BACHELOROPPGAVE:

# OREGO

Obligatorisk REGistrering Av Oppmøte

FORFATTERE:

Morten Holberg
Tor Kristian Kjelsberg Hansen

Dato:

19.5.2010

# SAMMENDRAG AV BACHELOROPPGAVEN

| | | |
|---|---|---|
| Tittel: | OREGO – Obligatorisk Registrering Av Oppmøte | Nr. : x |
| | | Dato : 19.5.2010 |
| Deltaker(e): | Morten Holberg | |
| | Tor Kristian Kjelsberg Hansen | |
| Veileder(e): | Frode Haug | |
| Oppdragsgiver: | Høgskolen i Gjøvik | |
| Kontaktperson: | Solveig Granseth | |
| Stikkord (4 stk) | Registrering, Oppmøte, Kortleser, Sykepleiere | |

| | | |
|---|---|---|
| Antall sider: **119** | Antall bilag: **6** | Tilgjengelighet (åpen/konfidensiell):**ÅPEN** |


Kort beskrivelse av Bacheloroppgaven:

Hovedprosjektet vårt har dreid seg om å lage et system for automatisk registrering av oppmøte for Høgskolen i Gjøvik. Det består av en registreringsmodul, en webside, et serverprogram og en database. Via registreringsmodulen Orego kan lærerne enkelt registrere studenter ved hjelp av en kortleser. På Websiden kan lærere og elever se på oppmøte. Serverprogrammet er til for å øke sikkerheten i systemet, og all informasjon ligger lagret i databasen. Registreringsmodulen og Websiden jobber begge mot Serverprogrammet som igjen jobber mot databasen.

Takket være dette systemet har oppdragsgiveren nå et verktøy som gjør hverdagen en hel del enklere. Det gjør registreringsjobben kjapp og effektiv, og det sparer oppdragsgiver for mye jobb med manuell registrering. Websiden gir lærerne en enkel oversikt over oppmøtte studenter, og gjør at de kan oppdatere/endre/slette øvingene når de måtte ønske. På Websiden kan også elevene sjekke sitt personlige oppmøte.

# Forord

Til leseren av prosjektrapporten for OREGO(Obligatorisk REGistrering av Oppmøte).

Denne rapporten er resultatet av et seks måneders prosjekt. Vi har jobbet hardt, og det har blitt en del sene kvelder. Etter at vi fikk tildelt oppgaven fra vår arbeidsgiver ved HIG så har vi jobbet iherdig med den. Vi har måttet lært oss mye nytt, og det har vært en utfordrende oppgave.

Vi vil gjerne benytte anledningen til å takke de personene, som har vært til stor hjelp og inspirasjon gjennom dette prosjektet. Uten råd og hjelp fra dem hadde veien frem til det ferdige produktet blitt mye lenger.

Takk til Frode Haug for all den hjelpen vi har fått fra han som veileder.
Takk til IT-tjenesten for all den hjelpen vi har fått.
Takk til vår arbeidsgiver, avdelingen for sykepleie ved Høgskolen i Gjøvik.
Takk til Didrik Sørensen, Petter Helset og Anders Fjelstad for råd, tips og hjelp.
Takk til HiG for lån av grupperom.

Gjøvik, 19.5.2010

_____ _____

Morten Holberg Tor Kristian K Hansen

# Innholdsfortegnelse

# 1. Innledning

## 1.1 Bakgrunn

Bakgrunnen til dette prosjektet kommer fra sykepleierstudiet ved Høgskolen i Gjøvik. I dag skjer registreringen av oppmøte manuelt ved hjelp av opplesning og avkryssing. Dette er forståelig nok tidkrevende, spesielt når øvingene kan ha opptil 150 studenter, og da forsvinner mye av undervisningstiden til denne opplesningen.

Senere må lærerne også registrere oppmøte manuelt for hver enkelt student via Fronter. Siden dette er veldig tidkrevende har det blitt én misnøye med denne modellen. Derfor ønsker de seg en ny elektronisk løsning på problemet.

Studentene går i dag inn i Fronter og får en enkel oversikt over sine godkjente øvinger. Dette vil lærerne at studentene også skal ha mulighet for å gjøre i den nye elektroniske løsningen.

## 1.2 Problemområde

Vi skal lage et system der sykepleieravdelingen slipper å registrere studentene som er på de obligatoriske øvingene manuelt. Dette tenkte vi å gjøre automatisk via elektronisk avlesning av studentkort, ettersom hver enkel student har sitt personlige studentkort. Deretter må oppmøte lagres i en applikasjon på datamaskinen, for så å bli registrert. Når det senere ønskes en oversikt over oppmøte, skal det vises frem via en webside.

Studentene skal også kunne se de øvingene han/hun har vært med på via Websiden.

## 1.3 Avgrensning

HiG kan ikke avsette mye penger på prosjektet. Derfor har de gjort det til en Bacheloroppgaven for datastudenter.

Det hadde vært optimalt om kortleseren hadde vært trådløs, men med tanke på tid og ressurser ble dette et punkt som vil ble nedprioritert i første omgang. Vi valgte heller å bruke en kortleser med USB siden det fungerer like bra.

I starten ville vi integrere systemet vårt i Fronter, men ettersom Fronter er et ferdiglaget system viste det seg umulig å få til dette. Derfor valgte vi det andre alternativet, som var å lage en webside med samme funksjonalitet. Det er var den løsningen som virket mest reell fra starten av, og det viste seg at vi endte opp med den.

## 1.4 Oppgavedefinisjon

Oppgaven går i hovedsak ut på å effektivisere registreringen av obligatorisk oppmøte på sykepleieravdelingen. Dette vil vi gjøre ved at studentene drar sine studentkort i en kortleser, og dermed skal deres tilstedeværelse bli midlertidig lagret på en datamaskin, før det til slutt ender opp på weben et sted.

Vår oppgave har vært å lage et program, som får ut informasjonen som ligger på studentkortet. Etter at informasjonen er lest ut fra kortet, vil vi ha tak i studentnummeret. Dette nummeret skal så sendes til ldap.hig.no, som er en database over studenter ved HiG. Her vil vi få ut navnet på personen sendt i retur. Når dette har skjedd skal navnene på de som har dratt kortet sitt fortløpende bli lagt inn i en sortert liste. Skulle det bli noen feil med godkjenningen, vil det finnes en funksjon som gjør at studentnumrene kan skrives inn manuelt i systemet (får ikke tilgang til LDAP, eller at en student har glemt kortet). Sistnevnte vil nok være det mest hyppige problemet). Denne listen skal vises i en enkel applikasjon på datamaskinen, som kortleseren er koblet til. Når alle de tilstedeværende elevene har fått registrert seg, kan læreren selv velge hvilken aktivitet som skal foregå, og deretter trykke "registrer". Om det ikke er tilgang til internett skal det komme melding om dette. I disse tilfellene blir listen midlertidig lagret på datamaskinen, slik at læreren kan laste opp listen senere. Hvis ikke skal oppmøte bli lastet opp til en database, hvor en oversikt over oppmøte kan genereres ut fra.

En webside generer et oppsett for de oppmøtte studentene i en tabell. Her vil du finne datoene og øvingene studentene var med på. Hver gang programmet oppdager en ny student, vil denne studenten legges inn i systemet alfabetisk. Dermed vil man enkelt se hvilke studenter som har vært med på hvilke øvinger. Her må det også finnes administrative funksjonalitet for å endre, slette og oppdatere på øvinger, og slette studenter.

Studentene vil kun få tilgang til sine egne øvinger ved hjelp av innlogging med samme brukernavn og passord som dem bruker til innlogging på skolenettet.

### 1.5 Rapportens målgruppe

Målet for denne rapporten er å gi et inntrykk av hvordan prosjektet har utviklet seg, hvordan vi har løst oppgaven, og kvaliteten på det endelige produktet. Rapporten kan også brukes for å forstå hvordan produktet fungerer, og gi et bra innsyn i forhold til helheten i systemet. Målgruppen for denne rapporten vil være oppdragsgiver, faglig veileder, driftansvarlige og brukere av systemet ved HiG, samt alle andre som har interesse av å lese denne rapporten.

Rapporten inneholder mye teknisk informasjon som er mer beregnet på de personene som kommer til å drifte systemet, enn de som kommer til å bruke det. Hvis prosjektet skal videreutvikles er det også viktig at de som skal gjøre det får et bra innsyn i hvordan vi har tenkt og utført oppgaven. Slik at de kan videreutvikle systemet på best mulig måte.

### 1.6 Studentenes faglige bakgrunn/kompetanse

Begge studentene i hovedprosjektgruppen går 3-årig ingeniørfag data ved HiG. Her har vi gjennomgått de grunnleggende programmeringsfagene i C++. Dette innebærer grunnleggende og objektorientert programmering, i tillegg til algoritmiske metoder. Vi har også hatt faget systemutvikling der vi har lært det teoretiske om å jobbe i større utviklingsprosjekter. I tillegg har vi hatt valgfaget Programutvikling i Java, der vi spesialiserte oss innen Java. Vi har også hatt et databasefag der vi fikk et bra innsyn innen det feltet. En av oss også har vært studentassistent i det faget, som har gitt han ekstra kompetanse innen databaser.

Vi har i tillegg hatt faget Informasjon og publikasjonsteknologi der vi lærte oss litt generelt om HTML, CSS og hjemmesider. Det har hjulpet oss litt, men ikke i nærheten nok i forhold til det vi har måttet lære oss på egen hånd under oppgaven.

Vi har tidligere jobbet sammen i prosjekter i Java og C++ fagene. Der hadde vi en god kjemi, så derfor valgte vi å ha Bacheloroppgaven sammen. Siden vi har jobbet sammen før kjenner vi hverandres sterke og svake sider, som har gjort det enklere å fordele arbeidsoppgaver.

Ingen av oss har noe spesiell erfaring fra større utviklingsprosjekter i arbeidslivet, da vi startet med dette prosjektet. Vi har jobbet sammen i mellomstore prosjekter, men ingenting som er helt på dette nivået. Så dette har vært en spennende utfordring.

### 1.7 Formål/hvorfor denne oppgaven

Da vi skulle velge den oppgaven vi ville ha som Bacheloroppgave så vi etter noe som var spennende, utfordrende og meningsfullt. Da vi leste om denne oppgaven bestemte vi oss for den relativt fort fordi den var visuell, utfordrende og spennende. Ikke minst fordi det er noe som vil bli tatt i bruk senere, og vi føler at det vil gi et bra tilskudd til hjelpemidlene for lærerne ved sykepleieravdelingen på HiG.

### 1.8 Ansvarsforhold

Prosjektleder fra starten av var Tor Kristian Hansen, men vi har besluttet i gruppereglene våre at denne tittelen rulleres på hver måned. Ansvaret for Websiden har Morten Holberg. Resten fordeles i fellesskap. Gruppa består av to medlemmer, så vi tror ikke det er hensiktsmessig å inndele noen flere oppgaver ytterligere.

Hovedansvar for prosjektet ligger likt fordelt på Morten og Tor Kristian. Det vil si å holde deadlines, melde i fra om møter og sørge for en jevn og stødig fremgang.

Vi har hatt ansvar for å møte med veilederen vår omtrent en gang i uken for å diskutere fremgang, og få aktuelle tips.

### 1.9 Øvrige roller og bemanning

Vi har fått tildelt Frode Haug som veileder. Arbeidsgivers kontaktperson er Solveig Granseth. HiG har en egen IT-avdeling hvor vår kontaktperson er Jon Langseth. Vi har også jobbet opp mot Fronter, hvor Rigmor Øren Øvstetun og Aleksander Pettersen har vært våre kontaktpersoner.

### 1.10 Terminologi og organisering av rapporten

I rapporten har vi brukt diverse faglige, forkortede og noen steder engelske uttrykk. Disse og andre uttrykk kan være vanskelig for leseren å forstå, og vil derfor samles og forklares inngående i terminologilisten i vedlegg A.

Vi har forsøkt etter best evne å skrive og uttrykke oss på norsk. Noen steder forekommer det allikevel engelske uttrykk ettersom det ikke finnes noen gode norske oversettelser, eller det

ikke finnes noe tilsvarende ord i det hele tatt. Dette er nok fordi det har blitt en vane å bruke disse engelske ordene når folk som arbeider med data kommuniserer seg i mellom. I tillegg finnes det mange forkortelser og interne begreper. Disse ligger også forklart i terminologilisten.

Hele rapporten er skrevet i skrifttype Calibri. Kapitteloverskrifter er skrevet med fet skrift og skriftstørrelse 14. Underoverskrifter er skrevet med fet skrift og skriftstørrelse 12. Vanlig tekst er skrevet med skriftstørrelse 12.

Knapper og applikasjoner vil være beskrevet på følgende møte:
Alle knapper vil starte med stor forbokstav og ha endelsen ”-knappen”. (Registrer-knappen)
Applikasjoner beskrives med stor forbokstav og endelsen ”-applikasjonen”. (Orego-applikasjonen). Kodefunksjoner(*’getStudentNumber(String field)*’) og variable(’conn’) blir definert med enkeltfnutter rundt seg.
Kode klasser blir beskrevet med dobbeltfnutter(”Connection”).
Mapper med filer i skrives etter hva de heter, med stor forbokstav.(Serverprogrammet)

Figurtekster vil være skrevet med fonten Calibri og størrelse 12. Den vil være beskrevet med forkortelsen ”Fig.” for figur. Deretter følger hvilket kapittel figuren tilhører, og hvilket nummer i rekken det er med en bindestrek mellom. Til slutt kommer det en beskrivelse av figuren. Et eksempel på dette er ” Fig. 3-3 Orego-applikasjonen.”. Figurteksten vil befinne seg midtstilt under figuren.

# 2. Kravspesifikasjon

## 2.1 Innledning

### 2.1.1 Omgivelser

Resultatet av dette prosjektet vil bli Orego som legges inn på aktuelle maskiner. I tillegg vil det også bli laget en webside der elevene kan se sitt eget oppmøte, mens lærerne kan se alle elvene. Orego bør installeres på lærernes bærbare maskiner, eller eventuelt labbmaskiner. Kortleseren bruker USB for å kommunisere med datamaskinene.

Systemet vil kommunisere med LDAP, som er en database for informasjon om studenter og ansatte ved HIG. Vi bruker LDAP for å hente ut informasjon om hvilken student som drar kortet, og for å sjekke om brukernavn og passord er korrekt ved innlogging på Websiden. I det første tilfellet bør systemet ha tilgang til internett, men det vil ha en reserve løsning.

Websiden for lærerne vil også fungere som administrator for systemet. Der kan de få oversikt over alle elevene, og muligheten for å kunne endre/slette/opprette navn på øvinger, samt å slette studenter.

### 2.1.2 Systemets brukere

Vår applikasjon har to typer hovedbrukere, elevene og lærerne. Elevene vil se minimalt av systemet som innbærer kortleseren hvor de drar studentkortet sitt, eller oppgir studentnummeret. De vil også ha tilgang til Websiden, hvor hver elev kan se sitt eget oppmøte for hva de har fullført og mangler. Lærerne vil derimot se hovedvekten av systemet via Orego og Websiden for lærere. På Websiden vil lærerne fungere som administratorer på nettet. Lærerne vil starte Orego på datamaskinen sin, og velge emne og øving før elevene drar studentkortene. I applikasjonsvinduet vil lærerne se navnet på hvilke studenter som har møtt opp til øvingen, og etter timen kan læreren laste opp de oppmøtte til Websiden. På Websiden vil lærerne ha full oversikt over alle elever og øvinger, og mulighet for å endre/slette/oppdatere øvinger, samt å slette studenter. Der kan de se hvem som har møtt opp, og ikke møtt opp. Systemet vil også være tilgjengelig for IT-tjenesten, som fungerer som driftsansvarlig.

### 2.1.3 Systemet

Her følger en forklaring av hvordan systemet fungerer generelt.

En lærer går til kullkoordinator og hentet en kortleser. Deretter går han/hun til timen (med en egen bærbar om det ikke finnes en stasjonær der fra før) og starter programmet. Så kommer studentene og drar kortene sine etter tur.

Hver student har et studentnummer som identifiserer studenten. Dette ligger på studentkortene gjemt i en usammenhengende streng. For eksempel: *ø0163081000703760_*. Av denne strengen er det de seks tallene fra tiende posisjon som er studentnummeret (i vårt eksempel: *070376*). Når programmet har fått ut dette sendes dette mot LDAP-databasen. LDAP mottar og svarer med å sende riktig navn på personen tilbake. Når alle studentene har dratt kortet sitt kan læreren laste opp oppmøtet til databasen. Orego er laget med utgangspunkt i at man har tilkobling til internett når kortene dras. Men om dette ikke skulle være tilfelle så finnes det en sikkerhetsløsning. Da vil bare studentnumrene dukke opp i listen i Orego. Der finnes det en knapp som heter Hent navn. Når du da har internett igjen trykker du på denne, og navnene på studentene vil bli lagt inn i listen. Etter du har gjort det kan du trykke på Registrer-knappen, og laster opp oppmøte. Hvis studenten har glemt studentkortet sitt går det an å trykke på Manuelt-knappen og skrive inn studentnummeret manuelt. Da vil programmet søke etter studenten ved hjelp av LDAP, og hvis det fantes vil han/hun dukke opp i et tekstfelt. Hvis programmet fant student kan læreren trykke på Legg til-knappen og studenten vil bli lagt inn i listen.

