HOVEDPROSJEKT:

SkateBase

FORFATTERE:
Rune Sørensen
John Rasmus Moen
Nina Ruthenbeck Hansen

Dato:
19.05.2003

Sammendrag av hovedprosjekt

Tittel:	SkateBase	Nr. :
		Dato : 19.05.03
Deltaker(e):	Rune Sørensen	
	John Rasmus Moen	
	Nina Ruthenbeck Hansen	
Veileder(e):	Harald Liodden	
Oppdragsgiver:	Norges Skøyteforbund (NSF)	
Kontaktperson:	Kjell Ingar Myren	
Stikkord (4 stk)	Database, ASP, Java, Skøyteløp	

Antall sider: 57	Antall bilag: 13	Tilgjengelighet (åpen/konfidensiell): Åpen

Kort beskrivelse av hovedprosjektet:

Systemet vi har utviklet består av en informasjonsdatabase med tilhørende Webgrensesnitt og et java program. Webgrensesnittet muliggjør søk etter norske skøyteløpere, deres prestasjoner og personlige rekorder, spesifikke stevneresultater med mer. Klubber har mulighet for å logge seg inn på systemet og melde på sine løpere til bestemte arrangement. Når påmeldingsfristen er løpt ut, kan arrangør laste ned en påmeldingsfil som igjen kan benyttes i java programmet. Programmet hjelper til med registrering av forfall, resultater og oppsett av parsammensetning. I etterkant av arrangement genereres en resultatfil som arrangør kan laste tilbake til Webgrensesnittet som igjen oppdaterer informasjonsdatabasen.

Forord

Hovedprosjektet Skatebase er et avsluttende prosjekt (6vt) ved den treårige høgskoleingeniørutdanningen ved Høgskolen i Gjøvik våren 2003. Gruppemedlemmene på prosjektet har vært Nina Ruthenbeck Hansen, John Rasmus Moen og Rune Sørensen.

Vi bestemte oss tidlig på høsten 2002 for at vi skulle danne en prosjektgruppe i forbindelse med hovedprosjektet, så hele høsten prøvde vi å klekke ut noen gode ideer til en interessant oppgave. John Rasmus tilhører skøytemiljøet på Hamar, og kunne fortelle oss hvor tungvint det er å arrangere et skøytestevne. Påmeldinger, etteranmeldinger, trekning, manuell tidtaking og liknende fører til mye papirarbeid og er svært tidkrevende. Vi fant ut at et prosjekt for å lette dette arbeidet kunne være en interessant oppgave. I oktober kom vi i kontakt med Norges Skøyteforbund og de kunne fortelle oss at de hadde opprettet en forprosjektgruppe som hadde som mål å finne ut muligheten for å realisere et slikt prosjekt, og kostnadene dette ville medføre. Vi ble da enig med NSF at vi skulle utvikle en prototyp for dem.

Etter at avtalen med NSF ble undertegnet har det blitt lange arbeidsdager med intens jobbing og til dels fortvilelse under arbeidet. Ser vi tilbake på kveldene hvor vi slet som verst, skjønner vi nå at prosjektet har vært svært lærerikt. Prototypen som vi sitter igjen med, oppfyller visjonen fra høsten 2002 for hva en interessant oppgave kunne være. Kunnskapen prosjektet medførte er uvurderlig for oss, og ser vi på resultatet kan vi ikke annet enn å si oss fornøyde med det.

Vi vil takke følgende personer for hjelp og støtte under hovedprosjektet:

- Harald Liodden (Veileder)
- Erik Busterud (HEGO Timing)
- Øyvind Kolloen (Tips og triks på ASP delen)
- Hans B. Moen (Tips og testing)

Gjøvik den 19.05.2003

Rune Sørensen	John Rasmus Moen	Nina Ruthenbeck Hansen

Innhold

1 Innledning

1.1 Problemområde

Registrering av resultater for skøyteløp gjøres i dag på mange måter. Det benyttes blant annet regneark og diverse andre standarder. Det finne svært få programmer beregnet på skøyteløp. ”IsTid” er et program utviklet til formålet, men det er DOS-basert, og trenger eventuelt en oppgradering til Windows om det skal ha noen fremtid.

Arrangører av skøyteløp sender resultater per faks eller mail til forbundets ansvarlige. Disse må videre mate tallene inn i sine egne systemer for statistikk og liknende. Dette gir mye arbeid for mange personer. Mulighetene for misforståelser og feil i tallgrunnlaget er absolutt til stede.

Norges Skøyteforbund er derfor interessert i å se på muligheten for å utvikle en database over resultater i skøyteløp.

1.2 Avgrensning

1.2.1 Tidsrom

Vi startet med prosjektoppgaven den 9. januar 2003 og oppgaven skal leveres inn den 19. mai 2003. I tillegg skal det lages en plakat som vil fungere som en ”reklame” for prosjektet, denne skal leveres innen 20.mai 2003.

1.2.2 Økonomi

Vi har skrevet en egen kontrakt med oppdragsgiver, denne omhandler blant annet økonomiske avgrensninger.

1.3 Oppgavebeskrivelse

Oppgaven er todelt;

- Første del går ut på å lage et Webgrensesnitt mot en database over aktive norske skøyteløpere.
- Andre del består av å utvikle et program som tar seg av påmelding og resultatregistrering.

Til sammen utgjør disse to delene et resultatbehandlingssystem for NSF.

1.3.1 Databasen

Databasen skal inneholde aktive norske skøyteløpere, det vil si løpere som har betalt lisens. Om disse løperne skal det lagres personalia og bestenoteringer. I tillegg skal det lagres stevneresultater. Siste års resultater linkes opp mot hver enkelt løper.

1.3.2 Webgrensesnittet

Hver klubb tildeles en ”del-administrator”. Han skal kunne ha mulighet til å melde på, stryke og evt. oppdatere personlige opplysninger om sin klubbs løpere. Del-administratoren til de klubbene som arrangerer stevner skal også ha mulighet til å laste ned påmeldingslister som brukes i java programmet, og laste opp den resultatfilen som java programmet genererer.

Administrator skal kunne registrere klubbskifte, legge inn terminlister, legge inn nye løpere og fjerne gamle løpere. Generere/endre brukernavn og passord er også en oppgave for administrator.

Det skal være mulighet for å søke opp løpere og resultater på Websiden.

1.3.3 Program

Programmet skal holde orden på resultatdata. Resultatene skal kunne skrives ut på ulike formater blant annet med hensyn på gjeldende regler for idrett i Norge (ref. barneidrett).

Innleggingen av resultater foregår enten manuelt eller ved direkte kommunikasjon med klokkesystemet HEGO-Timing-Systems i Vikingskipet på Hamar.

Programmet åpner for manuell innlegging av løpere og eventuelle strykninger. Innlegging av ikke-lisensierte løpere tildeles dummy nummer. Programmet skal også ta hånd om trekning; automatisk (random) eller manuell innlegging. I tillegg skal det ha ”Speaker funksjonalitet”, det vil si speaker skal kunne holdes oppdatert ved hjelp av et eget vindu med oversikt over aktuelt par, neste par, resultater på aktuell distanse og resultater sammenlagt.

1.3.4 Annet

Webgrensesnittet må ta hensyn til at NSF bruker en ASP-server.
Programmet skal kunne kjøres på en Windows 98 plattform eller nyere.

1.4 Formål

Målet for prosjektet er å utvikle et system som letter arbeidet rundt resultatregistrering av skøyteløp arrangert i Norge.

1.5 Målgruppen for prosjektrapporten

- Sensor
- Oppdragsgiver
- Fremtidig videreutvikler av prosjektet
- Andre med interesse innen samme fagområde

1.6 Vår bakgrunn og kompetanse

Gruppen består av tre medlemmer, men det er kun 2 av oss som tar dataingeniørstudiet på Høgskolen i Gjøvik, av disse går den ene programmeringslinja mens andremann går på driftslinja. Det tredje medlemmet av gruppa har tidligere tatt informatikk ved Høgskolen i Hedmark og tar for tiden data- og mulitmedieteknikk på HiG. Dette gjør at vi til sammen har en ganske bred faglig bakgrunn. I tillegg har ett av medlemmene på gruppa vært aktiv skøyteløper, noe som har vært veldig nyttig i arbeidet med å fastsette rammene for prosjektet.

Vi er alle relativt unge og har liten eller ingen erfaring fra tilsvarende prosjekter. Ett av gruppemedlemmene har gjennomført et hovedprosjekt tidligere, dette har også vært en nyttig erfaring å ha med seg i gjennomføringen av prosjektet.

1.7 Arbeidsformer i gruppa

Gruppa har hovedsaklig jobbet hver for seg, men vi har hatt regelmessige møter og har hatt daglig kontakt per telefon og internett. Dette har gjort at vi hele tiden har vært oppdaterte på hva de andre gruppemedlemmene gjør. Vi føler at samarbeidet har fungert bra selv om vi sitter å jobber mye hver for oss.

Den kontakten vi har hatt med oppdragsgiver har i hovedsak foregått per mail. I løpet av prosjektperioden har vi levert inn til sammen tre statusrapporter. I forbindelse med statusrapportene har det også blitt avholdt statusmøter med veileder. I tillegg har vi avtale om å møte veileder hver fjortende dag.

1.8 Ansvarsforhold

Gruppeleder: Nina R. Hansen
Hovedtalsmann: John Rasmus Moen
Versjonsansvarlig: Rune Sørensen

1.8.1 Øvrige roller

Oppdragsgiver: Norges Skøyteforbundet v/ Svein Inge Strugstad
Kontaktpersoner: Kjell Ingar Myren, Odd Aasegård og Bjørn Ove Indrøy
Veileder: Harald Liodden
Stevne arrangør og test person: Hans B. Moen, John Rasmus Moen

1.9 Terminologi

API	Application Programing Interface. Link til et eksternt bibliotek som gjør at programmet får tilgang til dens klasser og funksjoner.
ASP	Active Server Pages. Programmeringsspråk som benyttes i klient/server programmering. Webgrensesnitt opp mot en database.
Baud	Måleenhet
Bit	Det minste lagrings elementet i en datamaskin
Byte	8 bit
Paritets kontroll	Kontrollverdi representert med et bit
CSS	Cascading Style Sheets Beskriver regler for et stilsett som brukes i HTML dokumenter. Når du gjør forandringer i CSS dokumentet vil dette forplante seg ut over alle dokumentene som er linket til denne CSS fila.
DBMS	DataBase Management System. Software som kontrollerer organiseringen, lagringen, mottaket, sikkerheten og integriteten til dataene I databasen.
DSN	Data Source Name. Et datakildenavn som lagres i ODBC.
GUI	Graphical User Interface. Et grafisk brukergrensesnitt som inneholder flyttbare vinduer, ikoner og en musepeker
Hash	En algoritme som omgjøre en tekst med variabel størrelse til en tekst med fast størrelse (hash verdi)
Hex	Heksadesimalt, tallsystem med 16 som grunntall
HTML	Hyper Text Markup Language “Presentasjons språk” for Websider
IIS	Internet Information Server. Microsoft sin web server.
JVM	Java Viritual Machine. En virtuell maskin, det vil si en simulasjon som brukes til å lure et program til å tro at operativsystemet støtter alle programmets funksjoner.
MySQL	SQL – Structured Query Language MySQL er verdens mest populære database med fri kildekode (open source database)

ODBC	Open DataBase Connectivity. Et programmeringsgrensesnitt som gjør at programmerer kan få tilgang til data i databasesystemer som bruker SQL som standard for datatilgang.
Parser	Bryter en tekst opp i kjente elementer for å kunne foreta en tolkning
Prototyp	Utkast eller grunnform av noe som kan utvikles videre
Px	Piksel (Pixel – PIX [picture] EL [element]). Den minste adresserbare enheten på en skjerm.
RS-232	RecommendStandard number 232, COM port (seriell port)

2 Kravspesifikasjon

Siden vi ikke ble presentert for en kravspesifikasjon når vi startet på oppgaven, var vi nødt til å utforme denne på egen hånd. I den forbindelse har vi vurdert det som nødvendig å utføre en del analyser. Vi har lagt størst vekt på utforme en forandringsanalyse og lage en Use Case modell. Resultatene av disse analysene følger utover i kapittelet.

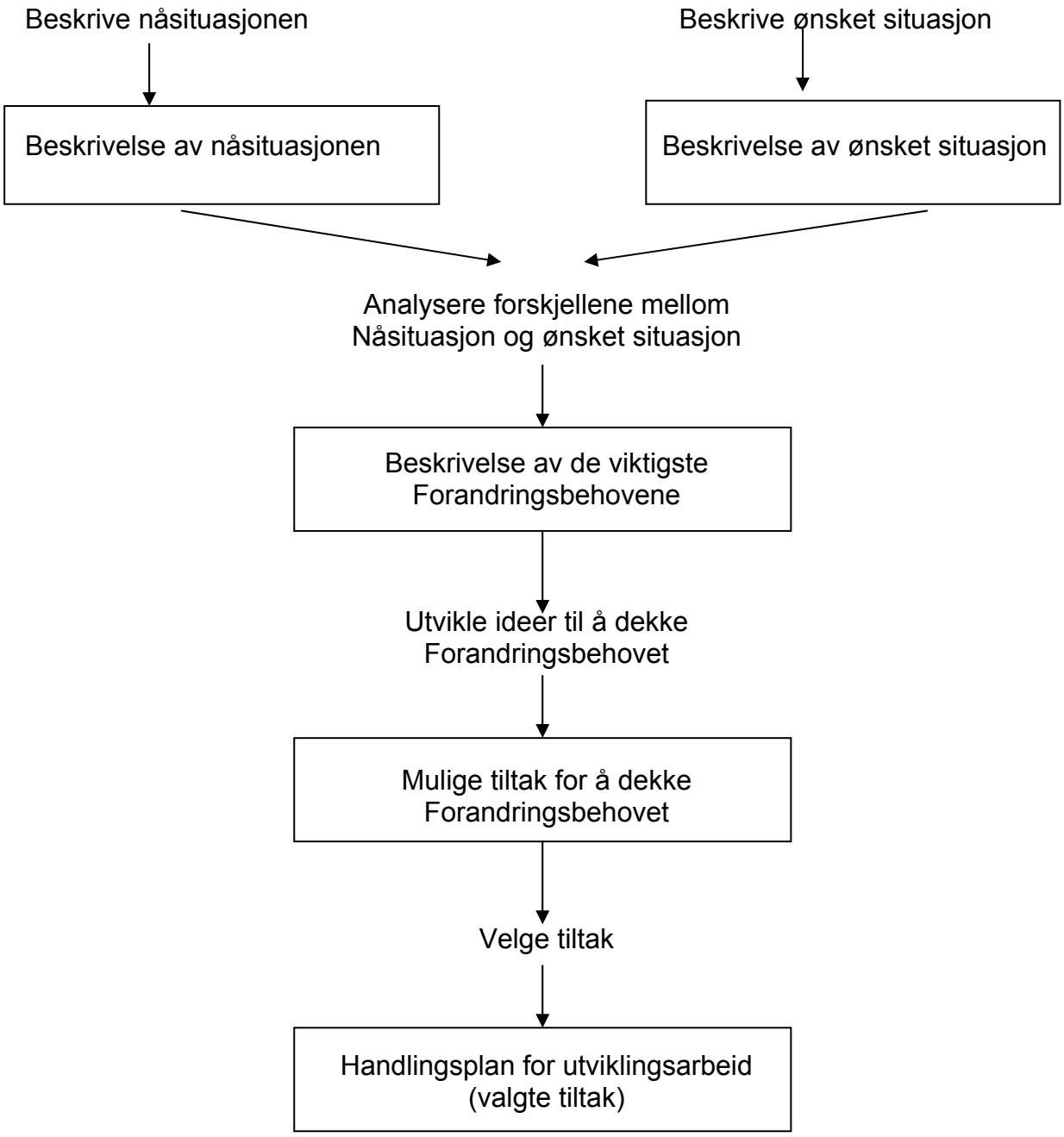
2.1 Forandringsanalyse

Hensikten med forandringsanalysen er å finne fram til hvilke forandringsbehov det er i virksomheten, og formulere en plan for det videre utviklingsarbeidet.

