BACHELOROPPGAVE:

Mobil skademeldingsmodul med etterbehandling

FORFATTERE:
Arve Eidjord
Øyvind M. Kongsengen

Dato: 23.5.2012

SAMMENDRAG AV BACHELOROPPGAVEN

Tittel:	Mobil skademeldingsmodul med etterbehandling	Nr. :
		Dato : 23.5.12
Deltaker(e):	Arve Eidjord	
	Øyvind M. Kongsengen	
Veileder(e):	Høgskolelektor Tom Røise	
Oppdragsgiver:	Electric Time Car AS	
Kontaktperson:	Dag L. Solhaug og Øyvind Flatval	
Stikkord (4 stk)	Skademelding, ETC, PhoneGap, Tomcat	

Antall sider: 125	Antall bilag: 12	Tilgjengelighet: Åpen

Kort beskrivelse av bacheloroppgaven:

Vi har for vår oppdragsgiver Electric Time Car AS (ETC), utviklet en applikasjon til mobile enheter for digital utfylling og behandling av skademeldinger ved trafikkuhell. ETC er et firma som leverer løsninger for bilhold til private bedrifter og offentlig sektor.
Prosjektet består av tre deler; en mobil-app, et web-grensesnitt og en serverdel.

Den mobile applikasjonen inneholder funksjonalitet for å registrere alle de påkrevde opplysningene som må oppgis i papirversjonen av skademeldingsskjemaet ved et trafikkuhell. Det er også i tillegg til skjemaet laget et verktøy for å visuelt beskrive skadesituasjonen og for å markere skader på bilen. Når utfyllingen av skademeldingen er ferdig kan denne sendes til server for etterbehandling og godkjenning.

I tillegg til appen er det også utviklet et web-grensesnitt for etterbehandling av skademeldinger. Her vil brukerne ha mulighet til å logge seg inn for å se over skademeldinger som er sendt inn fra appen, og gjøre eventuelle endringer før de signeres. Vi har også utviklet en serverdel slik at appen og web-grensesnittet skal kunne fungere sammen. Denne inneholder prosedyrer for behandling av skademeldingskjema, innlogging av brukere og database lagring.

En av hovedutfordringene i prosjektet har vært å finne en løsning for å bygge kryssplattform apps, og sette seg inn i de ulike rammeverkene vi har valgt for å utvikle appen. Videre har det vært en utfordring å lære seg serverteknologier som JSP og servlets, og å få alle systemets komponenter til å fungere sammen.

Forord

Skademeldingssystemet har blitt utviklet av to studenter ved Bachelor i Programvareutvikling ved Høgskolen i Gjøvik. Oppgaven har vært krevende, men også veldig lærerik og har gitt oss et godt innblikk i hvordan det er å drive systemutvikling. Vi har nå fått kjennskap til utvikling på mobile enheter og serversideprogrammering. Vi har også tilegnet oss praktiske kunnskaper i forhold til prosessen med å utvikle et større system. Til tross for mye hardt arbeid har prosjektet vært en veldig positiv opplevelse.

Vi takker veileder Tom Røise for gode tilbakemeldinger og hjelp gjennom hele prosjektet.

Vi vil også takke Dag L. Solhaug og Øyvind Flatval ved ETC for at vi fikk muligheten til å jobbe med denne oppgaven. Vi setter pris på at dere har vært så tilgjengelige for møter, og stilt opp når vi har hatt behov for det. Takk for god veiledning og et godt samarbeid underveis.

Gjøvik, 23.5.2012

Innhold

Figurliste

1 Innledning

1.1 Introduksjon

1.1.1 Problemområde

Dagens teknologiske utvikling gir oss stadig nye muligheter. Stadig flere tjenester blir digitalisert og dermed også forenklet. Bilførere som har vært ute for et uhell kjenner til skademeldingsskjemaet som må fylles ut og sendes til forsikringsselskapet. Dette er en prosess som kan være tidkrevende og lett bli ignorert ved mindre uhell, spesielt dersom føreren kjører for et firma og dermed ikke er eier av bilen.

Electric Time Car AS (ETC) som er vår oppdragsgiver leverer løsninger for bilhold til bedrifter og offentlig sektor gjennom systemet CarAdmin. Dette systemet er en hjelp til å administrere og ha oversikt over bilparken sin. ETC ser nå behovet for å kunne registrere skader på bilene på en enklere måte. I dag finnes det ikke noe elektronisk alternativ for å skrive en skademelding. Derfor ønsker ETC et system der skademeldinger kan behandles digitalt. Systemet skal være tilgjengelig for både privatpersoner og de som er ansatt i en organisasjon eller bedrift som har en bilpark. Meningen er at alle skal kunne bruke systemet. Men på grunn av at ETC må inngå avtataler med de ulike forsikringsselskaper før systemet kan tas i bruk vil ikke systemet bli offentlig tilgjengelig før dette er på plass.

Figur 1: Utfylling av en skademelding kan være utfordrende og tidkrevende

Et slikt system vil gjøre utfyllingen av skademeldingene mye enklere og vil ikke lenger kreve at partene i skademeldingen fyller ut all informasjonen på skadestedet. De kan fylle ut kun det viktigste for å se over alt i ettertid før skademeldingen signeres og sendes. For de som er ansvarlige for en bilpark vil det være en fordel fordi det vil gi et felles system der alle

skademeldingene samles. Det vil dermed være mer oversiktlig i forhold til hvilke biler eller førere som har vært involvert i ulykkene.

1.1.2 Oppgavedefinisjon og avgrensning

Vi har ut fra denne problemstillingen fått i oppgave å lage et system for å kunne skrive og behandle skademeldinger. Denne oppgaven kan deles inn i flere deler.

