flere forskellige database, som også vil være grund til at der vil være forskellige kode i de individuelle steder, hvor gruppemedlemmerne har oprettet deres kode som eksempelvis *url, username og password.* Serveren vil også være et abstrakt til hvordan børnehavens system senere skal se ud.
MySQL er også nemmere at tage hånd om, og danner et meget større overblik end hvad filer kan.

I det repository vi har vedlagt, kan man se hvordan vi kodet programmet, og vi har tilføjet kodens kernefunktioner som eksempelvis SQL*(Figur 12)* og den anden kerne funktion er nemlig Child klassen, hvor barnet bliver registreret og sat ind i databasen(**Figur 13**).

Github link:

I det nedenstående link kan man både finde projektets kode og design løsninger.

Link:
https://github.com/omar0190/BannehaveRoskilde

Bilag:

Figur 1: Risiko tabel

Risikoanalyse tabel			
Risikomoment	**Sandsynlighed**	**Konsekvens**	**Produkt**
Forsinkelser	**1**	**7**	**7**
Sygedage, samt fridage	**2**	**7**	**14**
Opgaverne bliver ikke fuldført af medlem.	**1**	**10**	**10**
Diskussioner i forhold til opgaven.	**3**	**2**	**6**

Figur 2: Udvidet risikotabel

Udvidet risikotabel							
Risikomoment	**Sandsynlighed**	**Konsekvens**	**Produkt**	**Præventive tiltag**	**Ansvarlig**	**Løsningsforslag**	**Ansvarlig**
Forsinkelser	**1**	**7**	**7**	**Skriv en sms i god tid, hvis du ved du bliver forsinket.**	**Gruppen**	**Stå lidt tidligere op om morgenen, hvis du ved du regulært**	**Enkelte medlemmer**

						er forsinket.	
Sygedage, samt fridage	**2**	**7**	**14**	**Informer gruppen.**	**Gruppen**	**Når man er syg, så er man syg, men ved man, at man har noget den ene dag, så informerer man gruppen i god tid.**	**Enkelte medlemmer**
Opgaverne bliver ikke fuldført af medlem.	**1**	**10**	**10**	**Hvis man har svært ved noget, så kan individet spørge de andre i gruppen om hjælp, ellers er der ingen god undskyldning. Konsekvensen er at bage kage til resten af gruppen.**	**¨** **Gruppen**	**Spørg om hjælp.**	**Gruppen**
Diskussioner i forhold til opgaven.	**3**	**2**	**6**	**Vær åben for andres idéer, og hør dem ad før man lukker døren helt**	**Gruppen**	**Demokrati**	**Gruppen**

FURPS+: Figur 3

Funktionelle krav

1. Registrerer børn
2. Rediger oplysninger
3. Slet børn
4. Se liste over børn
5. Tilføj børn til venteliste
6. Se venteliste
7. Se forældrenes oplysninger
8. Lave vagtplan
9. Se vagtplan
10. Rediger vagtplan
11. Se telefonliste

Usability:

- Hjælp til forskellige punkter
- Brugervejledning & lydfil
- Gør det brugervenligt

Reliability:

- (Try & catch)
- Lav en risikoanalyse

Performance:

- 20 brugere kan anvende ad gangen
- Kortere ventetid
- Startup Time max 1 min, så alle filer hentet.
- Hvis programmet går ned, så bliver en besked sendt til en medarbejder, så vil programmet være oppe at køre på max en time.

Supportability:

- Test af program én gang om dagen
- Opdatere programmet løbende
- Installation til Mac, Windows og Smartphones