- Skille mellom symptomer og problemer
- Finne tiltak for å løse problemene

2.1.1 Y-modellen

- Generell problemløsningsmodell
- Vesentlig skille fra andre modeller: beskriver ikke virksomhetens mål først
- Utgangspunkt: beskrive nåsituasjonen slik den er
- Nåsituasjonens grunnlag for å beskrive en ønsket situasjon
- Utføres av egne medarbeidere
- Sammenstille beskrivelse av ønsket situasjon med nåsituasjonen
- Analysere forskjellene – finne forandringsbehov – prioritere disse
- Forslag (ideer og tanker) til løsninger (kreativ aktivitet) -> tiltak!
- Utvalg blant foreslåtte tiltak
- Handlingsplan for utviklingsarbeid – hensyn både til økonomi og personlige ressurser



Figur 2.1: Forandringsanalyse modell

I figur 2.1 representerer teksten (uten ramme) aktiviteter, mens ”rektanglene” viser hvilken informasjon som blir brukt av etterfølgende aktiviteter. Pilene illustrerer hvilken vei sammenhengen går.

2.1.2 Nåsituasjon

Registrering av resultater for skøyteløp gjøres i dag på mange måter. Det benyttes regneark og diverse mer eller mindre formålstjenlige programmer. Det finnes svært få programmer beregnet på skøyteløp. IsTid er et program utviklet til formålet, men dette er et DOS-basert program, og derfor ikke spesielt brukervennlig

Arrangører av skøyteløp sender resultater enten per faks eller mail til forbundets ansvarlige. Disse må videre mate tallene inn i sine egne systemer for statistikk. Dette gir mye arbeid for mange personer. Mulighetene for misforståelser og feil i tallgrunnlaget er absolutt til stede.

2.1.3 Ønsket situasjon

NSF er interessert i å utvikle et system som er med på å redusere antall dugnadstimer. I tillegg ønsker de et system som minsker risikoen for misforståelser og feil.

2.1.4 Forskjellen mellom nåsituasjon og ønsket situasjon

- Håndtering av påmeldinger og resultater for skøyteløp forenkles.
- Antall dugnadstimer for de involverte reduseres.
- Risikofaktoren med misforståelser og feil minimeres - det blir et mer pålitelig system.

2.1.5 Ulike ideer til å dekke forandrings behovet

Utvikle et databaseverktøy for registrering av resultater, og en modul for automatisk registrering av tider fra elektronisk tidtagerutstyr. For å redusere feil og misforståelser må enkelhet etterstrebes. Jo enklere et system er å bruke, jo mindre brukerfeil finner sted.

2.1.6 Velge tiltak

Oppgaven blir todelt; Første del går ut på å lage et Webgrensesnitt som er basert på en database over aktive norske skøyteløpere. Andre del består av å utvikle et program som tar seg av påmelding og resultatregistrering. Til sammen utgjør disse to delene et resultatbehandlingssystem for NSF.

Med tanke på å redusere brukerfeil vil fokus på en enkel og intuitiv GUI stå sentralt i løsningen.

2.2 Use Case

2.2.1 Beskrivelse

En Use Case modell beskriver de funksjonelle krav til systemet som er under utvikling. Modellen bruker grafiske symboler og tekst for å beskrive hvordan brukere i gitte roller vil bruke systemet. Den tekstlige beskrivelsen av en Use Case gis fra en brukers ståsted. De vil ikke beskrive systemets interne struktur eller mekanismer.
En Use Case modell kan bestå av følgende elementer:

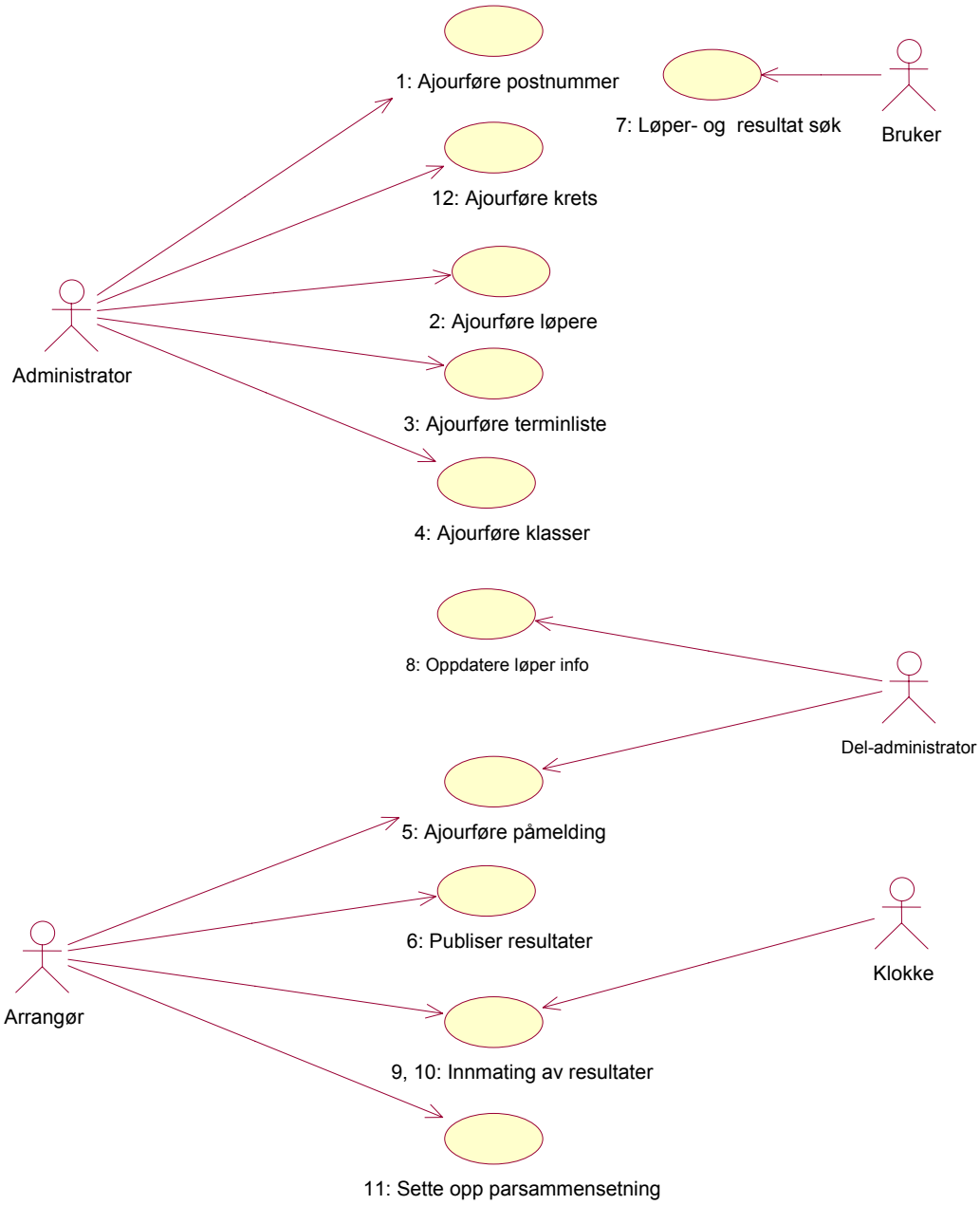
- Aktører (brukere av systemet)
- Use Cases (Beskrivelse av en gitt funksjon)
- Kommunikasjon mellom aktører og Use Cases
- Relasjoner mellom Use Cases
- Resultat etter gjennomført funksjon
- Situasjoner som påvirker resultat

2.2.2 Formål

Hovedmålet med Use Case modellen er å etablere grensesnittet til software systemet og beskrive fullt ut systemets funksjonelle muligheter sett fra brukeren. Use Case modellen skal :

- Lage en basis for kommunikasjon mellom sluttbruker og systemutvikler.
- Er den primære driveren for estimering av applikasjonsutvikling.
- Lage en basis for planlegging for utvikling av versjoner av applikasjonen.
- Lage en basis for identifisering av objekter, objektenes funksjonalitet, interaksjon og grensesnitt.
- Lage et første utkast for definisjon av brukergrensesnitt krav.
- Lage en basis for test- caser.
- Fungere som basis for akseptanse testing.
- Lage en basis for produksjon av bruker support materiale og dokumentasjon.

2.2.3 Modell



Figur 2.2: Use Case Modell

2.2.4 Use Cases

Use Case 1	Ajourføre postnummer
Mål	Til enhver tid skal oppdaterte postnummer være tilgjengelig I databasen
Aktør	Administrator
Beskrivelse	Enhver løper i databasen er tilordnet et postnummer med tilhørende adresse
Prebetingelse	Postnummeret må eksistere
Postbetingelse	Alle korrekte postnummer er å finne i databasen. Postnummerlisten er i samsvar med postens gyldige postnummer

Scenario	**Resultat**
Feil ved innleggingen av postnummer	Løper står oppført med feil adresse i databasen

Use Case 2	Ajourføre løpere
Mål	Databasen skal inneholde riktig informasjon om aktive norske skøyteløpere, det vil si løpere som har løst lisens
Aktør	Administrator
Beskrivelse	Administrator skal kunne redigere all personalia om alle løperne som befinner seg i DB. Han/hun skal også kunne slette løpere som ikke lengre er aktive, slik at man unngår unødvendige data i DB. Nye løpere skal kunne legges inn. Det skal også være mulig å aktivere / deaktivere en spesifikk løper med tanke på om lisens er løst inn eller ikke. Dette kan løses på serversiden ved at løpere i DB deaktiveres en bestemt dato etter hver sesong, for eksempel 1. mai. Løperne aktiveres manuelt av administrator etter hvert som de løser inn lisens. Dersom en norsk løper setter en personlig rekord utenlands eller norsk rekord / verdens rekord må denne oppdateringen foretas manuelt av administrator.
Prebetingelse	Løperen må løse lisens for å være registrert som en aktiv løper
Postbetingelse	Løperen har mulighet til å bli meldt på stevner gjennom ”Web-påmeldingen”

Scenario	**Resultat**
En løper har løst lisens, men er ikke blitt aktivert i den nye sesongen	Løperen får ikke mulighet til å melde seg på stevner

Administrator aktiverer en løper som ikke har løst lisens	En løper som ikke skulle hatt tilgang til å melde seg på stevner får muligheten til dette
Dersom en løper står oppført med feil fødselsdato, vil han bli registrert i feil klasse	Løperen får ikke tilgang til å gå løp i sin egen klasse og får heller ikke tilgang til å gå de distansene han skal

Use Case 3	Ajourføre terminliste	
Mål	Revidert terminliste med korrekt distanseoppsett skal ligge i DB	
Aktør	Administrator	
Beskrivelse	Administrator skal kunne legge inn terminliste for kommende sesong. Terminlisten inneholder navn på arrangement, sted, dato og mulige distansevalg for aktuelle årsklasser.	
Prebetingelse	Terminliste for aktuell sesong må eksistere	
Postbetingelse	Oppdatert terminliste befinner seg i DB	
Scenario		**Resultat**
Oppdatert versjon av terminliste ligger ikke tilgjengelig i DB		Påmeldinger til stevner vil ikke bli korrekte

Use Case 4	Ajourføre klasser	
Mål	Datogrensene for hver årsklasse skal være i samsvar med aktuell sesong	
Aktør	Administrator	
Beskrivelse	Datogrensene for klasseinndeling er dynamiske, det vil si de endres fra år til år. For hver nye sesong må datogrensene oppdateres slik at løperen står oppført i riktig årsklasse. Det gjøres manuelt av administrator.	
Prebetingelse	Løperens fødselsdato må være korrekt	
Postbetingelse	Alle løpere er tilknyttet riktig klasse	
Scenario		**Resultat**
Feil i innlegging av løpernes fødselsdato		Løperen blir oppført i feil klasse
Feil i opplysning om løpernes fødselsdato		Løperen blir oppført i feil klasse
Feil i start- og sluttdato grense for en klasse		Flere løpere vil bli tilordnet feil klasse, konkurransen blir ikke rettferdig og rekorder blir ikke gyldige

Use Case 5	Ajourføre påmelding
Mål	Å få tilgang til en komplett påmeldingsliste for aktuelt stevne
Aktør	Del-administrator, Arrangør
Beskrivelse	De to aktørene ovenfor skal utføre nesten de samme tingene – men de befinner seg på ”to ulike arenaer”. Del-administrator, som kun har tilgang på løpere fra egen klubb, skal kunne melde på en av sin klubbs løpere på ønskede distanser i et stevne. Så lenge ikke fristen for påmelding er gått ut skal del-administrator kunne gå inn og stryke en løper fra påmeldingslisten, eller endre distanseoppsettet for en løper. Alle disse operasjonene foregår via Webgrensesnittet. Arrangør sitter lokalt på en maskin med et program som blant annet håndterer påmeldingslisten og kan utfører følgende oppgaver; endring av distansevalg for en løper og strykninger. I motsetning til del-administratoren skal arrangør ha mulighet for å legge inn etteranmeldte løpere, rekrutter (som ikke løser lisens) og utenlandske løpere.
Prebetingelse	Terminliste må eksistere
Postbetingelse	Arrangør har en komplett påmeldingsliste for aktuelt stevne

Scenario	**Resultat**
Feil i fødselsdatofelt for løper(e)	Vil føre til at løpere står oppført i feil klasse, noe som automatisk kan føre til feil i distanseoppsett for aktuell løper
Feil i terminlisten	Kan føre til at løpere blir registrert på feil arrangement

Use Case 6	Publisere resultater
Mål	Arrangør skal kunne publisere resultatdata både via Web og i form av papiroppslag på selve stevnet
Aktør	Arrangør

Beskrivelse	Ved hjelp av program (java applikasjon) skal arrangør ha mulighet for å publisere diverse utskriftsformater: Resultater enkelt distanse Par for par resultater Sammenlagt resultater etter 2, 3 og 4 løp Hvilke løpere som startet i indre / ytre bane Resultatdataene lastes opp til DB og linkes opp mot korrekt løper på korrekt distanse og eventuelt oppdatere personlige rekorder til løpere. Programmet skal også generere en ska-fil for opplastning til Serveren (konkret stevne resultat).
Prebetingelse	Arrangør må ha en printer tilgjengelig for å kunne benytte utskriftfunksjonen, og internett tilgang for å få lastet opp resultatene til DB
Postbetingelse	Distansene er gjennomført og resultatene er registrert og publisert

Scenario	**Resultat**
Feil i opplastning av data til DB	Løpere vil da ligge med feil resultater i DB og evt. rekorder vil bli feil

Use Case 7	Løper- og resultat søk
Mål	Bruker skal kunne søke etter løperinformasjon og resultater
Aktør	Bruker
Beskrivelse	Bruker skal kunne kople seg opp mot databasen via Webgrensesnittet og søke opp relevant informasjon om aktive løpere og resultater fra forskjellige skøytestevner.
Prebetingelse	Bruker må ha internettilgang
Postbetingelse	Bruker får informasjonen han/hun søker etter

Scenario	**Resultat**
Bruker får ikke kontakt med database	Får ingen resultater på søket
Databasen er ikke oppdatert med korrekte data	Bruker får feil informasjon

Use Case 8	Oppdatere løper info
Mål	Databasen skal inneholde oppdatert og riktig informasjon om løperens personalia
Aktør	Del-administrator

Beskrivelse	Del-administrator skal ha mulighet til å legge inn/endre enkelte personalia om løpere i klubben som han/hun tilhører. Personalia som del-administrator skal kunne endre er adresse, postnummer, telefonnummer, e-post adresse og hjemmeside adresse. Hver enkelt løper kan bestemme selv om de valgfrie opplysningene skal være tilgjengelige.
Prebetingelse	Løper må være tilknyttet en adresse som har et gyldig postnummer og eventuelt ha telefon, e-post eller personlig hjemmeside
Postbetingelse	Løpere ligger i databasen med riktig personalia.