Systemet vil vise oppmøtet via den såkalte Websiden. Dette gjør den ved å hente ut studenter som har møtt opp fra databasen, og legger de inn i en tabell. Studentene kan logge seg inn på Websiden å se sitt eget personlige oppmøte. De vil enkelt se hva de har gjennomført og hva de mangler via en vertikal tabell. Lærerne logger seg inn på sin egen side der de har to tabeller med oversikt over alle studentene og deres oppmøte. Der har læreren mulighet for å gå til en side der man får se en oversikt over alle ferdige studenter, og mulighet for å slette de. Det er også en side for å slette en enkelt student via å oppgi studentnummeret. Den siste siden det er link til er siden for å endre/slette/opprette nye øvinger.

## 2.2 Detaljert kravspesifikasjon

### 2.2.1 Funksjonell kravspesifikasjon

#### 2.2.1.1 Use Case Diagram

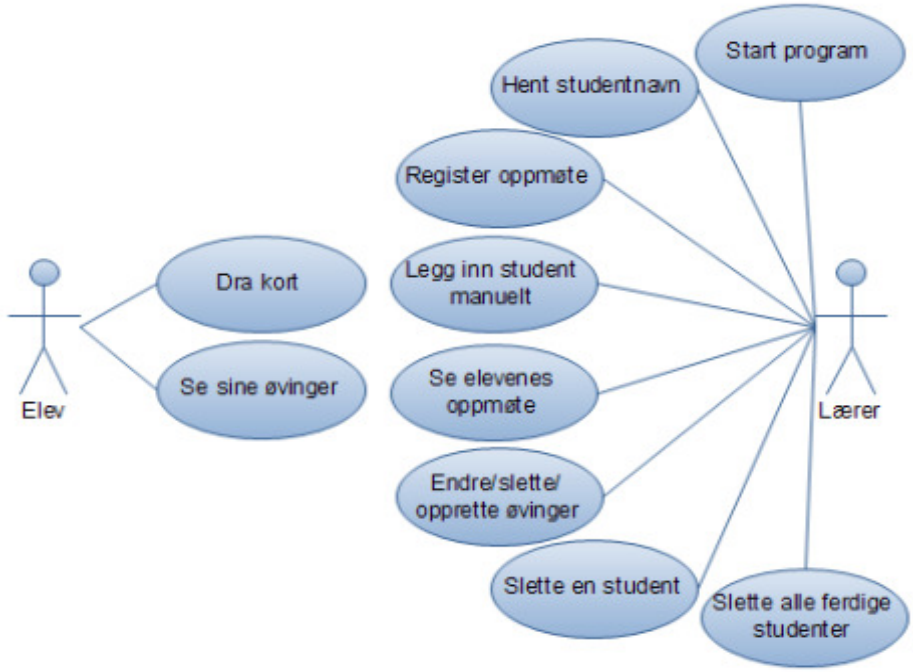



Fig. 2–1 Use Case diagram

Vi har valgt å kutte ut high level Use Case beskrivelsene, fordi vi ikke ser noe poeng i å ha dem med når low level Use Case beskrivelsene forklarer alt mye grundigere.

#### 2.2.1.2 Domenemodell

I domenemodellen nedenfor så har vi en enkel oversikt over hvordan applikasjonen henger sammen. Eleven drar kortet i Dra kort, hvor vi får ut kortnummeret. Dette blir sendt til Student. Studentnummeret blir sendt til LDAP-databasen, hvor det sjekkes opp, og navnet på eleven blir returnert til Student. Studenten blir så lagt inn i en liste over oppmøtte studenter. Der legges også attributtene kull, øving og dato til. Disse kommer fra gruppene kull og øving, som velges av lærer via et grensesnitt. Deretter legges listen med studentene, dato, kull og hvilken øving det er inn i Registrert oppmøte på Websiden. Her ligger alle oppmøtte på alle øvinger samlet i to tabeller så læreren får en grei oversikt over oppmøtet.

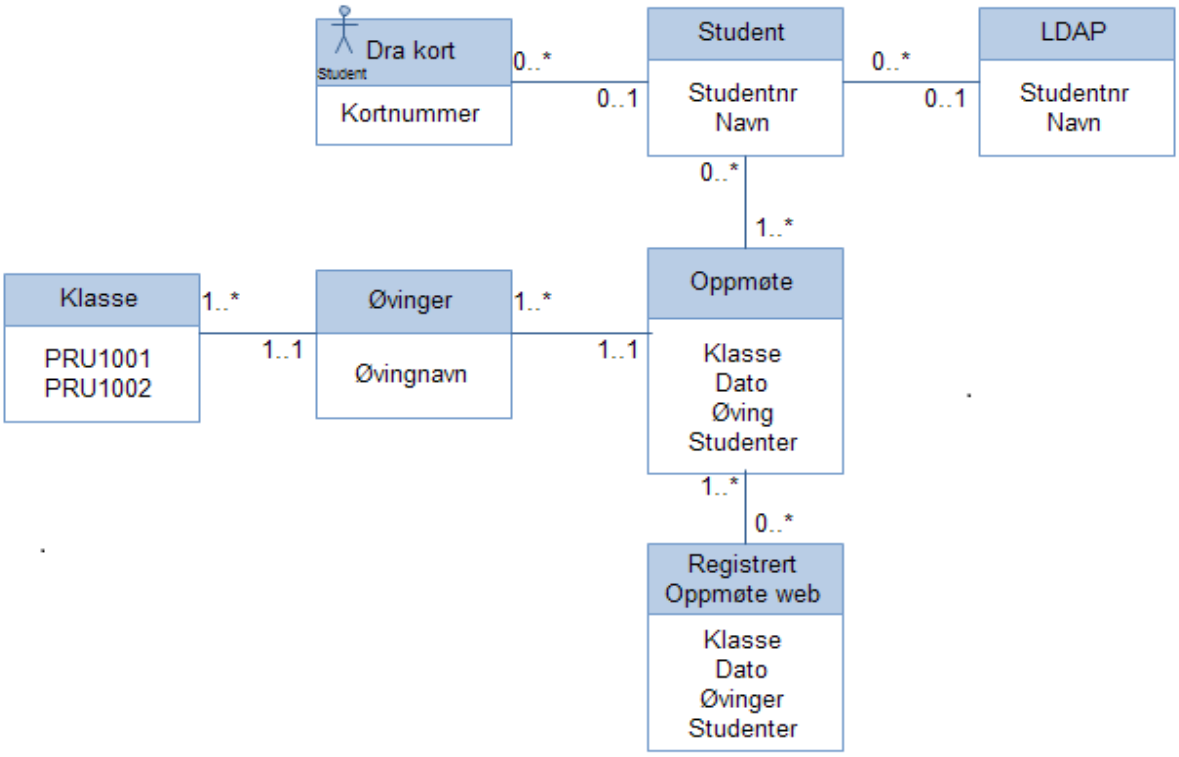



Fig. 2-2 Domenemodell

### 2.2.2 Overordnede operasjonelle systemkrav

#### 2.2.2.1 Brukervennlighet

Applikasjonen vil bli laget hovedsakelig på norsk, siden det er et norsk prosjekt og oppdragsgiver. Det vil bli en enkel og brukervennlig applikasjon. For læreren vil det gjøre registreringen mye mer effektivt en det var før. Nå trenger de bare å ha med seg korteleseren, og tilgang til en datamaskin med internett og applikasjonen på. Før måtte de ha med seg en lang liste og ta opprop, for så å registrere de manuelt på nett. Nå skjer dette elektronisk, slik at registreringsløsningen blir mye mer brukervennlig. Applikasjonen vil hovedsakelig bli laget for bruk i Windows, og når man vil sjekke oppmøte på nettet så vil hjemmesiden via Fronter være kompatibelt med de aller fleste nettlesere. De designmessige aspektene er bedre spesifisert i design delen.

Studenter trenger generell internett kunnskap for å logge seg inn å sjekke oppmøtet sitt. Lærerne vil trenge generell internettkunnskap, og innføring i hvordan systemet fungerer.

#### 2.2.2.2 Ytelse

Programmet vil bli designet for å bruke minst mulig ressurser, men også for å gå fortest mulig. Å starte applikasjonen skal ta maksimalt 5 sekunder, men det kommer litt an på datamaskinen. Når studentene drar kortet skal det ikke ta lenger enn maksimalt 2 sekunder før navnet ligger i listen. På denne tiden skal kortnummeret bli kontrollert opp mot LDAP databasen, og navn og nummer lagret unna i studentlisten. Når all registreringen er gjort unna så skal det ta maksimalt 10 sekunder fra man trykker på Registrer-knappen, til at all oppmøte er registrert på nettet og studentlisten i programmet er nullstilt.

#### 2.2.2.3 Systemkrav

Når man starter Orego-applikasjonen skal det ta maksimalt 5 sekunder, forutsatt relativt ny datamaskin, før Orego er operativt.

Fra man drar kortet skal det maksimalt ta 2 sekunder før navn og nummer ligger i studentlisten i Orego-applikasjonen. Det kan eventuelt ta litt lenger tid for den første personen som drar kortet ettersom tilkoblingen til LDAP skjer da.

Listen skal sorteres i alfabetisk rekkefølge. Om 2 personer har like etternavn vil det bli gjort en sjekk på fornavn, slik at listen alltid vil være alfabetisk sortert.

Det skal ta maksimalt 10 sekunder å laste opp studentlisten fra applikasjonen til Websiden, samt å nullstille studentlisten i programmet.

Kun en person kan laste opp til databasen og Websiden av gangen.

Hvis programmet klikker når det går, så vil den midlertidige listen lastes inn i programmet igjen når det startes på nytt.

### 2.3 Funksjonelle krav

#### 2.3.1 Expanded Use Case Diagrammer

| **Expanded Use Case** | |
|---|---|
| **Use Case:** | Starte program |
| **Aktør:** | Lærer |
| **Formål:** | Få startet programmet |
| **Sammendrag:** | Læreren må ha en datamaskin med applikasjonen på. Læreren må deretter starte opp maskinen og applikasjonen. Etter det er gjort må kortleseren plugges i, og så er det klart til bruk. |
| **Pre:** | Applikasjonen må være lagt inn på datamaskinen, og kortleseren plugget i. |
| **Post:** | Etter at programmet er startet opp kan elevene dra kortet |
| **Spesielle krav:** | Bør trykke på oppdater knappen når applikasjonen har startet. |
| **Hendelsesflyt** | |
| Aktør | System |
| 1. Læreren starter applikasjonen. | |
| | 2. Applikasjonen starter opp, og dukker opp på skjermen. |
| 3. Læreren plugger i kortleseren. | |
| | 4. Kortleseren gir fra seg et pip når den er klar. |
| | 5. Systemet er klart for bruk. |
| **Alternativ hendelsesflyt** | |
| 1. Hvis det er noe feil med applikasjonen. Prøve omstart av den. Hvis det ikke fungerer så bruke det gamle systemet.<br>4. Kortleseren gir ikke fra seg et pip om at den er klar til bruk. Ta en omstart av applikasjonen å plugge i kortleseren på nytt. Hvis det ikke fungerer. Kontakt IT-tjenesten ang ny kortleser. Hvis det ikke fungerer må de gå tilbake til det gamle systemet. | |

Fig. 2-3 Expanded Use Case diagram

| Expanded Use Case | |
|---|---|
| **Use Case:** | Se elevens oppmøte |
| **Aktør:** | Lærer |
| **Formål:** | Kunne se elevenes godkjente øvinger. |
| **Sammendrag:** | Læreren starter nettleseren sin og logger seg inn på hjemmesiden. Der vil det ligge en oversikt over alle studentene og deres godkjente øvinger. |
| **Pre:** | Tilgang til Fronter, og brukernavn og passord til Websiden for lærere. |
| **Post:** | Når læreren har logget seg inn er det mulig å se hvilke elever som har godkjente øvinger og ikke. |
| **Spesielle krav:** | Tilgang til internett, Fronter, og brukernavn og passord til Websiden. |
| **Hendelsesflyt** | |
| Aktør | System |
| 1. Lærer logger seg inn på Fronter. | |
| | 2. Godkjent innlogging i Fronter |
| 3. Læreren velger linken til siden for oversikten over studentene. | |
| 4. Skriver inn passord og brukernavn | |
| | 5. Sjekker om brukernavn og passord stemmer. |
| | 5. Hvis det stemmer så lastes siden opp til lærer. |
| 7. Lærer kontrollerer oppmøte | |
| **Alternativ hendelsesflyt** | |
| 1. Kontakt Studenttorget/IT-tjenesten.<br>4. Hvis du ikke får tilgang til å se oppmøte. Prøv igjen, hvis det ikke fungerer kontakt kullkoordinator først. Deretter IT-tjenesten hvis det ikke løser seg. | |

Fig. 2-4 Expanded Use Case diagram

| Expanded Use Case | |
|---|---|
| **Use Case:** | Legg inn student manuelt |
| **Aktør:** | Lærer |
| **Formål:** | Å legge inn en student manuelt |
| **Sammendrag:** | Læreren står og kontrollerer at elevene drar kortet. Hvis eleven har glemt kortet må læreren legge inn eleven manuelt |
| **Pre:** | Eleven har glemt kortet, og husker studentnummeret sitt |
| **Post:** | Læreren legger inn eleven manuelt med hjelp av studentnummeret. |
| **Spesielle krav:** | Eleven glemt kortet. Kan studentnummer. |
| **Hendelsesflyt** | |
| Aktør | System |
| 1. Student glemt studentkortet sitt. | |
| | 2. Lærer trykker på Manuelt-knappen. |
| 3. Eleven oppgir studentnummeret. | |
| 4. Lærer skriver inn nummeret og trykker søk. | |
| | 5. Nummeret blir sjekket i databasen og vist frem hvis funnet. |
| 6.Hvis man fant studenten trykker læreren på Legg til-knappen. | |
| | 7. Studenten blir lagt inn i listen i programmet. |
| **Alternativ hendelsesflyt** | |
| 2. Manuelt-knappen fungerer ikke. Restart programmet. Hvis dette ikke funger så noter studentnummeret og last opp en annen gang da det fungerer via kort eller studentnummeret. Deretter kontakt IT-tjensten.<br>4. Finner ikke eleven. Kontroller studentnummer. Hvis det ikke fungerer så restart programmet, og prøv på nytt. Fungerer ikke det så noter studentnummeret og last opp senere. Deretter kontakt IT-tjensten<br>6. Fungerer ikke Legg til-knappen, og studenten ikke blir lagt inn i listen med navn og studentnummer. Restart og prøv på nytt. Hvis det ikke fungerer så noter og last opp ved senere anledning. Deretter kontakt IT-tjenesten. | |

Fig. 2-5 Expanded Use Case diagram

De resterende syv low level Use Case diagrammene finner man i vedlegg E.