Sub-factors

- Overholde persondataloven

Use case diagram: figur 4

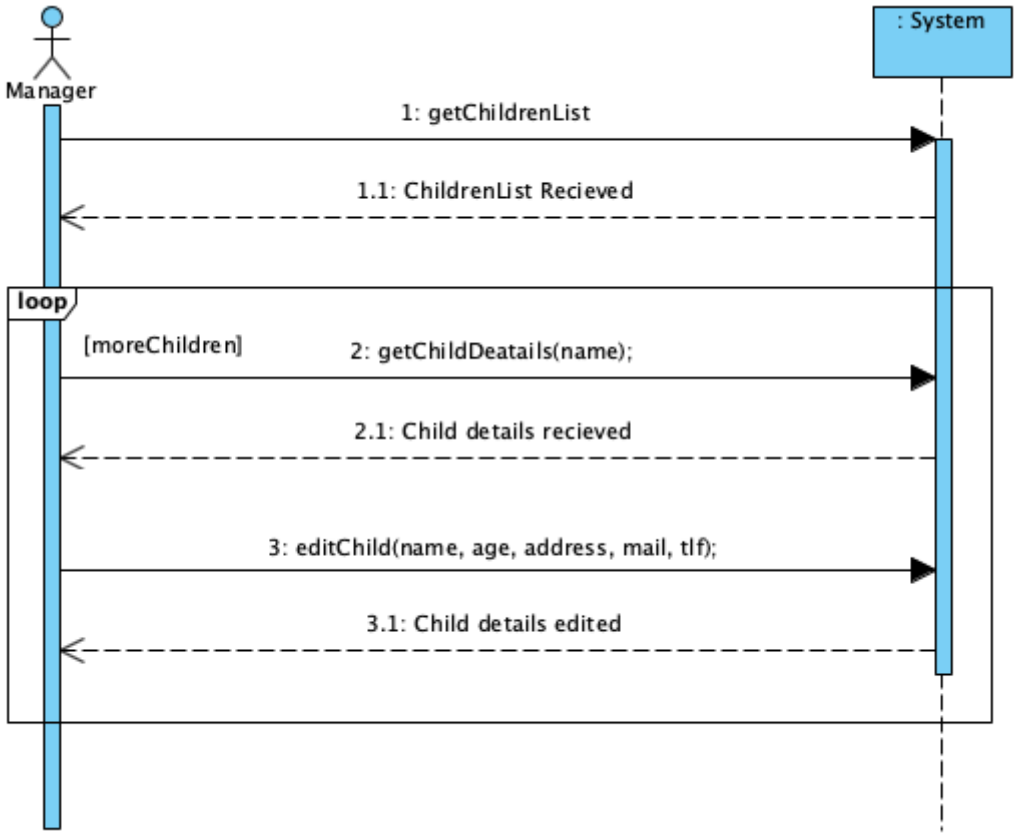
Use Case Fully Dressed: Figur 5

Use Case Section	Comment
Use Case Name	**Indskrive nyt barn**
Scope	**Roskilde børnehave program**
Level	**User goal**
Primary Actor	**Lederen, pædagog**
Stakeholders and Interests	**- Lederen: Vil gerne registrere/oprette børn. Ingen børn = ingen børnehave** **- Pædagog: Ønsker at se hvornår de skal være på arbejde, samt sikre at børnene trives, og hvis der sker noget, så kan de hurtigt komme i kontakt med forældrene**
Preconditions	**Lederen er identificeret og godkendt**
Success Guarantee	**Barnet er registreret i Systemet**
Main Success Scenario	**1. Lederen logger på Systemet.** **2. Lederen registrerer et nyt barn med de oplysninger han har fået.** **3. Systemet opretter barnet, så barnet kan begynde i børnehaven.** **4. Barnet er oprettet i børnehaven, og barnets forældre får tilsendt en velkomstmail.** **5. Barnet starter i børnehaven.**