Scenario	**Resultat**
Feil i innlegging av personalia	Løperen blir liggende med feil informasjon i databasen
Feil i opplysninger om personen	Løperen blir liggende med feil informasjon i databasen

Use Case 9	Innmating av resultater
Mål	Registrere korrekte rundetider og sluttider for løpere
Aktør	Klokke
Beskrivelse	Java-applikasjonen skal kunne kommunisere med klokkesystemet HEGO 8000 og hente ut riktige tider til riktig løper automatisk
Prebetingelse	Java applikasjonen må ha riktig grensesnitt (RS-232) mot klokka for å kunne kommunisere med den
Postbetingelse	Riktig løper får tilknyttet riktig tid

Scenario	**Resultat**
Klokkesensorene blir brutt av noe/noen andre enn riktig løper	Løper får feil runde- /sluttid
Java applikasjonen blir koplet mot et annet klokkesystem enn HEGO 8000	Informasjon om tider vil ikke bli sendt til programmet

Use Case 10	Innmating av resultater
Mål	Registrere korrekte rundetider og sluttider for løpere
Aktør	Arrangør
Beskrivelse	Arrangør skal kunne registrere tidene til løperne manuelt ved hjelp av et ”speaker vindu” i java applikasjonen

Prebetingelse	Arrangøren må ha java applikasjonen tilgjengelig på en PC på arrangørstedet
Postbetingelse	Riktig løper får tilknyttet riktig tid

Scenario	**Resultat**
Arrangør taster inn feil tid på løper	Løper blir liggende med feil informasjon i databasen, eventuelt rekorder vil ikke bli korrekte

Use Case 11	Sette opp parsammensetning
Mål	Kunne trekke parsammensetning i henhold til påmeldingslista
Aktør	DelAdmin (Arrangør)
Beskrivelse	Arrangør skal kunne laste ned påmeldingsfilen til aktuelt løp, og foreta trekning enten manuelt eller automatisk. Den automatiske trekningen vil være begrenset til å foreta en ”random” trekning. Dersom det finnes odde antall løpere i en klasse, så skal en løper gå alene i indre bane, første par.
Prebetingelse	Påmeldingslista må være tilgjengelig for arrangøren
Postbetingelse	”Lovlig” parsammensetning til aktuelt stevne blir generert

Scenario	**Resultat**
Påmeldingslista er ikke tilgjengelig	Trekningen vil ikke kunne gjennomføres

Use Case 12	Ajourføre krets
Mål	Sørge for at opplysninger om kretser og deres klubber er oppdaterte
Aktør	Administrator
Beskrivelse	Hver enkelt løper er medlem i en klubb. Klubben er igjen tilordnet en krets
Prebetingelse	Kretsen og klubben må eksistere
Postbetingelse	Databasen er oppdatert med riktig informasjon om klubber og kretser

Scenario	**Resultat**
Opplysninger om en klubb/løpers krets er feil	Løper får ikke deltatt på krets stevner.
En krets finnes ikke i tabellen	Får ikke lagt inn en ny klubb fordi man ikke får tilordnet den til riktig krets

2.3 Krav til grensesnitt

Databasen og Websidene vil bli lagt ut på Norges Skøyteforbund (NSF) sin server. Java-applikasjonen skal legges inn på PC-en til hver enkelt arrangør. Siden applikasjonen utvikles i java vil den være plattformuavhengig. I tillegg må arrangøren ha tilgang til en PC med internett tilkobling for å kunne laste ned påmeldinger, og laste opp igjen resultatene.

2.4 Menneskelige krav

Vi legger opp til at det meste skal skje ute på Web og at ingen skal jobbe direkte på databasen (kun via et Webgrensesnitt). Derfor stilles det heller ikke veldig store krav til datakunnskap hos brukerne av systemet. Vi legger opp til å lage et enkel og brukervennlig brukergrensesnitt (GUI). For arrangørene stilles det litt større krav. Her skal det utvikles en java-applikasjon for å ta seg av trekning og gjennomføring av selve stevnet. Vi legger opp til at det også her skal være mest mulig brukervennlig, men arrangør må kunne laste ned deltagerlisten og laste opp resultatlisten til databasen.

2.5 Funksjonalitet

2.5.1 Sikkerhet

For å gjøre endringer i databasen kreves det at du logger deg inn. Her finnes det to forskjellige nivåer over hvilke tilganger/rettigheter du har. Den første er Administratoren. Han kan gjøre forandringer på alle felter i databasen. Administratoren bør være en utvalgt person fra Norges Skøyteforbund. Sikkerhetsnivå 2 er for del-administratoren. Dette er en representant for hver enkelt klubb, som skal ha mulighet til å redigere navn, adresse, e-post og hjemmesider for sin klubbs løpere, samt ha mulighet til å melde disse løperne på skøytestevner.

2.6 Pålitelighet

2.6.1 Datalagring

Alle resultater for inneværende sesong vil bli oppført på hver enkelt løper og kunne søkes opp på lik linje med denne løperens personalia. I tillegg genereres et html-dokument med resultater for aktuelt stevne som legges på serveren. Det vil alltid være mulig å søke på resultatene i et gitt stevne, men de vil kun bli linket opp til hver enkelt løper den sesongen stevnet finner sted. Etter endt sesong vil resultatene kun bli å finne på resultatfilen. Denne filen vil alltid være tilgjengelig, så fremt databasen er tilgjengelig.

En oversikt over hver løpers personlige rekorder vil være tilgjengelig. Disse rekordene vil hele tiden holdes oppdatert av systemet.

2.6.2 Transaksjonshåndtering

Det vil være en sjekk på at det kun er løpere som er satt til å være aktive som kan meldes på et stevne. Databasen må i tillegg kunne takle at flere del-administratorer et logget inn på samme tidspunkt, og at disse kan sende inn sine påmeldinger samtidig. Dette er i utgangspunktet ikke et problem siden hver del-administrator kun har tilgang til sin klubbs løpere, og ingen løpere kan være medlemmer i flere klubber samtidig. Det vil heller ikke være et problem med samtidighet når det gjelder innleggelse av resultater siden dette skjer på slutten av hvert stevne og ingen løpere kan delta på flere stevner samtidig.

2.7 Ytelse

Siden Webgrensesnittet primært skal brukes til søking, stiller dette store krav til databasen. Det er viktig at den har enklest mulig tilgang til data. Dette kan i enkelte tilfeller bety at det ikke er hensiktsmessig at databasen er fullstendig normalisert. Hvis databasen normaliseres kan dette resultere i at søket involverer mange tabeller, noe som igjen øker søketiden. Det kan derfor være en bedre løsning og dobbeltlagre enkelte av dataene med tanke på høyere ytelse.

Ytelsen til java-applikasjonen begrenses først og fremst av kapasiteten på den maskinen den kjøres på. Opp- og nedlastingshastigheten vil begrenses av hastigheten til internett forbindelsen den enkelte arrangør sitter med.

Minimumskravene til maskinene som SkateBase skal kjøre på er at de har Windows 98 og internettilgang. Men siden vi baserer oss på å bruke en java-applikasjon til selve arrangør-applikasjonen kan denne også kjøres på andre operativsystemer.

2.8 Begrensninger

2.8.1 Implementering

Vi skal som sagt tidligere bruke en java-applikasjon som lastes ned og legges på maskinen til hver enkelt arrangør. Webgrensesnittet vil bli utarbeidet i ASP, fordi serveren til NSF er en Windows 2000 IIS-server. Dette vil si at den kun er kompatibel med ASP. Databasen vår vil bli utviklet i MySQL. Dette er en plattformuavhengig freeware database.

2.8.2 kostnader

Utviklingen av denne prototypen vil ikke føre med seg at NSF må investere i nytt utstyr eller programvare. NSF har allerede en Webserver som vi kan benytte, i tillegg er all programvaren vi benytter helt gratis.

2.9 Antagelser

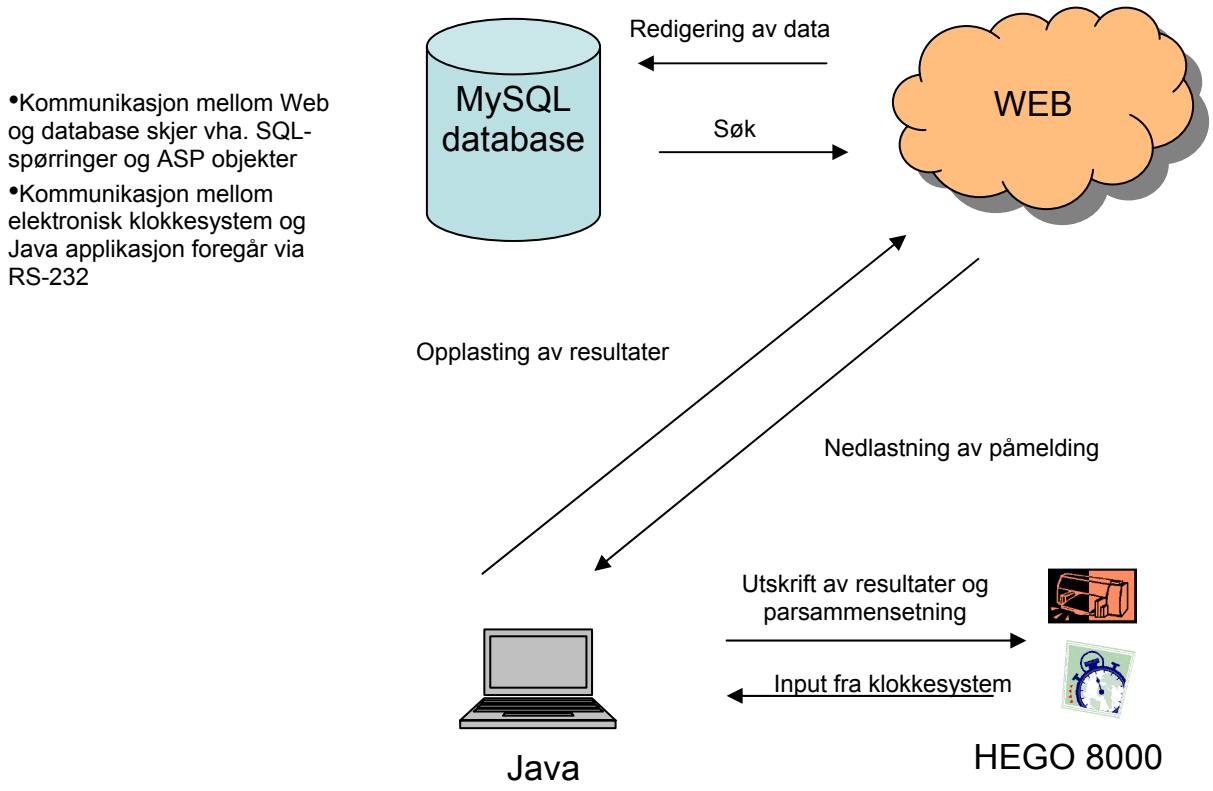
Vi antar at brukeren har en viss kjennskap til data og at han har nok tilgjengelig plass på maskinen til å kunne legge inn java-applikasjonen. I tillegg antar vi at alle brukerne av systemet har internett tilgang.

2.10 Aspekter omkring livssyklus

Innleggelse av data i databasen skal hovedsaklig gjøres av NSF og den eller de personene de velger ut til å være administrator. Ansvaret for vedlikehold og videreutvikling av prototypen ligger også hos NSF, men dette kan eventuelt gjøres ved hjelp/assistanse fra studentene. Alt vi skal gjøre skal dokumenteres nøye for at videreutvikling og vedlikehold skal bli enklest mulig.

2.11 Funksjonell struktur og tverrelasjoner

Figur 2.3 viser dataflyten mellom brukerne og systemet. Den viser også sammenhengen mellom de forskjellige modulene innad i systemet.



Figur 2.3: Dataflyt modell

2.12 Dataspesifikajons og dataordliste

2.12.1 Data Input

Administratoren legger inn de navn og personalia som måtte mangle eller som er feil. I tillegg har han/hun mulighet til å lage lister over internasjonale rekorder. Del-administratoren velger den løper han måte ønske og har der mulighet til å legge inn eventuell ny adresse, telefonnummer og så videre til denne løperen. I tillegg skal del-administrator ha mulighet til å merke av for hvilke distanser løperen skal delta på i de forskjellige arrangementene. Brukeren taster inn de ordene (navn, klubb, stevne og liknende) som han ønsker å søke på.

2.12.2 Data Output

Resultatene som legges inn i java-applikasjonen skal være mulig å skrive ut som en tekstfil. Når databasen oppdateres ved hjelp av Webgrensesnittet, skal vedkommende som legger inn opplysningene få tilbakemelding om at endringen er blitt lagret i systemet.

