Chatprogram til industriel support

Oliver Smith

Kgs. Lyngby 2007
IMM-B.Eng.-2007-26

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

Resumé

Opgave:
JLI vision a/s ønsker at få udviklet chatprogram, for at kunne yde bedre support til sine kunder.

Løsning:
Der er blevet udviklet en supportløsning, som giver JLI mulighed for at yde support til sine kunder rundt omkring i verdenen. En række klienter kan, uafhængigt af hinanden, forbinde til en server, og derefter chatte indbyrdes. Programmet er opbygget med tynde klienter og en central server, der via en brugerdatabase holder styr på de enkelte klienters kontaktliste.

Konklusion:
Projektet er overordnet set vellykket, men der mangler stadigvæk noget finpudsning inden programmet er klart til at blive installeret hos kunderne.

Forord

Denne rapport er dokumentationen af mit eksamensprojekt *Chatprogram til industriel support*, som afslutningen på diplomingeniøruddannelsen, IT-retningen.

Projektet er udført for JLI vision a/s i samarbejde med instituttet IMM (Informatik og Matematik Modellering) på DTU.

Vedlagt rapporten er en cd med en eksekverbar version af det udviklede program, samt kildekoden til det. Uddybende beskrivelse af cd'en findes i bilag E.

Eksamensprojektet er udført i perioden 19. februar til 20. maj 2007.

Kgs. Lyngby, 21. maj 2007

Oliver Smith

Tak til. . .

Min familie som har støttet og hjulpet mig, ikke mindst med korrekturlæsning af rapporten.

Jan Damsgaard og de andre ansatte på JLI vision a/s som har hjulpet mig undervejs i projektet. Har haft stor glæde af deres ekspertise og gode ideer.

Finn Gustafsson der som vejleder på DTU har hjulpet mig med formalia og gode råd til rapportskrivningen.

Indhold

1 Introduktion **1**

1.1 Opbygning af rapport 1

1.2 Virksomheden 1

1.2.1 Introduktion til JLI vision a/s 1

1.2.2 Eksisterende program 2

1.3 Projektet 3

1.3.1 Formål 3

1.3.2 Problemformulering 4

1.3.3 Afgrænsning 5

1.3.4 Værktøjer 5

1.3.5 Metoder 5

2 Analyse **7**

2.1 Planlægning 7

2.1.1 Iterationsplan og milepæle 7

2.1.2 Risikostyring 8

2.1.2.1 Datatab 8

2.1.2.2 Sygdom 9

2.1.2.3 Urealistisk tidsplan 9
2.1.3 Revidering af tidsplan 9
2.2 Use case-model 9
2.2.1 Aktører 9
2.2.2 Use cases 10
2.2.3 System sekvens-diagrammer 10

3 Design **11**
3.1 Arkitektur 11
3.1.1 Brugergrænseflade-lag 11
3.1.2 Logik-lag 12
3.1.3 Hukommelses-lag 12

4 Implementering **15**
4.1 Klassediagram 15
4.2 Modulopbygnig 15
4.2.1 Grafisk brugergrænseflade 16
4.2.1.1 Brugergrænsefladens opbygning 17
4.2.1.2 Implementering af brugergrænsefladen 17
4.2.2 Chat-protokol 18
4.2.2.1 Protokolstruktur 19
4.2.2.2 Implementering af protokol 20
4.2.3 Database 20
4.3 Udvidelser 21

5 Test **23**
5.1 Teststrategi 23
5.2 Udførelse af løbende test 24

5.3 Udførelse af funktionel test 24

5.4 Testkonklusion 24

6 Konklusion **25**

6.1 Mulige forbedringer 25

A Use cases **27**

B System sekvens-diagrammer **33**

C Klassediagram **35**

D Testrapporter **39**

E Indhold på cd **53**

Figurer **55**

Tabeller **57**

Litteratur **59**

Kapitel 1

Introduktion

1.1 Opbygning af rapport

Rapporten er opbygget af fem dele:

- Indledende sider som resumé, forord, takkeside og indholdsfortegnelse.
- Kapitler som udgør selve rapporten.
- Bilag som indeholder supplerende materiale til selve rapporten.
- Litteraturliste og liste over figurer og tabeller.
- En vedlagt cd med kildekode og eksekverbar version af programmet.

Kapitel 1 er introduktion til rapporten med indledende oplysninger og problemformulering. Dette efterfølges af kapitel 2, der er analysen, og som definerer hvilke krav der er til opgaven. I kapitel 3 beskrives designet af programmet, og i kapitel 4 redegøres der for hvordan programmet er implementeret. Kapitel 5 indeholder afprøvninger af programmet og til sidst kommer konklusionen i kapitel 6.

1.2 Virksomheden

1.2.1 Introduktion til JLI vision a/s

JLI vision a/s er et mindre dansk ingeniørfirma med cirka 15 ansatte beliggende i Søborg. I over 20 år har de været toneangivende indenfor vision-branchen i såvel Danmark som i

udlandet.

Virksomheden blev grundlagt i 1985 under navnet Jørgen Læssøe Ingeniørfirma Aps, og fik sit nuværende navn i 1998.

Firmaet har specialiseret sig i at udvikle vision-systemer til industri og laboratorier. Systemer, hvor digitale kameraer tilsluttet computere med speciallavet software, foretager automatisk inspektion af produktionen, og dermed sikrer høj effektivitet, kvalitet og sikkerhed.

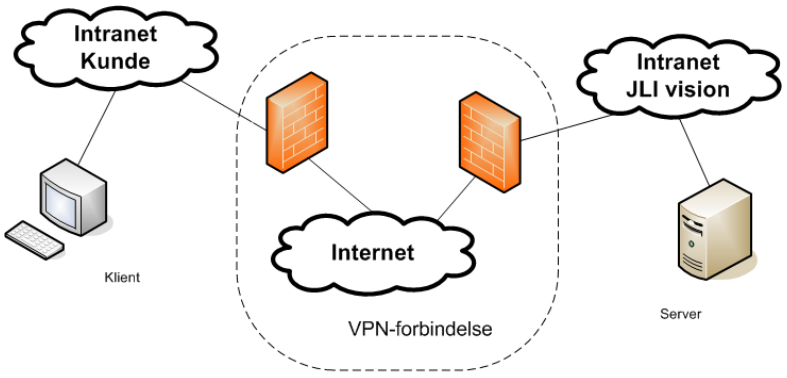
Gennem årene har JLI leveret over 1000 vision-systemer til hele verden til så forskellige brancher som glas, medicinal, emballage, plastik, stål og fødevarer.[10]

1.2.2 Eksisterende program

For nogle år siden opstod der i virksomheden et behov for at kunne yde support online ved hjælp af en eller anden form for chatprogram.

Til det formål blev der udviklet et simpelt chatprogram, der udfyldte de mest basale behov for en chatløsning.

Som det er illustreret på figur 1.1 består programmet af en server-del og en klient-del som taler sammen over en VPN-forbindelse.[1] En eller flere klienter kan forbinde til serveren, og sende beskeder direkte til den. De enkelte klienter derimod har ikke mulighed for at sende beskeder til hinanden. Desuden er der den uhensigtsmæssighed i programmet, at serveren sender sine beskeder ud til *alle* de forbundne klienter.



Figur 1.1: Illustration af det eksisterende program

Server-delen kræver en fast IP-adresse, og er derfor installeret på en servermaskine. Det betyder, at når en medarbejder skal bruge programmet til at kommunikere med en kunde, er han nødt til at gå ud i server-rummet og betjene serveren direkte, eller alternativt logge på maskinen via et fjernstyringsprogram som f.eks. VNC.[2] Installationen på serveren betyder også, at programmet ikke er velegnet til, at kunden henvender sig uopfordret. En uopfordret samtale via programmet vil blot åbne et vindue på serveren, og kunden har

[1] VPN: Virtual Private Network
[2] VNC: Virtual Network Computing

ingen garanti for at en medarbejder vil opdage henvendelsen inden for en rimelig tidsfrist.

Til et decideret supportformål er programmet ikke særligt hensigtsmæssigt. Derfor bruges chatprogrammet kun lejlighedsvist, og kun når det på forhånd er aftalt mellem begge parter.

1.3 Projektet

1.3.1 Formål

Der ønskes udviklet en løsning, som skal give virksomheden mulighed for at yde bedre support til sine kunder.

Traditionelle muligheder for at kunne yde support til kunderne:

- On-site support (personligt fremmøde)
- Telefonisk support
- Support via email

Alle disse mulighederne er i dag i brug, og fungerer for så vidt udemærket. Men der er en række situationer, hvor de alligevel ikke er tilstrækkelige:

On-site support er en temmelig dyr form for support. Det kræver, at en medarbejder tager ud til kunden, og da størstedelen af JLI's salg er til udlandet vil dette være en meget bekostelig affære. Det vil derfor være at foretrække at yde support hjemmefra, med mindre opgaven kun kan løses ved fysisk tilstedeværelse.

Telefonisk support er den mest udbredte form for support. Den er billig og nem at anvende, og derfor oftest den metode der tyes til. Men i en lang række tilfælde er telefonen dog ikke anvendelig. Da vision-anlæggene ofte står i meget støjfyldte produktionshaller, kan det være problematisk at skulle føre en samtale. Ydermere er der en del af de udenlandske kunder, f.eks. japanerne, der ikke er ligeså gode til engelsk i tale som i skrift, og som derfor ikke er glade for at skulle telefonere.

Support via email er heller ikke altid en god løsning. Ganske vist løser den problemerne med støj og manglende verbale engelskkundskaber, men til gengæld introduceres en række nye problemer. Først og fremmest skal man tænke på at brugeren af maskinerne (typisk en værkfører eller lign.) ofte ikke har adgang til almindelige pc'er med internetadgang. Heller ikke på vision-systemerne har vedkommende adgang til internettet, da maskinerne er låst til kun at kunne bruge vision-softwaren.

Der er altså en række situationer hvor de eksisterende muligheder for support ikke er tilstrækkelige, og det vil derfor være hensigtsmæssigt at kunne yde support på endnu en måde: Via et mere intelligent support-modul.

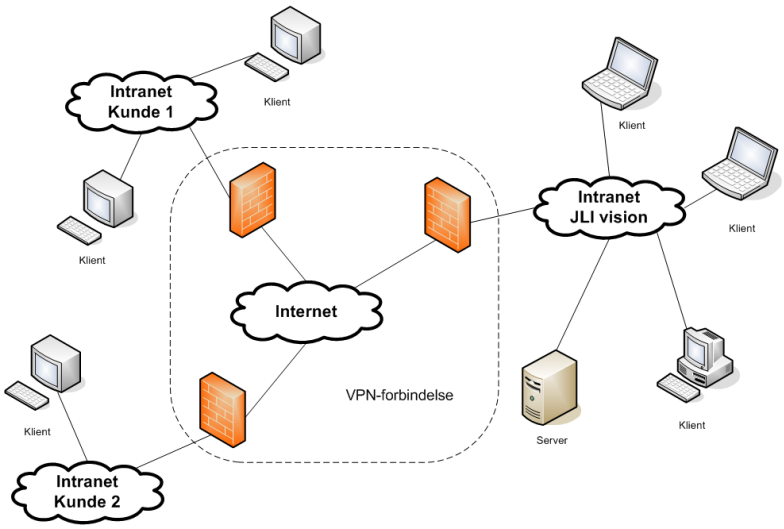
1.3.2 Problemformulering

Der ønskes et chatprogram der kan giver virksomheden mulighed for at yde bedre support til sine kunder. Det allerede eksisterende program (se afsnit 1.2.2) har en række mangler i forhold til at benytte det til at yde support, og derfor ønskes et nyt program udviklet.

For at opfylde behovet skal programmet kunne en række ting:

- Programmet skal være opbygget som et IM-program[3] med nogle klienter der kan kommunikere indbyrdes. Alle klienter kan starte og tage del i samtaler, og alle klienter kan i teorien skrive til hinanden. Dog vil en kontaktliste-funktion begrænse dette, og kun tillade kontakt mellem klienter på samme liste.
- Som bindeled mellem klienterne skal der være en server. Serveren skal modtage beskederne fra afsenderne, og sende dem videre til modtagerne. Inden klienterne kan kommunikere med hinanden kobler de sig op på serveren, som så tager sig af at holde styr på alle de tilsluttede klienter. En illustration kan ses på figur 1.2.
- Til forskel fra et normalt IM-program, hvor det er de enkelte brugere der selv styrer hvem de har på deres kontaktliste, skal det her konfigureres centralt i en database på serveren. Dette muliggører nem administration og vedligeholdelse af brugerdata.
- Da maskinerne hos kunderne typisk er koblet op til JLI via VPN-forbindelser, og derfor opfører sig som var de på lokalnet med serveren, er der ingen særlige krav til internetforbindelsen som f.eks. åbne porte i firewalls m.m.

