HOVEDPROSJEKT:

# MudoManager

FORFATTER(E):

SIMEN HIKEN GALTESTAD
PETTER MANFRED BJØRLIN

Dato:

19.05.2004

# Sammendrag av hovedprosjekt

| | | |
|---|---|---|
| Tittel: | MudoManager (no)<br>MudoManager (eng) | Nr. : 15<br>Dato : 19.05.2004 |
| Deltaker(e): | Simen Hiken Galtestad<br>Petter Manfred Bjørlin | |
| Veileder(e): | Tom Røise | |
| Oppdragsgiver: | Mudo Instituttet AS | |
| Kontaktperson: | Stig Lyshaug | |
| Stikkord (4 stk) | Medlemsadministrasjon, oppfølging, .net, databaser | |

| | | |
|---|---|---|
| Antall sider: 3 + 127 | Antall bilag: 10 | Tilgjengelighet (åpen/konfidensiell): Åpen |


Kort beskrivelse av hovedprosjektet:

Prosjektgruppen har utviklet en applikasjon som skal håndtere daglig drift ved instituttene i kjeden Mudo Instituttet AS, som er en kjede av taekwondo-institutter. Applikasjonen skal håndtere medlemsadministrasjon, oppmøte, arrangementer, graderinger, statistikker, timeplaner og oppfølging.

Bakgrunnen til denne oppgaven var at Mudo Instituttet AS ikke syntes det eksisterende systemet lengre var tilfredsstillende, og kunne ikke videreutvikles fordi firmaet bak dette har blitt lagt ned. De var ute etter et system som ikke bare skulle erstatte det eksisterende, men som også kunne forbedres på flere områder. To av de viktigste områdene for forbedringer var oppfølging og en sentral styring av alle medlemmer.

Applikasjonen har blitt utviklet i språket C# mot det nye rammeverket Microsoft® .net, noe som vil si at gruppemedlemmene har måttet sette seg inn i dette.

Resultatet av prosjektet skal være et ferdig produkt som skal settes inn i testkjøring etter at prosjektperioden er over. Eventuelle feil ved systemet skal så rettes opp i, og senere følger vedlikehold og videreutvikling av systemet.

# Forord

Hovedprosjektet MudoManager er et avsluttende prosjekt ved den treårige høgskoleutdanningen i data ved Høgskolen i Gjøvik våren 2004.

Prosjektets deltagere har vært Simen Hiken Galtestad og Petter Manfred Bjørlin

Oppdragsgiver for prosjektet er Mudo Instituttet As ved Fredrik Bjertnæs. Mudo Instituttet As er en kjede av taekwondoinstitutter, som består av 11 institutter, lokalisert fra Heimdal i nord til Lund i Sør.

Vi ønsker å takke følgende personer for hjelp og støtte under prosjektarbeidet:

- Mudo Instituttet for oppgaven og inspill underveis
- Marius Oppegård Nettum for design av webside
- Veileder Tom Røise for veiledning
- Stig Lyshaug (kontaktperson fra Mudo) for inspill og forslag
- Stian Skoglund for hjelp med oppsett av sentral database
- Jippii.no for avkobling i frustrerende perioder

# Innholdsfortegnelse

# 1 Innledning

Taekwondo er en koreansk selvforsvarsform med røtter som strekker seg flere tusen år bakover i tiden, og som ble innført i Norge i 1975 av Master Tien Ton-That. Mudo Instituttet ble startet våren 1998 med flere av Norges største taekwondo-institutter. Kjeden har nå flere tusen medlemmer, og er tilsluttet Norges Kampsportforbund og Norges Idrettsforbund.

## *1.1 Om oppgaven*

Videre følger noe generell oppgavedefinisjon slik den opprinnelig var gitt av oppdragsgiver.

### 1.1.1 Problemområde

Det finnes per i dag mange løsninger for forskjellige typer medlemsdatabaser, men ingen gode løsninger tilpasset spesifikt taekwondo-miljøet. Det er flere aspekter som gjør administrering av denne sportsgrenen spesiell. Blant annet har alle utøverne en beltegrad. Gjennom noe som kalles graderinger kan elevene avansere opp til høyere grader. Systemet som nå brukes i Mudo Instituttet As fungerer ikke tilfredsstillende. Det var derfor et ønske om å få utviklet en ny applikasjon som møtte deres ønsker og krav til funksjonalitet.

Da medlemmer registrert ved et institutt kan trene ved alle kjedens institutter var det et ønske fra oppdragsgivers side at registrering av treninger kunne gjøres ved alle institutter, uavhengig av den enkeltes elevs tilhørighet.

Oppdragsgiver hadde også ønske om utbedret funksjonalitet for oppfølging av eksisterende og nye elever. Dette innebar blant annet å kunne registrere inn i systemet personer som har meldt interesse for å bli kontaktet av instituttet ved et senere tidspunkt. De ville også ha mulighet til å gjøre oppslag på elever ved instituttet som har lav frekvens på treningsoppmøte, samt oversikt over hvilket nivå elevene er på når de avslutter sin treningskarriære, for å kunne følge opp den aktuelle gruppen bedre.

Mudo Instituttet As er på bakgrunn av det overnevnte interessert i å se på muligheten for å utvikle en applikasjon som baserer seg på en sentral database og som har noe oppdatert og utvidet funksjonalitet i forhold til det eksisterende systemet.

### 1.1.2 Avgrensninger

Den sentrale databasen vil ha et grensesnitt mot Mudo Instituttets regnskapssystem, slik at medlemsinformasjon legges til og oppdateres i regnskapssystemet, som igjen oppdaterer den sentrale databasen. Dette er en modul som utvikles av en ekstern utvikler paralelt med utviklingen av gruppens system.

### 1.1.3 Oppgavedefinisjon

Applikasjonen skal være et verktøy som skal ha i oppgave å administrere den praktiske driften ved de enkelte taekwondoinstituttene i Mudo-kjeden, med tanke på medlemsregister, treninger, graderinger og arrangementer. Dessuten vil den inneholde enklere og utvidede muligheter for oppfølging, i forhold til det eksisterende systemet, av elever og personer som melder sin interesse til instituttene.

Applikasjonen vil inneholde moduler for registrering av oppmøte, oppslag på elevdata, registrering av arrangementer, enkel mulighet for oppfølging av studenter, oppslag på statistikk ut ifra egendefinerte kriterier, registrering av graderinger og innlegging av timeplaner. Enkelte av modulene i applikasjonen vil være passordbeskyttet, slik at endringer i personalia og registrering av aktiviteter er forbeholdt instituttlederne.

Det vil være mulig å skrive ut lister over deltakere på arrangementer og graderinger, hvilke elever som er registrert inn på de forskjellige treningene og dagsrapport over antall elever på hver trening. Brukerne av systemet vil også ha muligheter til å skrive ut forhåndsdefinerte velkomst- og avskjedsbrev til valgte elever, samt mulighet til å skrive egendefinerte brev til grupper av elever ut ifra gitte kriterier. Det vil også være ønskelig å sende disse brevene på epost til de elevene som har en epostadresse registrert i systemet.

Etter endt prosjektperiode vil det foreligge en ferdig applikasjon som vil være klar for implementasjon hos Mudo Instituttet.

## *1.2 Målgrupper*

I dette kapittelet vil de ulike målgruppene for rapporten og applikasjonen drøftes.

### 1.2.1 Målgruppen for rapporten

Målgruppen for prosjektrapporten vil i tillegg til sensor og veileder, samt andre interesserte ved HiG være de ansatte i Mudo Instituttet As sentralt. Rapporten vil inneholde en del tekniske uttrykk samt en del ord og uttrykk relatert til taekwondo. Siden kunnskapen til leserne vil være varierende rundt begge disse feltene, er en del ord og uttrykk forklart i vedlegg A, side I

### 1.2.2 Målgruppen for applikasjonen

Målgruppen for applikasjonen er alle instruktører, assistenter, mastere og daglige ledere ved instituttene, samt administrasjonen i Mudo Instituttet sentralt. Siden datakunnskapene til brukerne av systemet vil være meget varierende, vil brukervennlighet og intuitivt design vektlegges.

## 1.3 Formål

Formålet med oppgaven vil være å utvikle en applikasjon som imøtekommer oppdragsgivers ønsker og krav til en registrerings- og administrasjons-applikasjon for en kjede av taekwondo-institutter. For oppdragsgiver vil en utvidet mulighet for oppfølging av personer som har meldt sin interesse til instituttene kunne føre til en økning i kundemassen, og derav en positiv økonomisk virkning.

Gjennom prosjektet ønsker prosjektgruppen å heve sin kompetanse innen prosjektarbeid, samt jobbe med programmering i større prosjekter. De vil også utvide vårt kunnskapsområde innen programmering, ved blant annet å lære oss et programmeringsspråk og rammeverk gruppemedlemenne ikke kunne før prosjektperioden.

## 1.4 Egen bakgrunn og kompetanse

Begge studentene i gruppen går tredje året på studiet Bachelor i ingeniørfag data med fordypning i programmering. Gruppemedlemmene har en generell kompetanse innen programmering, med hovedvekt på språkene C++ og Java. Studentene har også kunnskap i fagene systemutvikling og databaser, som vil komme til nytte under arbeidet med prosjektet.

Gruppen har ingen kompetanse innen C# eller .NET rammeverket. Da ingen foreleser noen av disse temaene ved skolen, så er det noe som måtte læres på egenhånd.

## 1.5 Rammer

I dette kapittelet vil valg rundt prosjektorganiseringen og –gjennomføringen beskrives.

### 1.5.1 Gjennomføring og arbeidsmetoder

I starten av prosjektperioden ble det bestemt å bruke inkrementell utvikling som utviklingsmodell. Inkrementell utvikling går ut på at systemet brytes ned i moduler, ut ifra en overordnet kravspesifikasjon, der en og en modul utvikles. Kravspesifikasjonen for den enkelte modul fastsettes ved starten av hvert inkrement. Dette åpner for store muligheter for endringer og nye idéer underveis i utviklingen. Grunnen til at nettopp denne modellen ble valgt for utviklingen var at kravspesifikasjonen var ikke helt fastsatt ved starten av prosjektet, samt at systemet enkelt kunne deles opp i moduler. Dette blir mer utfyllende forklart i senere kapitler.

Til å begynne med satt studentene sammen og satt opp en foreløpig kravspesifikasjon. Det ble også avsatt tid til opplæring i det nye språket og utviklingsverktøyet som skulle brukes.

Gruppen planla at kodingen stort sett skulle gjennomføres individuelt, men at gruppemedlemmene skulle jobbe sammen om de større og mer komplekse modulene i applikasjonen.

Kontakt med oppdragsgiver ble bestemt skulle foregå ved møter omtrent annenhver uke. Dette for at oppdragsgiver skal kunne komme med tilbakemeldinger på utført arbeid og forslag til ny funksjonalitet og utbedringer.

Gruppen skal også ha ukentlige møter med veileder, der fremdrift og arbeidsmetoder skal diskuteres. Her skal gruppen også få tilbakemeldinger på diverse skriftlig materiale.

For å samkjøre koden gruppemedlemmene har skrevet underveis har det blitt satt opp en CVS-server (Concurrent Versions System - versjonskontroll). Denne er beskrevet nærmere i senere kapitler.

### 1.5.2 Prosjektorganisering

Da gruppen kun består av to medlemmer, ble det bestemt at ingen av gruppemedlemmene skulle inneha rollen som gruppeleder, men at heller ansvaret skulle fordeles likt.

Ansvarsområdene under prosjekperioden har vært fordelt slik:
- Kontakt med oppdragsgiver og ekstern utvikler: Petter Manfred Bjørlin
- Kontakt med veileder: Simen Hiken Galtestad

### 1.5.3 Øvrige roller

I tillegg til de to personene i gruppen har oppdragsgiver leid inn en ekstern utvikler for hjelp til utvikling av den sentrale databaseløsningen. Denne utvikleren er, som nevnt i forordet av rapporten, Stian Skoglund.

## *1.6 Selve rapporten*

### 1.6.1 Organisering

I selve rapporten er det brukt standardformatene til Microsoft Word® når det gjelder overskrifter og normal skrift. Eksempler på kildekode er skrevet med Courier New, størrelse 10. Deloverskrifter i underkapitler som ikke er viktige for innholdsfortegnelsen er skrevet med fonten Times New Roman, 12 pt, fet. Bildeforklaringer er i Times New Roman, 12 pt, kursiv.

Rapporten er organisert i 9 kapitler:

**Kapittel 1 – Innledning:**
Gir en kort beskrivelse av prosjektet og planen for gjennomføringen.

**Kapittel 2 – Systembeskrivelse:**
Helheten i systemet slik den er planlagt

**Kapittel 3 – Kravspesifikasjon:**
Noe forenklet kravspesifikasjon, tilpasset rapporten.

**Kapittel 4 – Design:**
Viser valg og drøftinger gjort underveis med tanke på GUI, samt systemarkitekturen.

**Kapittel 5 – Implementering:**
Viser hvordan systemet er implementert. Det er her lagt stor vekt på selve kodingen og spesielle løsninger og funksjoner.

**Kapittel 6 – Beskrivelse av utviklingsprosessen:**
Beskrivelse av de forskjellige fasene i utviklingsprosessen ved valg av utviklingsmodell og gjennomføringen av dette.

**Kapittel 7 – Diskusjon av resultater:**
Diskusjoner og drøftinger rundt det endelige resultatet og mulige videreutviklinger.

**Kapittel 8 – Kilder:**
Kilder som er brukt av gruppemedlemmene under prosjektperioden

**Kapittel 9 – Vedlegg:**
Oversikt over rapportens vedlegg

### 1.6.2 Terminologi

Medlemmene av gruppen er i prosjektrapporten omtalt som gruppemedlemmene, gruppen og utviklerne.

Oppdragsgiver er også omtalt som Mudo Instituttet AS.

Enkelte tekniske ord og uttrykk er kommentert i vedlegg A, side I.

# 2 Systembeskrivelse

## *2.1 Kort systembeskrivelse*

Dette systemet vil være todelt. Det vil bestå av en sentral MS SQL server og brukerapplikasjoner på instituttenes datamaskiner. Serveren har, til en hver tid, en oppdatert database for alle instituttene. De lokale brukerapplikasjonene synkroniseres mot denne ved hver oppstart, og eventuelt ved behov, for å hente ut data lagret inn i databasen fra andre institutter. Tilgang for lagring av data i den sentrale databasen vil bestemmes ut ifra den innloggede brukerens tilgangsnivå i applikasjonen.

## *2.2 Visjonsdokument*

I de kommende underkapitlene vil det drøftes visjoner for den ferdige applikasjonen ved endt prosjektperiode.

### 2.2.1 Introduksjon

Mudo Instituttet AS ønsker en applikasjon som kan avløse det eksisterende systemet ”Mudo Data”. Dette er en applikasjon som ble utviklet for Mudo Instituttet tidligere, men oppfyller ikke lengre brukernes ønsker og krav til funksjonalitet. Det er ønskelig å forenkle oppgavene for instituttenes sentrale administrasjon, samt de enkeltes institutters ledere og instruktører. Videreutvikling av det eksisterende systemet er ikke mulig da firmaet som har utviklet denne har gått konkurs.

Den nye applikasjonen vil erstatte behovet for bruk av flettede dokumenter mot Microsoft Excel og Microsoft Word, samt begrense bruken av epost. Disse løsningene fyller instituttenes behov, men er noe kompliserende. De funksjonene som tidligere er utført av Word og Excel vil i den nye applikasjonen integreres, slik at brukeren kun har en applikasjon å forholde seg til.

Det er også et mål å eliminere begrensningene rundt treninger, graderinger og arrangementer som finnes i det eksiterende systemet. Dette vil i stor grad løses ved bruk av en sentral database. Dagens løsninger fyller ikke instituttenes ønsker og behov, og sees derfor på som en meget sentral del av utviklingen.

Innen prosjektperiodens slutt har gruppen som mål å utvikle en fullt fungerende applikasjon, som vil bli satt ut i drift ved instituttene.

### 2.2.2 Posisjonering

Systemet som skal utvikles skal hovedsakelig være tilpasset MudoInstituttets ønsker. Det har i lengre tid vært funksjoner både instruktører, instituttledere og den administrative avdelingen har savnet. Systemet vil enkelt kunne tilpasses andre institutter og kjeder innen kampsportmiljøet. Innlegging, endring og fjerning av programmoduler vil gjøres enkelt, slik at denne jobben ikke vil bli spesielt tidskrevende.

Det finnes andre systemer som har tatt sikte på samme målgruppe. Dette systemet vil derfor gå inn som en direkte konkurrent til enkelte systemer. Disse blir nærmere beskrevet i senere kapitler.

### 2.2.3 Problemer med dagens system

En av de viktigste årsakene til at det eksisterende systemet ikke fungerer tilfredsstillende er at hvert institutt i kjeden kun har informasjon om sine medlemmer. Dette fører til:

- Siden alle medlemmer i Mudo Instituttet kan trene ved alle kjedens institutter blir registrering av treninger upraktisk med denne løsningen. Antall treninger er blant annet viktig ved graderinger
- Påmelding til sentrale graderinger kan ikke gjøres i systemet
- Påmelding til arrangementer, både sentrale og lokale, kan ikke registreres i den eksisterende applikasjonen

Måtene problemene med påmeldinger løses i det eksisterende systemet er at instituttlederne ved de enkelte instituttene sender epost eller på annen måte gir beskjed til sentraladministrasjonen. Oppmøte av elever ved andre institutter enn det de er registrert ved må også manuelt meldes ifra til instituttet eleven tilhører.

Et kompliserende element ved det eksisterende systemet er også bruken av Microsoft Word og Microsoft Excel, som brukes til utskrift av brev, lister og timeplaner.

Mulighetene for oppfølging av elever er meget begrenset. Det er her et ønske om en stor utbedring av funksjonaliteten.

### 2.2.4 Produktfortrinn

Det nye systemet utvikles direkte mot Mudo Instituttets ønsker, slik at de får en skreddersydd applikasjon som tilfredsstiller deres behov. Det vil også være et ressursbesparende system med tanke på personalressurser, da mye informasjon sendes automastisk på epost eller oppdateres i den sentrale, felles databasen. Registreringer til sentrale arrangementer og graderinger legges direkte inn i det sentrale systemet, og behøver ikke sendes per epost eller lignende, slik oppgavene løses ved kjøring av det eksisterende systemet.

Integrering av moduler som teksteditor og utskrift av liste vil også virke ressursbesparende da brukerne av systemet slipper å bruke eksterne applikasjoner til dette.

### 2.2.5 Alternativer og konkurrenter

**Championsway Front-desk software**
Dette er et system utviklet generelt for alle institutter innen de ulike kampsportgrenene. Dette vil si at systemet blir ikke tilpasset de enkelte kampsportgrenene. Denne applikasjonen inneholder derfor funksjoner som ikke vil brukes av alle, samt mangler noen ønskede funksjoner.

Dette systemet inneholder en salgsmodul for instituttene, hvor de kan registrere salg og innkjøp av diverse artikler som gjøres gjennom instituttet, samt knytte dette opp mot den enkelte elev. En slik modul vil ikke komme til å brukes innen Mudo Instituttet. Det inneholder også en avtalekalender, som instituttene ikke har noe behov for i sitt system.