2.12.3 Tverrfunksjonelle datadefinisjoner

De forskjellige modulene i systemet jobber opp mot ulike data typer. Kommunikasjonen mellom klokken og java-applikasjonen foregår via serieporten, RS-232. Data blir sendt fram og tilbake mellom de forskjellige modulene. Arrangøren laster ned påmeldingslisten fra Webgrensesnittet. Denne kan deretter åpnes i java-applikasjonen og man fyller inn resultatene for stevnet. Påmeldingslisten (nå også med resultater) blir deretter lastet tilbake til databasen som en resultatfil.

2.13 Overordnede operasjonelle systemkrav

2.13.1 Normal operasjon

2.13.1.1 Modus og kontroll

Det er mulighet for å kjøre java-applikasjonen helt uavhengig av påmeldingsfila, for eksempel til mindre uoffisielle stevner, klubbmesterskap og liknende. Dataene lastes ikke tilbake til databasen uten at det sendes en kommando for dette.

2.13.1.2 Ytelse

Det stilles ingen spesielle krav fra oppdragsgiver til ytelse. Ytelsen vil avhenge av maskinen den kjøres på og hastigheten på internett forbindelsen

2.13.1.3 Sikkerhet

Det eneste kravet som stilles til sikkerhet er at det skal være tilgangskontroll. Det deles ut brukernavn og passord til de som skal ha tilgang til å redigere opplysningen i databasen.

3 Design

Brukervennlighet og at fargene skal være behaglige å se på er hovedkriteriene som har ligget til grunn for designet av Webgrensesnittet og java applikasjonen. Vi har valgt lys grå som grunn farge, men for å unngå at sidene skal bli kjedelige (tørre) har vi også lagt inn et bilde av tre skøyteløpere som bakgrunn. Vi har også tatt høyde for at det er en del brukere som fortsatt sitter med en skjermoppløsning på 800x600, slik at man slipper horisontal scrolling også på denne oppløsningen. Ved design av Webgrensesnittet har vi laget en egen CSS fil, se vedlegg[F]. CSS filen samler alt design på en plass, som igjen fører til enklere kontroll av designet. HTML egner seg dessuten dårlig til design.

Valg av farger er gjort i tråd med de fargene som finnes i NSF sin logo. Dette for å skape et mer helhetlig inntrykk.

Fonten Verdana, 11 px brukes som standard på alle Websider. Denne fonten er valgt fordi den er av de best leselige på skjerm da Verdana har sjenerøse ordmellomrom i tillegg til at den har jevne bokstavmellomrom. En annen faktor som gjør at Verdana gjør seg godt på skjerm er at denne skrifttypen har stor x høyde og åpne bokstaver. Verdana egner seg ikke fult så godt på utskrift, derfor har vi her valg å bruke fonten Arial 12 px.

3.1 Standarder

3.1.1 Tekstdokumenter

- Alle dokumenter som skal ligge på nett skrives i fonten Verdana og størrelsen 11 px
- Alle andre dokumenter skrives i Arial og størrelsen 12px
- All dokumentasjon vil foregå på norsk bokmål

3.1.2 Programmeringskode

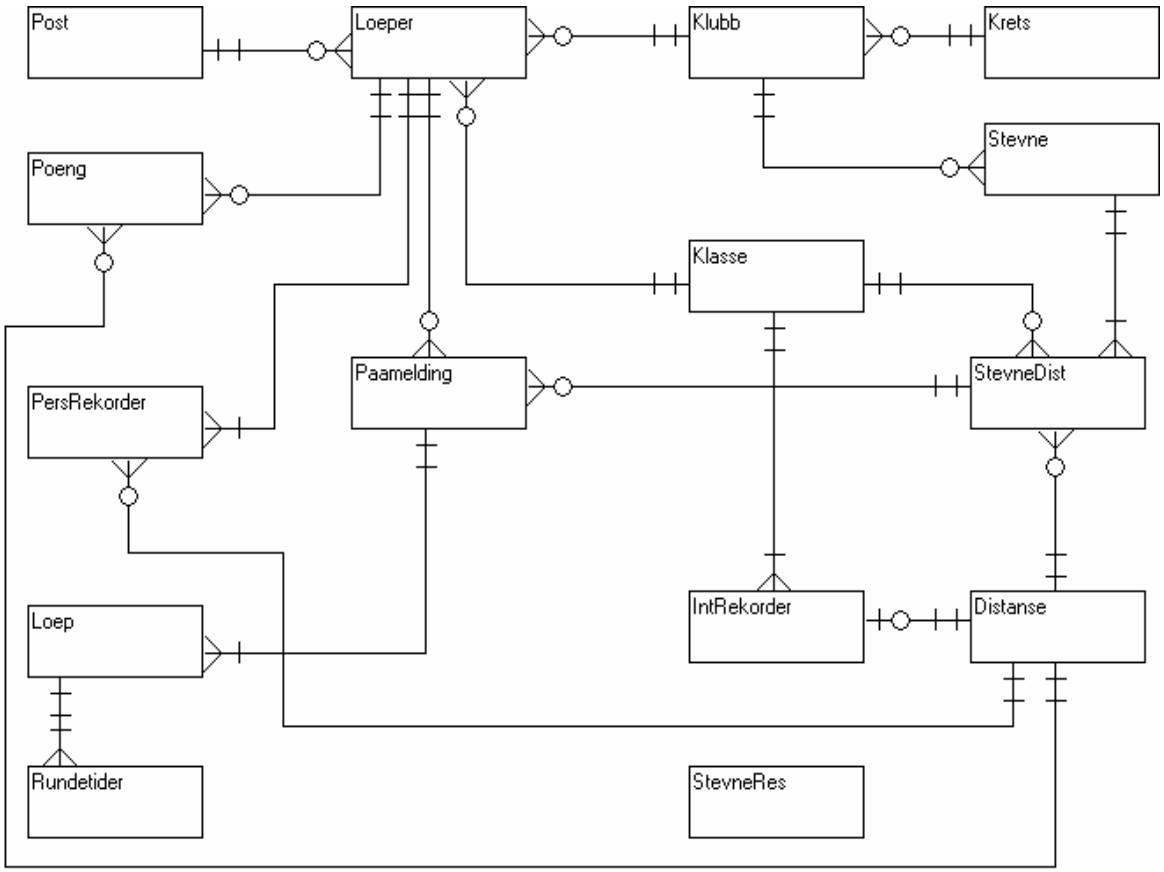
- Alle kommentarer foregår på norsk bokmål
- All kode skal kommenteres på funksjonsnivå
- Del 2 av prosjektet, se ovenfor, skal utvikles i java
- All kode utviklet i java skal dokumenteres vha java-dokumentasjonsstandarden
- All koding i forbindelse med Webgrensesnittet foregår i HTML, CSS og ASP

3.2 Databasen

Databasen er selve grunnpilaren i systemet vårt og vi har lagt ned mye arbeid i å designe denne best mulig. Vi har lagt vekt på at databasen skal være 3.grads normalisert, dette er for å unngå dobbeltlagring. Vi nevnte i kravspesifikasjonen

(kapittel 2.7) at vi vurderte å tillatte noe dobbeltlagring på grunn av at dette ville øke hastigheten. I ettertid har vi kommet fram til at dette ville føre til unødig store datamengder, noe som igjen vil gjøre at hastigheten på søket vil synke.

Nedenfor følger en enkel forklaring av databasen, figur 3.1. En fullstendig ER-modell med alle attributter ligger som et vedlegg[D] til denne rapporten, det samme gjør også SQL koden se vedlegg[E].



Figur 3.1: Database modell

Nedenfor følger en forklaring til alle tabellene i databasen:

3.2.1 Post

Dette er en oversikt over alle norske postnummer, med tilhørende poststed. Denne tabellen er kun linket til Loeper tabellen. Primærnøkkel: Postnummer.

3.2.2 Loeper

Denne tabellen inneholder alle aktive norske skøyteløpere. Fremmednøkler: Postnummer(fra Post), KlubbID(fra Klubb) og KlasseID(fra Klasse). Primærnøkkel: Lisensnummer, dette er et unikt nummer som følger løperen hele karrieren.

3.2.3 Klubb

Denne tabellen inneholder alle norske skøyteklubber. Fremmednøkkel: KretsID(fra Krets). Primærnøkkel: KlubbID.

3.2.4 Krets

Oversikt over alle norske skøytekretser. Primærnøkkel: KretsID.

3.2.5 Poeng

Denne tabellen holder oversikt over hver enkelt løpers norgescup og verdenscup poeng. Fremmednøkkel: Lisensnummer(fra Loeper), DistanseID(fra Distanse). Primærnøkkel: PoengID.

3.2.6 Klasse

Denne tabellen holder rede på alle aldersklassene. Primærnøkkel: KlasseID

3.2.7 Stevne

I denne tabellen legges årets terminliste inn, den holder rede på alle offisielle norske skøyteløp. Fremmednøkkel: KlubbID (fra Klubb), denne gir uttrykk for hvem som er arrangør på stevnet. Primærnøkkel: StevneID.

3.2.8 PersRekorder

I denne tabellen lagres alle løpernes personlige rekorder. Fremmednøkler: Lisensnummer (fra Loeper), DistanseID(fra Distanse). Primærnøkkel: PersRekordID.

3.2.9 Paamelding

Her registreres påmeldingene til alle sesonges løp, hver enkelt distanse på alle stevner får sin egen id. Fremmednøkler: Lisensnummer(fra Loep) og StevneDistID(fra SteveDist). Primærnøkkel: PaameldingsID.

3.2.10 StevneDist

Denne tabellen holder rede på hvilke distanser som kan gåes på hvert enkelt stevne. Fremmednøkler: KlasseID(fra Klasse), StevneID(fra Stevne) og DistanseID(fra Distanse). Primærnøkkel: StevneDistID.

3.2.11 Loep

Her registreres alle utøvernes løp i løpet av inneværende sesong. Fremmednøkkel: PaameldingsID(fra Paamelding). Primærnøkkel: LoepsID

3.2.12 IntRekorder

I denne tabellen ligger alle de internasjonale skøyterekordene, pluss nasjonale rekorder. Fremmednøkler: DistanseID(fra Distanse) og KlasseID(fra Klasse). Primærnøkkel: IntRekordID
Distanse Denne tabellen inneholder alle gyldige skøytedistanser. Primærnøkkel: DistanseID.

3.2.13 RundeTider

I denne tabellen ligger alle rundetidene til alle løpene i en sesong lagret. Det er i tillegg et Ja/Nei valg på om dette er løpets slutt tid eller ikke. På denne måten får vi med at løp kan registreres med alle rundetider, eller kun slutt tiden etter eget ønske. Fremmednøkkel: LoepsID(fra Loep). Primærnøkkel: RundeTidID

3.2.14 StevneRes

Dette er en tabell som ligger helt uten kobling til noen av de andre tabellene. Dette er en tabell som holder oversikt over alle Stevne resultat filer. Disse filene blir lagret til evig tid og ligger på serveren som en ska-fil. Filnavnet ligger i StevneRes tabellen. Dette er det eneste attributtet i denne tabellen.

3.3 Webgrensesnittet

3.3.1 Meny

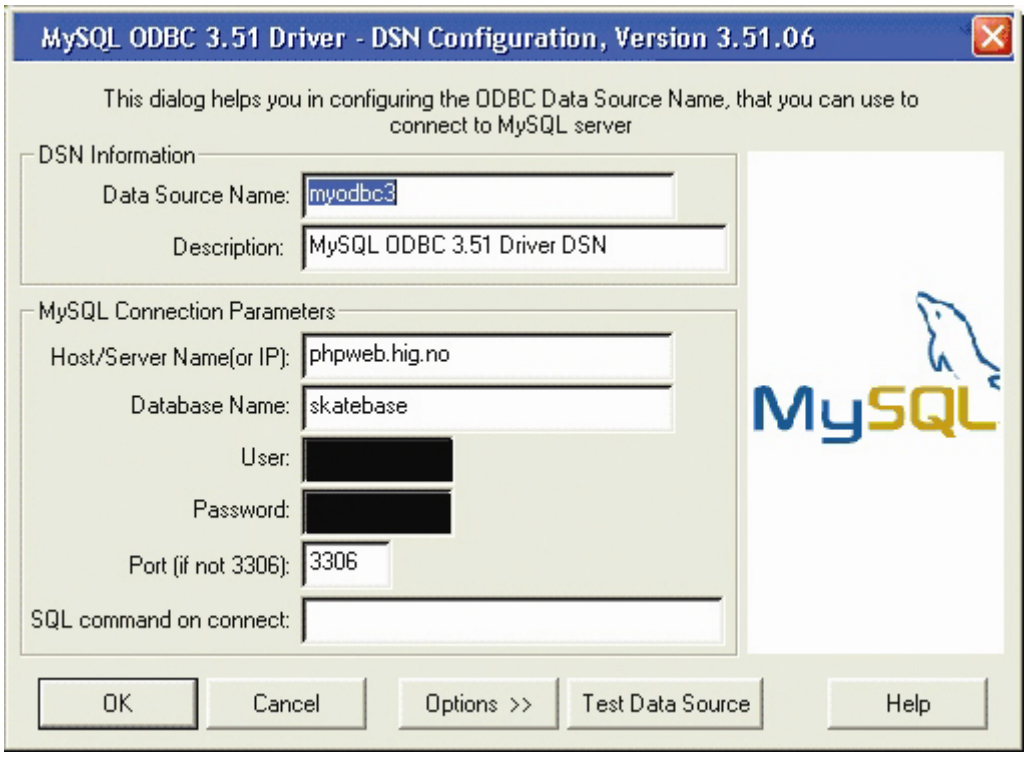
Utgangspunktet for vårt valg av menydesign har vært NSF's nåværende standard. http://www.skoyteforbundet.no. Menystrukturen er vist i figur 3.2:

Figur 3.2: Meny Webgrensesnittet

3.3.2 Oppkopling mot databasen

I oppkoplingen mot databasen valgte vi å bruke ODBC. Målet til ODBC er å gjøre det mulig slik at man kan fra hvilken som helst applikasjon aksisere data, uavhengig av hvilket DBMS som behandler dataen. Dette blir løst ved at ODBC legger databasedriveren i et lag mellom applikasjonen og DBMS. Dette laget oversetter applikasjonsdata og spørringer til kommandoer som DBMS skjønner.