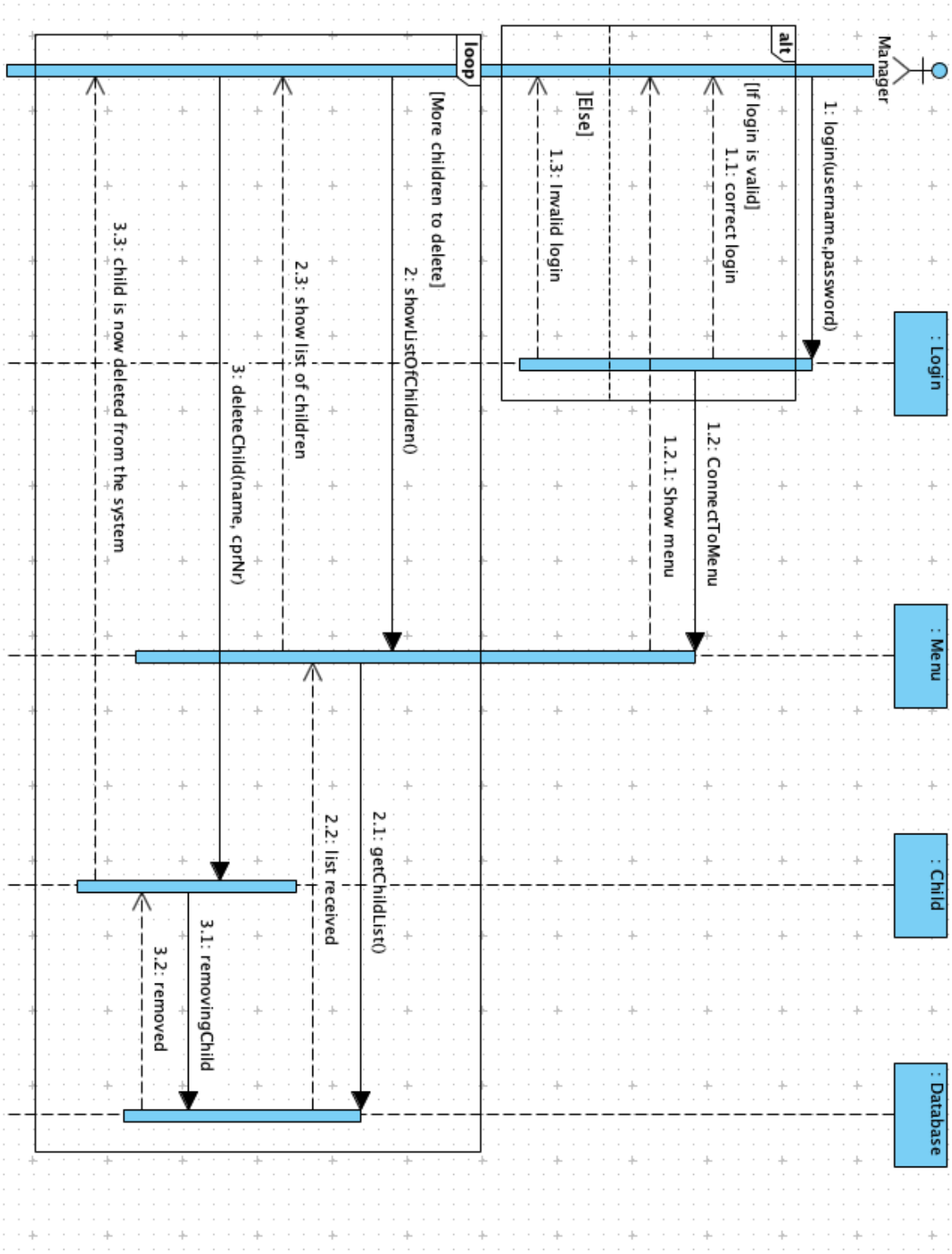
Extensions	**1a. Lederen kan ikke logge på Systemet.** **1. Lederen genstarter Systemet.** **1a. Lederen logger ind.** **1b. Lederen kan stadig ikke logge på Systemet og kontakter supporten.** **2a. Systemet kan ikke oprettet et nyt barn** **1. Lederen prøver at oprette et nyt barn i systemet.** **1a. Barnet bliver oprettet i systemet og leverer en velkomstmail** **2a. Systemet kan stadig ikke oprette et nyt barn. Lederen prøver andre metoder.** **2. Lederen genstarter Systemet.** **1a. Lederen logger på Systemet.** **1. Lederen registrerer et nyt barn.** **1a. Systemet opretter barnet og leverer en velkomstmail** **2a. Systemet kan stadig ikke oprette et nyt barn. Lederen prøver andre metoder.** **4a. Lederen kan ikke sende velkomstmail grundet forkert e-mailadresse.** **1. Lederen overskriver info med opdateret e-mailadresse og sender velkomstmailen.** **1a. Lederen kan stadig ikke sende velkomstmailen. Lederen prøver andre metoder.**
Special Requirements	**Usability: Brugervejledning, brugervenligt** **Reliability: Back-up servere** **Performance: 20 brugere, kort opstartstid** **Supportability: Test en gang om dagen, opdater løbende, Installation til Mac og Windows.**