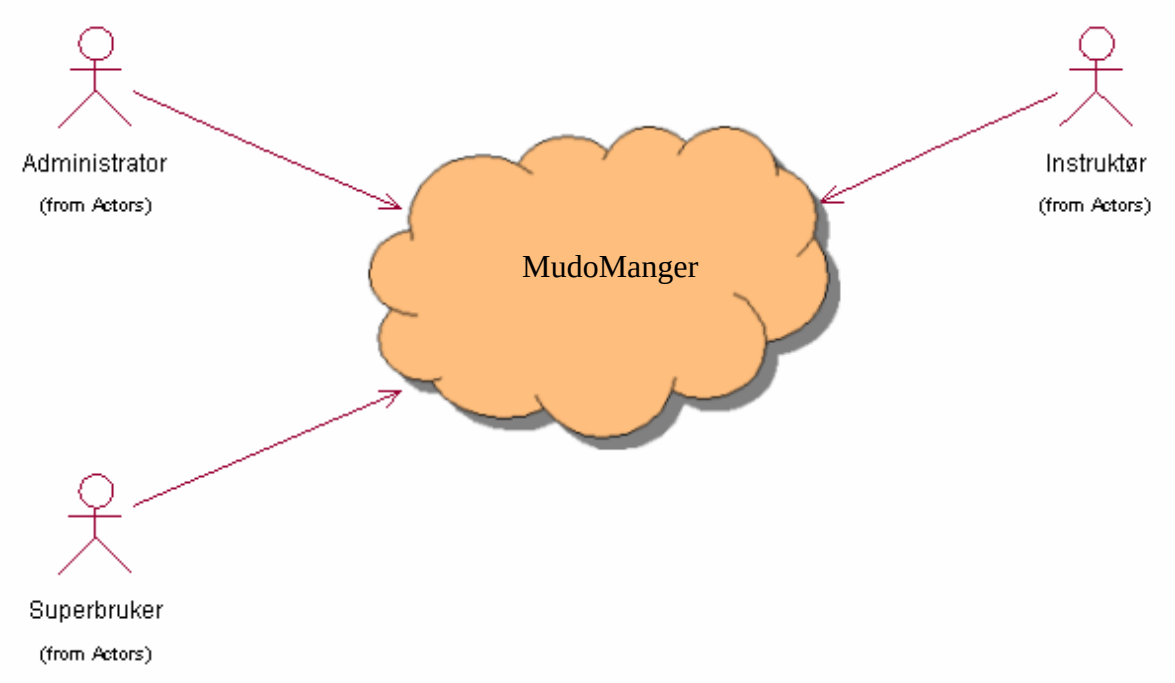
Mudo Instituttet så også nytten av å legge inn timeplaner i systemet, som ønskelig både skal brukes til utskrift og av oppmøtemodulen. Dette er en modul som ikke finnes i det omtalte systemet.

### 2.2.6 Brukermål

| | |
|---|---|
| Instruktører: | Registrere inn elever på trening, skrive ut liste over deltagere på treningen, gjøre oppslag på elevdata, melde elever på arrangementer, legge inn prospects, gjøre oppslag på registrerte propects og gjøre oppslag på elever med lite oppmøte på treninger. |
| Administratorer: | I tillegg til instruktørens mål vil en administrator kunne logge inn, sende melding om endring av elevdata til den sentrale administrasjonen, skrive ut velkomstbrev, ta bilder av elever, etterregistrere treninger for elever, skrive ut avskjedsbrev og gi melding til sentraladministrasjonen om at en gitt elev har sluttet, legge til lokale arrangementer, legge til lokale graderinger, registrere inn elever på graderinger, skrive ut graderingsliste, legge inn timeplan, endre passord, endre instituttdata, vise grafiske fremstillinger av statistikk, vise lister etter egendefinerte kriterier og skrive brev til elevene i listen, eller skrive ut listen. |
| Superbruker: | Denne brukeren vil i tillegg til administratorens mål kunne endre passord for administratorer uten å kunne gammelt passsord, legge inn sentrale arrangementer og graderinger, endre instituttdata for et valgt institutt, endre brukertilgang for brukere av systemet, opprette nye institutter i systemet, legge til nye programtyper, kontrakttyper og statustyper. |

### 2.2.7 Produktoversikt

Systemet vil utføre tjenester for følgende brukere:



# 3 Kravspesifikasjon

Videre følger en forenklet utgave av Use Case beskrivelsen, samt et par eksempler på detaljert beskrivelse. Noe av teksten er omformulert i forhold til den kravspesifikasjonen som er underskrevet av oppdragsgiver, men dette har ingen innvirkning på innholdet. Den fullstendige kravspesifikasjonen i sin helhet finnes i vedlegg I, side XVI.

## *3.1 Introduksjon*

Kravspesifikasjonen var delvis klar fra oppdragsgivers side allerede før prosjektet ble satt i gang, da det allerede finnes en eksisterende applikasjon. Gruppen har valgt å se fullstendig bort ifra oppsettet og organiseringen av den eksisterende applikasjonen, men kun sett på hva som skal lagres i systemet, og hvilke funksjoner som skal gjennomføres.

### 3.1.1 Kort om krav til systemet

Applikasjonen skal være et registrerings- og administrasjonsverktøy for en kjede av taekwondo-institutter. Det skal holdes orden på medlemsdatabasen og kontrakter, samt registrering av treninger på den enkelte elev, statistikkføring, administrasjon av graderinger og en avtalekalender til bruk for instituttets leder. Det skal være en enkel og selvforklarende GUI som skal kunne brukes av alle instituttets instruktører, med varierende datakunnskaper. Det er ønskelig å unngå kostnader for oppdragsgiver i størst mulig grad.

### 3.1.2 Kort om systemets omgivelser

Applikasjonen skal installeres på alle instituttenes datamaskiner. Applikasjonen skal kunne kjøres på alle Windows 98 maskiner, eller nyere OS. Det er varierende hvor nye maskiner de enkelte instituttene har, så det vil bli lagt vekt på en applikasjon som kan kjøre raskt og effektivt uansett plattform og hardware. Applikasjonen vil kreve en installering av Microsoft ® .NET framework 1.1 eller nyere, samt DirectX 8.1 eller nyere.

Systemet vil også kreve internett-tilgang for å få koblet til den sentrale databasen ved synkronisering av den lokale databasen mot denne. Mudo Instituttets regnskapssystem, Rubicon, vil i stor grad oppdatere den sentrale databasen med medlemsdata.

## *3.2 Systemets brukere*

Det vil være tre brukernivåer, der daglig leder vil ha tilgang til flere personopplysninger og brukervalg enn instruktørene, som vil være de daglige brukerne av systemet. De ansatte ved sentraladministrasjonen vil ha tilgang til noen tilleggsfunksjoner i forhold til de andre brukerne.

Brukeren ”administrator” vil være administrator for systemet ved det instituttet vedkommende er registrert på. Superbruker vil stort sett brukes av Mudo Instituttet AS sentralt, da denne brukeren vil fungere som en administrator på alle instituttenes systemer.

Applikasjonen må enkelt kunne installeres på alle instituttenes maskiner uten spesiell kunnskap. Installasjonen skal være selvforklarende og informerende, både med tanke på valg som brukeren må gjøre underveis og hva som skjer når maskinen jobber.

## 3.3 Livssyklus

Systemet skal enkelt kunne utvikles videre med nye moduler, eller forbedring av de eksisterende modulene.

Vedlikehold skal begrenses til et minimum. Det er ønskelig at applikasjonen skal kunne kjøres i lang tid uten noen behov for manuellt vedlikehold. Dette vil kun være ved endrede krav fra oppdragsgiver.

## 3.4 Ytelse

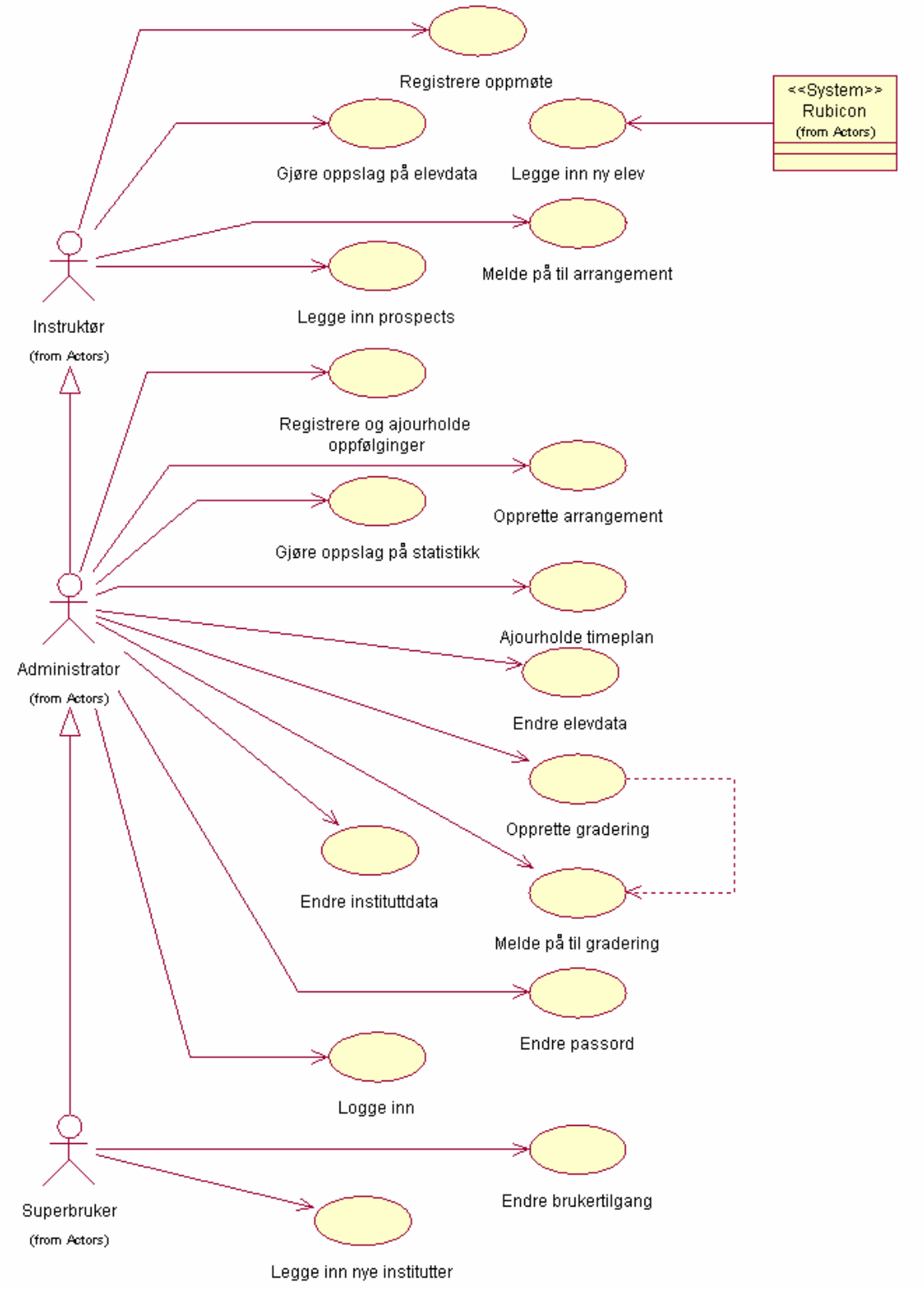
En bruker skal raskt, uten at dette er til hinder for de oppgaver som skal utføres, kunne få opp resultatet av sine valg og/eller operasjoner delvis uavhengig av hardware. Det er viktig å fokusere på et grensesnitt som krever et minimum av ressurser.

## 3.5 Sikkerhet

Det er ønskelig å passordbeskytte de delene av applikasjonen som ikke er nødvendig for alle brukerne av systemet, eller som kun instituttleder skal ha tilgang til.

## *3.6 Forenklet Use Case-beskrivelse*

### 3.6.1 Use Case diagram



### 3.6.2 Forenklede Use Case beskrivelser

**Use Case: Registrere oppmøte**
I oppmøtemodulen av applikasjonen vil instruktørene ved instituttet registrere hvilken trening som skal gjennomføres i en gitt sal, samt registrere inn de elevene som skal delta på denne treningen. Instruktøren og eventuelle assistenter registrerer også sitt oppmøte på treningene.

**Use Case: Gjøre oppslag på elevdata**
Instruktører kan i elevoppslagsmodulen hente ut informasjon om elevene ved instituttet. Instruktører vil kun gjøre oppslag på elever registrert på det aktuelle instituttet, administratorer kan i tillegg gjøre oppslag på elever registrert ved ”sitt” institutt og superbrukere vil kunne gjøre oppslag på alle elevene for alle instituttene.

Her kan det hentes ut personalia, samt opplysninger om treninger, graderinger og arrangementer en elev har deltatt på.

**Use Case: Legge inn ny elev**
Denne modulen omfattes ikke av systemet Mudora, men er tatt med i Use Case diagrammet for å vise helheten i systemet. Rubicon-systemet vil selv legge inn brukerdata om nye studenter når de har blitt registrert inn.

**Use Case: Legge inn prospects**
Alle brukere av systemet vil ha muligheten til å registrere personalia om personer som har meldt sin interesse for å bli kontaktet av instituttet ved et senere tidspunkt. Dette kalles et prospect.

**Use Case: Melde på til arrangement**
Etter at arrangementer er lagt inn i systemet vil det være mulig å melde på elever ved instituttet til de aktuelle arrangementene. Det skilles mellom sentrale og lokale arrangementer, samt de forskjellige arrangementstypene kurs, stevne, internt arrangement og eksternt arrangement.

**Use Case: Opprette arrangement**
Administratorer i systemet kan legge inn arrangementer ved sitt eget institutt. Superbrukere kan i tillegg legge til sentrale arrangementer.

**Use Case: Registrere og ajourholde oppfølging**
Det vil være mulig å gjøre oppslag på elever som har mindre enn et gitt antall treninger innen en gitt periode. Disse vil brukerne kunne skrive ut en liste over, eller eventuelt sende egendefinert brev til.

**Use Case: Gjøre oppslag på statistikk**
Administrator og superbruker vil ha tilgang til å vise statistikk over medlemmene av instituttet. Det vil være mulighet for å legge inn egendefinerte kriterier for søk.

**Use Case: Ajourholde timeplan**
Hvert institutt har en registrert timeplan. Denne vil noen ganger i året ha behov for å legges til en ny timeplan. Timeplanen brukes i oppmøtemodulen av applikasjonen, samt til utskrift.

**Use Case: Endre elevdata**
Superbruker vil ha tilgang til å endre den registrerte personalia om elevene ved instituttene. En administrator vil kun, ved å velge endring av elevdata, sende en epost til sentraladministrasjonen om hvilke personalia som skal endres.

**Use Case: Opprette Gradering**
Det vil også finnes to typer graderinger; lokale og sentrale. De lokale graderingene er de som holdes lokalt ved instituttene, mens en sentral gradering vil typisk være en gradering på en sommerleir eller lignende. Kun superbruker kan registrere sentrale graderinger, men både administrator og superbruker kan registrere lokale.

**Use Case: Melde på til gradering**
En administrator kan melde på elever til både sentrale og lokale graderinger. Brukeren vil her registrere inn deltagerens id samt eventuelt høyden til eleven. Dersom høyde registreres, vil eleven oppdateres i databasen med den nye høyden.

**Use Case: Endre instituttdata**
Enhver administrator vil kunne endre informasjon om sitt institutt. En superbruker kan endre denne informasjonen om alle institutter.

**Use Case: Logge inn**
For å få tilgang til de passordbeskyttede delene av applikasjonen vil alle brukerne bli tildelt et passord. Brukernavnet ved innlogging vil være det samme som medlemsnummeret i systemet.

**Use Case: Endre passord**
Administrator og superbruker vil ha tilgang til å endre passord. En bruker kan endre passord for seg selv, og brukere med lavere tilgangsnivå.

**Use Case: Endre brukertilgang**
Superbruker vil kunne endre tilgangsnivået for brukerne i systemet.

**Use Case: Legge inn nye institutter**
Dersom det opprettes nye institutter i kjeden, så vil superbruker ha tilgang til å legge inn nye institutter i systemet.

### 3.6.3 Eksempler på detaljerte Use Case beskrivelser

De Use Case bekskrivelsene som her er tatt med er de viktigste funksjonene i applikasjonen.

**Use Case: Registrere oppmøte**

**Aktører:** Instruktører, Administrator, Superbruker
**Beskrivelse:**
Etter at timeplan er lagt inn, så henter oppmøtemodulen inn hvilke klasser som går i det aktuelle rommet denne dagen. Det registreres så inn hvilke elever som skal trene på denne treningen, hvilken instruktør som har denne og eventuelle assistentinstruktører.

Det er også mulig å gjøre et søk etter navn i elevregisteret, for de som ikke husker sitt elevnummer. Det vil i denne modulen bli mulig å registrere inn på trening alle medlemmer i hvilket som helst av instituttene i kjeden.

Ved registrering av elev til trening vil det også komme opp, i tillegg til navn, antall treninger totalt, antall treninger siden siste gradering og antall treninger siden siste bestått gradering.

Det vil også være en oversikt over hvilke som har registrert seg inn på den aktuelle treningen på det aktuelle rommet.
**Hyppighet:**
Denne modulen vil brukes daglig
**Prebetingelse:**
At timeplanen er lagt inn korrekt for instituttet
**Hensikt:**
Holde ordent på antall treninger og hvilke treninger den enkelte har deltatt på, samt hvem som har vært instruktør og assistent på de enkelte treningene. Antall treninger brukes ved oppmelding til gradering, som krever et minimum antall treninger siden siste bestått gradering.
**Happy day scenario:**

| *Aktørhendelse* | *Systemhendelse* |
|---|---|
| Bruker velger trening ut ifra en liste | |
| Bruker taster inn elevnummer på de som skal registreres inn på denne treningen. | |
| | Databasen oppdateres med informasjon om trening og rolle (elev, instruktør eller assistent) |
| Bruker får opp på skjermen hvem som har registrert seg inn på denne treningen. | |

**Scenarier:**
Timeplanen er ikke lagt inn korrekt. Beskjed gis til brukeren om at timeplanen må registreres inn.

**Use Case: Gjøre oppslag på elevdata**

**Aktører:** Instruktører, Administrator, Superbruker
**Beskrivelse:**
Enhver bruker vil kunne gjøre oppslag på elevdata for elevene i det gjeldende instituttet. Det vil ikke være mulig å gjøre oppslag på elevdataene for en elev ved et annet institutt. Dette fordi instituttene drives på franchis-basis, og elevene ved ett institutt er da dette instituttets private kunder. En superbruker vil derimot ha tilgang til å se alle kjedens elever.

I denne modulen vil det også være mulig å gjøre oppslag på den gjeldende brukerens aktivitetslogg. I aktivitetsloggen vil alle aktiviteter vedkommende har deltatt på være registrert. Det være seg alt fra treninger og graderinger til eksterne arrangementer og kontrakstegning, osv.

Det vil fra denne modulen være mulig for administrator og superbruker å klikke seg videre til modul for endring av elevdata.

Personalia det kan gjøres oppslag på er:

- Medlemsnummer
- Grad
- Navn
- Fødselsdato
- Adresse
- Epost
- Tlf hjem, arbeid og mobil
- Tittel
- Tilgang (for Mudora-systemet)
- Kontrakstype
- Antall treninger (totalt, siden siste gradering og siden siste bestått gradering)
- Bilde av bruker

**Hyppighet:** Denne modulen vil brukes av og til; når en systembruker ønsker informasjon om en gitt elev
**Prebetingelse:** At brukeren vet medlemsnummer eller navn på vedkommende som det skal gjøres oppslag på
**Hensikt:** For å få informasjon om diverse personalia på de forskjellige elevene
**Happyday scenario:**

| *Aktørhendelse* | *Systemhendelse* |
|---|---|
| Bruker taster inn medlemsnummer | |
| | Oppslag gjøres i databasen og vises i skjermbildet |

**Scenarier:**

- Medlemsnummer finnes ikke i databasen.
- Bruker, ikke superbruker, forsøker å gjøre oppslag på student ikke registrert på gjeldende institutt.

Bruker vil få opp en enkel og forklarende meldingsboks om hva som er feil.

