Alternativ studieplatform til kursister på VUC&hf

Project Report
Group DS308e16

Aalborg University
Department of Computer Science
Selma Lagerlöfsvej 300
DK-9220 Aalborg Øst

Department of Computer Science
Selma Lagerlöfs Vej 300
DK-9220 Aalborg Ø
http://cs.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Alternativ studieplatform til kursister på VUC&hf

Theme:
Studieplatform

Project Period:
02/09/2016-21/12/2016

Project Group:
DS308e16

Participant(s):
Jacob Sloth Thomsen
Benjamin Jhaf Madsen
Lasse Fisker Olesen
Kim Larsen
Rasmus Kjettrup Thomsen
Mathias Wael Nielsen
Dennis Lolholm Sand Christensen

Supervisor(s):
Jakob Haahr Taankvist

Copies: 0

Page Numbers: 84

Date of Completion:
January 4, 2017

Abstract:

This paper details our process of creating a studyplatform based on an informants complaints about her currently used system, as well as her ideas about what functionalities and design a better system would have. This process was done through iterative cycles, where each cycle would get feedback from said informant. The first cycle was a prototype, second a developed system using C# and a combination of ASP.NET and HTML. The third did not get feedback, but was based on the critique and comments of the second iteration.

Benjamin Jhaf Madsen
<bjma15@student.aau.dk>

Kim Larsen
<klars15@student.aau.dk>

Jacob Sloth Thomsen
<jsth15@student.aau.dk>

Lasse Fisker Olesen
<lolese15@student.aau.dk>

Dennis Lolholm Sand Christensen
<dlsc14@student.aau.dk>

Mathias Wael Nedergaard Nielsen
<mwnn15@student.aau.dk>

Rasmus Thomsen
<rkth15@student.aau.dk>

Contents

1 Indledning

VUC&hf er en forkortelse for voksenuddannelsecenter, hvor der er mulighed for at tage en uddannelse efter man er fyldt 18 år. VUC&hf tilbyder heriblandt folkeskolens afgangseksamen og Højere Forberedelseseksamen. Der er 30 VUC&hf spredt over hele Danmark, hvor ni af dem er i Nordjylland. Vi har valgt at fokusere på Aalborg VUC&hf, hvor deres nuværende studieplatform er Ludus.[6]

Kursister på VUC&hf benytter primært Ludus til visning af skema, afleveringer, lektier og begivenheder. Derudover kan kursister aflevere afleveringer og lektier, samt bruge et besked system.

Platformen Ludus kan være besværligt at benytte som en ny bruger, da systemet virker uoverskueligt grundet begrænsede navigeringsmulighederne. Platformen er udviklet med et fokus på at skulle hjælpe de nuværende kursister med at holde sig opdateret på, hvad de skal igennem i studieperioden, se afsnit 2.1.

Vi har inddraget en informant, som er en kvinde på 21 år. Hun er er kursist på VUC&hf. Informanten er på sit andet år af sin Højere Forberedelseseksamen og har benyttet sig af Ludus i cirka et år. Derudover har informanten haft negative oplevelser med Ludus, heriblandt at Ludus var svært at benytte i starten, for at nævne et eksempel.

For at vi får et kerneprodukt, som kan benyttes af kursister og opfylder de krav af funktionaliteter, som en kursist har brug for, vil vi inddrage vores informant indover projektet i form af interview og usability test. Herudover vil vi også få informanten til at vælge en systemsdefinintion, som vi udarbejder. Derudover vil vi uddele spørgeskema til VUC&hf, for at få flere synsvinkler ind over projektets produkt herfra lærere og sekretærer.

Vores produkt er en studieplatform designet til at være en webapplikation efter informantens ønske. Vi har taget udgangspunkt i de mangler og opdateringer, som vores informant mente, at Ludus manglede. Da vores studieplat-

form er en webapplikation, valgte vi at skrive vores program i ASP.NET.

Projektet er udarbejdet iterativt i tre iterationer. Den første iteration endte med en observation på en udarbejdet prototype. Den anden iteration endte efter observation af det endelige program. Tredje iteration afsluttede med forbedringer af produktet baseret på feedback af vores anden iterations usability test.

2 Problemanalyse

2.1 Interview med informant

Vi har foretaget et interview med vores informant, her valgte vi at anvende et semi-struktureret interview. Dette gav os muligheden for at udforske nye områder af systemet, som vi ikke har kendskab til. Vi havde i forvejen kendskab til systemet, hvilket gav os en generel idé om relevante spørgsmål vi kunne stille. Der blev forberedt nogle spørgsmål til interviewet, som blev brugt supplerende, således at spørgsmålene kunne bruges, hvis emnet der blev omtalt omhandlede spørgsmålet. På denne måde fik vi både uventet information, men samtidig information som vi var forberedte på.

Vi startede med at få informanten til at fortælle om, hvordan hendes hverdag var, herefter spurgte vi ind til til de områder der blev nævnt som vi mente var relevante for et nyt system. Gennem interviewet kom vi frem til at informanten primært bruger skemaet, tjek af fravær, og aflevering af opgaver. Informanten bruger skemaet til at tjekke lokale og kurser samt lektier. Derudover bruger informanten en funktion til at aflevere sine opgaver, hvor hun fortalte at det ville være hjælpsomt, hvis der kom en bekræftelse på at opgaverne er afleveret. Hun fortalte, at systemet ikke skal logge af, således at den forbliver logget ind.

Informanten fortalte at hun gerne ville have et system der fungerede til computer og smartphone. Ud fra dette interview er vi kommet frem til en liste af funktioner, som informanten gerne vil have.

Liste af funktioner

1. Tydelig bekræftelse, ved aflevering af opgaver.
2. Der logges direkte ind på skemaets side.
3. Der skal kun være ét skema.
4. Programmet skal ikke logge brugeren ud.
5. Simpel navigering.

6. Påmindelse af opgaver der skal afleveres.
7. Skal være let for nye kursister.
8. Systemet skal virke til både smartphone og computer.

2.2 Spørgeskema VUC&hf Aalborg

Der er samtidig blevet tilsendt et spørgeskema til VUC&hf Aalborg. Dette anvendes til at supplere problemområdet og anvendelsesområdet. Dette er blevet sendt til både kursister, lærer og administrative stillinger, så vi kan finde ud af, hvordan lærer og sekretærer optræder i problem- og anvendelsesområdet. Derudover bruges spørgeskemaet også til at tilegne information fra andre brugere af Ludus, heriblandt lærere og andre kursister.

2.2.1 Opbygning af spørgeskema

Spørgeskemaet består af to dele. Den ene del består af at deltagerne skal angive personlige informationer. Den anden del består af hvordan deltagerne bruger Ludus samt deres forslag til forbedringer af undervisningsmetoder og rangeringer af deres IT færdigheder. Følgende er spørgskemaet samt hvordan der skal svares på det.

Hvad er du? Deltagerne skal angive om de er kursist, lærer, administrativ stilling eller andet ud fra svares mulighederne.

Hvad er din aldersgruppe? Deltagerne skal angive hvilken aldersgruppe de tilhører. Her kan deltagerne vælge mellem aldersgrupperne: 15 til 24, 25 til 34, 35+

Hvor tilfreds er du med designet i Ludus? Deltagerne skal angive, hvor tilfreds de er med designet i Ludus ud fra er fem trins skala, hvor 1 er utilfreds og 5 er tilfreds.

Hvad er dit primære brug af Ludus? Deltagerne skal beskrive, hvilke funktionaliteter de benytter mest i Ludus.

Hvor højt rangerer du dit kompetenceniveau med IT? Deltagerne skal angive, hvor højt deres IT kompetence er ud fra fem trins skala, hvor 1 er dårlig og 5 er god.

Hvilke undervisningsmetoder benytter du? Hvis deltagerne er en lærer, så skal de beskrive, hvilke undervisningsmetoder de benytter mest. Dette kan også være andre programmer uden for Ludus.

Er der nogen undervisningsmetoder som kunne forbedres med brug af IT? Hvis deltagerne enten er lærer eller kursist, så skal de beskrive, hvilken undervisningsmetoder der kan forbedres med IT. Dette kan være slides fra powerpoint som blev lagt op på Ludus efter undervisning.

Hvilke funktionaliteter i Ludus er nødvendige for dig? Deltagerne skal angive, hvilke funktionaliteter er nødvendige for dem i Ludus. Besvarelsen af dette spørgsmål foregår gennem et afkrydsningsskema, hvor svar mulighederne er: Skemaplanlægning, skemavisning, studieplan, karaktergivning, karaktervisning, kommunikation med andre på Ludus og andet. Deltagerne kan afkrydse flere muligheder i denne besvarelse.

2.2.2 Besvarelser af spørgeskemaet

Der er modtaget 18 besvarelser på spørgeskemaet, hvoraf 12 af deltagerne er kursister, fire lærere og to administrative stillinger eller andet. Hvoraf de to var en sekretær og en IT supporter. Besvarelsen på spørgeskemaet viste sig at kursister ligger i aldersgruppen 18-34, Lærer har aldersgruppen 35+ administrative stillinger har aldersgruppen 35+.

Vi har valgt at tage de mest relevante diagrammer ud fra spørgeskemaet, og beskrevet dem. Derudover kan resten af spørgeskemaet ses i bilag A.1, A.2, A.3, A.4, A.5, A.6, A.7, A.8.

På figur 2.1 fra spørgeskemaet, kan man se at størstedelen af deltagerne har en neutral holdning til designet af Ludus. Udover disse neutrale otte deltagere, ligger fire deltager på utilfreds og fem på meget utilfreds med designet, hvilket udgøre at halvdelen af deltagerne er utilfreds med designet. Derfor det vurderes at designet på Ludus ikke er tilfredsstillende for brugerne.

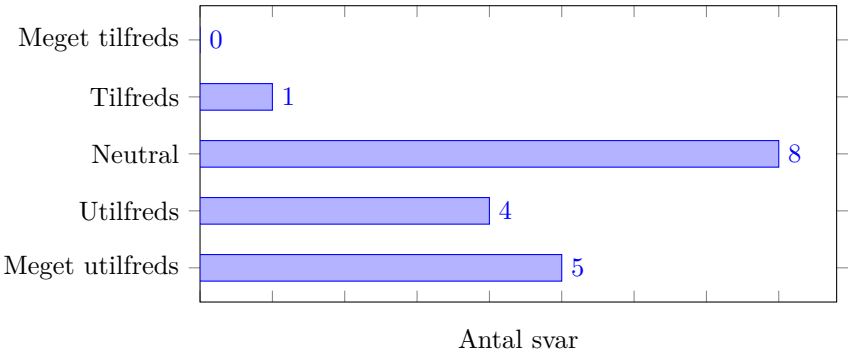


Figure 2.1: Hvor tilfreds er du med designet i Ludus?

På figur 2.2 har deltagerne angivet de funktionaliteter, som er nødvendige for dem. Alle deltager har besvaret at skemavisning er en af de nødvendige funktionaliteter, hvilket understøtter vores informants ønske om, vores system starter på med et skema på forsiden se afsnit 2.1. Samtidig er der 11 besvarelser på skemaplanlægning og 9 på studieplan. Dog vurderes det at vi undlader skemaplanlægningen i vores system, på grund af projektets omfang blev dette ikke implementeret.

Dem har svarede "Andet", har primært beskrevet i deres svar at de benytter funktionaliteter til at tjekke fravær og aflevere afleveringer, som vi derfor ser

som vigtige funktionaliteter. Selvom der er lavere prioriteter på karaktervisning, karaktergivning og kommunikation, er dette stadigvæk en central del af en studieplatform, da lærer, sekretærer og kursister skal kunne kommunikere med hinanden, hvis en er sygemeldt, eller en lektion er aflyst. Samtidig er det cirka hver fjerde person som anvender karaktervisning og karaktergivning, vi har derfor vurderet dette til at være essentielt. Vi har derfor valgt at disse også skal implementeres i systemet.

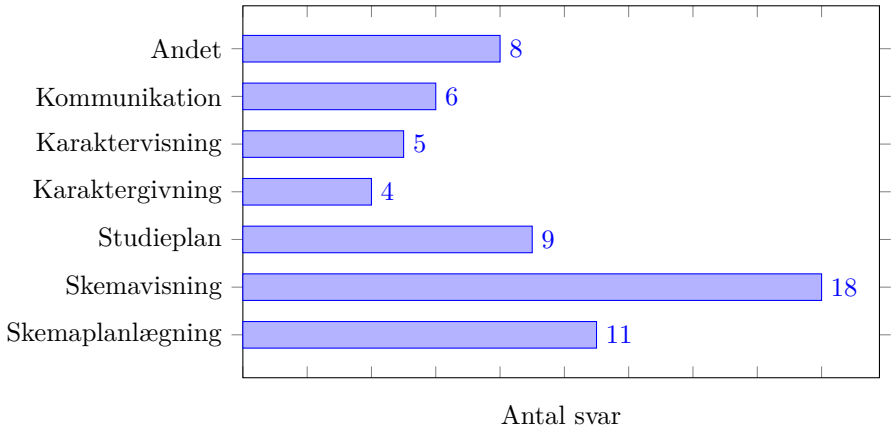


Figure 2.2: Hvilke funktionaliteter er nødvendige for dig?

Udover informantens feedback er det også vigtigt at få flere aspekter af brugerene der benytter en studieplatform. Vi valgte derfor at stille spørgsmålet *"Hvad er dit primære brug af Ludus?"*. Her viste det sig at 13 deltagere primært benytter sig af skemaet, fraværsregistrering og afleveringer. Dette betyder at disse er centrale dele for det fremtidige system.

2.3 Systemvalg

VUC&hf udbyder ungdomsuddannelser både hf og hf enkeltfag. Derudover udbydes der også ordblindeundervisning samt almen uddannelse for voksne og unge. VUC&hf er en stor organisation der består af 154 ansatte, hvor af 122 er undervisere. Derudover er der en ledelse, en administration, studievejledning, læsevejledere, socialrådgivere og konsulenter. Vi vil primært fokusere på undervisere og kursister i denne projekt. Derfor har vi indraget en kursist som informant. [6]

I afsnit 2.1, fandt vi ud af at VUC&hf benytter sig af systemet Ludus. Vi kom derudover også frem til en liste af ændringer eller nye funktioner som vores informant har brug for. I interviewet blev der forklaret, at informanten gerne vil benytte systemet mobilt. Vi er derfor kommet frem til to systemdefinitioner, som opfylder dette krav.

2.3.1 Systemdefinition: Mobil applikation

Et informations- og kommunikationssystem beregnet til smartphones. Systemet skal gøre det muligt for kursister at tjekke deres skema, fravær, afleveringer og lektier. Et besked system der gør det muligt at kommunikere med lærere eller andre kursister til eventuelt gruppearbejde. Systemet vil benytte en database til håndtering af forskellige data som bruges i applikationen heriblandt lokaler, kursus, lektioner, studiematerialer og kursister. Systemet skal kunne anvendes på både nye og ældre smartphones. Udviklingen skal foregå i tæt samarbejde med informanten.

BATOFF: Mobil applikation

Betingelser Udvikles i samarbejde med informant.

Anvendelsesområde Administrationen og lærere, planlægningsværktøjer

Teknologi Systemet vil benytte sig af en server med en database. Brugerne benytter en smartphone med et vilkårlig styresystem, skærmstørrelse og hardware.

Objekter Kursister, Lærer, Administratorer, Lokaler, Kurser, Lektioner, Materialer, Afleveringer

Funktioner Tilpasning til skærmstørrelse, tilpasning af enhedsknapper og database .

Filosofi Informationssystem(opdatering, skema, aflyste timer, hvilke afleveringer, tjekke fravær, mail system) og administrativt værktøj (Opsætning af kurser, bruger, lektioner, osv).

2.3.2 Systemdefinition: Web applikation

Et informations- og kommunikationssystem med administrative værktøjer til VUC&hf. Systemet gør det muligt at håndtere informationen omkring kurser, lektioner, afleveringer, fravær, lærer og kursister. Det skal være muligt for lærerne at formidle lektier og afleveringer til kursister gennem systemet. Systemet skal være kompatibelt med en bred række af computere, og brugerfladen skal dermed være fleksibel. Udviklingen skal foregå i tæt samarbejde med informanten.

BATOFF: Web applikation

Betingelser Udvikles i samarbejde med informant.

Anvendelsesområde Administrationen og lærere, planlægningsværktøjer.

Teknologier Systemet vil benytte sig af en server med en database. Brugerne benytter en bærbar, stationær eller smartphone.

Objekter Kursister, Lærer, Administratorer, Lokaler, Kurser, Lektioner, Materialer, Afleveringer.

Funktioner Administrativt værktøj der bruges af kursister, lærere og administrationen til at kommunikere og dele information.

Filosofi Informationssystem og administrativt system

2.3.3 Valg af definition

Ud fra systemdefinitionerne, som vi introducerende til informanten, valgte hun at kerneproduktet skulle være baseret på en web applikation, se underafsnit 2.3.1 og 2.3.2. Begrundelsen for dette var, at informanten ønskede et system som kunne bruges på både mobil og computer, eftersom hun i sin hverdag benyttede både computer og smartphone. Derudover havde hun dårlige erfaringer med Ludus' mobil applikation, se afsnit 2.1.

Ud fra informantens mening og resultaterne fra spørgeskemaet, se afsnit 2.2, vurderede vi at en web applikation var løsningen. Dette begrundes i at både sekretærer og lærer også skal benytte systemet. Dette indebærer at lærere og sekretærer skal have mulighederne for at redigere specifikke ting i systemet, hvilket primært udføres på deres arbejdscomputere og ikke smart-telefoner. Dette betyder ikke at det ikke var en mulighed, men det blev besluttet at dette var det mest optimale i forhold til nuværende arbejdsformer på disse institutioner.

3 Problemformulering

I dette afsnit er der formuleret arbejdesspørgsmål, som anvendes supplerende til både vores problemformulering og udarbejdelse af kerneprodukt.

Igennem problemanalysen er der blevet undersøgt, hvilke problemer en VUC&hf kursist oplever med Ludus. Samtidigt er der blevet undersøgt, hvilke tilføjelser som informanten har brug for. Her kom vi frem til at informanten har brug for, simpel navigering, anvendeligt for nye kursister og påmindelse af opgaver se afsnit 2.1.

Der er tidligere blevet introduceret to systemdefinitioner se afsnit 2.3, her valgte informanten web applikation. Vi har derfor valgt, at udforme problemformuleringen og disse arbejdsspørgsmål, så de tilpasser sig systemdefinitionen, men samtidig så de tilpasser det informanten ønsker. Disse arbejdsspørgsmål fokuserer primært på funktionerne og grænsefladen i det fremtidige produkt, da disse elementer var kritiske fra informantens synspunkt.

3.1 Problemformulering

Er det muligt at at udarbejde en studieplatform baseret på information og kommunikation mellem kursister og institutionsansatte, der kan give intuitivt design og bedste mulighed for kommunikative redskaber for begge parter.

3.1.1 Arbejdsspørgsmål

1. Hvordan kan der udarbejdes en ny studie platform, og hvilke komponenter skal benyttes for at opnå simplere navigering, samt anvendeligt for nye studerende?
2. Hvordan kan vi udvikle systemets funktionaliteter og grænseflade, så det passer til informantens præferencer?

4 Prototype

Baseret på vores interview, se afsnit 2.1, og vores systemvalg, se afsnit 2.3, har vi udarbejdet en prototype. Den indeholder de funktioner som vores informant bruger mest i hendes dagligdag, altså skemaet, aflevering af opgaver og tjek af fravær. Prototypen er udformet fra systemdefinitionen, og derfor er prototypen lavet som en web-applikation. Til denne prototype er der anvendt AxureRP8, til udarbejdelse af prototypen. AxureRP8 er et program der har formålet at designe prototyper, lignende powerpoint prototyper.

Der er anvendt en beroligende blå farve til udseendet af prototypen, da denne fremstår som en baggrundsfarve.[1, s. 277] Der er også anvendt røde farver, hvor brugeren har brug for at fokusere på noget, dette er valgt da den røde farve er iøjnefaldende. For at gøre systemet let anvendeligt for nye kursister, er der brugt metaforer. Dette gør at brugeren vil kunne huske og genkende de enkelte faner og knapper [1, s. 191].