Det skal for det første utvikles en applikasjon for bruk på håndholdte enheter. Applikasjonen er tiltenkt både privatbrukere og firmabrukere og skal være et alternativ til papirutgaven av skademeldingen som brukes i dag. Dette betyr at man ved en ulykke, skal kunne registrere alle nødvendige opplysninger om de involverte, bilinformasjon, skadeomfanget, og stedet hvor ulykken skjedde. Dette vil da til sammen utgjøre et komplett skademeldingsskjema. Opplysninger om føreren/brukeren skal hentes fra tidligere skademeldinger der dette er naturlig, for å gjøre utfyllingen raskere. Den mobile applikasjonen skal ha et visuelt rapporteringsgrensesnitt som inkluderer opptegning av skadesituasjonen. En skademelding skal videre kunne sendes til sentral server for videre behandling og til slutt signering og innsending til forsikringsselskap. Selve innsendingen til forsikringsselskapet ligger utenfor oppgaven ettersom det ikke finnes noen garanti for at disse ønsker å ta i mot skademeldingene på en slik måte. Skademeldingsskjemaet skal videre være konfigurerbart slik at det kan tilpasses etter kundens behov ved at ulike felter i skademeldingen kan skjules. Dette gjelder spesielt for bedrifter hvor alle felter som finnes i et vanlig skademeldingsskjema kanskje ikke vil være nødvendig. Applikasjonen skal lages og testes for en plattform, men skal utvikles på en slik måte at den senere kan kjøre kryssplattform. De plattformer som ansees som aktuelle er Android, iOS og Windows Phone.

Det skal også utvikles et webgrensesnitt for etterbehandlingen av skademeldingene. Dette skal være et selvstendig system som videre skal kunne brukes i tredjepartsløsninger. Bruker skal her kunne logge seg inn, hente ut igjen de innsendte opplysningene fra skademeldingene, og gjøre eventuelle endringer før skjemaet blir godkjent. Oppdragsgiver vil legge premisser for dette grensesnittet slik at modulen kan integreres i det eksisterende CarAdmin systemet.

For at disse to delene skal kunne fungere sammen skal det også utvikles en serverdel som behandler dataene. Selve serveren vil eies og driftes av oppdragsgiver. Det er dermed kun programvaren som skal utvikles.

I framtiden vil systemet vårt kunne integreres tettere med CarAdmin-systemet til oppdragsgiver for å dele informasjon mellom disse to.

1.2 Målgrupper

Denne oppgaven består av en denne rapporten og i tillegg det som er prosjektet, selve systemet som utvikles. Disse har ulike målgrupper.

Målgruppen for rapporten er i hovedsak sensor, veileder og andre studenter som kan ha interesse av den. I tillegg er også oppdragsgiver i målgruppen fordi denne rapporten vil være en dokumentasjon og en videre hjelp i utviklingen av systemet.

I hovedsak er det oppdragsgiver selve systemet er laget for. Dette er naturlig ettersom disse har etterspurt systemet og skal utvikle det videre.

1.3 Mål

1.3.1 Resultatmål

Målet med oppgaven er å utvikle et system for innlevering av skademeldinger, både for privatpersoner og for personer som er ansatt i firma hvor CarAdmin-systemet brukes.

App-delen av systemet kan lastes ned og testes og vil være tilgjengelig fra 07. juni 2012 til 1. august 2012[1].

1.3.2 Effektmål

Formålet med oppgaven er å forenkle prosessen med å skrive og levere skademeldinger. Dette vil også kunne føre til at ennå flere leverer inn skademeldinger. Det vil gjøre prosessen mer tilgjengelig på en ny arena. Samtidig er formålet også å gjøre hverdagen enklere for de som er ansvarlige for en bilpark ved å forenkle saksbehandlingen av skademeldingen. Også forsikringsselskapene vil kunne nyte godt av dette. For ETC sin del er et mål at systemet skal være lansert for alle deres kunder innen ett år og at alle disse da vil bruke systemet.

1.4 Vår faglige bakgrunn og kompetanse

Begge medlemmene av prosjektgruppen er studenter ved Høgskolen i Gjøvik, og går linjen for programvareutvikling. Gjennom dette studiet har vi opparbeidet oss kunnskap om systemutvikling. Sentralt i studiet har vært emner innen programmering, databaser, og teorifag knyttet til organisering av systemutviklingsprosjekter. Av programmeringsspråk har vi gjennom studiene blant annet fått kjennskap til C++ og Java, men har også blitt introdusert for andre språk som PHP, JavaScript og SQL.

[1] http://hovedprosjekter.hig.no/v2012/imt/in/skademelding/app/

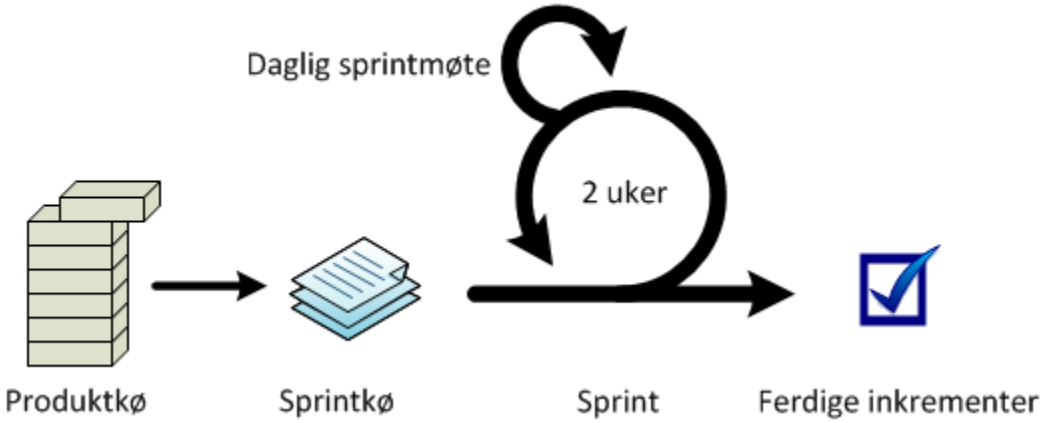
Ingen av oss hadde noen erfaring med programmering på håndholdte enheter, så dette var noe vi måtte sette oss inn i ved starten av prosjektet. Selv om vi hadde erfaring med Java hadde vi ikke brukt dette til server-programmering før, og dermed måtte vi også lære oss hvordan JSP og Java-servlets fungerer.