### 2.3.2 Sekvensdiagram

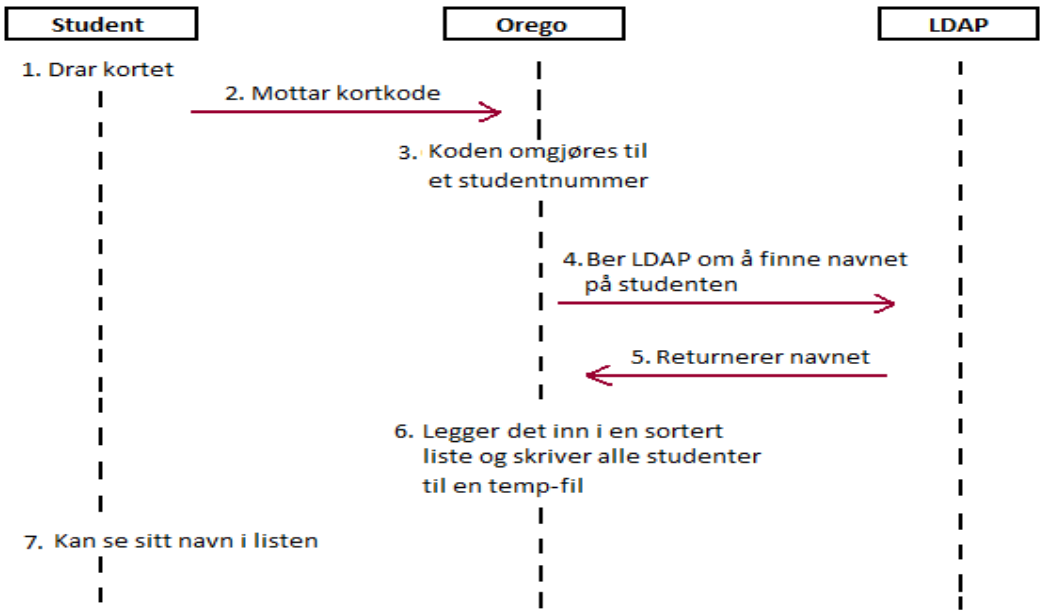



Fig. 2-6 Sekvensdiagram for å dra kort

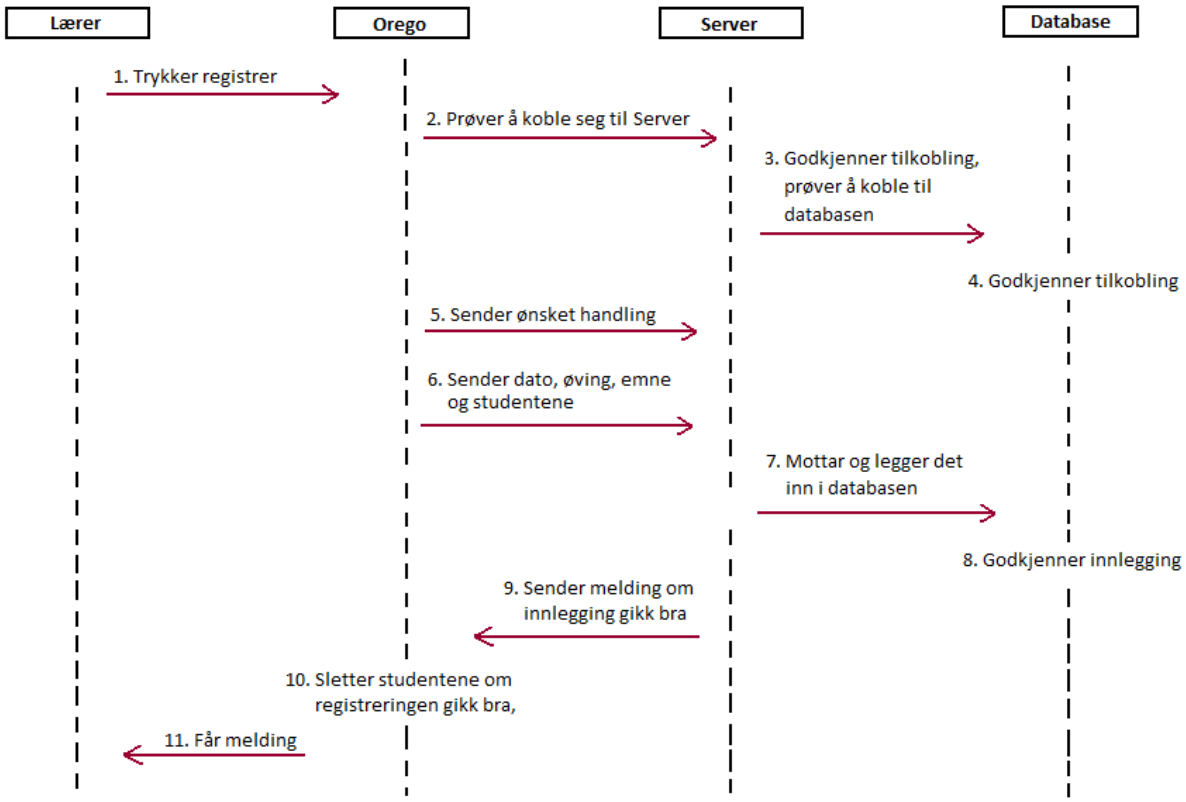



Fig. 2-7 Sekvensdiagram for å registrere oppmøte

Det siste Sekvensdiagrammet for Websiden finnes i vedlegg E.

## 2.4 Begrensninger

### 2.4.1 Software design og begrensninger

#### 2.4.1.1 Software standarder og språk

Vi kommer til å programmere Orego i Java. Websiden blir kodet i HTML, Java og JavaScript.

All kode som skrives vil bli kommentert og en mer utfyllende dokumentasjon vil følges etter punkt 2.5.1

#### *2.4.1.2 Nettleser*

Vi vil tilpasse webgrensesnittet slik at det er kompatibelt med de fleste av dagens nettlesere, slik som Firefox, Opera og Internett Explorer.

### 2.4.2 Hardware design begrensninger

Kravene til hardware for å kunne drive Orego er relativt lave. Man trenger en helt standard datamaskin med tilgang til en internett-tilkobling. Det eneste det forlanges er at studentene har tilgang til en datamaskin med en ordinær nettleser.

## 2.5 Aspekter omkring livssyklus

### 2.5.1 Dokumentasjon

Under utvikling av produktet dokumenteres all jobbing og beslutninger via en logg. I denne loggen skrives hvem som har jobbet, hva de har jobbet med, og hvor lenge det har blitt jobbet. På hjemmesiden til prosjektet har det blitt lagt ut logg over ukentlige fremgang og aktiviteter for prosjektet. Programmeringskode dokumenteres med kommentering i kildekoden, og noe Javadokumentasjon. Kildekoden vil forbli lukket hos produsent og oppdragsgiver. Det vil også følge med dokumentasjon som gir en forklaring på hva som ligger i hvilke filer og hva de gjør. Dette kan være til stor hjelp ved eventuell videreutvikling av systemet. Det er avholdt tre statusrapport-møter, hvor fremgangen og utformingen av systemet har blitt diskutert og evaluert. Resultatet av dette vil forekomme som statusrapporter, som vil bli lagt ved som vedlegg.

### 2.5.2 Krav til support, service og vedlikehold

Oppmøte vil ligge lagret på Websiden, som igjen ligger på serveren til HiG, hvor vi regner med at de har standardprosedyrer for sikring av data. Dette for å forhindre tapet av det registrerte oppmøte hos elevene, siden dette er en viktig del av utdanningen til sykepleierstudentene. En måte de kan gjøre dette på er at de bruker en funksjon i MySql der de logger seg inn som administrator med GUI-verktøy. Der finnes det en funksjon som heter schedualed backup, som betyr avtalt backup. Dette er noe som vil skje automatisk, og vi

anbefaler IT-tjensten å bruke det på det sterkeste. IT-tjenesten ved HiG vil ha en person som har innsikt i programmet. Denne personen har mulighet for å ta seg av eventuelle problemer som dukker opp. Lærerne hos sykepleierne vil fungere som administrator på Websiden, hvor de kan oppdatere og vedlikeholde øvingsnavn og studenter. Hvis det er behov for å oppdatere/ordne Orego så er det noe IT-tjensten må ta seg av. Hvis ikke må utviklerne kontaktes. Oppdragsgiver har ønsket seg tre kortlesere. Det bør i tillegg ligge en ekstra kortleser hos IT-tjenesten, hvis en av de operasjonelle kortleserne skal slutte å fungere. Det er også mulig å bestille flere hvis det vil vise seg å bli nødvendig. Vi som produsent vil også være tilgjengelig, hvis det skal oppstå vansker.

## 2.6 Aspekter omkring installasjon

### 2.6.1 Software installasjon

Siden det er IT-tjenesten som tar over og drifter dette systemet, så er det opp til dem hvordan de vil installere det. Orego er dog bare en fil som ikke trenger noen installasjon, og kortleseren vil bli installert av datamaskinen når den plugges i. Vi ser for oss to muligheter for hvordan Orego kan bli lagt inn. Enten ved å legge ut Orego i Fronter hvor lærerne selv kan laste det ned, eller at IT-tjenesten legger det inn på de ønskede datamaskinene selv. De resterende delene av systemet vil bli installert etter hvordan IT-tjenesten ønsker. De vil kunne sette opp deler av systemet selv ved hjelp av ressursfiler.

### 2.6.2 Opplæring

Vi vil holde et kurs hos lærerne der vi vil vise frem systemet, og gi lærerne en opplæring i hvordan det hele fungerer. Dette for å skaffe trygghet rundt Orego, slik at det blir enklere for dem å ta det i bruk. Kullkoordinatorene på sykepleierstudiene vil få en ekstra grundig innføring i systemet.

En ansatt på IT-tjenesten vil også få en grundig innføring i systemet, og hvordan det hele er satt sammen og fungerer.

# 3. Design

## 3.1 Introduksjon

Da vi startet med prosjektet var vi klar over at det ville bli mye jobb designmessig. Programmet vi har fått i oppdrag å lage skal være et meget visuelt program, derfor har det gått med mye tid til selve designdelen og hvordan det skal se ut. Vi har hatt forskjellige ideer og tanker angående utseende, så det har blitt mye drøfting frem og tilbake.

Ut i fra oppdragsbeskrivelsen kom vi frem til at vi måtte lage en registreringsapplikasjon og to websider. Den ene skal benyttes av lærerne for å sjekke oppmøte og administrere, mens den andre skal elevene benytte for å sjekke sitt personlige oppmøte.

Under utviklingen av applikasjonen og Websidene har vi fokusert på å holde det mest mulig oversiktelig, enkelt og effektivt. Selv synes vi at vi har fått det til bra i forhold til disse kriteriene.

## 3.2 Alternative design

### 3.2.1 Første utkast til løsning

Når vi startet utviklingen av registreringsmodulen bestemte vi oss for å lage den i Java, fordi Java er veldig GUI-vennlig. Der skal man kunne få lest inn studentene og registrert de på riktig øving. Vi tenkte også å bruke den til å administrere sletting av studenter, endring på øvinger og å legge til nye øvinger.

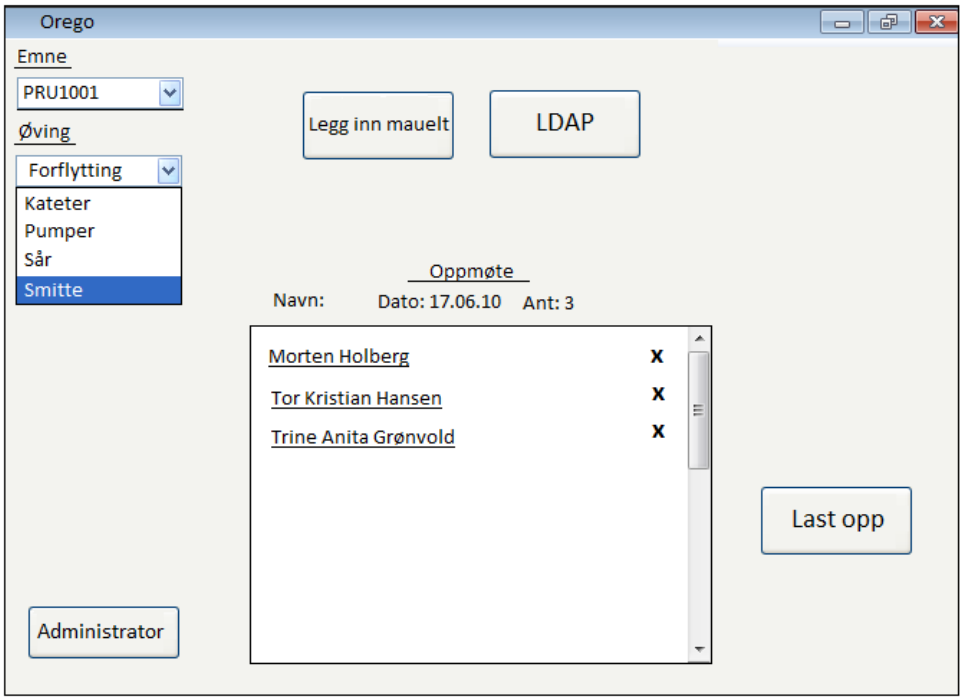
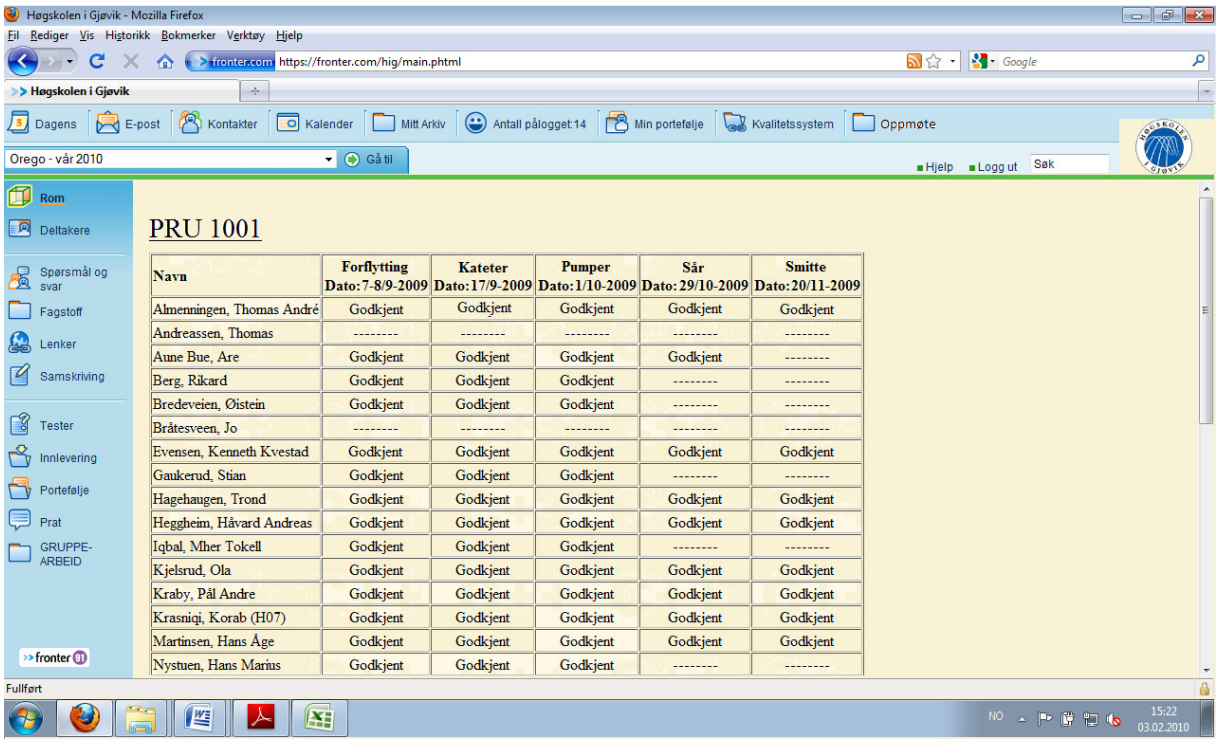



Fig. 3-1 Første utkast til Orego-applikasjonen laget i Paint.

Websiden tenkte vi å integrere i Fronter via en link. Slik at lærerne fikk se alle, og studentene bare seg selv. Dette var den første tanken vi hadde, men etter som vi har kommet videre ut i prosjektet har jo det forandret seg.




| Navn | Forflytting Dato: 7-8/9-2009 | Kateter Dato:17/9-2009 | Pumper Dato: 1/10-2009 | Sår Dato: 29/10-2009 | Smitte Dato:20/11-2009 |
|---|---|---|---|---|---|
| Almenningen, Thomas André | Godkjent | Godkjent | Godkjent | Godkjent | Godkjent |
| Andreassen, Thomas | -------- | -------- | -------- | -------- | -------- |
| Aune Bue, Are | Godkjent | Godkjent | Godkjent | Godkjent | -------- |
| Berg, Rikard | Godkjent | Godkjent | Godkjent | -------- | -------- |
| Bredeveien, Øistein | Godkjent | Godkjent | Godkjent | -------- | -------- |
| Bråtesveen, Jo | -------- | -------- | -------- | -------- | -------- |
| Evensen, Kenneth Kvestad | Godkjent | Godkjent | Godkjent | Godkjent | Godkjent |
| Gaukerud, Stian | Godkjent | Godkjent | Godkjent | -------- | -------- |
| Hagehaugen, Trond | Godkjent | Godkjent | Godkjent | Godkjent | Godkjent |
| Heggheim, Håvard Andreas | Godkjent | Godkjent | Godkjent | Godkjent | Godkjent |
| Iqbal, Mher Tokell | Godkjent | Godkjent | Godkjent | -------- | -------- |
| Kjelsrud, Ola | Godkjent | Godkjent | Godkjent | Godkjent | Godkjent |
| Kraby, Pål Andre | Godkjent | Godkjent | Godkjent | Godkjent | Godkjent |
| Krasniqi, Korab (H07) | Godkjent | Godkjent | Godkjent | Godkjent | Godkjent |
| Martinsen, Hans Åge | Godkjent | Godkjent | Godkjent | Godkjent | Godkjent |
| Nystuen, Hans Marius | Godkjent | Godkjent | Godkjent | -------- | -------- |

Fig. 3-2 Første utkast til Websiden for oversikt over studenter laget i Paint.