Man kan endvidere udvide projektet med en historikfunktionalitet så man kan se tidligere samtaler, samt en funktionalitet til at håndtere offline-samtaler, f.eks. ved at sende en email eller SMS.



Figur 1.2: Illustration af det ønskede program

[3]IM: Instant Messaging

1.3.3 Afgrænsning

For at undgå at skulle begynde helt forfra med opgaven tilstræbes det at genanvende dele af det eksisterende program. Her tænkes især på den del af kommunikationen, som foregår på laveste niveau, dvs. opbygning af pakkerne mellem klient og server.

Ligeledes vil hjælpe-biblioteket commom.lib, som er et standardmodul i alt JLI's software, blive anvendt, f.eks. til at lette kommunikationen med registreringsdatabasen.

1.3.4 Værktøjer

Til at udvikle chatprogrammet er C++ valgt som programmeringssprog, og som IDE[4] *Borland C++ Builder 5*. Dette er også den primære udviklingsplatform hos JLI, og det giver en række fordele f.eks. i integreringen af deres hjælpe-biblioteket *common.lib*.

Som databaseløsning i serverdelen af programmet er valgt *Corel Paradox 7*, da den understøtter de muligheder der er brug for direkte i IDE'en.

Selve rapporten er skrevet i LaTeX med *MiKTeX 2.5* som distribution og *TeXnicCenter 7.01 beta 1* som editor.

Til tegning af iterationsplanen er benyttet Microsoft Excel 2003, og til de øvrige diagrammer og figurer benyttes *Umbrello UML Modeller 1.5.6* og *Microsoft Visio 2003*.

1.3.5 Metoder

Som udviklingsmetode er valgt UP,[5] der som en objektorienteret iterativ udviklingsmetode deler udviklingen op i en serie mindre bidder, og som for hvert gennemløb bygger flere funktioner på projektet. Dette er en god måde at udvikle på, da den gennem den iterative og trinvis voksende opbygning gør det let at styre programudviklingen.

Til at beskrive udviklingen af programmet i analyse- og designfasen er valgt UML,[6] der vel nærmest kan betragtes som de facto standard inden for OOAD.[7]

[4] IDE: Integrated Development Environment
[5] UP: Unified Process
[6] UML: Unified Modeling Language
[7] OOAD: Object-oriented analysis and design

Kapitel 2

Analyse

2.1 Planlægning

Til at styre planlægningen af projektet benyttes to metoder:

- Iterationsplan
- Risikoanalyse

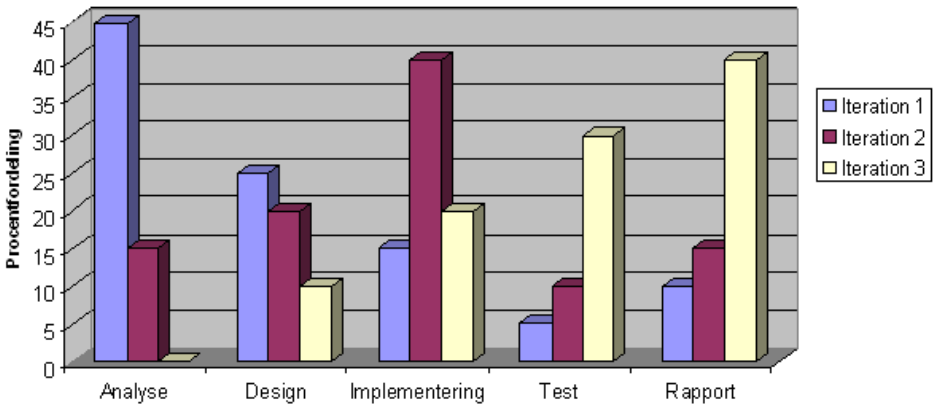
2.1.1 Iterationsplan og milepæle

Til den overordnede styring af tidsforbruget bruges en iterationsplan. Den består af en række iterationer eller gennemløb, som hver især er en slags små del-projekter, hvor der for hvert gennemløb bygges mere og mere på projektet. Det er en god måde at styre tidsforbruget på, da man nemt kan fjerne eller tilføje en iteration, hvis tidsplanen ændres.

Til dette projekt er antallet af iterationer sat til tre, da det er det, der skønnes sandsynligt at nå inden for den fastsatte tidsramme.

På figur 2.1 ses iterationsplanen, der giver et overblik over de enkelte fasers arbejdsfordelingen for hver af iterationerne.
Til iterationsplanen hører også nogle milepæle. Det er de datoer, der er sat som deadline for de enkelte iterationer.

Milepælene ses i tabel 2.1.



Figur 2.1: Iterationsplan

9. marts	Iteration 1 færdig
11. april	Iteration 2 færdig
16. maj	Iteration 3 færdig
21. maj	Aflevering af rapport

Tabel 2.1: Milepæle

2.1.2 Risikostyring

For at imødekomme uforudsete overraskelser, har jeg set på de risici der er forbundet med projektet:

- Datatab
- Sygdom
- Urealistisk tidsplan

2.1.2.1 Datatab

Den allerstørste risiko i projektet er nok at miste data undervejs. Da datatab kan opstå ud af den blå luft, og medføre at man bliver sat langt tilbage i projektet er det en meget vigtigt risikofaktor at have fokus på. For at komme dette til livs er der etableret en backup-rutine hvor alle filer kopieres til en usb-nøgle et par gange i løbet af dagen, samt løbende backup til andre pc'er samt jævnlig upload til CampusNet.

2.1.2.2 Sygdom

Med jævne mellem opstår sygdom, og sandsynligheden for at det sker under et længerevarende projekt som dette er absolut til stede. Er sygdommen af kortere varighed, og ligger den i starten af projektforløbet, vil det være muligt at indhente det forsømte efterfølgende. Men er sygdommen langvarig eller opstået i sidste øjeblik kan det blive nødvendigt at justere iterationsplanen eller ansøge om udsættelse. En egentlig forebyggelse mod sygdom er svær, men jeg har forsøgt at komme det i forkøbet med frisk luft og motion, samt fokus på god arbejdsstilling og regelmæssige pauser.

2.1.2.3 Urealistisk tidsplan

En ofte overset risikofaktor er, at den tidsplan man har lagt fra start ikke er realistisk. Eksempelvis kan det undervejs i projektet vise sig at dele af programmet er mere komplekst at udvikle end først antaget. Hvis dette er tilfældet vil man selvsagt få problemer med at overholde de forventede deadlines, og der er ikke andet for end at lægge en ny plan for projektet.

2.1.3 Revidering af tidsplan

Det vidste faktisk at være nyttigt at have en risiko-strategi: Cirka 5-6 uger inde i projektet begyndte jeg at få kraftige smerter i hænder, håndled og underarme, og efter en lægekonsultation fik jeg konstateret musearm og seneskedehindebetændelse. Jeg blev sat på smertestillende medicin og bedt om at sætte arbejdstempoet ned. Dette gjorde selvfølgelig at iterationsplanen ikke kunne holdes, og jeg ansøgte derfor om tre ugers udsættelse af projektet, hvilket blev bevilliget. Iterationsplanen og milepælene blev herefter revideret så de passede med den nye afleveringsdato.

2.2 Use case-model

Use case-modellen er en metode indenfor UML, der ved hjælp af diagrammer og beskrivende tekst giver et overblik over programmets ønskede funktionaliteter. Dette gør det muligt for de involverede i projektet at definere programmet, og da modellen er holdt på et abstrakt niveau er den også velegnet til ikke teknisk kyndige.

2.2.1 Aktører

Til use case-modellen hører en række aktører, der hver især repræsenterer en rolle som kan agere i programmet.

Der er defineret fire aktører til use case-modellen, som dækker de forskellige roller der vil blive brug for. Disse er listet i tabel 2.2.

Kunde:	En kunde der har brug for support og derfor benytter chatprogrammet.
Medarbejder:	Ansat hos JLI. Skal yde support til kunden.
Klient:	Klientversion af chatprogrammet.
Server:	Serverversion af chatprogrammet.

Tabel 2.2: Aktører

2.2.2 Use cases

Use cases er den skriftlige del af use case-modellen. De beskriver i ord hvordan forskellige dele af programmet skal fungere.

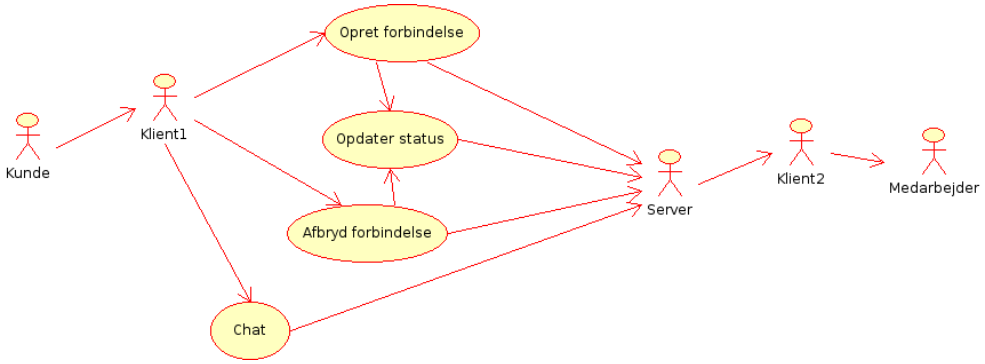
I tabel 2.3 ses de fire use cases, der beskriver de vigtigste funktionaliteter i programmet.

Opret forbindelse:	Etablerer forbindelse mellem klient og server, og henter klientens kontaktliste.
Afbryd forbindelse:	Afbryder klientens forbindelse til serveren.
Chat:	Skriver mellem to klienter.
Opdater status:	Opdaterer klienternes kontaktliste.

Tabel 2.3: Use cases

Da de enkelte use cases fylder en del, er de ikke medtaget i dette kapitel, men vedlagt i bilag A.

For at anskueliggøre sammenhængen mellem aktørerne og de forskellige use cases og give et samlet overblik, bruges use case-diagrammet (se figur 2.2). Det viser en grafisk fremstilling af hvordan aktører og use cases interagerer med hinanden.



Figur 2.2: Use case diagram

2.2.3 System sekvens-diagrammer

Et system sekvens-diagram er en grafisk fremstilling af hovedforløbet i en use case. De viser på overordnet systemniveau hvordan programmet fungerer.

System sekvens-diagrammerne er vedlagt i bilag B.

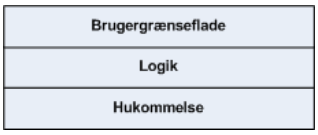
Kapitel 3

Design

3.1 Arkitektur

Programarkitekturen er bygget op efter den klassiske 3-lags-model (se figur 3.1), hvor man deler programmet op i et brugergrænseflade-lag, et logik-lag og et database-lag.

Opdelingen har til formål at adskille dele der ikke direkte har med hinanden at gøre, og derved gøre dem uafhængige af hinanden. Dette vil gøre det nemmere at udskifte dele af programmet f.eks. erstatte databasen med en anden.



Figur 3.1: 3-lags-model

3.1.1 Brugergrænseflade-lag

Programmet skal betjenes af en brugergrænseflade, f.eks. en GUI[1] med følgende krav:

- Der skal være et centralt placeret chatvindue til samtale mellem klienterne.
- En liste skal vise hvem ens kontakter er, og dermed hvem man kan skrive til.
- En funktionalitet skal gøre det muligt at ændre i programmets opsætning.

[1] GUI: Graphical User Interface

- Der skal indbygges en funktion der logger, hvad der sker i programmet, og viser det på en nem og overskuelig måde.
- Undervejs i programmet skal man nemt kunne danne sig overblik over programmets status, f.eks. om programmet kører i klient-tilstand eller om klienten er forbundet til serveren.
- Det skal tænkes ind i designet af brugergrænsefladen at vision-maskinerne typisk er tilsluttet skærme i størrelsen 15–17", hvilket kan give pladsproblemer, hvis der skal være flere åbne vinduer på en gang. Chatprogrammets størrelse bør derfor kunne tilpasses, f.eks. ved en mulighed for at skjule kontaktlisten, når den ikke bruges.
- Det skal være muligt at tilgå databasen på en nem måde så man gennem programmet (i server-tilstand) kan redigere i tabellerne.

3.1.2 Logik-lag

Logik-laget skal tage sig af kommunikationen mellem klienterne og serveren. Det skal fungere som klient-server-klient kommunikation, og ikke som P2P[2]. Da programmet skal styres fra en central server, og klinterne skal være tynde klinter vil dette også være den mest oplagte kommunikationsform.