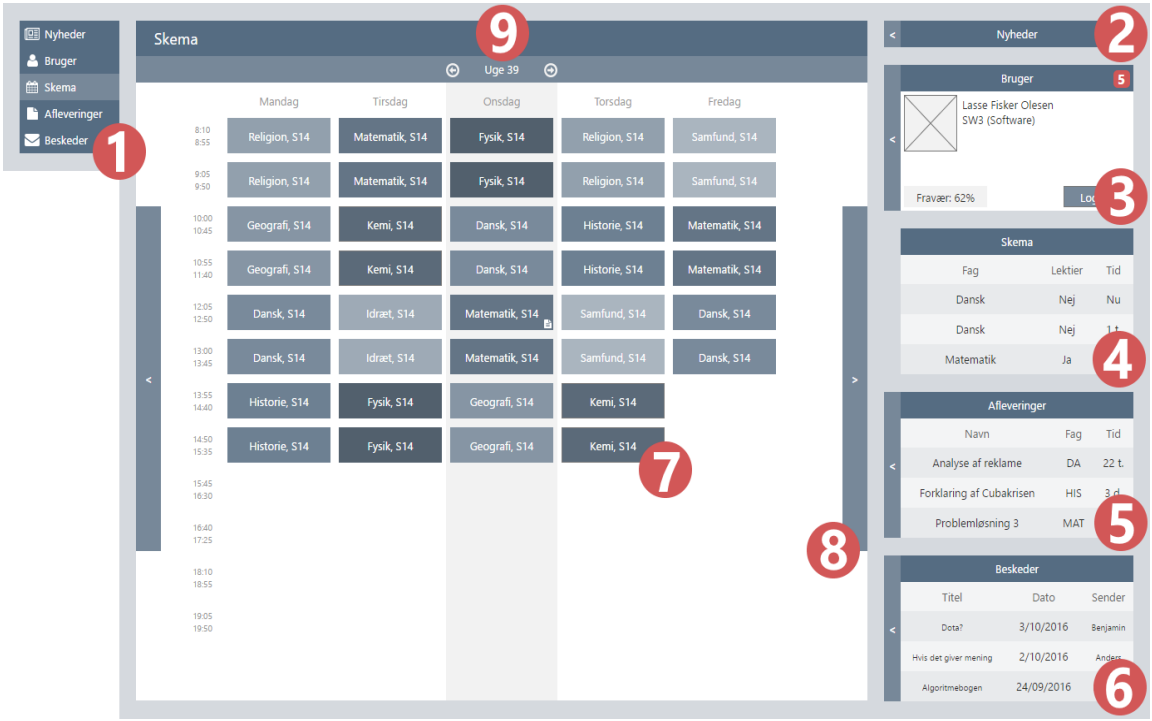
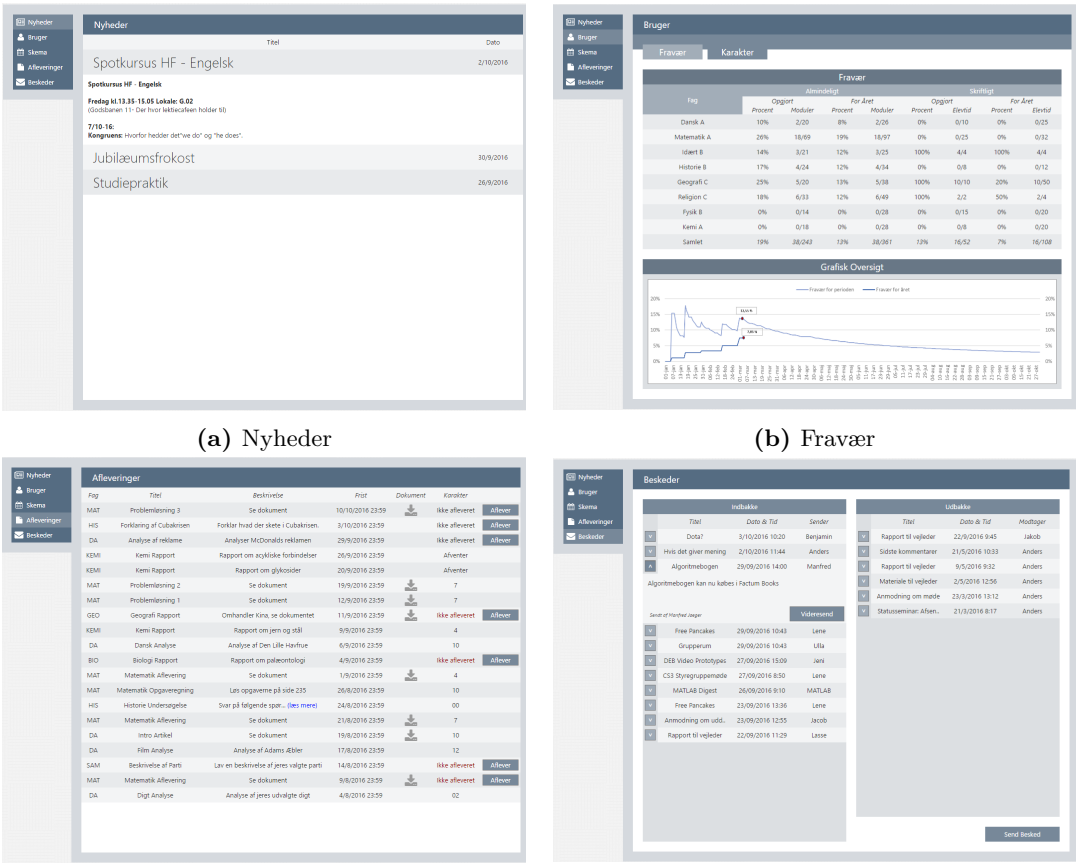


Figure 4.1: Skema i Prototypen

På figur 4.1 kan man se startsiden *Skema* på prototypen, dette var en af de funktionaliteter informanten ville have, at skema var på startsiden. Derudover er der forhåndsvisnings vinduer, som bliver forklaret senere i dette afsnit. På listen nedenfor er der gennemgået de forskellige punkter på billedet.

1. En menu til at tilgå de forskellige sider, her er der anvendt metaforer i form af ikoner, for at gøre det genkendeligt for brugeren.
2. En knap til at tilgå siden *Nyheder*.
3. Her vises informationer om hvem brugeren er, derudover kan brugeren se fraværsprocent, samtidigt med brugeren også kan logge af systemet.
4. Her vises informationer om de næste lektioner inden for tidsrummet på 3 timer.
5. Her vises de 3 næste afleveringer, som skal uploades.
6. Her vises de 3 seneste beskeder der er modtaget.
7. Skemaet.
8. Pile knapper til at skifte uge på skemaet.
9. Alternative knapper til at skifte uge på skemaet.



(a) Nyheder **(b)** Fravær

(c) Aflevering **(d)** Beskeder

Figure 4.2: Sider fra Prototypen som Informanten gennemgik under observationsundersøgelsen, se bilag for større billeder.

På figur 4.1 punkt 8 og 9, er der valgt at disse har samme funktionalitet, det er gjort så vi kan se hvilken af disse der er mest optimal at bruge, under observeringen.

Navigation i prototypen kan foregå på to forskellige metoder, med navigations baren i venstre side (Punkt 1), og forhåndsvisningsvinduerne i højreside (Punkt 2-6). Forhåndsvisningsvinduerne er lavet efter et krav fra informanten. Informanten fortalte at hun gerne ville have et element der viser de kommende afleveringer. Vi har implementeret denne idé i prototypen, og derudover er det også lavet forhåndsvisningsvinduer til kommende lektioner, afleveringer og seneste beskeder. Hvis brugeren holder musen over 5 tallet på bruger vinduet (Punkt 3), vil der blive vist en drop down menu, der fortæller brugeren om hændelser der er sket siden sidste brug af systemet. I denne drop down kan brugeren eksempelvis få at vide, at en lektion er omplanlagt, eller at der er kommet en ny besked og så videre.

Siden *Nyheder* viser de nyheder der er omkring skolen, så som fester, praktik og generelle informationer som kursisterne behøver at blive orienteret om.

Siden *Bruger* er opdelet i to faner, fravær og karakter, her kan brugeren få et overblik over sin fraværsprocent for de enkelte kurser. Fraværet er opdelt mellem almindeligt og skriftligt. Fraværsprocenten bliver angivet i "Opgjort" og "For året", hvor opgjort er de lektioner, brugeren har haft indtil videre, og for året er antal af lektioner i alt. Under fravær oversigten kan brugeren se en grafisk oversigt, som angiver brugerens fraværsprocent ud fra de forskellige måneder. På denne måde skaber det et overblik for brugeren over hvor meget fravær man har, men samtidig over hvor meget man må have.

Hvis brugeren vælger fanen *Karakter*, så kan brugeren se sin karakter for enkelte kurser, samtidig med karakterens vægt på kurserne. Derudover er karakter opdelt mellem skriftlig og mundtlig, hvor brugeren kan se årskarakter og prøver. Gennemsnit angiver det samlede snit af alle karaktererne, så brugeren har et overblik over, hvor brugeren samlede set ligger ud fra alle kurserne.

Siden *Afleveringer* indeholder en tabel med forrige og fremtidige afleveringer. Tabellen giver informationer om kurset, titlen på afleveringen, en beskrivelse, fristen, eventuelle dokumenter til afleveringen, og karakteren. Sidst er der en knap der gør det muligt at aflevere opgaven, ved tryk på knappen åbnes der en pop-up boks, hvor brugeren kan vedhæfte en fil, samt give en kommentar til afleveringen. Når brugeren har afleveret en opgave, bliver det bekræftet med en flade med teksten *"Afleveret"*, der glider op og dækker pop-up boksen, som er gjort baseret på et krav fra informanten. Der er valgt at bruge en tabel, så brugeren nemt kan sortere afleveringerne, og dermed nemmere finde frem til afleveringen der bliver ledt efter. Som standard vil tabellen sortes efter frist, så det er nemmere at lokalisere de nye afleveringer.

Siden *Beskeder* indeholder både en indbakke og en udbakke. Her kan beskederne åbnes ved at trykke på en pil i venstre side af beskeden. Derefter er der mulighed for at videresende beskeden. Der er også mulighed for at lave en ny besked, ved at trykke på send besked knappen. Hvis man trykker på knappen vil en pop-op boks åbne, her vil der være et tekstfelt, som brugeren kan skrive sin beskeden i og derefter sende den. Når beskeden er sendt, vil en lille boks poppe op med en bekræftelse på at beskeden er sendt.

4.1 Observation af Prototype

Vi valgte at lave en usability test af vores prototype, formålet med dette var at teste om systemet er sikkert at benytte, effektivt og let anvendeligt. Samtidigt med at finde ud af hvilke problematikker systemet har. Denne usability test er en bruger-baseret evaluering, hvilket er en metode til at identificere usability problemer. Ved denne metode får vi informanten til at interagere med prototypen, hvor hun skal løse nogle specifikke opgaver. Herefter fik vi en liste af problematikker der opstod efter opgaveløsningen.

Der vil blive gennemgået disse specifikke problematikker, som informanten angav under observationsundersøgelsen af projektets prototype. Observationen blev udført i et mindre lokale med to interviewere og en skribent se figur 4.4. Skribenten skulle tage notater af observationen, interviewer skulle give opgaverne til informanten og observatøren skulle sørge for at informanten ikke blev hjulpet for meget. Der blev formuleret en række opgaver, der skulle udføres af informanten uden hjælp fra interviewerne. Spørgsmålene kan ses nærmere i appendix A.9, A.10, A.11. Der blev lavet videooptagelse af observationen, og noteret en log af sessionen. Observationen varede omkring 23 minutter i alt fordelt over 7 opgaver og et konkluderende interview. Figur 4.3 viser opstillingen af observationen.

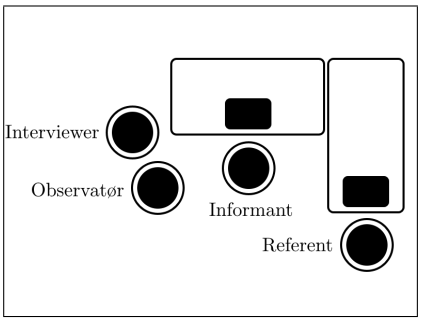


Figure 4.3: Observationsopstilling med involverede medlemmer af gruppen og informant

4.2 Problematikker

Informanten gav feedback på prototypen undervejs i observationen, dette gav os mulighed for at finde de problematikker der findes i vores prototype. Disse problematikker vurderes og derefter kategoriseret som et, kosmetisk, alvorligt eller kritisk problem.

1. Kunne ikke finde siden *Karakter*, samt forveksling af karakter med aflevering

Informanten fik til opgave at finde sit karaktergennemsnit. Informanten havde set at karakterer var blevet givet på individuelle afleveringer, så hun kiggede på siden *Aflevering* først. Herefter begyndte hun at læse karakterende op for de enkelte kurser. Efterfølgende spurgte interviewerne om man kunne se sit gennemsnit på siden. Eftersom hun ikke kunne finde sit gennemsnit på siden. Begyndte hun at navigere tilfældigt rundt i programmet indtil hun så en *Karakter* fane inde på *Bruger* siden. Da hun kom ind på *Karakter* fanen, opdagede hun hurtigt gennemsnittet.

2. Misvisende ikon-knap til lektievisning på skemaet.

Informanten fik til opgave, at hun skulle se sin lektier for matematik. Herefter kiggede hun på skemaet efter lektierne. Efter Informanten så ikon-knappen på skemaet, prøvede hun at trykke på denne knap. Efter nogle tryk på ikon-knappen fandt informanten ud af, at man skulle trykke på lektion. Herefter kunne hun aflæse de informationer hun skulle.

3. Fandt det besværligt at finde frem til siden *Fravær*.

Informanten fandt det besværligt at finde frem til information omkring brugerens fravær. Hun gik først ind på skemaet og efterfølgende navigererede hun tilfældigt og fandt ud af det var på siden *Bruger*. Herefter kunne hun se fravær for året og skriftligt.

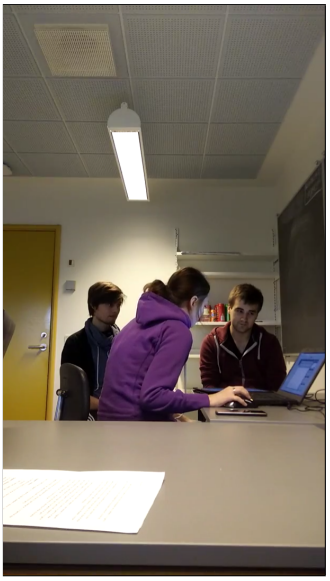


Figure 4.4: Observation på prototypen i samarbejde med informanten

4. Brugergrænsefladen var for lille og elementer var for små til at læse ordentligt.

Informanten havde ikke problemer med at læse teksten i programmet, til gengæld synes hun, at teksten var for lille, og kunne godt ønske at teksten var lidt større. Derudover synes hun også, at brugergrænsefladen var for lille i forhold til de andre systemer, som hun benyttet.

5. *Aflevering* siden er generalt rodet, samt informanten vidste ikke den var sorteret efter dato.

Informanten havde problemer med at finde den aflevering hun skulle finde. Hun ledte efter afleveringen nede fra og op, og på denne måde fandt hun den til sidst, men ikke på en systematisk måde.

4.2.1 Rangering af problematikker

De angivede problematikker er kategoriseret i følgende skema på figur 4.1 efter navigation og generelle problematikker. Derudover er de kategoriseret efter alvor på kosmetiske, seriøse eller kritiske problemer. Dette skal illustrere de kerneproblematikker der blev overvejet til udarbejdelsen af brugergrænsefladen. Da design af grænseflade bliver rekonstrueret i kapitel 8 vil disse problematikker blive løst der.

- Kosmetisk-kategorien er indikerende for en problematik der ikke gav de store besværligheder til løsningen af opgaverne. Dette løses på under et minut.

- Alvorlig-kategorien er angivende for en problematik der kunne have implikationer for løsningen af opgaverne, dog blev de i sidste ende stadig løst. Dette løses på flere minutter.
- Kritisk-kategorien er indikerende for et problem der gjorde det umuligt for informanten at løse en opgave. Informanten kan ikke løse problemet uden hjælp.

	Brugbarhedsproblem	**Kategori**
1	Navigation: Kunne ikke finde siden *Karakter*, samt forveksling af karakter fra aflevering	Kritisk
2	Navigation: Fandt det besværligt at finde frem til fravær.	Alvorlig
3	Generel: Misvisende ikon-knap til lektievisning på skemaet.	Kosmetisk
4	Navigation: Brugergrænsefladen var for lille og elementer var for små til at læse ordentligt.	Kosmetisk
5	Generel: *Aflevering* siden er generelt rodet, samt informanten vidste ikke den var sorteret efter dato.	Alvorlig

Table 4.1: Kategorisering af problematikkerne som fundet i observationsundersøgelsen med informanten på prototypen.

5 Problemområde

I dette kapitel vil vi forklare analysen af vores problemområde. Analysen er baseret på systemvalget i afsnit 2.3, problemformulering i afsnit 3, dialog med vores informant samt den feedback vi fik på vores prototype. Vi bruger denne analyse af problemområdet til at hjælpe os med at sætte krav for, hvilke funktioner vi vil inkludere i vores produkt.

5.1 Klasser og hændelser

Igennem projektet har der været interaktioner med informanten for at få indsigt i hvilke objekter og hændelser der er i modellen. I dette afsnit er der blevet undersøgt hvilke af disse objekter og hændelser der skal inkluderes i vores model. Derudover er der klassificeret objekter og hændelser, hvorefter disse er vurderet ud fra system definitionen. På tabel 5.1 vises en hændelsestabel over vores klasser og hændelser i modellen. Derudover er der lavet en beskrivelse for hver klasse.

	Lærer	Kursist	Sekretær	Kursus	Lektion	Lokale	Nyhed	Besked	Aflevering	Afleveringsbeskrivelse.
begyndt	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
aflyst				✓	✓				✓	✓
ændret				✓	✓		✓		✓	✓
forladt	✓	✓	✓							
besked sendt	✓	✓	✓					✓		
besked modtaget	✓	✓	✓					✓		
tilmeldt	✓	✓		✓						
afmeldt	✓	✓		✓						
reserveret					✓	✓				
afleveret		✓							✓	✓
udløbet				✓					✓	✓
fravær givet	✓	✓			✓					✓
karakter givet	✓			✓					✓	

Table 5.1: Hændelses- og klasse diagram på programmet.

5.1.1 Klasser

Kursist Denne klasse bliver brugt til at opbevare informationer som navn, id, kurser, afleveringer, beskeder, fravær og årgang.

Lærer Denne klasse opbevare informationer som navn, afleveringsbeskrivelser, lektioner, fravær, kurser og beskeder.

Sekretær Denne klasse har adgang til at oprette kurser, kursister, lærer og nyheder. Derudover opbevare den informationer om navn, id og beskeder.

Kursus Indeholder *lektioner*, en administerende lærer (*Personale*) og et vilkårligt antal *kursister*.

Lektion Klassen har informationer som, *Lokale*, *Kursus*, lektier, dato og eventuelle noter til lektionen.

Lokale Har et identifikations nummer og reservationer.

Nyhed Har en brødtekst, titel, dato og forfatter. Forfatteren af en *Nyhed* kan kun være *Personale*.

Besked Indeholder selve beskeden, samt informationer omkring dato, afsender, modtager og eventuelle vedhæftninger.

Aflevering Indeholder information om et vedhæftet dokument, en kommentar til dokumentet, og karakteren der blev givet.

Afleveringsbeskrivelse Indeholder information om *Kursus*, fristen, og en beskrivelse af opgaven. Der kunne muligvis være et dokument der indeholder denne beskrivelse. Ligesom en lektion, kan denne aflyses.

5.1.2 Evaluering af klasser

Klasserne er blevet evalueret ved brug af en metode af Stage [3, s. 61]. Metoden består af at man skal kunne besvare fire spørgsmål, disse spørgsmål er beskrevet her.

- Kan man identificere objekter fra klassen?
- Indeholder klassen unik information?
- Omfanger klassen adskillige objekter?
- Har klassen en tilpassende mængde af hændelser?

Disse spørgsmål er stillet til alle vores klasser, hvoraf der er blevet rekonstrueret nye klasser og slettet andre. Derudover er der også evalueret om klasserne kan tilpasse sig til system definitionen, på denne måde sikres det sig at modellen bliver så korrekt som muligt.

5.2 Struktur

For at beskrive relationerne mellem klasserne fundet i afsnit 5.1.1, er der lavet et klassediagram som kan ses på figur 5.1. Formålet med klassediagrammet er at give et overblik over problemområdet.

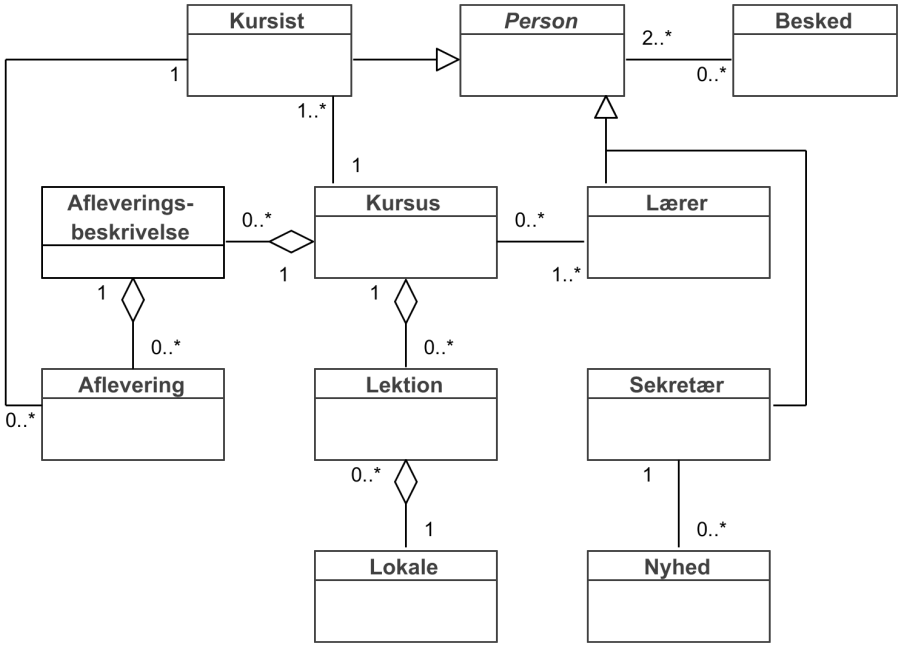


Figure 5.1: Klassediagram for problemområdet

I midten af klassediagrammet ses klasserne *Kursus*, *Lektion* og *Lokale*. Disse tre klasser er forbundet med aggregeringer, da et *Kursus* består af *Lektioner*, og ligeledes består en *Lektion* af en eller flere *Lokaler*. Klassen *Afleveringsbeskrivelse* aggregere også til *Kursus* klassen, fordi *Lærer* klassen danner *Afleveringsbeskrivelser* gennem et *Kursus*. *Aflevering* klassen aggregere dernæst til *Afleveringsbeskrivelse* klassen, i det at en *Kursist* afleverer der til.

Derudover er *Kursist*, *Lærer* og *Sekretær* klasserne generaliseret til den abstrakte klasse *Person*. Alle klasserne der nedarver fra *Person* klassen kan sende beskeder til hinanden, og *Person* klassen er derfor associeret til *Besked* klassen. Klassen *Nyhed* er kun forbundet til *Sekretær* klassen, da det kun er *Sekretær* klassen der kan oprette nyheder. Medlemmer af *Kursist* og *Lærer* klasserne kan godt se nyhederne men det berettigede ikke en association.

5.3 Adfærd

I dette afsnit er der en beskrivelse af adfærdsmønstrene for klasserne i problemområdet. For at finde ud af hvordan klasserne og hændelserne samarbejder, er der lavet tilstandsdiagrammer for klasserne.

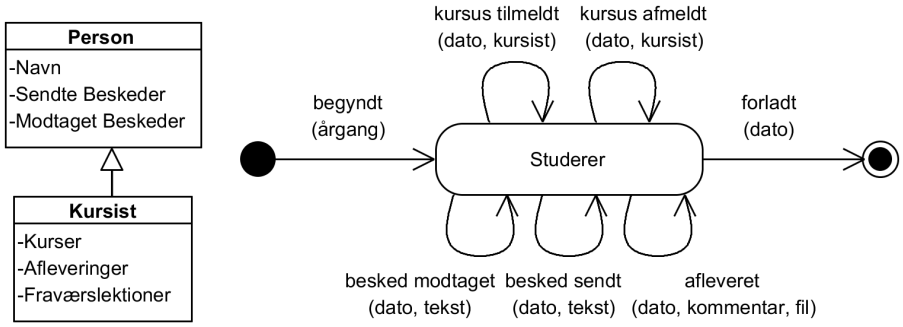


Figure 5.2: Tilstandsdiagram for Kursist klassen

På figur 5.2 kan det ses at *Kursist* klassen arver fra *Person* klassen. Derudover er der tilføjet attributter til de to klasser, der blev udledt gennem analysen af adfærdsmønstrene. Tilstandsdiagrammet illustrerer hvordan *Kursist* klassen samarbejder med hændelserne.