### 3.6.4 Risikovurdering av Use Cases

Ved planlegging av prosjektet ble det besluttet at de ulike inkrementene blir gjennomført slik at de modulene som er viktigst for at systemet skal kunne brukes blir gjennomført først, da disse er de mest kritiske funksjonene ved implementasjon av systemet.

Rekkefølgen på utvikling av de ulike UseCases vil derfor bli:

1. Registrere oppmøte
2. Gjøre oppslag på elevdata
3. Endre elevdata
4. Logge inn
5. Endre passord
6. Endre Instituttdata
7. Endre brukertilgang
8. Legge inn nye institutter
9. Opprette gradering
10. Melde på til gradering
11. Opprette arrangement
12. Melde på til arrangement
13. Legge inn prospects
14. Registrere og ajourholde oppfølginger
15. Ajourholde timeplan
16. Gjøre oppslag på statistikk

Noen av de Use Cases som ligger sist på listen burde egentlig ha blitt gjennomført tidligere, men de er totalt avhenige av at databasestrukturen er klar, og at studentene og den eksterne utvikleren har en felles forståelse av databasen.

### 3.6.5 Supplementær kravspesifikasjon

**Logging og feilhåndtering**
Ved feil i applikasjonen vil det komme opp en feilmelding om hva som er galt, dette vil bli logget til en loggfil. Det vil også være mulig at brukeren skriver inn en kort beskrivelse av hva som ble gjort før feilen oppstod. Dette er spesielt nyttig om det er en feil som forekommer ofte slik at feil-loggen kan oversendes utviklerne for utbedringer

**Implementasjonsbegrensninger**
Det er et ubestridelig krav at applikasjonen installeres og kjøres på Microsoft® Windows 98-plattform eller nyere. Installasjonen vil kreve installering av Microsoft® .NET framework og DirectX 8.1 eller nyere. Dette vil skje automatissk under installeringen, men krever at brukeren følger de retningslinjene som gis under installering.

**Grensesnitt**
Applikasjonen utvikles i programmeringsspråket C#. Rammeverket som brukes for GUI (Graphical User Interface, grafisk brukergrensesnitt) er Microsoft® .NET framework. Bak de lokale applikasjonene vil det ligge en Microsoft ® Access ™ database, og den sentrale databasen vil være en MsSQL-database, lokalisert på ActiveISP sine servere.

# 4 Design

## 4.1 Innledning

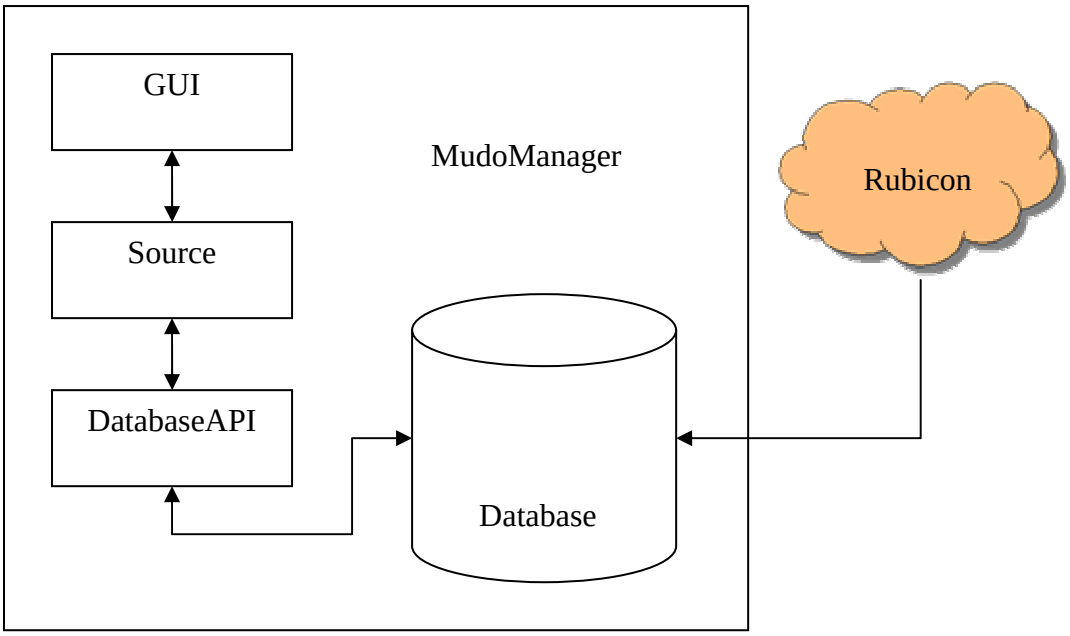
I denne delen av rapporten er det forsøkt å gi et bilde av oppbygningen og virkemåtene ved applikasjonen som er utviklet.

## 4.2 Systemarkitektur

Systemet er organisert i en 3-lags arkitektur som består av et GUI-lag, et databehandlingslag og et databaselag. Databaselaget er igjen skilt i to deler; en lokal og en sentral database. Av hensyn til senere utvikling så gruppemedlemmene fordelen av denne oppdelingen av systemet. Dette gjør det enklere å endre GUI-en i applikasjonen uten at selve databehandlingen blir berørt. Alle spørringen som kjøres mot databasen er skilt ut i en egen pragramtilleggsfil, slik at endringer i databasen vil i stor grad kun føre til endringer i denne filen. Dette vil gjøre oppdatering av systemene rundt hos instituttene enklere.

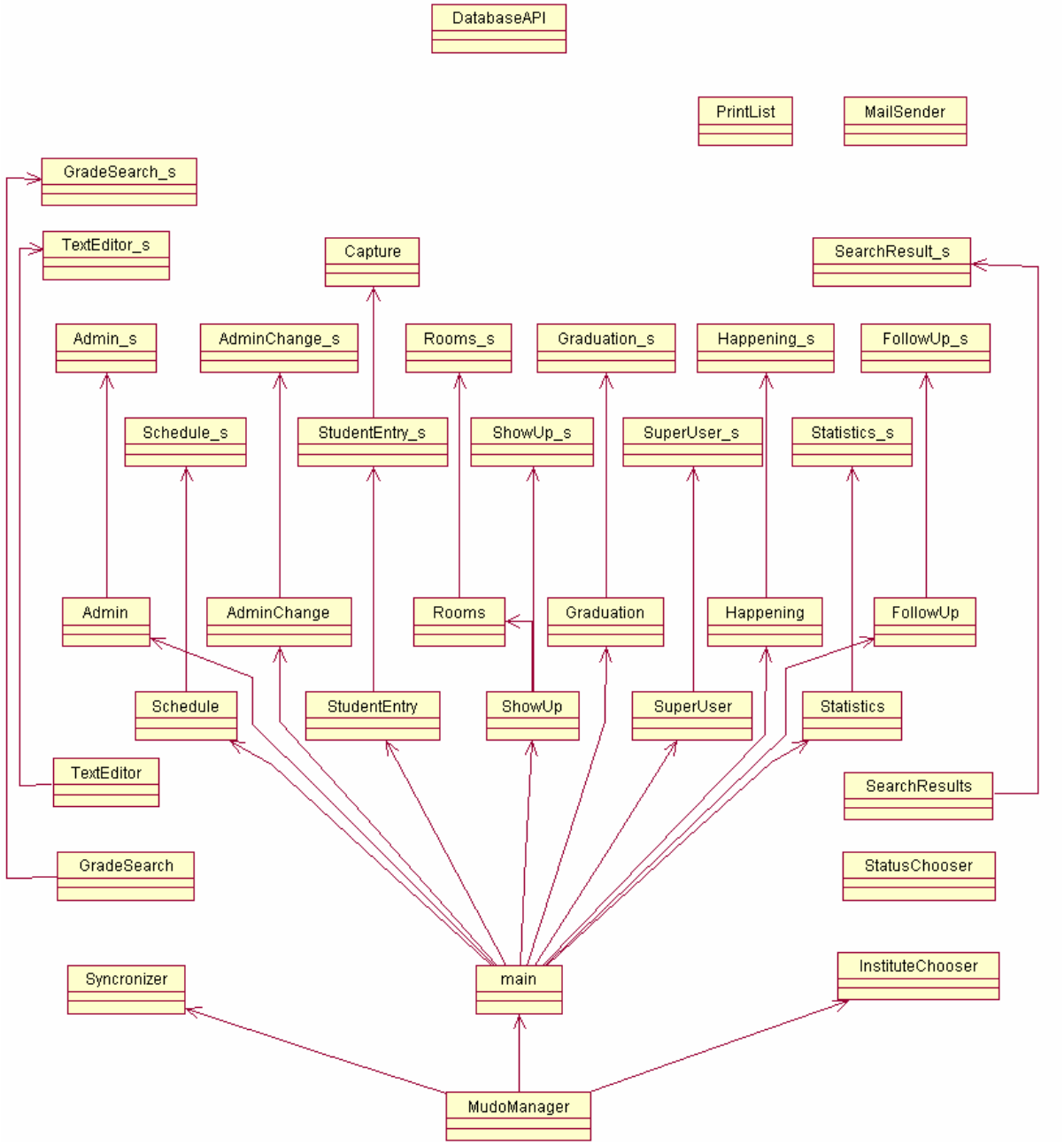
Grunnen til at databasen er skilt i to deler, en lokal og en sentral, er at instituttene ikke behøver å være avhengig av en konstant internettoppkobling. Den lokale databasen vil derfor synkroniseres med den sentrale ved oppstart av applikasjonen.

Den sentrale databasen vil ha et grensesnitt mot Mudo Instituttets regskapssystem, Rubicon. Dette systemet vil i stor grad oppdatere den sentrale databasen med elevdata. Endringer som må gjøres direkte i systemet er endringer av grad, tittel, tilgang, epost, enkelte telefonnummer og kjønn.



## 4.3 Filstruktur

### 4.3.1 Modell



### 4.3.2 Forklaring

Filene er organisert i 4 – fire – namespaces, som gir en klar struktur i filene. Et namespace er en gruppering av namespaces, klasser og andre datastrukturer. Namespacene i dette prosjektet er:

- Mudora – inneholder hoved-filene til systemet
- Mudora.GUI – inneholder GUI-filene for modulene i systemet
- Mudora.Source – inneholder source-filer som brukes av GUI-filene
- DatabaseAPI – Denne inneholder en fil, som tar seg av all kommunikasjon mellom databasen og applikasjonen.

Grunnen til at gruppen valgte denne inndelingen av filene var for å skille mellom systemfiler, filer som håndterer GUI, filer som utfører oppgavene og den delen av applikasjonen som tar seg av kommunikasjonen mot databasene.

**MudoManager.cs: (Mudora)**
Hoved- og oppstartsfil for hele applikasjonen. Starter InstituteChooser som applikasjon ved første gangs kjøring av systemet, starter så Syncronizer før main startes til slutt.

**InstituteChooser.cs: (Mudora.GUI)**
Bruker velger institutt og skriver inn rett informasjon.

**Syncronizer.cs: (Mudora.GUI)**
Synkroniserer den lokale databasen mot den sentrale. Synkroniserer hovedsakelig medlemsregister, aktivitetslogg og timeplan. Flere deler av databasen synkroniseres dersom endringer har forekommet i disse.

**main.cs: (Mudora)**
Oppretter hoved-GUI-en til systemet, samt oppretter instanser av alle filer i namespacet Mudora.GUI.

**Admin.cs, AdminChange.cs, Graduation.cs, Happening.cs, FollowUp.cs, Schedule.cs, StudentEntry.cs, ShowUp.cs, SuperUser.cs, Statistics.cs: (Mudora.GUI)**
GUI-filer for de ulike modulene I systemet. Har hver en referanse til en Source-fil i Mudora.Source namespacet. ShowUp har i tillegg en referanse til **Rooms.cs**, som er den delen som styrer med oppmøte i instituttets ulike saler.

**Admin_s.cs, AdminChange_s.cs, Graduation_s.cs, Happening_s.cs, FollowUp_s.cs, Schedule_s.cs, StudentEntry_s.cs, ShowUp_s.cs, SuperUser_s.cs, Statistics_s.cs, Rooms_s.cs: (Mudora.Source)**
Source-filer som utfører jobben for GUI-filene. Dette er organisert slik at GUI-filene *kun* tar seg av oppdatering av GUI, mens Source-filene behandler alle data, og håndterer funksjonskall videre til DatabaseAPI, for å hente ut nødvendig informasjon fra databasen.

**Capture.cs: (Mudora.Source)**
Brukes av StudentEntry for å ta bilde med webcamera av elevene.

**TextEditor.cs: (Mudora.GUI)**
Brukes av StudentEntry og Statistics. Dette er en tekst-editor for utskrift av egen- eller forhåndsdefinerte brev til et gitt utvalg av elever. Har referanse til en Source-fil, **TextEditor_s.cs (Mudora.Source)**, som tar seg av utskrift av brevene, samt sending av epost.

**GradeSearch.cs: (Mudora.GUI)**
Brukes av StudentEntry når bruker velger å legge til en elev til gradering. En liste over graderinger vil da vises. Bruker en Source-fil, **GradeSearch_s.cs (Mudora.Source)**, for å hente data ut fra databasen og melde studenten på den rette graderingen.

**SearchResults.cs: (Mudora.GUI)**
Ved søking på navn vil denne vise treff i databasen. Bruker Source-filen **SearchResult_s.cs (Mudora.Source)** for å hente ut data fra databasen.

**StatusChooser.cs: (Mudora.GUI)**
Brukes for å endre status for elever på graderinger og andre arrangementer, samt vise en fremdriftsindikator når det søkes etter elever med gitte kriterier i FollowUp.

**PrintList.cs: (Mudora.Source)**
Dette er en statisk klasse som tar seg av alt som har med utskrift av diverse lister å gjøre. Dette være seg graderingslister, timeplan, deltakerlister, samt lister skrevet ut fra statistikk-modulen.

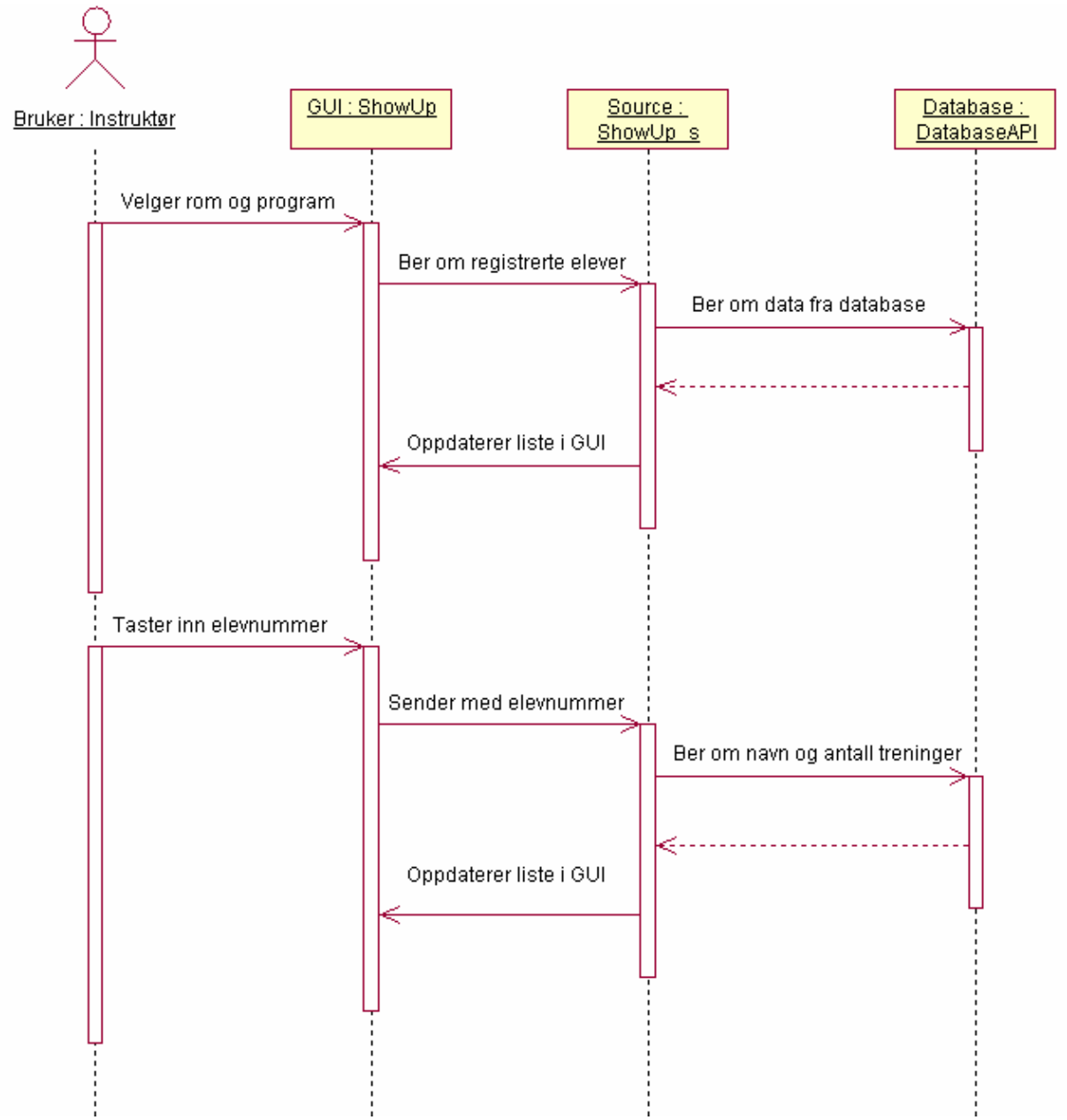
**MailSender.cs: (Mudora.Source)**
Dette er en fil som brukes til feil-rapportering i applikasjonen. Om en feil i applikasjonen oppstår vil det vises en dialog hvor brukeren kan taste inn hva som ble gjort før feilen oppstod, og sender så en epost til utviklerne med informasjon om hva som gikk galt.

**DatabaseAPI.cs: (DatabaseAPI)**
Denne er skilt ut i et eget prosjekt, for å kreere en programutvidelses-fil, samt skilt ut i en egen namespace. Denne filen tar seg av all kommunikasjon mot databasen, både den lokale og den sentrale.