Opbygningen med at alle beskeder skal forbi serveren har også den fordel at den muliggører email- og SMS-kommunikation. Visionen med at udvikle en email- og SMS-løsningen er at udfylde det kommunikationsmæssige tomrum der opstår når kunden ikke kan få fat i en medarbejder da denne er offline. Det skal derfor være muligt at sende vedkommende en email eller SMS. Men hvis de funktionaliteter skal indbygges i klienterne, er der en masse problematikker man skal tænke ind i udviklingen. Hvad gør man f.eks hvis kunden ikke kan huske medarbejderens email-adresse, og kan man være sikker på at kundens firewall tillader afsendelse af emails. Løsningen er at klienten behandler alle typer af beskeder (chat, email og SMS) på samme måde, og blot sender disse videre til serveren. Det vil så være op til serveren at ekspedere beskeden videre til det rigtige medie, f.eks. gennem en webserver eller SMS-gateway. På den måde lægges logikken centralt på serveren, og klient skal ikke bekymre sig om andet en at sende traditionelle beskeder.

3.1.3 Hukommelses-lag

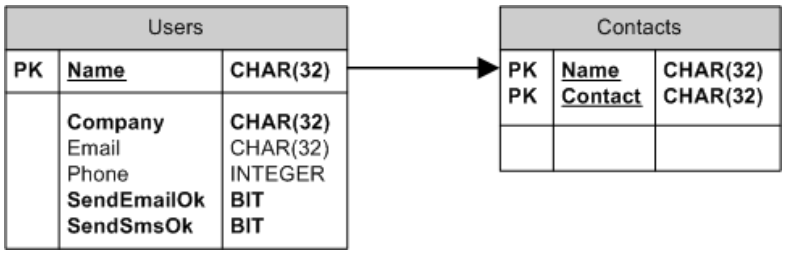
Til at gemme oplysninger om brugerne (klientnavne) og deres kontaktpersoner laves en database på serveren.

Databasen består af to tabeller: en med brugeroplysninger og en med kontaktlister, der kæder brugere sammen.

Databasen er bygget op så den overholder reglerne for normalisering af databaser, og der er valgt BCNF[3] som normaliseringsform. [11]

På figur 3.2 er databasens design skitseret i et E/R-diagram.[4]

[2] P2P: Peer to peer
[3] BCNF: Boyce-Codd Normal Form
[4] E/R-diagram: Entity-relationship diagram



Figur 3.2: E/R-diagram

Kapitel 4

Implementering

4.1 Klassediagram

Til at vise programmets struktur og opbygning bruges et klassediagram. På det kan man se programmets klasser og deres indbyrdes relationer, samt klassernes funktioner og variable.

Klassediagrammet er vedlagt i bilag C, men da det er for stort til at kunne være på en A4-side, har det været nødvendigt at formindske det. Dette kan gøre det vanskeligt at læse, og derfor er det vedlagt på cd'en i sin fulde størrelse.

4.2 Modulopbygnig

Som omtalt i afsnit 3.1 skal programmet opdeles i tre lag for let at kunne udskifte dele af koden. Det er gjort ved at opdele koden i flere cpp-filer (se tabel 4.1), som hver især repræsenterer et lag.

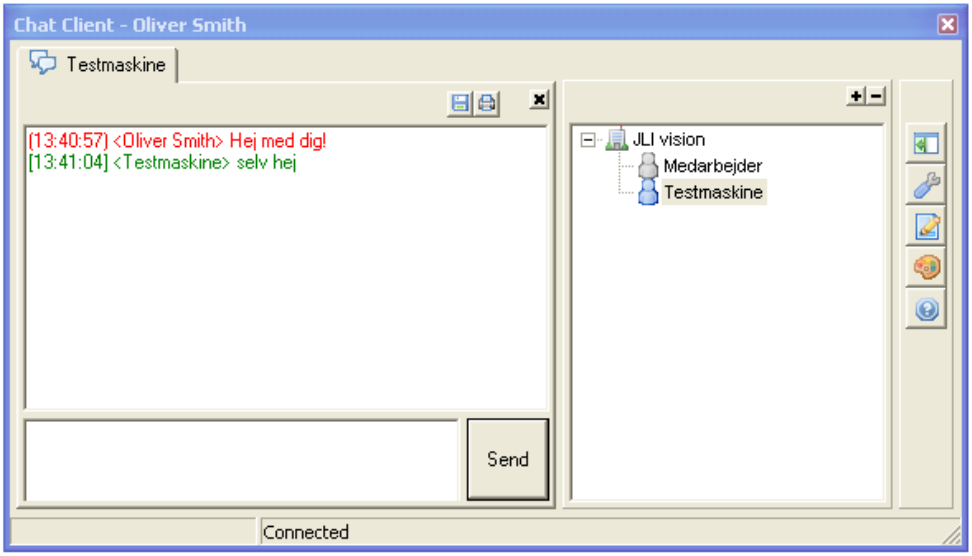
Brugergrænseflade-lag:	Grafisk brugergrænseflade (GUI.cpp)
Logik-lag:	Chat-protokol (Pack.cpp)
Hukommelses-lag:	Database (Database.cpp)

Tabel 4.1: Lagdeling af programmet

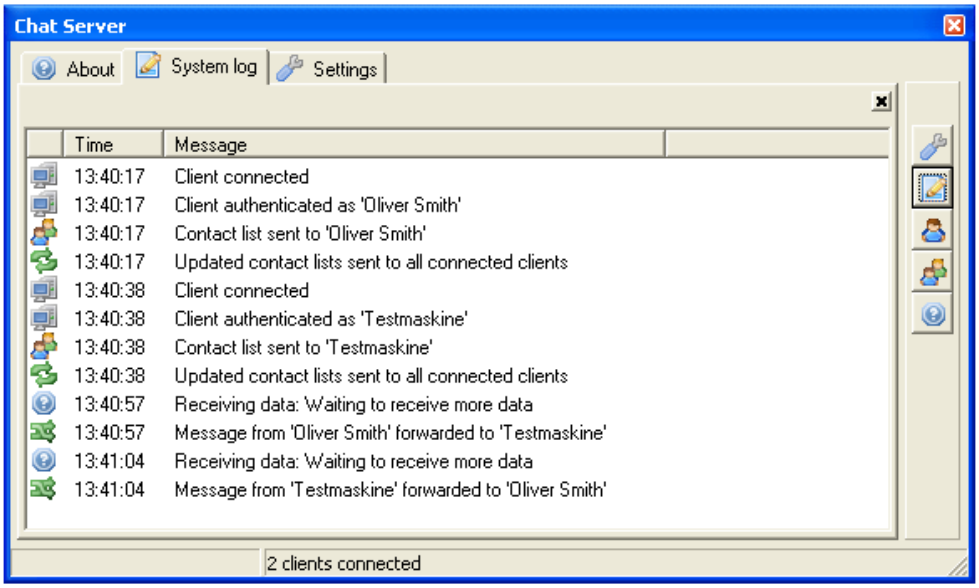
Man kunne argumentere for, at hvert lag skulle have været placeret i sin egen dll eller evt. lib, da dette ville mindske afhængigheden mellem filerne yderligere. Dette er dog bevidst fravalgt da det er prioriteret højere at programmet kun består af en enkelt exe-fil, med så få medfølgende dll'er som muligt.

4.2.1 Grafisk brugergrænseflade

Til at betjene programmet er der lavet en grafisk brugergrænseflade (se figur 4.1 og 4.2). Den er lavet, så det overordnede udseende er det samme, hvad enten programmet kører som server eller klient, men med den forskel at serveren ikke har kontaktliste og chat-faneblade, men til gengæld har to database-faneblade.



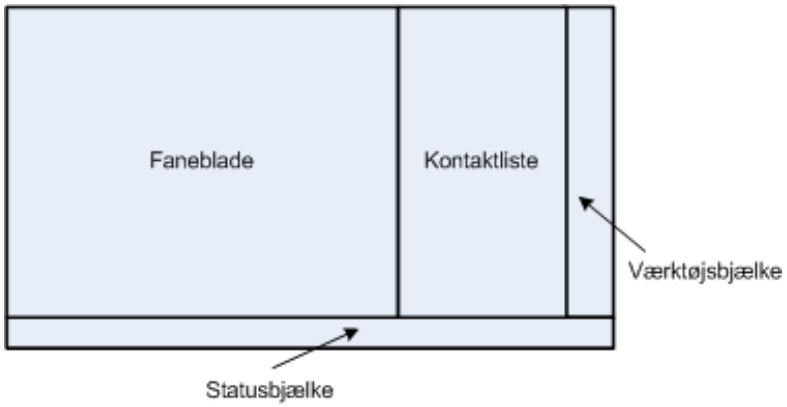
Figur 4.1: Klientens brugergrænseflade



Figur 4.2: Serverens brugergrænseflade

4.2.1.1 Brugergrænsefladens opbygning

Overordnet set består brugergrænsefladen af fire dele (se figur 4.3):



Figur 4.3: Skitse over den grafiske brugergrænseflade

Fanebladene er den del af programmet hvor samtalerne foregår, og opsætning af programmet foregår. Samtale-fanebladene vælges ved at vælge en online kontaktperson i kontaktlisten, mens de øvrige faneblade kaldes frem via værktøjsbjælken. Når et samtale-faneblad er valgt har man mulighed for at udskrive samtalen eller at gemme den som en tekstfil via de to knapper øverst på fanebladet. Fanebladene lukkes ved at klikke på krydset i højre hjørne.

I midten findes kontaktlisten, som viser alle de kontaktpersoner, der er tilknyttet klienten. Kontaktlisten findes ikke når programmet kører i server-tilstand. Når en kontaktperson er online kan denne vælges, og man kan starte en samtale. Oven over kontaktlisten er der to knapper til at *'folde'* kontaktlisten ud og ind.

Længst til højre er der en værktøjsbjælke med knapper til at betjene programmet. Der er syv forskellige knapper, som kan ses i tabel 4.2.

Nederst i programmet er der en statusbjælke, der fortæller hvor mange klienter der er online, dermed forbundet til serveren.

4.2.1.2 Implementering af brugergrænsefladen

For at illustrere implementeringen af brugergrænsefladen, gives der her et eksempel på et interessant sted i koden: Hvordan serveren holder styr på klienternes identitet.

Som udgangspunkt er det eneste serveren ved om en forbunden klient hvilken socket der er brugt til forbindelsen. Klienten sender derfor en datapakke af typen *PACK_logon_c* når forbindelsen etableres, og ud fra pakken kan serveren identificere klienten ved navns nævnelse. Serveren skal så lave en liste hvor den kæder klientens navn sammen med nummeret på socket-forbindelsen. Listen, kaldet client-map, implementeres som et C++–map (std:map) med en AnsiString og en int som værdier.

Vis/skjul kontaktliste	Viser eller skjuler kontaktlisten. Er kun tilgængelig i klient-tilstand.
Opsætning	Opsæt ip-adresse og portnummer til kommunikationen mellem klient og server.
Systemlog	Viser programmets log med oversigt over diverse hændelser samt klokkeslæt for disse.
Tilpas	Gør det muligt selv at vælge farve på teksten når der chattes.
Håndtér brugere	Tilføj, slet eller rediger brugerer i databasen. Er kun tilgængelig i server-tilstand.
Håndtér kontakter	Tilføj, slet eller rediger kontakter i databasen. Er kun tilgængelig i server-tilstand.
Om	Viser information om programmet.

Tabel 4.2: Knapper

På figur 4.4 ses det udsnit af koden hvor serveren modtager en PACK_logon_c-pakke og hvor listen client-map benyttes.

```
//Logon-besked til serveren
case PACK_base_c::LOGON:
  int index = get_connection_index(Socket);

  //Klientnavn er gyldigt
  if(DatabaseForm->CheckUser(src_id)){
    log(ClientPic, "Client authenticated as '" + src_id + "'");
    client_map.insert( process(src_id, index ) );
    //send en kontaktliste tilbage
    pack_ptr = new PACK_list_c(pack_ptr->SERVER_ID_STR, src_id.c_str());
    pack_list_ptr = dynamic_cast<PACK_list_c*>(pack_ptr);
    DatabaseForm->GetList(pack_list_ptr, src_id);
    log(ContactsPic, "Contact list sent to '" + src_id + "'");
    //Afsend beskeden
    pack_list_ptr->send(ServerSocket->Socket->Connections[index]);
    send_update_list();   //Opdaterer status på alle klienters lister
  }

  //Klientnavn er ikke gyldigt
  else{
    log(ErrorPic, "Unknown client name: '" + src_id + "'");
    //send en fejlmeddelse tilbage til klienten
    pack_ptr = new PACK_failure_c(pack_ptr->SERVER_ID_STR, src_id.c_str(), "Authentication failed");
    pack_ptr->send(ServerSocket->Socket->Connections[index]);      //Afsend beskeden
  }
  break;
```

Figur 4.4: Implementering af client-map

4.2.2 Chat-protokol

Til det oprindelige program (se afsnit 1.2.2) blev der udviklet en protokol til at sende datapakker mellem serveren og klienterne. I stedet for at begynde helt forfra blev denne brugt som grundlag for en ny protokol.