Disse utkastene ble ikke så langt unna den egentlige løsningen, og det var et veldig bra utgangspunkt. Men mange forandringer har blitt gjort, og alt det til det bedre mener vi.

### 3.2.2 Den endelige løsningen

Den endelige løsningen ble at vi brukte Java applikasjonen til bare å lese inn studentene og laste de opp til databasen på serveren med riktig øving. Vi har endt opp med to hjemmesider, en for studenter og en for lærerne. Studentene vil få en link via Fronter der de må logge seg inn, som viser dem deres eget oppmøte. Lærerne vil få en link i Fronter der de kommer til en side der de må logge seg inn. Etter det får de en oversikt over alle studenters oppmøte. Det vil ligge noen knapper der for å administrere sletting av studenter, og endring på øvinger.

## 3.3 Grafisk design og grensesnitt

Det er to selvstendige deler i det grafiske grensesnittet til vårt produkt. Vi har Orego, og de to Websidene. Under vil vi gå mer detaljert inn i hver av de to delene.

### 3.3.1 Oregomodulen

Denne applikasjonen var litt tidkrevende, men den ble fin og oversiktlig. Utseende på den ble ikke så langt fra det originale utkastet. Den eneste forskjellen er at vi tar ikke oss av noe som har med administrator å gjøre her lenger, det skjer på Websiden. Under ser man et bilde av hele applikasjonsvinduet for registrering av studenter, som lærerne vil bruke.

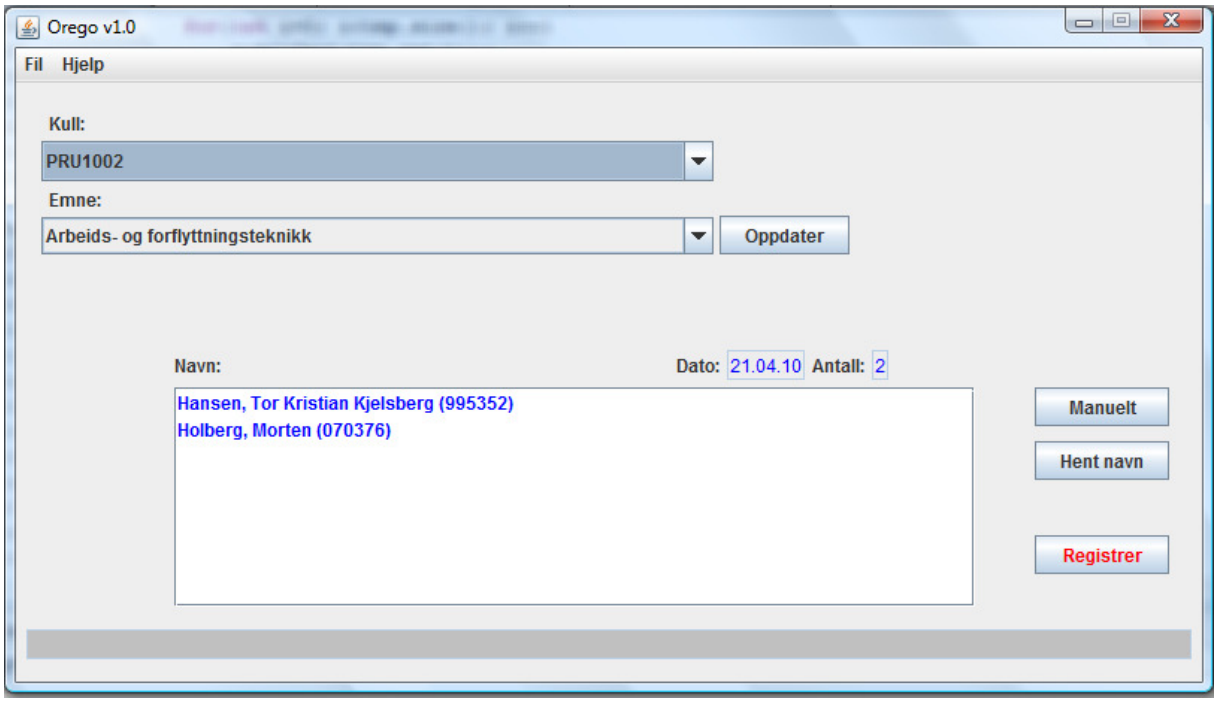



Fig. 3-3 Orego-applikasjonen.

I ned trekks boksen (”drop down box”) under kull velger man hvilket kull studentene som skal registreres går i. Under den igjen har vi en ny ned trekks boks der vi velger hvilken øving studentene skal registreres på. Disse forandres ettersom hvilket kull som står i ned trekks boksen for kull ovenfor. Dette gjør det meget plassparende, effektivt og brukervennlig.

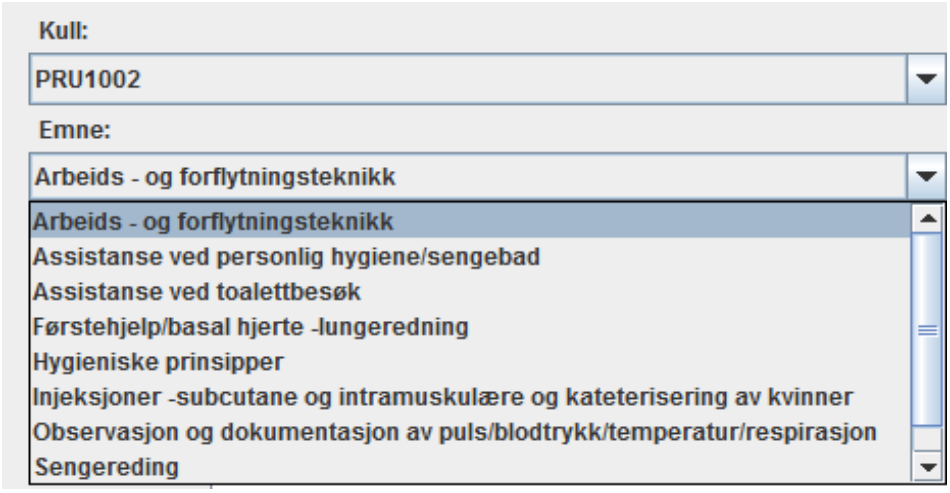



Fig. 3-4 Dropp down boksene i Orego-applikasjonen.

I midten av bildet har vi en liste over alle studentene som har dratt kortet sitt. Dette gjør det veldig oversiktlig for lærerne å se hvilke studenter som har møtt opp til undervisningen. Studentene ligger sortert på etternavn i denne listen, så det er lett å finne igjen en student.


Fig. 3-5 Listen over oppmøte studenter i Orego-applikasjonen.

Ovenfor listen har vi et datofelt som viser hvilken dato det er. Dette er fint for læreren med tanke på hvilke datoer de forskjellige øvingene skal utføres. Ved siden av dato ligger et antall på hvor mange studenter som har dratt kortete sitt i løpet av øvingen. Dette kan være behjelpelig for å kontrollere hvor mange som virkelig har vært til stede ved øvingen.

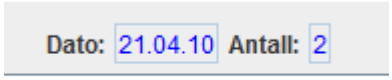



Fig. 3-6 Dato og antall oppmøte i Orego-applikasjonen.

Nedenfor går vi gjennom de fire knappene i Orego-applikasjonen. Det er disse som utgjør hovedfunksjonaliteten i programmet. Det er de som får ting til å skje, og til å registrere oppmøte.

**Oppdater:**

Oppdater-knappen er en knapp for å oppdatere navnene på de forskjellige øvingene som kullene har, hvis de har blitt endret. Denne bør trykkes på når det er laget en ny, slettet eller endret på en øving. Slik at applikasjonen fungerer korrekt, og man får gjennomført riktig øving.

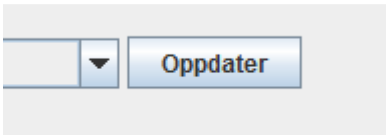



Fig. 3-7 Oppdater-knappen i Orego-applikasjonen

**Manuelt:**

Dette er knappen man trykker på hvis studenten har glemt studentkortet sitt. Det fungerer bra så lenge studenten husker sitt eget studentnummer, noe som vi tar høyde for.


Fig. 3-8 Manuelt-knappen i Orego-applikasjonen.

Når man trykker på denne knappen kommer det opp et lite vindu hvor du kan skrive inn studentnummeret og søke etter studenten.

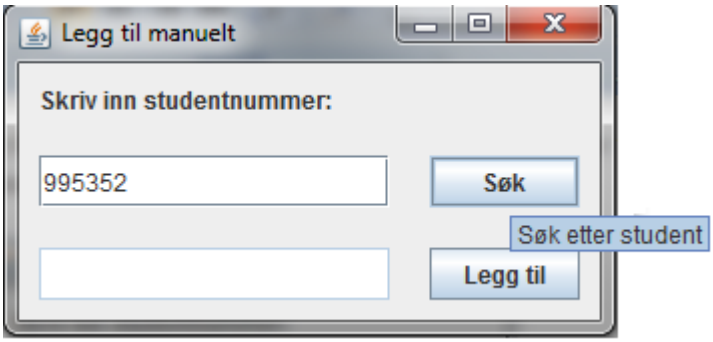



Fig. 3-9 Popp up vindu som kommer når man trykker på Mauelt-knappen i Orego-applikasjonen.

Hvis man finner studentene vil navnet på studenten komme opp i den nederste feltet, og man kan trykke på Legg til-knappen for å legge studenten inn i listen i hovedvinduet.

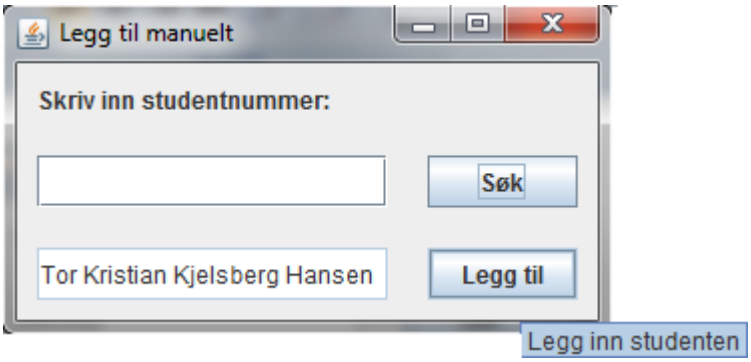



Fig. 3-10 Popp up vindu som kommer når man trykker på Mauelt-knappen i Orego-applikasjonen.

**Hent navn:**

Navnene til studentene bør alltid ligge i listen før man laster opp til Websiden/databasen. Denne funksjonen gjør slik at det hele kan fungere, selv om man ikke har løpende kontakt med internett. Noe som vi ser på som en veldig praktisk, funksjonell og ikke minst viktig funksjon.

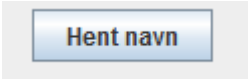



Fig. 3-11 Hent navn-knappen i Orego-applikasjonen.

Når man ikke har internett så vil man bare se studentnummeret til studenten som drar kortet komme frem i listen, som du ser et eksempel på under.


Fig. 3-12 Listen over studenter uten internett i Orego-applikasjonen.

Når man da har internett igjen kan man da trykke på Hent navn-knappen. Den vil da gå inn i LDAP-databasen å hente ut navnene til studentene, og legge de inn i listen sammen med studentnummeret. Her er et eksempel på det, og det er slik det skal se ut til vanlig.


Fig. 3-13 Listen over studenter i Orego-applikasjonen med internett.

**Registrer:**

Når vi trykker på Registrer-knappen så vil det dukke opp et vindu. Der vil du bekrefte at du er sikker på at du vil laste opp alle oppmøtte studenter.

Dette er den magiske knappen som lærerne trykker på da de vil registrere oppmøte.

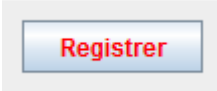



Fig. 3-14 Registrer-knappen i Orego-applikasjonen.


Fig. 3-15 Sjekkboks der du bekrefter at du vil laste opp oppmøtet.

Hvis du trykker ja, så vil studentene bli sendt til databasen og lagret der på riktig øving. Hvis opplastingen av listen med studenter var vellykket vil en melding vises i statusfeltet nederst i applikasjonen, og listen med studenter som ble sendt vil bli slettet fra Orego.

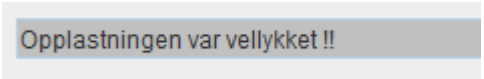



Fig. 3-16 Statusfeltet nederst i vinduet til Orego-applikasjonen.

**Status felt:**
Nederst i applikasjonen så finner vi et statusfelt, der kommer det opp beskjeder om hva som skjer. For eksempel om du ikke har internett, ikke fikk kontakt med serveren, eller at opplastningen var vellykket.

Dette er veldig fint siden det ofte kan informere om hva slags feil det kan være, hvis det ikke fungerer.

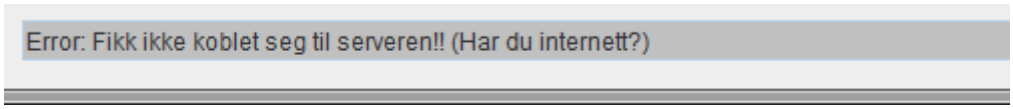



Fig. 3-17 Statusfeltet nederst i vinduet til Orego-applikasjonen.

### 3.3.2 Webmodulen

Resultatet av registreringen vill komme i form som to websider. Den ene er beregnet på lærere, mens den andre er beregnet på studenter. De bruker samme side for innlogging.

#### 3.3.2.1 Administrator (lærere)

Lærere vil få en link i Fronter der de kommer til en innloggingsside. Der må de logge seg inn med brukernavn og passord for å få tilgang til Websiden.

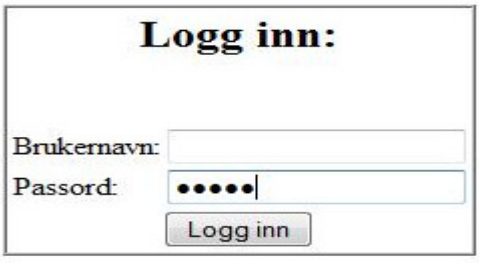



Fig. 3-18 Innloggingsskjerm for lærere i webmodulen.

Når de har logget inn så kommer de til hovedsiden der man har oversikt over alle studentene, og hvilke øvinger de har gjort i de to kullene. Når de er inne på denne siden har de også administratorrettigheter. Nederst på denne siden har vi tre knapper som utgjør administratorfunksjonene i dette produktet. De tre knappene er Fjern en student-knappen , Ferdige studenter-knappen, og Endre øvinger-knappen. Vi har diskutert utseende på denne tabellen, om det skal ligge vertikalt eller horisontalt. Vi kom frem til at det vil bli best om det ligger horisontalt med tanke på at det blir mange studenter etter hvert. Det vil etter hvert bli ganske mange studenter i tabellen. Derfor har vi lagt inn en funksjon slik at du kan se hvilken person og øving det er når du holder over et felt. Dette hjelper å holde lettere oversikt over hvem som er hvem, i tabellen over studentene. Vi har også lagt inn slik at hver trede linje i tabellen blir blå for å holde lettere oversikt når man blar seg utover i tabellen.

Orego

Obligatorisk
Registrering Av Oppmøte
Ved Høgskolen I Gjøvik

**PRU1002**

| Navn | Arbeids- og forflyttningsteknikk | Assistanse ved toalettbesøk | Hygieniske prinsipper |
|---|---|---|---|
| Hansen, Tor Kristian Kjelsberg (995352) | 10.05.10 | 10.05.10 | 10.05.10 |
| Holberg, Morten (070376) | 10.05.10 | | 10.05.10 |

**PRU2002**

| Navn | BHLR | Blodprøvetaking | Blodsukkermåling |
|---|---|---|---|
| Hansen, Tor Kristian Kjelsberg (995352) | 10.05.10 | | 10.05.10 |
| Holberg, Morten (070376) | 22.04.10 | 22.04.10 | 22.04.10 |

Ferdige studenter | Fjern én student | Endre Øvinger

Logg ut

Fig. 3-19 Hovedsiden med oversikt over alle studentene på Websiden.

**Fjern en student:**

Når man trykker på Fjern en student-knappen vil man bli sendt til siden under. For å slette en student skriver man inn studentnummeret til studenten på 6 siffer inn i tekstfeltet. Så kan man trykke på Slett studenten-knappen, og studenten vil bli fjernet fra systemet.


Fig. 3-20 Siden for å slette en student i webmodulen.

**Ferdige studenter:**

Når man trykker på Ferdige studenter-knappen vil man bli sendt videre til siden under. Her får man en oversikt over alle studentene i de to kullene, som har fullført alle øvingene. Det er en slettfunksjon her via knappen Slett. Før man trykker på den så må man skrive ”slett” i testfeltet ved siden av for at det skal fungere. Dette er for å øke sikkerheten. Når man trykker på Slett-knappen vil alle studenter som har fullført alle øvinger bli slettet. Det er derfor meget viktig at lærerne noterer hvem de sletter.