## 4.4 Designmodeller

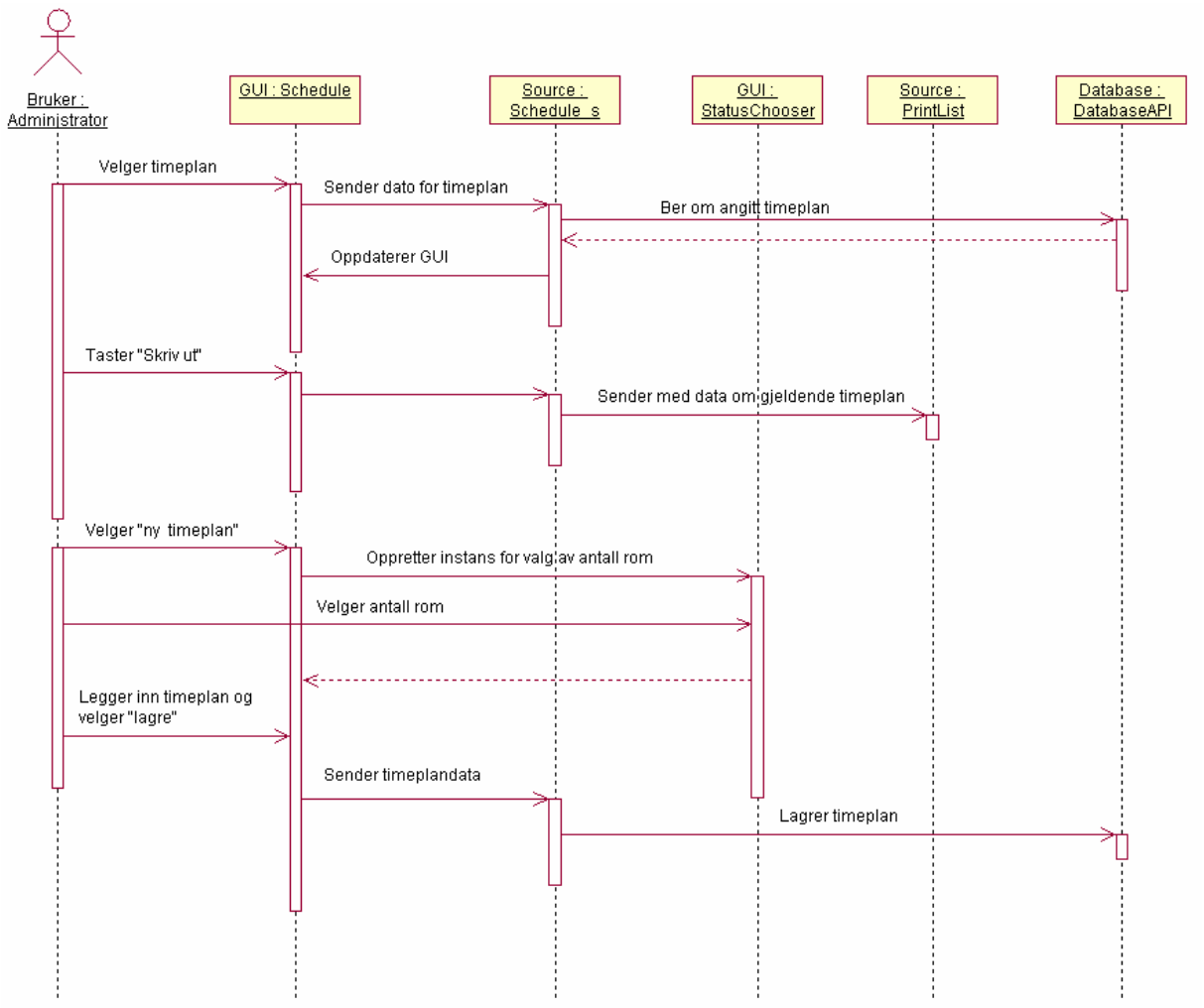
Under følger et utvalg av sekvensdiagrammer. De resterende diagrammene kan finnes i vedleggene. Disse diagrammene ble valgt fordi de viser noen av de viktigste funksjonene i systemet, og viser godt hvordan systemet er lagt opp. Disse diagrammene følger ikke fullstendig UML-standarden, men gir en indikasjon på hvordan utviklerne har tenkt under planleggingen og oppleggingen av filstrukturen.

### 4.4.1 Sekvensdiagram: Registrere oppmøte



**Forklaring:**
Instruktøren velger hvilket rom og hvilket program elever skal registreres inn på. Det hentes da inn fra databasen hvilke elever som er registrert inn på denne treningen. Dette vises i en liste. Instruktøren taster så inn elevnummer på elevene som skal trene. Treningen registreres i databasen, den påmeldte eleven vises i listen over påmeldte elever og antall treninger for den aktuelle eleven vises på skjermen.

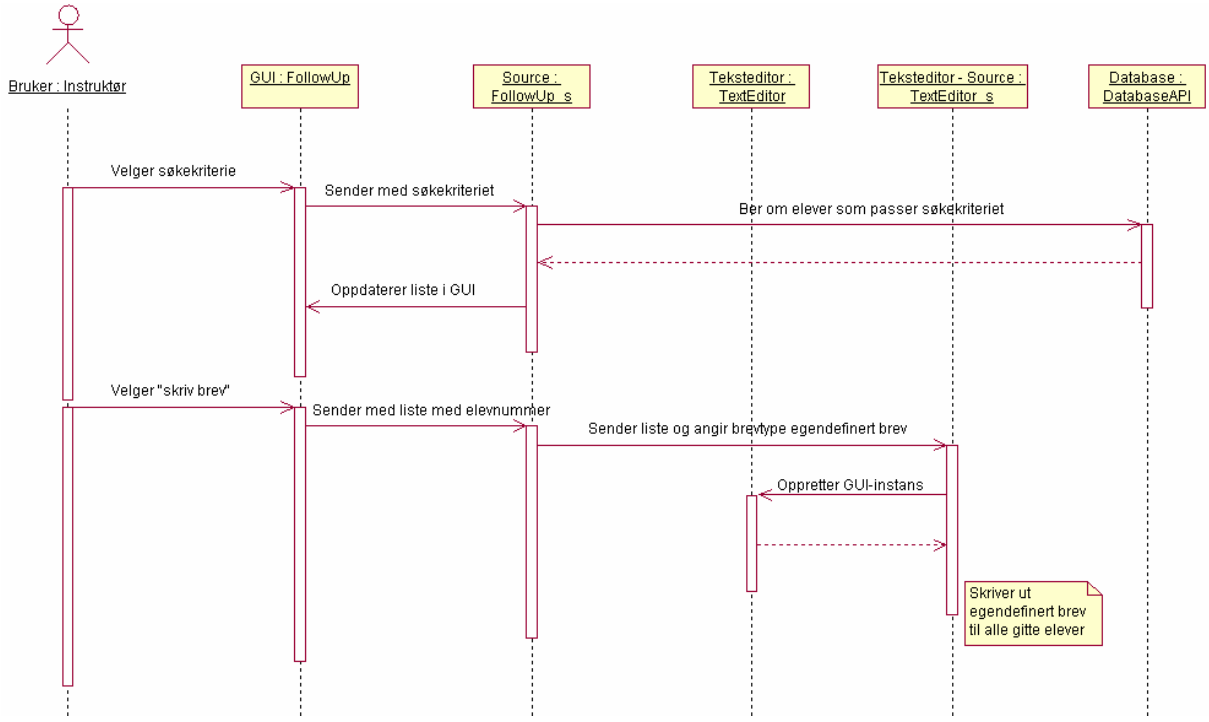
### 4.4.2 Sekvensdiagram: Skrive ut og legge til timeplan




**Forklaring:**
Bruker velger hvilken timeplan som skal vises. Den aktuelle timeplanen hentes så ut fra databasen og vises. Bruker vil så skrive ut den valgte timeplanen, og klikker så ”skriv ut”. Data om gjeldende timeplan sendes til PrinList, som tar seg av utskriften av timeplanen.

Bruker vil legge inn ny timeplan; velger da ”Ny timeplan. Det vises da en dialog hvor bruker taster inn antall saler. Bruker legger så inn programmene på rett dag og tid, og velger ”lagre timeplan”. Timeplanen lagres så i databasen.

### 4.4.3 Sekvensdiagram: Oppfølging – gjøre oppslag og skrive brev

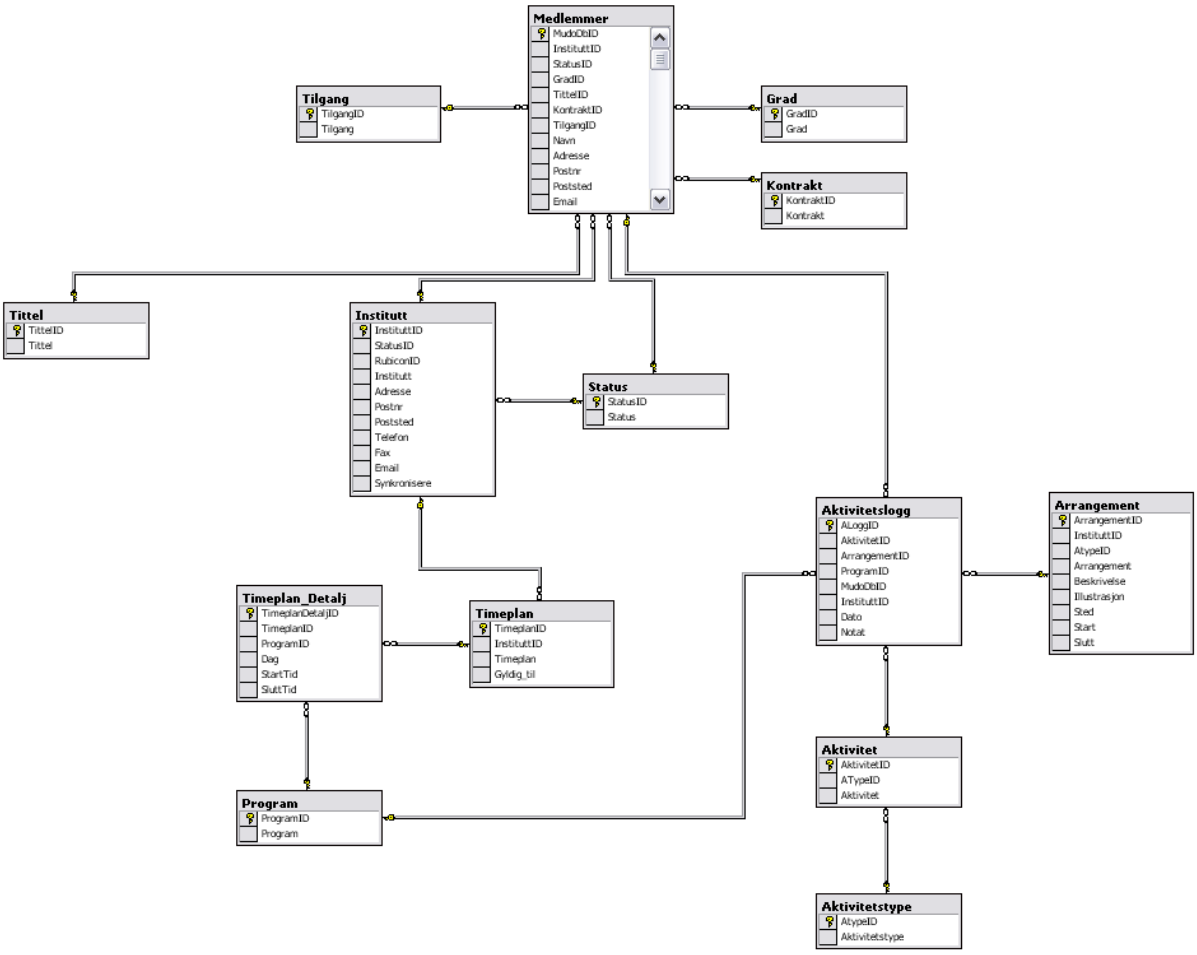



**Forklaring:**
Bruker velger kriteriet som skal søkes opp i databasen (Antall treninger siden en gitt dato). Elevene som passer kriteriet hentes fra databasen og vises i en liste.

Bruker ønsker så å skrive egendefinert brev til elevene i listen; og velger ”skriv brev”. Det opprettes da en ny instans av teksteditoren hvor det sendes med elevnummer til mottakerne og angir at det er egendefinert brev som skal skrives. Bruker skriver brevet, skriver inn avsender og trykker ”skriv ut”. Bruker får spørsmål om det skal sendes epost til de elevene med registrert epostadresse. Det skrevne brevet skrives da ut til de gitte studentene, og epost sendes om dette er valgt.

## *4.5 Databasestruktur*

### 4.5.1 Modell av databasen




### 4.5.2 Generell forklaring til databasen

Medlemmer er den sentrale delen av denne databasen. Her ligger informasjon om alle elevene med personopplysninger. Hvert medlem har en grad, en bestemt kontraktstype, et tilgangsnivå i applikasjonen, en tittel og en status.

Alle aktiviteter et hvert medlem er med på lagres i Aktiviteteslogg. I denne ligger informasjon om alle treninger (da også angitt hvilket treningsprogram som er gjennomført), graderinger og andre arrangementer, samt en aktivitetsstatus på denne (fra tabellen Aktivitet).

Arrangementer og graderinger som arrangeres sentralt eller lokalt ligger i Arrangement, hvor det er beskrivelse av arrangementet med start- og sluttdato og informasjon om sted og eventuelt hvilket institutt som arrangerer denne.

Alle elever er også tilknyttet et gitt institutt. For hvert institutt ligger det lagret adresse og kontaktinformasjon, samt en status for instituttet.

Timeplan_Detalj inneholder hvilket program som gjennomføres på et gitt tidspunkt på en gitt dag. Hver post i denne tabellen er knyttet opp mot et institutt, en sal og en gyldig til-dato, som ligger i tabellen Timeplan.

Tabellene Grad, Tittel, Kontrakt, Tilgang, Program og Aktivitetstype inneholder det tabellnavnet angir.

### 4.5.3 Forklaring av tabeller

**Medlemmer:**
Det har hele tiden vært klar at dette vil være hoved-tabellen i databasen. I første utkastet til databasen var også ”Betaler”, ”Betalers adresse”, ”Betalers telefonnr” og ”Kontonummer” i tabellen. Ettersom denne databasen skal oppdateres via regnskapssystemet til Mudo Instituttet, ble beslutningen tatt om å fjerne disse, da dette er et medlemsregistrerings og –administreringssystem og ikke et regnskapssystem. Det ble også lagt inn et ekstra felt kalt ”Notat”, som senere kan brukes til oppfølging og andre opplysninger knyttet til et spesielt medlem. Et felt ”Fingeravtrykk”, som kommer til å brukes senere, dersom oppdragsgiver går inn for implementering av dette senere, ble også lagt til.

Denne tabellen inneholder også både et MudoDbID og et MedlemsNR, som begge er unike identifikatorer. Grunnen til at dette er slik er at prospekter som legges inn ved det enkelte institutt, altså personer som melder sin interesse for å bli kontaktet av instituttet må ha en unik identifikator i databasen selv om de ikke har noe medlemsnummer.

**Tilgang:**
Inneholder de ulike tilgangsnivåene for en bruker av systemet.

**Grad:**
Inneholder alle beltegradene en elev kan ha.

**Kontrakt:**
Inneholder kontrakttypene ved instituttene.

**Tittel:**
Inneholder de ulike titlene et medlem kan ha. Dette kan være elev, instruktør, assistent eller master.

**Aktivitetslogg:**
Poster i denne tabellen er knyttet til en bestemt elev ut ifra MudoDbID og et bestemt institutt ut ifra InstituttID. Her vil det ligge en ArrangementID dersom dette er et arrangement, en ProgramID dersom dette er en trening, samt en AktivitetID, som angir elevens status ved det aktuelle arrangementet eller elevens rolle på den aktuelle treningen.

Feltet ”Notat” ligger også i denne tabellen som et element det kan videreutvikles på senere. Her kan det for eksempel legges inn informasjon om medisiner en elev trenger ved påmelding til arrangementer eller lignende.

**Arrangement:**
Alle arrangementer og graderinger registreres inn i denne tabellen, med korrekt AtypeID fra tabellen Aktivitetstype. Denne angir om dette er gradering, kurs eller andre arrangementer. ”Illustrasjon” vil inneholde filnavnet til en illustrasjon knyttet til dette arrangementet. Dette feltet er for tiden ikke i bruk.

**Aktivitet:**
Denne tabellen inneholder de mulige aktivitetsstatusene en elev kan ha ved de ulike aktivitetstypene.

**Aktivitetstype:**
Inneholder de ulike aktivitetstypene et arrangement kan være.

**Institutt:**
Denne tabellen inneholder informasjon om de enkelte instituttene i kjeden. Også denne tabellen har to unike identifikatorer, InstituttID og RubiconID. Grunnen til at dette er slik i denne tabellen er at RubiconID-en kan endres for instituttene.

Feltet ”Synkronisere” er kun i bruk i den sentrale databasen. Dette er et true/false felt som settes til ”true” om det er oppdatert i noen av de andre tabellene enn de som normalt synkroniseres ved oppstart. Dette er gjort slik for at instituttene skal bruke kortest mulig tid på synkroniseringen, da det stort sett ikke er endringer i alle tabellene.

**Status:**
Denne tabellen inneholder de mulige statusene et institutt eller et medlem kan ha. Dette være seg aktiv, inaktiv, frys / sykdom, ikke betalt eller ny. Kun de to første brukes av institutt.

**Timeplan_Detalj og Timeplan:**
Timeplan_Detalj inneholder informasjon om et program som gjennomføres på en gitt dag med et gitt starttidspunkt og et gitt sluttidspunkt, samt hvilken timeplan denne tilhører. Timeplan inneholder informasjon om hvilket institutt den tilhører, hvilket rom timeplanen er for og gyldighetsdatoen for denne.

**Program:**
Inneholder de ulike treningsprogrammene som gjennomføres ved instituttene.

### 4.5.4 Om utarbeidelsen av databasen

Da det ble klart at applikasjonen skulle utvikles for alle instituttene ved Mudo Instituttet AS, og samkjøres ved en sentral database ble det, fra Mudo Instituttets side leid inn en ekstern utvikler som skulle assistere med opprettelsen av den sentrale databasen. Den gjeldende databasen ble utviklet som et resultat av dialog mellom gruppens medlemmer og den eksterne utvikleren. Vedkommende fikk da i hovedoppgave å opprette den sentrale databasen.

## *4.6 Valg av databasetyper*

Før utviklingen ble startet var det flere alternativer for databasetype for den lokale databasen. Det ble først valgt å bruke mySQL databaseserver lokalt på alle klientene. Dette ville krevd at brukeren måtte installert og satt opp denne serveren korrekt før systemet kunne brukes. Dessuten måtte oppdragsgiver kjøpt lisens for disse serverne, da mySQL kun er gratis for ikke-kommersiell virksomhet.

Valget falt derfor på å legge en Microsoft Access database som grunnlaget lokalt. Dette krever ingen konfigurering for bruker og er klar til bruk etter installering av applikasjonen. Da alle instituttene har lisens for Microsoft Office ble det heller ikke nødvendig med innkjøp av nye lisenser.

Sentralt står det en MS SQL database. Bakgrunnen for dette valget var at Mudo Instituttet allerede har internettsidene sine hos ActiveISP, som da kunne tilby plass på en MS SQL server til en rimelig pris.

## *4.7 Beskrivelse av brukermiljø*

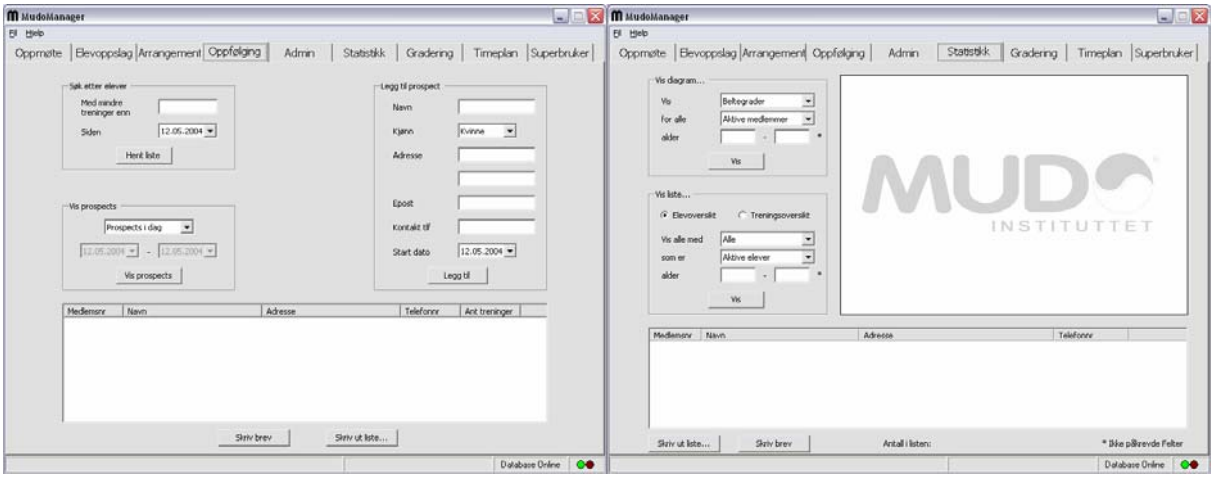
Videre følger en beskrivelse av oppbygningen av brukergrensesnittet, samt en beskrivelse av brukermiljøet for de ulike brukerne av systemet.