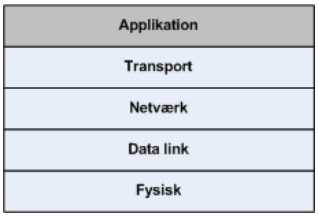
Det viste sig hurtigt, at den gamle protokol kunne en masse af de ting, der var anvendelige i det nye program, men at der, grundet de store forskelle mellem det nye og gamle program, også var en del protokollen ikke kunne. Den oprindelige protokol var designet således at en datapakke fra en klient automatisk var til serveren, mens en datapakke fra

serveren var til klienterne. Dette kunne ikke genbruges i det nye program, hvis f.eks. en chat-samtale mellem to klienter skulle oprettes. I det tilfælde sender den ene klient beskeden til serveren, som så skal sende den videre, hvilket jo kræver at serveren ved hvem modtageren er. Protokollen blev derfor restruktureret så både afsender og modtager kom med i datapakkens header.

Den originale protokol havde heller ikke ret mange forskellige datapakker man kunne sende, og da der var behov for at kunne sende fem forskellige, måtte der udvikles adskillige nye.

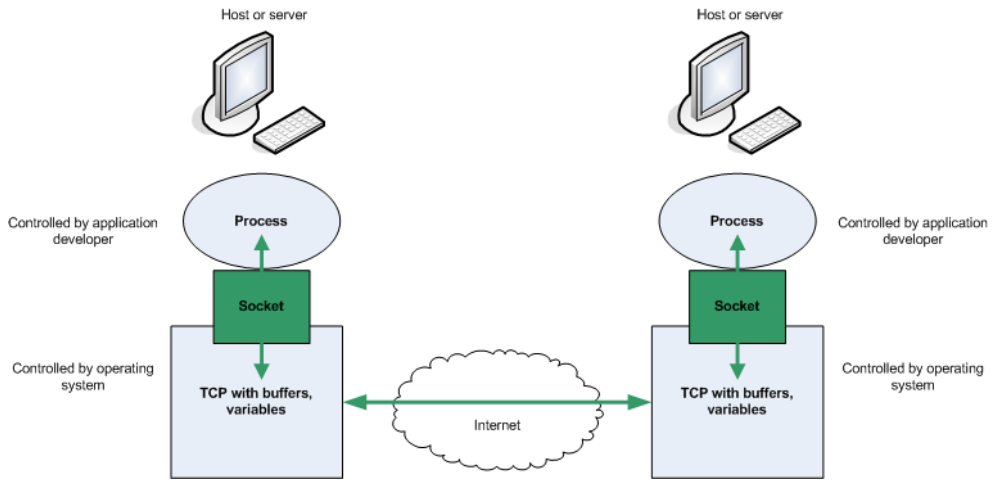
4.2.2.1 Protokolstruktur

Protokollen er en applikationslags-protokol, der benytter sockets til at kommunikere med([3], side 146–156). Applikationslaget er det øverste af lagene på protokol-stakken (se figur 4.5), og ved at holde kommunikationen i applikationslaget sørger Windows for den underliggende kommunikation, og man kan koncentrere sig om de centrale opgaver.



Figur 4.5: Protokol-stak

På figur 4.6 ses hvordan sockets bruges til kommunikation. (Figur kopieret fra [3], side 147).



Figur 4.6: Socket-kommunikation

4.2.2.2 Implementering af protokol

Protokollen (Pack.cpp) er opbygget omkring basis-klassen PACK_base_c. Klassen udgør rygraden i datapakkerne, og indeholder al den kode, der er universel for alle de forskellige typer af datapakker. Hver type af datapakke er implementeret i sin egen klasse, der så nedarver fra basis-klassen.

Se figur 4.7 for eksempel på implementering af en datapakke.

```
class PACK_chat_c : public PACK_base_c {

  public:

    static const int MAX_MSG_SIZE = 256;

    PACK_chat_c(const PACK_base_c &base) :  PACK_base_c(base)
    {
      data_ptr  = (data_s*)(PACK_base_c::get_buffer_ptr());
    }

    PACK_chat_c(
      const char *src_id_str,
      const char *des_id_str,
      const char *msg_str) :
      PACK_base_c(src_id_str, des_id_str, PACK_base_c::CHAT, sizeof(data_s) )
    {
      data_ptr = (data_s*)PACK_base_c::get_buffer_ptr();

      strncpy(data_ptr->msg_str, msg_str, sizeof(data_ptr->msg_str)-1);
      data_ptr->msg_str[sizeof(data_ptr->msg_str)-1] = 0;
    }

    const char *get_msg_str(void)  const          { return data_ptr->msg_str; }

  protected:
    struct data_s {
      char msg_str[MAX_MSG_SIZE];
    } *data_ptr;
};
```

Figur 4.7: Implementering at datapakken PACK_chat_c

Den vigtigste del af basis-klassen er headeren. Den udgør starten af hver datapakke, og indeholder informationer om datapakken f.eks. afsender, modtager, type og størrelse. Headeren, der er bygget som en struct (kaldet header_s), kan ses på figur 4.8.

```
struct header_s {
  char pw_str[8+9+1];                 //skal stå først
  int version;
  char src_id_str[ID_STR_SIZE];       //afsender id
  char des_id_str[ID_STR_SIZE];       //modtager id
  type_e type;
  int buffer_size;
  int checksum;                       //skal stå sidst
};
header_s header;
```

Figur 4.8: Implementering af headeren i basis-klassen

4.2.3 Database

Databasen er implementeret som en Corel Paradox database med to tabeller. Paradox er direkte understøttet i builderen, og den er nem at kommunikere med via de indbyggede

database-komponenter som TTable, TDataSource og TQuery.

Kommunikation mellem programmet og databasen varetages af Database.cpp, der på run-time-niveau opretter databaseforbindelsen hvis programmet køres i server-tilstand.

Den direkte understøttelse af Paradox i builderen gør, at almindelig behandling af data i databasen, så som opret, nedlæg og rediger, udføres direkte ved at knytte komponenterne TDBGrid og TDBNavigator til databasen. Til mere avancerede opslag benyttes TQuery til at sende en SQL-forespørgsel til databasen. Se eksempel på dette på figur 4.9.

```
bool TDatabaseForm::GetList(PACK_list_c *pack_list_ptr, AnsiString usr)
{

  //Laver liste over kontaktpersoner
  DatabaseForm->Query->Close();
  DatabaseForm->Query->SQL->Clear();
  DatabaseForm->Query->SQL->Add("SELECT Users.name, Users.company AS Name, Company FROM Users,
Contacts WHERE Users.Name = Contacts.Contact AND Contacts.Name = '" + usr + "'");
  DatabaseForm->Query->Open();

  DatabaseForm->Query->First();
  for(int i=0; i < DatabaseForm->Query->RecordCount; i++){
    AnsiString name = DatabaseForm->Query->FieldByName("Name")->AsString;
    AnsiString company = DatabaseForm->Query->FieldByName("Company")->AsString;
    pack_list_ptr->set_list_info(i, company.c_str(), name.c_str());

    DatabaseForm->Query->Next();
  }

  return true;

}
```

Figur 4.9: SQL-forespørgsel i database

4.3 Udvidelser

Det har været et ønske, at kunne udvide programmet med de tre ekstrafunktioner email, SMS og historik (se afsnit 1.3.2 og 3.1.2), hvis det viste sig at der blev tid til det.

Email- og SMS-funktionaliteten har der ikke været tid til at udvikle og der er heller ikke blevet implementeret en historik-funktionalitet i overensstemmelse med det der var ønsket. Men der er alligevel en simpel, men dog anvendelig, måde at gemme information om chatsamtaler: Når et chatvindue er åbent vil man, via de to knapper ovenover, kunne udskrive samtalen på papir eller gemme den som en tekstfil på computeren.

Kapitel 5

Test

5.1 Teststrategi

Til at teste programmet er der brugt to forskellige teststrategier:

- Løbende afprøvning af programmet under udviklingen.
- Funktionel test.

Den løbende afprøvning af programmet har til formål at fange de mest alvorlige fejl umiddelbart efter de opstår. Det forgår på den måde at hver gang der er blevet tilføjet nye funktionaliteter til programmet kontrolleres det om de virker efter hensigten. Dette gøres dels ved hjælp af builderens indbyggede debug-værktøj der gør det muligt at overvåge de enkelte variables værdier, indsætte break points m.m., og dels ved at indsætte dialogbokse strategiske steder i koden, og så bruge dem til at udskrive debug-information.

Den funktionelle test, som udføres efter programmets færdiggørelse, bruges til at kontrollere at programmet fungerer som ønsket. Der opsættes en række scenarier, hvor det angives, hvordan man skal teste programmet, og hvad programmets forventede opførsel er. Efter udførelsen af testen sammenlignes den faktiske opførsel med den forventede, og man kan se om programmet opfører sig som ønsket.

Der udfærdiges otte test-scenarier, og til hver af scenarierne laves en testrapport til at dokumentere testen.

5.2 Udførelse af løbende test

Den løbende test blev udført som beskrevet i afsnit 5.1.

Dette var en god måde at teste på, da det udryddede en stor del af fejlene før de nåede at give store problemer, og dermed gjorde det lettere at fejlrette.

Da den løbende test ikke er en test i traditionel forstand, men snarere en integreret del af selve udviklingen, er den ikke dokumenteret yderligere i denne rapport.

5.3 Udførelse af funktionel test

En god regel siger, at det er en anden person der skal teste et program, end ham der har udviklet det. Det vil også være at foretrække, men da testen blev foretaget på en helligdag hvor jeg sad derhjemme og skrev, var der ikke andre til at teste end mig selv. Det vurderes dog at dette ikke giver anledning til problemer.

Testrapporternes omfang er ret omfattende, og de derfor ikke medtaget i dette afsnit, men vedlagt i bilag D. I stedet er resultaterne opsummeret i tabel 5.1.

Test-id	Titel	Resultat	Uddybning af resultat
Test1	Indsæt data i tabellen Users.	✔	Alt gik godt.
Test2	Indsæt data i tabellen Contacts.	✔	Alt gik godt.
Test3	Rediger data i tabellen Users.	✔	Alt gik godt.
Test4	Rediger data i tabellen Contacts.	✔	Alt gik godt.
Test5	Slet data i tabellen Contacts.	✔	Alt gik godt.
Test6	Slet data i tabellen Users.	✔	Alt gik godt.
Test7	Test af oprettelse af forbindelser.	✔	Alt gik godt.
Test8	Test af afbrydelse af forbindelser.	✖	Programmet gik ned under testen.

Tabel 5.1: Opsummering af den funktionelle test

5.4 Testkonklusion

Den løbende test var en stor succes, og den funktionelle test er også i nogen grad forløbet tilfredsstillende. Test8 fejlede (se tabel D.8), hvilket selvfølgeligt er beklageligt, men på den anden side også understreger nødvendigheden af at teste.

Skal man se lidt kritisk på testen må man konstatere at otte testscenarier langt fra er nok til at komme rundt i *hele* programmet, og at man ikke derfor ikke kan bruge testen til at konkludere noget om programmet som helhed.

Kapitel 6

Konklusion

Der er blevet udviklet en supportløsning, som giver JLI mulighed for at yde support til sine kunder rundt omkring i verdenen. En række klienter kan, uafhængigt af hinanden, forbinde til en server, og derefter chatte indbyrdes. Programmet er opbygget med tynde klienter og en central server, der via en brugerdatabase holder styr på de enkelte klienters kontaktliste.

Testen viste at programmet i det store hele fungerer efter hensigten, men at der stadigvæk er fejl der skal rettes.

Konklusionen må være at projektet er vellykket og at der er udviklet et chatprogram der vil lette det daglige arbejde hos JLI. Men der mangler stadig væk en del rettelser og finpudsning inden programmet er klart til at blive installeret hos kunderne.

6.1 Mulige forbedringer

Skal man se på, hvordan programmet kan gøres bedre, er det mest oplagte selvfølgelig at se på testen og rette fejlen i test8 (se tabel D.8).