Denne oppgaven, sammen med flere andre, ble første gang presentert høsten 2011 som gruppearbeid i emnet IMT3102 Objektorientert Systemutvikling. Øyvind sin gruppe utarbeidet da en kravspesifikasjon og et designdokument. Han fikk med det en intro til bacheloroppgaven. Den oppgaven har vært et utkast til deler av denne bacheloroppgaven, men er blitt svært mye omarbeidet.

1.5 Arbeidsprosess

Vi har valgt å arbeide fast fire hele dager i uken ettersom dette passet godt med timeplanen i forhold til fag ved siden av dette prosjektet. Vi har hatt som utgangspunkt at vi skal samles som en gruppe når vi arbeider. Dette har gjort kommunikasjonen enkel og har ført til godt samarbeid for eksempel ved at en kunne hjelpe til å løse de ulike problemene en sto overfor.

I prosjektplanen, se vedlegg F, ble det kartlagt at vi ønsket å benytte oss av Scrum som utviklingsrammeverk. Dette gjør at vi, slik som Hasle trekker fram i sin bok at vi får en klar markering i forhold til "selvstyring og unngåelsen av et utviklingsforløp beskrevet på forhånd" [1].



Figur 2: Scrumprosessen

Ved starten av dagen har vi hatt daglige møter for å kartlegge fremdriften av prosjektet. Her har vi fått fram hva vi gjorde i går, hvilke utfordringer vi har hatt og hva vi skal gjøre i dag.

Når det gjelder lengden på sprinten er disse ofte fra to til seks uker [2]. Å velge korte sprinter vil hjelpe oss til å holde oss smidige(agile) siden vi ofte kan stake ut ny retning på arbeidet. Hvis vi ser at arbeidet går i feil retning kan vi raskt gjøre endringer i neste sprint. Dette gjør at vi får hyppige tilbakemeldinger og leveringssekvens, men på en annen side kan det være lurt å velge lengre sprinter. Dette gjør at man i hver sprint får bedre tid til og oversikt over de oppgavene som skal utføres. Samtidig har man bedre tid til å hente seg inn hvis man starter på en oppgave i feil retning, og kan dermed klare å nå sprintmålet. I tillegg vil det brukes mindre tid på de ulike møtene siden disse forekommer mer sjeldent, noe som vil si at du får mer tid til selve utviklingen [2]. Vi har valgt å benytte oss av korte sprinter på bare to uker. Dette har gjort at vi får mer erfaring med gjennomføring av for eksempel sprintmøter, demomøter og retrospektive møter. Samtidig vil det hjelpe oss til å ikke miste fokuset fordi vi har hyppige leveringer. Det gjør også at oppdragsgiver og veileder kan rettlede oss på tidlige tidspunkt.

Produktkøen består av alle de elementene som skal utføres, den funksjonaliteten som ønskes. I følge Kniberg er det produkteier som eier produktkøen. Ofte så er det også han som utarbeider den, i samarbeid med Scrum-teamet. Oppdragsgiver har overfor oss fungert som produkteier, men i hovedsak er det vi som har utarbeidet produktkøen ut fra de innspill vi har fått fra oppdragsgiver. Dette har gjort at vi har fått satt oss inn i hvordan en slik produktkø fungerer og oppdragsgiver har hatt kontrollen på at det som er i den til enhver tid er relevant.

Ved hvert sprintmøte utarbeidet vi en sprintkø som besto av den funksjonaliteten vi skulle utvikle i den kommende sprinten. Som produkteier har oppdragsgiver prioritert hvilke elementer som var de viktigste å få gjort. Selv om det var de som hadde ansvaret for dette så hadde vi en dialog om dette på møtene for å tilpasse det til vår oppgave. Videre ble det estimert av oss hvor lang tid vi mente at vi kom til å bruke på de enkelte elementene. Estimering er alltid en utfordring. Vi har valgt å benytte oss av timer til estimering, noe som mange gjør. Men det er også vanlig å benytte seg av dager, og da at en dag er for eksempel seks timer eller timer omregnet til poeng [3]. Stort sett har vi estimert sammen i gruppe og da med å kombinere estimater vi har gjort hver for oss. Dette gjør at vi har fått fram hvor mye tid de enkelte ting kan ta. Simula, som driver forskning på blant annet områder innen systemutvikling har funnet at det har vært en stor økning i bruken av gruppeestimering, både der prosessen er strukturert med for eksempel planning poker og der diskusjonene er mer løse [4].

For å holde oversikt over arbeidet underveis i sprinten, har vi benyttet oss av taskboard. Vi valgte å dele dette taskboardet opp i Todo, Doing og Done. Siden vi ikke har hatt noe fast grupperom under bachelorprosjektet hadde vi ikke muligheten til å bruke en tavle/vegg til taskboard, slik Kniberg anbefaler [2]. Vi eksperimenterte derfor først med å tegne opp

taskboardet på papir og brukte post-it-lapper for elementene. Dette fungerte forholdsvis greit for oss som bare er en gruppe på to. Etter sprint 3 oppdaget vi Trello som er en web-applikasjon for å organisere prosjekter i et taskboard [5]. Trello dekket de behov vi hadde i forhold til taskboardet, vi valgte derfor å ta i bruk dette verktøyet i resten av sprintene.