### 4.7.1 Kort om brukergrensesnitt

Første prioritet for brukergrensesnittet i applikasjonen var å utarbeide et intuitivt og enkelt brukergrensesnitt for alle brukerne av systemet, da datakunnskapene er meget varierende. Det ble lagt liten eller ingen vekt på brukergrensesnittet for det eksisterende systemet. Utviklerne mente at dette var lite intuitivt, til dels rotete og krevde en viss grad av opplæring før bruk.

Utviklerne så også fordelen med å begrense mengden informasjon på skjermen for enkelte brukernivåer. Da brukerne ved daglig kjøring av applikasjonen ikke behøver alle modulene, samt at en del av informasjonen ikke bør være offentlig, ble noen av modulene skilt ut i en passordbeskyttet del.

Brukerne vil navigere i applikasjonen ved hjelp av ”arkfaner” i toppen av skjermbildet. Dette er et grensesnitt som er godt kjent for de fleste brukerne, og skiller modulene på en god måte.




Det er i alle modulene lagt vekt på et uniformt utseende som vil gjøre at brukeren lett kan bruke de ulike delene av applikasjonen. Opplegget av GUI har vært en stor utfordring for gruppen, da ingen av medlemmene har noen erfaring i dette og at de ikke har vært gjennom noen slags opplæring innen dette. Det har derfor blitt fokusert på en ”generell oppfatning” fra utviklernes side. Med dette menes det umiddelbare inntrykket utviklerne har hatt da de har sett disse for første gang, sett ut ifra et kritisk synspunkt.

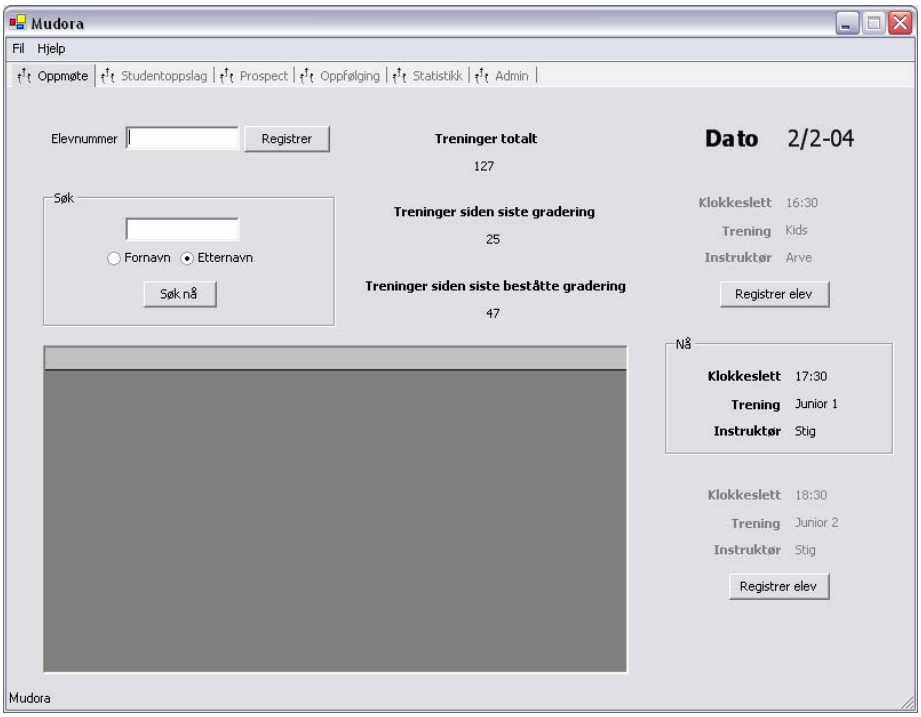
Det ble tidlig bestemt at størrelsen på GUI-en skulle bindes til en fast størrelse. Grunnen til at dette ble gjort var at det ville krevd store omlegginger i GUI-koden for å gjøre denne dynamisk. Siden da alle GUI-elementer er lagt opp i forhold til hverandre, ville dynamisk plassering kunne gjort brukergrensesnittet mindre oversiktlig.

Detaljert beskrivelse av de ulike skjermbildene vil finnes i vedlegg H, side XXXIX, som brukerveiledning.

### 4.7.2 Vurderinger og endringer av brukergrensesnitt underveis

I dette kapittelet blir det vist vurderinger og endringer i to av applikasjonens moduler underveis i utviklingen av systemet. Grunnen til at disse to modulene er valgt er at dette er de to viktigste modulene, samt at disse ble utviklet først og har hatt flest endringer. Disse to modulene har satt standarden for utviklingen av de andre delene av applikasjonen.

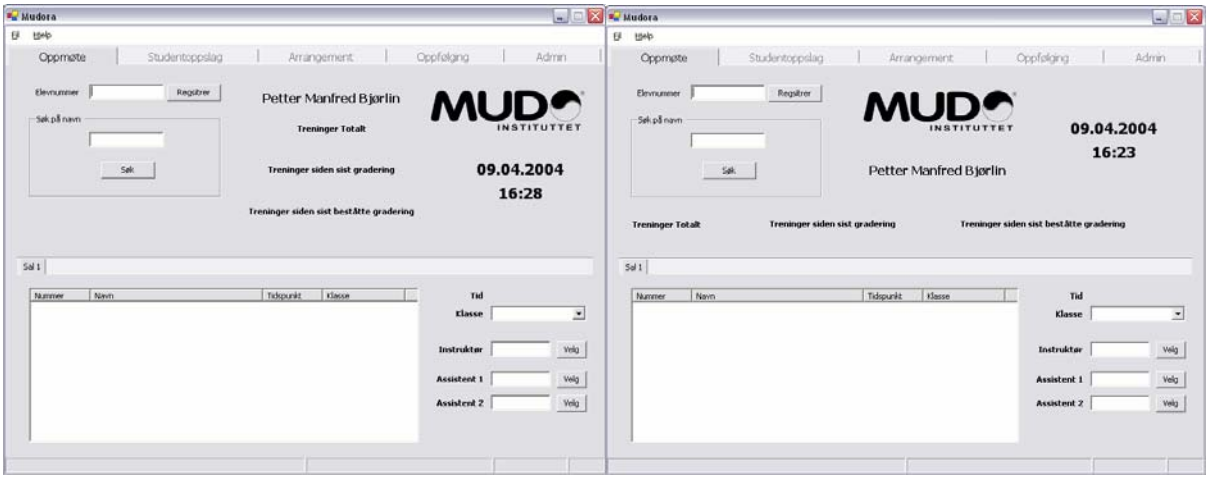
## Oppmøte:




Dette er et bilde av brukergrensesnittet slik det var i en tidlig utgave. Det var her lagt opp til at timene, på høyre siden, skulle oppdatere seg automatisk ut ifra timeplanen. Forrige og neste time skulle vises på listen, slik at brukerne kunne registrere inn elever på disse treningene (for eksempel ved for sent oppmøte). Listen over de påmeldte til den aktuelle treningen skulle vises nede til venstre. I denne utgaven var det kun lagt opp til at instituttene hadde en sal for treninger.

Grunnen til at det her var valgmuligheter på å søke etter fornavn eller etternavn var at det i starten var lagt opp til at disse skulle lagres i hvert sitt felt i databasen. Oversikten over antall treninger er knyttet til den siste eleven registrert inn på trening. Det var et ønske fra oppdragsgiver at dette skulle vises ved hver registrering av trening.

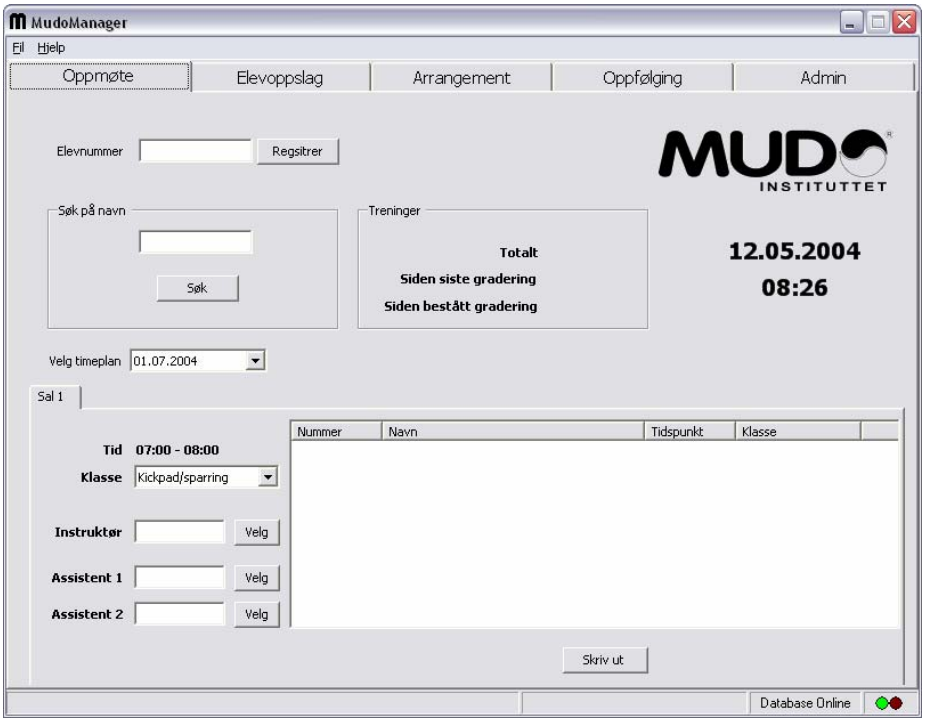
Feltet for inntasting av brukernummer er plassert øverst til venstre fordi dette er det feltet som kommer til å brukes mest. Grunnen til at denne plasseringen da er fornuftig er bestemt ut ifra at dette er den naturlige plassen å starte for en bruker.




Etter ønske fra oppdragsgiver ble det lagt inn mulighet for å skille mellom oppmøte i flere forskjellige saler, slik at elevene ble registrert inn på den rette treningen. Det ble også klart at de ville ha med logoen på denne modulen, da denne vil være denne modulen som blir sett av flest og brukt oftest. Det ble her, som bildene viser, gjort flere vurderinger på hvordan informasjonen skulle presenteres bruker. En klokke er også lagt til i dette skjermbildet.

Som disse skjermbildene også viser ble valget for fornavn eller etternavn fjernet, da det ble klart at databasen kun vil inneholde et felt ”navn”. Det vil være muligheter her for å søke på kun deler av navn.

Det ble også lagt til muligheter for at instruktører og assistenter melder seg på en trening, for å kunne ha en oversikt i ettertid over hvem som har instruert og hatt ansvar for de enkelte treningene.

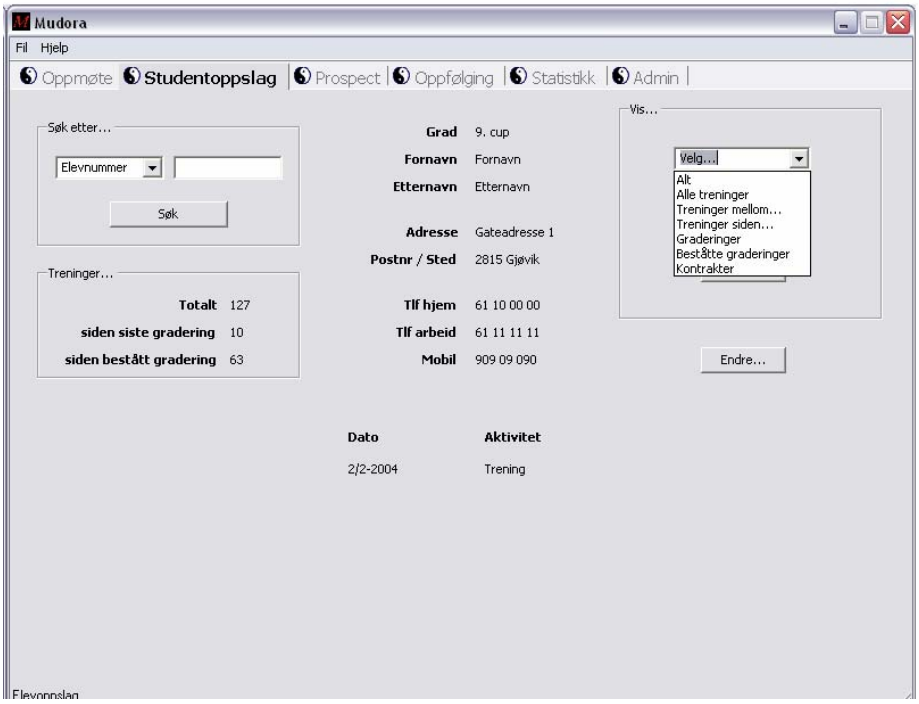


Her er det vist hvordan det endelige resultatet av oppmøte-modulen. Det er her lagt til et valg for å velge hvilken timeplan det skal gjøres oppslag ut ifra. Fra denne timeplanen hentes antall rom, samt hvilke timer som skal gjennomføres denne dagen.

Valg av klasse og innregistrering av instruktører og assistenter er flyttet over til venstresiden. Årsaken til dette er at det blir en naturlig inndeling da alle ”input-felter” finnes i en kolonne. Presentasjonen av antall treninger er forandret så den skulle bli lik den samme rammen i modulen for elevoppslag.

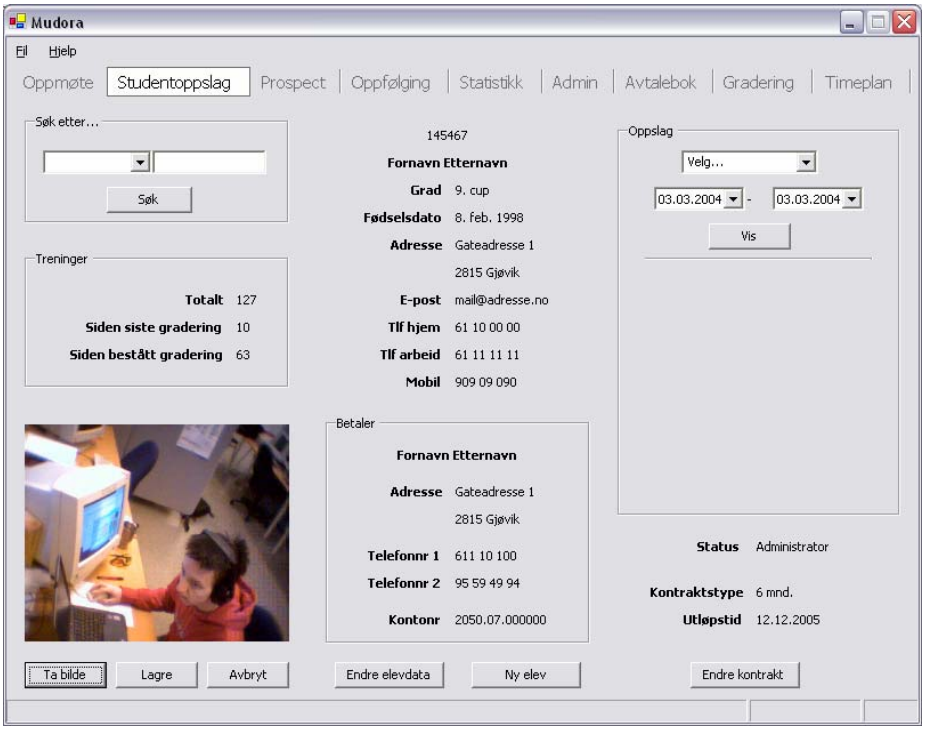
Grunnen til at logoen ble bestemt skulle være på høyresiden, var at denne skulle være minimalt blikkfang, samt at dette gir en klar kolonnedeling mellom input, informasjon om den sist innregistrerte elev og annen statisk informasjon.

**Elevoppslag:**




Dette er en tidlig utgave av brukergrensesnittet for elevoppslag. Området i bunnen av skjermbildet skulle innholde resultatet av søkningen fra feltet ”Vis…”.

Denne viser også en av vurderingene vi hadde angående arkfanene i toppen av applikasjonen ved å legge til bilder. ”Søk etter…” er her plassert på samme sted som på modulen for oppmøte, av samme grunn som forklart i beskrivelsen av oppmøte-brukergrensesnittet.




Øverst på dette skjermbildet vises en annen vurdering vi hadde i forhold til inndeling av programmodulene. Det er her brukt en type flate knapper med utheving på den valgte modulen.

Brukergrensesnittet nå ble gjort om til en klar inndeling i tre kolonner, for å forbedre oversikten. Alle funksjonsknappene ble lagt til i bunnen, for å samle disse på ett sted. Det ble også lagt til rammer rundt enkelte elementer, for å samle de som hører naturlig sammen.

Det ble etter hvert ytret ønske fra oppdragsgiver om å ha muligheten for å legge til bilde av hver enkelt elev. Dette for at instruktører kunne gå inn å gjøre oppslag på elever og samtidig få opp et bilde. Det vil da bli enklere for en instruktør å huske hvem de forskjellige elevene er, særlig for nye instruktører.

Det var på denne tiden også planer om å legge inn i dette systemet hvem som er satt opp som betaler av medlemmenes faktura, samt noe personalia og bankkontonummer. Dette ble senere eliminert, da dette skal være et program for administrering av elever og praktisk drift og ikke fakturering, da dette er en jobb som vil bli utført av regnskapssystemet.

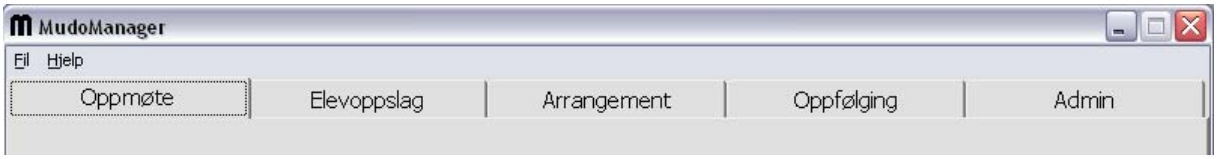

Dette er det endelige utseendet på modulen for elevoppslag. Her kan man i toppen se hvordan det endelige resultatet for vurdering av skillet mellom de ulike programmodulene er.

Endringene i denne i forhold til forrige skjermbilde er hovedsaklig byttingen av plassering for ”Oppslag” og bilde. Grunnen til at dette ble flyttet var for å her, som i oppmøte-modulen, ha all inntasting, som brukes ofte, i en kolonne til venstre, så den viktige informasjonen knyttet til dette i neste kolonne.

Det ble også lagt til en funksjon for å kunne registrere inn oppmøte på tidligere timer. Dette var etter ønske fra oppdragsgiver, for å kunne registrere treninger på en elev, selv om vedkommende glemte å registrere seg da treningen ble gjennomført.

### 4.7.3 Brukermiljø for vanlige brukere

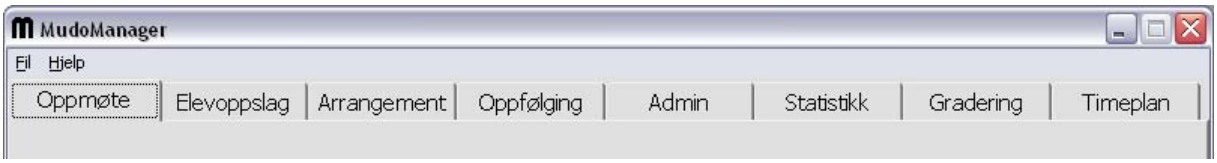
Vanlige brukere vil ha tilgang til modulene oppmøte, elevoppslag, oppfølging og arrangementer. Tilgangen på enkelte av disse vil også være noe begrenset, da vanlige brukere ikke kan endre elevinformasjon eller legge til arrangementer. Dersom en bruker velger en av disse funksjonene vil det komme opp en forklarende dialog med spørsmål om brukeren ønsker å logge inn.