Vi lagde en System DSN hvor brukernavn og passord til databasen ble lagret og kunne da opprette en forbindelse til databasen gjennom ASP sidene:



Figur 3.3: System DNS

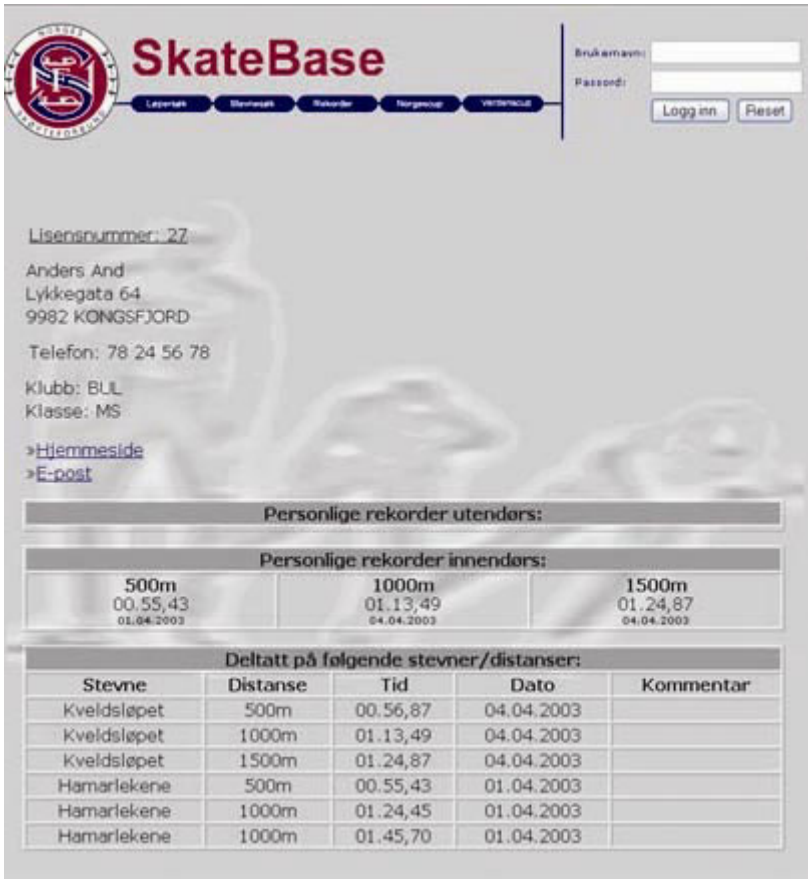
```
Set objConn = Server.CreateObject("ADODB.Connection") ‘Lager et forbindelses objekt
objConn.Open "DSN=myodbc3"   'Åpner databaseforbindelsen vha. DSN
```

Hvis brukernavn og passord endres på databasen, trenger man ikke å endre noe i ASP koden, men kun i DSN konfigurasjonen.

3.3.3 Brukerfunksjonalitet

3.3.3.1 Løpersøk

En vanlig bruker kan søke etter en spesiell løper. Du kan søke på en kombinasjon av fornavn, etternavn og klubb. Fornavn og etternavn er text-bokser hvor brukeren kan skrive inn tekststrenger, mens klubbene ligger i en dropdown-liste. Hvis søket ga et eller flere treff, vil løperen(e) bli listet opp som hyperlinker. Når brukeren trykker på linken, vil informasjon om løperen bli vist på skjerm.



Figur 3.4: Løpersøk

3.3.3.2 Stevnesøk

Brukeren kan søke etter stevneresultater på stevner som er arrangert i Norge. Stevnene som er registrert er presentert i en dropdown-liste hvor brukeren kan velge hvilket stevne han/hun vil se resultatene fra. Når et stevne er valgt, vil et nytt vindu åpnes og resultatlista vil vises. Resultatlisten er en tekstfil som ligger på serveren hvor den blir parset og det genereres HTML ut av filen.

3.3.3.3 Norgescup

Brukeren kan få opp en oversikt over stillingen så langt i Norgescupen. Klassene menn- / kvinner senior og gutter- / kvinner junior A er det mulig å få sammenlagt listen av. Når brukeren har valgt Norgescup må brukeren velge hvilken klasse

han/hun vil se sammenlagt listen fra. Klassene blir presentert for brukeren ved hjelp av radioknapper.

3.3.3.4 Rekorder

Verdens-, olympiske-, norges- og landsrekorder har brukeren mulighet for å se. Brukeren må også her velge en radioknapp over hvilken type rekord han/hun vil se. Alle klassene for en type rekord vil vises på skjerm. Denne siden blir ikke oppdatert automatisk, så administrator må vedlikeholde de internasjonale/nasjonale rekordene.

3.3.4 Innlogging / utlogging

Webgrensesnittet er delt inn i tre hoveddeler; bruker, DelAdmin og Admin. Brukeren må logge seg inn med brukernavn og passord for å få tilgang til DelAdmin- og Admin sidene. Når DelAdmin eller Admin logger seg inn, sjekkes brukernavnet og passordet opp mot databasen om de er gyldig. Passordet som sendes fra grensesnittet til databasen er krypter, men PASSWORD(string) funksjonen som MySQL bruker til kryptering er ikke av de sikreste. Den lager kun en 8 byte (16 hex karakterer) hash. Når det krypterte passordet ankommer databasen er det autentiseringssystemet som er innebygd i MySQL serveren som verifiserer passordet.

Hvis brukernavn og passord er lovlig, vil brukeren få tildelt sesjonsvariable.

```
session("klubb") = objRS("KlubbID")
session("nivaa") = objRS("Sikkerhetsnivaa")
```

Sikkerhetsnivået angir om det er DelAdmin eller Admin som er innlogget. Hvis det er en DelAdmin, så vil brukeren bli sendt til startsiden for DelAdmin, mens hvis det er Admin vil han/hun bli sendt til Admin siden.

Sesjonsvariablene vil være aktive så lenge brukeren er aktiv eller til brukeren logger seg ut. Hvis brukeren er inaktiv i tjue minutter (default verdi), vil sesjonen lukkes automatisk.

Når brukeren logger seg ut, stenges sessionen.

```
Session.Abandon
```

3.3.5 DelAdmin

3.3.5.1 Påmelding

Når DelAdmin er innlogget vil han/hun ha mulighet for å melde på løpere som er tilknyttet samme klubb som DelAdmin. Stevner som er lagt inn i terminlista og registrert i databasen kan løpere melde seg på. Stevnene er listet opp i en dropdown-liste inntil påmeldingsfristen går ut. Når en løper og et stevne velges fra dropdown boksene vises distanser som løperen kan meldes på ved hjelp av checkboxer. DelAdmin merker av hvilke distanser løperen skal gå og sender påmeldingen til databasen for registrering. Vi har tatt høyde for at alle påmeldinger foregår

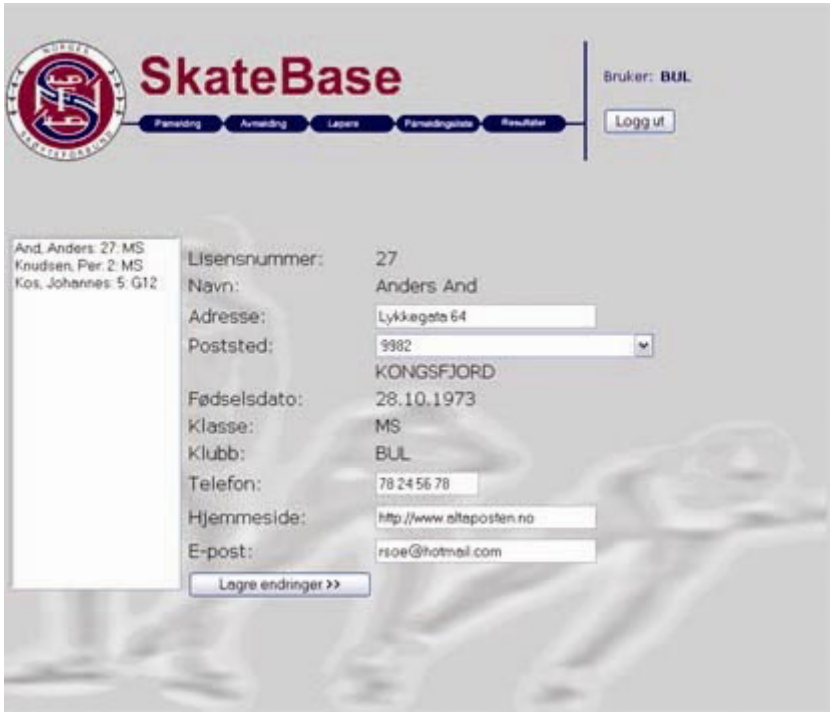
elektronisk via Webgrensenittet. Hvis en påmelding skjer per telefon, vil ikke løperen få tildelt en påmeldings ID som er nødvendig for at systemet skal fungere.

3.3.5.2 Avmelding

DelAdmin har mulighet til å melde av løpere som er meldt på stevner. Løpere som er tilknyttet samme klubb som DelAdmin listes opp i en dropdown boks, og DelAdmin velger aktuell løper. Alle stevner løperen er tilknyttet kommer opp på skjerm representert ved hjelp av checkboxer. Stevne(r) og distanse(r) som løperen skal melde avbud til merkes av og avmeldingen sendes til databasen. Løpere kan ikke melde seg av distanser etter at påmeldingsfristen har gått ut.

3.3.5.3 Endre personalia

En vanlig bruker kan søke etter en løper og få opp informasjon om løperen. Informasjon som DelAdmin kan endre er adressen, telefon, hjemmeside og e-post adresse. Hvis en løper får nytt postnummer, så velger DelAdmin det nye postnummeret fra en dropdown boks. Resten av de redigerbare feltene er tekstfelter.



Figur 3.5: Endre personalia

3.3.5.4 Laste ned påmeldingsliste

Hvis klubben til DelAdmin arrangerer et eller flere stevner har han/hun mulighet til å laste ned påmeldingslisten til aktuelt stevne. Stevnene som klubben arrangerer blir vist på skjerm som linker.

IUSR_*Machine* kontoen har ikke default rettigheter til å opprette mapper på serveren, men kun filer. Det vil si mapper og filer blir behandlet forskjellig, så administrator må lage en mappe manuelt på serveren som heter ”paamelding”.

Når en DelAdmin trykker på en link, utføres en del SQL spørringer opp mot databasen for å finne alle som er meldt på aktuelt stevne. Resultatene av spørringene blir skrevet til en ny fil som blir mellomlagret i mappen ”paamelding” se vedlegg[H].

```
'Finner path'en til mappen "paamelding"
Set strPath = fso.GetFolder( Server.MapPath("\paamelding\") )
path = strPath.Path
```

Når filen er opprettet, blir den lastet ned til bruker. Brukeren kan da velge om han/hun vil lagre filen lokalt på egen maskin eller på et flyttbart lagringsmedium.

```
function DownloadFile(sti)

  strAbsFile = sti & "\" & stevneNavn & ".ska"

   Set objFSO = Server.CreateObject("Scripting.FileSystemObject")

  if objFSO.FileExists(strAbsFile) then
     Set objFile = objFSO.GetFile(strAbsFile)
     Response.Clear
     Response.AddHeader "Content-Disposition", "attachment; filename=" & objFile.Name
     Response.AddHeader "Content-Length", objFile.Size
     Response.ContentType = "application/octet-stream"
     Set objStream = Server.CreateObject("ADODB.Stream")
     objStream.Open
     objStream.Type = 1
     Response.CharSet = "UTF-8"
     objStream.LoadFromFile(strAbsFile)
     Response.BinaryWrite(objStream.Read)
     objStream.Close
      Set objStream = nothing
      Set objFile = nothing
  else
     .
     .
     .
end function
```

Når filen er lagret hos brukeren, slettes den fra Webserveren. Filen slettes for å unngå unødig lagring på serveren.

Filen må ikke redigeres av brukeren fordi java-applikasjonen forventer et spesielt format og vil ikke kunne lese den hvis den blir endret.

3.3.5.5 Laste opp resultater

Programmet bruker et klassebibliotek for å laste opp filer til serveren. Biblioteket heter upload.asp [13].

Når et stevne er fullført og java-applikasjonen har lagd filen *stevnenavn_år.ska,* er resultatene fra stevnet klare til å registreres i databasen. Webgrensesnittet ber brukeren finne filen fra stevnet slik at filen kan leses og prosesseres av serveren. Filen er ”uleselig” se vedlegg[H] og inneholder ingen navn på løpere, så en ny

resultatfil lages. Det må opprettes en mappe manuelt på serveren som heter ”registrering” som filen blir lagret i.

Det første programmet gjør er å lese to linjer fra .ska filen. Den første linjen inneholder diverse informasjon om løperen og stevnet, mens den andre linjen inneholder passeringstidene og sluttiden for en distanse. Linjene blir prosessert og lagt i to arrayer.

```
'Leser "info-linjen" fra filen
strInfo = CStr(objFile.ReadLine)
'Splitter strengen og legger resultatene i en array. Denne er alltid 6 lang (0-5).
infoArr = split(strInfo, ";")

'leser "tid-linjen" fra filen
strTid = CStr(objFile.ReadLine)
'Splitter strengen og legger rundetidene i en array.
tidArr = split(strTid, ";")
'Teller hvor lang arrayen er. I siste "skuff" ligger slutt-tiden.
antRundeTid = ubound(tidArr)
```

Det blir nå utført SQL-spørringer opp mot databasen som registrerer passeringstidene og sluttiden. Hvis løperen satt personlig rekord, blir denne oppdatert. Vi har ikke tatt hensyn til tangering av personlige rekorder, så hvis løperen tangerer vil den gamle rekorden være gjeldene.

Resultatfilene blir ikke lagret i databasen, men på Webserveren. I databasen ligger kun filnavnene til de forskjellige stevnene som er lastet opp. For å unngå duplikater av filnavn, har vi tatt med året stevnet ble gjennomført med i stevnenavnet.

Når dataene er registrert i databasen, sendes filnavnet (uten filendelse) til databasen. Denne blir brukt når en vanlig bruker vil se resultatlisten fra stevnet.

Administrator må også her lage en mappe manuelt (se; *last ned påmeldingsliste*). Denne mappen må hete ”stevner”. Det er i denne mappen resultatfilene blir lagt når DelAdmin har lastet opp resultatene.

Serveren leser de to første linjene fra filen som DelAdmin lastet opp (*stevnenavn_år.ska*), splitter linjene og legger delene inn i to arrayer. Lisensnummeret til løperen ligger alltid i den første arrayen i første skuff. Resultatfilene skal alltid være tilgjengelig, selv om løper legger opp. Filen inneholder ikke navnet på løperen, så vi utfører en SQL-spørring for å få tak i personlig informasjon om løperen. Hvis en løper legger opp, skal han/hun slettes fra databasen og lisensnummeret er ledig for andre. Vi lager derfor en ny fil hver gang et stevne lastes opp. Det opprettes nå en ny fil som med navnet *stevnenavn_år.ska.* Informasjonen fra filen det leses fra og personlig informasjon om løperen (navn) skrives til den nye filen. Denne filen inneholder ikke passeringstider, kun sluttiden.

```
do while not objFile.AtEndOfLine
  strInfo = CStr(objFile.ReadLine)
  infoArr = split(strInfo, ";")

  strTid = CStr(objFile.ReadLine)
  tidArr = split(strTid, ";")
```

```
antRundeTid = ubound(tidArr)

distanse = infoArr(2)
lisensnummer = infoArr(0)
kommentar = infoArr(3)
slutt_tid = tidArr(antRundeTid)

 'Finner opplysninger om løperen
 strSQL = "SELECT Fornavn, Etternavn, KlasseID, KlubbID FROM Loeper "
 strSQL = strSQL&"WHERE Lisensnummer = '"&lisensnummer&"'"
 Set ObjRS = ObjConn.Execute(strSQL)

 'Skriver til filen
 filen.WriteLine( distanse & ";" & ObjRS("KlasseID") & ";" & ObjRS("Fornavn") & " " &
ObjRS("Etternavn") & ";" & ObjRS("KlubbID") & ";" & slutt_tid & ";" & kommentar )

loop
```

Når den nye resultatfilen er opprettet og skrevet til, slettes filen som DelAdmin lastet opp fra mappen ”registrering”, fordi denne filen blir ikke brukt mer. Dette for å unngå unødig lagring.