Dernæst kunne man se på de tre anbefalede udvidelser som aldrig blev implementeret:

- **Send email**
- **Send SMS**
- **Historik**

Skal man tænke i lidt større baner kan man også forstille sig følgende udvidelser, hvoraf flere dog må betegnes som eksotiske:

- **Voice-support.** Mulighed for tale med kunden via chatprogrammet.
- **Video-support.** Mulighed for tale med kunden og se denne samtidig. Vil lette fejlfindingen, da kunden via et trådløst web-kamera kan fotografere omgivelserne og vise dem til medarbejderen.
- **Konference.** Mulighed for at flere JLI medarbejdere kan chatte med den samme kunde samtidigt.
- **Offline-beskeder.** Sender man en besked til en kontaktperson der ikke er online gemmer serveren beskeden, og sender den til modtageren når denne kommer online.
- **Fjernstyring.** Nogle gange er det en fordel at se med på skærmen på vision-systemet, og til det formål kan man forestille sig at man via programmet kan starte *VNC* eller *Remote Desktop Connection*, og derved overtage styringen af vision-systemet ude hos kunden.
- **Afregningsmodul.** En mulighed for at logge supportens varighed, vil gøre det muligt at afregne kunden for den ydet support.
- **Ordbog.** En automatisk oversættelse af samtaler, som den på `http://babelfish.altavista.com`, vil lette supporten til udenlandske kunder betragteligt. Det er dog tvivlsomt om oversættelsen vil være af en kvalitet så den kan bruge til at yde teknisk support.
- **Stavekontrol.** En indbygget stavekontrol kan sikre at der ikke opstår misforståelser når der ydes support. Stavekontrol er også en forudsætning for at ordbogen overhovedet vil kunne virke. Open source-projektet *GNU Aspell* (`http://aspell.net`) er et godt bud på en at implementere en stavekontrol i sit program.

Bilag A

Use cases

ID:	§0001
Navn:	Opret forbindelse.
Forudsætninger:	Klient og server er startet og sat korrekt op.
Succeskriterier:	Klienten får forbindelse til serveren og opdaterer data.
Primær aktører:	Klient (timer).
Sekundær aktører:	Server.
Triggere:	En timer, der er sat op til automatisk at forsøge etablere forbindelse.
Formål:	At etablere forbindelse mellem klient og server, og hente klientens kontaktliste.
Hovedforløb:	1. Klienten kontrollerer at forbindelsen ikke allerede er etableret. 2. Klienten etablerer forbindelse til serveren. 3. Klienten sender en meddelelse med sit brugernavn til serveren. 4. Serveren kontrollerer i databasen at brugernavnet er gyldigt. 5. Serveren sender en meddelelse til klienten om at denne er godkendt som bruger. 6. Serveren opdaterer sin interne liste over forbundne klienter. 7. Serveren henter oplysninger om klientens kontaktliste i databasen. 8. Serveren og sender oplysninger om kontaktlisten til klienten.
Alternative forløb:	1a. Klienten er allerede forbundet til serveren. 2a. Klienten foretager sig ikke yderligere. 4b. Brugernavnet er ikke gyldigt, og serveren sender en meddelelse til klienten om dette. 5b. Klienten foretager sig ikke yderligere.
Prioritet:	Høj.
Hyppighed:	Sjælden - da der normalt er forbindelse.

Tabel A.1: Use case – Opret forbindelse

ID:	§0002
Navn:	Afbryd forbindelse.
Forudsætninger:	§0001 skal være vel gennemført.
Succeskriterier:	Forbindelsen bliver afbrudt.
Primær aktører:	Kunde, klient
Sekundær aktører:	Server.
Triggere:	Kunde vil lukke klient ned.
Formål:	At afbryde klientens forbindelse til server.
Hovedforløb:	1. Kunde lukker klient. 2. Klienten fortæller serveren at den afbryder forbindelsen. 3. Serveren opdaterer sin interne liste over forbundne klienter.
Alternative forløb:	*Intet alternativt forløb.*
Prioritet:	Lav.
Hyppighed:	Sjælden - da der normalt er forbindelse.

Tabel A.2: Use case – Afbryd forbindelse

ID:	§0003
Navn:	Chat
Forudsætninger:	§0001 skal være gennemført succesfuldt.
Succeskriterier:	Kunden får fat i en medarbejder og får support.
Primær aktører:	Kunde, medarbejder, to klienter.
Sekundær aktører:	Server.
Triggere:	Kunden ønsker support.
Formål:	At få fat i en medarbejder der kan yde support.

Hovedforløb:

1. Kunden vælger en medarbejder på sin kontaktliste.
2. Medarbejderen er online, og kunden skriver en besked til ham.
3. Beskeden sendes til serveren.
4. Serveren modtager beskeden og sender den videre til medarbejderens klient.
5. Medarbejderens klient modtager beskeden og viser den i et chatvindue.
6. Medarbejderen svarer på beskeden.
7. Medarbejderens klient sender beskeden til serveren.
8. (...)

Fortsættes på næste side

Alternative forløb:

1a. Medarbejderen er ikke på kontaktlisten

2a. Kunden må finde en anden medarbejder at skrive til eller alternativt kontakte firmaet på anden vis.

2b. Medarbejderen er ikke online og kunden skriver en email i stedet.

3b. Emailen sendes til serveren.

4b. Serveren modtager meddelelsen.

5b. Serveren finder medarbejderens email-adresse i databasen, og sender en email med den modtagne besked.

6b. Serveren finder medarbejderens mobiltelefonnummer i databasen, og sender en SMS der fortæller, at der er kommet en email.

Prioritet:	Høj.
Hyppighed:	Middel.

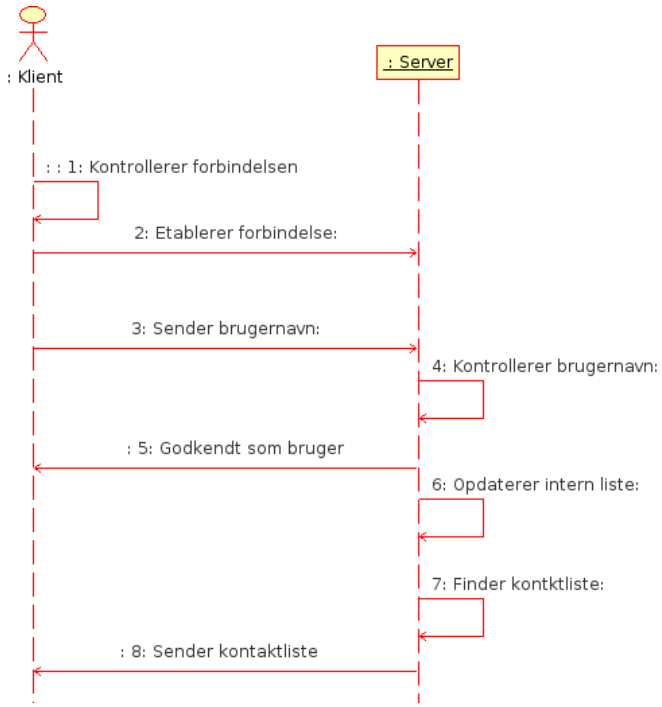
Tabel A.3: Use case – Chat

ID:	§0004
Navn:	Opdater status.
Forudsætninger:	Server og klient (evt. flere) er startet og sat korrekt op.
Succeskriterier:	Status bliver opdateret.
Primær aktører:	Klient (evt. flere)
Sekundær aktører:	Server
Triggere:	At §0001 eller §0002 gennemføres.
Formål:	At opdatere klienternes kontaktlister.
Hovedforløb:	1. Klient opretter eller afbryder forbindelse til serveren. 2. Serveren opdaterer sin intern liste de klienter der er online. 3. Sender listen til klienter. 4. Klienterne modtager listen, og opdaterer status på kontaktpersonerne på kontaktlisten.
Alternative forløb:	*Intet alternativt forløb.*
Prioritet:	Lav.
Hyppighed:	Middel.

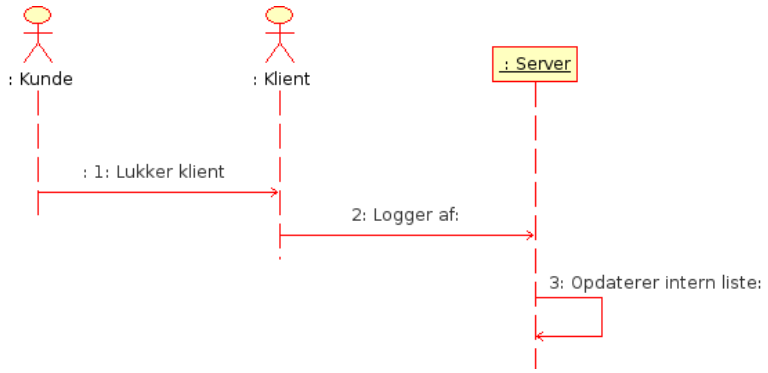
Tabel A.4: Use case – Opdater status

Bilag B

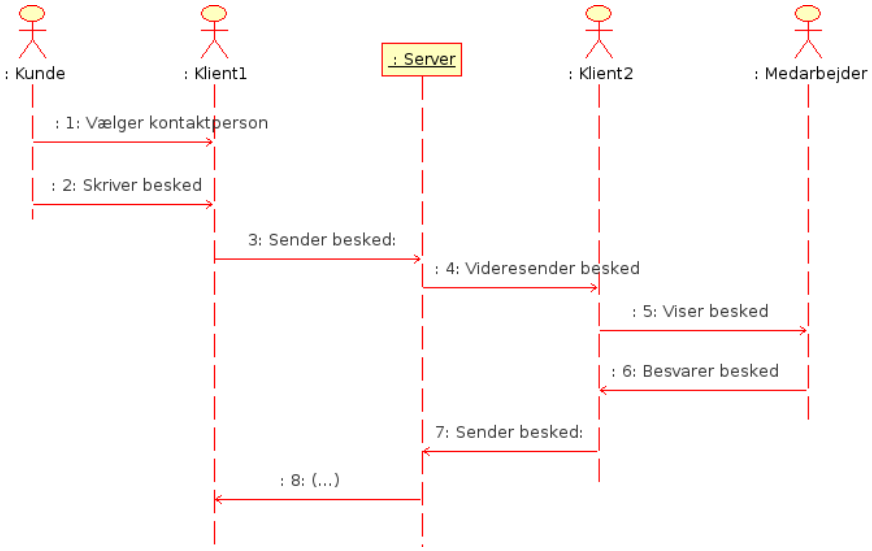
System sekvens-diagrammer



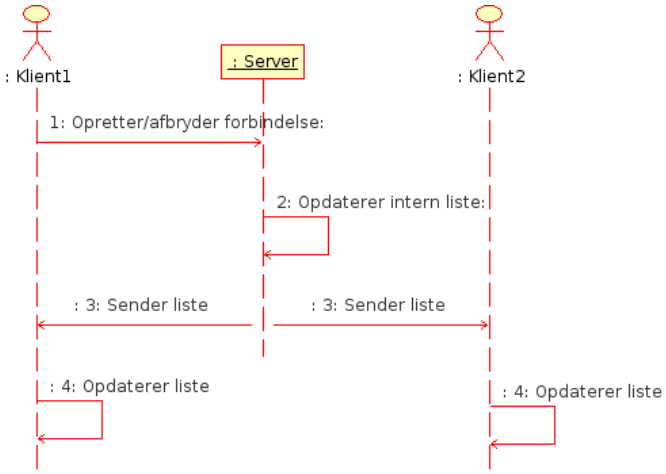
Figur B.1: System sekvens-diagram - Opret forbindelse