Ved slutten av hver sprint hadde vi et demomøte med oppdragsgiver. Her ble de elementene vi hadde arbeidet med vist fram, og deretter testet av oppdragsgiver. Sprintmøtet og demomøtet ble laget samme dag, rett etter hverandre. Dette har gjort at både vi og oppdragsgiver får effektive møter. Gruppen hadde også et retrospektivmøte for å se hva vi hadde gjort i sprinten og hva vi kunne gjøre bedre til neste.

Ettersom ETC har mye erfaring innen utvikling har vi også underveis i sprintene benyttet oss av deres kunnskap ved å ha møter. Dette har hjulpet oss til å vise framgangen i sprinten og samtidig gjort at de kan rettlede oss på punkter der vi sitter fast eller må gjøre ulike valg.

1.6 Øvrige roller

1.6.1 Oppdragsgiver

Oppdragsgiver er ETC som er et produkt- og konsulentselskap som utvikler og selger løsninger for blant annet bilhold. CarAdmin-systemet er hovedløsningen og er en løsning for kjøretøyoppfølgning av alle typer kjøretøy. Det er rettet mot både private bedrifter og offentlig sektor.

Formelt sett er det daglig leder Dag L. Solhaug som er kontaktpersonen vår i ETC. Han er den som har lagt fram oppgaven og sitter dermed med selve ideen rundt den. Han har dermed hatt mange innspill i hvordan vi burde gå fram i selve løsningen. I praksis er det Øyvind Flatval vi for det meste har forholdt oss til. Det vil si at det er han som har bistått oss og veiledet oss underveis i arbeidet fra ETC sitt ståsted. Vi har til sammen hatt 11 møter med oppdragsgiver, på tre av disse var Dag til stede, på de andre bare Øyvind.

Vi har merket oss at det kan være utfordrende når flere personer i praksis blir produkteiere. Øyvind og Dag har til tider hatt ulike oppfatninger om hvilken funksjonalitet som bør prioriteres og hvordan de bør implementeres. Siden Dag kun har vært med på noen få møter har dette ført til at vi har gjort i visse justeringer i funksjonaliteten i ettertid av implementasjonen.

1.6.2 Veileder

Veilederen vår i forbindelse med denne oppgaven har vært Høgskolelektor Tom Røise. Han underviser i blant annet Objektorientert Systemutvikling der Øyvind fikk en intro til denne

oppgaven. Veiledningsmøter har vi som oftest hatt ukentlig. Her har vi tatt opp ulike spørsmål rundt utviklingen og rapportskriving. Dette har hjulpet oss til å holde rett kurs.

1.7 Terminologi

Vi vil i oppgaven benytte oss av betegnelsen **app** når vi skal omtale en applikasjon på en håndholdt enhet. Dette kan enten være en smarttelefon eller et nettbrett. Med **systemet** menes hele løsningen og inkluderer da både web-løsningen, appen og serverdelen.

En **skademelding** viser til alle de feltene som er nødvendige for å beskrive og dokumentere hele skadesituasjonen og samsvarer i stor grad med papirutgaven av en skademelding. Papirutgaven av en skademelding kan sees i Vedlegg B. I en skademelding er det minst én part, maksimalt to. En del i skademeldingen er felles for begge partene, mens den andre delen er individuell. Denne individuelle delen kan deles inn i **basisopplysninger** og **supplerende opplysninger**. Basisopplysninger er i hovedsak informasjon om kjøretøy, fører, forsikringstaker og forsikringsselskap. Supplerende opplysninger er informasjon om fart, veidekke, skadede personer og lignende. Forskjellen på disse to settene med opplysninger, er at begge partene må godkjenne og signere det som er skrevet om begge sine basisopplysninger.

For en fullstendig terminologiliste, se vedlegg A.

1.8 Organisering av rapporten

Rapporten er delt inn i åtte kapitler og har i tillegg egne underkapitler som til sammen gir tre nummererte nivåer. Bare de to øverste nivåene finnes i innholdsfortegnelsen. I de innledende sidene blir det benyttet romertall for å skille disse fra selve rapporten. Figurer i rapporten har tilhørende nummer og tekst. En figurliste finnes mellom innholdsfortegnelsen og kapittel 1.

1.8.1 Kapitteloppsummering

1 Innledning

Inneholder en introduksjon til prosjektet, hva målene er, beskrivelse av arbeidsmetoder og de ulike rollene.

2 Kravspesifikasjon

Inneholder de krav som er satt til oppgaven. Dette kommer til uttrykk gjennom blant annet Use case og sekvensdiagram.

3 Design

Denne delen tar for seg hvordan systemet er utformet med tanke på brukergrensesnittet og arkitektur.

4 Valg og bruk av teknologi

Denne delen inneholder en beskrivelse av valg og bruk av ulike verktøy og teknologier.

5 Implementasjon
Inneholder beskrivelse om hvordan de ulike delene av systemet er implementert.

6 Testing
Beskriver hvordan vi har utført testing på systemet.

7 Avslutning
Inneholder drøftinger, evalueringer, forbedringspotensialer, videreutvikling og konklusjon.

8 Litteraturliste
Inneholder en liste over de ulike kildene vi har brukt for å begrunne og diskutere ulike deler i oppgaven.

9 Vedlegg
Vedlegg i oppgaven er: Terminologiliste, Europeisk skademeldingsskjema, Eksempelkode, SQL-kommandoer, Skjermbilder, Prosjektplan, Sprintkø, Logg, Møtereferater, Prosjektavtale, Oppsett av utviklingsmiljø.