En *Kursist* begynder, herefter kommer man til tilstanden *Studerer*. Under denne tilstand er der forskellige hændelser der kan forekomme. Heriblandt fem iterationer, hvor kursisten kan tilmelde og afmelde sig kurser, modtage og sende beskeder og aflevere afleveringer. Derudover kan kursisten forlade stedet, hvilket betyder at kursisten enten er dimitteret, eller er droppet ud, og derfor terminere den.

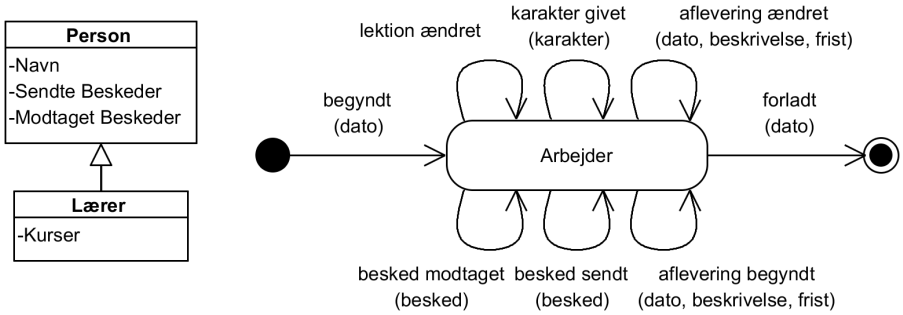


Figure 5.3: Tilstandsdiagram for Lærer klassen

Figur 5.3 viser et tilstandsdiagram for *Lærer* klassen. Ligesom Kursist klassen, har læren kun én tilstand, hvilket er tilstanden Arbejder. Den primære tilstand er når læreren Arbejder. Læreren kan modtage og sende beskeder ligesom kursisterne og de kan også forlade deres stilling. Forskellen er at en lærer kan ændre lektioner, give karakterer på afleveringer som de selv har oprettet og har mulighed for at ændre disse afleveringer.

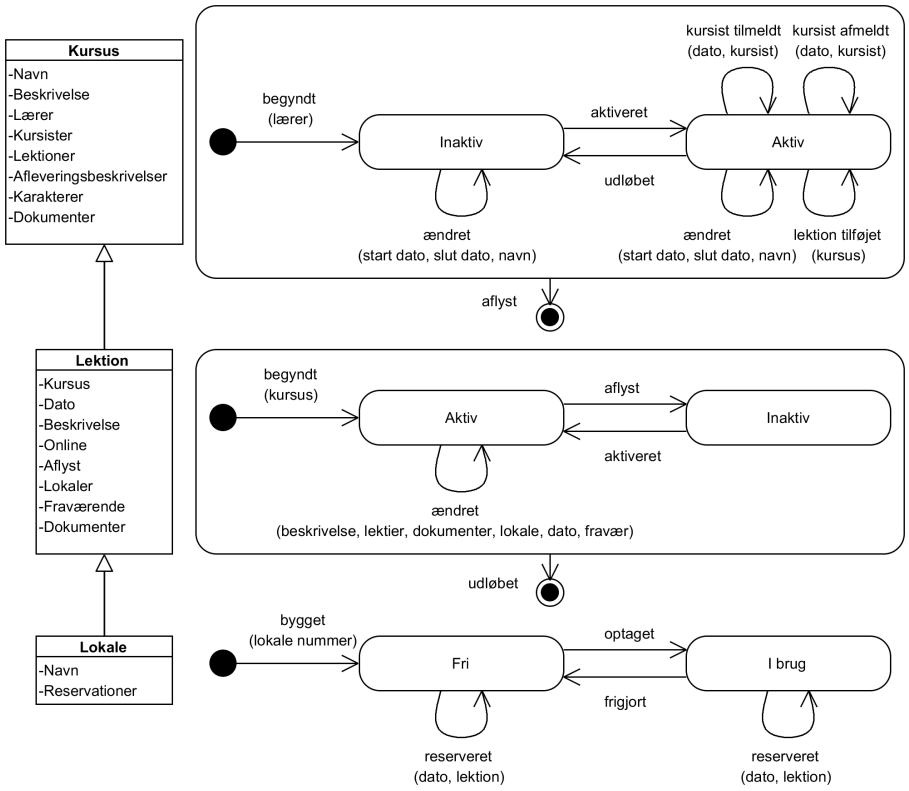


Figure 5.4: Tilstandsdiagrammer for Kursus, Lektion og Lokale klasserne

På figur 5.4 vises der tre tilstandsdiagrammer for *Kursus*, *Lektion* og *Lokale*. Samtidig vises aggregeringen mellem de tre klasser, hvor man kan se at *Lektion* aggregere fra *Kursus* og *Lokale* aggregere fra *Lektion*. I Tilstandsdiagrammet for *Kursus* kan man se at et kursus begynder og derefter får tilstanden inaktivt. Dette er gjort, da man ikke vil aktivere et kursus lige så snart det er oprettet, der skal først ændres på kurset og tilføjes lære osv. Herefter kan kurset blive aktiveret, hvor det får tilstanden aktiv. I denne tilstand kan kursister bliver tilmeldt eller afmeldt, samt lektioner kan tilføjes. Derudover kan kurset stadigvæk ændres i den aktive tilstand.

Tilstandsdiagrammet for *Lektion* starter med at en lektion begynder og derefter får tilstanden aktiv. Herefter kan lektionen enten blive ændret eller aflyst. Hvis lektionen bliver aflyst, så vil den få tilstanden inaktiv, dvs. at man kan se lektionen, men ikke som en aktiv lektion. I tilstanden aflys, så vil lektionen kun kunne aktiveres. Lektionen skal kunne aktiveres igen, hvis det har været en fejl at aflyse den.

Tilstandsdiagrammet for *Lokale* starter med at lokalet er bygget i systemet og herefter får tilstanden fri. Her kan lokalet enten blive reserveret eller optaget, hvis lokalet bliver optaget, så vil lokalet få tilstanden i brug. I denne tilstand kan lokalet stadigvæk reserveres, da man heletiden skal kunne reservere et lokale

uanset tidspunkt.

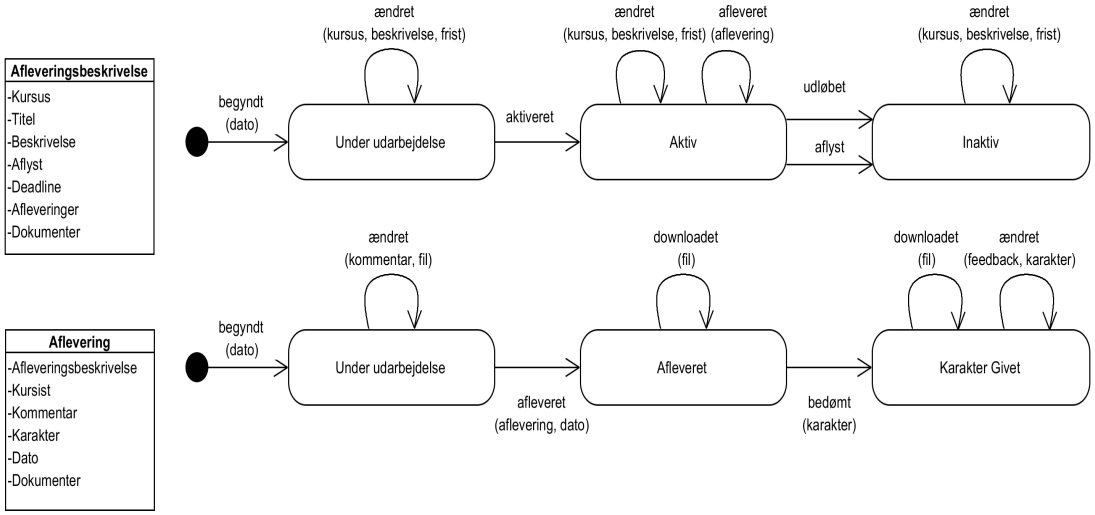


Figure 5.5: Tilstandsdiagram for Aflevering klassen

I figur 5.5 vises tilstandsdiagrammet for *Aflevering* og *Afleveringbeskrivelse* klasserne. Afleveringsbeskrivelsen begynder og opnår herefter tilstanden under udarbejdelse. I denne tilstand kan Afleveringsbeskrivelsen ændres eller aktiveres. Hvis Afleveringsbeskrivelsen aktiveres får den en ny tilstand som er aktiv. I denne tilstand kan den enten ændres, afleveres, udløbe eller aflyses. Afleveringsbeskrivelsen får tilstanden inaktiv hvis den udløber eller aflyses. Den inaktive tilstand lukker muligheden for at en studerende kan aflevere yderligere filer til afleveringen.

En aflevering skal først begyndes af en *kursist* med en specificering om hvilket kursus afleveringen hører til. Afleveringen får så tilstanden under udarbejdelse, hvor den kan ændres. Derudover kan den indeholde en kommentar og en fil. Herefter kan den afleveres og opnår så tilstanden afleveret. I denne tilstand kan filen downloades og derefter bedømmes. Når den bliver bedømt vil afleveringen få tilstanden karakter givet, hvor karakteren kan ændres.

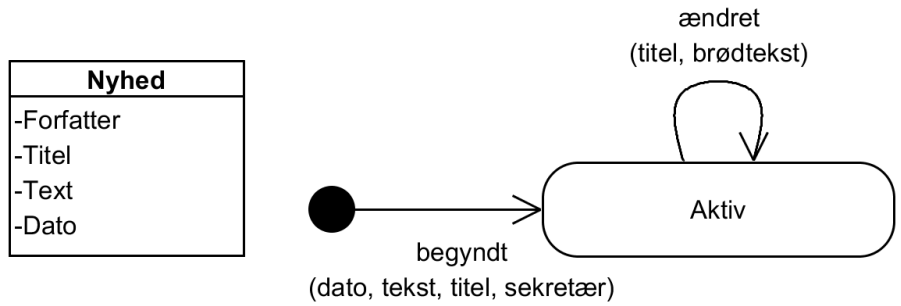


Figure 5.6: Tilstandsdiagram for Nyhed klassen

På figur 5.6 vises tilstandsdiagrammet for *Nyhed* klassen. Det starter med at nyheden begynder hvorefter *Nyhed* får tilstanden aktiv. Her kan nyheden så ændres, som vist på diagrammet er det titel og brødtekst som kan ændres.

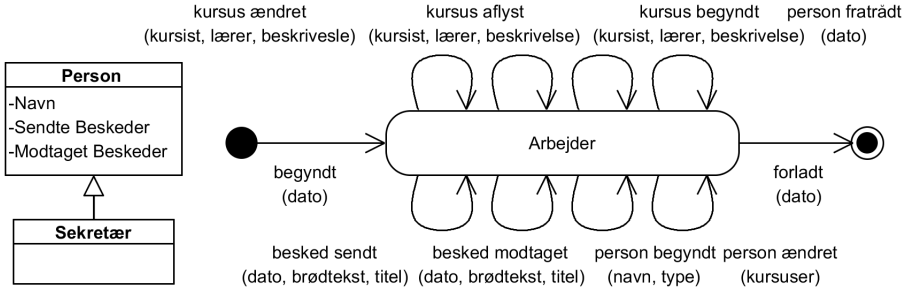


Figure 5.7: Tilstandsdiagram for SekretærKlassen

På figur 5.7 vises tilstandsdiagrammet for *Sekretær* klassen. Her begynder en sekretær og opnår tilstanden arbejder. Under denne tilstand kan sekretæren både, ændre kurser, aflyse kurser og begynde nye kurser. Lærer og kursister kan tilføjes til kurserne, samt kursus beskrivelser. Derudover under tilstanden arbejder, kan der også sendes og modtage beskeder samt begynde nye personer og ændre allerede eksisterende personer.

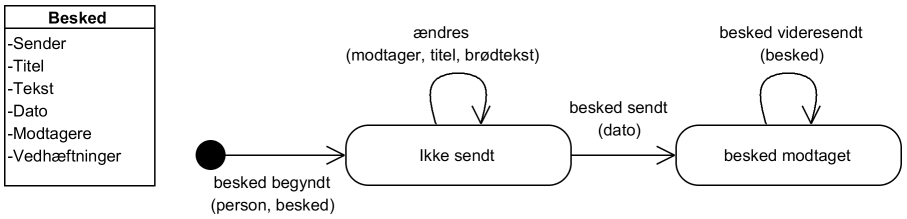


Figure 5.8: Tilstandsdiagram for Besked klassen

På figur 5.8 vises tilstandsdiagrammet for *Besked* klassen. Her er beske-

den begyndt og opnår herefter tilstanden ikke sendt. Under denne tilstand kan beskeden ændres, hvilket er titel, modtager og brødtekst. Herefter kan beskeden sendes og opnår tilstanden besked modtaget i denne tilstand kan beskeden videresendes, og returnere derfor tilbage til samme tilstand.

6 Anvendelsesområde

I sidste afsnit blev der gennemgået tilstandsdiagrammerne for klasserne. I dette afsnit vil der fokuseres på hvordan systemet vil benyttes af aktørerne. Formålet med kapitlet er at definere systemets krav samt systemets brugergrænseflade. Der vil defineres betingelser for systemets funktioner og grænseflade. Samtidig vil der være beskrivelser af aktører, brugsmønstre og funktioner.

6.1 Brug af system

I dette afsnit undersøger vi de aktører og brugsmønstre som er i vores system. Dette gøres for at forbedre forståelsen af hvordan systemet skal bruges, samt hvem der bruger det.

6.1.1 Aktør tabel

Aktørtabellen på figur 6.1 er lavet for at skabe oversigt over, hvilke interaktioner vores system har med de forskellige aktører. Samtidig er der illustreret relationerne mellem aktørerne og brugsmønstrene. I denne del af anvendelsesområdet er analysen baseret på det tidligere undersøgte spørgeskema samt problemområdet.

	Aktører		
Brugsmønstre	Kursist	Lærer	Sekretær
upload aflevering	✓		
feedback		✓	
registrering af fravær		✓	
besked	✓	✓	✓
oprette nyhedsopslag		✓	✓
oprette aflevering		✓	
oprette lektioner		✓	✓
oprette kursus			✓
rediger skema		✓	✓
tjek fravær	✓		
tjek skema	✓	✓	✓
tjek nyheder	✓	✓	✓

Table 6.1: Aktørtabel

6.1.2 Aktør specifikationer

På figur 6.1, 6.3 og 6.2 er der en aktør specifikationer. Disse består af tre dele formål, karakteristik og eksempler. Formålet beskriver relationen mellem aktøren og systemet. Karakteristikken beskriver de vigtige aspekter af aktørene som bruger systemet. Derudover har vi valgt at supplere specifikationerne med eksempler på hvordan aktørene benytter sig af systemet.

Kursist

Formål En person, som studerer på uddannelsesstedet. Kursisten skal kunne se skema, afleveringer, lektier og lignende informationer vedrørende sit uddannelse.

KarakteristikIfølge spørgeskema afsnit 2.2 ligger kursisterne i aldersgruppen 18-34. Deres gennemsnitlige kompetenceniveau fra en skala fra et til fem ligger på 3,916. Dette betyder at deres erfaring med IT ligger lidt over middel, men samtidig varierer deres niveau.

Eksempler Kursist A er okay med IT og benytter systemet til at tjekke hans skema, aflevere afleveringer og føre fravær. Kursist B er god med IT men benytter de samme funktionaliteter som A men udover det benytter B også systemet til at læse diverse nyheder.

Figure 6.1: Aktør specifikation for kursister

Lærer

Formål En person, som underviser på uddannelsesstedet. Kan give feedback på afleveringer, redigere i skema, registrere fraværende og oprette nyhedsopslag i systemet.

Karakteristik Ifølge spørgeskema afsnit 2.2 Er lærer på VUC&hf 35 år gamle. Deres erfaring med IT kan variere, dog viser det gennemsnitlige kompetenceniveau 4,5 ud fra en skala fra et til fem.

Eksempler Lærer A er erfaren med IT og bruger næsten alle funktionaliteterne i systemet (skema, fraværsregistrering, afleveringer, beskeder, fraværsoversigt, kursisters kontakinformation og lignende). Lærer B er lidt mindre erfaren end A og bruger derfor kun skema, fraværsregistrering, lektier og beskeder, altså de mest relevant funktionaliteter.

Figure 6.2: Aktør specifikation for lærer

Sekretær

Formål En person, som arbejder på uddannelsesstedet som sekretær. Sekretæren følger de samme brugsmønstre som en studievejleder, derudover kan de oprette kurser i starten af året og redigere i skemaer.

Karakteristik Ifølge spørgeskema afsnit 2.2 Er sekretærer på VUC&hf 35 år gamle. Samtidig er deres gennemsnitlig kompetenceniveau på 4,5.

Eksempler Sekretær A er erfaren med IT og bruger systemet til eksamensplanlægning, skema samt kommunikation med andre brugere. Sekretær B er også erfaren med IT men har en anden rolle. Han benytter alle funktionaliteterne i systemet jævnligt da han giver support til andre brugere af systemet.

Figure 6.3: Aktør specifikation for Sekretær

6.2 Brugsmønstre

Vi har nu skabt et overblik over hvilke aktører og brugsmønstre vi har. I dette afsnit vil de vigtigste og hyppigste brugsmønstre blive beskrevet. Vi har valgt at undlade beskrivelser af de brugsmønstre som er informations aflæsning. Der vil i afsnittet være beskrivelse af brugsmønstrene login, karaktergivning, aflevering, oprettelse af aflevering, beskeder og fravær.

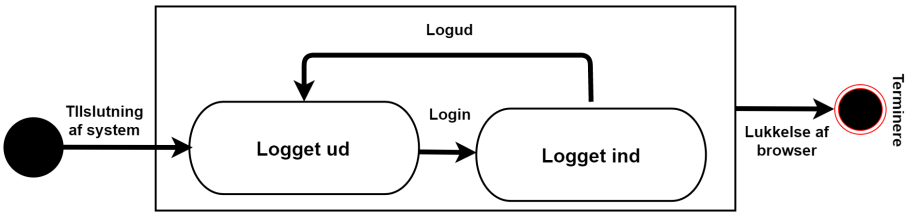


Figure 6.4: Tilstandsdiagram for Login

Login-brugsmønstret er illustreret på figur 6.4. Systemet er en webapplikation der skal reagere på forskellige typer af login alt efter om man er en del af personalet eller en kursist. Man tilsluttes til systemet ved at gå ind på hjemmesiden, hvorfra man bliver præsenteret for en login-anmodning. Man indtaster sit login og derfra er man inde i systemet. Herfra kan man logge ud og ind igen. Terminering sker ved lukkelse af vinduet eller browseren.

Objekter Person

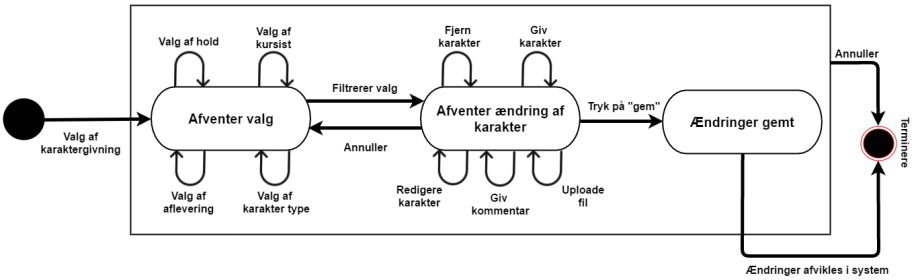


Figure 6.5: Tilstandsdiagram for Karaktergivning

På figur 6.5 vises det at karaktergivning udføres af en lærer, når vedkommende ønsker at angive karakterer for afleveringer, årskarakterer eller lignende. Læreren starter med at angive hvilket hold, kursist, aflevering og type karakter det handler om. Herefter indikerer grænsefladen om at læreren nu kan ændre karakterer. Nu kan læreren uploade filer som besvaring eller give kommentarer samt redigere, fjerne eller give karakterer. Når læreren har fuldført karaktergivningen gemmes besvarelsen.

Objekter Kursus, Lærer, Karakter, Kursist, Aflevering.

Funktioner (Giv karakter, rediger karakter)

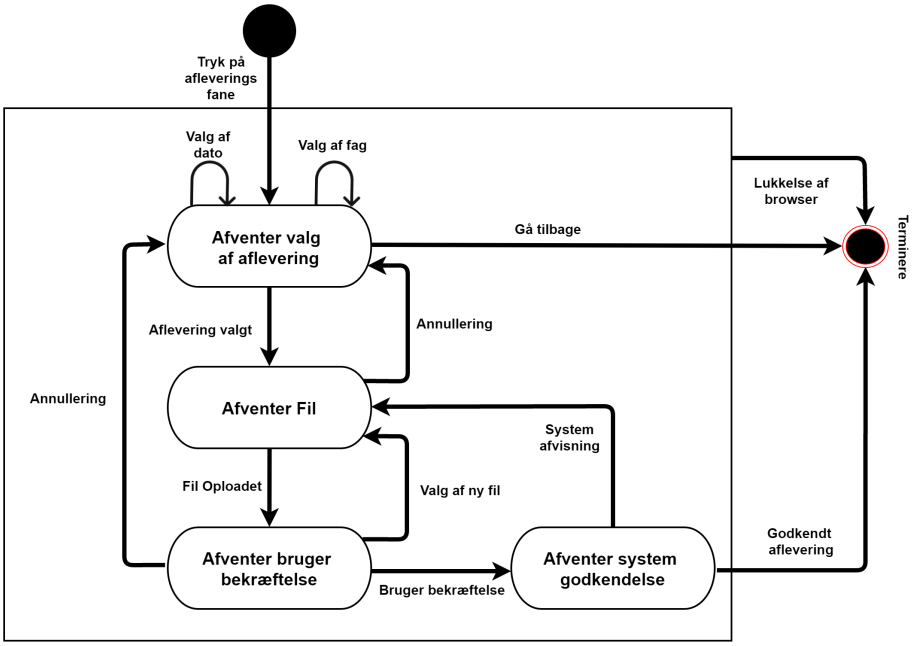


Figure 6.6: Tilstandsdiagram for Aflevering

På figur 6.6 vises det at aflevering igangsættes af kursisten, når vedkommende ønsker at aflevere en opgave tilkoblet et kursus. Kursisten logger ind via sit brugernavn og adgangskode, hvorefter kursisten vælger afleverings-fanen. Kursisten vil nu have mulighed for at vælge en aflevering ud fra en liste af afleveringer, ved valg af aflevering vil afleveringsfasen igangsættes. Kursisten vil få vist et vindue som beder om afleverings-filen, og vil ikke kunne fortsætte processen før en fil er vedhæftet. Kursisten kan vælge at afslutte, eller fortsætte processen ved at trykke på "vælg fil" knappen, hvor kursisten har mulighed for at kigge sine filer på enheden igennem. Når kursisten bekræfter sit valg af fil vil systemet lave en validering af afleveringens-protokollen og enten afvise eller fuldføre afleveringens-processen.