Figur B.2: System sekvens-diagram - Afbryd forbindelse



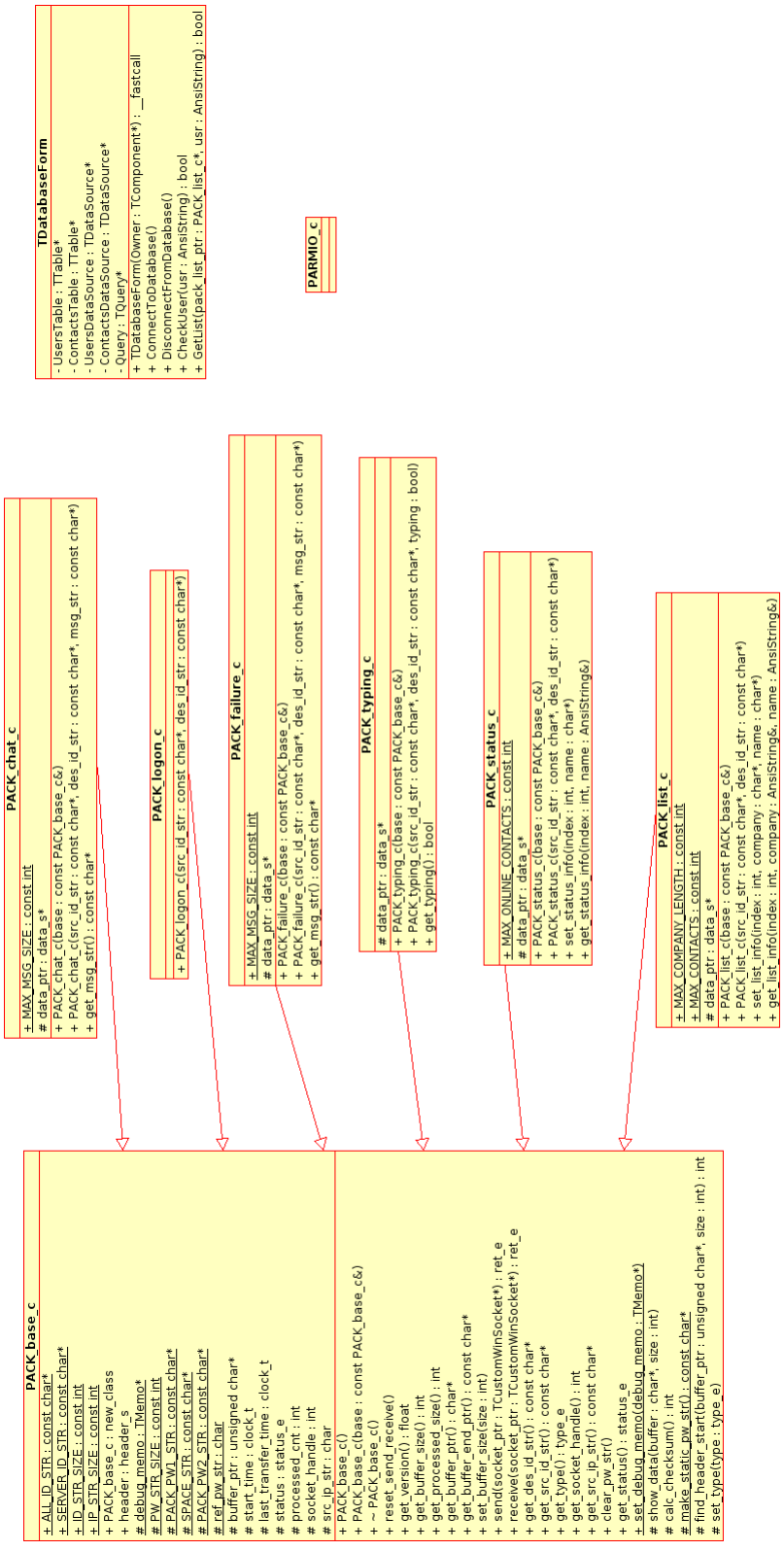
Figur B.3: System sekvens-diagram - Chat



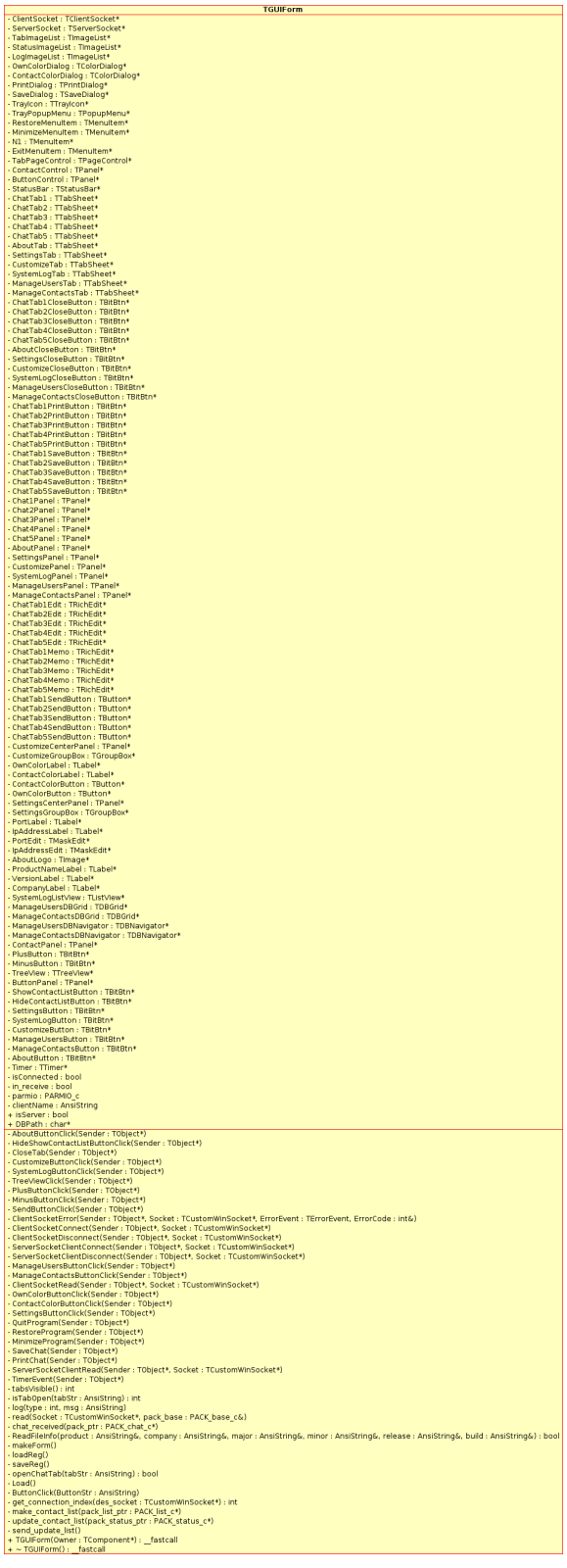
Figur B.4: System sekvens-diagram - Opdater status

Bilag C

Klassediagram



Figur C.1: Klassediagram – del 1



Figur C.2: Klassediagram – del 2

Bilag D

Testrapporter

Test-id: Test1 **Titel:** Indsæt data i tabellen Users.

Formål: At teste om alle data der indtastes i tabellen *Users* overføres korrekt til databasen.

Forudsætning: At server og database er korrekt installeret.
At tabellen er tom, så der ikke ligger data, der forstyrrer testen.

Udførelse:

- Serveren startes og fanebladet til administration af brugere vælges ved at klikke på knappen *Manage users* i højre side af programmet.
- Data fra kolonnen *forventet* indsættes i tabellen og gemmes.
- Serveren lukkes ned, og startes igen.
- Fanebladet til administration af brugere vælges igen.
- Værdierne i tabellen sammenlignes med de der blev indtastet.

Acceptkrav: Der accepteres ikke nogen forskelle mellem det forventede og det aktuelle.

Kolonne	Forventet	Aktuelt	Status
Name	Testbruger1	Testbruger1	✔
Company	Testfirma1	Testfirma1	✔
Email	br1@firma1.dk	br1@firma1.dk	✔
Phone	12345678	12345678	✔
SendEmailOk	True	True	✔
SendSmsOk	True	True	✔
Name	Testbruger2	Testbruger2	✔
Company	Testfirma1	Testfirma1	✔
Email	br2@firma1.dk	br2@firma1.dk	✔
Phone	23456789	23456789	✔
SendEmailOk	False	False	✔
SendSmsOk	True	True	✔
Name	Testbruger3	Testbruger3	✔
Company	Testfirma2	Testfirma2	✔
Email	br3@firma2.dk	br3@firma2.dk	✔
Phone	34567890	34567890	✔
SendEmailOk	False	False	✔
SendSmsOk	False	False	✔

Kommentar: *Ingen.*

Udførelse: **Dato:** 17. maj 2007 **Testet af:** Oliver Smith
Godkendelse: **Dato:** 18. maj 2007 **Godkendt af:** Oliver Smith

Tabel D.1: Indsæt data i tabellen Users

Test-id: Test2	**Titel:** Indsæt data i tabellen Contacts
Formål:	At teste om alle data der indtastes i tabellen *Contacts* overføres korrekt til databasen.
Forudsætning: At alle de foregående test er gennemført tilfredsstillende.	At server og database er korrekt installeret. At tabellen er tom, så der ikke ligger data der forstyrrer testen.

Udførelse:

- Serveren startes og fanebladet til administration af kontakter vælges ved at klikke på knappen *Manage contacts* i højre side af programmet.
- Data fra kolonnen *forventet* indsættes i tabellen og gemmes.
- Serveren lukkes ned, og startes igen.
- Fanebladet til kontakter af brugere vælges igen.
- Værdierne i tabellen sammenlignes med de der blev indtastet.

Acceptkrav: Der accepteres ikke nogen forskelle mellem det forventede og det aktuelle.

Kolonne	Forventet	Aktuelt	Status
ID	Testbruger1	Testbruger1	✔
Contact	Testbruger2	Testbruger2	✔
ID	Testbruger1	Testbruger1	✔
Contact	Testbruger3	Testbruger3	✔
ID	Testbruger2	Testbruger2	✔
Contact	Testbruger1	Testbruger1	✔
ID	Testbruger2	Testbruger2	✔
Contact	Testbruger3	Testbruger3	✔
ID	Testbruger3	Testbruger3	✔
Contact	Testbruger1	Testbruger1	✔
ID	Testbruger3	Testbruger3	✔
Contact	Testbruger2	Testbruger2	✔

Kommentar: *Ingen.*

Udførelse: **Dato:** 17. maj 2007 **Testet af:** Oliver Smith
Godkendelse: **Dato:** 18. maj 2007 **Godkendt af:** Oliver Smith

Tabel D.2: Indsæt data i tabellen Contacts

Test-id: Test3	**Titel:** Rediger data i tabellen Users.
Formål:	At teste om alle data der redigeres i tabellen *Users* overføres korrekt til databasen.
Forudsætning:	At server og database er korrekt installeret. At alle de foregående test er gennemført tilfredsstillende.

Udførelse:

- Serveren startes og fanebladet til administration af brugere vælges ved at klikke på knappen *Manage users* i højre side af programmet.
- Testbruger1 ændres til Testbruger10 og Testfirma1 ændres til Testfirma10.
- Testbruger2 ændres til Testbruger20 og Testfirma1 ændres til Testfirma10.
- Testbruger3 ændres til Testbruger30 og Testfirma2 ændres til Testfirma20.
- Serveren lukkes ned, og startes igen.
- Fanebladet til administration af brugere vælges igen.
- Værdierne i tabellen sammenlignes med de der blev indtastet.

Acceptkrav: Der accepteres ikke nogen forskelle mellem det forventede og det aktuelle.

Kolonne	Forventet	Aktuelt	Status
Name	Testbruger10	Testbruger10	✔
Company	Testfirma10	Testfirma10	✔
Email	br1@firma1.dk	br1@firma1.dk	✔
Phone	12345678	12345678	✔
SendEmailOk	True	True	✔
SendSmsOk	True	True	✔
Name	Testbruger20	Testbruger20	✔
Company	Testfirma10	Testfirma10	✔
Email	br2@firma1.dk	br2@firma1.dk	✔
Phone	23456789	23456789	✔
SendEmailOk	False	False	✔
SendSmsOk	True	True	✔
Name	Testbruger30	Testbruger30	✔
Company	Testfirma20	Testfirma20	✔
Email	br3@firma2.dk	br3@firma2.dk	✔
Phone	34567890	34567890	✔
SendEmailOk	False	False	✔
SendSmsOk	False	False	✔

Kommentar: *Ingen.*

Fortsættes på næste side

Udførelse:	**Dato:** 17. maj 2007	**Testet af:** Oliver Smith
Godkendelse:	**Dato:** 18. maj 2007	**Godkendt af:** Oliver Smith

Tabel D.3: Rediger data i tabellen Users

Test-id: Test4	**Titel:** Rediger data i tabellen Contacts
Formål:	At teste om alle data der redigeres i tabellen *Contacts* overføres korrekt til databasen.
Forudsætning:	At server og database er korrekt installeret. At alle de foregående test er gennemført tilfredsstillende.

Udførelse:

- Serveren startes og fanebladet til administration af kontakter vælges ved at klikke på knappen *Manage contacts* i højre side af programmet.
- I kolonnen *Contact* ændres Testbruger1 til Testbruger10, Testbruger2 til Testbruger20 og Testbruger3 til Testbruger30.
- Serveren lukkes ned, og startes igen.
- Fanebladet til kontakter af brugere vælges igen.
- Værdierne i tabellen sammenlignes med de der blev indtastet.

Acceptkrav: Der accepteres ikke nogen forskelle mellem det forventede og det aktuelle.

Kolonne	Forventet	Aktuelt	Status
ID	Testbruger10	Testbruger10	✔
Contact	Testbruger20	Testbruger20	✔
ID	Testbruger10	Testbruger10	✔
Contact	Testbruger30	Testbruger30	✔
ID	Testbruger20	Testbruger20	✔
Contact	Testbruger10	Testbruger10	✔
ID	Testbruger20	Testbruger20	✔
Contact	Testbruger30	Testbruger30	✔
ID	Testbruger30	Testbruger30	✔
Contact	Testbruger10	Testbruger10	✔
ID	Testbruger30	Testbruger30	✔
Contact	Testbruger20	Testbruger20	✔

Kommentar: *Ingen.*

Udførelse: **Dato:** 17. maj 2007 **Testet af:** Oliver Smith
Godkendelse: **Dato:** 18. maj 2007 **Godkendt af:** Oliver Smith

Tabel D.4: Rediger data i tabellen Contacts

Test-id: Test5	**Titel:** Slet data i tabellen Contacts
Formål:	At teste om man kan slette data i tabellen *Contacts*
Forudsætning:	At server og database er korrekt installeret. At alle de foregående test er gennemført tilfredsstillende.
Udførelse:	- Serveren startes og fanebladet til administration af kontakter vælges ved at klikke på knappen *Manage contacts* i højre side af programmet. - De fire rækker hvor *Testbruger3* optræder slettes. - Serveren lukkes ned, og startes igen. - Fanebladet til kontakter af brugere vælges igen. - Værdierne i tabellen sammenlignes med de der blev indtastet.
Acceptkrav:	Der accepteres ikke nogen forskelle mellem det forventede og det aktuelle.