Technology and Data Variations List	**1a. Kontooplysninger bliver indtastet via tastatur.** **2a. barnets soplysninger bliver indtastet via tastatur.**
Frequency of Occurrence	**Løbende**
Miscellaneous	

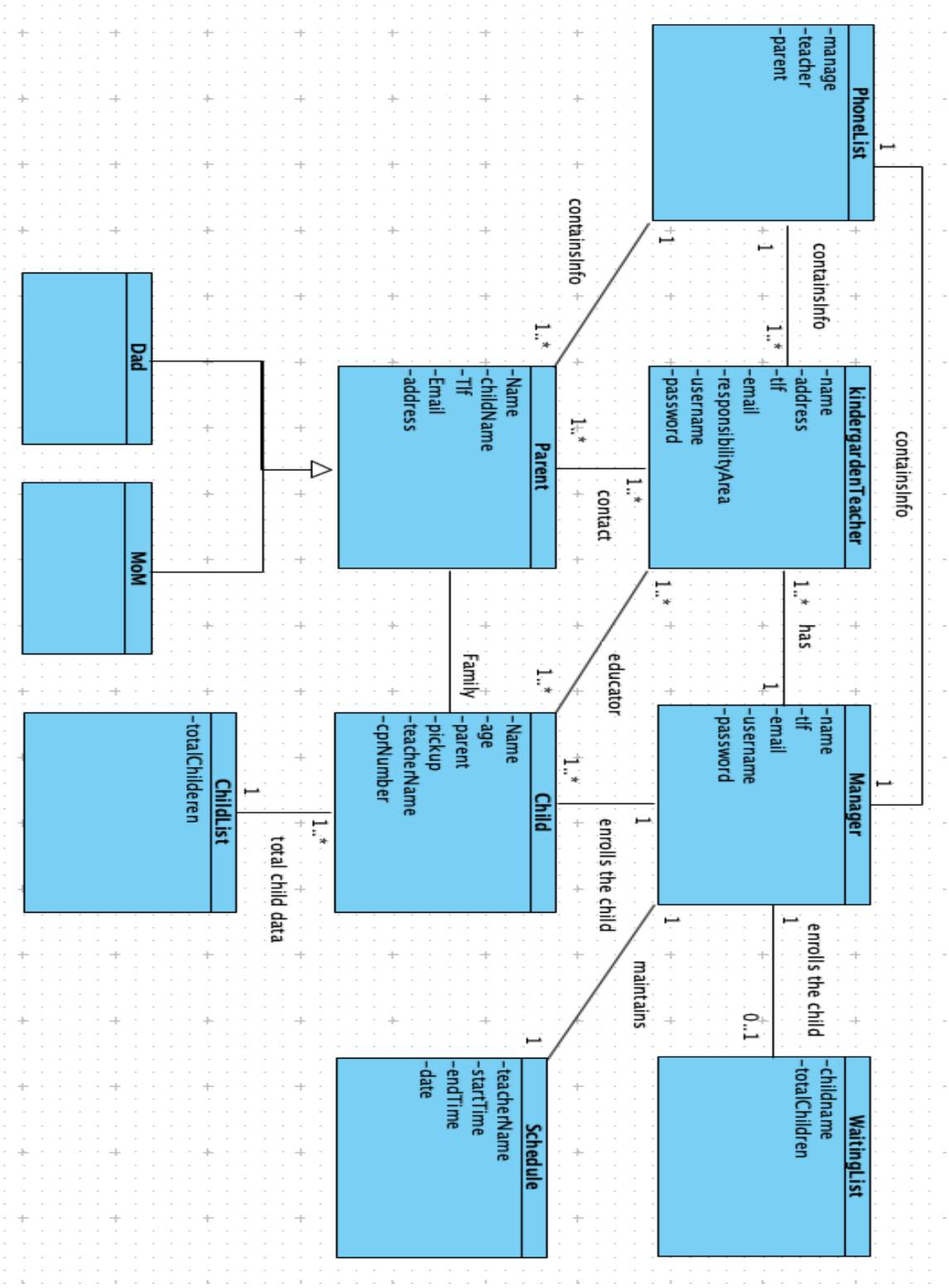
System Sequence Diagram: Figur 6



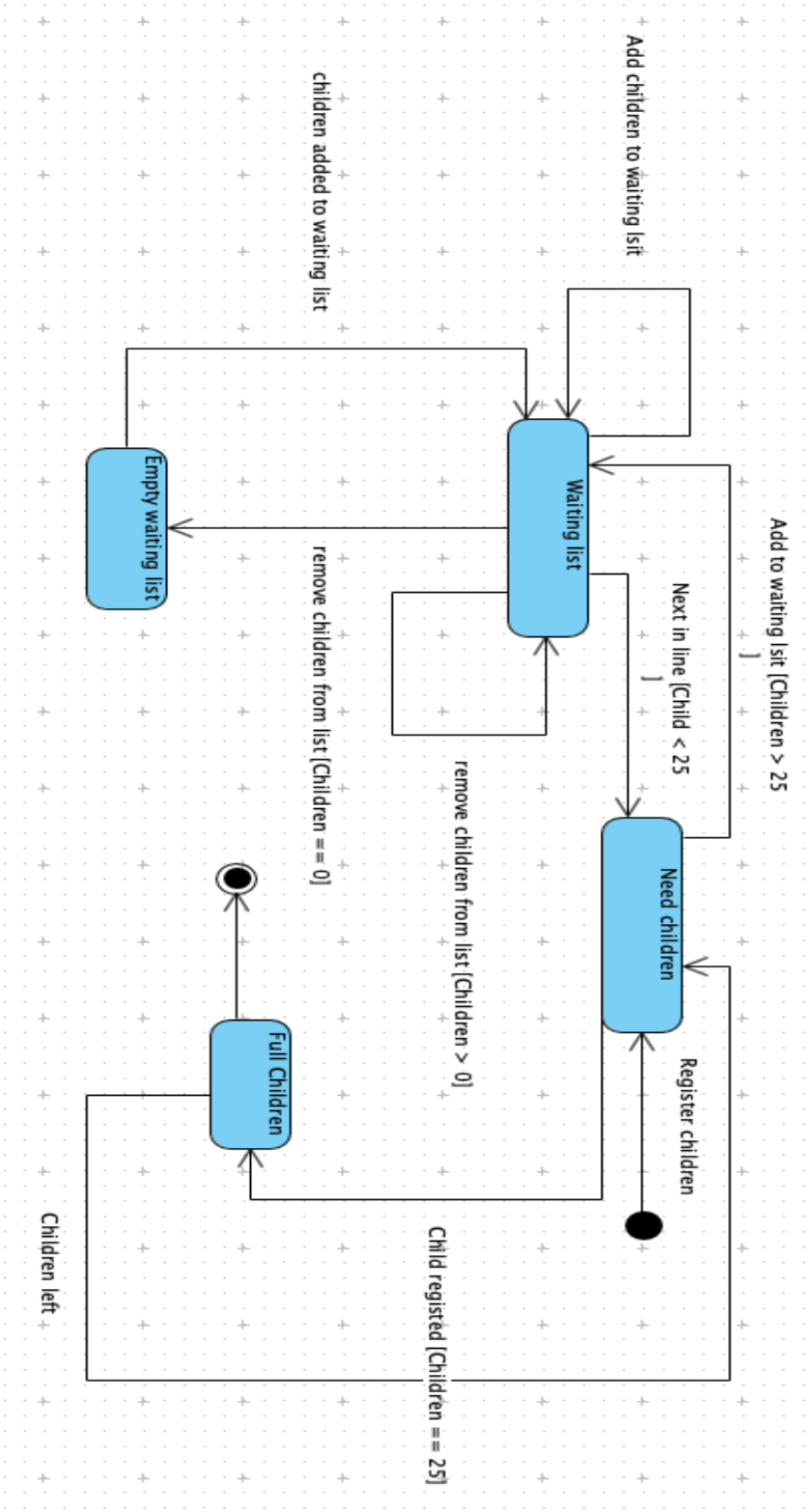