*Arkfanene som vil være synlig for vanlige brukere*

### 4.7.4 Brukermiljø for administratorer

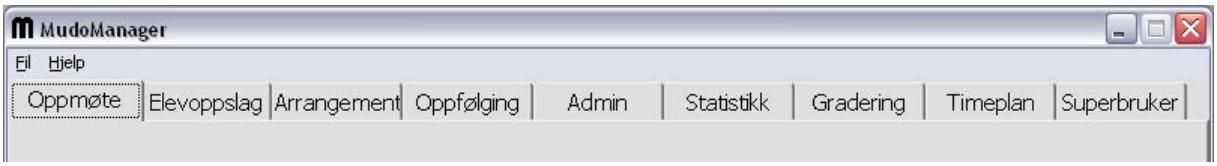
Ved innlogging vil administrator få opp 3 – tre – nye arkfaner, i tillegg til de en vanlig bruker har. Disse er for å aksessere informasjon om instituttet som ikke nødvendigvis er offisiell. En administrator vil også ha mulighet for å endre brukerdata for elevene.




*Slik en administrator vil se arkfanene*

### 4.7.5 Brukermiljø for superbrukere

Superbrukere vil i tillegg til arfanene for administratorer få opp en ekstra arkfane. Denne gir brukerne mulighet til å legge til data i tabellene ”Kontrakt”, ”Program” og ”Status”, som andre brukere ikke vil ha tilgang til. En superbruker kan i tillegg endre brukertilgangen for en annen bruker i systemet, samt endre instituttdata for et hvilket som helst institutt, eventuelt legge til et nytt. En superbruker vil også legge elevendringene direkte inn i den sentrale databasen, i motsetning til en administrator som automatisk vil sende en epost til den sentrale administrasjonen.




*Alle arkfanene som finnes i applikasjonen. Disse vil være synlig for superbruker*

# 5 Implementering

## *5.1 Utviklingsmiljø*

Deltakerne i gruppen stilte selv med 2 – to – bærbare datamaskiner, og fikk låne en maskin av skolen. Denne maskinen har stått som CVS-server for prosjektet

Maskinene har hatt installert:

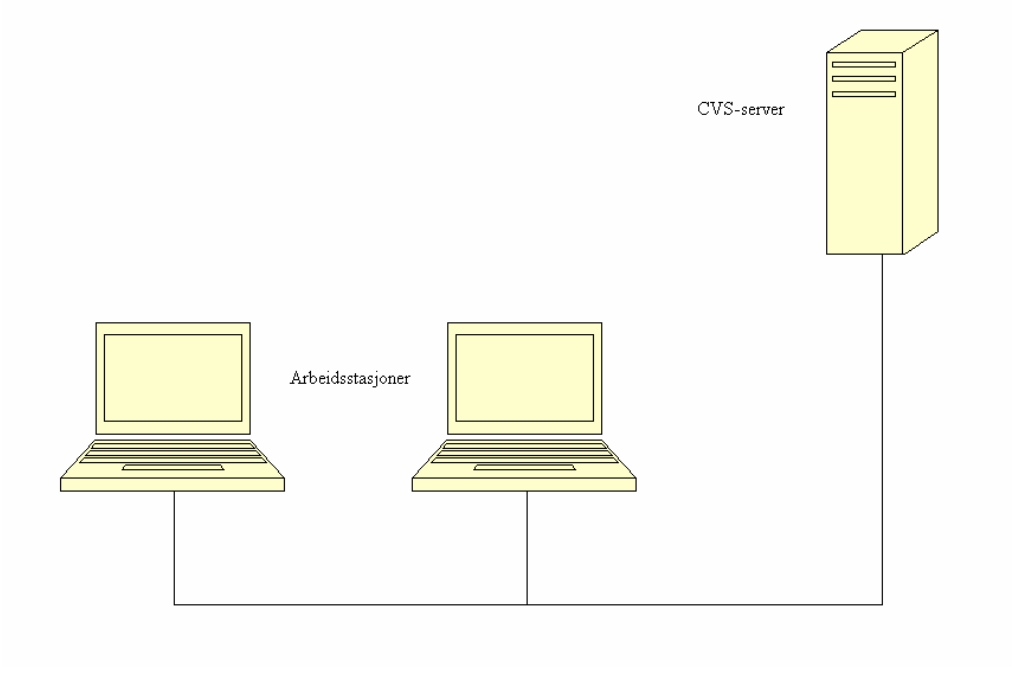
- Microsoft Visual Studio .net med Jalindi Igloo plugin
- Microsoft Office XP med Access
- Microsoft Outlook XP

Serveren har hatt installert:

- CVS server

Gruppen har også brukt maskiner på skolen for å benytte applikasjonene:

- Rational Rose
- Microsoft Project



## *5.2 Valg av verktøy*

### 5.2.1 Programmeringsverktøy

**Microsoft Visual Studio .net:**
Gruppen valgte tidlig å bruke Microsoft Visual Studio .net for utviklingen. Grunnen til at valget falt på MS VS var at gruppemedlemmene hadde brukt tidligere versjoner av dette og hadde den oppfatningen at det var et godt og komplett utviklingsverktøy. Gruppemedlemmene gikk derfor til innkjøp av lisenser for dette produktet.

Noen av de nyttige funksjonene i dette utviklingsmiljøet gruppemedlemmene har brukt under utviklingen av systemet:

- Muligheter for å dele inn kode i regioner
- Gode standarder for kommentering av kode
- Enkelt å holde orden på alle filer i en løsning / prosjekt.
- En ”call stack” som brukes under debugging, som viser alle funksjonskall som er ”aktive” da programmet stoppes.
- Enkelt å aksessere en databaseserver
- Innebygget stor hjelpefunksjon med dokumentasjon (MSDN)

Deltagerne i gruppen er etter denne perioden meget fornøyde og fortrolige med utviklingsverktøyet Microsoft Visual Studio .net.

**Jalindi Igloo plugin v1.03:**
Microsoft Visual Studio har et system for versjonskontroll, Microsoft SourceSafe. Da gruppen valgte å bruke en CVS-server i stedet for denne var de avhengige av en API mellom Microsoft Visual Studio og CVS serveren, siden denne serveren bruker et litt annet system for versjonskontroll enn det SourceSafe gjør. Bruken av dette plugin-et gjør at utviklerne kan bruke de funksjonene som er beregnet for SourceSafe i Visual Studio mot CVS-serveren.

Drøfting rundt valget av CVS fremfor SourceSafe finnes i neste kapittel.

### 5.2.2 Verktøy for versjonskontroll

Gruppen bestemte seg under planleggingen av prosjektet å bruke en Concurrent Versions Systsem-server (CVS) for å holde orden på alle prosjektets kildefiler og versjoner av disse. Serveren som ble brukt er standard CVS for Linux, versjon 1.11.1p1. Alternativet til dette var bruk av Microsoft SourceSafe eller eventuelt manuell backup og utveksling av kodefiler. Microsoft SourceSafe er utviklet direkte mot Microsoft Visual Studio, som ble brukt i utviklingen, og er derfor bedre integrert mot dette med tanke på funksjonalitet. Grunnen til at dette produktet ikke ble valgt til håndteringen av kodefilene var at lisens for dette produktet er ikke gratis. Alternativet med manuell utveksling av filer var ikke på noe tidspunkt aktuelt, da dette både er upraktisk og langt fra sikkert.

En CVS er en applikasjon som kjører på en maskin, som gir brukerne muligheter til å legge inn og hente ut siste versjoner av filer. Når en bruker skal editere en fil, henter han/hun denne på CVS-en, den blir da ”merket” som ”sjekket ut” på serveren. Brukeren legger den så inn igjen når editeringen er fullført. Filen på serveren blir da oppdatert slik at nyeste versjon ligger der. Det lagres også informasjon om hva som er endringer siden forrige versjoner, slik at dersom det oppstår feil, så kan man hente ut filene slik de var for en gitt tid siden.

### 5.2.3 Databaseverktøy

**Microsoft Access:**
For å opprette og editere på den lokale databasen har gruppen brukt Microsoft Access XP. Dette ble brukt mye i starten av prosjektet, men mindre etter hvert som applikasjonen fungerte som den skulle.

**Microsoft Visual Studio .net:**
Når den sentrale databasen var tilgjengelig ble denne aksessert fra Microsoft Visual Studio .net. Ingen endringer ble gjort på databasen herifra, kun innlegging, fjerning og endring av data.

## *5.3 Programmeringsspråk og rammeverk*

Ved prosjektstart sto valget mellom to programmeringsspråk, og to løsninger for GUI. Språkene det stod mellom var C++ eller C#, GUI-løsningene var rammeverkene .net eller MFC.

### 5.3.1 Kort om .net rammeverket

Microsoft® .net rammeverket er et nyutviklet komponent for Microsoft Windows® operativsystemer. Det er fundamentet for neste generasjon av Windows-baserte applikasjoner som er enklere å utvikle, anvende og integrere med andre nettverkssystemer. Dette rammeverket gjør det enklere å utvikle applikasjoner på tvers av systemer, som for eksempel lomme-PC’er, web-applikasjoner, server-applikasjoner og Windows bruker-applikasjoner, da det gir en felles tilnærming til disse. De samme verktøyene og kunnskapene brukes ved utvikling mot alle disse systemene.

Spesielt for .net-rammeverket i motsetning til utvikling med for eksempel MFC-klassebiblioteket er at alle applikasjoner utviklet under dette rammeverket blir ikke oversatt til direkte maskinkode, men til noe som kalles Common Language Runtime (CLI). Dette vil i praksis si at moduler i applikasjoner under .net vil kunne skrives i omtrent hvilket som helst programmeringsspråk, da rammeverket inneholder en støtte for over 20 språk. Disse kan kombineres i et prosjekt for å kombinere fordeler med de ulike språkene. Det vil også si at utviklere behøver i mindre grad å ta hensyn til minnehåndtering og feilhåndtering, som ofte er

en stor kilde til feil i programmer, da dette i stor grad blir håndtert av rammeverkets ”runtime environment”.

.net-rammeverket inneholder også et komplett klassebibliotek for alle de over beskrevne systemene for .net-applikasjoner. Dette inneholder følgende hovedelementer:

- ASP.NET – for utvikling av web-applikasjoner og web-tjenester
- Windows Forms – for utvikling av GUI for alle typer applikasjoner
- ADO.NET – for enkelt å samkjøre applikasjoner mot databaser

*(kilde: Microsoft® - www.microsoft.com)*

### 5.3.2 Kort om C#

C# (eng. ”C sharp”) er et relativt nyutviklet programmeringsspråk utviklet hovedsaklig mot .net-rammeverket. Dette er et språk som er utviklet for å gi en god balanse mellom ytelse og produktivitet på utviklernes side. Dette er et moderne, objektorientert språk som gir utviklere muligheten til å raskt lage et bredt spekter av ulike applikasjoner under .net-rammeverket.

Mer enn noe annet er C# utviklet for å gi C++ programmerere muligheten til å raskt kunne utvikle applikasjoner uten å måtte ofre den fleksibiliteten og de styrkene som har vært et kjennetegn for C og C++. På grunn av dette har C# mye til felles med C og C++. Utviklere, som er godt kjent med disse språkene vil raskt kunne sette seg inn i C#.

*(kilde: Microsoft® - www.microsoft.com)*

### 5.3.3 Kort om MFC-rammeverket

Microsoft Foundation Class biblioteket (MFC) er et  rammeverk for å programmere GUI til Windows®. Dette rammeverket er skrevet i C++ og gir utviklere den koden som behøves for å blant annet lage GUI-elementer, som vinduer, menyer, dialogbokser og mye mer, utføre grunnleggende input og output og lagre samlinger av dataobjekter.

Dette er en relativt gammel teknologi, og det er opp til utviklerne å måtte styre minnehåndtering og feilhåndtering.

*(kilde: Microsoft® - www.microsoft.com)*

### 5.3.4 Vurderinger og valg

I starten hadde gruppen egentlig bestemt seg for å utvikle applikasjonen i C++ med MFC rammeverket. Dette fordi C++ var et språk de begge hadde brukt tidligere, og det ville kun vært MFC som måtte læres. Gruppemedlemme bestemte seg alikevel for å undersøke hva .net rammeverket består av, samt fordeler og ulemper ved å bruke dette.

Fordeler ved MFC:

- Relativ rask
- Godt utprøvd og godt dokumentert
- Fungerer uten at et ekstra rammeverk må installeres på brukermaskinene, da dette er implementert i Windows®

Ulemper ved MFC:

- Utvikler må være veldig oppmerksom på minne- og feilhåndtering
- Relativt tungt å sette seg inn i

Forldeler ved .net:

- Lett å sette seg inn i
- Et nytt og fremtidsrettet utviklingsmiljø
- Styrer mer eller mindre automatisk minne- og feilhåndtering

Ulemper ved .net:

- Noe tregere enn MFC ved kjøring av applikasjoner
- Få store prosjekter er gjennomført i dette rammeverket, noe som gir lite eksempler på kode.
- .net-rammeverket på installeres på brukermaskinene (vil antageligvis bli inkludert i Windows XP ServicePack 2 og nyere systemer).

På bakgrunn av disse punktene gjorde gruppen den vurderingen å utvikle applikasjonen i .net-rammeverket. Dette fordi utviklingen vil gå lettere, særlig med tanke på GUI-elementer, database-kommunikasjon og minnehåndtering, i tillegg til at utviklerne ser på dette som noe meget fremtidsrettet og en god mulighet til å lære noe nytt.

Etter at valg om rammeverk var gjort kom det til valg av programmeringsspråk. Utviklerne stod også her, som beskrevet tidligere, ovenfor to valg. Under det valgte rammeverket kunne det programmeres i både C++ og C#. Da C# hovedsakelig er utviklet for .net-rammeverket ble dette et naturlig valg. Gruppemedlemmene så også her da muligheten for å lære noe helt nytt.

## *5.4 Kodestandarder*

Gruppen valgte å ha et standardoppsett på variabelnavn i GUI-filene. Formatet som ble bestemt var forkortelse for navnet på ”tab”-en, underscore (”_”), forkortelse for variabeltype og navnet på variablen. Det ble også bestemt at alle variabelnavn, funksjonsnavn og kommentering skulle være på engelsk. For eksempel:

- tsu_textStudentNumber – tab showUp (oppmøte), tekstboks som inneholder studentnummer
- tse_labName – tab studentEntry (elevoppslag), label som inneholder navn.

Som standard for indentering ble Microsoft Visual Studio sine standarder brukt. Det ble også brukt ”regions” rundt alle funksjoner, for å forenkle lesing og oversikt, samt bruk av Visual Studio sine standarder for kommentering av kode.

Det ble også bestemt at GUI-filene ikke skulle utføre minimalt med ”arbeid”. De skulle kan ha en referanse til sin Source-fil, der alt arbeidet skulle utføres, for så å oppdatere GUI-en.

**Regions:**
En enkel måte å gruppere kode-deler som hører sammen. Enten det er funksjoner, deler av funksjoner eller større områder. Man kan, etter å ha definert regioner med #region og #endregion, velge om hele koden i regionen skal vises, eller kun en egendefinert overskrift. For eksemplel vil kodeeksempelet i 5.5.1 se slik ut, dersom kun overskriften skal vises:

```
Function openExternalDatabase: Opens a connection to the central database
```

**Visual Studio sine kommenteringsstandarder:**

```
/// <summary>
/// Sends a query to the database to get student’s member ID
/// </summary>
/// <param name="name">Name of the student</param>
/// <returns>The student’s member ID</returns>
```

Dette eksempelet viser kun et utdrag av de muligheter som finnes ved bruk av denne kommenteringen. Dette er de delene som er brukt av gruppen. Mellom <summary></summary> finner man en kort forklaring av hva funksjonen utfører, <param name=xxx></param> Forklarer hva parameteret xxx er og <returns></returns> angir hva funksjonen returnerer. Dette kan også brukes til å lage en organisert fremstilling av alle klasser og dets funksjoner, i et prosjekt, i html-format.

## *5.5 Eksempler på kode*

Indentering kan være noe misvisende i disse kodeeksemplene, da flere av linjene er bredere enn sidebredden, men det gir et inntrykk av hvordan gruppen har skrevet koden. De følgende eksemplene er valgt fordi de tilhører meget forskjellige deler av applikasjonen, og kan være med på å gi et helheltlig bilde av hvordan de ulike oppgavene er løst.

### 5.5.1 DatabaseAPI – åpne tilkobling til sentral database

```
#region Function openExternalDatabase: Opens a connection to the central
database
/// <summary>
/// Opens a connection to the central database
/// </summary>
public static void openExternalDatabase()
{
      Microsoft.Win32.RegistryKey inkey;
```

```
        inkey =
        Microsoft.Win32.Registry.LocalMachine.OpenSubKey("software\\Mudora");

        externalDatabase = new OleDbConnection();
        externalDatabase.ConnectionString = @"Provider=SqlOleDb;" +
                    @"Data source=" + inkey.GetValue("IP").ToString() + ";" +
                    @"Initial catalog=********;" +
                    @"User Id=" + inkey.GetValue("UN").ToString() + ";" +
                    @"Password=" + inkey.GetValue("PW").ToString() + ";";

        externalDatabase.Open();
}
#endregion
```

Dette kodeeksemplet åpner en tilkobling til den sentrale databasen med rett adresse, brukernavn og passord. Dette er informasjon som hentes ut fra registeret, hvor denne informasjonen er lagret. Grunnen til at dette ble brukt som et eksempel er for å vise hvordan man enkelt kan lese inn og bruke verdier fra windows-registeret. Grunnen til at vi valgte å gjøre dette slik, var at alternativene var enten å lagre denne informasjonen på en fil, eller skrive det direkte inn i koden. Sistnevnte var i realiteten ikke et alternativ, og gruppen ville prøve å minimere bruken av filer til applikasjonen.

### 5.5.2 Oppfølging (FollowUp) – Skrive ut oppfølgingsliste

```
#region Function tfu_buttonPrint_Click: prints the contents in the listview
/// <summary>
/// prints the contents in the listview
/// </summary>
/// <param name="sender">not used</param>
/// <param name="e">not used</param>
private void tfu_buttonPrint_Click(object sender, EventArgs e)
{
      if(this.tfu_listViewFU.Items.Count == 0)
      {
            MessageBox.Show("Listen er tom", "Tom liste",
                  MessageBoxButtons.OK, MessageBoxIcon.Information);
            return;
      }
      string[] columnNames = new string[this.tfu_listViewFU.Columns.Count];
      string[,] printData = new string[this.tfu_listViewFU.Items.Count,
                                       this.tfu_listViewFU.Columns.Count];

      for(int i = 0; i < this.tfu_listViewFU.Columns.Count; i++)
      {
            columnNames[i] = this.tfu_listViewFU.Columns[i].Text;
      }
      for(int i = 0; i < this.tfu_listViewFU.Items.Count; i++)
      {
            for(int j = 0; j < this.tfu_listViewFU.Columns.Count; j++)
            {
                  printData[i,j] =
                  this.tfu_listViewFU.Items[i].SubItems[j].Text;
            }
      }
```

```
        try
        {
                followUp_s.Print(columnNames, printData);
        }
        catch(Exception exp)
        {
                MessageBox.Show(exp.Message, "Informasjon",
                        MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
}
#endregion
```

Denne funksjonen kalles når knappen ”skriv ut liste” trykkes i oppfølgings-delen av applikasjonen. Den henter ut alle kolonnenavn og innholdet i listen. Dette sendes til Source-filen som setter sammen all informasjon som behøves for å kalle en statisk metode i PrintList-filen, som tar seg av all utskrift av alle mulige lister i applikasjonen. Oppbygningen av variabelnavn vises også i dette eksempelet.