2 Kravspesifikasjon

I dette kapittelet skal vi se på hva systemet skal gjøre og hvilke krav som stilles til det. Vi skal først se på systemets brukere før vi definerer kravene og kommer inn på supplementærspesifikasjonen.

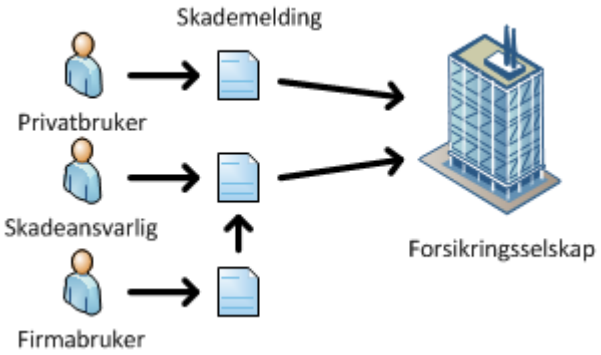
2.1 Systemets brukere

I systemet finnes det tre forskjellige brukere som innehar forskjellige roller.

En privatbruker kan opprette skademeldinger og sende dem til server for å kunne gjøre endringer på dem før han sender med til forsikringsselskapet.

En skadeansvarlig har firmabrukere under seg og er ansvarlig for å godkjenne og sende inn skademeldinger som firmabrukere har skrevet. Han kan også tilføye kommentarer på disse skademeldingene.

En firmabruker kan opprette skademeldinger, sende de inn til server og gjøre endringer på de på web. Firmabrukeren kan ikke selv sende skademeldingen til forsikringsselskapet ettersom dette er skadeansvarlig sin oppgave. Dette kan sees i figuren under.



Figur 3: Figuren viser hvem som kan opprette skademeldinger, og flyten for levering til forsikringsselskapet

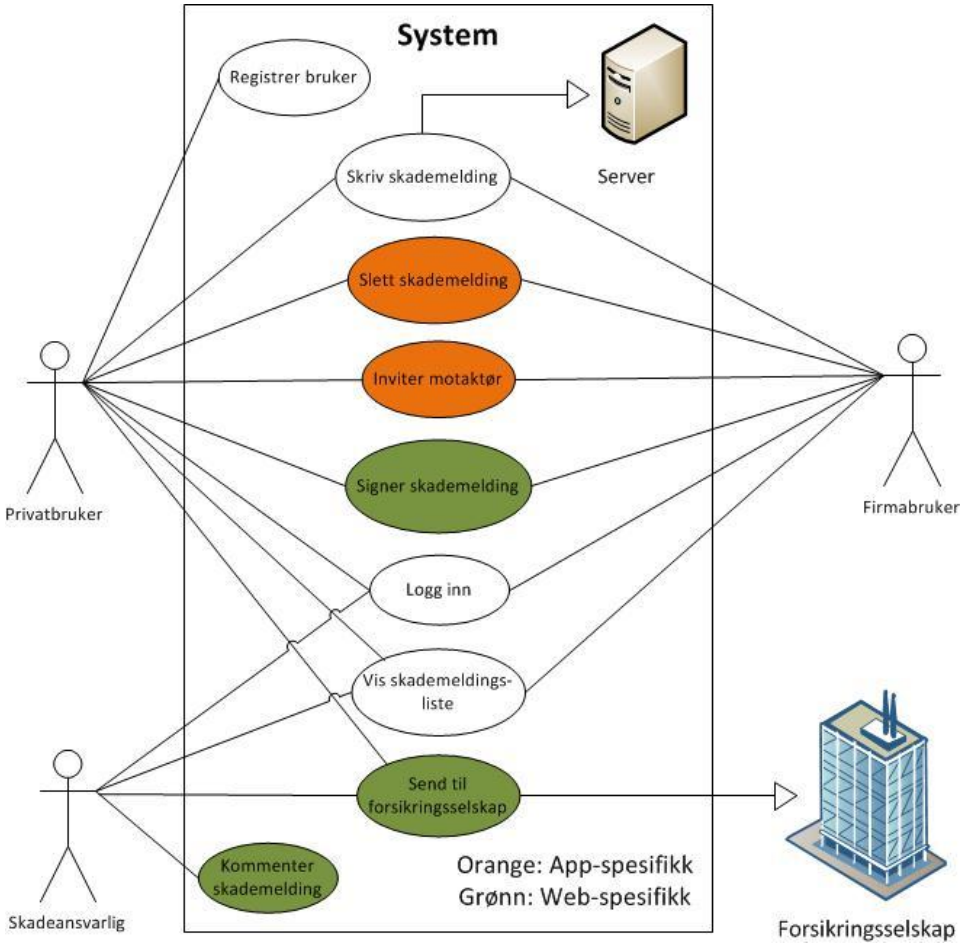
2.2 Funksjonelle krav

Å beskrive funksjonelle krav til systemet kan gjøres på forskjellige måter. En kjent metode er å bruke Use case. Her er fokuset på å få fram interaksjonen mellom brukeren og systemet. Denne metoden gjør bruk av Use case diagrammer og overordnet og detaljerte Use case beskrivelser. I følge Jeff Sutherland er praksisen i Scrum å benytte seg av User stories. Disse er tekstlige beskrivelser av hva brukeren gjør, eller må gjøre for å få uført en oppgave [6]. User story-ene er da ofte beskrivelser av de ulike elementene i produktkøen. Noen kan være korte med lite detaljer, mens andre er mer detaljerte. Det kan være vanskelig å utforme gode User stories, så

derfor kan Use Case føles mer håndfaste [7]. På grunn av dette og fordi vi tidligere i utdanningsløpet har jobbet en god del med Use case ønsker vi her å benytte oss av dette når vi nå skal se på de funksjonelle kravene.

2.2.1 Use case diagram

Use case diagrammet under er en beskrivelse av funksjonaliteten til de ulike aktørene i systemet.