Sequence diagram: Figur 7



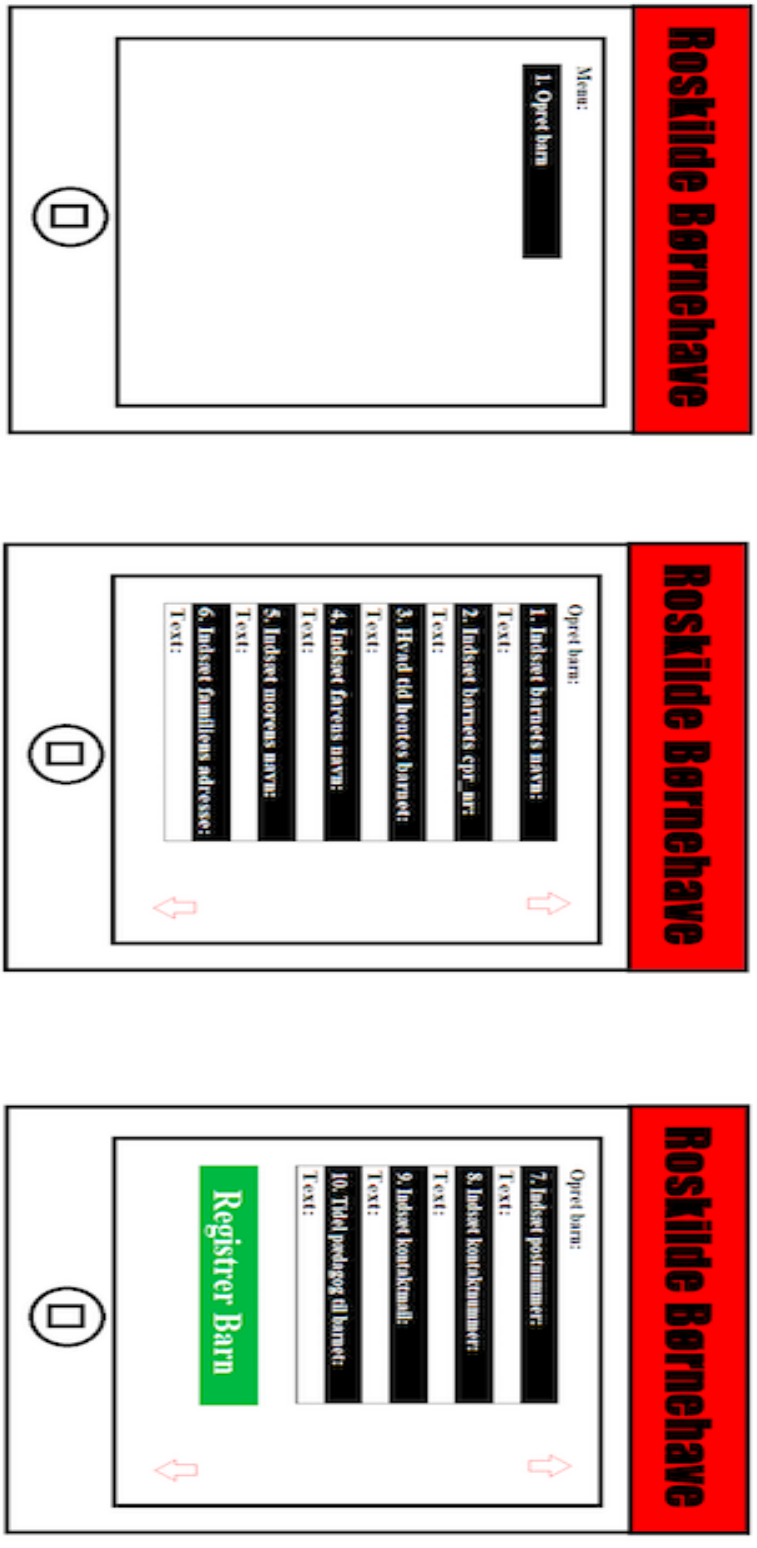
Konceptuel model: figur 8



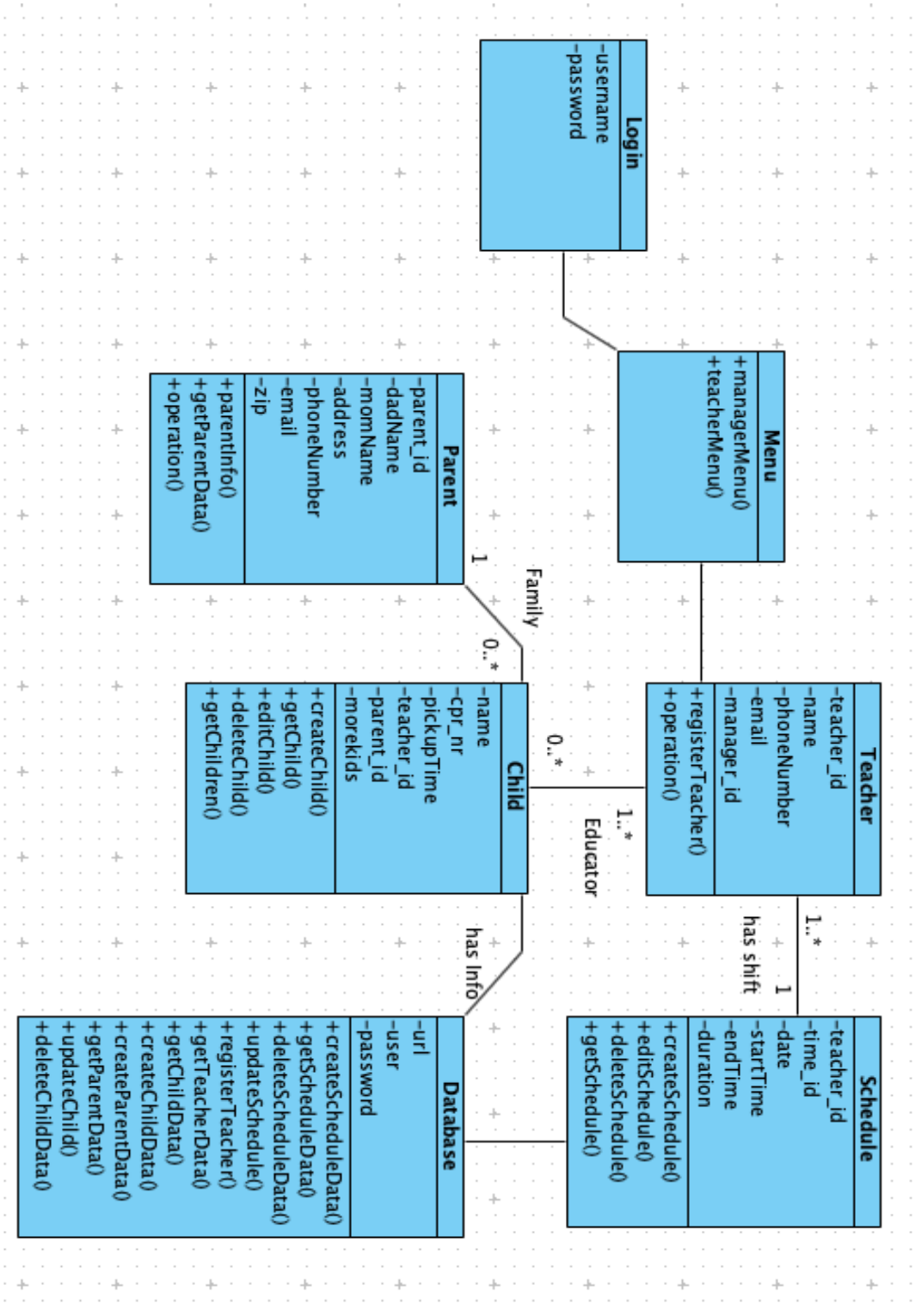
State machine diagram: Figur 9