Objekter Kursist, Aflevering.

Funktioner (Upload aflevering, send aflevering, modtag aflevering, download afleveringsdokument)

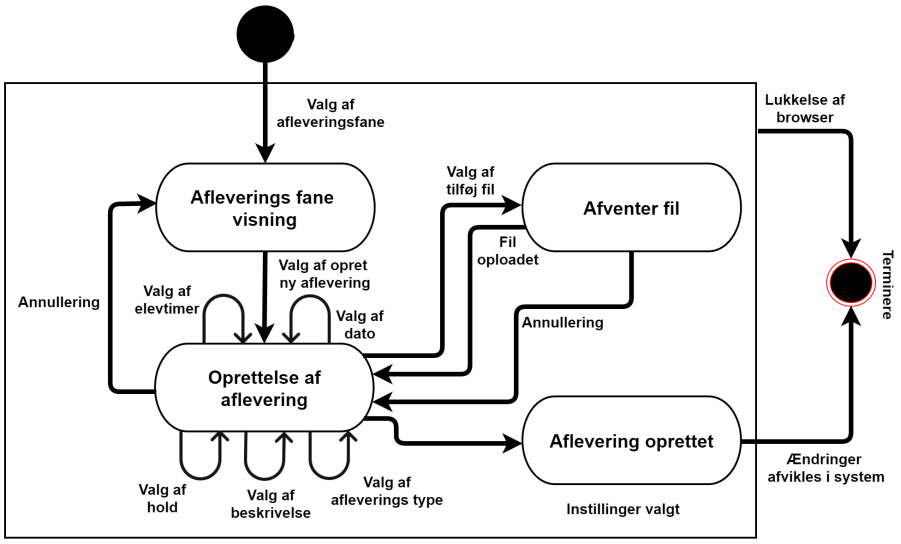


Figure 6.7: Tilstandsdiagram for oprettelse af Aflevering

På figur 6.7 vises det at læreren starter med at vælge afleverings-fanen, hvorefter vedkommende kan oprette en aflevering. I dette oprettelses-vindue kan man vælge hold, afleverings-type, mængde af elevtimer, dato og beskrive afleveringen. Når disse indstillinger er valgt bliver afleveringen oprettet og kan herefter tilgås af de specifikke kursister.

Objekter Lærer, Kursist, Aflevering.

Funktioner (Opret aflevering, upload dokument)

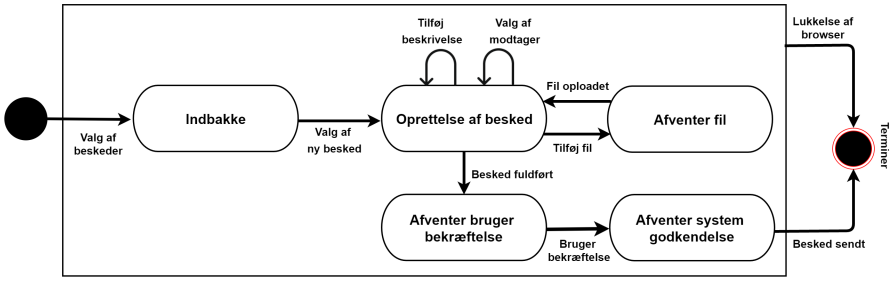


Figure 6.8: Tilstandsdiagram for Beskeder

På figur 6.8 vises hvordan brugsmønstret for at sende beskeder ser ud. At sende en besked igangsættes af brugeren, når vedkommende ønsker at sende en besked til andre brugere i systemet. Efter vedkommende har navigeret sig til indbakken er der mulighed for at oprette en ny besked, hvori der kan tilføjes en beskrivelse, filer og vælges modtager. Når vedkommende er færdig med dette bekræftes meddelelsen og sendes til den/de valgte modtagere.

Objekter Lærer, Kursist, Sekretær, Besked.

Funktioner (Send besked, videresend besked, modtag besked)

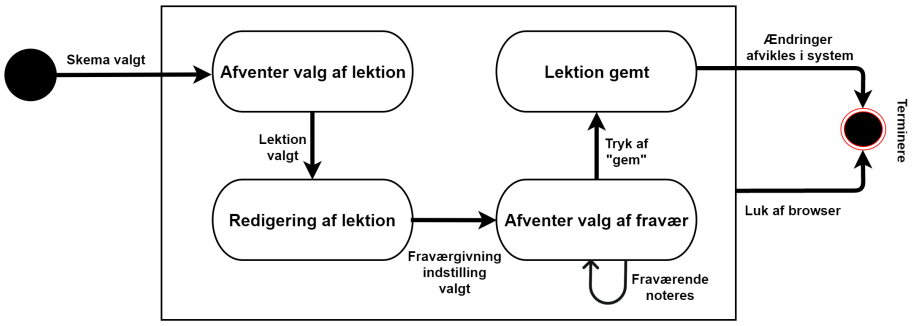


Figure 6.9: Tilstandsdiagram for fravær

På figur 6.9 kan man se at læreren igangsætter dette brugsmønster for at kunne registrere fravær. Vedkommende tilgår først den lektion som de vil registrere i, hvor der herefter vil være mulighed for at tilgå fraværsregistreringen. Heri kan læreren redigere i fravær samt registrere de fraværende og gemme disse informationer som automatisk vil vise sig i de valgte kursisters grænseflader.

Objekter Lærer, Kursist, Hold, Fravær.

Funktioner (Giv fravær)

Vi har derudover beskrevet information aflæsning, men har ikke valgt at lave et tilstandsdiagram dertil. beskrivelsen er nedenfor. "Se information" igangsættes af kursisten, når vedkommende ønsker at se diverse informationer vedrørende deres uddannelse. Kursisten vælger et emne (fane) i brugergrænsefladen. Herefter fremstilles det valgte emnes informationer.

Objekter Kursist, Nyhed, Lektion, Fravær, Aflevering, Karakter, Studieplan

Funktioner (Vis fraværsprocent, vis karaktergennemsnit, Se lektion)

6.3 Funktioner

Baseret på de tidligerenævnte brugsmønstre er der blevet dannet en liste af essentielle funktioner, se 6.2, over hvilke funktioner det fremtidige system burde indeholde. Funktionerne er opdelt i funktionstyper og i forskellige kompleksiteter. Hvoraf aflæsning er en funktionstype som aktiveres af aktørens behov for informationer, dette resultere i at der bliver vist relevante dele af systemettype. Opdatering er en funktionstype som aktiveres af hændelser, samt resultere i en opdatering i systemet. Signalering er en funktionstype som aktiveres når der sker en forandring i systemet, dette resultere i at denne forandring vises til de enkelte aktører.

Funktioner	Kompleksitet	Funktionstype
Aflæs besked	Simpel	Aflæsning
Send besked	Medium	Opdatering
Videresend besked	Simpel	Opdatering
Svar besked	Simpel	Opdatering
Aflæs aflevering	Simpel	Aflæsning
Aflever aflevering	Simpel	Opdatering
Hent afleverings dokumenter	Simpel	Aflæsning
Aflæs fravær	Simpel	Aflæsning
Aflæs skema	Simpel	Aflæsning
Hent dokumenter fra skema	Simpel	Aflæsning
Aflæs nyhed	Simpel	Aflæsning
Opret nyhed	Simpel	Opdatering
Slet nyhed	Simpel	Opdatering
Opret brugere	Medium	Opdatering
Vis brugere	Simpel	Aflæsning
Slet brugere	Simpel	Opdatering
Vis kurser	Simpel	Aflæsning
Opret kurser	Medium	Opdatering
Slet kurser	Simpel	Opdatering
Giv fravær	Medium	Opdatering
Aflys afleveringsbeskrivelser	Simpel	Opdatering
Tilmeld kursister	Simpel	Opdatering
Tilmeld lærer	Simpel	Opdatering
Giv karakter aflevering	Simpel	Opdatering
Giv karakter kursus	Simpel	Opdatering

Table 6.2: Funktionsliste, Oversigt over alle vores funktioner

Aflæs besked Alle brugere skal kunne aflæse deres beskeder.

Send besked Alle brugere skal kunne sende beskeder til alle brugere.

Videresend besked Der skal kunne videresendes fra alle brugere til alle brugere.

Svar besked Alle brugere skal kunne svare beskeder.

Aflæs aflevering Kursister skal kunne se deres afleveringer for deres kurser.

Aflever aflevering Kursister skal kunne aflevere deres afleveringer , således deres lærer kan se det.

Hent afleverings dokumenter Kursister skal kunne download afleveringer.

Aflæs fravær Alle brugere skal kunne aflæse fravær. Dette betyder at sekretærer og lærer skal kunne aflæse kursisters fravær, hvor kursister skal kunne aflæse deres eget.

Aflæs skema Lærer og kursister skal kunne aflæse deres skema, hvor deres kurser med lektioner findes.

Hent dokumenter fra skema Lærer og kursister skal kunne download dokumenter fra en lektion på skemaet.

Aflæs nyhed Alle brugere skal kunne aflæse der oprettet nyheder.

Opret nyhed En sekretær skal kunne oprette nyheder, således alle bruger kan aflæse dem.

Slet nyhed Sekretærer skal kunne slette nyheder der er oprettet i systemet.

Opret brugere Sekretærer skal kunne oprette brugere, dvs. både lærer og kursister.

Vis brugere Sekretærer skal kunne se alle de oprettet brugere, dvs. lærer og kursister.

Slet brugere Sekretærer skal kunne slette brugerer, hvis en lærer har sagt op eller hvis en kursist er droppet ud.

Vis kurser Kursister og sekretærer skal kunne aflæse de tilkoblet kurser der er til kursisterne. Derudover skal alle brugere kunne se deres lektioner samt kursister og lærer se de afleveringsbeskrivelser og dokumenter der eksistere til kurset.

Opret kurser Sekretærer skal kunne oprette kurser, således lærer kan gå ind og oprette lektioner til kurset. Herinde skal lærer også kunne oprette afleveringsbeskrivelser og dokumenter.

Slet kurser Sekretærer skal kunne slette kurser, hvor læreren kan slettet de tildelte lektioner. Derudover skal læren også kunne slette dokumenter og afleveringsbeskrivelser her.

Giv fravær En lærer skal kunne give skriftligt fravær og mødepligtigfravær, dvs. fravær for kursistens afleveringer, men også fravær for lektionerne.

Aflys afleveringsbeskrivelser Her skal lærer kunne aflyse afleveringsbeskrivelserne, således de stadig eksisterer i systemet, men får tilstanden aflyst.

Tilmeld kursister Her skal sekretærer kunne tilmelde kursisterne til de forskellige kurser de har valgt.

Tilmeld lærer Sekretærer skal kunne tilmelde lærerne til de kurser, som de underviser i, på denne måde kan læreren oprette lektioner, afleveringsbeskrivelser.

Giv karakter aflevering Lærer skal kunne give karakter til deres kursister igennem afleveringsbeskrivelserne.

Giv karakter kursus Her skal lærerne oprette årskarakter for deres kursister.

7 Design af arkitektur

I dette kapitel vil vi udarbejde en struktur til vores fremtidige program. Til at starte med vil vi udarbejde vores designmål samt deres prioritering. Herefter designes komponenterne i vores system, dette gøres ved brug af komponent arkitektur, disse komponenter vil herefter beskrives.

7.1 Kriterier

Formålet med dette afsnit er at formulere vores designmål samt sammenligne dem ved hjælp af prioritering. For at gøre dette har vi lavet en tabel af klassiske kriterier. Disse vises i tabel 7.1.

Kriterium	**Meget vigtigt**	**Vigtigt**	**Mindre vigtigt**	**Irrelevant**
Brugbart	✓			
Sikkert			✓	
Effektivt		✓		
Korrekt		✓		
Pålideligt	✓			
Vedligeholdbart			✓	
Testbart			✓	
Fleksibelt				✓
Forståeligt			✓	
Genbrugbart				✓
Flytbart				✓
Integrerbart		✓		

Table 7.1: Prioritering af designkriterier

Brugbart (meget vigtigt) Systemet skal være brugbart, da brugerne anvender det i dagligdagen under forskellige forudsætninger. Derfor skal det

kunne benyttes på både computer og smartphones. Det er derudover vigtigt at brugere vil have nemt ved at benytte og lære systemet, selv ud fra forskellige kompetencer og forudsætninger, da platformen vil være et vigtigt redskab for deres studie. Derfor vurderes brugbart til meget vigtigt.

Sikkert (mindre vigtigt) Efter ønske fra informanten om at programmet skulle gemme login informationerne, og hun selv ikke ønskede mere sikkerheds, er dette blevet vurderet til mindre vigtigt. Grundet det ikke er vurderet til irrelevant, er at programmet vil indholde informationer omkring karakter, fravær, afleveringer og beskeder. Derfor kræver det at der er et login til den enkle bruger.

Effektivt (vigtigt) En bruger vil benytte systemet mange gange i løbet af dagen, og derfor er det vigtigt at systemet kører hurtigt, da en bruger vil blive træt af at vente på systemet flere gange i løbet af dagen. Desuden vil mange brugere være tilkoblet systemet på samme tid, og de vil alle have forskellige enheder med forskellige hardware-niveauer. Vi ser det derfor som vigtigt at systemet ikke bruger tid på avanceret visualiseringer som animationer, som vil sænke hastigheden mellem systemet og brugeren, samt at systemet ikke kræver et for højt hardware-niveau.

Korrekt (vigtigt) Systemet er udarbejdet primært efter informantens krav for det færdige produkts design. Derudover er systemet også udarbejdet ud fra et spørgeskema, se afsnit 2.2. Dette betyder derfor vi har prioriteret korrektheden efter hvor godt disse krav er opfyldt.

Pålideligt (meget vigtigt) Vores system skal være meget pålideligt, da informationerne man får udgivet fra systemet skal være rigtigt. Hvis brugerne får forkerte oplysninger om hvor de skal være, eller afleverer en forkert aflevering mister vores system dens pålidelighed som ellers er en vigtig del af systemet.

Vedligeholdbart (mindre vigtigt) Vedligeholdbart vurderes til at være mindre relevant for vores system, fordi de grundlæggende krav for systemet kommer ikke til at ændre sig særlig meget. Dog skal systemet have en mulighed for at blive opdateret, hvis der forekommer nogle fejl.

Testbart (mindre vigtigt) Omkostningerne ved at sikre, at systemet opholder vores krav er en mindre vigtigt prioritet for os da vi ikke har nogle fastsatte krav og ikke nuværende har nogle planer for systemet udover i dette projekt.

Fleksibelt (Irrelevant) Da vi nuværende ikke har planer for det fremtidige produkt udover dette projekt har vi rangeret dette kriterie til irrelevant.

Forståeligt (mindre vigtigt) Som nævnt i kriteriet ovenover, "Fleksibelt", har vi ikke planer for dette produkt udover projektet. Vi har dog alligevel valgt denne til mindre vigtigt, da vi gerne vil have kode som er let at forstå.

Genbrugbart (Irrelevant) Genbrugbart kriteriet vurderes til at være irrelevant, da der er ikke blevet sat krav til at systemet skal være genbrugbart i andre systemer.

Flytbart (Irrelevant) Da informanten valgte at systemet skulle være en web-applikation, har vi valgt ikke at prioritere hvor flytbart systemet kan være.

Integrerbart (vigtigt) Vi har valgt at prioritere dette højt, da vores slutprodukt vil fungere som en testversion af hvordan det ville virke i problemområdet. Så hvis systemet skulle bruges i af et uddannelsessted, burde det være let at integrere.

7.2 Komponenter

Vores overordnede komponentarkitektur tager udgangspunkt i mønstret grundarkitektur kombineret med klient-server arkitekturen. Dette er gjort da der bruges et ukendt antal af brugere, som bruger den samme funktionalitet.

Figur 7.1 viser den overordnede komponentarkitektur. komponenten *Server* indeholder komponenter såsom *Brugergrænsefladen*, *Funktioner*, *Model* og den tekniske platform. Dataen skal være tilgængeligt for brugerne og derfor er alle komponenterne samlet på en server.

Klienten kobler sig til serveren via brugergrænsefladen i programmet og hvis der sker ændringer i brugergrænsefladen, vil dette få en effekt på *Browser* klienten. Samtidig er brugergrænsefladen relateret til funktioner, da funktionerne kan blive rekonstrueret, slettet eller oprettes nye, det er netop funktionerne som bestemmer hvordan brugergrænsefladen ser ud og hvordan den opføre sig. Derudover er *brugergrænsefladen* afhængig af komponentet *Bootstrap*, da bootstrap bestemmer udseendet af brugergrænsefladen. *Model* kan give en effekt til funktonerne, da modellen inderholder alle klasserne, og hvis de ændre sig så opføre funktionerne sig anerledes. *Model* er afhængig af komponentet database, da alt dataen bliver tilført lister i modellen, således disse kan tilgås i *Funktioner*.

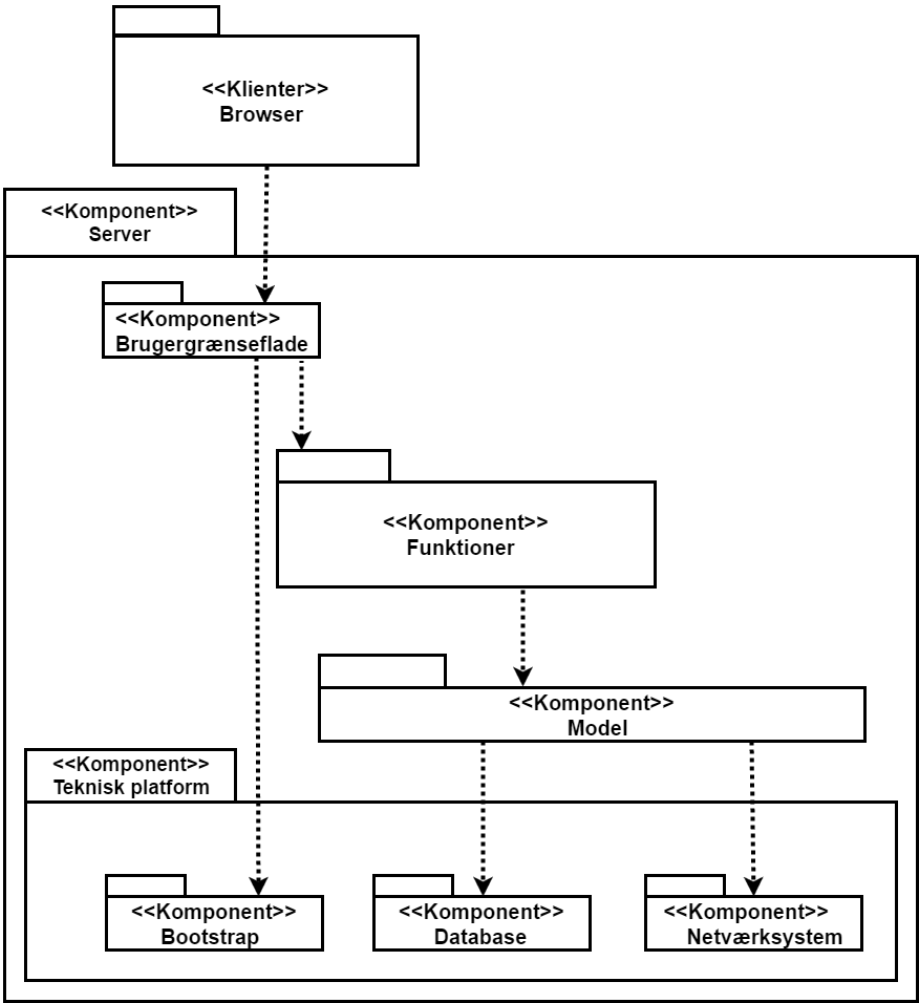


Figure 7.1: Overordnede grundarkitektur

7.2.1 Modelkomponent

På baggrund af vores problemområde har vi udarbejdet en modelkomponent for systemet. Der er blevet rekonstrueret det tidligere klassediagram se afsnit 5.2 og vurderet hændelser og klasser fra problemområdet. Dette er gjort for at finde ud af om der er opstået nye klasser eller om hændelserne som kan ændres til klasser.

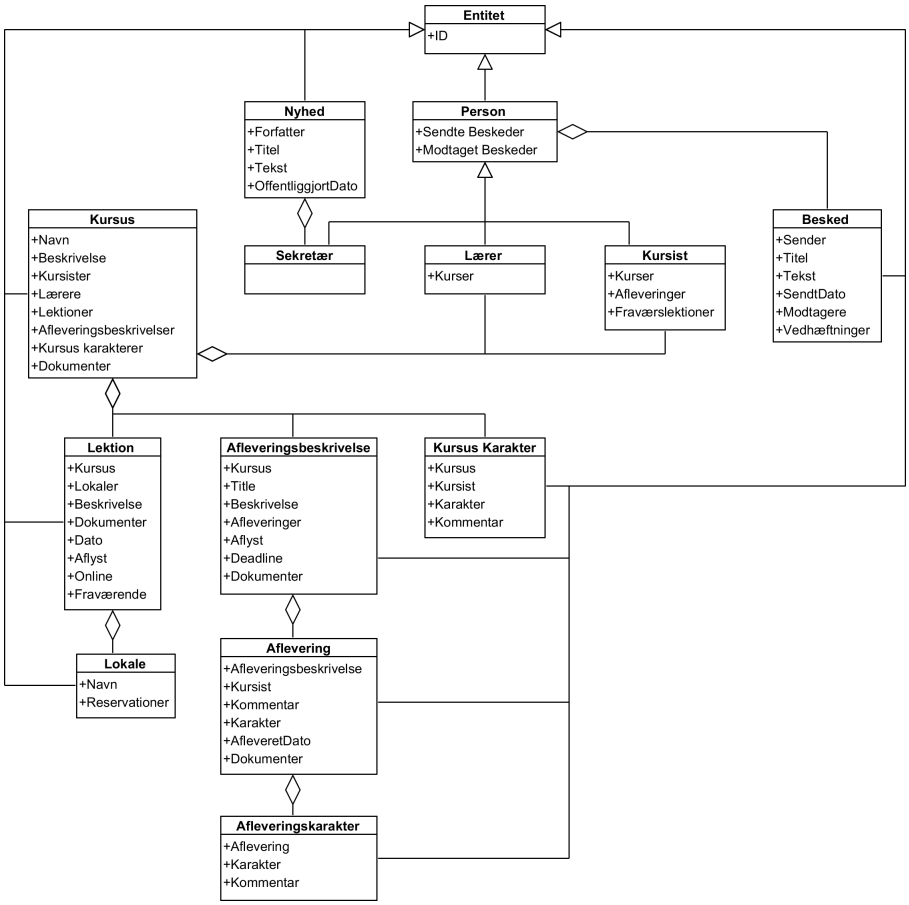


Figure 7.2: Klassediagram for modelkomponent

Figur 7.2 viser det rekonstrueret klassediagram, som er vores modelkomponent. Dette består af 14 klasser, hvoraf de 13 klasser arver fra *Entitet*. Der er gjort dette, da alle klasserne skal bruge et id, hvilket er generet af vores database og disse bruges til associeringer mellem klasserne. Derudover er der tilføjes attributter til alle klasserne, således man kan se hvad de indeholder.