Figur 4: Use case diagram

Use case-et Skriv skademelding er et Use case som er i størrelsesorden mye større og mer komplekst en de andre. Vi ser at en firmabruker og en privatbruker er de som oppretter og skriver skademeldinger. Skriv skademelding består av alt fra skrivingen og avslutter ikke før skademeldingen er sendt til server. Dette kommer fram når vi beskriver dette i de detaljerte use case-ene.

Server peker til den serverdelen vi skal lage. Legg merke til at bare skadeansvarlige og privatbruker kan sende til forsikringsselskap. Vi skal ikke implementere denne funksjonen, men den er tatt med for å vise hvordan dette er tenkt i framtiden.

2.2.2 High level use case

Use case	**Registrer bruker**
Aktører	Privatbruker
Mål	Opprette en bruker slik at man kan logge inn i systemet
Beskrivelse	Når en aktør første gang bruker appen, blir han bedt om å registrere seg for å få tilgang. Dette gjøres ved å oppgi en e-post og velge passord.
Variasjoner	

Use case	**Logg inn**
Aktører	Firmabruker, privatbruker, skadeansvarlig
Mål	Få tilgang til systemet, enten på app eller web.
Beskrivelse	Bruker angir sitt brukernavn og passord, og får tilgang til systemet.
Variasjoner	Dersom en bruker har logget inn en gang på appen skal han senere bli automatisk logget inn.

Use case	**Vis skademeldingsliste**
Aktører	Firmabruker, privatbruker, skadeansvarlig
Mål	Se en liste over alle aktuelle skademeldinger
Beskrivelse	Bruker får en oversikt over alle de skademeldingene han selv har skrevet. Slik at han kan endre på dem, signere dem eller sende dem til forsikringsselskapet.
Variasjoner	Dersom brukeren er skadeansvarlig får han også en oversikt over skademeldingene til alle sine firmabrukere.

Use case	**Send til forsikringsselskap**
Aktører	Skadeansvarlig, Privatbruker
Mål	Bekrefte at en skademelding er rett utfylt og sende den til forsikringsselskapet.
Beskrivelse	Bruker ser over en skademelding for å forsikre seg om at alt er rett fylt ut. Denne sendes så til forsikringsselskapet.
Variasjoner	

Use case	**Kommenter skademelding**
Aktører	Skadeansvarlig
Mål	Legge til kommentar på en skademelding
Beskrivelse	Skadeansvarlig kan gå inn på hver enkelt skademelding og tilføre kommentarer før han sender den.

Variasjoner	

Use case	Signer skademelding
Aktører	Firmabruker, Privatbruker
Mål	Signere en skademelding
Beskrivelse	Bruker signerer skademeldingen, og godkjenner at basisopplysningene er korrekte
Variasjoner	Bruker velger “avslå signering”. Begge brukerne må da signere på nytt etter eventuelle korreksjoner.

Use case	Slett skademelding
Aktører	Firmabruker, Privatbruker
Mål	Slette en skademelding i appen.
Beskrivelse	Bruker kan velge å slette skademeldinger fra listen i appen.
Variasjoner	

2.2.3 Detaljerte use case

I use caset under blir det forklart hva som skjer når 2 aktører skriver skademelding på samme telefon. Det er viktig å merke seg at i punkt 15 er det fører av kjøretøy B som er aktøren. I resten av punktene er det fører av kjøretøy A som er aktøren.

Use case	**Skriv skademelding**
Aktør	Privatbruker, firmabruker
Mål	Fylle ut en skademelding
Sammendrag	Bruker fyller ut de ulike delene av en skademelding (fellesinformasjon, basisinformasjon og supplerende informasjon) og sender den deretter til server.
Hendelsesforløp	
Aktør	**Systemrespons**
1. Bruker velger ny skademelding	
	2. Systemet viser valg: “Inviter motpart”, “Egenskade”, “To involverte kjøretøy”, “Hent invitasjon”

3. Bruker velger “To involverte kjøretøy”	
	4. Systemet viser skademeldingsskjemaet
5. Bruker fyller inn formalia (felter) som er felles for fører A og B (Skadested, dato, klokkeslett osv.). Og går til neste steg.	
	6. Systemet gir bruker mulighet til å tegne skadesituasjonen.
7. Bruker velger “legg til objekt” på tegningen.	
	8. Systemet viser en liste over alle mulige objekter.
9. Bruker velger et objekt (en bil, et veikryss eller lignende)	
	10. Systemet legger til objektet på tegningen.
11. Bruker velger neste steg eller går til steg 7.	
	12. Systemet viser de felter som skal fylles inn for kjøretøy A (basisinformasjon og supplerende opplysninger)
13. Bruker fyller inn basisinformasjon og supplerende opplysninger for kjøretøy A, og velger så kjøretøy B.	
	14. Systemet viser de felter som skal fylles inn for kjøretøy B (basisinformasjon og supplerende opplysninger)
15. Bruker fyller inn basisinformasjonen og supplerende opplysninger for kjøretøy B	
16. Bruker velger “Send skademelding”	
	17. Systemet kontakter server og sender inn skademeldingen.
18. Bruker får beskjed om at skademeldingen er sendt inn til server.	
Alternativ hendelse 1: Ingen internettilkobling ved forsøk på innsending	
	17. Skademelding lagres lokalt og vises i en liste over skademeldinger som ikke enda er sendt inn
18. Bruker får beskjed om å koble til Internett.	
Alternativ hendelse 2: Bruker avbryter	

underveis	
	Skademeldingen lagres i listen over skademeldinger.
Alternativ hendelse 3: “Egenskade” valgt	
3. Bruker velger “Egenskade”	
	4. Systemet viser skademeldingsskjemaet uten mulighet for å fylle inn for fører B. Fortsetter fra punkt 5. Men punkt 13-15 er ikke tilgjengelige