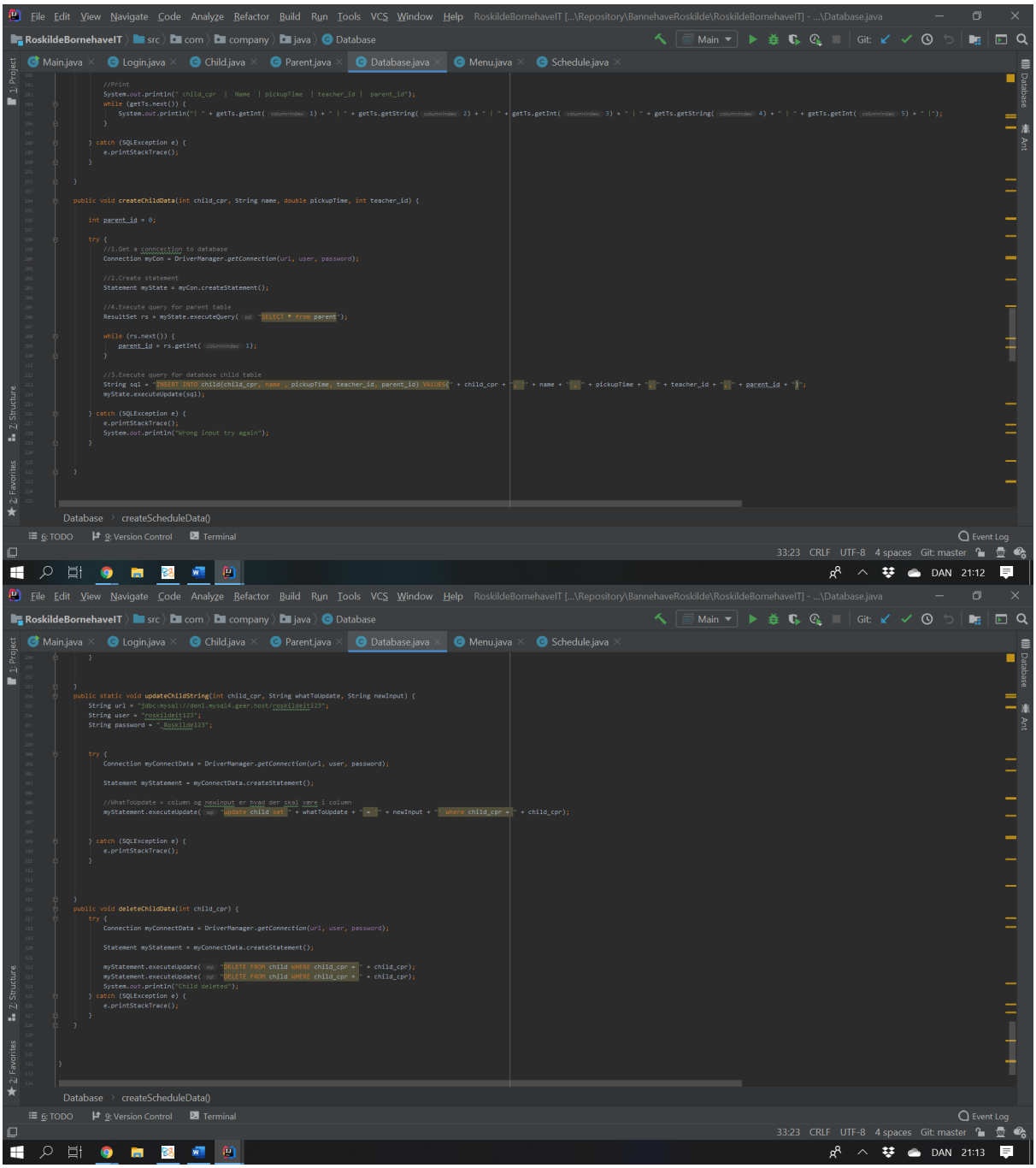
prototype: **figur 10**



Class diagram: figur 11



Figur 12: Illustrerer vores kode, der sætter data direkte ind i vores database via vores driver.



Figur 13: Her illustrerer en af vores child klasse, og metoder, der bruges til at sætte data ind i MySQL, samt tage data ud af MySQL.