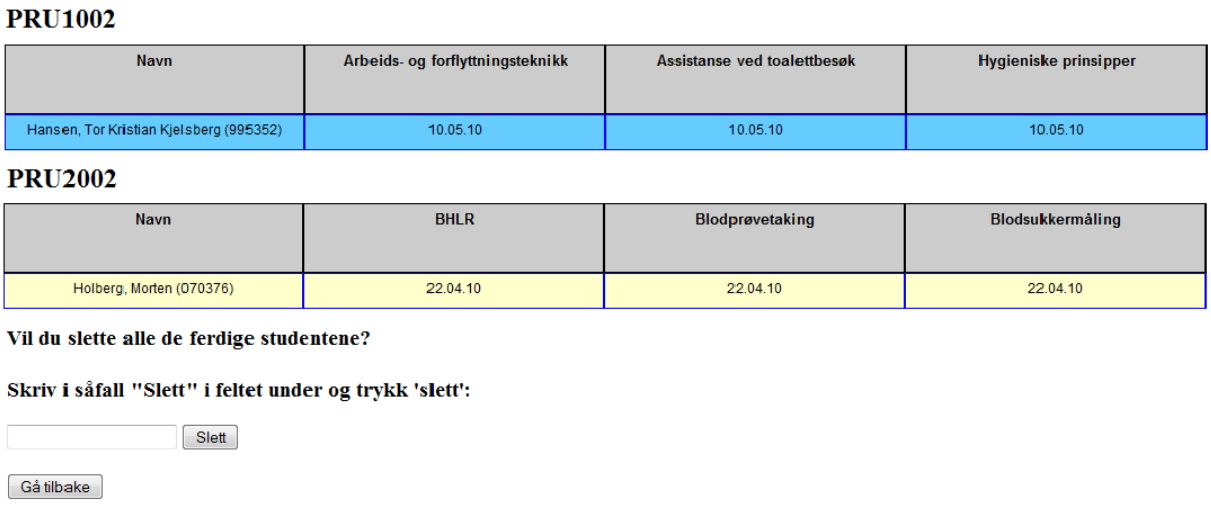



Fig. 3-21 Siden i webmodulen for å se og slette studenter, som har fullført alle øvingene.

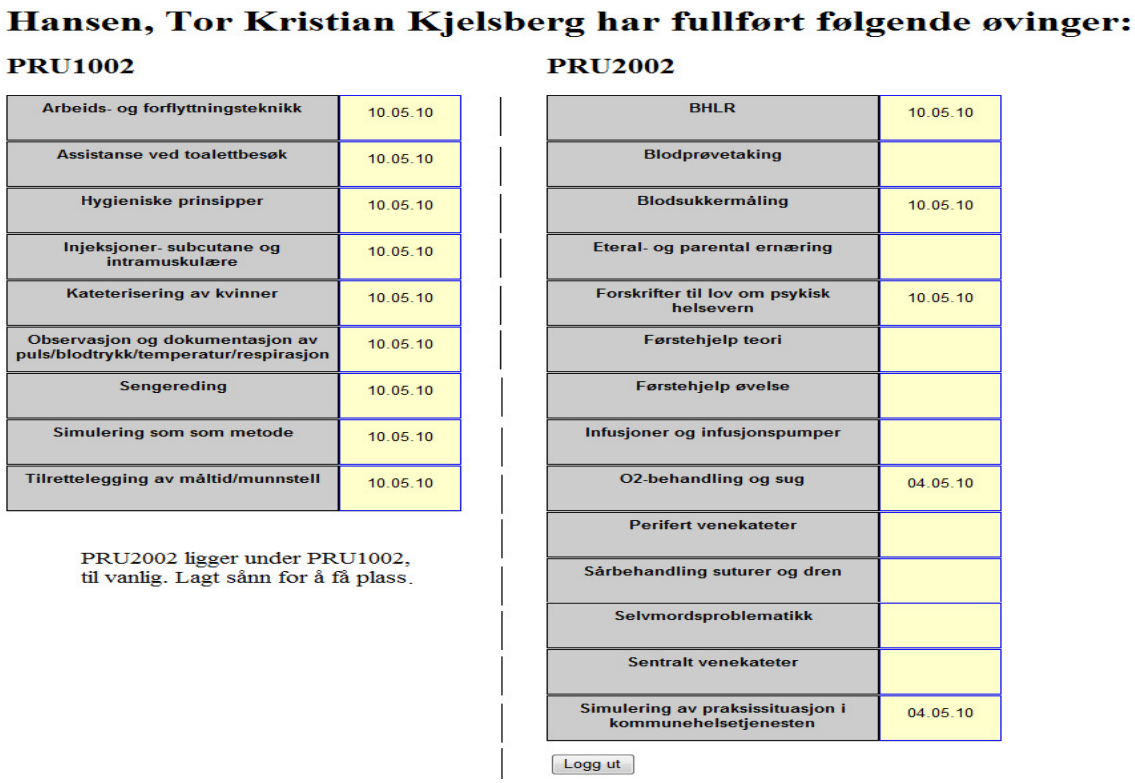
**Endre øvinger:**

Når man trykker på Endre øvinger-knappen så blir man sendt til siden du ser under. Her har vi delt inn PRU1002 og PRU2002 i to deler. Der har vi lagt inn øvingene til de to kullene i to ”dropp down bokser”. Dette for å gjøre det enkelt å velge hvilken øving man vil jobbe med. Etter å ha valgt en øving vil den valgte øvingen bli lagt inn i tekst feltet under, og da kan læreren bestemme om øvingen skal slettes eller endres på. Hvis læreren skal endre på øvingen så skriver man inn endringene på øvingen i tekstfeltet og trykker på Endre-knappen. Da blir navnet på øvingen automatisk oppdatert. Hvis læreren vil slette en øving så må riktig øving velges først. Den valgte øvingen vil igjen dukke opp i tekstfeltet under. Så er det bare å trykke på Slett-knappen, og øvingen vil bli fjernet. Ved en ny øving så skriver man inn navnet på den nye øvingen i tekstfeltet ved siden av Opprett-knappen, under riktig PRU. For så å trykke på Opprett og den nye øvingen vil bli lagt inn i systemet.


Fig. 3-22 Websiden for å endre, slette og opprette øvinger i webmodulen.

### 3.3.2.2 Studenter

Når studentene skal se oppmøtet sitt så må de gå via en link i Fronter. Denne sender dem videre til Websiden der de må logge seg inn med studentnummer og passord. Den samme siden som lærerne bruker. Forskjellen når de logger seg inn er at de bare får se en tabell over seg selv og sitt oppmøte, slik som på bildet under.



**Hansen, Tor Kristian Kjelsberg har fullført følgende øvinger:**

**PRU1002**

| | |
|---|---|
| Arbeids- og forflyttningsteknikk | 10.05.10 |
| Assistanse ved toalettbesøk | 10.05.10 |
| Hygieniske prinsipper | 10.05.10 |
| Injeksjoner- subcutane og intramuskulære | 10.05.10 |
| Kateterisering av kvinner | 10.05.10 |
| Observasjon og dokumentasjon av puls/blodtrykk/temperatur/respirasjon | 10.05.10 |
| Sengereding | 10.05.10 |
| Simulering som som metode | 10.05.10 |
| Tilrettelegging av måltid/munnstell | 10.05.10 |

PRU2002 ligger under PRU1002, til vanlig. Lagt sånn for å få plass.

**PRU2002**

| | |
|---|---|
| BHLR | 10.05.10 |
| Blodprøvetaking | |
| Blodsukkermåling | 10.05.10 |
| Eteral- og parental ernæring | |
| Forskrifter til lov om psykisk helsevern | 10.05.10 |
| Førstehjelp teori | |
| Førstehjelp øvelse | |
| Infusjoner og infusjonspumper | |
| O2-behandling og sug | 04.05.10 |
| Perifert venekateter | |
| Sårbehandling suturer og dren | |
| Selvmordsproblematikk | |
| Sentralt venekateter | |
| Simulering av praksissituasjon i kommunehelsetjenesten | 04.05.10 |

Logg ut

Fig. 3-23 Oversikt over en enkelt student sine øvinger i webmodulen.

Vi prøvde å gjøre det slik at studenten slipper å logge seg inn på Websiden. Ved at vi hentet ut studentens id fra Fronter, og at vi på den måten skulle få tak i øvingene til studenten. Men det viste seg å ikke være mulig, siden studentene har en annen id i Fronter. Denne løsningen hadde vært det optimale, men løsningen med innlogging fungerer like fint.

Vi diskuterte hvilken vei, horisontalt eller vertikalt, øvingene skulle ligge i oversikten for selve studenten. Vi kom frem til at uansett hvilken vei de lå så måtte studentene bla bortover eller nedover. Derfor la vi dem vertikalt siden det var litt penere utseendemessig.

### 3.4 Teknisk design

I vårt system har vi to moduler som jobber mot en database. Denne databasen er grunnlaget for hele systemet, og det er her dataene om oppmøte blir lagret. Her ligger hvilke øvinger, og når de forskjellige personene har gjennomført de. Her har vi delt opp det tekniske i to deler siden de to har forskjellig klientsituasjoner. Den ene modulen dreier seg om Websiden, der vi har klienter som kjører en tynn klient. En tynn klient vil si at et minimum av prosessering utføres på klientmaskinen, mens det meste av dataprosessering foregår på en eller flere sentrale servere. Den andre modulen er selve Orego-applikasjonen der vi bruker både en tynn og en tykk klient. Det vil si at en del av prosesseringen foregår i selve applikasjonen hos klienten, mens det andre blir utført på en sentral server.

#### 3.4.1 Systemarkitektur

Websiden bruker en firelags systemarkitektur med *presentasjonslaget*(klienten) øverst, et *applikasjonslag* på Websiden der alle forespørsler av data blir prosessert. I *sikkerhetslaget* i Serverprogrammet(Java) før databasen blir passordet sjekket, og data hentes via Sql-spørringer i databasen og returneres. I *datalaget* blir all data blir lagret i databasen(databaseserveren).

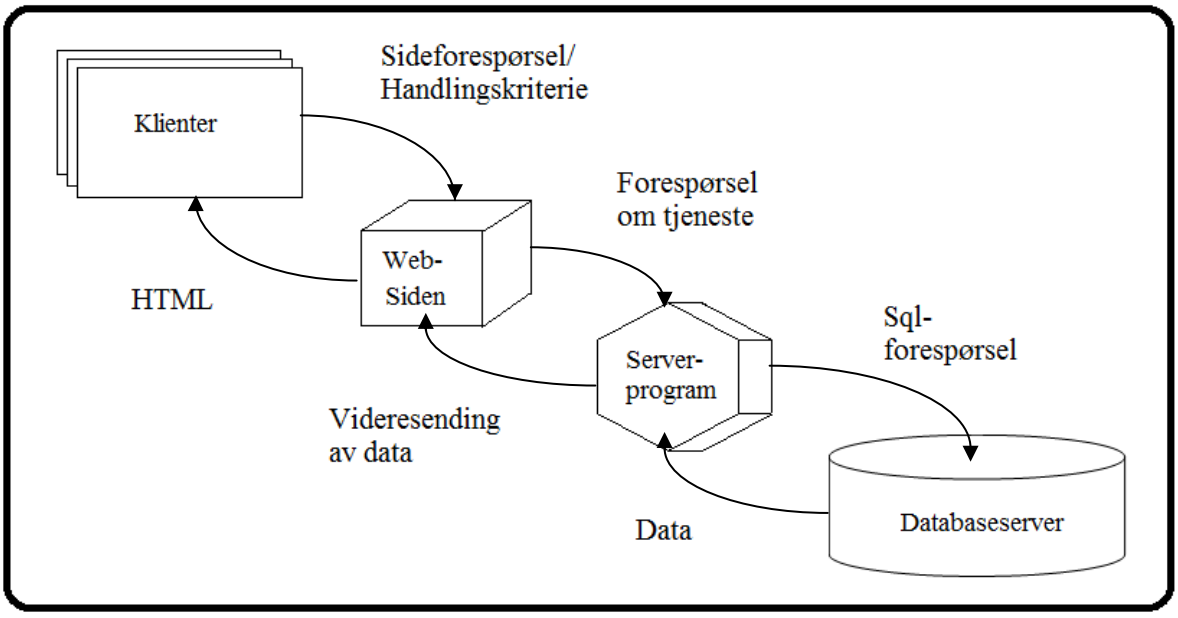



Fig. 3-24 Systemarkitektur for Websiden.

På figuren ovenfor ser du at en klient gjør en forespørsel fra Websiden mot Serverprogrammet. Videre vil Serverprogrammet sende en forespørsel til databaseserveren, som sjekker passord. Hvis det er i orden vil Serverprogrammet henvende seg til databasen med en Sql-spørring om data. Data ut i fra spørringen hentes inn og blir returnert til Serverprogrammet, og videre til Websiden. Der blir dataene vist frem som en HTML-side og klienten får se resultatet..

Orego-applikasjonen bruker en trelags systemarkitektur mot databaseserveren, med en tynn klient. Mot databaseserveren ligger *presentasjonslaget* i Orego-applikasjonen øverst, som sender en forespørsler til *sikkerhetslaget(Serverprogrammet)*, som igjen utfører Sql-spørringer møt databaseserveren i *datalaget*. Samtidig bruker applikasjonen en tolags systemarkitektur for å kommunisere mot LDAP-databasen, med en tykk klient. Der Sql-spørringene blir sendt direkte fra *presentasjonslaget* i applikasjonen til *datalaget* i LDAP-databasen.

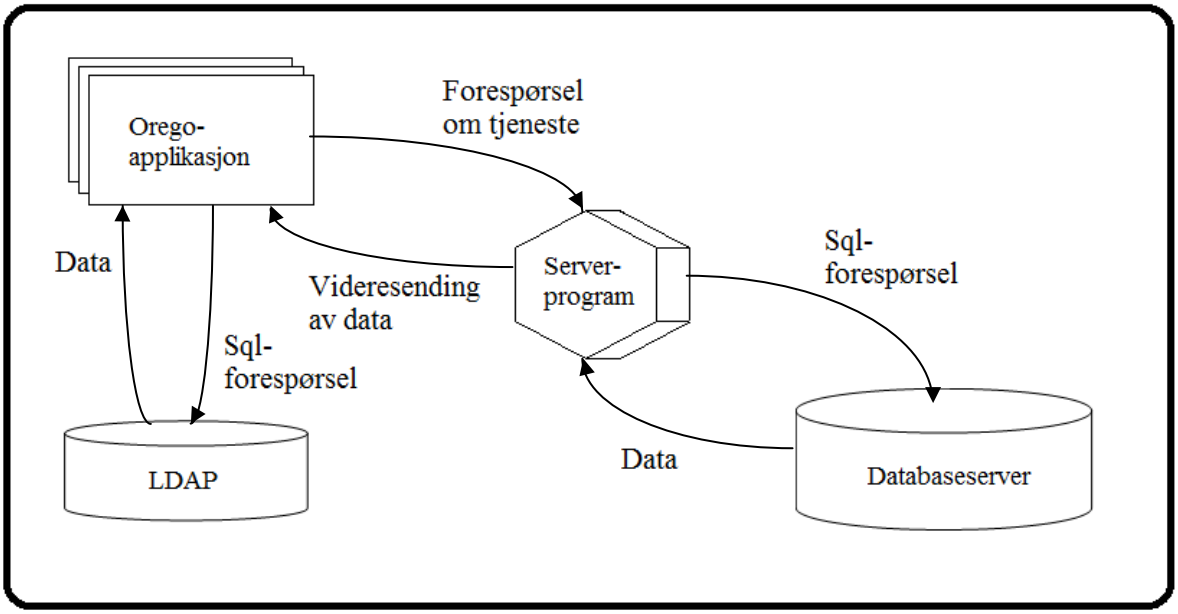



Fig. 3-25 Systemarkitektur for Orego-applikasjonen.

På figuren ovenfor ser man at Orego-applikasjonen sender en forespørsel til Serverprogrammet, som sjekker om passordet er riktig. Hvis det er i orden utførers en Sql-spørring mot databaseserveren. Her sendes det mer data til databasen enn det mottas. Mot LDAP-databasen sendes det en Sql-spørring direkte, som returnerer dataene som ble samlet inn.

### 3.4.2 Database design

Databasen er selve grunnlaget for applikasjonen vår, der all data om øvinger og oppmøte blir lagret. Når vi satte oss ned for å lage databasen hadde vi i tankene å gjøre den enkel, effektiv og presis. Vi brukte en halv dag på å lage databasen vi så for oss. Den har fungert perfekt siden, og vi mener den oppfyller alle våre kriterier. Vi synes det er en bra struktur, og vi tror ikke at det er mulig å forbedre den på noen måte. Fig. 3-24 viser databasestrukturen.