Siden denne listen viser både elever med mindre enn et gitt antall treninger og prospects, som har ulike data som vises og ulikt antall kolonner i listen, så viser denne funksjonen hvor enkelt det er å lage dynamiske funksjoner, som har flere bruksområder.

### 5.5.3 MudoManager – hovedfunksjon for applikasjonen

```
#region MAIN
/// <summary>
/// Main entry point for application. Runs InstituteChooser on first-time-
///run, then starts syncronizing the database.
/// Last it starts the main application
/// </summary>
[STAThread]
static void Main()
{
        if(inkey.GetValue("ID").ToString() == "")
        {
                System.Windows.Forms.Application.Run( new InstituteChooser() );
                Database.setInstituteUpdated(Mudora.getInstituteID(), false);
        }

        System.Windows.Forms.Application.Run( new Syncronizer(true) );

        Microsoft.Win32.RegistryKey inkey =
Microsoft.Win32.Registry.LocalMachine.CreateSubKey("software\\mudora");

        Microsoft.Win32.RegistryKey dbKey =
Microsoft.Win32.Registry.LocalMachine.OpenSubKey("software\\mudora\\resourc
es");

        int temp =
                Convert.ToInt32(inkey.GetValue("programStarts").ToString());
        if((temp % 10) == 0)
        {
                JRO.JetEngineClass jro = new JRO.JetEngineClass();
```

```
            jro.CompactDatabase(
                  @"Provider=Microsoft.Jet.OLEDB.4.0;Data source="
                  +dbKey.GetValue("db").ToString() + ";Jet OLEDB:Database
                  Password=*******;",

                  @"Provider=Microsoft.Jet.OLEDB.4.0;Data
                  source=mudoraDB.temp;Jet OLEDB:Database
                  Password=*******;Jet OLEDB:Engine Type=5;");
      }
      inkey.SetValue("programStarts", Convert.ToString(temp + 1));

      System.Windows.Forms.Application.Run( new Mudora() );
}
#endregion
```

Dette er hovedfunksjonen i applikasjonen. Denne sjekker først antall ganger applikasjonen har startet. Ved hver tiende start kjøres en funksjon som komprimerer den lokale databasen. Dette ble lagt til for å forhindre at databasen vokser enormt i størrelse etter en stund. Dernest sjekkes det om det ligger en verdi for ”ID” i registeret. Denne verdien angir hvilket institutt den aktuelle maskinen står på. Om det ikke ligger noen verdi der, så er det første gangen applikasjonen startes, og et vindu vises for å velge institutt, og kontrollere instituttinformasjon. Etter dette startes en liten tilleggsapplikasjon som synkroniserer den lokale databasen med den sentrale. Når dette er gjennomført startes hovedapplikasjonen.

Grunnen til at databasesynkroniseringen er lagt ut i en egen liten applikasjon er at denne skal vise en animasjon samt en fremdriftsviser, og når dette ble forsøkt samtidig med at hovedapplikasjonen ble kjørt oppstod det problemer med GUI-tråden.

### 5.5.4 TextEditor – Sende epost

```
private void sendEmail()
{
      MailMessage mail = new MailMessage();
      mail.From =
            DbInstitute.getInstituteData(Mudora.getInstituteID(),
            DbInstitute.InstituteData.RubiconID)[0] + " <" +
            DbInstitute.getInstituteEmail(Mudora.getInstituteID()) + ">";

      mail.Subject = "Informasjonsmail fra " +
            DbInstitute.getInstituteData(Mudora.getInstituteID(),
            DbInstitute.InstituteData.RubiconID)[0];

      mail.Body = this.mailContent + "\n\n\nmvh\n" + this.sender;
      mail.To = "";

      rtb.Dispose();

      for(int i = 0; i < studentNumbersEmail.Length; i++)
      {
            mail.To += DbStudent.getEmail(studentNumbersEmail[i]) + ";";
      }

      try
```

```
        {
                SmtpMail.Send(mail);
                MessageBox.Show(studentNumbersEmail.GetLength(0) +
                        " epostmeldinger ble sendt", "Sendt epost",
                        MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
        catch(Exception exp)
        {
                if(MessageBox.Show("En feil oppstod ved sending av epost. " +
                        "Vil du skrive ut brev til disse i stedet?\n\n" +
                        exp.Message, "Feil ved sending av epost",
                        MessageBoxButtons.YesNo, MessageBoxIcon.Error) ==
                DialogResult.Yes)
                {
                        studentNumbers = studentNumbersEmail;
                        startPrint();
                }
        }
}
```

Denne funksjonen blir kalt dersom bruker velger at det skal sendes epost til de elevene med dette lagret i databasen, etter å ha skrevet brev. Først opprettes en instans av MailMessage før alle atributter som behøves for å sende epost fylles inn. Elevnummeret til alle elevene det skal sendes til hentes ut fra en array, og epostadressene til disse hentes så ut ifra databasen. Avsenders epostadresse hentes også ut ifra databasen, fra informasjonen om instituttet. Etter at innholdet i eposten er lag til sendes denne.

Grunnen til at dette kodeeksempelet er tatt med er for å illustrere hvor enkelt det er å implementere epostsending i .net-rammeverket.

### 5.5.5 Admin – hashing av passord

```
public static byte[] MD5hash( byte[] data)
{
        System.Security.Cryptography.MD5 md5 =
                new System.Security.Cryptography.MD5CryptoServiceProvider();
        byte[] result = md5.ComputeHash(data);
        return result;
}
```

Denne delen av koden tar en array av bytes som parametre, en innebygd klasse for md5-hashing brukes for å returnere en hashet verdi av denne.

Grunnen til at dette kodeeksempelet er tatt med for å vise hvordan sikkerheten ivaretas i systemet.

# 6 Beskrivelse av utviklingsprosessen

## *6.1 Valg av systemutviklingsmodell*

Det var hovedsakelig tre utviklingsmodeller som ble vurdert fra utviklernes side. Disse tre var fossefallsmodellen, evolusjonær utvikling og inkrementell utvikling. En beskrivelse av disse tre ligger som vedlegg K, side LXI.

Fossefallsmodellen krever en klar og ferdigdefinert kravspesifikasjon og ingen prototyper er klare før utviklingen er ferdig. Evolusjonær utvilkingsmodell er åpen for endinger i alle moduler i alle faser av utviklingen, og kravspesifikasjonen tilpasses fortløpende. Inkrementell utvikling krever kun en overordnet skisse over modulene i applikasjonen, kravspesifikasjonen spesifiseres før hver modul, og ingen endringer skal utføres på en modul etter at inkrementet er gjennomført.

Det var relativt klart fra gruppens side i starten av prosjektet at det var mest hensiktsmessig å bruke modellen for inkrementell utvikling. En av grunnene til at dette var en praktisk modell i dette prosjektet var at kravspesifikasjonen fra oppdragsgivers side ikke var helt klar for de ulike inkrementene. Det var derfor praktisk å kunne sette opp en overordnet kravspesifikasjon, for så å spesifisere de ulike modulene etter hvert. En annen grunn som talte for bruk av denne modellen var det var naturlig å dele opp applikasjonen i moduler, da den består av relativt like store, ulike deler som utfører ulike oppgaver.

Noe både gruppen og oppdragsgiver så som fordeler med dette valget var at oppdragsgiver enkelt kunne komme inn med forslag og endringer underveis i utviklingen, i tillegg til at gruppen hadde ferdige moduler som kunne fremvises oppdragsgiver underveis, så de ble mer engasjert i utviklingen og kunne få deler av applikasjonen nøyaktig slik de hadde tenkt seg. Utviklerne ville også ha muligheten til å teste underveis, og å se del-resultater for motivasjonens del.

Gruppen bestemte seg også for å ikke slavisk følge en utviklingsmodell, men også å bruke moduler fra andre modeller der dette var passende. Utviklingsmessig er inkrementell utvikling fulgt fullt ut. Enkelte modeller og dokumentmaler er hentet inn fra RUP. Dette være seg Use Case diagram og beskrivelse, prinsippene bak sekvensdiagrammer og klassediagram (filstruktur). Hovedgrunnen til dette er at dette er modeller som skal være relativt enkle å forstå for ikke fullt så datakyndige lesere, samt at de gir en enkel og total oversikt over systemet.

Det er også hentet inn noen arbeidsprinsipper fra utviklingsmodellen XP. Gruppen har under hele utviklingen hatt en meget tilgjengelig oppdragsgiver som har kommet med tilbakemeldinger og forslag til forbedringer. Det har også blitt praktisert parprogrammering i de større og noe komplekse modulene i applikasjonen.

Under vurderingen av ulike utviklingsmodeller kom gruppen frem til at RUP var en modell som passer bedre til større grupper og større prosjekter enn det er snakk om i denne sammenheng. Spesielt med tanke på at gruppen består av to medlemmer, ble dette ansett som en lite hensiktsmessig løsninig.

Planlegging og tidsfrister var viktige i dette prosjektet. Derfor så også gruppen borti ifra XP som utviklingsmodell. Denne utviklingsmodellen resulterer også i lite dokumentasjon. Da denne applikasjonen med størst sannsynlighet skal videreutvikles senere, så gruppen verdien av relativt grundig dokumentasjon av systemet.

## *6.2 Planlegging og gjennomføring*

Gruppen startet prosjektet ved å sette opp et Gantt-skjema for å planlegge fremdriften i prosjektet. Etter et par møter med oppdragsgiver med litt klarlegging ble skjemaene relativt kraftig lagt om, da det kom til nye moduler, samt at noen ble fjernet. Da det også ble klart at applikasjonen skulle kjøres ved alle instituttene mot en sentral database, ble noen av inkrementene endret også her. Fremdriftsplanen ble holdt godt, men det ble noen endringer i den reelle fremdriften. Opprinnelig og reelt gantt-skjema finnes i vedlegg C, side VIII

Selv om det kom mye endringer i starten, samt også noen endringer underveis, da blant annet et par moduler ble slått sammen, så har gruppen fulgt tidsskjemaet angitt på den opprinnelige fremdriftsplanen godt. Gruppemedlemmene er totalt sett fornøyde med gjennomføringen av prosjektet.

# 7 Diskusjon av resultater

## *7.1 Vurdering av resultater*

I dette kapittelet vil resultatet av prosjektarbeidet vurderes.

### 7.1.1 Det endelige resultatet

Det endelige resultatet tilfredsstiller alle krav gitt fra oppdragsgivers side, og er klar for implementering hos instituttene. Begge gruppens medlemmer har bidratt med stor innsats og pågangsmot for å oppnå det endelige resultatet. Applikasjonen vil implementeres hos et par av kjedens institutter for test-kjøring paralellt med kjøring av det gamle systemet i en måned. Om applikasjonen fungerer tilfredsstillende vil den bli implementert hos alle kjedens institutter, eventuelt vil utviklerne rette opp i de feil og mangler som oppdages i test-perioden.

I tillegg til de opprinnelige kravene gitt fra oppdragsgivers side er også flere av oppdragsgivers ønsker til tilleggsfunksjoner tatt med i applikasjonen. Blant annet har gruppen implementert mulighet for å ta bilde av de enkelte elevene ved hjelp av webcamera.

Det er også implementert en enkel teksteditor for utskrift av egendefinerte brev til instituttets medlemmer. I tillegg vil det, dersom feil oppstår i systemet og brukeren velger dette, sendes en epost til utviklerne om denne feilen. I tillegg til hva som var opprinnelige krav fra oppdragsgivers side, er det også implementert en modul som sender epost til sentraladministrasjonen ved endring av elevdata og lignende.

Det ble også valgt å passordbeskytte deler av applikasjonen, slik at ikke alle brukere har tilgang til alle funksjoner.

Det har dessverre ikke blitt tid til å sette i gang testing før prosjektet avsluttes. Testperioden er fastsatt, fra Mudo Instituttet AS,  til å starte 20. mai. Dette er en avgjørelse tatt for lenge siden av oppdragsgiver sammen med utviklerne.

### 7.1.2 Oppgaver som kunne vært løst annerledes

Hadde gruppemedlemme sittet med den samme kunnskap rundt enkelte elementer før prosjektstart som de kan i ettertid, er det noen oppgaver som ville vært løst annerledes. Under følger en oversikt over disse.

**Egendefinere kontroller**
Hver arkfane i applikasjonen er en ”TabPage” som inneholder en eller flere ”UserControl”. Alle GUI-elementer ligger i UserControl-en. Hver TabPage har også en referanse til sine kontroller.

Ved å ha laget egendefinerte TabPages og UserControls kunne en del standardfunksjoner ha blitt lagt inn i disse, som for eksempel en funksjon for å sette markøren til rett tekstboks når tab-en velges. Med dagens løsning gjøres dette på følgende måte (utdrag av kode):

```
private void mainTab_SelectedIndexChanged(object sender, EventArgs e)
{
      switch(this.mainTab.SelectedTab)
      {
            case this.tabShowUp:
                  this.showUp.setCaret();
                  break;
            case this.tabStudentEntry:
                  this.studentEntry.setCaret();
                  break;
            (...)
      }
}
```

Ved å utvikle en egendefinert TabPage og UserControl kunne dette blitt løst på følgende måte (fullstendig kode):

```
private void mainTab_SelectedIndexChanged(object sender, EventArgs e)
{
      this.mainTab.SelectedTab.Control.setCaret();
}
```

Ved å gjennomføre dette vil koden bli enklere og mer oversiktlig, samt at eksekveringstiden vil bli kortere.

Ved å egendefinere TabPage kontrollen, samt andre kontroller som for eksempel knapper og menyer vil det være større mulighet for å endre utseendet utover standarden for disse.

## *7.2 Alternative muligheter og valg underveis*

### 7.2.1 Implementering av webcamera

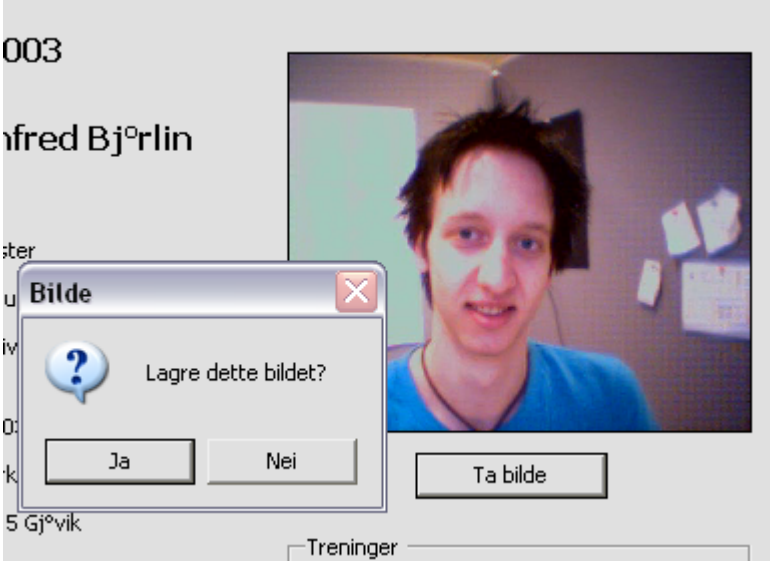
Denne modulen var ikke i oppdragsgivers opprinnelige kravspesifikasjon, men kom som et ønske etter at utviklingen var satt i gang. Med dette ønsket oppdragsgiver at man på hvert enkelt institutt kunne ta bilder av den enkelte elev. Dette for at instruktører kunne gå inn i applikasjonen, slå opp på et elevnummer, eller søke på navn, og sammen med elevinformasjon få opp et bilde av eleven.

Valgene gruppen stod ovenfor da dette skulle implementeres var å enten bruke et eksternt program for å ta bildet, for så å legge dette inn i databasen, eller å implementere en modul i selve applikasjonen som tar bildet og legger det direkte inn i systemet.

Dersom første alternativ skulle velges ville dette innebære mer jobb for brukeren, samt bruk av tredjeparts programvare, som gruppen helst ville unngå. Bruker måtte da legge bildet på rett plass, samt legge inn rett bildenavn i systemet. Dette åpner for større mulighet for at det vises feil bilde for elevene og at bildet ikke finnes, fordi det er lagret feil filnavn, eller lagret i feil katalog. Dette hadde vært en enkel løsning for utviklerne, men en tilsvarende mer arbeidskrevende for brukerne.

Å implementere en modul direkte i applikasjonen som tok hånd om bildetagning og lagring i systemet ville eliminere problemene med førstnevnte løsning. Dette ville også samle alle systemets funksjoner innen selve applikasjonen; noe som var et av målene for prosjektgruppen allerede fra starten av prosjektet. Etter denne vurderingen var gjort ble sistnevnte løsning valgt for applikasjonen.

Valget av denne løsningen innebar at gruppen måtte sette seg noe inn i USB og bruken av DirectX. Det ble funnet en løsning som tilnærmet oppfylte gruppens behov, men var mer omfattende enn det som var nødvendig for å gjennomføre dette. Denne kilden ble da noe modifisert for å passe inn i applikasjonen. Både kilden for denne løsningen og utviklingsmiljøet til DirectX er gratis for nedlasting og bruk. Denne kilden ble innhentet fra CodeProject.



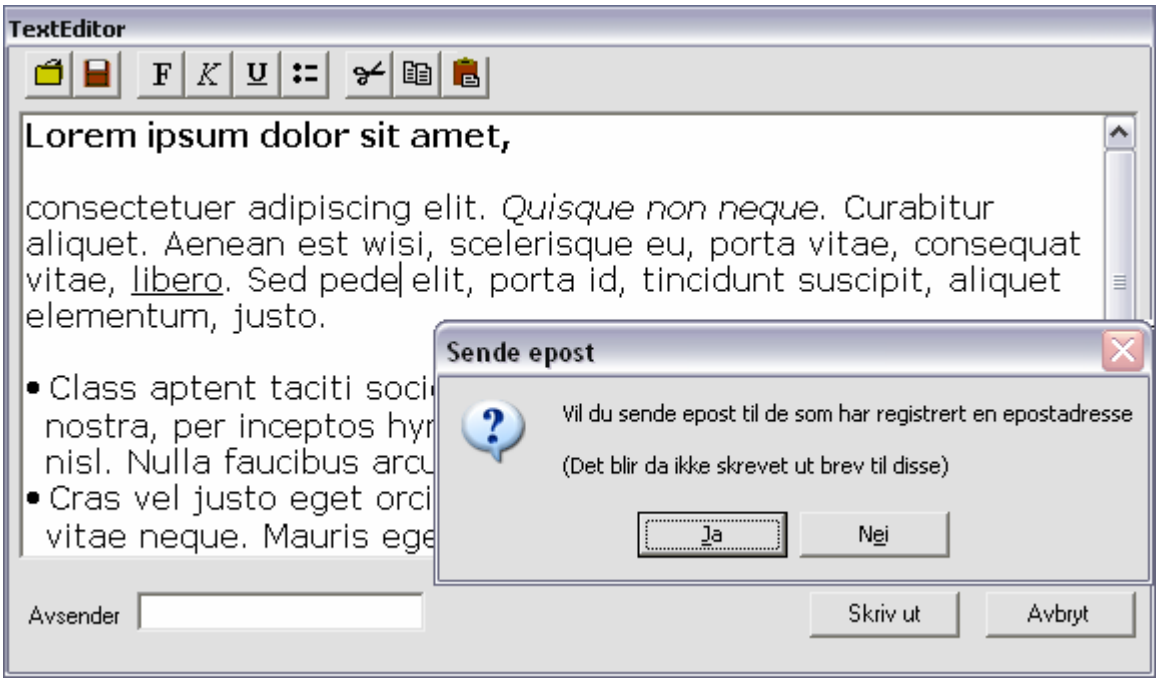

### 7.2.2 Implementering av enkel teksteditor

Etter ønske fra oppdragsgiver, så ble det også implementert en enkel teksteditor i systemet. Grunnen til dette var at brukeren skulle slippe å bruke et annet program for å skrive brev. Som forklart i kapittel 7.1 var det et ønske både fra opdragsgivers og utviklernes side å samle all funksjonalitet i det nye systemet.