Sammenlignet med vores model fra problemområdet er der kommet 3 nye klasser, disse er *Entitet*, *Afleveringskarakter* og *Kursus Karakter*. Hvoraf *Afleveringskarakter* klassen bruges til at give karakterer og kommentarer til en specifik aflevering. Derudover er den også en aggregering af aflevering, da den altid skal bruges til en aflevering.

Samtidig er der tilføjet klassen *Kursus Karakter* disse bruges til kursisten, når der skal gives årskarakter bådes skriftligt, men også mundtligt. *Kursus Karakter* er relateret til *Kursus*, det det netop er denne som indeholder karakterene for de individuelle kurser. Derudover ligner klassediagrammet det som vist i afsnit 5.2

7.2.2 Funktion komponent

I dette afsnit vil funktionskomponentet blive designet med udgangspunkt i funktionerne fra afsnit 6.3 anvendelsesområdet. Der vil være beskrivelser af funktionerne i forhold til hvordan de skal designes. Disse beskrivelser bruges herefter til implementeringen af funktionerne. Der er taget udgangspunkt i en række funktioner, disse funktioner er valgt da de er prioriteret højere end de andre funktioner. Disse er prioriteret højere, da det var de centrale funktioner som informanten ønskede, derudover er de også prioriteret ved en analysering af spørgeskemaet. Funktionerne er listet med beskrivelser nedenfor.

Aflæs aflevering Denne funktion har til formål at kursister kan aflæse deres afleveringer. Dette gøres ved at attributterne fra afleverings klassen opstilles systematisk i en tabel, således der skabes et overblik over de forskellige afleveringer. Samtidig skal den opdatere om den er afleveret eller ikke afleveret.

Aflever aflevering Da en af attributterne på aflevering indeholder en filepath, så skal denne tages i brug her. Funktionen skal sørger for at der kan uploades et dokument så det kan downloades igen af andre brugere. Denne opdatere modellen ved at gemme en ny aflevering.

Hent afleveringsdokumenter Denne funktion har til formal at downloade afleveringsdokumenter, og dette gøres ved brug af en filepath. Måden man henter dokumenterne er ved brug af afleveringstabellen, herefter skal den finde den bestemte filepath til afleveringsdokumentet.

Vis kurser Der skal laves en tabel over alle kurser der er tilkoblet brugeren. Herefter skal det være muligt at tilgå hver kursus side. Der skal laves en tabel som indeholder alle attributterne fra klassen kursus, dvs. at alle lektionerne skal være tilgængelige at se for brugere samt kursister, dokumenter og afleveringsbeskrivelser. Alle disse skal laves som faner, hvorefter der laves en tabel for hver fane.

Opret kurser Der skal være mulighed under kursus siden at sekretærer kan oprette nye kurser. De skal udfylde informationerne; hvilken lærer står for kurset, en detajleret beskrivelse af kursets mål og en plan over lektioner gennem et skoleår.

Slet kurser Sekretærer skal også have muligheden for at slette kurser igen fra kursus oversigten. Det kan være på grund af at kurset ikke længere er aktivt.

Tilmeld kursister En attribut i kursus klassen er kursister, som kan indeholde tilmeldte kursister. Kurser kan tilmeldes kursister af sekretærer. Funktionen skal sørge for at kun sekretærer kan tilmelde og fjerne kursister fra et kursus.

Tilmeld lærere Som med tilmeldelse af kursister kan sekretærer også tilmelde lærere til kurser. Funktionen virker på akkurat samme måde som tilmeldelse af kursister.

Aflæs Skema Skemaet er det første brugeren skal se når de har logget ind. Brugeren skal have vist en tabel over den nuværende uge og de lektioner de har i den uge. Brugeren skal kunne aflevere afleveringer igennem skemaet, se hvilket lokale lektionerne finder sted i og hvilken lærer der står for lektionerne. Herfra bruges attributter fra Kursus, Kursist, og Lektion. Fra Kursus kommer læreren som set på figur 7.2. Skemaet skal kunne opdatere hvilke lektioner hører til en given uge som brugeren vælger.

Aflæs besked Beskeder opdeles i indbakke og udbakke. Aflæsning af beskeder skal foregå i indbakken. Det kan ses at beskeder indeholder en sendt dato, vedhæftninger og en sender. Disse skal vises når en besked bliver valgt og brugeren skal have adgang til filer der er vedhæftet beskederne. Derudover skal beskeden kunne videresendes herfra.

Send besked Afsendelse af beskeder skal foregå på udbakke siden. Funktionen skal bruge en titel og en modtager og ikke nødvendigvis en brødtekst. Funktionen skal ikke kunne sende til en mail der ikke eksisterer ellers skal den gøre opmærksom på dette.

Videresend besked En videresendelse kan udføres på en besked der i øjeblikket bliver aflæst af brugeren. En videresendelse skal fungere ligesom en normal send besked funktion bortset fra at den skal bruge attributter fra den modtagede besked i form af tekst og titel.

Svar besked Svar besked er en funktion til at skrive en besked tilbage til afsenderen af en i øjeblikket åbnet besked. Herfra bruger funktionen afsenderen som modtageren og en ny besked kan sendes. Funktionen skal kun være tilgængelig på en modtaget besked i indbakken.

Opret brugere Som en sekretær skal det være muligt at lave nye brugere i form af kursister og lærere. Kun sekretærer/administratorer skal have denne mulighed. Til at oprette en bruger skal alle attributter i Kursist og Lærer udfyldes af sekretæren. En mail skal også laves til den nye bruger der kan bruges til at sende og modtage beskeder. Brugeren skal blive indsat i databasen under deres respektive tabeller.

Vis brugere Sektretærer og lærere skal kunne se brugere der ligger i databasen. Dette skal kunne gøres igennem lektion- og kursus-oversigten. Der skal kunne ses tilmeldte kursister og lærere. Kursister skal kunne se deres lektioners undervisere og deres medstuderende under de samme kurser. Administratorer/sekretærer skal kunne finde en given kursist og lærer og ændre i deres informationer.

Slet brugere Derudover skal det være en mulighed for administratorer at slette brugere der ikke længere er nødvendige at have i systemet. Dette kan være både lærere og kursister. Grunden ligger i at både kursister og lærere kan forlade institutionen. Det hjælper til at holde styr på brugerne i systemet så det ikke bliver utilpasseligt for administratorerne at arbejde med brugerne i systemet.

8 Design af brugergrænseflade

Dette kapitel har til formål at præsentere systemets brugergrænseflade, samt argumentere for valg i forhold til brugergrænsefladen. I afsnit 4 er der tidligere designet en prototype, dette afsnit vil derfor bygge videre på de designvalg der blev anlagt. Samtidig vil dette afsnit indrage forbedringer af prototypen samt gøre brug af spørgeskemaet.

8.1 Design principper

Vi har valgt at følge nogle design principper der beskrives af Benyon[1, s. 88]. Disse principper vil generelt sikre sig kvalitet og brugervenlighed af brugergrænsefladen. Der er lavet en liste over design principperne, i næste afsnit vil disse blive anvendt.

1. **Visibility** Det skal være let for brugeren at se hvad systemet foretager sig. Systemet skal være genkendeligt, da man nemmere vil genkende frem for at huske. Samtidig skal systemet sættes op, således layoutet af information kan tilpasse alle skærmstørrelser, på denne måde øger man muligheden for visibility.

2. **Familiarity** Der skal sørges for at sprog og symboler er noget som brugeren allerede kan relatere til i forvejen. Samtidig skal der være brug af metaforer, da brugeren kan relatere til disse fra andre lignende systemer og hjemmesider.

3. **Recovery** Systemet skal indeholde *"error recovery"* hvilket betyder at der skal forekomme advarsels-signaler, så som *"Er du sikker på du vil slette denne kursist"*.

4. **Affordance** Designet skal være genkendeligt for brugeren. Dette betyder systemet kan knyttes til andre systemer, hvilket brugeren har kendskab til. Samtidig skal der gøres brug af knapper, tekstbokse og navigationsbar af samme grund.

5. **Navigation** Der skal være simpel navigering. Dette kan gøres ved brug af en navigationsbar som menu. Her kan brugeren bruge denne til at vælge

elementer der navigere dem i systemet. Samtidig har brugeren også altid et overblik over, hvor brugeren befinder sig i systemet.

6. **Control** Det skal være klart hvem har kontrol i systemet. Dette betyder at brugren skal være klar over hvordan systemet giver respons. Samtidig kan systemet også tage brug af kontrol, ved opbevaring af afleveringer og lektier.

7. **Feedback** Brugeren skal altid have feedback af systemet. Hvis systemet laver en handling, hvor der skal afventes fra brugerens side, så skal systemet vise dette til brugren. Derudover kan der gives toner eller signaler, hvis der modtages beskeder.

8. **Flexibility** Systemet skal kunne tilpasse sig flere forskellige brugere. Dette gøres ved at systemet kan udføre handlinger på forskellige måder. Der kan gøres brug af genveje for erfaren brugere, men samtidig kan nye brugere stadig benytte andre metoder.

9. **Conviviality** Systemet skal være venligt for brugeren at benytte. Dette betyder at der ikke skal være unødvendige kommunikationsbokse mellem bruger og system. Samtidig skal det ikke være irriterende for brugeren, hvilket betyder at der skal være en tilpas mængde af besked bokse.

10. **Consistency** Der skal gøres brug af standard web features, så som et link er blåt. Samtidig skal elementerne gøres synlige for brugeren.

8.2 Brugergrænseflade

Vi har udvalgt at beskrive fire af vores faner som er i vores program. Disse er vurderet til at være vigtige til opfyldelse af kravene til programmet. Derudover vil design principperne beskrives de steder de er taget i brug.

8.2.1 Skema

Når der bliver logget ind med en kursist, vil skemaet være den første side der er vist. Skemaet er forsiden, da vores informant ønskede dette. Siden indeholder 3 dele; Menu oversigt**(1)**, ugeskift funktion**(2)** og oversigt over skemaet**(3)**.

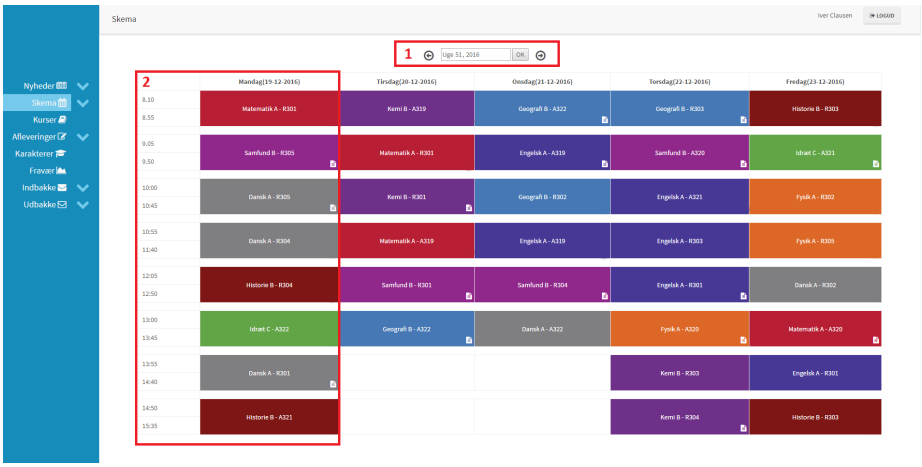


Figure 8.1: Design for skema

Navigationsbar oversigt(1) viser de muligheder som kursisten har at vælge mellem, når man vil skifte side, alt efter hvilket information kursisten har brug for. Dette design er generelt for alle fanerne, som er med til at skabe overblik for brugerne.

Der er brugt Familiarity princippet her, da vi benytter metaforer for at hjælpe kursisten med at holde overblik. Da kursisten genkender symbolerne, og kan relatere til deres betydning. Pilene indikere at den kan foldes ud, her kan man se de seneste nyheder, beskeder, lektioner og afleveringer.

Ved nyheder har vi brugt en foldet avis som et metafor for nyheder, i den grund at aviser indeholder nyheder. Vi benytter et kalender ikon for at repræsentere skemaet, da et skema minder generelt om en kalender og fungerer på samme måde.

Der benyttes en bog som ikon til kursus siden, da der er tilknyttet bøger til et hvert kursus. Samtidig repræsenterer det også at der er noget som skal læses her. Vi har benyttet en blyant og papir som et ikon til afleveringer. Dette repræsenterer at noget skal skrives ned eller løses.

Studenterhat er en vores metaforer for karaktererne, da karaktererne afgør om man får sin studenterhat. Derfor benytter vi hatten som et ikon for karakterer. Ved fravær har vi brugt en graf som et ikon, da man på siden kan se en graf over ens fravær. Ind- og udbakke benytter sig af det samme kuvert ikon, indbakke ikonet er fuldt ud, hvor udbakke er gennemsigtig. Grunden til at kuvert ikonerne er forskellige, er så man kan skelne mellem dem.

Derudover benyttes navigationsprincippet, da der gøres brug af en navigationsbar. Denne hjælper brugerne til at navigere rundt i programmet, og samtidig skaber et overblik over hvor kursisten befinder sig, da der lyses op for den side man befinder sig på.

Ugeskift(2) funktionen består af to knapper, hvor man kan skifte frem og tilbage i ugerne. Grunden til vi har designet denne funktion, var fordi det skal være nemt at navigere til de nærmeste uger.

Vi har benyttet pile som metaforer for ugeskiftning, hvor vi benytter familiarity princippet, hvilket simpelgøre hvad der menes med funktionen at kunne skifte uger i skemaet. Dette er med til at opfylde informantens krav om simpel navigering, og navigationsprincippet.

Oversigt af skema(3) viser de kurser som kursisten deltager i, i løbet af dagen. For at gøre det nemt for kursisten at have overblik, har vi indsat tidspunkter hvor det enkelte kursus starter og tilføjet en specifik farve for ethvert kursus. Farverne sørger for at kursisten ikke skal læse kursus navnet, men bare kan relatere til farven og på denne måde ved kursisten hvilken lektion man har. Der vil komme en metafor på skemaet, som symbolisere hvis der er blevet uploaded en fil til en bestemt lektion.

Der bliver benyttet visibility, affordance og flexibility principperne her, som de er beskrevet i afsnit 8.1. Visibility og affordance principperne spiller en fælles rolle, da vi benytter forskellige farver til at gøre det nemt for kursisten at skelne mellem sine kurser.

8.2.2 Indbakke

Indbakke fanen 8.2 består af to dele; en send funktion **(1)** og besked oversigt **(2)**, som er markeret på figuren.

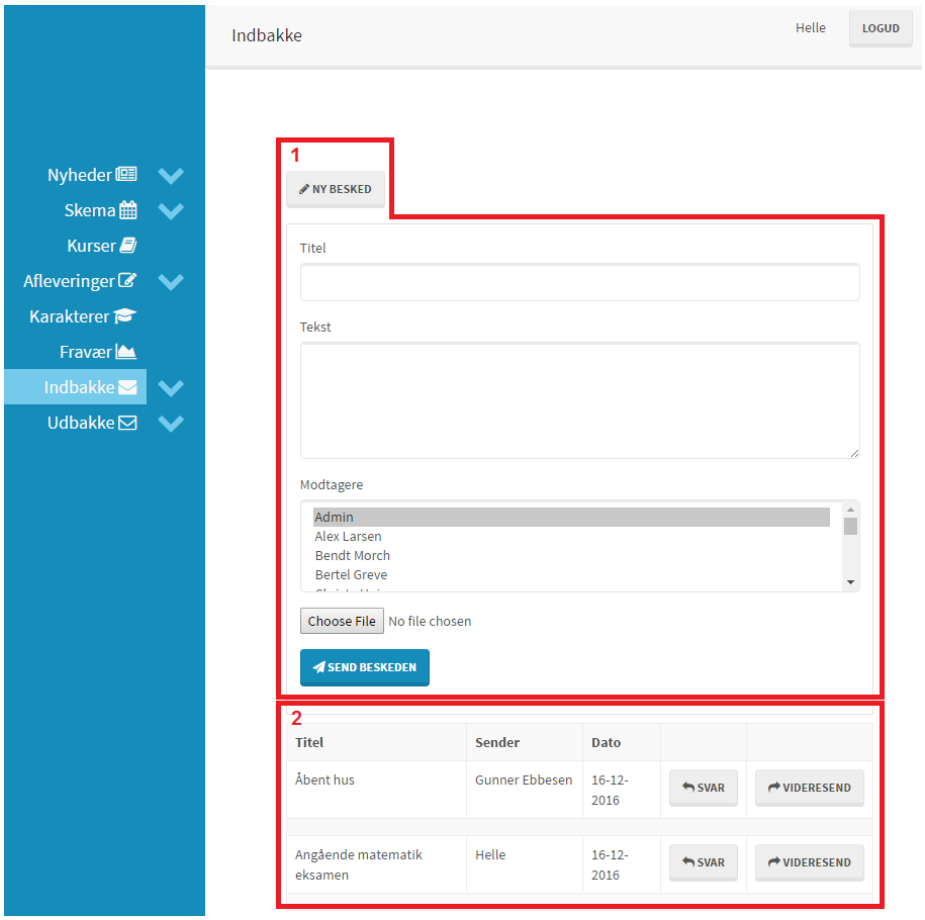


Figure 8.2: Design for indbakke

Send funktionen(1) består af en *NY BESKED* knappe, hvis man trykker på den, så kommer denne boks op, set på figur 8.2. Her har man muligheden for at sende en besked til en bestemt person eller flere. Da vores kriterie "brugbart" er stillet højt på vores prioritetsliste, har vi opsat designet på denne måde, så det ville være nemt for brugeren at bruge.

Her er der brug af principperne navigation og affordance. Der er brug af affordance, da knapperne, tekstboksene og listboksen er noget brugeren kan relatere til fra andre programmer. Dette er valgt, så brugeren i forvejen kender elementernes formål. Knappen *NY BESKED* befinder sig i toppen, da det skaber et overblik over hvor brugeren befinder sig. Dette betyder at der er tages brug af navigationsprincippet her. Samtidig er der valgt at *NY BESKED* knappen er blå, så den fremhæves.

Besked oversigt(2) viser de beskeder som man har modtaget fra andre personer. Her har man muligheden for at læse beskeden der er blevet sendt til brugeren, men også svare tilbage på beskeden. Hvis der er andre der også skal have beskeden, så har man også muligheden for at videresende beskeden. For at åbne beskeden skal man trykke på selve beskeden, herefter kan man læse

beskedens indhold.

Vi har opstillet det på denne måde, så det er simpelt og samtidig skaber et overblik over beskederne. Her opfyldes informantens krav, ved brug af principperne affordance og navigation, fordi der er taget brug af knapper og en tabel.

Her er brug af principperne affordance og control. Der er brug af affordance, da der er anvendt knapper og en tabel. Dette er noget brugeren kan relatere til fra andre programmer, Vi har gjort dette, for at simpelgøre det for brugeren, da elementernes formål er noget brugeren kender til.

Derudover er der brug af control, fordi systemet opbevarer beskederne både i indbakke, men også i udbakke.

8.2.3 Afleveringer

Afleveringsfanen indeholder en oversigt over afleveringer **(1)** og en aflever funktion **(2)**, som vises på billede 8.3.

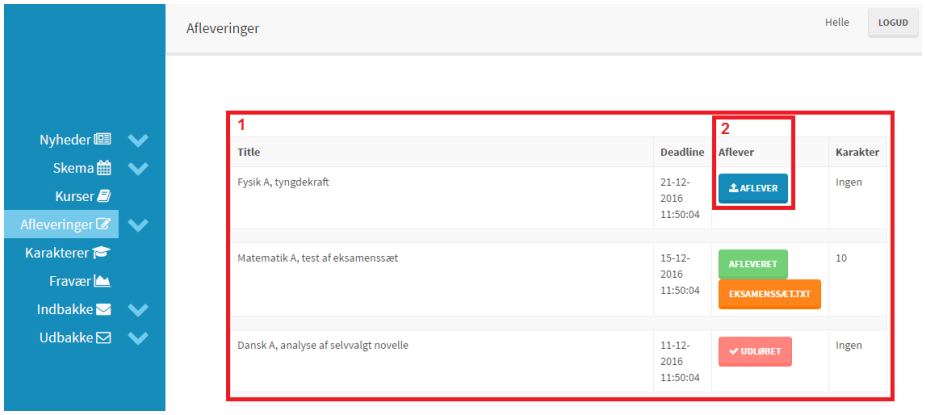


Figure 8.3: Design for aflevering

Oversigt over afleveringer(1) viser de afleveringer som kursisten har. Oversigten viser også om en bestemt aflevering er blevet afleveret, aktiv eller udløbet. Man har mulighed for at se afleveringens beskrivelsen, vedhæftet dokumenter eller afleveringens deadline. Dette hjælper kursisten med at have et nemt overblik over de vigtige informationer til de hver enkelte afleveringer. Når kursisten uploader en fil, så kommer der en orange boks, hvor man kan downloade filen ved at klikke på den orange boks.

Her er der anvendt principperne flexibility, control, Affordance. Flexibility er anvendt, så brugerne både kan se om afleveringerne er afleveret inde på kursus siden eller på denne side. Dette har gjort, så programmet kan tilpasse flere brugere, så både erfaren brugere og nye brugere kan anvende det. Control er anvendt, fordi systemet opbevarer afleveringerne, på denne måde skal brugeren ikke huske på alle sine afleveringer. Affordance er anvendt igen så det er genkendeligt for brugeren, ved brug af knapper og en tabel.