Inviter motaktør

Use case	**Inviter motaktør**
Aktør	Firmabruker, Privatbruker
Mål	La begge brukere fylle ut sin del av skademeldingen på hver sin mobile enhet
Sammendrag	Aktør A inviterer aktør B til en skademelding ved å oppgi brukernavnet til aktør B og kan starte utfylling av fellesdelen og sin del av skademeldingen. B godtar invitasjonen og starter utfylling av sin del av skademeldingen.
Prekrav	Begge aktørene må være koblet til Internett for å kunne utføre invitasjonen.
Postkrav	
Spesielle krav	Fellesdelen kan bare fylles ut på aktør As telefon.
Hendelsesforløp	
Aktør A	**Systemrespons**
1. Aktør A velger “Ny skademelding”	
	2. Systemet viser en meny der bruker har valg om å invitere motpart.
3. Aktør A velger “inviter motpart”	
	4. Systemet gir aktør A mulighet til å oppgi motaktørens brukernavn.
5. Aktør A fyller inn motaktøren sitt brukernavn og velger “Inviter bruker”.	
	6. Systemet viser en melding om at invitasjon er sendt.
7. Aktør A velger “OK”.	
	8. Systemet viser første siden av skademeldingsskjemaet.
9. Aktør A starter utfylling av fellesdelen og sin del av	

skademeldingen.	
Aktør B	**Systemrespons**
10. Aktør B velger “ny skademelding”	
	11. Systemet viser meny hvor bruker har mulighet til å velge “hent invitasjon”
12. Aktør B velger “hent invitasjon”	
	13. Systemet gir melding om at det finnes en ny invitasjon og at brukeren kan begynne å fylle ut sin del av skademeldingen.
14. Aktør B starter utfylling av sin del av skademeldingen	

2.3 Produktkø

Produktkøen viser hvilken funksjonalitet som skal implementeres i løpet av prosjektperioden. Hvert element i køen har en ID og navn, samt en beskrivelse av hva den går ut på. I tillegg har vi valgt å merke hvilken del av systemet den gjelder. Dette har vi gjort for å lettere få oversikt over hva som for eksempel må være på plass på serveren for å få testet appen skikkelig. I tillegg viser oversikten et estimat på hvor mye tid vi trenger for å fullføre den valgte funksjonaliteten, og et notat for å forklare hva som skal implementeres. Figuren under viser et utdrag fra produktkøen vår. For å se all funksjonalitet som er implementert, se vedlegg H.

ID	Navn	Type	Prioritet	Estimat	Notat
1	Registrere bruker	app	19	21	Enkel registrering, brukernavn/passord
2	Logg inn	app	20	14	Loggin på app
3	Logg inn	web	29	7	Loggin på web
4	Egenskade formalia	app	1	7	Gjelder bare 1 person (lage skjema)
5	Egenskade tilleggsopplysninger	app	2	7	Utfylling av bakside av SM
6	Synlige skader	app	3	1	Markere skader på en "tegning"
7	Situasjonsriss	app	6	14	Tegning av skadesituasjon
8	Første berøringspunkt	app	7	14	markere første berøringspunkt på "tegning"
9	Invitasjon	app	23	40	invitere motpart til å skrive SM
10	Autoutfylling	app	22	5	hente lagret info om bruker fra siste skademelding lagret på telefonen
11	Innsending til server	app	17	5	sende inn SM
12	Geolokasjon av skadested	app			
13	Bilde som vedlegg	app			Kunne ta og legge til bilder i skademeldingen
14	Sette opp database	Server	4	7	

Figur 5: Figuren over viser et utdrag fra produktkøen.

2.4 Sekvensdiagram

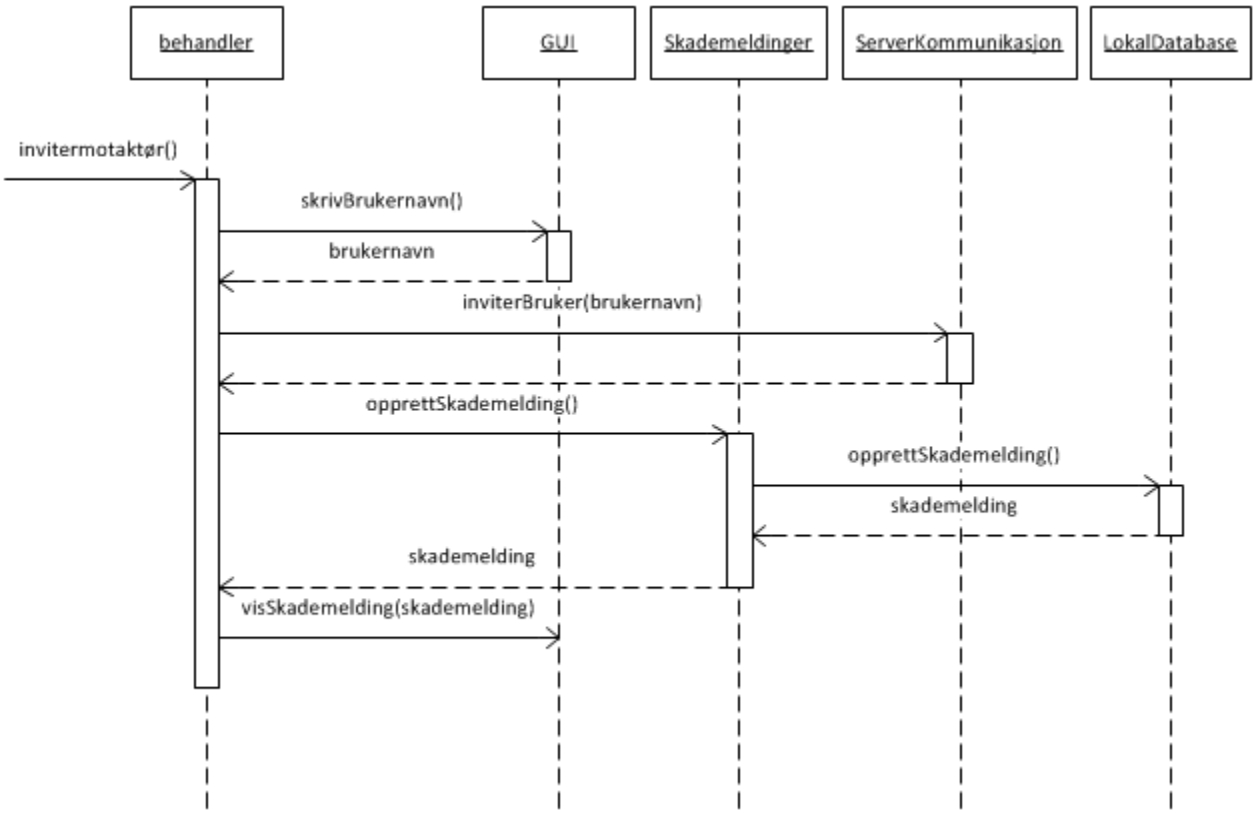
Sekvensdiagram blir ofte benyttet for få fram interaksjonen mellom ulike objekter og dens sekvensielle rekkefølge og når de inntreffer. I følge Donald Bell er disse ikke bare en hjelp for utviklere, men kan også være en hjelp til utenforstående til å forstå hvordan en funksjon i et system skal oppføre seg, eller det kan fungere som dokumentasjon for funksjonen [8].