Også ved denne modulen stod gruppen ovenfor to valg. Enten kunne det utvikles en løsning som fletter applikasjonen mot Microsoft Word, eller så kunne det lages en modul som kunne bli implementert direkte inn i applikasjonen. Gruppen valgte også her den løsningen som innebar at all funksjonalitet ble lagt inn i applikasjonen.

For å relisere dette brukes den innebygde kontrollen for Rich Text Format (rtf) som er implementert i .net rammeverket. Denne hadde dog ingen funksjon for utskrift av det gjeldende dokumentet. Dette innebar at gruppen måtte selv lage en funksjon som kunne skrive ut dette, samt legge til adressat, Mudo instituttets logo og bunntekst.

Utfordringen gruppen støtte på under utvikling av denne var å få med tekstformatering og lister slik det var skrevet av brukeren.




Det ble også lagt til muligheter for lagring av disse brevene, for senere bruk, samt muligheter for automatisk sending av epost til de medlemmene som har dette registrert i databasen. Dette var et ønske fra oppdragsgiver for at instituttene skulle spare penger på papir og porto ved sending av brev.Brukeren får ved utskrift et spørsmål om det skal sendes epost til de aktuelle medlemmene. Det skrives da kun ut papirkopier til de som ikke har en epostadresse.

### 7.2.3 Implementering av modul for epostsending

Det ble på tre områder implementert muligheter for sending av epost. Første bruksområde er beskrevet i forrige kapittel. Epostsending ble også implementert for sending av feilrapporter til utviklerne og for å sende melding om endringer av elevdata til sentraladministrasjonen i Mudo Instituttet AS.

Dersom en feil oppstår i systemet, vil brukeren få opp et vindu med en kort beskrivelse av feilen, samt et tekstfelt slik at brukeren kan skrive inn hva som ble gjort før feilen oppstod. Når brukeren da velger ”send” vil det sendes en epost til utviklerne, som inneholder:

- Hvilket institutt feilmeldingen kommer fra
- Om noen er logget inn, eventuelt hvem dette er
- Dato og klokkeslett for feilen
- Windows-versjon
- Selve feilmeldingen

- Brukerens beskrivelse av hva som ble gjort
- En oversikt over hvor i applikasjonen feilen oppstod

Dette vil gi utviklerne større muligheter for å rette opp i eventuelle feil, og kan finne ut om det er brukerfeil som fører til at denne feilen oppstår.

Alternativet til denne løsningen ville være å gi brukeren en relativt stor og utfyllende feilmelding på skjermen, som brukeren så måtte notert ned, for å etterpå ta kontakt med utviklerne. Gruppen anså dette som en urealistisk og lite praktisk løsning, da man ikke kan forvente at de som bruker systemet nødvendigvis er spesielt datakyndige.

Grunnen til at det ble valgt å sende epost til sentraladministrasjonen ved endring av elevdata var at hele dette systemet skal samkjøres med Mudo Instituttets regnskapssystem. Hovedsaken av all elevinformasjon skal derfor oppdateres i dette systemet, som så automatisk skal oppdatere medlemsdatabasen. Eposten som da sendes sentraladministrasjonen vil inneholde:

- Medlemsnummer på den eleven det skal endres data om
- De dataene som skal endres om denne eleven
- Institutt denne eposten sendes fra
- Hvilken bruker som sender denne eposten

Det var to andre alternativer for dette: enten kunne instituttlederne selv endre dataene direkte inn i medlemsdatabasen, eller så kunne de sende en epost manuelt til sentraladministrasjonen. Om instituttlederne selv skulle oppdatert medlemsdatabasen ville det raskt ha blitt en forskjell i innholdet mellom databasen i regnskapssystemet og medlemsdatabasen. Den siste løsningen ville innebære ekstra oppgaver for instituttlederne, samt at dette hadde vært oppgaver som raskt kunne blitt glemt.

### 7.2.4 Passordbeskyttelse

Da enkelte av modulene i systemet ikke skulle være tilgjengelig for alle brukerne av systemet, ble det valgt å passordbeskytte enkelte av delene i applikasjonen. De delene som ble beskyttet var oppslag på statistikk, alle funksjoner rundt gradering, innlegging av timeplan, samt mindre oppgaver som endring av brukertilgang, instituttdata, passord og databaseinformasjon, og tillegging av data i enkelte av tabellene i databasen.

I de ”åpne” delene av applikasjonen er det også noen begrensninger for brukere som ikke er logget inn. Det er kun mulig å gjøre oppslag på elever ved det aktuelle instituttet. En administrator kan gjøre oppslag på elever ved det aktuelle og ”sitt” institutt. Denne brukeren kan også gjøre endringer i elevdata og ta bilde av elever. En superbruker har tilgang til alle elever i denne modulen.

Alle passord i systemet blir md5-hashet før de lagres i databasen, som igjen er sikret med passord. Grunnen til at passordene er md5-hashet er for at ingen andre brukere skal på noen måte få tilgang til disse gjennom databasen.

### 7.2.5 Implementering av egenutviklet TextBoxControl

En TextBox er GUI-elementet for en tekstboks. Da denne ikke innehar noen mulighet for direkte input-kontroll anså gruppen det som en fordel å utvikle en egen versjon av denne. Dette for å maskere slik at en tekstboks kun godtar enten tall, bokstaver, poststed (0000 [a-Å]*) eller telefonnummer (på formen 00 00 00 00 eller 000 00 000). Dette hindrer brukere å skrive feil inn i feltene hvor denne er brukt.

### 7.2.6 Valg av språk

Valg av programmeringsspråk og rammeverk for utviklingen er beskrevet i kapittel 5.3

## *7.3 Endringer i forhold til opprinnelig kravspesifikasjon og hvorfor*

Hovedendringen fra den opprinnelige kravspesifikasjonen er at det da ikke var tatt høyde for en sentral database. Det var i første omgang snakk om et prosjekt som skulle gjennomføres for et av instituttene i kjeden, Mudo Gjøvik. Setraladministrasjonen så muligheten for nå å kunne oppdatere hele systemet, og løse mange av de problemene som eksisterte i det nåværende systemet; særlig med tanke på treninger på tvers av instituttene, samt bedre muligheter for en total oversikt over medlemmene i kjeden.

En annen av de store forskjellene på det ferdige resultatet og den opprinnelige kravspesifikasjonen er at lite eller ingen personalia endres direkte inn i medlemsdatabasen, men gjennom regnskapssystemet. Denne løsningen ble valgt for å unngå forskjeller i medlemsinformasjon i de to databasene.

I den opprinnelige kravspesifikasjonen var det planlagt en avtalekalender til bruk for instituttenes ledere. Denne funksjonen så Mudo Instituttet ikke nytten av, så modulen ble fjernet. Grunnen til at denne modulen var med i den opprinnelige kravspesifikasjonen var at denne funksjonaliteten fantes i det eksisterende systemet. Oppdragsgiver mente denne ikke var i bruk hos noen av instituttene, og var derfor ikke nødvendig å ha med.

## *7.4 Utvidelses og forbedringsmuligheter*

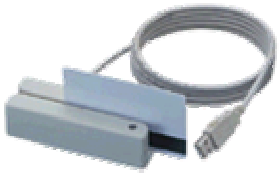
I dette kapittelet vil det drøftes ulike muligheter for utvidelser og forbedringsmuligheter i applikasjonen ved senere versjoner, da gruppen har en avtale med oppdragsgiver om videreutvikling av systemet.

Flere forslag til utbedringer enn de omtalt i dette kapittelet vil antageligvis komme etter hvert som oppdragsgiver tester systemet, og ser flere muligheter.

## 7.4.1 Implementering av fingeravtrykkleser eller kortleser

Fra oppdragsgivers side har det vært ytret et ønske om implementering av fingeravtrykksleser eller kortleser for medlemskort. Dette for å forenkle registrering av oppmøte ved treninger. Gruppen har udersøkt og vurdert ulike aspekter rundt disse to løsningene. Det er kommet frem til en anbefaling fra utviklernes side etter en vurdering av fordeler og ulemper ved disse.

**Kortleser:**



Dette var den første muligheten som ble tenkt på fra oppdragsgivers side. Dette er en løsning som brukes ved mange andre treningssentre av ulike slag, så dette er en teknologi man er sikker på vil fungere som ønsket. Etter å ha foretatt undersøkelser rundt bruk og implementasjon av denne løsningen kom gruppen frem til følgende fordeler og ulemper ved dette:

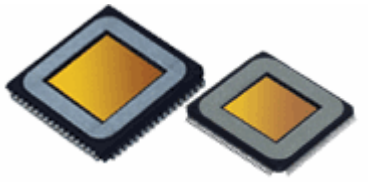
Fordeler ved bruk av denne teknologien:

- Godt kjent og godt utprøvd teknologi
- God tilgang på ressurser rundt utvikling

Ulemper ved denne løsningen:

- Relativt dyrt
- Krever en del utstyr
- Lett for elever å glemme, miste eller ødelegge kort.
- Kort må trykkes opp sentralt for så å sendes ut til instituttene

**Fingeravtrykksleser:**



Denne løsningen ble først fremmet nærmest som en spøk i et møte mellom oppdragsgiver og gruppens medlemmer. Utviklerne tok på seg jobben å undersøke de ulike aspektene rundt dette. Denne teknologien er ennå lite brukt og derfor ikke så kjent blant de fleste. Gruppemedlemmene så på dette som en spennende og utforende idé og begynte jobben med å undersøke leverandører og implementasjonsmuligheter. De kom så i kontakt med et firma i Sverige, Comptronic AB, som leverer utstyr og utviklingspakker for slike løsninger. Etter en gjennomgang av den innsamlede informasjonen kom gruppen frem til følgende fordeler og ulemper ved bruk av denne teknologien:

Fordeler:

- Lite eller ikke noe vedlikehold på enhetene og lang holdbarhet
- Fingeravtrykk kan lagres direkte ved det enkelte institutt inn i den sentrale databasen
- Krever ikke noe ekstra utstyr, annet enn selve leseren
- Umulig å glemme fingeren hjemme
- Rimelig i innkjøp
- Godt dokumentert for enkel utvikling

Ulemper:

- Ny og relativt ukjent teknologi for mange

**Drøfting av resultater:**
På bakgrunn av all den innsamlede informasjonen og vurderinger med fordeler og ulemper har gruppen kommet frem til den konklusjonen at det både vil være rimeligere å kjøpe inn det utstyret som trengs i tillegg til at utviklingskostnader vil være lavere ved valg av fingeravtrykkslesere fremfor kortlesere. Utviklerne ser også på dette som en ny og spennende teknologi de kan tenke seg å sette seg inn i og lære mer om.

Holdbarhetstester på fingeravtrykksleserne viser at disse i daglig bruk, om de brukes hvert femte minutt 20 timer i døgnet vil holde godt over 10 år. De er også testet kjemisk og termisk i tillegg til impakttester og slitasjetester. De kjemiske testene har innebåret tester med kaffe, håndkrem og andre kjemikalier som kan komme i nærheten av enhetene ved normal bruk. Impakttestene har blant annet innebåret tester med kulepenner og andre ”ikke-deformerbare objekter”. (Informasjon innhentet fra Authentec’s testdokumenter av egne enheter)

Gruppen går derfor inn å anbefale implementasjon av fingeravtrykkslesere fremfor kortlesere i systemet.

### 7.4.2 Fullstendig kjøring mot sentral database uten lokal kopi

Slik det nyutviklede systemet er så ligger all medlemsinformasjon på en sentral databaseserver. Ved oppstart av systemet synkroniseres den lokale databasekopien mot den sentrale serveren. Dette fører til at programmet bruker lang tid på å starte opp, da relativt mye data skal hentes ned fra den sentrale databaseserveren.

Dersom det kan garanteres at alle institutter har en relativt god internettforbindelse, vil det være å anbefale kun kjøring mot den sentrale database, uten noen lokale kopier. Dette vil gjøre oppstart av programmet raskere, og ingen merkbar økning i responstid for brukerne.

## *7.5 Evaluering av eget arbeid*

### 7.5.1 Innledning

Gruppemedlemmene har kjent hverandre siden første året på HiG og har jobbet sammen på flere prosjekter tidligere. Det ble da naturlig å jobbe sammen også på dette prosjektet. Da medlemmene tidligere også har vist at de jobber godt sammen på prosjekter gikk de ut ifra at gjennomføringen av dette også ville gå bra, og at arbeidsfordelingen skulle bli jevn. Gruppens medlemmer var også godt kjent med hverandres kunnskapsområder og interessefelt innen programmering; noe som var med på å lette fordelingen av modulene i prosjektet.

Under hele prosjektperioden har alle vurderinger blitt gjort fortløpende, så gruppemedlemmene har ikke hatt noen interne møter.

### 7.5.2 Organisering

Da gruppen kun bestod av to medlemmer, så de ingen nødvendighet for å tildele en oppgaven som gruppeleder. I stedet valgte de å fordele ansvarsområder. Dette er nærmere beskrevet i kapittel 1.5.2 – Prosjektorganisering. Denne planlagte fordelingen fungerte som den skulle. Utover fordeling av disse ansvarsområdene hadde gruppen en flat struktur; noe som fungerte meget bra.

### 7.5.3 Arbeidsfordeling

I starten av perioden forsøkte medlemmene av gruppen å fordele oppgavene likt mellom seg. Et av problemene med dette var at det var ikke helt klart hvilke moduler som skulle med og ikke i applikasjonen. På tross av dette har den totale arbeidsmengden vært tilnærmet lik for gruppemedlemmene. All rapportering har blitt utført i fellesskap, samt en del koding. Mange av de større modulene i applikasjonen er et resultat av godt samarbeid mellom gruppemedlemmene.

Endelig timeantall for gruppemedlemmene:

| | |
|---|---|
| Simen Hiken Galtestad | 590 timer |
| Petter Manfred Bjørlin | 589 timer |
| **Totalt** | 1179 timer |

### 7.5.4 Prosjekt som arbeidsform

Gruppemedlemmene har tatt med seg lærdom fra tidligere prosjekter. Dette var noe som har vært med på å gjøre gjennomføringen av dette prosjektet organisert og strukturert. Selv om dette prosjeketet har vært mye mer omfattende enn noe tidligere prosjekt medlemmene har gjennomført, så har denne lærdommen fra tidligere prosjekter vært til stor hjelp.

Medlemmene har i fellesskap, og sammen med oppdragsgiver, måttet finne frem til tilfredsstillende løsninger for alle parter. Dette er noe gruppen har tatt lærdom av, og vil ha god nytte av senere.

Ut ifra tidligere og nylig tilegnede kunnskaper om prosjekt er dette en arbeidsform som passer gruppemedlemmene godt.

### 7.5.5 Problemer underveis

Gruppen har opplevd overraskende lite problemer underveis i gjennomføringen av prosjektet. Det eneste problemet direkte knyttet til prosjektgjennomføringen gruppen opplevde var å måtte vente på noen avgjørelser fra oppdragsgiver da det ble bestemt at prosjektet skulle omfatte hele kjeden av institutter, og ikke bare et institutt. Dette førte til en liten senking av arbeidstempo en kort periode; noe som ble tatt inn igjen relativt raskt etter at arbeidet kom i gang for fullt igjen.

Et annet problem gruppen har slitt med på slutten av perioden er sen start av rapportskriving. Det er her mye som kunne vært gjort underveis, men som har blitt utsatt. Mye av grunnen til dette kan være at gruppen under hele perioden har sittet sammen, og tatt beslutninger forløpende, så dette har ikke blitt notert underveis.

### 7.5.6 Tilegnet kunnskap

Gjennom arbeidet med prosjektet har gruppen fått erfaring med håndtering av større prosjekter og mange av aspektene rundt dette. Medlemmene i gruppen har tilegnet seg kunnskap rundt formelle møter med oppdragsgiver, formulere og forhandle frem kontrakt, tilegne kunnskap ut ifra ressurser / fora på internett, jobbe sammen med eksterne ressurspersoner, et nytt rammeverk og utviklingsmiljø i .net, et nytt programmeringsspråk - C# -, programmering med bruk av CVS og utarbeidelse av prosjektrapport.

## *7.6 Konklusjon*

Selv om ingen av gruppemedlemmene hadde hatt noen erfaring med prosjekter av denne størrelsen, har gjennomføringen, med tanke på tidsfrister og kontakt med eksterne aktører gått meget bra. Medlemmene i gruppen har fått erfaring i både kontakt med en oppdragsgiver, jevnlig kontakt med en veileder, samt samarbeid med en ekstern utvikler. Systemet ble utviklet i tråd med de krav satt opp fra oppdragsgiver, og lå klart til den avtalte tiden. Selv om enkelte små problemer underveis har bremset ned arbeidet har de fleste tidsfrister underveis har blitt holdt.

Utviklerne ser i ettertid ulike oppgaver som kunne vært løst annerledes, både med tanke på kode og rapportering, men tar dette som en lærdom til senere utviklingsprosjekter. De har også kommet i kontakt med fagområder og utviklingsrammeverk som har vært helt nye og ukjente for medlemmene i gruppen før prosjektet. Også på dette punktet har gruppemedlemmene tatt til seg mye og nyttig lærdom, som kan komme til nytte senere.

Gruppemedlemmene ser frem til at testperioden, med planlagt start 20. mai, kommer i gang, for å se hvordan systemet vil fungere i praksis. De ser også fremover mot videreutvikling av systemet, med videre definering av oppgaver og implementering av blant annet fingeravtrykksleser.

Medlemmene i gruppen er også fornøyd med valget som ble gjort med hensyn til utviklingsmodell. De er også fornøyd med måten de kombinerte den valgte modellen med elementer fra andre modeller. Dette er en modell som har passet både prosjektet og gruppemedlemmene godt.

Gruppen anser det også som en verdifull arbeidserfaring, både med tanke på den utvikling som er gjort i prosjektperioden og den videre utviklingen, som vil innebære vedlikehold, oppdatering og videreutvikling av systemet.

Løsningen som er utviklet anser gruppens medlemmer som meget tilfredsstillende og håper Mudo Instituttet også er fornøyd og vil bruke denne i lang tid fremover.

# 8 Kilder

**Litteratur:**


| *Forfatter* | *Tittel* | *Forlag* | *ISBN* |
|---|---|---|---|
| Templeman / Vitter | The .NET Framework Black Book | Coriolis | 1-57610-995-X |
| Connolly / Begg | Database systems – Third edition | Addison Wesley | 0-201-70857-4 |
| Sommerville | Software Engineering – 6th edition | Addison Wesley | 0-201-39815-X |


**Internett:**


| | |
|---|---|
| CodeProject | www.codeproject.net |
| Hardware.no forum | forum.hardware.no |
| Neowin forum | www.neowin.com/forum |
| Microsoft MSDN | www.msdn.com |

# 9 Vedlegg

| Kapittel | Vedlegg | Side |
|---|---|---|
| Ordforklaringer | A | I |
| Forprosjektrapport | B | II |
| Gantt-skjemaer | C | VIII |
| Statusrapporter | D | XI |
| Eksempler på logger / møtereferater | E | XIV |
| Kravspesifikasjon | F | XVI |
| Sekvensdiagrammer | G | XXX |
| Brukermanual | H | XXXIX |
| Innholdet på CD-en | I | LV |
| Utskrifter fra applikasjonen | J | LVI |
| Forklaring av Systemutviklingsmodeller | K | LXI |