3.3.6 Admin

Når du logger deg inn som Admin har du tilgang til å redigere alle dataene som ligger i databasen. I utgangspunktet skal det kun være en Admin bruker, men det er åpning for at man kan legge til flere brukere med sikkerhetsnivå Admin.

3.3.6.1 Distanse

Her har administratoren mulighet for å redigere hvilke lovlige distanser som finnes. Når en distanse registreres gjøres dette ved hjelp av en DistanseID og et attributt som angir det antall runder denne distansen tilsvarer.

3.3.6.2 Rekorder

Her har administratoren to valg. Han kan enten redigere Internasjonale rekorder; det vil si landsrekorder, norgesrekorder, verdensrekorder og olympiske rekorder, eller redigere løpernes personlige rekorder.

3.3.6.3 Klasse

Her kan administratoren velge å redigere de forskjellige aldersklassene. Dette er en jobb han må gjøre hvert år før sesong start. StartGrense og SluttGrense for aldersklassene vil jo variere fra år til år.

3.3.6.4 Klubb

Hvis Admin velger Klubb i menyen har han ikke bare mulighet ti å legge til og fjerne skøyteklubber. Han har også mulighet til å legge inn passord og sikkerhetsnivå til de forskjellige klubbene. Til innlogging bruker hver klubb sin klubbID som brukernavn og passordet utdelt av Admin. Når han setter sikkerhetsnivået har han to valg, DelAdmin

og Admin hvor DelAdmin er default. Hva en DelAdmin har tilgang til å gjøre beskrives i kapitelet om DelAdmin.

3.3.6.5 Krets

Her har administrator mulighet til å legge til nye norske skøytekretser. Her ligger det også med et nasjons attributt, slik at det her er mulig å utvide tabellen til også å fungere på utenlandske kretser.

3.3.6.6 Løp

Hvis administratoren velger dette valget i menyen har han mulighet til å redigere på den informasjonen som har blitt sendt til databasen om et løp. Dette er en sikkerhet som må være tilstede siden det kan oppstå feil i fila som sendes til databasen. Det må derfor være mulig å rette opp disse feilene. Han har også mulighet til å registrere nye løp, fordi de som er etteranmeldt til et stevne ikke blir tatt med i resultatfila som sendes tilbake til databasen. Denne tabellen må tømmes etter endt sesong.

3.3.6.7 Løper

Her kan administratoren endre en løpers personalia samt administrere klubbskifter. DelAdmin har også mulighet til å endre på enkelte ting i denne tabellen, men dette er bedre forklart i kapitlet om DelAdmin. En løper settes til aktiv når han har løst lisens.

3.3.6.8 Poeng

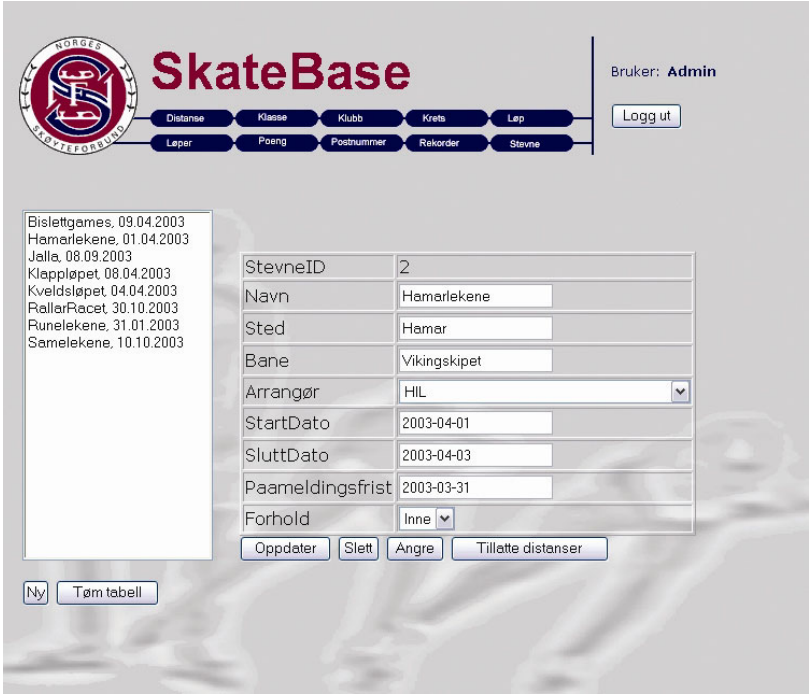
Her har administratoren muligheten til å legge inn Norgescup og Verdenscup poeng. Norgescup poengene blir ikke automatisk generert i java programmet, fordi reglene for tildeling av Norgescup poeng endres hyppig. Det er derfor enklere om Admin legger inn disse manuelt etter hvert Norgescup løp. Siden dette er en løsning som kun tar hensyn til norske løp, må Admin legge inn Verdenscup poengene som skal registreres manuelt. Poengene må slettes manuelt etter endt sesong.

3.3.6.9 Postnummer

Her har Admin mulighet til å legge inn nye postnummer. Dette må gjøres hver gang posten oppretter eller fjerner norske postnummer.

3.3.6.10 Stevne

Her legger Admin inn årets terminliste. I tillegg så legger han også inn hvilke distanser de forskjellige klassene skal ha mulighet til å gå på hvert enkelt stevne. Denne tabellen skal tømmes etter endt sesong.



Figur 3.6: Redigering av stevne

3.4 Java program

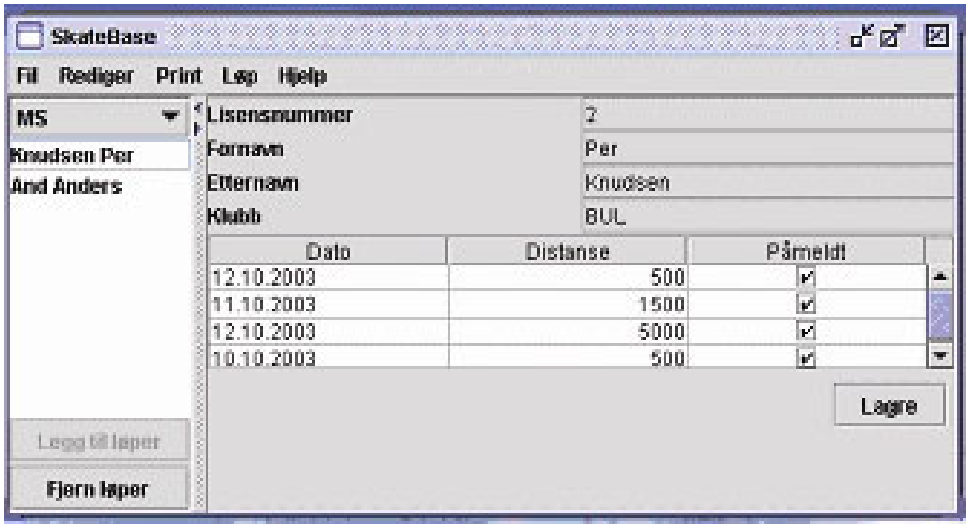
Java-applikasjonen, heretter kalt programmet, bygger fullt og helt på påmeldingsfilen generert av Webgrensesnittet. Vedlegg [F] viser påmeldingsfilens format.

3.4.1 Funksjonalitet

Hovedfunksjonaliteten i programmet er behandling av påmeldinger, oppsett av parsammensetning og kjøring av løp. Menyvalgene som er tilgjengelige i programmet er som følger:

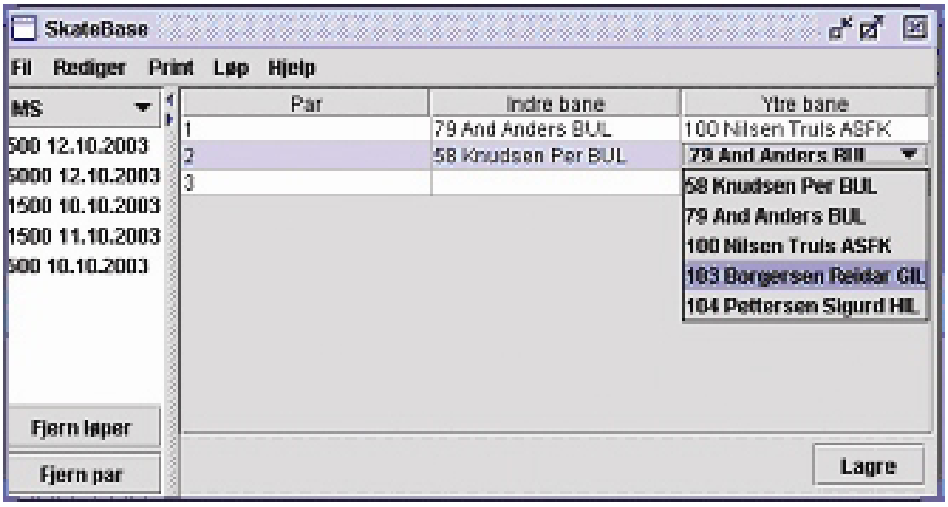
- Fil
  - Åpne påmelding: Ved hjelp av en open dialog velger bruker hvilken påmeldingsfil (stevnenavn.ska) han vil laste inn i programmet
  - Lagre påmelding: Overskriver eksisterende påmeldingsfil med eventuelle endringer
  - Lagre resultater: Skriver stevneresultater til fil. Formatet for resultatfilen er å finne i vedlegg [F]
  - Exit: Avslutter programmet

- Rediger
  - Påmelding: Gir mulighet for redigering av distanseoppsett for en bestemt løper (registrerer strykninger)



Figur 3.7: Redigering av påmelding

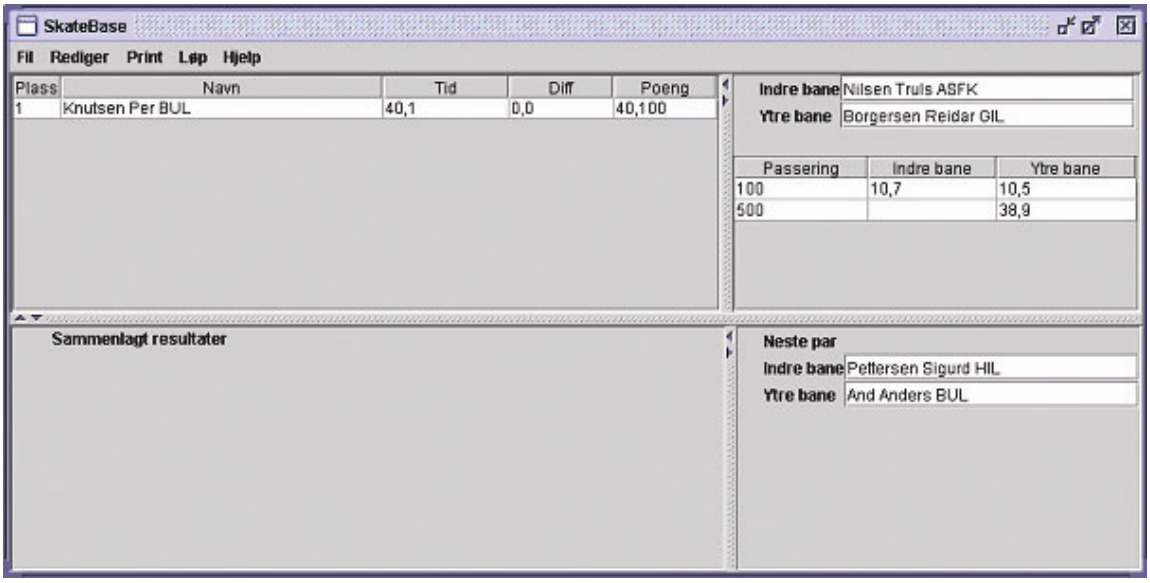
  - Parsammensetning
    - Manuell trekning: Bruker velger løper ut fra dropdown liste gitt i tabell
    - Automatisk trekning: Trekningen gjøres av programmet



Figur 3.8: Manuell parsammensetning

  - Resultater: Gir mulighet for endring av sluttider dersom feil under kjøring av løpet
- Print
  - Parsammensetning: Skriver ut til printer parsammensetning for en bestemt klasse på en spesifisert distanse
    - Velg klasse
      - Velg distanse
  - Resultater: Skriver ut til printer distanseresultater for en bestemt klasse på en spesifisert distanse

        - Velg klasse
            - Velg distanse
- Løp
    - Elektronisk: passeringstider og slutt-tider registreres automatisk fra klokkesystem
        - Velg klasse
            - Velg distanse
    - Manuelt: bruker mater inn passeringstider og slutt-tider for løper(e)
        - Velg klasse
            - Velg distanse



Figur 3.9: Speakervindu

- Hjelp
    - Om: informasjon om utviklerne bak løsningen

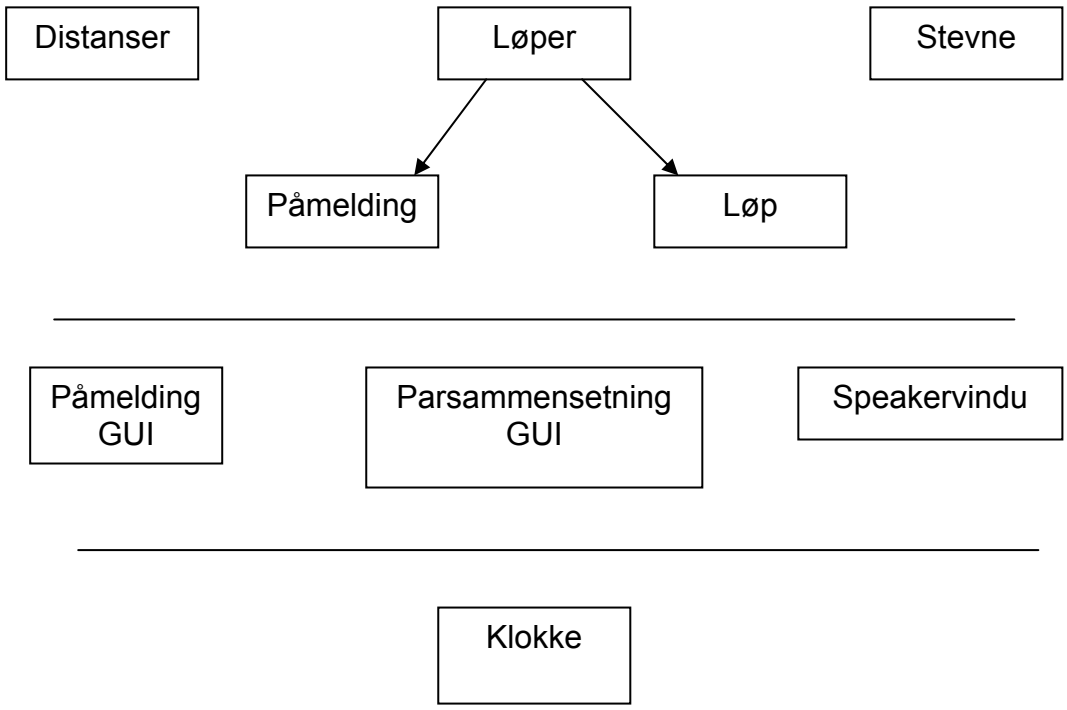
Programmet håndterer kun terminlistede stevner, det vil si stevner oppført i databasen. I påmeldingsfilen ligger det flere påmeldings ID'er for hver påmeldte løper generert av databasen. Påmeldings ID'en forteller programmet at løperen er meldt på til aktuelt stevne på en bestemt distanse som går en bestemt dato. Dette belyser en av begrensningene i programmet. Det er ikke åpning for etteranmeldinger. Løpere som kommer i etterkant og sier at de vil gå en eller flere distanser på et aktuelt stevne, må derfor stille "utenfor konkurranse". Disse påmeldingene vil ikke få en unik påmeldings ID som samsvarer med databasen. De vil derfor ikke inngå i resultatfilen fra programmet som er ment for opplastning til databasen. Løperne som går "utenfor konkurranse" vil kun figurere på print-resultatlistene som genereres av programmet. De vil ikke kunne søkes opp i Webgrensesnittet dersom ikke administrator legger løperne manuelt inn i databasen. Løperne vil ikke ligge i stevneresultatfilen. Utenlandske skøyteløpere er en annen gruppe løpere som må gå "utenfor konkurranse", da disse ikke har noen tilhørighet (lisensnummer) i databasen.