Aflever funktionen(2) består af en afleveringsknap, som ændre farve alt efter hvilke status afleveringen har. Her anvendes principperne affordance, familiarity og feedback. Knappen har tre forskellige farver, hvor knappen er grøn hvis den er afleveret, rød hvis den er udløbet og ikke afleveret og blå hvis den ikke er afleveret. Disse farver er valgt, fordi rød symboliserer noget farligt, blå symboliserer neutralt og grøn symboliserer noget godt. Dette leder til princippet feedback, da programmet viser om afleveringen er afleveret eller ikke. Samtidig er der feedback i form af en dato og Dokumenter, så brugeren altid ved hvornår der skal afleveres, men også om der findes dokumenter. Hvis der ikke er findes dokumenter vil feltet have teksten *INGEN DOKUMENTER*. Derudover anvendes feedback også når en kursist har afleveret, da programmet viser en bekræftelse på at man har afleveret en opgave.

Familiarity er anvendt ved dokument ikonet ved afleveringer, fordi brugeren nemmere kan relatere til denne og ved i forvejen at dette er et dokument.

8.2.4 Karakter

Karakter siden se 8.4, består af en oversigt af karakterer **1** og kursistens gennemsnitskarakter **2**. Der er ikke lagt vægt på karakterne i programmet, hvilket der burde, men det blev ikke prioriteret så højt som de andre funktioner der skulle implementeres.

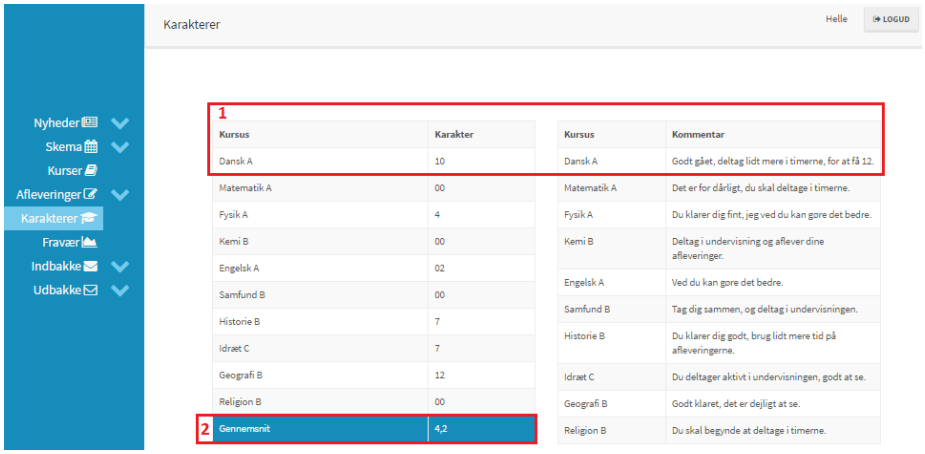


Figure 8.4: Design for karakter

Oversigt over karakterer(1) indeholder kursistens karakterer for de forskellige kurser, som personen deltager i. Oversigten viser både karakteren for det bestemte kursus, og kommentaren som læreren har vedlagt karakteren. Conviviality princippet er også benyttet her, da det skal være venligt og simpel for brugeren at kunne se sine karaktere, og ikke blive irriterende, hvis det er besværligt at finde sine karaktere.

Gennemsnitskarakter(2) bliver vist i bunden af siden, så kursisten har muligheden for at se sit samlede gennemsnitskarakter for de kurser personen har tilmeldt sig.

9 Implementation

I dette kapitel, vil vi gennemgå, hvad MySQL er, samt opsætningen af vores database med tabeller og relationer. Herefter vil vi beskrive hvilke klasser og metoder, som vi har implementeret i vores program for at oprette de enkelt elementer i databasen, efterfulgt af test af databasen. Til sidst vil vi beskrive, hvilke valg vi har fortaget omkring opbygning af vores brugergrænseflade.

9.1 Implementation af database

I dette afsnit vil vi beskrive vores valg af database samt implementationen og testningen af dette.

9.1.1 MySQL

MySQL(Structured Query Language) er et program som er opensource, hvilke betyder at programmet er tilgængeligt for hvem som helst. MySQL benyttes til at håndtere en større mængde af data i form af relationelle databaser. Dette gøres ved at programmet opretter en database ved at strukturere datasættet ind i separate tabeller, fremfor at lave én tabel, hvor alt data bliver indsat. Udover at separere data ind i tabeller, kan MySQL opstille relationer mellem tabellerne, som fortæller hvordan tabellerne tilgår hinanden. Nogle af disse relationer kan være en til en-relation, en til mange-relation og pointers mellem tabellerne.

Da MySQL kører som en server bliver denne database tilgængelig for alle, som har adgang til den, og hermed får hvert enkelt computer deres egen database. [4]

9.1.2 Opsætning af database

Vi valgte, at bruge MySQL til at opsætte vores database. På denne måde fik vi vores data separeret i tabeller, og hermed oprettet en database som indeholdte disse tabeller. I figurerne 9.1, 9.2, 9.3, 9.4 vises tabellernes opsætning af data, hvor linjen mellem tabellerne er relationerne.

Tabeller

Databasen indeholder to forskellige typer tabeller. Den ene er en normal tabel som indeholder data om en person, kursus og så videre. Den anden er en relationstabel som forbinder to tabeller ved at bruge deres unikke IDs, eller forbinder filer til en tabel. For at give et overblik over, hvilke tabeller vi har i vores database, vil vi beskrive alle tabellerne. Vi har valgt at markere de tabeller med fed skrift og relationstabellerne med kursiv.

Det skal nævnes at i alle de normale tabeller, er der et unikt ID, som ikke bliver beskrevet i listen nedenunder, da det er det samme for dem alle, altså et unikt ID som vi bruger til at skelne dem med.

- **Assignment** indeholder data om en aflevering. Aflevering indeholder et afleveringsbeskrivelses ID, et student ID, en afleveringskarakter ID, en kommentar og en dato.
- *AssignmentFile* indeholder alle de filer som er tilknyttet til afleveringer. Den bruger afleveringernes unikke ID til at tilknytte disse filer.
- **AssignmentGrade** indeholder data om en afleveringskarakter. Afleveringskarakteren indeholder et afleverings ID, en karakter og en kommentar.
- **AssignmentDescription** indeholder data om en afleveringsbeskrivelse. Afleveringsbeskrivelsen indeholder et kursus ID, en beskrivelse og en deadline.
- *AssignmentDescriptionFile* indeholder alle de filer som er tilknyttet til afleveringsbeskrivelserne. Den bruger afleveringsbeskrivelsernes unikke ID til at tilknytte disse filer.
- **Course** indeholder data om et kursus i systemet. Kursuset indeholder et navn og beskrivelse af kurset.
- *CourseFile* indeholder alle de filer som er tilknyttet kurser. Den bruger kursers unikke IDs til at tilknytte disse filer.
- **CourseGrade** indeholder data om en karakter til et kursus. Karakteren indholder et kursus ID, og et studenter ID. Sammen med karakteren er der også tilknyttet en kommentar.
- **Lesson** indeholder data om en lektion til et kursus. Lektion indholder et kursus ID. Sammen med lektion er der tilknyttet en beskrivelse, og en dato for, hvornår lektion er aktiv. Derudover kan lektion være online eller aflyst.
- *LessonFile* indeholder alle de filer, som er tilknyttet til en lektion. Den bruger lektions unikke ID til at tiknytte disse filer.
- *LessonRoom* indeholder data om et lokale, som er tilknyttet til en lektion. Den bruger et unik lektions ID og et unik lokale ID til at tilknytte disse to.
- **Message** indeholder data om en besked. Beskeden indholder et sender ID, en titel, tekst og en dato.

- *MessageFile* indeholder alle de filer, som er tilknyttet til en besked. Den bruger beskedens unikke ID til at tilknytte disse filer.
- *MessageRecipient* indeholder data om en besked, som er tilknyttet til en person. Den bruger en persons unikke ID og en beskeds unikke ID til tilknytte disse to.
- **News** indeholder data for en nyhed. Nyheder indeholder et forfatter ID, en titel, en tekst og en dato.
- **Person** indeholder data om en person i systemet. Person indeholder et navn, et brugernavn og en adgangskode.
- **Room** indeholder data om et lokale. Lokalet indeholder et navn.
- *StudentAbsence* indeholder data om en studerendes fravær, som er tilknyttet mellem en studerende og en lektion. Den bruger et unik ID for en studerende og et lektions ID til at tilknytte disse to.
- *StudentCourse* indeholder data om et kursus, som er tilknyttet til en studerende. Den bruger et unik ID for en studerende og et unik kursus ID til at tilknytte disse to.
- *TeacherCourse* indeholder data om et kursus, som er tilknyttet til en lærer. Den bruger et unik ID for en lærer og et unik kursus ID til at tilknytte disse to.

Relationer

I vores database har vi benyttet tre relationer mellem tabellerne. Relationerne viser, hvordan en tabel tilgår en anden tabel. Vi vil beskrive de tre relationer.

- **Mange til mange-relation** viser at flere tabeller bliver relateret til flere tabeller. I figur 9.1 vises en relationslinje mellem kursus og kursusfiler. Relationslinjen viser at flere kursuser kan have flere filer.
- **En til mange-relation** viser at en tabel bliver relateret til flere tabeller. I figur 9.3 vises en relationslinje mellem person tabellen og nyheds tabellen. Relationslinjen viser, at en person kan have flere nyheder.
- **En til en-relation** viser at en tabel bliver relateret til en tabel. I figur 9.2 vises en relationslinje mellem afleverings tabellen og afleveringskarakter tabellen. Relationslinjen viser, at en aflevering kun kan have en karakter.

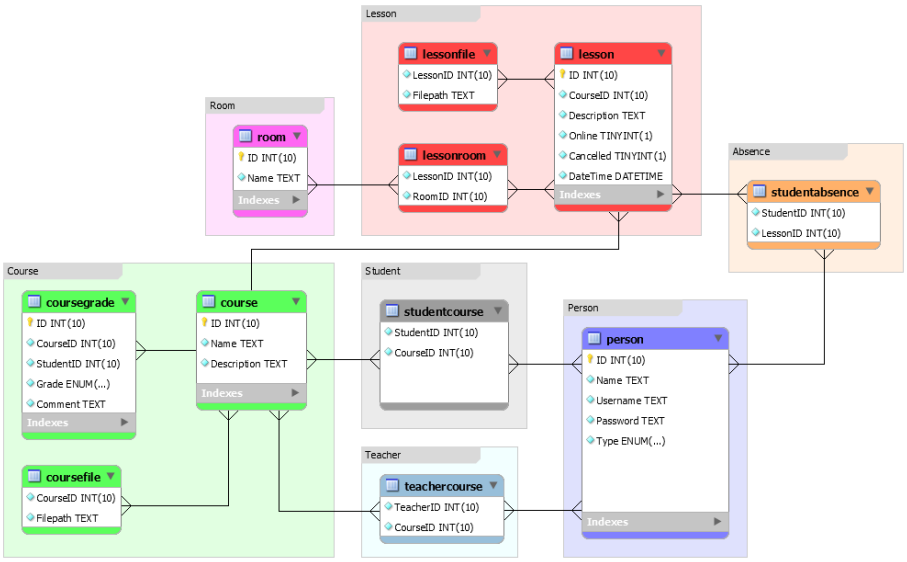


Figure 9.1: Database opsætning af person, kursus, lektion og lokale

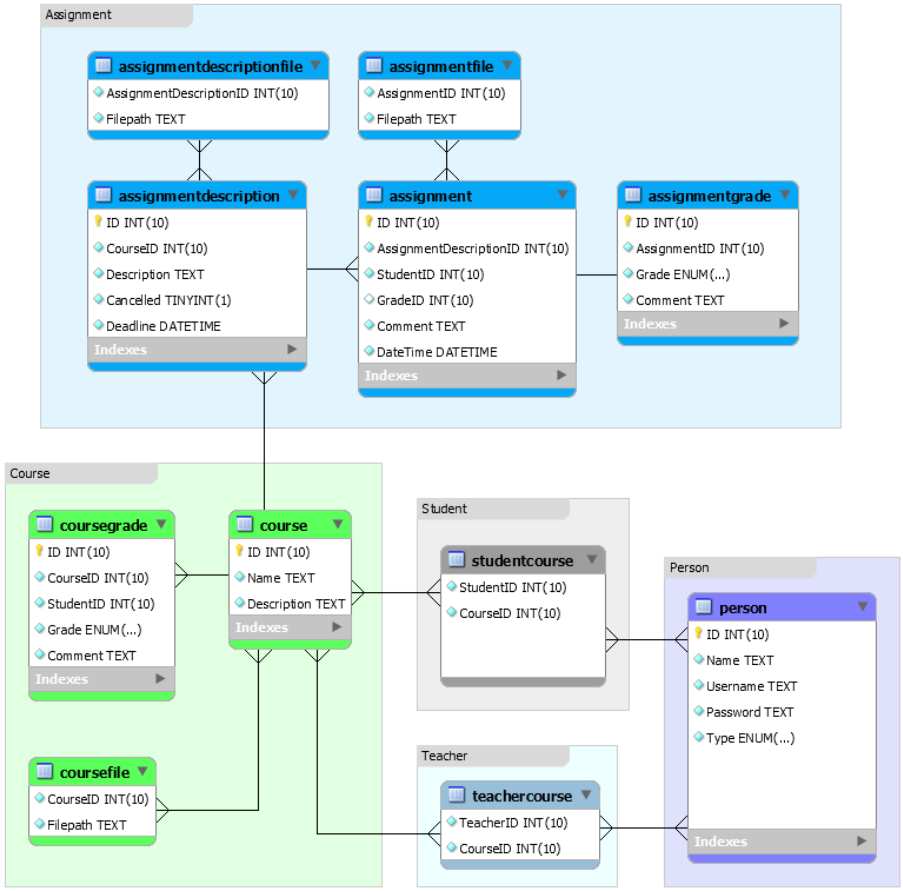


Figure 9.2: Database opsætning af person, kursus og aflevering

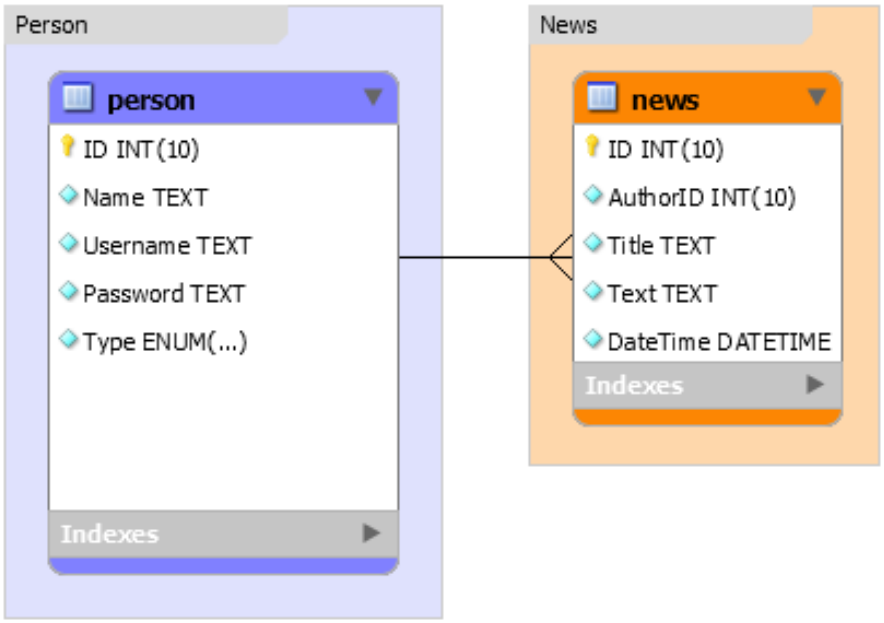


Figure 9.3: Database opsætning af person og nyheder

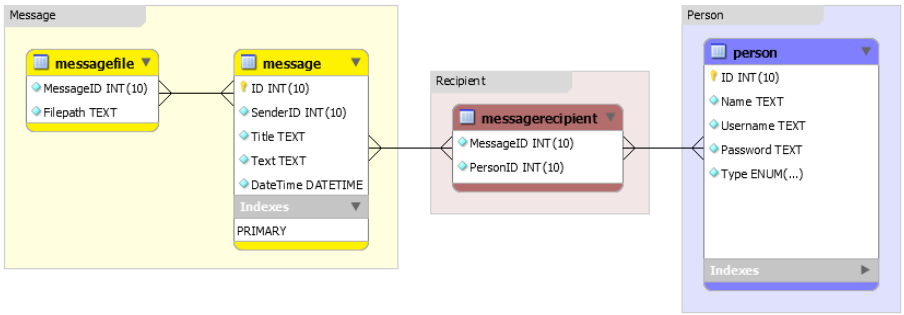


Figure 9.4: Database opsætning af person og beskeder

9.1.3 Implementation

Her vil vi gennemgå de forskellige klasser i vores program, som interagerer med databasen. Disse klasser er følgende:

- **Creator**, bruges til at oprette nye elementer i systemet. Eksempelvis ville denne klasse bruges når en besked blev oprettet, til at gemme den i databasen.
- **Remover**, bruges til at fjerne elementer i systemet. Hvis et kursus for eksempel skulle aflyses, ville denne klasse håndtere det.
- **Extractor**, bruges til at hente data fra vores database.

- **Entity**, indeholder metoder som alle model-klasserne benytter. Eksempelvis kan *Person*-klassen tilgå metoder til at hente alle personer i databasen, eller hente en specifik persons relationer til kurser og lignende igennem denne klasse.

Udover disse klasser som interagerer med databasen, er der også *Setup*-klassen. Denne bruges til at oprette de tidligere beskrevne tabeller i MySQL-databasen, og skal derfor kun bruges første gang programmet starter.

Creator

Denne klasse indeholder alle metoderne for når nye elementer bliver oprettet i systemet. Nogle eksempler på dette er:

- **Person**. Når en person bliver sat ind i systemet, kræves der navn, brugernavn og et password. Udover disse 3, er der også ID. Et ID bliver automatisk lavet af databasen, som også sørger for at alle IDs er unikke, ved at inkrementere det med 1 hver gang en ny bruger laves. Når dette er gjort vil personen være i systemet og det vil være muligt at logge ind og tilmelde sig kurser, sende beskeder eller lignende.
- **Kursus**. At lave er kursus følger den samme procedure som en person, dog med andre variabler.
- **Besked**. En besked oprettes i systemet når en bruger sender den. Udover de variabler som gemmes i tabellen til beskeder, vil der også opstå en linje for hver modtager af beskeden i en anden tabel, nemlig MessageRecipient-tabellen, som tidligere beskrevet indeholder alle modtageres ID af alle beskeder samt beskedernes ID.

Denne klasse indeholder altså alle metoderne som gemmer nye elementer i databasen. For det meste følger disse metoder denne procedure:

- Der checkes at parametrene er iorden.
- Der indsættes værdierne i den sammenhængende tabel. (Hvis en person er ved at blive oprettet ville den indsætte i Person-tabellen).
- Hvis specifikke relations-tabeller eksisterer til dette element, indsættes der også den sammenhængende data i dem.

Remover

Når et element i systemet skal fjernes, bliver denne klasse brugt.

Den fjerner de steder i databasen hvor elementets ID opstår, altså derfor i tabellen til elementet. Udover dette, fjerner den også fra elementets relations-tabeller. Hvis elementet var en student ville den derfor fjerne de linjer i relations-tabellen mellem personer og kurser hvor studentens ID opstod.

Dette virker på samme måde for alle elementer der kan slettes i systemet. En større metode ville være den til kurser, som fjerner følgende:

- Relationer mellem kurset og personer

- Relationer mellem kurset og kursets filer
- Kursets lektioner og alle relationer som de indeholdte
- Kursets opgavebeskrivelser og alle relationer som de indeholdte
- Kursets karakterer

Det burde nævnes at vi ikke benytter *cascade delete*[2] i vores database. Dette ville tillade os at slette et element og alle dens relationer med én kommando, istedet for manuelt at fjerne disse relationer.

Extractor

Denne klasse indeholder metoder til at hente data fra databasen, som bliver brugt igennem *Entity*-klassen, se 9.1.3. Disse metoder er:

- **ExtractIDs**, som henter alle IDs i en bestemt kolonne, baseret på krav. Eksempelvis bruges den til at hente alle personers IDs som er tilmeldt et specifikt kursus. Denne søgning er baseret på kursets ID og der søges i relations-tabellerne mellem personer og kurser. De IDs som bliver returneret kan herefter bruges til at hente alle de valgte personers informationer i databasen.
- **ExtractFilepaths**, som bruges til at hente et bestemt elements fil-lokationer. Da flere elementer i programmet kan bruge filer (beskeder, afleveringer, kurser og så videre), er denne metode lavet til at kunne hente disse.
- **Extract(element)**, som bruges til at hente elementer fra databasen. Der er sådan en metode her for hver model i databasen.

Entity

Entity-klassen indeholder metoder som henter fra databasen, nogle ved hjælp af *Extractor*-klassen, se afsnit 9.1.3. Der bliver her forklaret nogle af de essentielle dele af klassen:

- **GetByID**, bruges til at hente et specifikt element (eksempelvis en person) fra databasen, ved hjælp af elementets unikke ID.
- **GetAll**, henter alle elementerne fra en bestemt tabel og returnerer disse i en liste af det sammenhængende objekt-type, bruges igennem *Extractor*-klassen.
- **GetLatest**, henter den sidste linje i den bestemt tabel. Hvis denne metode blev kaldt igennem *Student*-klassen, ville den altså hente den nyeste student i systemet.

Som nævnt tidligere, arver alle model-klasserne fra denne klasse. Grunden til dette er at alle modellernes data er gemt i databasen, og derfor har brug for metoder til at hente denne data.

9.1.4 Test

For at teste vores implementation af databasen, har vi lavet teste til *Creator*-klassen, se afsnit 9.1.3 og *Remover*-klassen, se afsnit 9.1.3, *Extractor*-klassen, se afsnit 9.1.3, og*Entity*-klassen, se afsnit 9.1.3, er der ikke lavet teste til da de indirekte bliver testet af de 2 klasser nævnt tidligere.

Dette er fordi når vi tester *Creator* og *Remover*-klasserne bruger vi metoderne i *Extractor* og *Entity*-klasserne, og hvis der var fejl i dem ville de findes i disse tests. Til sidst burde det nævnes at vi ikke har lavet specifikke teste til *Setup*-klassen som opretter tabellerne i databasen, da disse tabeller også bliver testet af alle de andre tests.

Creator

For at teste om *Creator*-klassen virker korrekt, er der lavet teste til hver metode som bruges i klassen. For det meste følger disse teste denne procedure:

- Sikrer først at der forekommer undtagelser hvis metoden bliver kaldt med værdier som ikke eksisterer eller er tomme.
- Sikrer at der ikke forekommer undtagelser hvis metoden bliver kaldt med korrekte værdier.
- Sikrer at elementet som bliver lavet eksisterer i databasen efter metoden er kaldt.

Hvis disse teste lykkes, fungerer metoden.