Kolonne	Forventet	Aktuelt	Status
ID	Testbruger10	Testbruger10	✔
Contact	Testbruger20	Testbruger20	✔
ID	Testbruger20	Testbruger20	✔
Contact	Testbruger10	Testbruger10	✔

Kommentar:	*Ingen.*	
Udførelse:	**Dato:** 17. maj 2007	**Testet af:** Oliver Smith
Godkendelse:	**Dato:** 18. maj 2007	**Godkendt af:** Oliver Smith

Tabel D.5: Slet data i tabellen Contacts

Test-id: Test6 **Titel:** Slet data i tabellen Users.

Formål: At teste om man kan slette data i tabellen *Users*.

Forudsætning: At server og database er korrekt installeret.
At alle de foregående test er gennemført tilfredsstillende.

Udførelse:

- Serveren startes og fanebladet til administration af brugere vælges ved at klikke på knappen *Manage users* i højre side af programmet.
- Rækken med Testbruger30 slettes.
- Serveren lukkes ned, og startes igen.
- Fanebladet til administration af brugere vælges igen.
- Værdierne i tabellen sammenlignes med de der blev indtastet.

Acceptkrav: Der accepteres ikke nogen forskelle mellem det forventede og det aktuelle.

Kolonne	Forventet	Aktuelt	Status
Name	Testbruger10	Testbruger10	✔
Company	Testfirma10	Testfirma10	✔
Email	br1@firma1.dk	br1@firma1.dk	✔
Phone	12345678	12345678	✔
SendEmailOk	True	True	✔
SendSmsOk	True	True	✔
Name	Testbruger20	Testbruger20	✔
Company	Testfirma10	Testfirma10	✔
Email	br2@firma1.dk	br2@firma1.dk	✔
Phone	23456789	23456789	✔
SendEmailOk	False	False	✔
SendSmsOk	True	True	✔

Kommentar: *Ingen.*

Udførelse: **Dato:** 17. maj 2007 **Testet af:** Oliver Smith
Godkendelse: **Dato:** 18. maj 2007 **Godkendt af:** Oliver Smith

Tabel D.6: Slet data i tabellen Users

Test-id: Test7 **Titel:** Test af oprettelse af forbindelser.

Formål: At teste om klienterne logger rigtigt på.

Forudsætning: At server og klienter er korrekt installeret.
Serveren og klienterne er sat op til at forbinde til hinanden med samme ip-adresse og portnummer.
At alle de foregående test er gennemført tilfredsstillende.

Udførelse:

- To klienter sættes op med henholdsvis brugernavnet *Testbruger10* og *Testbruger20*.
- Serveren og en klient startes på hver sin computer.
- Når der i statusbjælken på serveren står at der er en forbunden klient startes den anden klient på en særskilt computer.
- Når der i statusbjælken på serveren står at der er to forbundne klienter åbnes serverens log, og loggen sammenlignes med det forventede.

Acceptkrav: Det er et krav at alle beskederne er i loggen, og at de er der i den nævnte rækkefølge. Det accepteres dog at der er andre beskeder i loggen som f.eks. *Empty package* og *Receiving data*.

Forventet	Aktuelt	Status
Client connected	Client connected	✔
Client authenticated as 'Testbruger10'	Client authenticated as 'Testbruger10'	✔
Contact list sent to 'Testbruger10'	Contact list sent to 'Testbruger10'	✔
Updated contact lists sent to all connected clients	Updated contact lists sent to all connected clients	✔
Client connected	Client connected	✔
Client authenticated as 'Testbruger20'	Client authenticated as 'Testbruger20'	✔
Contact list sent to 'Testbruger20'	Contact list sent to 'Testbruger20'	✔
Updated contact lists sent to all connected clients	Updated contact lists sent to all connected clients	✔

Kommentar: *Ingen.*

Udførelse: **Dato:** 17. maj 2007 **Testet af:** Oliver Smith
Godkendelse: **Dato:** 18. maj 2007 **Godkendt af:** Oliver Smith

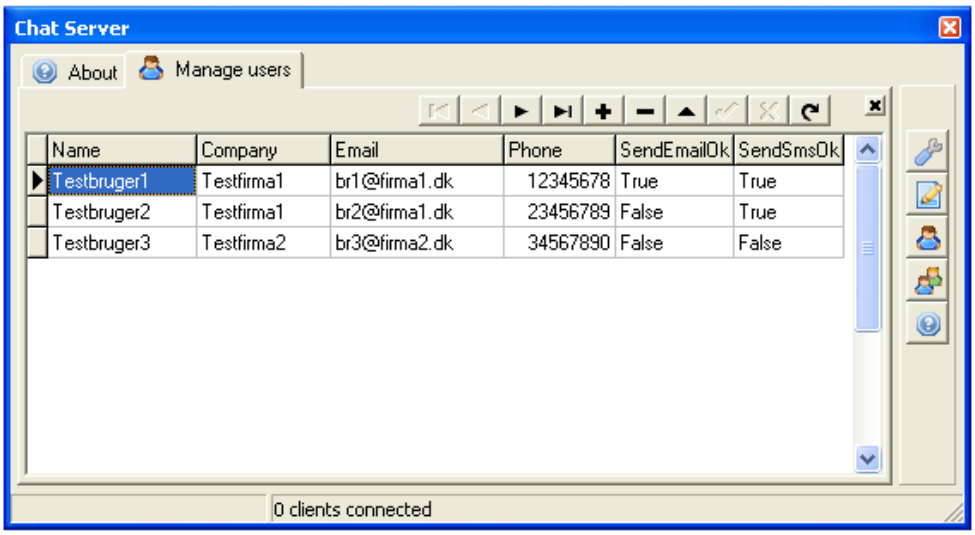
Tabel D.7: Test af oprettelse af forbindelser

Test-id: Test8	**Titel:** Test af afbrydelse af forbindelser.
Formål:	At teste om klienterne logger rigtigt af.
Forudsætning:	At server og klienter er korrekt installeret. Serveren og klienterne er sat op til at forbinde til hinanden med samme ip-adresse og portnummer. At alle de foregående test er gennemført tilfredsstillende. To klienter er forbundet til serveren.
Udførelse:	- Den ene klient lukkes ned. - Den anden klient lukkes ned. - Når der i statusbjælken på serveren står at der er nul forbundne klienter åbnes serverens log, og loggen sammenlignes med det forventede.
Acceptkrav:	Det er et krav at alle beskederne er i loggen, og at de er der i den nævnte rækkefølge. Det accepteres dog at der er andre beskeder i loggen som f.eks. *Empty package* og *Receiving data*. Desuden accepteres det at der i starten af loggen vil være informationer fra klienterne loggede sig på.

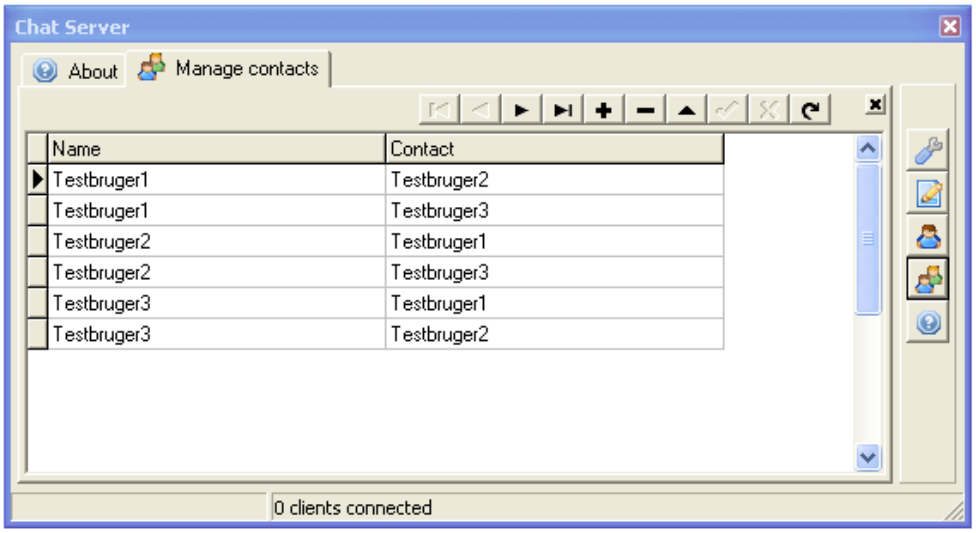
Forventet	Aktuelt	Status
Client 'Testbruger10' disconnected	*Ukendt*	✖
Updated contact lists sent to all connected clients	*Ukendt*	✖
Client 'Testbruger20' disconnected	*Ukendt*	✖
Updated contact lists sent to all connected clients	*Ukendt*	✖

Kommentar:	Da klient nummer to loggede af, fremkom der uventet en dialogboks i serverprogrammet med teksten: "Stack overflow", hvorefter programmet gik ned. Testen fejlede.
Udførelse:	**Dato:** 17. maj 2007 **Testet af:** Oliver Smith
Godkendelse:	**Dato:** 18. maj 2007 **Godkendt af:** *Ikke godkendt*

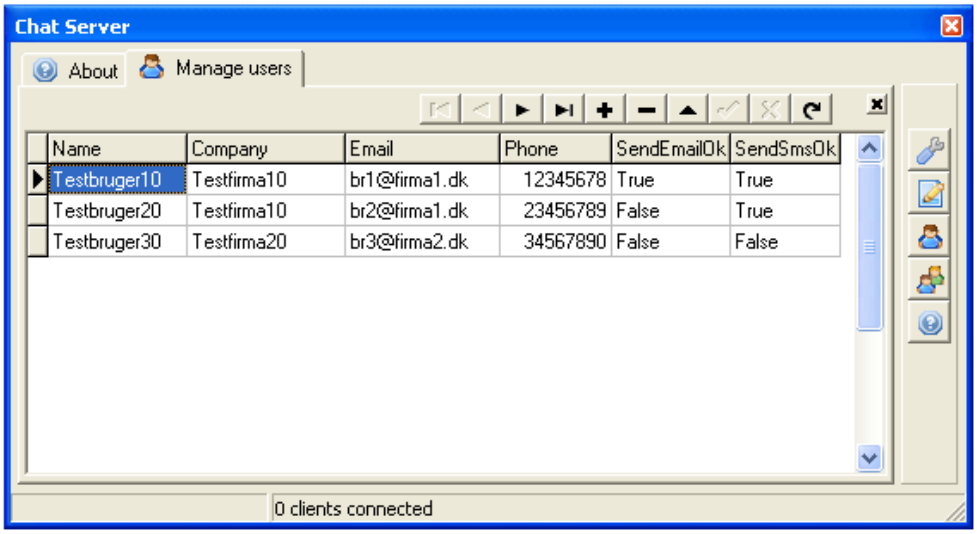
Tabel D.8: Test af afbrydelse af forbindelser