Redigering av påmelding åpner for at en løper kan melde seg av en eller flere distanser, og eventuelt hele stevnet. Dette kan være aktuelt ved sykdom, skade eller andre uforutsette hendelser.

Om begrensningen nevnt ovenfor er en svakhet eller stryke ved systemet kan diskuteres. Slik vi ser det er det en styrke. En styrke i den forstand at man sparer det ekstra arbeidet det faktisk medfører å ta hånd om etteranmeldinger. Etteranmeldingene kommer som regel veldig tett opp til arrangementstart, og da har arrangør mer enn nok med andre oppgaver. Dessuten er en påmeldingsfrist en endelig frist, og overholder man ikke denne fristen så må man stille utenfor konkurranse.

3.4.2 Oppbygning

Programmeringsspråket java bygger på bruk av klasser. Java har for eksempel egne klasser som tar seg av GUI og layout, andre klasser tar seg av lagringsstrukturerer og så videre. For vår del ble det naturlig å ta utgangspunkt i databasen og dens tabeller når programmets grunnmur skulle utarbeides.



Figur 3.10: Klasseoversikt

I den øverste delen av figuren 3.10 finner man de klassene som holder på informasjon gitt fra databasen (les: påmeldingsfil). Pilene på figuren indikerer arv. Den midterste raden viser brukergrensesnittklassene. Nederst er klassen som sørger for kommunikasjon med det elektroniske klokkesystemet. Vedlegg [G] viser en detaljert oversikt over objektene. Oppdelingen av figur 3.10 illustrerer også et viktig prinsipp bak kodingen av programmet; et klart skille mellom GUI klassene og de klassene som holder på data. Dette gjør programkoden oversiktlig og mer åpent for endringer. Dersom man for eksempel ønsker å endre koden for GUI'en, gjøres dette kun på en plass, og dette får ikke ringvirkninger andre steder i koden. Dersom man

ønsker å implementere andre klokkesystemer i tillegg til HEGO 8000, lar dette seg gjøre på en enkel måte.

Det er kun brukt private variable i alle klasser, noe som gjør det lettere å finne ut hvor de ulike variablene hører hjemme. Når det allikevel har vært behov for disse variablene i andre deler av koden, er det brukt egne funksjoner (get-funksjoner) i hver av klassene som returnerer disse.

3.4.3 Grensesnitt

Brukervennlighet har vært vektlagt ved utformingen av grensesnitt i programmet. Utformingen på speakervinduet er etter ønske fra våre kontaktpersoner.

3.4.4 Kommunikasjon mot klokkesystem

Klokkesystemet vi har jobbet opp mot i programmet er HEGO 8000. Dette systemet har vi hatt tilgang til i Vikingskipet på Hamar. HEGO 8000 er et tidtakersystem bestående av en prosessorstyrt elektronikkenhet (selve klokken) og en PC som styrer klokken. I tillegg kan det tilkoples diverse eksternt utstyr som startpistol, fotoceller, rulletidsdisplay og liknende.

Klokkesystemet HEGO 8000 benytter seriell kommunikasjon. Java har en egen API som muliggjør seriell kommunikasjon. Denne API'en heter javax.comm.

Det fysiske grensesnittet til displaybus'en for HEGO 8000 har følgende spesifikasjoner:

- RS-232
- 9600 baud
- 8 data bits
- 1 stop bit
- Ingen paritetskontroll

Displaybus'en tar imot informasjon fra klokken og sørger for at den sendes ut til enheter tilkoplet systemet via RS-232, som for eksempel java-applikasjonen. Eksempel på informasjon som diplaybus'en sender ut er ID'en til løper som går (et unikt nummer generert av klokkesystemet), passeringstid, rundetid og antall runder igjen til mål. Det spesielle ved utsending av informasjon er at displaybus'en kun sender ut de dataene som endres eller er nye. Med andre ord, displaybus'en sender minst mulig.

Displaybus'en mottar klokkeimpulser hvert tidelssekund. Ut fra denne opplysningen skjønner man hvorfor displaybus'en kun sender data når den må. For ikke å sende unødvendig mye data videre klarer displaybus'en å skille mellom en vanlig klokkeimpuls og en løperpassering. For enhver løperpassering sendes det også med hundredeler. Så hver gang hundredelen er satt, skjønner displaybus'en at det er snakk om en løperpassering, og sender denne passeringstiden videre.

Displaybus'en nøyer seg ikke med å skille mellom tideler og hundredeler. Dersom en løper først passerer på for eksempel 1.11,12 og deretter på 1.50,50 så sender

klokken kun de delene av tiden som har endret seg fra første til andre passering. I dette tilfellet vil det si at passering nummer to mottas som 50,50 og ikke som 1.50,50. Dette krever at alle data som er av interesse og som skal benyttes må buffres i minnet slik at man kan sammenlikne med forrige passeringstid.

Som man skjønner er det snakk om et stort antall dataoppdateringer på displaybus'en, og det krever en god organisering. Totalt består displaybus'en av 1024 byte delt inn i 32 linjer hvor hver linje tilsvarer 32 byte. Man kan si at displaybus'en er en 32x32 matrise som holder 32 byte per linje. Informasjonen som displaybus'en mottar fra klokken har sin bestemte plass i denne matrisen. Ved endrede data sender displaybus'en dataene videre. I tabell 3.1 finner man oversikt over hvilke data som ligger hvor i displaybus'en når det gjelder data som sendes til display-enheter, som for eksempel resultattavlen.

Bane	ID	Tid	Runde tid	Ranking	Differanse	Poeng	Tot. ant poeng	Ant. runder igjen
Indre	1,1,3	1,4,8	1,12,5	1,17,4	1,21,8	2,4,7	2,11,7	2,1,2
Ytre	2,19,3	2,22,8	2,30,5	3,3,4	3,7,8	3,22,7	3,30,7	3,19,2

Tabell 3.1: HEGO 8000 displaybus

Første tallet i hver celle i tabell 3.1 indikerer hvilken linje man befinner seg på i displaybus-matrisen. Neste tall, byte nummeret, indikerer hvor langt inn på aktuell linje man skal posisjonere seg. Byte nummeret etterfølges av antall byte som kommer. Det vil si Indrebane-løper sin tid kommer på linje 1, fra og med byte 4 og 8 byte "utover" i matrisen. Signalet som kommer inn på com-porten for passeringstid for løper i indre bane er følgende:

Faktisk signal: <0x02><0x20><0x27>10,23<0x03>
Forklaring på signal: <stx><linjenummer><byte nummer><**tid**><etx>

Startsignal (stx), linjenummer, byte nummer og sluttsignal (etx) kommer som heksadesimale verdier, mens tiden ligger som en ren ASCII-verdi.

Linje en, to og tre i displaybus'en holder det vi trenger av informasjon for å fange opp løpernes passeringsdata til java-applikasjonen. Linje åtte og ni sørger for signaler ut til TV. De resterende linjene i displaybus-matrisen har også sine funksjoner, men disse går vi ikke nærmere inn på.

I blant sender klokkesystemet ut synkroniseringssignaler for å sikre at alle komponentene i systemet fungerer som de skal. Synkroniseringssignalet avbryter det eventuelt pågående signalet. Dette innebærer at et signal ikke med sikkerhet fullføres, dermed kan det hende at signalets tilhørende sluttsignal aldri kommer før ett nytt startsignal sendes. Man kan faktisk ende opp med at ett og samme signal sendes to ganger da displaybus'en resettes og deretter sender alle data på ny.

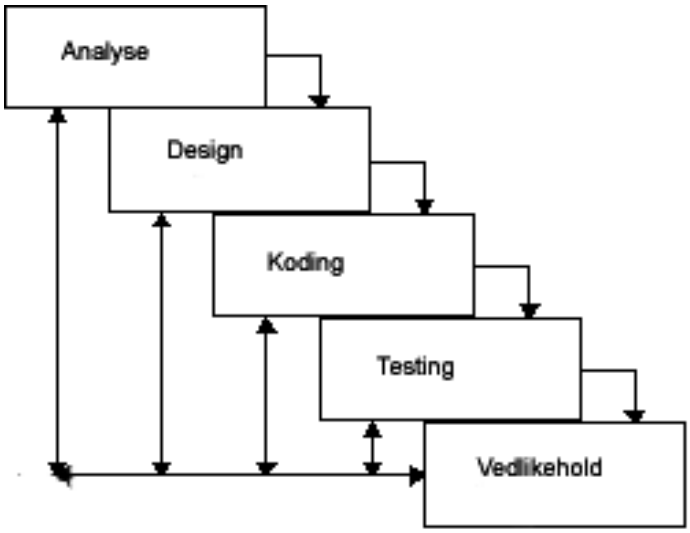
4 Koding

4.1 Systemutvikling

En livssyklusmodell er en plan for å gjennomføre en vellykket systemutviklingsprosess. Den består av en samling fastsatte trinn og oppgaver, og skal fungere som et hjelpemiddel for å nå et mål. Vi har valgt å jobbe etter fossefallsmodellen i hovedsak fordi den fokuserer mye på dokumentasjon og har klare skiller mellom fasene.

4.1.1 Fossefallsmodellen

Fossefallsmodellen er en dokumentdrevet og lineær livssyklusmodell. Hver enkelt fase avsluttes med at det leveres et dokument for godkjenning, før man tar fatt på neste fase. Dette kan føre til at det produseres store mengder dokumentasjon, og at man kan ta ressurser fra det som egentlig skal utvikles, nemlig et system.



Figur 4.1: Fossefallsmodell

De første trinnene i prosessen består av analyse og design av systemet. Deretter går man over i programmering, testing og implementering. Hver fase gjøres ferdig før man går over til neste.

Modellen er godt egnet for prosjekt hvor man har stabile krav, og hvor man jobber med velkjente og gjerne komplekse problemer. Man kan på denne måten håndtere kompleksiteten på en ryddig måte. Fossefallsmodellen gjør at man bruker mindre tid på planlegging, fordi man gjør all planleggingen i begynnelsen av hver fase.

Problemet med Fossefallsmodellen er at det kan være vanskelig å spesifisere alle kravene og ressursene i begynnelsen av prosjektet, før man har begynt å designe og kode noe. I mange tilfeller vet ikke brukerne hva de egentlig vil ha, noe som fører til at kravene fort kan endre seg underveis i prosjektet. Dette gjør at man skyver

risikoen foran seg i tid. På denne måten kan prosjektet bli svært kostbart, da man sent i prosjektet må rette opp feil fra tidligere faser. Det at man oppdater feil seint i prosjektet kan resultere i kostnadsoverskridelse og kansellering.

4.2 Valg av verktøy

- ASP velges som verktøy i utviklingen av Webgrensesnittet fordi NSF kjører en Microsoft IIS-server
- Java velges som verktøy i utvikling av programmet. Språket åpner for bruk av ulike operativsystemer, og er enkelt med tanke på å utvikle et godt brukergrensesnitt
- MySQL velges som databaseverktøy i hovedsak fordi dette er det beste freeware alternativet og er en flerbruker database.

5 Kvalitetssikring, testing og realisering

5.1 Organisering av kvalitetssikring

5.1.1 Dokumentasjon og konfigurasjonsstyring

All dokumentasjon skal til enhver tid være tilgjengelig via gruppas felles område på skolens server. De dokumenter som er ferdig redigerte og oppdaterte skal legges i en egen mappe.

Alt gruppa foretar seg (møter, planlegging og liknende) skal dokumenteres slik at vi til enhver tid er à jour med tanke på sluttrapporten.

Gruppen skal foreta back-up av dataene to ganger i uken.

5.1.2 Risikoanalyse

Det er viktig å bedømme risikoen i et prosjekt. Prosjektet påtar seg å skape resultater av en viss kvalitet innenfor en tids- og ressursramme.

Vår tidsramme er fra begynnelsen av januar og fram til innleveringsfristen 19. mai. I tillegg er det en begrensning at vi har liten økonomisk frihet. Det er også en risiko i at innleveringsfristen vår er en absolutt frist (19. mai). Derfor må vi hele tiden drive tett prosjektoppfølging for å få et best mulig resultat.

RISIKO	SANNSYNLIG	KONSEKVENS	TILTAK
En av gruppemedlemmene uteblir	Middels	Middels	Lagre og oppbevare alt materiale tilgjengelig for alle på gruppa.
Avtaler angående prosjektoppgaven blir brutt	Middels	Høy	Jevnlig kontakt med oppdragsgiver.
Uklare ansvarsforhold	Middels	Høy	Skriftlige avtaler. Klargjøre roller og ansvar.
Fare for å få en treg start	Middels	Middels	Oppstartsmøte. Fordele oppgaver på alle prosjekt deltakerne.
Prosjekt vårt blir ikke opplevd som viktig for oppdragsgiver.	Høy	Middels	Få en prosjektbeskrivelse tidlig. Avtale oppfølging/rapportering. Avtale fullføringskriteria.
Prosjektomfanget øker i løpet av prosjektperioden	Høy	Høy	Sette klare grenser for hva prosjektet skal omfatte

Tabell 5.1: Risikoanalyse

5.2 Testing

5.2.1 Webgrensesnittet

Vi har dessverre ikke hatt muligheten til å teste Webgrensesnittet i sitt rette miljø, fordi vi ikke har hatt tilgang til NSF sin serveren. Vi har derfor satt opp vår egen IIS server og testet løsningen på denne. Vi kan derfor ikke garantere at overgangen går knirkefritt. Vi hadde default oppsett på vår IIS-server, og har jobbet ut i fra det. Hvis fil-/mapperettigheter ikke er default på NSF sin server vil det bli problemer med opp- og nedlastning av påmeldings- / resultatfiler.