Remover

For at teste *Remover*-klassens metoder følges denne procedure:

- Elementet der skal fjernes laves.
- Mulige relationer til elementet laves.
- Sikrer at både elementet og dens relationer eksisterer.
- Metoden til at fjerne elementet bliver kaldt.
- Sikrer at både elementet og dens relationer ikke eksisterer længere.

Som eksempel til denne procedure tages der her udgangspunkt i en person som skal fjernes:

- Der skabes en person i systemet.
- Herefter bliver person tilmeldt kurser, afleveringer, tildelt fravær samt sender og modtager nogle beskeder i systemet.
- Der bliver tjekket om personen og disse relationer er i systemet.
- Metoden til at fjerne personen bliver kaldt.
- Til sidst bliver det sikret at personen og dens relationer ikke længere er i systemet.

Kodedækning

Integrations testene dækker 75% af de eksekverbare instruktioner i database- og modelklasserne, derimod er brugergrænsefladen ikke dækket af nogen teste. Grunden til at der ikke er benyttet programmatisk testing af brugergrænsefladen er at der ikke er andre komponenter der afhænger af den. Brugergrænsefladen bliver i stedet testet gennem simpel *Ad Hoc* testing hver gang der laves ændringer i den bag liggende kode.

9.2 Brugergrænseflade

Dette afsnit vil omhandle vores opbygning, valg af ASP.NET model samt hvordan de enkle dele arbejder sammen til skabe brugergrænsefladen for programmet. Det blev besluttede at løsningen skulle laves i asp.net web forms ved brug af 2page-Model. Hvilket betyder at hver enkel side i programmet består af en aspx fil(In-Line Code) og en cs fil (Code-Behind). Dette blev valgt da denne model passer bedre til en web applikation eftersom koden bliver separeret fra markup brugergrænsefladen. Denne adskilling gør at designet på de enkle sider kan ændres og tilpasses, hvis der sker en ændring. Derudover giver det os mulighed for at genbruge kode til flere sider, f.eks. download af fil hændelsen.

9.2.1 Aspx Brugergrænseflade

Aspx filen har til formål at indeholde visuelle elementer, markup, server kontroller, og statisk tekster, denne del bygger altså på html og css kode. Sammen skaber de brugergrænsefladen til siden. Men eftersom vores side skal præsentere data unikt til den enkle bruger fra en database, samt mængden af data varierer fra bruger til bruger, er der lavet en dynamisk skabelse af html i vores cs fil, som så sættes ind i aspx filen se afsnit 9.2.2. Derfor indeholder vores aspx fil generelt kun statiske elementer som er fælles for alle brugere. Et eksempel på dette kunne være *table headers*.

Eftersom vores programs formål generelt er at vise informationer til brugeren har vi valgt at bruge asp: tabeller til at vise dataene, samt for at gøre den dynamiske skabelse mere simpel. Herved kan der tilføjes og fjernes rækker og celler dynamisk for den enkle tabel.

De forskellige elementer i aspx vil desuden have tilknyttet forskellige klasser og styles via bootstrap og egne css stylesheets. Dette vil skabe et ens design for de forskellige sider, samt tilpasning til forskellige skærmopløsninger under præsentationen af siden.

9.2.2 Code-Behind

Cs filen har til formål at indeholde vores præsentations logik, dynamisk skabelse, samt hændelser tilknyttet sidens elementer, denne del bygger altså på c# kode.

Når den enkle side bliver indlæst bliver metoden `Page_Load()` i cs filen kaldt, og det er her den dynamiske skabelse sker. Ved hjælp af *HtmlGenericControl*

klassen kan html kontroller skabes, samt klasser, styles og andre attributter kan sættes. Herved kan elementer som f.eks. *TableCells* skabes og værdier kan indsættes, hvor ved hjælp af *Webcontrol.Controls.Add()* metoden kan html logikken skabes, og de enkle elementer vil blive tilføjet i aspx filen under den rigtige container, som findes ved kontrollens unikke id, og herved skabes sidens layout.

Desuden indeholder cs filen også andre hjælpe metoder, samt hændelser. Et eksempel kunne være indbakke siden, hvor hændelserne til videresend og svar findes. Disse hændelser er tilknyttet dynamiske elementer som bliver sat ind i aspx filen, som i dette tilfælde er lavet dynamisk i aspx filen, men ved andre sider er lavet statisk.

9.2.3 Layout sider

I vores program har vi valgt at have *master* sider, hvilket er sider som har et layout som andre sider i programmet kan arve fra. Formålet med disse sider er de kan benyttes til layouts som bruges på adskillige sider. Heriblandt vores generalle layout side, som indeholder navigations bar og header med logud funktionalitet, som ses på alle vores sider undtagen login siden. Vi har samtidigt valgt at lave layout sider som arver fra hinanden, f.eks. New message layout siden som arver fra det generalle layout, samt indeholder sit eget layout. Herved kan vi sætte vores sider som f.eks. indbakke og udbakke til at arve fra New message layout siden, og have begge layouts se figur 9.5.

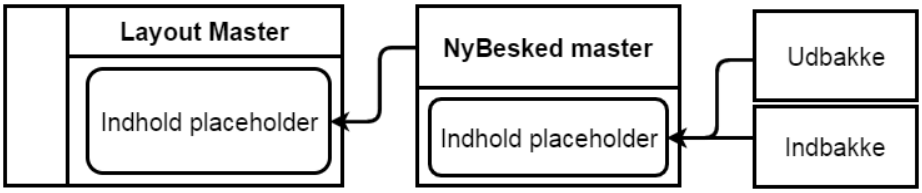


Figure 9.5: Layout for shares pages

10 Observation af det udviklede system

I dette afsnit vil der blive gennemgået den anden observation der blev foretaget som en del af projektets anden iteration. Informanten fik til formål at udføre lignende opgaver som blev gennemgået på prototypen, men denne gang på det udviklede system. Observationen forløb på samme facon som til den første observation, hvori informanten blev stillet en række af opgaver der skulle reflektere de ønsker og krav der blev klargjort til systemdefinitionen baseret på informantens egne. Spørgsmålene blev valgt for at gøre det muligt at lettere stille en sammenligning mellem prototypen og det udviklede system ud fra informantens synspunkt.

Formålet med den anden observation var at se om usabillity problemerne der blev opdaget fra den første observation var blevet overholdt og taget i betragtning til det nye systemdesign. Og derefter se om der var opstået nye usabillity problematikker i designet.

Observationen blev opstillet på akkurat samme måde som til den første, se 4.3, bortset fra at der ikke var en observatør kun en interviewer. Dette var begrundet på at det ikke var en nødvendighed efter første observation og derfor ikke i den anden. Observationen tog omkring 45 minutter fordelt over 10 opgaver med et konkluderende interview til sidst for at opsummere.

10.1 Problematikker

Informanten blev bedt om at informere intervieweren undervejs imens hun løste opgaverne og efter hver løst opgave blev informanten spurgt om hvad hun mente om elementerne hun lige havde arbejdet med. Dette gjorde vi for at informanten lettere kunne huske hvordan systemet så ud.

1. Problem med navigering, mest på 'Kursus'-siden

Informanten havde en række problemer undervejs til at navigere på nogle specifikke sider, mest på 'Kursus'-siden. Dette begrundede hun i at navigerende tekst

ikke var iøjnefaldende nok til at lettere kunne finde det, opgaverne bad om. Hun foreslog at ændre teksten til en fed skrifttype eller større skriftstørrelse.

Informanten blev stillet til opgave at finde sit Matematik-kursus og derefter finde sig selv på listen under kursussiden og de lærere der var ansvarlige for de lektioner hun var tilmeldt. Da hun først stødte på kursussiden på Matematik klikkede hun rundt for at se hvad de enkelte tabs indeholdte, som set på figur 10.1.'

På figuren kan der ses at teksten på de enkelte faner på kursussiden alle var en lille skriftstørrelse. Her foreslog hun også at skriften kunne gøres en smule større nogle steder i programmet, dog at ikke alle steder var svære at navigere. Dette betyder også at navigeringen var mindre intuitivt. Eksempelvis gik hun tilbage til skemaet for at se om lektionen på en given dato var aflyst, selvom det skulle være let at se inde på kursussiden.

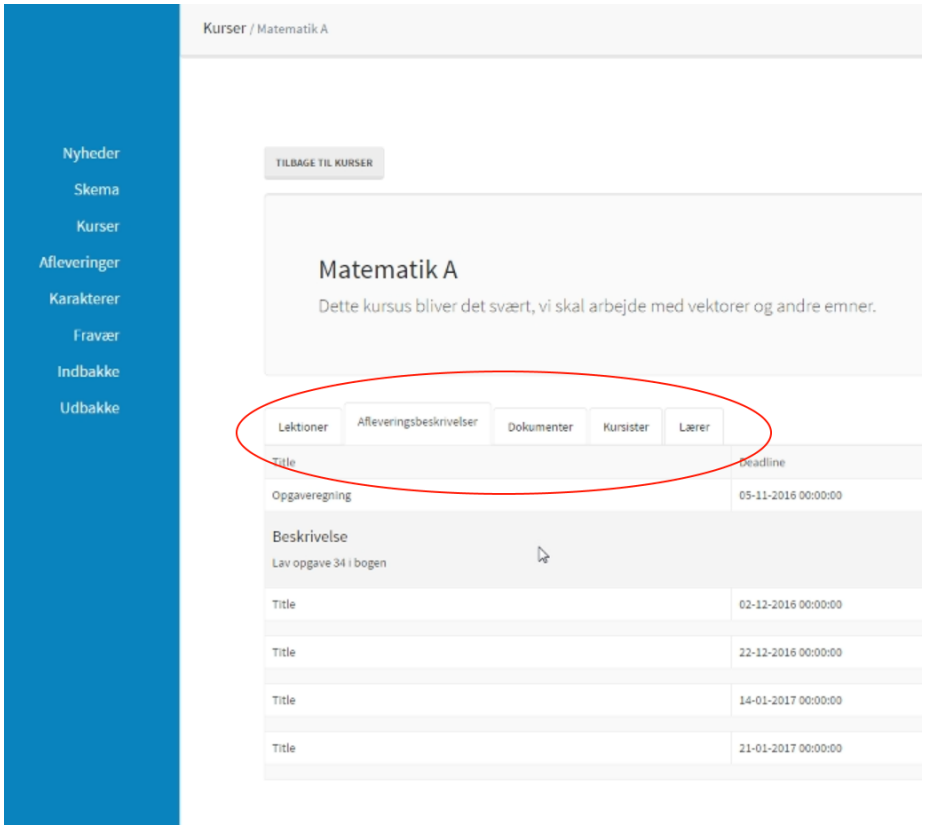


Figure 10.1: Kursussiden for Matematik. Informanten fandt det svært at finde rundt i den.

2. Problem med at aflevere en fil

Dette problem er et eksempel på en teknisk problematik som opstod under observationen. Der fandtes en lille række af fejl i systemet, der blev noteret og rettet i en videreudviklet version af systemet.

Derfor havde denne observation også til formål at orientere om tekniske fejl der ikke tidligere var opdaget. Fejlen med ikke at kunne aflevere en fil blev opdaget da informanten fik til opgave at aflevere en aflevering.

3. Fejl ved afsendelse af besked

En af opgaverne bedte informanten om at sende en besked til en lærer. Da hun gjorde dette glemte hun at sætte en modtager til beskeden.

Der skulle derfor komme en fejl-besked frem på skærmen, derimod blev beskeden lukket som om den var afsendt. Bag ved den lukkede besked lå fejl-beskeden, så den var skjult og brugeren var ikke informeret om om den var afsendt eller ikke.

4. Navne på lokation af dokumenter i stedet for deres rigtige navne

På Kursus-siderne for de individuelle kurser var tilhørende dokumenter ikke navngivet som de burde. I stedet for dokumenternes titler, blev dokumenternes titels lokation i programmet vist.

10.2 Opsummerende interview

Efter opgaverne var gennemgået blev der som i forrige observation lavet et interview for at opsummere på de ting informanten havde gjort opmærksom på til designet.

Hendes generelle mening om systemet var at det var en klar forbedring fra prototypen. Herunder er listet de design-valg hun noterede ud fra dette interview.

- Konsekvent design

Informanten synes designet var konsekvent og intuitivt. Elementer hun mente opfyldte disse beskrivelser var navigations-menuen på venstre side af applikationen, som også set på figur 10.1.

Derudover mente hun at designet ikke var for forskelligt på siderne og det hjalp på at gøre det intuitivt. Dette er også en markant forskel fra prototypen, hvori der var to navigationsbarer på hver sin side af applikationen. Hun

- Skemaet opdelt i farver

Informanten var glad for at skemaet var opdelt i forskellige farver efter hvilket kurus det var. Hun ville være glad for at se det andre steder i programmet også.

F.eks. rød advarsel ved for højt fravær.

- Funktionaliteter

Informanten havde et ønske om at der var mulighed for at aflevere en aflevering direkte igennem skemaet til en specifik lektion, hvis sådan en aflevering blev lavet. Derudover måske også bruge farver til at indikere resterende tid til at en aflevering skulle afleveres.

Yderemere hvis grafen på fraværssiden over gennemsnitlig fraværsprocent gav en helt præcis fraværsprocent helt ned til 2 decimaler.

10.3 Vurdering

Der er en klar forbedring fra prototypen og informanten mente at designet var blevet mere intuitivt og fulgte hendes ønsker hun havde givet til den første observation.

Hun gjorde opmærksom på nogle yderligere elementer der kunne gavne systemet til de formål hun ville bruge det til. Navigationen i applikationen var ikke perfekt, men hun mente den klart var blevet bedre på grund af at der kun var én menu med et klart konsekvent design. Observationen gav mulighed for at opdage små tekniske fejl i programmet til rettelse i den sidste iteration af projektet.

11

Diskussion

I vores diskussion har vi valgt at følge projektets opstilling kronologisk og diskutere de mest relevante elementer. Diskussionen vil bearbejde de vurderinger der blev taget undervejs i projektets udvikling.

Fremgangsmåden til problemanalysen virkede godt til vores formål, at kunne give informanten et bedre alternativ til hendes nuværende studieplatform. Siden informanten var vores kilde til de initierende problemer samt valg af systemdefinition, ville det have været bedre med flere informanter, af forskellige typer. Dette kunne have ledt til et bedre grundlag for projektet.

Ud fra problemanalysen udviklede vi en prototype. Prototypen blev lavet i Axure RP der er designet til at lave prototyper. Dette gjorde det nemt at designe, men også at standarden blev forhøjet. Da vi længere inde i projektet skulle udvikle en løsning til informantens krav, opstod der problemer med at opnå standarden der blev sat med prototypen, hvilket betød at vi måtte gå på kompromis med programmets standard.

Baseret på interviews med informanten, samt spørgskema til VUC&hf, blev der fundet frem til hvilke brugsmønstere de enkle aktører havde i systemet, samt hvilke funktionaliteter aktørerne havde behov for. Men eftersom vi kun havde én informant og vores spørgeskema kun tog udgangspunkt i xx deltagere, kan det diskuteres om den fundne data kan ses som tilstrækkeligt bred. En forbedring ville være at have sendt spørgskemaet ud til flere klasser, og eventuelt også andre VUC&hf afdelinger, for at se om der ville være forskel på afdelingerne i form af for eksempel undervisningsmetoder. Derudover ville nogle mere dybgående interviews med de forskellige aktører kunne have givet et bedre indsigt i aktørenes kompetencer og synspunkter, som vil kunne have øget udbyttet af analysen og herved givet en bedre forståelse af hvilke funktioner, samt vægten af de forskellige funktioner som var mest vigtige til at fuldfører en løsning, som de forskellige aktører generalt vil kunne benytte til deres behov, og opfylde kravene.

Kriterierne i afsnittet blev ikke ændret og var konsekvent afklaret efter informantens ønsker til de funktionaliteter hun helst ville se i systemet. Dette

gælder især for kriteriet om sikkerheden af systemet, hvorfra informanten gerne ville se at man ikke kunne blive automatisk logget ud af systemet. Dette er ikke et godt designvalg når man bearbejder vigtige informationer som faglige karakterer.

Til at designe brugergrænsefladen brugte vi metoder fra Benyon [1] i kombination med krav fra informanten. Denne sammensætning skabte en konflikt, eftersom at design bogen snakker om et design som gavner den generelle bruger og skaber mindst støj, hvor vores informant har egene præferencer som vil skabe mere støj for andre.

Det kunne muligvis have været bedre, hvis spørgeskemaet der blev sendt til VUC, inkluderede nogle skitser af forskellige design. Responsen på disse skitser ville gøre det muligt at designe en brugergrænseflade der passer til flere personer.

Til implementeringen af brugergrænsefladen blev der valgt at bruge ASP.Net platformen, da den er i stand til at koble C# kode sammen med HTML sider, men vi havde ikke på forhånd tilstrækkelig viden til at udnytte de egenskaber som ASP.Net tilbød. Konsekvensen af dette er at koden bag brugergrænsefladen indeholder mange gentagene instruktioner, hvilket gør koden svær at vedligeholde. Et alternativ til dette kunne have været at bruge WinForms, som er et .Net Framework der bruger C# kode. Men da dette ikke ment til at designe hjemmesider, besluttede vi at det ville være bedre at bruge ASP.Net.

Resultatet af vores udviklede systems observation var en forbedring fra prototypen, men der var stadig mulighed for forbedringer. Der var nogle tekniske fejl i programmet, som blev opdaget i testen da vi ikke selv havde lavet en prøve-test af programmet før observationen. Hvis dette var blevet gjort ville vi have haft en bedre observation af vores system, da det førte til forvirring og skabte en mindre pause under observationen.

Herefter foretog vi ændringer i programmet som var baseret på idéer fra informanten, disse gav hun i interviewet efter observationen. Dette betragtede vi som programmets tredje iteration.

Fejl og Mangler

I funktionskomponent afsnittet, se afsnit 7.2.2, blev der gennemgået de funktionaliteter der var vigtige i forhold til de undersøgelser vi havde foretaget med vores informant og deltagerne fra VUC&hf i Aalborg. Ikke alle funktionaliteterne endte med at blive implementeret, generelt til den administrative del. Funktionerne 'Opret kurser', 'Slet kurser', 'Tilmeld kursister' og 'Tilmeld lærere' blev ikke implementeret. Der blev primært fokuseret mere på kursistens brugergrænseflade og funktionaliteter da informanten var en kursist og skulle deltage i en observation på det udviklede system.

Beskeder i systemet kan ikke håndtere at afsende mere end én fil med samme navn. En måde vi kunne klare det på ville være at give hver fil der blive vedhæftet i en besked et unikt ID og med det ID kunne programmet kende

forskel på filerne og hente filerne på et senere tidspunkt gennem beskeden.

På fraværssiden, hvor kursisten kan se sit gennemsnitsfravær i en kurve, indlæser den siden meget langsomt. Grunden er at den bliver genereret af en metode dynamisk. En løsning kunne være at bruge 'SwiftPlot' i ASP.Net til at generere et plot, men der er uvished om det ville hjælpe.

Under designfasen blev der ikke taget forbehold for korrekt udregning af gennemsnitskarakter. Programmet ville forbedres, hvis vægt bliver tilføjet til karakterene i modellen.

12 Konklusion

Der tages udgangspunkt i problemformuleringen for at undersøge om målene deri er blevet løst, disse kan ses i afsnit 3. Udover disse mål fra problemformuleringen, vil vi også vurdere om vores system passer til vores systemdefinition se 2.3.

Målene fra vores problemformulering var følgende.

- *Hvordan kan der udarbejdes en ny studieplatform, og hvilke komponenter skal benyttes for simplere navigering, samt anvendeligt for nye studerende?*
- *Hvordan kan vi udvikle systemets funktionaliteter og grænseflade, så det passer til informantens præferencer?*

Det første mål i problemformuleringen, har vi opnået ved at benytte metaforer i systemets grænseflade, som gør det lettere for brugerne. Målet med at udarbejde en ny studieplatform, har vi opnået ved at bruge den feedback vi fik igennem interviews med informanten og spørgeskema til VUC. Det næste mål fra vores problemformulering opnåede vi ved at have konstant kontakt med vores informant, samt bruge hendes præferencer som kriterier for programmet, og iterativt få feedback på vores løsninger til disse kriterier.

Vi kan dog også konkludere at disse mål kun var lavet på baggrund af vores ene informant, så derfor kan vi konkludere at løsningen ikke passer til alle kursister på VUC, men kun vores informant.

Systemdefinitionen er delvist opfyldt, da der både er implementeret et informations- og kommunikation system. Samtidig kan det konkluderes at der mangler administrative værktøjer til sekretærer, dog skulle dette stadigvæk implementeres, men de andre elementer blev prioriteret højere. Derfor endte vi med et system som primært kun kan benyttes af kursister.

Det kan konkluderes at der er mangel af funktionalitet for lærere, da lærernes aspekt ikke blev inkluderet i ligesom en informant. Derfor indeholder systemet

ikke muligheder som at kunne redigere lektioner, lektier, afleveringer og fravær.

Derudover konkluderes det at systemet generelt kan benyttes af alle computere og smartphones, da det er en web-applikation. Hvilket også betyder at systemet er fleksibelt.

Samtidig er der mangel på informanter, da flere informanter ville give et større overblik over hvad systemet skulle indeholde, derudover ville vores usability test også blive bedre, da det vil skabe flere aspekter af vores system.

Dog er informantens krav blevet opfyldt, og vi endte derfor ud med et slutprodukt, som informanten var tilfreds med. Dette blev opfyldt, da vi igennem projektet har opfyldt informantens ønsker.

13

Perspektivering

Denne perspektivering omhandler de valg der er truffet igennem vores projekt, samt de fremtidige tilføjelser til projektet.

Det færdige program er udviklet efter en kursists behov, og skulle tilpasses til den enkelte kursist, og formindske de informationer kursisten ikke har behov for. Dermed skabe mindre forvirring, når det kommer til at skulle finde noget bestemt information. Dette kræver test på forskellige brugere, såsom kursister, lærere og sekretærer. I fremtiden ville det være relevant at kontakte VUC&hf, og test programmet med dem i en periode. På denne måde kan vi finde flere usability problemer, samt forbedre det.

I vores projekt var der taget brug af én informant, hvilket betød at vores usability test fandt færre problemer. I følge Jakob[5] kræver det mindst 5 informanter, disse vil finde størstedelen af problemerne. Samtidig vil dette også give flere synsvinkler af programmet, hvis der er anvendelse af flere forskellige type brugere.

Bibliography

[1] David Benyon. *Designing Interactive Systems, A comprehensive guide to HCI, UX and interaction design.* PEARSON EDUCATION LIM ITED, 3 edition, 2014.