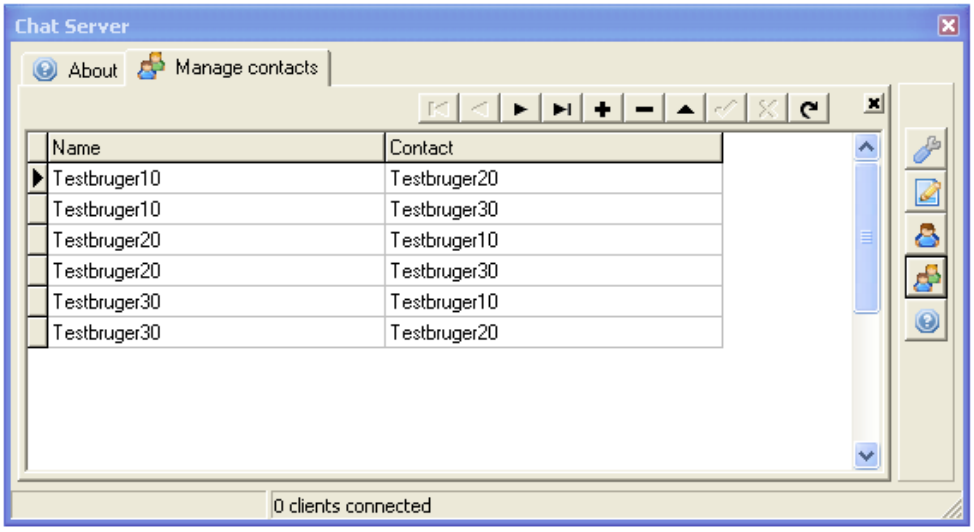
Figur D.1: Dokumentation af test1



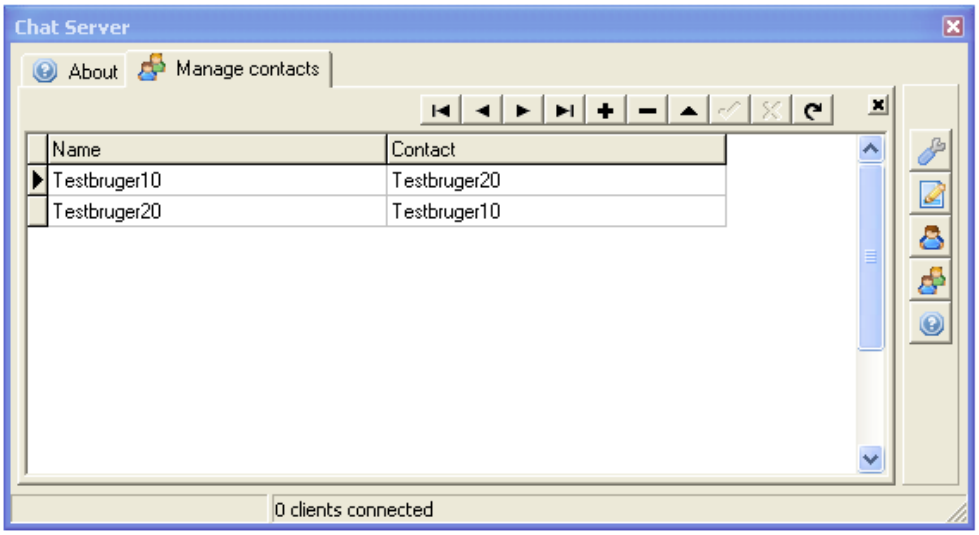
Figur D.2: Dokumentation af test2



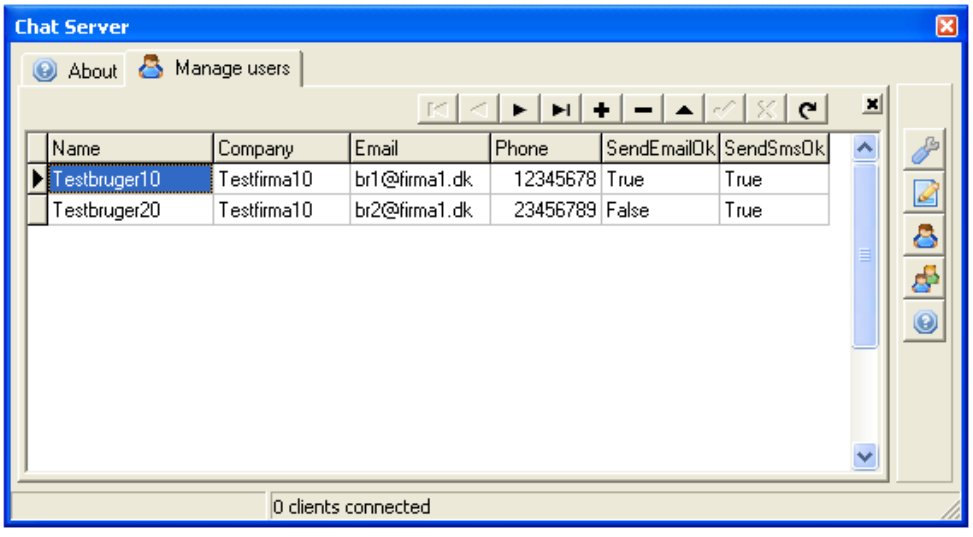
Figur D.3: Dokumentation af test3



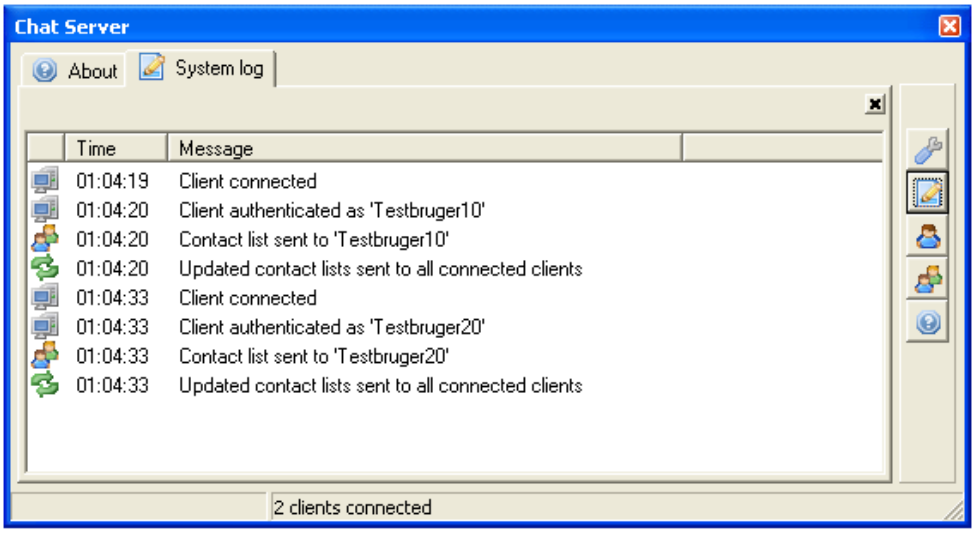
Figur D.4: Dokumentation af test4



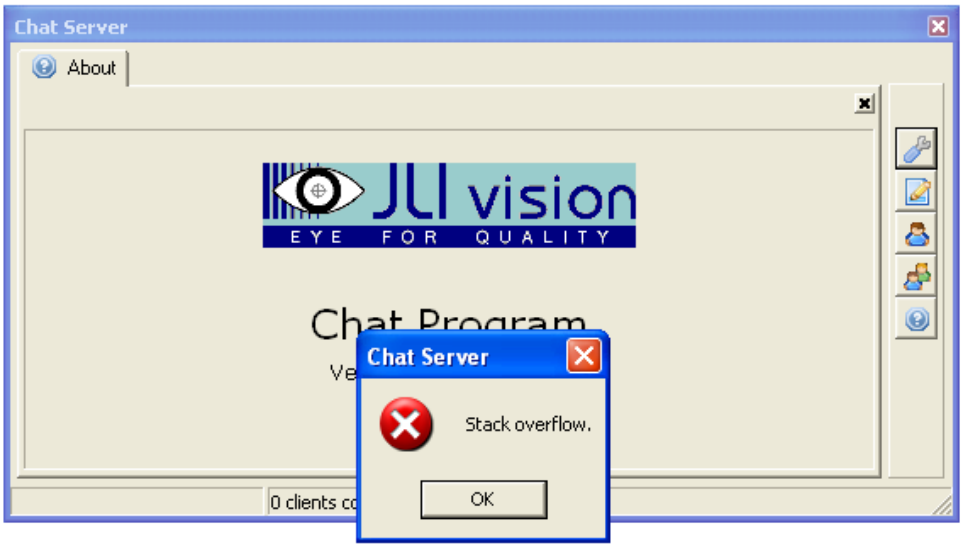
Figur D.5: Dokumentation af test5



Figur D.6: Dokumentation af test6



Figur D.7: Dokumentation af test7



Figur D.8: Dokumentation af test8

Bilag E

Indhold på cd

Bagerst i rapporten er vedlagt en cd med yderligere materiale. Det drejer sig bl.a. om en oversat (kompileret) version af programmet samt kildekoden, som ville være alt for omfattende at vedlægge rapporten i papirform.

Cd'en indeholder følgende mapper:

Bin	Selve programmet, run-time afhængige dll'er og batchfiler til at starte programmet med.
Install	Databasedriver, ny tom database og vejledning til installering og opsætning.
Klassediagram	Klassediagrammet, som er vedlagt på cd'en pga. sin størrelse.
Guide	Vejledninger til installering og opsætning af program og database.
Pdf	To versioner af rapporten: En version med hyperlinks, der er velegnet at læse på en pc, og en almindelig version der er velegnet til udskrift.
Src	C++ kildekode til programmet. Bemærk at projektet ikke kan kompileres da JLI's hjælpebibliotek *common.lib* ikke er vedlagt.
Test	Diverse filer der gør det nemmere at prøvekøre programmet (database med brugerdata og reg-filer til nem opsætning af program).

Figurer

1.1 Illustration af det eksisterende program 2

1.2 Illustration af det ønskede program 4

2.1 Iterationsplan 8

2.2 Use case diagram 10

3.1 3-lags-model 11

3.2 E/R-diagram 13

4.1 Klientens brugergrænseflade 16

4.2 Serverens brugergrænseflade 16

4.3 Skitse over den grafiske brugergrænseflade 17

4.4 Implementering af client-map 18

4.5 Protokol-stak 19

4.6 Socket-kommunikation 19

4.7 Implementering at datapakken PACK_chat_c 20

4.8 Implementering af headeren i basis-klassen 20

4.9 SQL-forespørgsel i database 21

B.1 System sekvens-diagram - Opret forbindelse 33

B.2 System sekvens-diagram - Afbryd forbindelse 34

B.3 System sekvens-diagram - Chat 34

B.4 System sekvens-diagram - Opdater status 34

C.1 Klassediagram – del 1 36

C.2 Klassediagram – del 2 37

D.1 Dokumentation af test1 49

D.2 Dokumentation af test2 49

D.3 Dokumentation af test3 50

D.4 Dokumentation af test4 50

D.5 Dokumentation af test5 51

D.6 Dokumentation af test6 51

D.7 Dokumentation af test7 52

D.8 Dokumentation af test8 52

Tabeller

2.1 Milepæle 8

2.2 Aktører 10

2.3 Use cases 10

4.1 Lagdeling af programmet 15

4.2 Knapper 18

5.1 Opsummering af den funktionelle test 24

A.1 Use case – Opret forbindelse 28

A.2 Use case – Afbryd forbindelse 29

A.3 Use case – Chat 31

A.4 Use case – Opdater status 32

D.1 Indsæt data i tabellen Users 40

D.2 Indsæt data i tabellen Contacts 41

D.3 Rediger data i tabellen Users 43

D.4 Rediger data i tabellen Contacts 44

D.5 Slet data i tabellen Contacts 45

D.6 Slet data i tabellen Users 46

D.7 Test af oprettelse af forbindelser 47

D.8 Test af afbrydelse af forbindelser 48

Litteratur

[1] Kim Hamilton og Russell Miles. *Learning UML 2.0*. O'Reilly, 2006.

[2] Arne Jørgensen. *Kom godt i gang med LaTeX*. 2004. Kan downloades fra: http://www.tug.dk/_media/foredrag/2004-10-23/linuxparty.pdf.

[3] James F. Kurose og Keith W. Ross. *Computer networking: A top-down approach featuring the internet*. Addison-Wesley, 3. udgave, 2005.

[4] Robert Lafore. *Object-Oriented Programming in C++*. SAMS, 4. udgave, 2001.

[5] Jørgen Larsen. *Den nye LaTeX for forfattere – En introduktion til LaTeX og IMFUFA-LaTeX*. 2005. Kan downloades fra: http://dirac.ruc.dk/imfufalatex/ltxnoterh.pdf.

[6] Lars Madsen. *Introduktion til LaTeX – beregnet for nye og øvede brugere*. 3. udgave, 2007. Kan downloades fra: http://www.imf.au.dk/system/latex/bog/version3/beta/ltxb-2007-05-03-11-12.pdf.

[7] Tobias Oetiker, Hubert Partl, Irene Hyna og Elisabeth Schlegl. *The Not So Short Introduction to LaTeX 2ε – Or LaTeX 2ε in 139 minutes*. 4. udgave, 2006. Kan downloades fra: ftp://ftp.dante.de/tex-archive/info/lshort/english/lshort.pdf.

[8] Kendall Scott. *Introduktion til UML*. IDG, 1. udgave, 2002.

[9] Dreamtech Software Team. *Instant Messaging Systems: Cracking the Code*. Wiley Publishing, 1. udgave, 2002.

[10] Website. *JLI vision a/s*. http://www.jli.dk.

[11] Website. *Rules of Data Normalization*. http://www.datamodel.org/NormalizationRules.html.

[12] Peter Wilson. *The Memoir Class for Configurable Typesetting*. The Herries Press, 6. udgave, 2004. Kan downloades fra: ftp://tug.ctan.org/pub/tex-archive/macros/latex/contrib/memoir/memman.pdf.