For å teste ut brukervennlighet og funksjonalitet har vi hatt hjelp av Hans B. Moen. Han har erfaring med å arrangere skøyteløp og representerer de som kommer til å bruke systemet i framtiden. Han har gitt oss mange konstruktive ideer om hva som var bra og hva som ikke var fullt så bra. Vi har prøvd å ta mest mulig hensyn til dette i utviklingen av Webgrensesnittet. Dette er noe som har ført til at sluttproduktet har blitt mer brukervennlig.

5.2.2 Programmet

En fullstendig testing av java programmet har vært vanskelig siden vi ikke har hatt ubegrenset tilgang til klokkesystemet i Vikingskipet. Det har i tillegg vært umulig å få testet det på et ”live” skøyteløp, siden sesongen ble avsluttet i februar/mars. Vi har allikevel fått testet hvilke data som kommer fra klokken og har fått disse korrekt inn i

programmet. I arbeidet med å planlegge hvilke moduler som bør være med i programmet har vi hatt stor nytte av Hans B. Moen og hans kompetanse på hvordan man arrangerer et skøytestevne.

Uforutsigbare hendelser som for eksempel:

- Løper faller og sklir over i feil bane ved passering
- Oppvarmingsløper som ved en feiltagelse bryter mållinjen (som resulterer i en passering)
- ”Iskloss” (brukt til oppmerking) bryter fotocellen

Dette er hendelser vi ikke har fått simulert. Låsning av fotocellene (25-30 sekunder) er med på å redusere risikoen for nevnte hendelser.

5.3 Installasjons program

Vi valgte å bruke en gratis “web installer” fra Zero G Software, Inc [14]. Dette er et installasjonsprogram som passer veldig godt til java applikasjoner. Bakdelen med denne versjonen er at den ikke støtter en ren CD installasjon. Web installeren bruker en applet til eksekveringen gjennom nettleseren og har ingen ”autorun” av .exe-filen. Det er allikevel mulig å installere programmet direkte fra en CD, men dette er en mer omfattende prosess. Vi har vært på utkikk etter andre gratisversjoner som støtter CD-installasjon, men uten hell. Hvis NSF skal distribuere Skatebase via CD, kan de kjøpe en lisens for å få til dette, eller bruke den ”web installeren” som vi har lagt opp til.

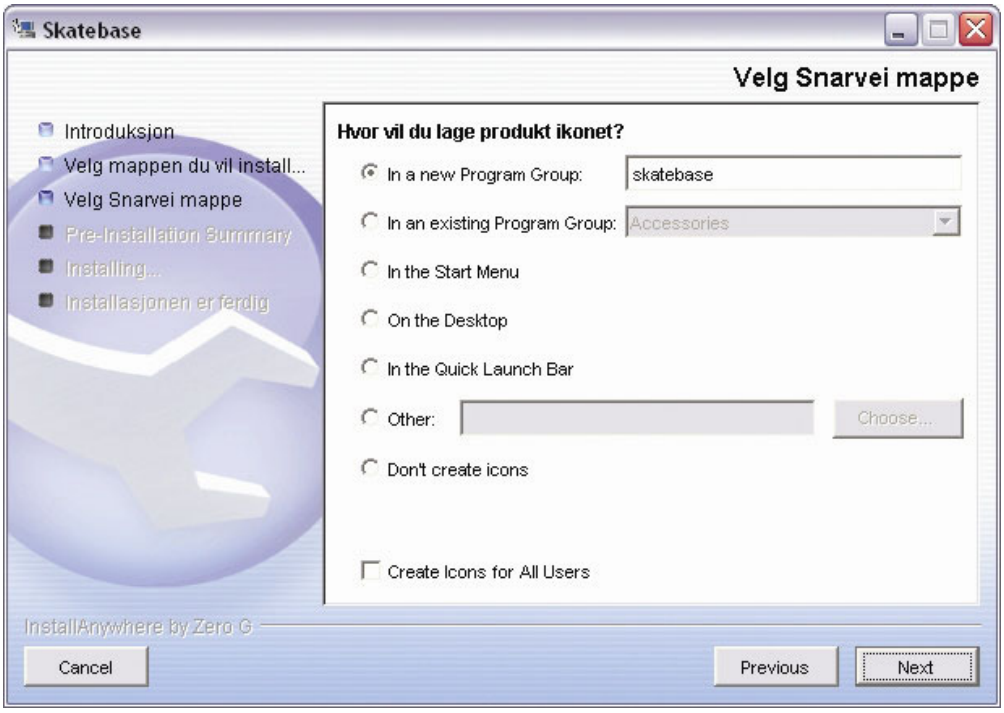
Web installeren sin HTML-fil skal virke i så å si alle nettlesere. Det vil si at alle brukere skal ha mulighet til å åpne installasjonssiden. Java applet som brukes til å eksekvere installasjonsprogrammet i nettleseren er testet i følgende nettlesere:

- Netscape Navigator 3.0 og nyere,
- Microsoft Internet Explorer 4.0 og nyere
- Opera 6.0 og nyere

For å installere filer på brukeren sin maskin bruker web installeren et signert sertifikat fra Zero G Software Inc. for å gå ut av java security ”sandbox. Hvis nettleseren støtter digitalt signerte java applets og brukeren har gitt appleten rettigheter til å utføre installasjonen, vil appleten laste ned og eksekvere softwaren. Hvis brukeren har java støtte avslått, eller nekter appleten rettigheter til å utføre installasjonen vil brukeren ha mulighet til å laste ned installasjonsprogrammet fra en normal link og kjøre den eksternt i nettleseren.

Når bruker skal installere løsningen må han ha administrator rettigheter på maskinen. Uten disse rettighetene får ikke bruker installert JVM som installasjonsprogrammet bruker til installasjonen.

Det er mulig å installere SkateBase ved hjelp av installasjonsprogrammet på Linux og Macintosh maskiner i tillegg til Windows. Installasjonen foregår i et brukervennlig grensesnitt. Se figur 5.1.



Figur 5.1: Installasjon

6 Konklusjon

6.1 Hva har vi lært i forbindelse med prosjektet?

I løpet av prosjektet har vi fått:

- God erfaring med gruppearbeid, og samarbeidet som må til for å nå felles målsetninger.
- Erfaringer med hvordan det er å gjennomføre et større prosjekt.
- En smakebit på hva som kan vente oss i en jobbsammenheng.
- God kjennskap til programmering av dynamiske Websider, samt filhåndtering og liknende ved bruk av ASP.
- Kompetanse innen COM-port programmering ved bruk av java API, samt generell bedre kunnskap om java.
- Erfaring med hvordan man kan kommunisere med andre protokoller, HEGO 8000.
- Bedre kjennskap til MySQL databaser

6.2 Hva kunne vært gjort annerledes?

De to første månedene, januar og februar, jobbet vi intensivt med kravspesifikasjonen til oppgaven. Da spesifikasjonene var på plass, startet vi med Webgrensesnittet og databasen. Vi kunne med fordel ha startet med java applikasjonen samtidig. I mars hadde vi et annet prosjekt vi måtte sette i fokus. Prosjektet tok mer tid enn beregnet, noe som gikk mest utover java delen i hovedprosjektet. Det er mulig vi undervurderte omfanget av java applikasjonen. Sett i ettertid fikk vi nok for liten tid til applikasjonen. Hvis vi hadde klart å forutse dette, ville vi fått en jevnere arbeidsinnsats med hovedprosjektet. Det ble litt for mye intensive jobbing i starten og slutten av prosjektet.

I starten la vi liten vekt på GUI i Webgrensesnittet. Vi konsentrerte oss mer om hvordan vi skulle få ting til å virke. Mulig vi skulle lagt litt mer vekt på GUI tidligere, slik at vi kunne gitt ut smakebiter og fått tilbakemeldinger på dette.

I og med at HIG kun kunne låne oss en datamaskin, og den ikke var tilgjengelig før ut i februar, valgte vi å opprette hjemmekontor. Mulig vi kunne presset mer på for å få lånt flere maskiner til gruppen, men dette valgte vi ikke å gjøre.

6.3 Vurdering av samarbeidet

6.3.1 Innad i gruppen

Alle på gruppen kjente hverandre før hovedprosjektet startet, og har jobbet sammen på mindre prosjekter tidligere. Dette følte vi var betryggende, fordi vi visste da hva vi

kunne forvente oss av hverandre før vi gikk i gang med et stort prosjekt. I likhet med tidligere prosjekter har samarbeidet fungert veldig bra.

Siden vi ikke tok i bruk grupperommet HIG tilbød oss, opprettet vi to hjemmekontor. Vi opplevde ingen problemer med denne løsningen, fordi vi møttes daglig og hvis noen hadde problemer jobbet vi i fellesskap for å løse disse. Vi hadde jevnlig statusmøter og fikk på denne måten en god oversikt over framdriften i prosjektet.

Siden alle gruppemedlemmene var fra forskjellige studieretninger, følt vi det var naturlig å dele oppgaven inn i tre forskjellige hovedtemaer; java applikasjonen, Webgrensesnittet og databasen. Vi visste hvilken kompetanse hver enkelt hadde, så hvert medlem ble ”hovedansvarlig” for hver sitt hovedtema. Hovedansvarlig planla og fordelt oppgaver innenfor sitt hovedområde til de andre gruppemedlemmene. Vi prøvde i størst mulig grad å forhindre at kun en person ble sittende med et hovedtema, uten deltagelse fra de andre. Mot slutten av prosjektet når deadline begynte å nærme seg, hadde vi ikke noe annet valg enn å la hver hovedansvarlig kun jobbe med ”sitt” hovedtema.

6.3.2 Med oppdragsgiver

Samarbeidet med oppdragsgiver har ikke fungert helt tilfredsstillende. Kontaktpersonene kom fra forskjellige deler av landet, så det var vanskelig for dem å møtes. De kunne da ikke komme fram til et felles innspill på hva som var bra, hva de savnet og hva som burde endres. Vi fikk kanskje for mange ”uavhengige” kontaktpersoner som kom med sine personlige innspill, meninger og ideer. Det ble dermed litt vanskelig for oss å tilfredsstille alle kontaktpersonene sine ønsker.

Planleggingen av hva systemet grovt sett skulle gjøre, syntes vi gikk greit. Vi fikk kartlagt hva prototypen skulle omfatte, og hvilke funksjoner vi skulle utelate. Vi hadde da enkelte retningslinjer å jobbe etter.

6.4 Subjektiv opplevelse av prosjektet

Hovedprosjektet er som kjent avslutningen på ingeniørutdanningen. Vi har gjennom tre år hatt mange forskjellige fag, men det er først nå vi ser den virkelige nytten av det brede fag aspektet ingeniørutdanningen gir. Arbeidsprosessen i et prosjektet er ikke noe man kan lese seg til. Kunnskapen kommer gjennom erfaringer.

Gjennom prosjektet har vi jobbet på en gruppe hvor alle medlemmene har sine egne meninger om hvordan ting bør gjøres. Vi syntes vi har fått mye ut av dette samarbeidet, samt fått en god pekepinne på hvordan det er å jobbe med andre mennesker i en jobb relatert sammenheng.

Vi har dokumentert prototypen godt og fått med oss de funksjonene som oppdragsgiver ville ha med i løsningen, så vi kan ikke annet enn si oss fornøyde med sluttproduktet.

6.5 Konklusjon på prosjektarbeidet

Skøytemiljøet i Norge har stort sett enkle, men tidkrevende metoder for å arrangere et stevne. Det er mange dugnadstimer som går med til nettopp dette formål. Enkelte ”ildsjeler” i miljøet har utviklet enkle programmer som foretar trekning og liknende for blant annet å få ned antall dugnadstimer. Slik vi har oppfattet det er det registreringen av påmeldinger og stevneresultater som tar mest tid. Hvordan kan vi forenkle denne prosessen ytterligere? Dette er et spørsmål som vi har hatt i tankene fra starten av. Vi ønsket å lage et system som er brukervennlig, effektivt og som kan gjøre jobben så enkel som mulig for en arrangør.

Prototypen som er utviklet gjør at mye av den jobben som tidligere har blitt gjort manuelt nå kan skje automatisk. Det finnes selvfølgelig rom for å videreutvikle systemet, men vi mener selv at vi har fått med de viktigste oppgavene systemet bør kunne utføre.

Ved hjelp av Webgrensesnittet kan en klubb melde på sine løpere til et stevne. Når påmeldingsfristen er passert, kan arrangøren laste ned påmeldingslisten. Java applikasjonen tar imot påmeldingslisten, og arrangøren kan gjøre endringer på den, foreta trekninger samt manuell eller elektronisk tidtaking ved bruk av applikasjonen. Når et stevne er ferdig genererer applikasjonen en resultatliste som kan skrives ut på printer. Arrangøren laster så opp resultatfilen til databasen ved hjelp av Webgrensesnittet, og stevnet blir registrert. Alle tider, plasseringer og personlige rekorder blir da oppdatert automatisk i databasen.

Disse oppgavene har tidligere ført til mye papirarbeid og telefoner/faksing for selve registreringen. Vi føler at vi har klart å effektivisere dette på en tilfredsstillende og effektiv måte gjennom prototypen. Så vi mener selv vi har klart å tilfredsstille målsetningen vi hadde i starten av prosjektet.

7 Litteraturliste

1. www.java.sun.com
2. http://java.sun.com/j2se/1.4.1/docs/api/
3. Java - How to Program, Third Edition by Deitel&Deitel
4. http://www.taltech.com/TALtech_Web/resources/
5. http://java.sun.com/products/javacomm/javadocs/javax/comm/package-tree.html
6. www.arcelect.com/rs232.htm
7. Data protocol Displaybus HEGO 8000 speed skating - Siv.Ing. Haakon Wiig AS / HEGO Timing Systems
8. Aktive Server Sider – kompendium av Øyvind Kolloen
9. http://www.activeserverpages.com
10. http://www.mysql.com/doc/en/index.html
11. http://www.w3schools.com
12. MySQL Bible – Steve Suehring
13. Jacob ”Beezle” Gilley, avis7@airmail.net, upload.asp, filopplastningsbiblotek for ASP.
14. http://www.zerog.com/products_ia_07.html#4

8 Figur- og tabelloversikt

9 Vedlegg

A. Gantt skjema
B. Milepælsplan
C. Dataflytdiagram
D. ER-modell
E. SQL kode
F. CSS fil
G. Filformat påmelding
H. Filformat resultatliste
I. Objekter i java programmet
J. Java kode
K. Møtereferater
L. Statusrapport
M. Filer på CD-ROM