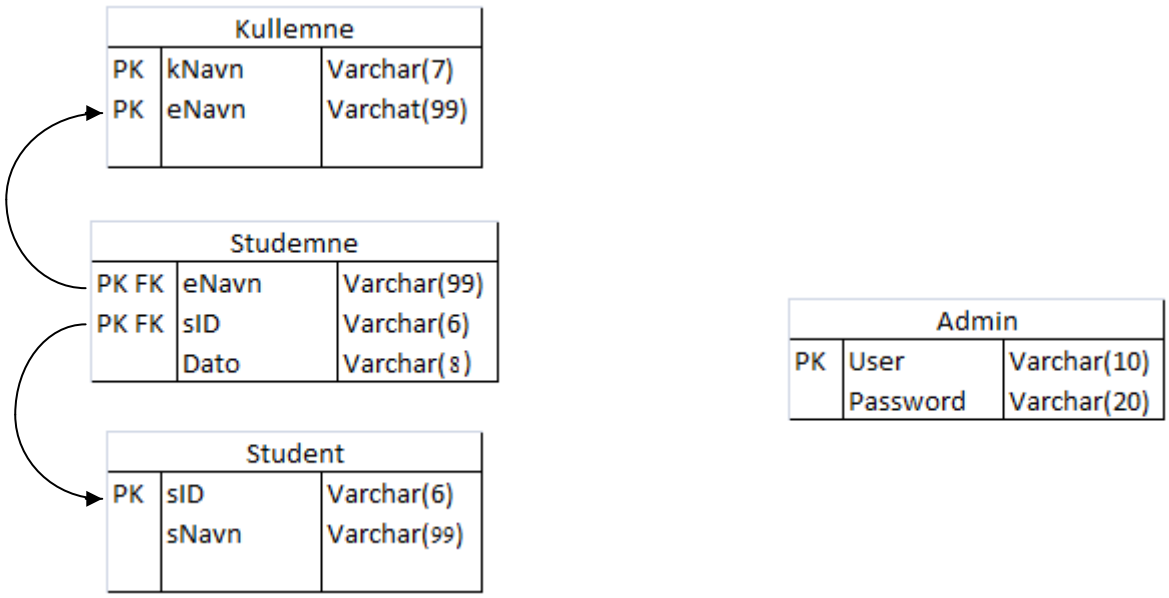



Fig. 3-26 Databasestruktur.

Under følger informasjon om de forskjellige tabellene:

- **Kullemne:**
  Her ligger navnene på de forskjellige kullene(kNavn) og deres tilhørende emner/øvinger(eNavn). Begge disse attributtene er primærnøkler mot hverandre, som vil si at man ikke kan legge inn et kull uten at det har en tilhørende øving. Det gjør det mulig å se under hvilket kull øvingene hørere til. Emnenavn er også primærnøkkel(PK) for fremmednøkkel(FK) til emnenavn(eNavn) i *Studemne.*

- **Studemne:**
  I denne tabellen ligger alle øvinger som studentene har tatt, og datoen de ble gjennomført. Emnenavn(eNavn) og studentenes id(sID) er primærnøkler (PK) mot hverandre, slik at vi kan se hvilke studenter som har hatt de forskjellige øvingene. Man kan ikke registrere et oppmøt uten at vi har et emne/øving og en student. Emnenavn(eNavn) er fremmednøkkel(FK) for primærnøkkelen(PK) til emnenavn i *Kullemne*, slik at vi kan se under hvilket kull øvingen og studenten ligger. Studentens id(sID) er også primærnøkkel(PK) for fremmednøkkelen(FK) til sID i *Student* tabellen.

- **Student:**
  Denne tabellen inneholder alle studentenes navn og studentid. Studentens id(sID) er primærnøkkel (PK) for fremmednøkkelen(FK) til sID i *Studemne.* Dette gjør det mulig å knytte navnet til studenten til de forskjellige øvingene, under de to kullene.

- **Admin:**
  Denne tabellen innholder attributtene brukernavn(User) og passord(Password). Disse brukes for å få full tilgang når man logger seg på Websiden til lærerne. Slik at de kan få en full oversikt over studentene, og muligheten for å endre/slette ting. User er primærnøkkel(PK) rett og slett fordi man trenger det for at databasen skal fungere.

I denne databasen kommer det til å ligge ganske mange studenter etter hvert. Derfor har vi prioritert å holde det så enkelt og oversiktlig som mulig. Slik at det ikke tar lang tid når oppmøtet skal vises frem i Websiden.

# 4. Implementering av systemet

## 4.1 Generell implementering

### 4.1.1 Utviklingsverktøy og utviklingsspråk

Vi har utviklet hele systemet i Java, ved unntak av Websiden hvor det i tillegg er blitt brukt JavaScript og HTML. Driveren til kortleseren installerer seg selv, og det den gjør er at kortleseren fungerer som et tastatur. Dette betyr kort og godt at de tegnene den leser blir omgjort til tastetrykk. Dette har vi valgt og ikke å gjøre noen endringer på.

Vi valgte Java siden dette språket er GUI-vennlig samt at vi synes det er oversiktlig og greit å kode i. Java har også mange ferdiglagede biblioteker som gjør det lett å kommunisere med andre språk og systemer. JavaScript er blitt brukt mest for å gi enkle meldinger til klientene på Websiden, mens HTML brukes til grensesnittet på Websiden.

Vi har utviklet i Eclipse fordi dette verktøyet samarbeider bra med Java, det gir god informasjon om funksjoner, og det holder orden på filer som trengs for å kjøre prosjekter. I tillegg implementerte vi noen ekstrafiler, som hjalp oss å kompilere Websiden og sette den opp på en Tomcat server. Dette var veldig behjelpelig siden det sparte oss for mye tid.

### 4.1.2 Eksterne kodebiblioteker

Vi har brukt to eksterne biblioteker i Java utenom de som ligger i standardpakkene. Det ene er ”*MySql connector java 5.1.10*”, som er en driver for JDBC (Java DataBase Connectivity). Dette er et bibliotek som gjør at vi enkelt kan kommunisere mellom Java og en MySql-database. Dette brukes av Serverprogrammet. Det andre biblioteket ”*jldap-4.3.jar*” brukes for å kommunisere med LDAP. Dette brukes både hos Orego og Websiden.

### 4.1.3 Kommentering av kode

All kode skal være kommentert og lesbar, så det skal være enkelt for andre å skjønne hva som skjer. Noen steder har vi også med Java-dokumentasjon, som beskriver funksjoner og deres parametere. All kommentering vil være på norsk.
Alle funksjoner og variabler skal ha engelske navn siden dette er god skikk og bruk, og siden alle ferdiglagde funksjoner allerede er på engelsk. Disse skrives med liten forbokstav, men hvis de er sammensatt av flere ord, skrives de påfølgende navnene med store forbokstaver (*eks. thisIsAnExample*). Konstanter skrives med kun store bokstaver og disse kan enkelt endres i ressursfilene som følger med systemet. Det er en ressursfil til hver del av systemet. Dette er meget gunstig om man for eksempel forandrer IP-adressen på serveren, ønsker nye passord eller lignende.

Om man vil utvikle flere funksjoner som skal samarbeide med Serverprogrammet, så pass på og legg denne funksjonen inn i ”enum Valg” siden Serverprogrammet kun vil gjenkjenne navnene som ligger der. Man sender ønsket valg som en linje med tekst.

### 4.2 Implementering av Orego

Orego er bygd opp ved at ”main” kun lager et objekt av typen Orego. Oregokonstruktoren tar seg av medhørende grensesnitt og fillesing. Funksjonen ”makeGUI()” setter opp komponenten på riktig plass med satt størrelse, og setter opp komponentenes funksjoner. Det er også her alle ”Listener”-ene blir laget. Orego er basert på grensesnittet sitt, og derfor kjøres mye kode etter hvert som man trykker på knappene. Når man ønsker oppdaterte øvingslister eller å registrere studentene blir koden for tilkobling til Serverprogrammet kjørt og avsluttet igjen. Dette skiller seg ut i forhold til for eksempel Websiden, siden denne behøver å være tilkoblet mye lenger.

Orego brukes til avlesing av studentkort. Dette er jo en av hoveddelene til systemet, så det er derfor meget viktig at dette blir gjort ordentlig. Siden kortleseren vi valgte leser

```
public void keyPressed(KeyEvent e) {
    String keyString = e.getKeyChar()+"";
    cardField.append(keyString);
    if(cardField.getText().length()==1){
        Thread t1 = new Thread(checkInput);
        t1.start();
    }
    if(cardField.getText().length() == 20) {
        newStudent(getStudentNumber(cardField));
        cardField.setText("");
    }
}
```

Fig. 4-1 Funksjonen for lesing av studentkort

studentkortene som tekststrenger er det meget viktig at disse strengene har korrekt lengde. Vi har derfor en tråd som starter etter første tegn blir lest inn (se Fig. 4-1). Den kjører funksjonen *’checkInput’* som nullstiller kortlesertekstfeltet etter 0,5 sekunder. Dette for å fjerne eventuelle tastetrykk i mellomtiden. Studentkortene inneholder 20 tegn, så derfor legges det til en ny student hver gang lengden på feltet blir 20. Funksjonen *’getStudentNumber(String field)*’ tar ut studentnummeret fra de 20 tegnene, og sender dette videre til *’newStudent(String id)’*.

Nå har vi studentnummeret, men vi ønsker også å vite navnet på studenten. Derfor sender vi studentnummeret til en av HiGs Ldap-servere ved hjelp av Ldap-biblioteket (se punkt 4.1.2) og koden som er vist her (se Fig. 4-2). Den sier at det skal gjøres en tilkobling på gitt adresse med gitt brukernavn og passord. Deretter skal det gjøres et søk hvor vi vil ha ut attributtet ”*cn*”, som er hvor studentenes navn ligger. Hvis læreren har tastet riktig studentnummer vil dette blir returnert. Om andre feil kommer det melding. Siden brukernavn og passord bestemmes av IT-tjenesten ser vi ingen grunn til andre ”*Exceptions*” enn at læreren ikke har internettilgang.

```
try {
    ldap.connect( ldapHost, ldapPort );
    ldap.bind( ldapVersion, loginDN, password.getBytes("UTF8"));
    LDAPSearchResults searchResults =
        ldap.search(baseDN,searchScope,searchFilter,new String[] {"cn"}, false);

    if (searchResults.hasMore()) {
        return(searchResults.next().getAttribute("cn").getStringValue());
    }
}catch (Exception e) { return("no net"); }
return "false";
```

Fig. 4-2 Viser hvordan vi kobler oss til Ldap og henter ut navnet vi søker

Da læreren har valgt øving og emne er det klart for registrering av oppmøtte. Dette er systemets viktigste oppgave og nettopp derfor bør det ikke kunne være mulig at noe går feil. Når en lærer trykker på Registrer-knappen vil han få en melding (se Fig. 3-15) hvor han må

```
private void regStuds() {
    Connection c = new Connection();
    try {
        if(c.gettingAccess()) {
            c.sendLine("REGISTRER_STUDENTER");
            c.sendLine(topicBox.getSelectedItem().toString());
            c.sendLine(practiceBox.getSelectedItem().toString());
            c.sendLine(date.getText());
            c.sendList(studListModel);
            setStatusField("Opplastningen var vellykket !!");

            studListModel.clear();
            antStudents.setText("");
            delFile(studentfil);
        }else setStatusField("Error: Feil opplastningspassord");
    } catch (Exception e) {
        setStatusField("Error: Fikk ikke koblet seg til serveren!! (Har du internett?)");
    }
    c.endConnection();
}
```

Fig. 4-3 Viser kommunikasjonen med Serverprogrammet når vi registrerer studenter

bekrefte at han vil registrere oppmøtet. Hvis valget er ja vil funksjonen 'regStuds()' kjøres. Denne funksjonen tar først og oppretter en tilkobling til Serverprogrammet. Deretter prøver vi å få tilgang med et passord som ligger i ressursfilen. Om det gikk greit sier vi til Serverprogrammet hva vi ønsker å gjøre. Deretter sendes emne, øving, dato og studentene linje for linje. Hvis alt gikk greit får man melding om dette, listen med studentene slettes, og den midlertidige filen med alle studentene slettes. Hvis noe gikk galt vises det en feilmelding i statusfeltet nederst i Orego-applikasjonen.

### 4.3 Implementering av Serverprogrammet

Serverprogrammet er et program som skal ta i mot og sende informasjon mellom Orego/Websiden og databasen. Dette for å skape mer sikkerhet, slik at ikke uvedkommende kan komme inn i databasen eller sende informasjon dit. Siden dette skal kjøres som en service har det intet grensesnitt.

```
if(accessGranted(input = mottaLine())) {
    db = new DatabaseHandler();
    while((input = mottaLine())!=null){
        keepConnected=true;
        switch (Valg.valueOf(input)){
          case REGISTRER_STUDENTER  :   regStuds();          break;
          case OPPDATER_EMNELISTER  :   sendSubLists();      break;

          /* Ligger mange flere handlinger her
              siden denne switchen takler Oregos
              og Websidens ønsker                  */

          default: sendLine(input + " er ikke mulig !?");
        }
    }
} else sendLine("error: feil passord!");
```

Fig. 4-4 Serverprogrammets store 'switch'

For å koble seg til Serverprogrammet må det sendes en tekststreng på følgende måte: "*pass: <passordet>*". Man vil da motta "*ok*" og du kan da be om ønsket handling. Det er også fullt mulig å be om flere handlinger av gangen. Fig. 4-4 viser hvordan Serverprogrammet takler forespørsler etter en vellykket innlogging. Den leser ønsket handling som en linje med tekst og sjekker i 'enum Valg' om handlingen finnes. I så fall gjøres denne handlingen, ellers kommer det melding. Vi bruker en 'while' siden det kan bes om flere handlinger, så lenge klienten ikke har koblet fra eller klienten har brukt for lang tid. 'keepConnected' er en variabel som forteller at tilkoblingen fortsatt brukes, og vil i utgangspunktet være negativ. Men hver gang en klient ønsker en handling vil denne variabelen settes til positiv, og tråden som ønsker å lukke tilkoblingen må prøve på nytt etter en gitt periode.

For hver applikasjon som kobler seg til Serverprogrammet vil det også bli opprettet en tilkobling til databasen. Denne tilkoblingen blir åpen helt til applikasjonen er ferdig. Det kreves et brukernavn og passord for å få tilgang til databasen. Dette for å nekte uvedkommende tilgang. Om disse dataene er korrekte vil Serverprogrammet bli en bruker på lik linje med manuell innlogging.

```
public String findStudent(String studnr) {
    try {
        st=conn.prepareStatement("SELECT sNavn FROM student WHERE sID = ?");
        st.setString(1,studnr);
        rs=st.executeQuery();
        if(rs.first())
            return( rs.getString("sNavn"));
    }catch (Exception e) {
        e.printStackTrace();
    }
    return "";
}
```

Fig. 4-5 Viser et eksempel på hvordan vi kommuniserer med databasen

I fig. 4-5 vises hvordan vi søker etter en student i databasen ved hjelp av et studentnummer. 'st', 'conn' og 'rs' er alle variabler fra klassene "Connection", "PreparedStatement" og "ResultSet", som er implementert ved hjelp av et databasebibliotek (se punkt 4.1.2). Disse brukes som vist over, hvor 'conn' er det som ønsker en handling, 'st' er den som utfører den, og 'rs' er hvor eventuelt resultat av handlingen blir lagt. Her ser vi om vi får noe resultat, og så returnerer vi dette, som eventuelt er navnet på en student. Vi bruker spørsmålstegnet når vi vil sende med variabler. Variablene blir ikke definert under selve spørringen, men før den skal gjennomføres. Ved sending av flere variable er det bare å gjenta setning fire i koden, og telle opp for hver variabel.

### 4.4 Implementering av Websiden

Websiden er den delen av systemet som skal vise oss hva som ligger i databasen. Det skal vise alle studenter med øvinger for administrator, og studentene skal selv kunne se sine egne godkjente øvinger. Dermed var det naturlig å lage en innloggingsside. Den er kodet slik at den vil først se i databasen om brukernavn og passord stemmer overens med administratorens. Om dette ikke er tilfelle vil det bli gjort et søk i LDAP, om dette er en student med riktig tilhørende passord. Om begge tilfellene viser seg å være feil, vil det komme en melding om dette.

Om brukernavn og passord er korrekt vil brukeren bli logget inn til siden han ønsker. Det vil også bli startet en tidtager, som passer på at brukeren ikke blir stående tilkoblet for lenge uten at han gjør noe. Denne samarbeider for øvrig med tråden i Serverprogrammet.