[2] Cascade Delete. https://www.techonthenet.com/sql_server/foreign_keys/foreign_delete.php/, 2016. Accessed: 18-12-2016.

[3] Jan Stage Andreas Madsen Lars Mathiassen, Peter Axel. *Object Oriented analysis design.* Marko, 3 edition, 2000.

[4] What is MySQL. https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html, 2016. Accessed: 18-12-2016.

[5] Why You Only Need to Test with 5 Users. https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/, 2000. Accessed: 16-12-2016.

[6] Om VUC og hf Nordjylland. http://www.vucnordjylland.dk/main/om-vuc-hf-nordjylland, 2016. Accessed: 05-12-2016.

A

Appendix

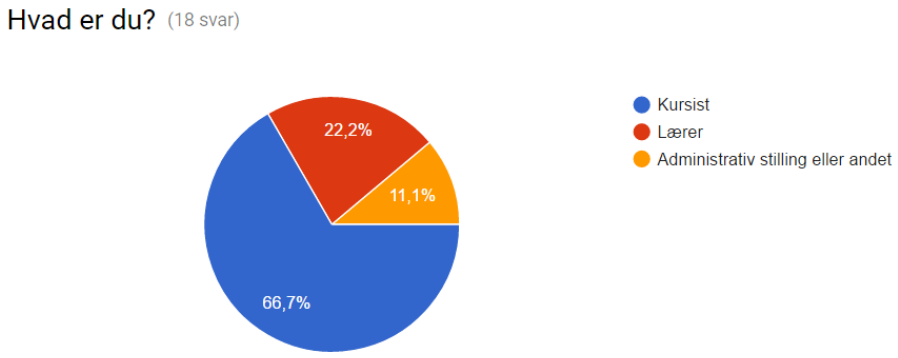


Figure A.1: Spørgeskema spørgsmål 1: Hvilken stilling har de 18 deltager.

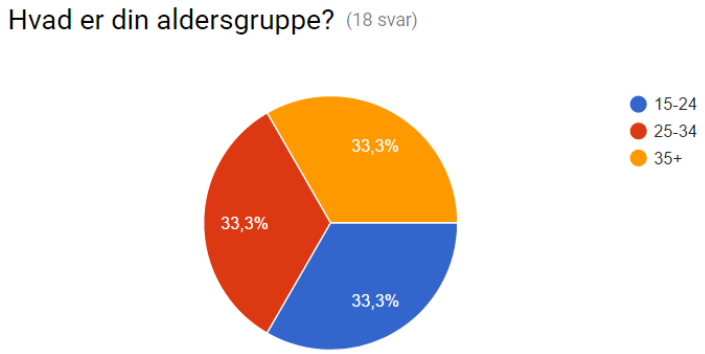


Figure A.2: Spørgeskema spørgsmål 2: Aldersfordeling mellem de 18 deltager.

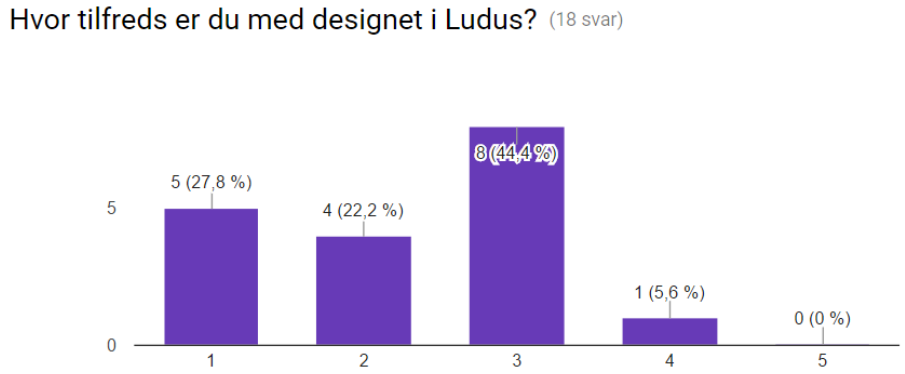


Figure A.3: Spørgeskema spørgsmål 3: De 18 deltagers tilfredshed med designet i Ludus.

Figure A.4: Spørgeskema spørgsmål 4: Hvad bruger de 18 deltager i Ludus.

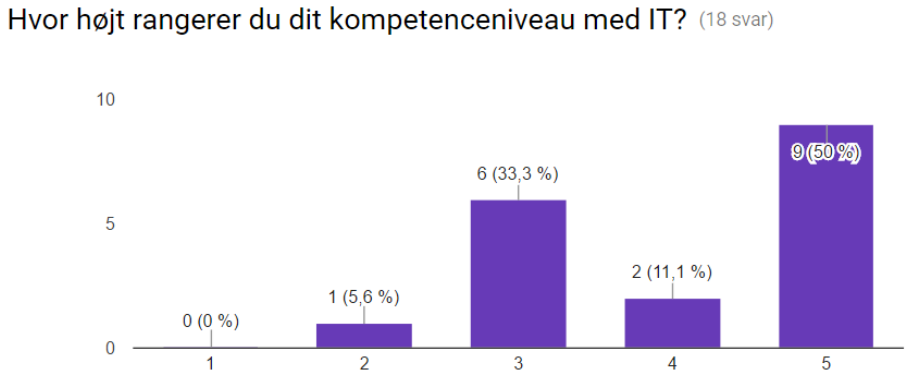


Figure A.5: Spørgeskema spørgsmål 5: De 18 deltagers IT kompetenseniveau.

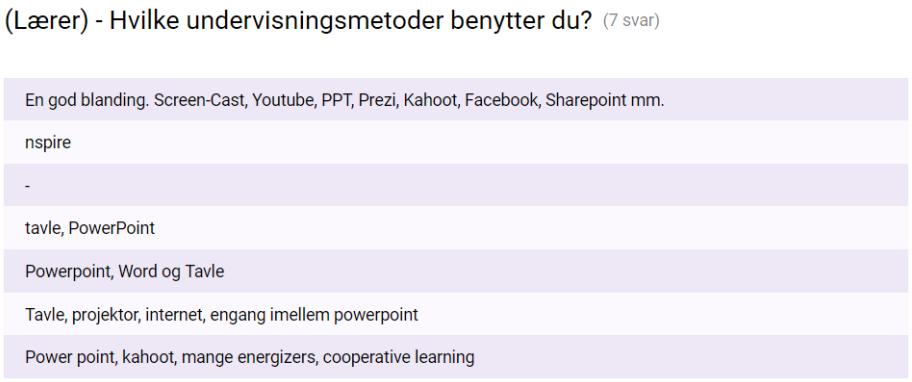


Figure A.6: Spørgeskema spørgsmål 6: Hvilke undervisningsmetoder benytter lærerne.

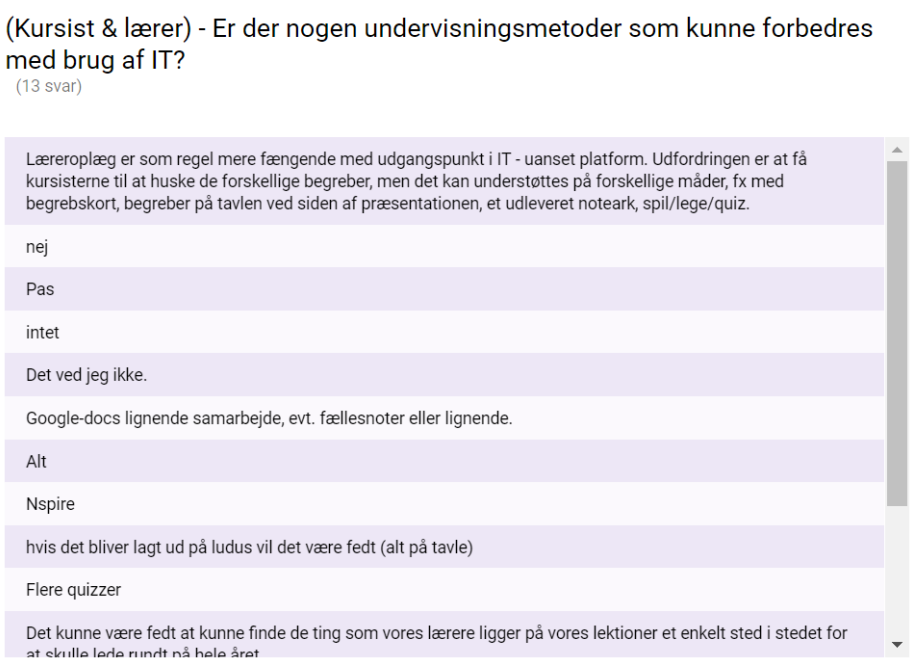


Figure A.7: Spørgeskema spørgsmål 7: Kursist og lærer vurderer, hvilke undervisningsmetoder kan forbedres med brug af IT.

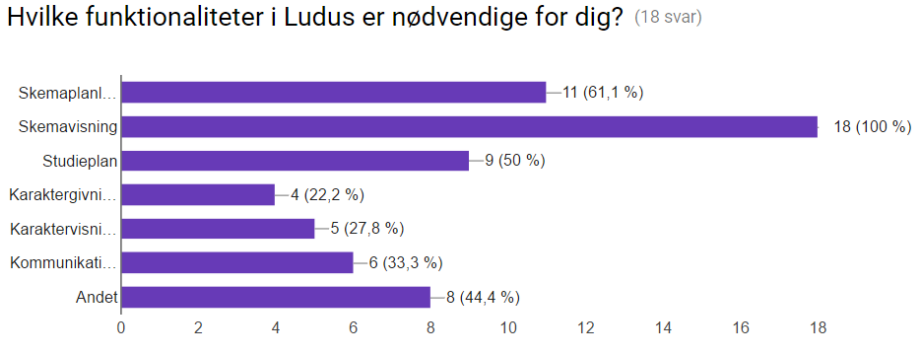


Figure A.8: Spørgeskema spørgsmål 8: De 18 deltager vurderer, hvilke funktionaliteter er nødvendige for dem i Ludus.

Hej Helle

Vi vil gerne takke dig for at hjælpe os med vores undersøgelse ved at deltage. Vi vil gerne gøre dig opmærksom på, at den undersøgelse går ud på at teste systemet, og ikke dig.

Før undersøgelsen vil vi gerne gøre dig opmærksom på et par ting. Det system du skal hjælpe os med at teste er en prototype. Testen forløber på den måde, at du får stillet et antal opgaver som du skal forsøge at løse ved hjælp af systemet. Opgaverne afspejler, hvordan du som kursist på skolen bruger systemet i din dagligdag.

Vi vil bede dig løse opgaverne på følgende måde. Først læser du hele teksten til opgaven højt. Derefter fortæller du os, hvad du forstår ved opgaven, og hvad du vil gøre for at løse den. Så går du i gang med løsningen på opgaven. Når du er i gang med opgaven vil vi gerne have at du tænker højt. Dette betyder, at du skal sige hvad du har tænkt dig at gøre, hvordan du har tænkt dig at gøre det. Vi ved godt det ikke er naturligt at sidde og snakke højt, så hvis du glemmer det, vil vi minde dig på det.

Hvis du får problemer under opgaveløsningen, så kan du godt spørge os. Vi vil derimod ikke hjælpe dig direkte, men nærmere få dig selv til at komme igang.

Når du er færdig med opgaven, vil vi bede dig sige det, således vi ikke er i tvivl.

Brugernavn: informant
Adgangskode: 1234

Opgave 1: Skema.
Du skal tjekke dit skema, for at finde ud af hvilke timer du skal have i løbet af dagen.

1. Se skema
2. Skift til skemaet i næste uge

Opgave 2: Aflevering.
Du har en aflevering i matematik, som skal afleveres 10/10 kl 23.59. Du har lavet denne aflevering, og skal nu afleverer den. Læreren har ikke oprettet et link gennem skemaet, og derfor kan du ikke aflevere igennem skemaet.

1. Aflever din aflevering.

Figure A.9: 1. Observationsspørgsmål til informanten, s. 1

Opgave 3: Lektier.

Du har fået lektier for til matematik modulet på onsdag.

3. Se dine lektier

Opgave 4: Besked.

Du er i tvivl om en opgave, og du vil gerne have afklaret med din lærer om hvad denne opgave går ud på.

1. Illustrerer hvordan du vil sende en besked til din lærer via. systemet.
2. Formuler et spørgsmål til læreren fordi du er i tvivl om en afleveringsdato, simuler at du sender den til læreren.

Opgave 5: Fravær.

Du har fået fravær for nogle moduler, hvor du ikke har mødt op. Du vil nu gerne ind og tjekke din fraværsprocent.

1. Illustrerer hvordan du vil tjekke dit fravær.
2. Formuler mundtligt hvor meget fravær du har i almindeligt fravær for året, både procent og modul.

Opgave 6: Karakter.

Du har fået angivet et ny gennemsnitlig karakter.

1. Gå ind og se din gennemsnitlige karakter.

Opgave 7: Nyheder.

Du har fået nogle nyheder, du vil gerne finde ud af, hvad der står i dem.

1. Gå ind i nyheder
2. Tjek nyheden “Spotkursus HF - Engelsk” og derefter “Jubilæumsfrokost”

Figure A.10: 1. Observationsspørgsmål til informanten, s. 2

X Interview efter observationen

I dette kapitel/afsnit/underafsnit har vi testet vores prototype med vores informant. Vi havde forberedt nogle opgaver til hende, som vi derefter kunne observere. Efter dette havde vi planlagt et struktureret interview omkring selve testen, for at kunne få så meget feedback som muligt. I slutningen af dette kapitel/afsnit/under afsnit samler vi op på den feedback vi har fået.

X.1 Spørgsmål efter observation

X.1.1 Generelt

- Hvad synes du generelt om systemet?
- Hvilke opgaver havde du problemer med?
- Hvilke opgaver syntes du var nemme?

X.1.2 Funktionalitet

- Var der nogle funktionaliteter i systemet du ikke kunne lide?
- Var der nogle funktionaliteter i systemet du godt kunne lide?
- Hvilke funktionaliteter synes du at der mangler i programmer?

X.1.3 Interface

- Hvad synes du generelt om interfacet?
- Var der noget ved interfacet du ikke kunne lide?
- Var der noget ved interfacet du godt kunne lide?
- Hvad synes du om farverne? Gjorde farverne dig opmærksom på visse områder i interfacet?
- Gjorde størrelse på kasser eller lignende dig mere opmærksom på visse områder i interfacet end andre?
- Hvad synes du om teksten i programmet, var det for småt eller ulæselig?

X.1.4 Vurdering

- I forhold til den studie platform som du benytter nu, hvordan vil du vurderer vores system i forhold til dette? Du kan også benytte en skala fra 1 til 5, hvor 5 er bedst, og 1 er dårligst.

X.2 Konklusion af prototype afprøvning

Figure A.11: 1. Observationsspørgsmål til informanten, s. 3

Observation

Hej Helle

Vi vil gerne takke dig for at hjælpe os med vores undersøgelse ved at deltage. Vi vil gerne gøre dig opmærksom på, at den undersøgelse går ud på at teste systemet, og ikke dig.

Før undersøgelsen vil vi gerne gøre dig opmærksom på et par ting. Det system du skal hjælpe os med at teste er et nyt system, som vi har udarbejdet. Testen forløber på den måde, at du får stillet et antal opgaver som du skal forsøge at løse ved hjælp af systemet. Opgaverne afspejler, hvordan du som kursist på skolen bruger systemet i din dagligdag.

Vi vil bede dig løse opgaverne på følgende måde. Først læser du hele teksten til opgaven højt. Derefter fortæller du os, hvad du forstår ved opgaven, og hvad du vil gøre for at løse den. Så går du i gang med løsningen på opgaven. Når du er i gang med opgaven vil vi gerne have at du tænker højt. Dette betyder, at du skal sige hvad du har tænkt dig at gøre, hvordan du har tænkt dig at gøre det. Vi ved godt det ikke er naturligt at sidde og snakke højt, så hvis du glemmer det, vil vi minde dig på det.

Hvis du får problemer under opgaveløsningen, så kan du godt spørge os. Vi vil derimod ikke hjælpe dig direkte, men nærmere få dig selv til at komme igang.

Når du er færdig med opgaven, vil vi bede dig sige det, således vi ikke er i tvivl.

Figure A.12: 2. Observationsspørgsmål til informanten, s. 1

Opgave 1: Log ind

Der er oprettet en ny bruger, som du skal logge dig ind med i systemet. Brugeroplysningerne til brugeren er angivet herunder.

Brugernavn: helle Adgangskode: 1234

1. Log ind med brugeroplysningerne.

Hvad synes du om 'log ind' siden?

Opgave 2: Nyheder

Du har modtaget en nyhed fra skolen. Du vil gerne finde ud af, hvad der står i denne nyhed.

1. Gå ind på siden "Nyheder".
2. Åben nyheden med titlen "Velkommen til vores afdeling i Aars!".
3. Læs brødteksten for nyheden.
4. Læs datoen for nyheden.

Hvad synes du om siden 'Nyheder'?

Opgave 3: Skema

Du vil gerne tjekke dit skema for at finde ud af, hvad den først lektion er om mandagen. Derudover vil du også gerne se nogle af de tidligere lektioner for forrige uge.

1. Gå ind på siden "Skema".
2. Find og læs faget på den første lektion om mandagen.
3. Find og læs faget på den sidste lektion om fredagen i uge 48.
4. Find den første lektion om onsdagen i uge 48.
    a. Find ud af hvem læreren er.
    b. Hvad siger beskrivelsen?
    c. Hent filen der hører til lektionen.

Hvad synes du om siden 'Skema'?

Figure A.13: 2. Observationsspørgsmål til informanten, s. 2

Opgave 4: Kurser

Du har fået en opgave for i faget Matematik. Du vil gerne finde denne lektion, hvor denne opgave hører til. Derudover vil du også gerne finde dig selv på listen, og finde ud af hvem lærer er for denne lektion.

1. Gå ind på siden "Kurser".
2. Åben siden for "Matematik A".
3. Find lektionen med datoen 05-09-2016.
    a. Læs beskrivelsen.
    b. Er lektionen aflyst?
4. Find afleveringsbeskrivelsen med titlen "Opgaveregning".
    a. Læs beskrivelsen.
    b. Har du afleveret?
5. Find dokumenterne til kurset.
    a. Åben det første dokument.
    b. Læs teksten i dokumentet.
6. Find dig selv på listen af kursister.
7. Find ud af hvem der er lærer for kurset.

Hvad synes du om siden 'Kurser'?

Opgave 5: Afleveringer

Du har fået en aflevering for i Historie omkring Japans Historie, som du skal til at afleverer.

1. Gå ind på siden "Afleveringer".
2. Find afleveringen med titlen "Japans Historie".
    a. Læs beskrivelsen.
    b. Hvad er fristen for opgaven?
3. Tryk på Aflever knappen
    a. Upload din aflevering
    b. Tilføj en kommentar (Du kan bare skrive 'Hav en god dag').
    c. Afleverer opgaven.
4. Læs hvad der står på skærmen.

Hvad synes du om siden 'Afleveringer'?

Figure A.14: 2. Observationsspørgsmål til informanten, s. 3

Opgave 6: Karakterer

Du har fået angivet en karakter i Kemi, som har ændret dit karaktergennemsnit. Udover karakteren har din lærer tilføjet en kommentar til denne karakter. Du vil gerne finde denne karakter med kommentar, og se dit nye karaktergennemsnit.

1. Gå ind i karakter.
2. Find karakteren i Kemi.
3. Find kommentaren til denne karakter i Kemi.
4. Find dit karaktergennemsnit.

Hvad synes du om siden 'Karakterer'?

Opgave 7: Fravær

Du har fået fravær i Dansk, og du vil gerne se, hvor mange moduler du har fraværende i faget Dansk. Derudover vil du også gerne tjekke dit fravær for året, både almindeligt og skriftligt.

1. Gå ind på siden "Fravær".
2. Hvor mange moduler har du været fraværende i faget 'Dansk A'?
3. Hvor mange procent almindeligt fravær har du samlet for året?
4. Hvor mange procent skriftligt fravær har du samlet for året?
5. Find grafen over almindeligt fravær.
    a. Hvor mange procent fravær har du på datoen 23-12-2016?

Hvad synes du om siden 'Fravær?

Figure A.15: 2. Observationsspørgsmål til informanten, s. 4

Opgave 8: Indbakke

Du har modtaget en ny besked i din indbakke fra Kent Poulsen, som du arbejder sammen med. Du vil gerne finde ud af, hvad der står i denne besked, og besvar Kent Poulsen på hans besked. Derudover vil du også gerne oprette en ny besked til din lærer, angående din aflevering.

1. Gå ind i Indbakke.
2. Læs beskeden fra Kent Poulsen.
3. Svar Kent Poulsen på hans besked.
    a. Skriv 'ja' eller 'nej' som teksten til beskeden.
    b. Send beskeden.
4. Opret en besked til din lærer.
    a. Formulerer et spørgsmål angående omkring din aflevering.
    b. Tilføje din afleverings fil til beskeden.
    c. Send beskeden.

Hvad synes du om siden 'Indbakke?

Opgave 9: Udbakke

Du har sendt en besked til Kent Poulsen, som du har arbejdet sammen med. I har fået et nyt gruppemedlem, Henry Hoj, som ikke har modtaget disse informationer. Du skal videresende beskeden du sendte i den forrige opgave til det nye gruppemedlem.

1. Gå ind i udbakke.
2. Find beskeden til Kent Poulsen.
3. Videresend beskeden til Henry Hoj.

Hvad synes du om siden 'Udbakke?

Opgave 10: Log ud

Du er færdig med at benytte systemet, og skal til at afslutte.

1. Find log ud knappen, og log derefter ud af systemet.

Figure A.16: 2. Observationsspørgsmål til informanten, s. 5

Opsummering

Generelt:

- Hvad synes du generelt om systemet?
- Hvilke opgaver havde du problemer med?
- Hvilke opgaver syntes du var nemme?

Funktionaliteter:

- Var der nogle funktionaliteter i systemet du ikke kunne lide?
- Var der nogle funktionaliteter i systemet du godt kunne lide?
- Hvilke funktionaliteter synes du at der mangler i programmer?

Interface:

- Hvad synes du generelt om interfacet?
- Var der noget ved interfacet du ikke kunne lide?
- Var der noget ved interfacet du godt kunne lide?
- Hvad synes du om farverne? Gjorde farverne dig opmærksom på visse områder i interfacet?
- Gjorde størrelse på kasser eller lignende dig mere opmærksom på visse områder i interfacet end andre?
- Hvad synes du om teksten i programmet, var det for småt eller ulæselig?

Vurdering:

- Hvad synes du om programmet i forhold til den prototype, som vi fremviste dig i forrige observation?

Figure A.17: 2. Observationsspørgsmål til informanten, s. 6