Websidene inneholder, som sagt, Java-kode, og denne kjøres hver gang man laster en av sidene (se fig. 4-6). Denne koden vil ikke være synlig i klientenes nettlesere. I koden vår her forsøkes det å finne navnet på brukeren som er utgir seg for å være innlogget. Om han ikke finnes vil han bli sendt til innloggingssiden. Ettersom dette er en av sidene som kun administrator har tilgang til, vil det også sjekkes om brukernavnet stemmer overens med hans. Om det viser seg at det er administrator vil konstruktoren til "WebManager" kjøre, som henter oppdaterte lister på øvinger og studenter. Deretter ber vi om navnene på de to

emnene, og så skrives disse samt øvinger og studenter ut.

```
<%@ page contentType="text/html; charset=UTF-8" language="java" import="java.util.ArrayList" %>
<%@ page import="no.hig.orego.WebManager"%>

<%  String user = (String)request.getSession().getAttribute("user");
    if(user != null && WebManager.checkUser(user)){
        new WebManager();
        String pru1 = WebManager.getPruTitle("emne1");
        String pru2 = WebManager.getPruTitle("emne2");%>
```

Fig. 4-6 Viser hvordan vi bruker Java-kode til å finne brukeren av Websidene

Websidene har fått endelsene ".jsp" siden vi skal kunne bruke både Java og JavaScript på dem. Dette gjør at vi kan skrive kode som i fig. 4-6, for eksempel "<% (ønsket Java-kode)%>". Disse symbolene forteller at det nå skal kjøres Java-kode, som ikke vil returnere noe direkte. Om man bruker "<%= (Java-kode) %>" vil man være nødt til å returnere noe. Dette har vi brukt når vi blant annet skriver ut øvingslister, studentlister og lignende direkte inn i HTML-koden.

Når vi for eksempel ønsker å gi en melding til brukeren bruker vi JavaScript. Da forteller vi først nettleseren at det kommer JavaScript, før vi sier hva som skal gjøres. Her ligger JavaScript-koden inne i Java-kode, og det fungerer fint, selv om utviklingsverktøyet vårt reagerer. Den gjør det ettersom skriptene ligger feil plassert, og den tror de vil kjøres med en gang Websiden lastes. Men takket være Java-koden kjøres ikke JavaScript-koden før den skal.

```
    <%}else{%>
        <script language="javascript">
            alert("Feil brukernavn eller passord");
        </script>
    <%}
}else{%>
    <script language="javascript">
     alert("Begge feltene må være utfylt");
    </script>
<%}
```

Fig. 4-8 Viser et eksempel på JavaScript og Java blandet sammen

# 5. Installasjon

## 5.1 Systemkrav og forberedelser

### Server

For at systemet vi har utviklet skal fungere bør systemet bli installert på en datamaskin med et Windows operativsystem. Deretter skal følgende programmer installeres (disse vil bli å finne på den vedlagte CD-en):

- MySql Community Server. (MySql Server 5.1)
    - MySql GUI Tools 5.0 (Anbefales sterkt å ha installere siden installasjonsguiden bruker dette, men er dog ikke nødvendig for at systemet skal virke senere)
- Apache Tomcat6.
- Java™ 6 fra java.com. Utgiver Sun Microsystems. (Anbefaler siste versjon)

### Klient

Datamaskinene lærerne bruker er ikke avhengig av noe spesielt operativsystem. Det eneste de trenger er følgende:

- En oppdatert nettleser
- Java™ 6 fra java.com. Utgiver Sun Microsystems. (Anbefaler siste versjon)

## 5.2 Installasjon og oppsett av databaseserver

### 5.2.1 Installasjon av databasen

1. Kjør installasjonsfilen. Velg ”typical” og ønsket plassering av programmet. Etter programmet er installert fjern haken ved registrering av produktet og hak av for konfigurering av serveren. (Om det velges og ikke å konfigurere nå, vil du finne igjen ”MySql Server Instance Config Wizard” i startmenyen)
2. Velg ”Detailed Configuration”. (Om programmet er blitt installert før velg ”Reconfigure Instance”, for så deretter punkt to.
3. Velg ”Server Machine”.
4. Velg ”Multifunctional Database”.
5. Det anbefales å la guiden velge plassering av databasen, men om man ønsker den plassert et bestemt sted velg ”Modify”. Ellers bare fortsett.
6. Velg ”Manual Setting” og tast inn 25, siden vi ikke tror pågangen blir så stor.
7. Velg ”Enable TCP/IP Networking”, ”Enable strict mode” og ”Port Number” skal være 3306. (Obs! I enkelte tilfeller bør det også velges å slå av brannmuroverv̊åking for denne porten, men ligger Serverprogrammet og databasen på samme datamaskin bør det gå fint uten)
8. Velg ”Manual Selected Default Character Set/ Collation” og velg karaktersettet “utf8”.
9. Trykk fortsett. Om man ønsker å enkelt logge seg på database velg ”Include Bin Directory in Windows PATH”.

10. Kryss av for “Modify Security Settings”, og skriv inn nytt root passord etter ønske. (Obs! Dette bør ikke være brukeren Serverprogrammet får tildelt)
11. Klikk på ”Execute” for å installere.

### 5.2.2 Opprettelse av database

1. Start ”MySQLAdministrator” (finnes i pakken GUI Tools som vi anbefalte i punkt 5.1).
2. Logg deg inn som root-bruker med tilhørende valgte passord.
3. Gå til ”Restore Backup”, og velg ”Restore Backup File”. Finn så filen ”OregoDatabase.sql” på vedlagt CD.
4. Trykk så ”Start Restore” og databasen legges inn.

### 5.2.3 Opprett brukerkonto

1. Start ”MySQLAdministrator” (finnes i pakken GUI Tools som vi anbefalte i punkt 5.1).
2. Logg deg inn som root-bruker med tilhørende valgte passord.
3. Velg ”User Administration”.
4. Velg ”Add New User”.
5. I ”MySQL User” skrives ”orego” og ”Password” skrives ”orego123”. (Obs! Om en annen bruker eller annet passord ønskes, så husk å endre dette i ressursfilen til Serverprogrammet.
6. Gå til ”Schema Privileges”. Her velger du ”orego” og gir brukeren full tilgang ved hjelp av de dobbelte pilene som peker mot venstre. Brukeren skal nå ha fått ”Assigned Privileges”.
7. ”Apply changes” for å opprette brukeren.

## 5.3 Installasjon av Apache Tomcat6 og oppsett av ”OregoWeb.war”

### 5.3.1 Installasjon av Tomcat6

1. Kjør installasjonsfilen. Tomcat6 skal nå være installert.
2. Deretter åpner man en nettleser og går til adressen http://localhost:8080/manager/html.
3. Bla nedover siden til man ser ”WAR file to deploy” og trykk ”Bla gjennom”.
4. Finn så ”OregoWeb.war”. (Obs! Om endringer skal gjøres må ”OregoWeb.war” legges på harddisken)
5. Trykk ”Deploy”. Webserver skal nå være installert.

### 5.3.2 Oppsett av ”OregoWeb.war”

1. Gå inn på den vedlagte CD-en og kopier ”OregoWeb.war” til ønsket plassering.
2. Finn igjen den ønskede plassering og høyreklikk på ”OregoWeb.war”.
3. Velg ”åpne i” og bruk ”WinRAR”, ”WinZIP” eller lignende program.
4. Velg ”WEB-INF”, så ”classes”, og deretter ”resources”. Åpne så ”resources.properties”

5. Gjør dine ønskede endringer (for eksempel å endre IP-adressen til Serverprogrammet) og trykk deretter lagre og avslutt.
6. Gjør punkt 5.3.2 valg tre, fire og fem på nytt.

### 5.4 Installasjon av Java

1. Kjør installasjonsfilen.
2. Følg instruksjonene og Java skal bli installert.

### 5.5 Oppsett av Serverprogrammet

1. Gå inn på den vedlagte Cd-en og kopier mappen Serverprogrammet til ønsket plassering, og åpne den.
    a. Åpne filen ”ServerProgrammet.conf”.
    b. Skriv inn ønsket IP-adresse til databasen (default er ”localhost”)
    c. Velg lagre og avslutt.
2. Åpne Kjør fra Startmenyen (Windows-knapp + ”R”), og skriv kommandoen ”cmd”.
3. Finn igjen hvor du plasserte mappen Serverprogrammet.
4. Kjør kommandoen ”install.bat”. Serverprogrammet skal nå kjøre som en service. Om noe gikk feil, pass på at du er administrator og prøv igjen.
5. For å stoppe servicen, kjør kommandoen ”remove.bat”

### 5.6 Oppsett av Orego

1. Gå inn på den vedlagte CD-en og kopier ”Orego.jar” til ønsket plassering.
    a. Høyreklikk på ”Orego.jar”, og velg ”åpne i” for eksempel ”WinRAR”, ”WinZIP” eller lignende utpakkingsprogram.
    b. Velg så ”resources”-mappen, for deretter å åpne ”resources.properties”.
    c. Finn ”SERVERHOST” og skriv inn den ønskede IP-adressen.
    d. Velg lagre og avslutt.
2. Høyreklikk på ”Orego.jar”, trykk ”Send” og velg ”Skrivebord (snarvei).

### 5.7 Tips og råd

1. Det anbefales sterkt å ha Websiden, Serverprogrammet og databasen på samme datamaskin. Dette for å slippe og endre på ressursfilene.
2. Siden Orego skal ligge på mange datamaskiner er det lurt å gjøre punkt 5.6 på en datamaskin, og så videreføre denne endrede versjonen. Dette for å slippe og endre ressursfilen mer enn en gang.
3. Man kan endre passord, IP-adresser og lignende i ressursfilene, men pass på å kontrollere og oppdatere alle ressursfilene.

# 6. Testing

**Test strategi:**

Vi har nå utført en rekke testing siden vi ønsket et mest mulig stabilt system. Vi testet hver enkel modul etter hvert som de ble laget, ettersom vi jobbet etter Fossefallsmetoden. Deretter satte vi flere og flere deler sammen mens vi testet hvordan det oppførte seg.

Når vi arbeidet med en bestemt del har vi også hele tiden kompilert og prøvd ut koden vi endret på. Dette gjorde at vi fikk rettet opp småfeil underveis, og slapp å endre på alt for store deler om gangen. Typiske feil var at når vi lagde en ny funksjon, og endret litt gammel kode, så ville ikke noen av de gamle funksjonene fungere lenger.

Vi får først muligheten til å teste programmet fult ut den. 20 mai. Dette er også samme dag som oppgaven skal leveres for kopiering! Vi har derfor måtte testet det hele en rekke ganger, for å være sikre på at det skal fungere fint. Skulle noe allikevel skjære seg vil vi gjøre vårt beste for å ordne disse feilene. Vi fikk forøvrig testet Orego i en av veilederes timer.

**Testing av grensesnitt:**

Vi har brukt en god del tid på testing av grensesnittet siden dette skal være et oversiktlig og enkelt program å bruke. Vi delte det opp slik at vi testet grensesnittet til Orego først, og Websidegrensesnittet senere. Det største problemet med Orego var å få komponentene til å holde samme størrelse når man gjorde forskjellige ting. Dette tok en god del av testingen. Websiden hadde ikke dette problemet ettersom det får en fast størrelse hver gang sidene kjøres. Problemet her ble å få alle nettleserne til å oppføre seg likt og ikke la dem få sette egne størrelser på tabellene våre.

**Testing av database:**

Vi har aldri hatt noen spesielle problemer med databasen når vi har jobbet mot den ved hjelp av Serverprogrammet, men den har hjulpet oss å finne feil i annen kode. Det eneste som viste seg å være litt brysomt var endring av såkalt ”metadata”. Primærnøkler, endring av karaktersett og lignende var ting vi fikk feilmeldinger på når vi prøvde å endre dem. Vi løste det ved at vi lagde en helt ny databasen med korrekt innhold.

**Parallelltesting**:

Etter at vi hadde testet Orego og Websiden hver for seg, så var det tid for å samkjøre de, og teste programmet som en helhet. Når vi gjorde det så oppdaget vi en del småpirk og feil som måtte ordnes. Dette tok mye mer tid og jobb enn det vi først hadde antatt. Et problem som skapte mye kaos var aspektet omkring likt karaktersett mellom modulene. Vi trodde først at det kun var databasen som ikke taklet spesielle tegn, men når vi satte karaktersettet i databasen til ”utf8” løste dette bare en del av problemet. Det viste seg dermed at Java-koden også måtte settes til å skrive og lese i ”utf8”. Websiden hadde vi allerede satt siden vi brukte norsk språk på den. Dette var en typisk bagatell som vi vil nå kunne møte bedre forberedt i fremtiden.

**Brukertesting:**

Mesteparten av brukertesting har foregått som lærer, som har alle administrasjonsrettighetene på hjemmesiden. De funksjonene vi har testet mest har vært sletting av studenter, samt endre, slette og legge til nye øvinger. En student skal jo kun kunne se sine egne øvinger, og ettersom funksjonaliteten for det er veldig enkelt har vi fokusert mye mer på lærerdelen.

Vi fant en feil når vi ønsket å slette alle ferdige studenter, altså de som har vært ferdig med alle øvingene i ett emne. Problemet gjaldt de studentene som hadde øvinger i begge emnene, ettersom han ikke skal slettes fra det emnet han ikke er ferdig med. Dette lot seg ordne ganske fort ved hjelp av noen ekstra sjekker i koden.

En annen feil som dukket opp var når vi ville endre navnet på en øving som studentene hadde gjort. Dette reagerte databasen på siden vi hadde satt øvinger som primærnøkler, og studentenes fullførte øvinger var fremmednøkler til disse. Vi måtte dermed skrive om koden slik at det først ble laget en helt ny øving. Deretter oppdaterte vi alle øvinger som hadde en fremmednøkkel til den gamle øvingen med den nye. Til slutt slettet vi den gamle øvingen.

**Sikkerhetstesting:**

Systemet er kodet slik at Orego og Websiden går gjennom Serverprogrammet når de ønsker kontakt med databasen. For at Orego og Websiden skal få kontakt med Serverprogrammet kreves et passord. Orego-databasen krever et brukernavn og passord for å gi tilgang, som Serverprogrammet er implementert med. En utenforstående kan derfor ikke komme seg direkte inn i verken Orego-databasen eller i Serverprogrammet uten videre.

Ved innlogging på Websiden bruker studentene samme brukernavn og passord som de gjør når de logger seg på HiGs nett, og sin egen e-postadresse som er utlevert av skolen. Dette gjør det lettere for studentene å huske brukernavnet og passordet sitt. Lærerne bruker en felles profil som sjekker brukernavn og passord opp mot Orego-databasen. Det er viktig med sikkerhet i systemet for å bevare personvern, og ikke minst å få tilgang til administratorfunksjonene. Det er spesielt viktig at ingen utenforstående får tilgang til administratorfunksjonene, siden disse kan fjerne/legge til studenter og deres oppmøte.

# 7. Sluttdiskusjon

## 7.1 Diskusjon og drøfting av valg og resultater

### 7.1.1 Valg at systemutviklingsmodell

Ved dette prosjektet diskuterte og vurderte vi de forskjellige utviklingsmodellene, samt at vi forhørte oss hos veileder om hva slags modell som ville passe oss best.

Etter å ha diskutert oss to i mellom en stund så vurderte vi å bruke den smidige modellen Scrum. Dette fordi den er veldig direkte og utviklingsvennlig. Når man jobber med Scrum har man først et møte, for så å jobbe i en sprint på så og så mange dager før man igjen har et møte. Vi tenkte å ha sprinter på 2 uker som passet perfekt med vår møteplan med arbeidsgiver. Scrum har lett for å tilpasse seg for endringer, den har høy produktivitet, og stor grad av kundeinvolvering. Dette syntes vi virket positivt og kanskje noe for oss, siden vi ikke visste nøyaktig hvordan det endelige resultatet vil bli. Problemet når man bruker Scrum modellen er at det ikke fastsettes noen krav, eller resultat på forhånd. Dette er ikke positivt med tanke på at dette er en Bacheloroppgave, hvor vi er mer eller mindre avhenging av et resultat. Siden vi er studenter har vi dessuten mest erfaring med å jobbe etter frister.

Vi hadde et møte med veilederen vår Frode i starten av semesteret. Der han kom med et tips om at vi burde se på Fossefallsmodellen. Fossefallsmodellen legger vekt på at det skal være planlagt, godt strukturert, utprøvd, vedlikeholdbart og godt dokumentert. Som utviklere i Fossefallsmodellen kan vi også komme med forbedringer og forslag under veis. Fossefall egner seg godt for litt større prosjekter over lengre tid. Dette passer ganske bra til oppgaven vår, og vi tror at denne modellen kan hjelpe oss en hel del. Da vet vi akkurat hva vi skal jobbe med, når vi skal jobbe med det, og når det skal være ferdig. Den avsetter også tid til integrering av andre systemer, noe som er bra med tanke på vår kommunikasjon med LDAP. Den tar også høyde for drifting og vedlikehold, som er meget viktig ettersom IT-tjenesten har sine egne krav til hvordan dette gjøres. Det eneste som er litt negativt er at vi ikke vil se hvordan det hele blir før vi er ferdig. Dette er en liten risikofaktor, men vi tror alle de positive tingene med denne modellen vil hjelpe oss og veie opp for det, slik at det ikke blir noe problem. Basert på dette har vi derfor valgt å benytte oss av Fossefallsmodellen under utviklingen.

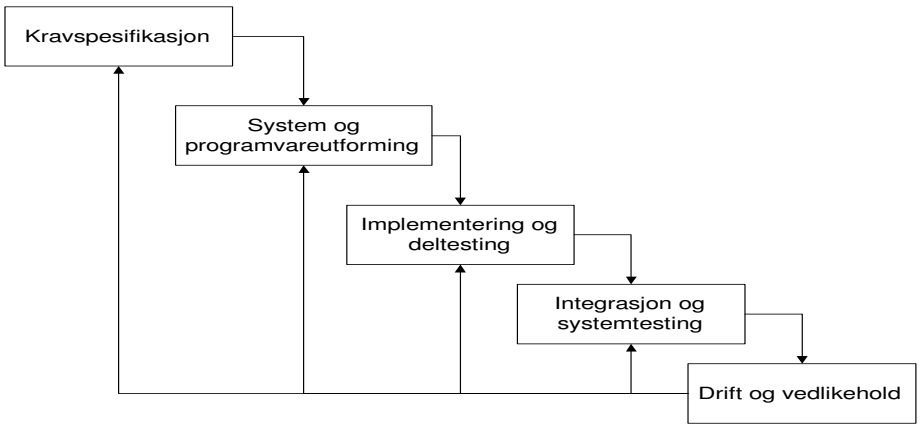



Fig. 7-1 Fossefallsmodellen.

### 7.1.2 Resultat

Da vi startet med prosjektet hadde vi et veldig åpent utgangspunkt. Vi visste ikke om det var mulig å integrere systemet vårt i Fronter. Vi hadde ingen spesifikasjoner på hvordan utseende for fremmøtet eller registreringsmodulen skulle være. Det eneste vi fikk vite var at det skulle være mulig å slette studenter, endre/slette/opprette øvinger, og at studentene bare skulle få se sitt eget oppmøte. Vi fokuserte på dette fra starten av, men det gav oss et rimelig vidt utgangspunkt å jobbe med.  Etter en del diskusjon med oppdragsgiver og utforming av design og ideer, kom vi frem til noe mer fast å jobbe ut fra. Vi fikk bestilt kortleseren tidlig som var til stor hjelp, og skrevet en midlertidig kravspesifikasjons som hjalp oss på vei. Det var mye frem og tilbake med Fronter, og det var vanskelig å få til en dialog. Det viste seg også vanskelig å integrere systemet vårt i Fronter, eller i det hele tatt å hente ut informasjon som vi trengte derfra. Vi fikk ut en unik id, men det viste seg at iden vår i Fronter ikke var den samme som på HiG. Derfor ble dette alternativet utelukket, noe som vi hadde en antagelse om fra starten av.

Det ferdige systemet implementerer alle de funksjoner i kravspesifikasjonen, og i tillegg har vi en del ekstra funksjonalitet, og noe nødvendig. Verken vi, oppdragsgiver eller veileder kunne forutse disse da vi skrev den første kravspesifikasjonen. Så en del av systemet har vi måtte tenkt oss til og utviklet under veis. Det endelige systemet oppfyller alle krav og litt til, så vi tørr å påstå at vi har løst oppdraget bra.

Etter hvert som vi implementerte delene av systemet oppdaget vi ny funksjonalitet som vi trengte, samt noe som hadde vært fint og hatt. Vi inkluderte den funksjonaliteten vi trengte, og vi la til noe ekstra som vi syntes forbedret det helhetsmessige perspektivet. Noen eksempler på dette var innloggingen, oversikten over alle ferdige studenter, hvordan Orego fungerte uten internett, og en bra måte å lese av kortene ved hjelp av kortleseren. Under dette arbeidet lærte vi mye om LDAP, innlogging, JavaScript, og kortlesere.

Systemet er i dag relativt robust, og gjør den jobbet det er laget for å gjøre. Det har et stort utviklingspotensial, og med noen endringer går det an å bruke det i alle fag og undervisninger.

Vi har laget et system som vi er ganske fornøyde med, og vi antar at det kvalitetsmessig er bra nok til daglig bruk ved sykepleieravdelingen på HiG. Vi har demonstrert programmet for lærerne og de studieansvarlige på sykepleieren. Da fikk vi gode tilbakemeldinger, og de har planer om å ta det i bruk allerede fra høsten av. Noe som tyder på at de er fornøyde med produktet, og da føler vi at vi har gjort en bra jobb og oppnådd et bra resultat.

## 7.2 Kritikk av oppgaven

Det er mye vi har ønsket å kunne gjøre i løpet av oppgaven som enten ikke har vært mulig, eller at vi ikke har hatt tid til å få det gjennomført.

I starten lekte vi med tanken på en trådløs kortleser. Det hadde optimalisert programmet litt, men ikke nevneverdig. Vi hadde en tanke at lærerne kunne sendt kortleseren rundt i klasserommet, mens de underviste. Men ettersom læreren bør stå og følge med på at de som drar kortet blir lagt inn i listen, gikk vi bort fra dette. Det ville også vært enklere for en

student å registrere flere studenter om kortleseren hadde gått rundt i klasserommet. Det hadde vært en risikofaktor, men en trådløs kortleser hadde nok vært lettere å håndtere. Når vi allerede har brukt tid og penger på en USB-kortleser som gjorde jobben minst like bra, ble det ikke til at vi gikk til innkjøp av en trådløs kortleser.

På Websiden har vi lagt alle studentene under hverandre ettersom hvilket emne de hører til. Studentene i de to emnene hører igjen innunder klasser. Det kunne vært en mulighet å sortere studentene på disse forskjellige klassene. Dette er mulig med informasjon fra LDAP, men vi plasserte de i en liste siden vi syntes det gjorde samme nytten, og vi fikk ikke noe krav om at de skulle være sortert på klasser. Det er dessuten studentregistreringen som er det viktigste.

Vi mener vi har utført oppgaven bra, men det finnes alltid rom for forbedringer uansett hva man jobber eller driver med.

### 7.3 Videre arbeid

Det er store muligheter for å videreutvikle dette programmet. Grensesnittet har store utviklingsmuligheter om man ønsker dette. Blant annet i forhold til bakgrunn, knapper og bilder på selve Websiden og Orego. Det er også mulighet for å legge inn nye funksjoner, som å sortere studentene inn under klasser på Websiden, eller å lage en ny knapp som linker til en side for dette. Der kan man for eksempel skrive inn navnet på klassen og man vil få frem studentene i denne klassen og hvilke øvinger de har tatt. Dette er selvfølgelig en del jobb, men bør være fullt mulig.

Det er også forbedringspotensial i Orego ved å legge inn flere undermenyer på menylinjen med forskjellige funksjoner. Samt muligheten for å gjøre kortleseren trådløs ved hjelp av Bluetooth, mobilnett eller å bruke HiGs trådløse internett.

### 7.4 Evaluering av eget arbeid

#### 7.4.1 Innledning

Gruppen har jobbet sammen ved flere andre prosjekter tidligere. Dette har fungert bra og vi har lært hverandre å kjenne, og vet hvem som egner seg best til de forskjellige oppgavene. Vi fungerer bra som en gruppe, og jobber bra sammen. Dette har gjort det lettere for oss å fordele oppgavene og ansvaret gjennom prosjektet. Det at vi har kjennskap til hverandre fra før har vist seg som en stor fordel gjennom prosjektet.

#### 7.4.2 Organisering av gruppen

Organiseringen av gruppen har fungert bra, med tanke på at vi kjenner hverandre godt. Vi har byttet på jobben å være gruppeleder, men i praksis har vi egentlig vært likestilte hele tiden. Når vi har støtt på utfordringer og problemer har vi diskutert oss i mellom, og kommet frem til en løsning vi begge har vært enige i. Siden det bare har vært to personer i gruppen har det vært ganske greit å komme til enighet, og få organisert jobbingen.

### 7.4.3 Arbeidsfordeling

Vi har fordelt arbeidsmengden relativt likt. En av oss har hatt hovedansvaret for kodingen, mens den andre har hatt ansvaret for GUI, rapportskriving og organisering av gruppes arbeid. Begge gruppemedlemmene har vært klar over sine arbeidsoppgaver i løpet av prosjektet, og hva den andre har drevet med parallelt. Vi er noenlunde fornøyd med fordelingen av arbeidsmengden i gruppa, og arbeidsmengden har vært ganske likt fordelt. Rapportskriving, møtereferater, loggføring og lignende ble utført når en eller begge i gruppen har hatt tid.

### 7.4.4 Prosjekt som arbeidsform

Med tanke på videre jobb har dette hovedprosjektet vært en god erfaring å ha med seg videre ut i arbeidslivet for medlemmene i gruppen. Vi har ikke vært den største gruppen med tanke på antall medlemmer, men dette prosjektet har i hvert fall gitt oss en liten smakebit av hva som venter oss når vi starter i jobb.

Når man jobber i et prosjekt med andre mennesker over lengre tid, så lærer man mye om sine medarbeidere både på godt og vondt. Da er det viktig at man kommer overens med sine medarbeidere, slik at fokuset alltid blir på det endelige målet, uten at det prosjektet blir avviklet eller utsatt.

I prosjekter har man alltid personer med forskjellig kunnskap, kompetanse og bakgrunn. Det er viktig å være oppmerksom på disse, og delegere de riktige personene til riktig arbeid, ettersom det vil høyst sannsynlig øke kvaliteten på prosjektet. Deltagerne i vårt prosjekt har bidratt med sitt pågangsmot, kompetanse og gode egenskaper, som vi mener har gitt et bra sluttresultat.

En av de store fordelene med prosjekt som arbeidsform er at prosjektdeltakerne selv kan bestemme når de skal jobbe, så lenge det ikke påvirker fremgangen i prosjektet på en negativ måte. Man trenger ikke være på jobb i en spesiell arbeidstid, slik som de har i vanlige jobber fra 9 til 16. Under prosjektet vårt har vi benyttet oss av dette, og det har fungert fint for oss. Det har blitt en del sene kvelder med kaffe til hjelp.

### 7.4.5 Subjektiv opplevelse av Bacheloroppgaven

Vi har opplevd dette hovedprosjektet som en stor utfordring, og en meget viktig og verdifull erfaring å ha med seg videre ut i arbeidslivet. Vi har måttet takle mange nye og uvante situasjoner, som har vært både utfordrende, krevende og spennende. Vi har blant annet måtte forholde oss til å legge opp vår egen agenda, ta selvstendige beslutninger, og forholde oss til absolutte tidsfrister. Vi tar med oss som alt dette som en positiv erfaring videre i arbeidslivet.

Til hovedprosjektet på HiG fikk vi låne et grupperom i kjelleren, som vi delte med en annen gruppe. Her fikk vi sitte i fred og ro, og jobbe for oss selv med oppgaven. Dette var en positiv forandring fra den vanlige skoledagen der man tilbrakte dagen i lesesalen. Vi fikk styre arbeidsdagen selv, som har fungert bra, og det har samtidig vært et sosialt samlingspunkt og trivselsfaktor. Det har blitt mange sene kvelder på det hyggelige grupperommet vårt med

kaffe og godteri.

Vårt samarbeid med oppdragsgiver og veileder har fungert meget bra i løpet av denne perioden. I startfasen tilbrakte vi mye tid med oppdragsgiver for å få på plass hva de ønsket seg av det ferdige resultatet. Under utviklingen har vi i tillegg holdt god kontakt via jevnlige møter for å diskutere fremgang og eventuelle endringer. Under utformingen av systemet har vi prøvd å sette oss inn i lærerne på sykepleieren sitt synspunkt og arbeidsdag, og utvikle systemet ut fra det. Vi har brukt veilederen vår flittig i løpet av prosjektet og han har vært til god hjelp. Vi har hatt nesten ukentlige møter med han hvor vi har diskutert fremgang, hvordan vi skulle løse problemer, og utforme systemet. Han har gitt oss mye gode råd og tips, som har hjulpet oss en hel del i prosjektet.

Gruppens samarbeid har fungert ganske greit. Vi har lært en god del om oss selv, og tilegnet oss masse ny kunnskap. Det har vært moro å gjøre noe annet enn å sitte på skolebenken dag ut og dag inn. Det har vært spennende å lage et ordentlig program fra bunnen av, der vi har kodet og laget så å si alt selv. Det har vært en utrolig spennende utfordring å utvikle noe selv for første gang. Det vi synes vi har lært mest om er serverfunksjonalitet, webdesign og integrering av ferdige systemer. Dette har vært interessant og utfordrende, men definitivt noe vi kommer til å få bruk for senere.

Noen ganger vi har vært litt usikre på om vi ville klare å fullføre prosjektet, men vi kom oss forbi de verste hindringene med litt ekstra jobbing. Vi har jobbet mye mer enn det vi trodde vi måtte for å komme i mål, og vi synes vi har oppnådd et bra resultat. Alt i alt er vi fornøyd med hovedprosjektet, og det har vært en stor opplevelse og veldig lærerikt for oss. Vi har opplevd alt fra glede, moro og interesse over innsikt i nye kunnskaper til frustrasjon og oppgitthet.

## 8. Konklusjon

Vi benyttet oss av fossefallsmodellen under utviklingen av dette prosjektet. I likhet med de aller fleste prosjekter av denne typen så har dette prosjektet gitt et kompatibelt og bra produkt. Vi har fulgt denne modellen etter beste evne, og vi har laget et produkt som tilfresstiller brukernes grafiske grensesnitt og krav til funksjonalitet etter beste evne. Vi har i tillegg gjort produktet meget robust, og det har endt opp i et resultat som oppfyller oppdragsgiveren og våre egne forventninger. Når vi startet å jobbe arbeidet vi etter fossefallsmetoden. Vi har forsøk å holde oss så mye som mulig til den i løpet av prosjektet, men det har ikke alltid vært like lett. Mot slutten har vi måttet arbeide med ting litt om hverandre, som å teste og utvikle samtidig. Dette førte til at vi ikke har fått tid til å gjennomføre så mye testing som vi ønsket. Det har igjen ført til at det har blitt noen endringer i prosjektplanen underveis. I forhold til den originale prosjektplanen så har vi brukt mye mer tid til utvikling enn vi forutså. Endringene i prosjektplanen kan man se i vedlegg B. I begynnelsen hadde vi planer om å integrere systemet i Fronter, men dette ble ikke noe av. Grunnen til dette var at det var vanskelig å få kontakt og svar fra Fronter, og i det hele tatt å få noe implementert der, med tanke på at det er et stort operativt system. Vi hadde en antagelse om at vi måtte lage en webside der lærerne og elevene kunne logge seg inn fra starten av for å se oppmøte, og det viste seg og stemme.

Oppdragsgiver ved avdelingen for sykepleie ville ha et system som tok seg av å registrere oppmøte ved de obligatoriske øvingene elektronisk. De ønsket at det skulle være mulig å vises frem oppmøtet via nettet. Lærerne skulle ha muligheten til å se oppmøtet for alle studentene, mens en student bare kunne se seg selv. Dette var et av de viktigste kravene. Underveis i prosjektet kom vi frem til mye ekstra funksjonalitet og brukervennlighet vi trengte i systemet, som vi ikke forutså på forhånd. Med tanke på den ekstra funksjonaliteten som for eksempel innlogging måtte vi jobbe en del ekstra, men det var utelukket å levere et produkt som ikke oppfylte kravet om personvern på en grundig måte.
Vi har hatt bra kontakt med arbeidsgiver, og de virker fornøyd. Det eneste vi savnet var at vi ikke fikk en utfyllende kravspesifikasjon, men at vi måtte lage og utvikle en selv på grunnlag av den informasjonen vi fikk ut fra samtaler. På en annen side var dette positivt med tanke på at vi fikk øvd oss på å egenutvikle et produkt.

I sluttfasen måtte vi forholde oss mye til IT-tjenesten, siden de skal drifte systemet. De hadde en del krav til hvordan de ville ha systemet, så vi fikk en del ekstra arbeid med å oppfylle disse kravene. Vi burde nok tatt kontakt med IT-tjenesten tidligere for å ha diskutert hvordan de ønsket å sette opp systemet. Men i løpet sluttperioden har vi hatt en bra dialog, og det har allikevel gått i orden. Vi håper at systemet blir godkjent og satt i drift.

Gruppen har fungert bra sammen gjennom hele prosjektet. Vi har vært flinke til å tilpasse oss eventuelle uventede problemer eller situasjoner. Dette har gått ut over vår egen fritid ved ekstra innsats for å få ting ferdig. Vi har unngått personlige konflikter og uenigheter, noe som ellers i arbeidslivet skjer fra tid til annen. Vi har lært hverandre å kjenne enda bedre i løpet av denne perioden, og det bare til det bedre for vår del. Alle de gode minnene og erfaringene fra HiG er ikke det eneste vi tar med oss videre i livet, men også uvurderlig kunnskap.