Vi har i denne oppgaven valgt å benytte oss av sekvensdiagram for å få klarhet i hvordan invitasjonsdelen kan implementeres. Dette vil gi oss verdifull informasjon i hvordan de ulike objektene samarbeider.

Ettersom en invitasjon består av hendelser som skjer på to separate enheter, for eksempel to smarttelefoner har vi valgt å vise dette i to separate sekvensdiagrammer.

2.4.1 Inviter en motaktør

Sekvensdiagrammet under viser hvordan aktør A ønsker å invitere en annen person til sin skademelding. Dette viser hva som skjer i den normale hendelsesflyten på appen.



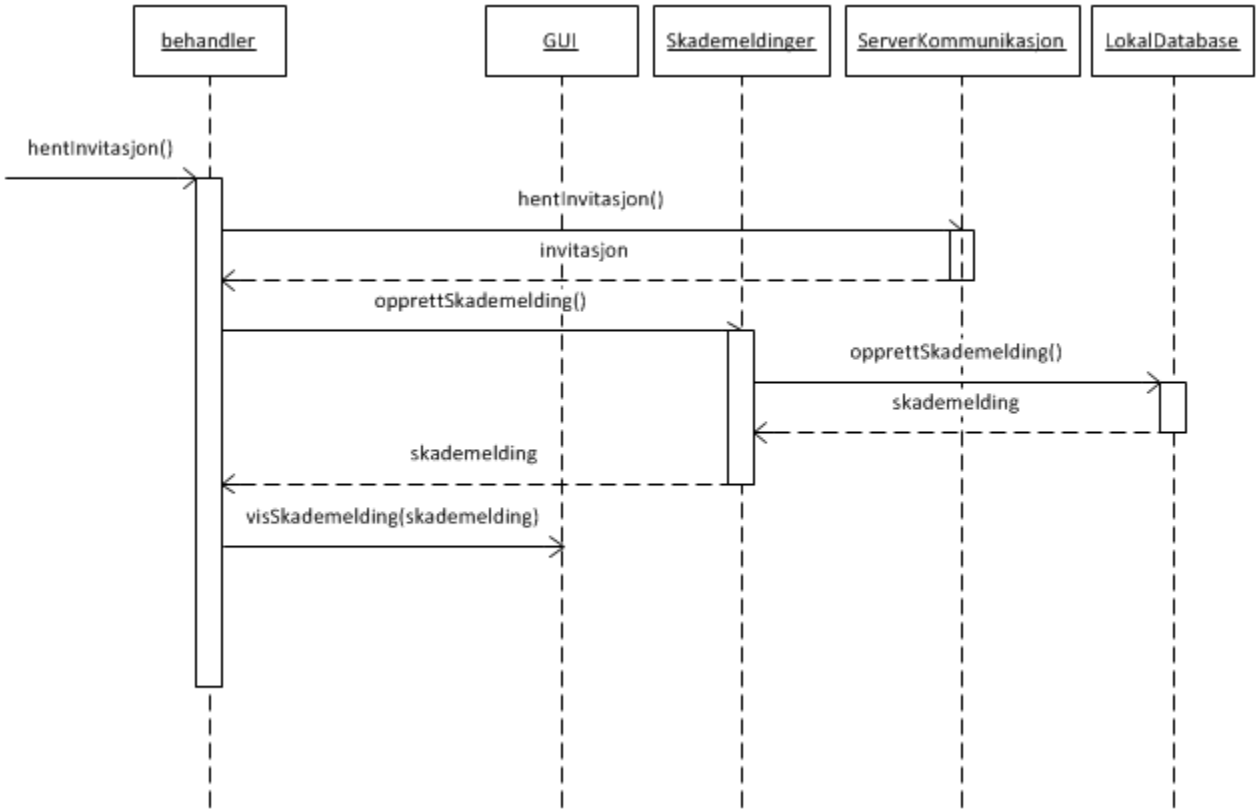
Figur 6: Sekvensdiagram over inviter mottaker

Vi ser at en bruker ønsker å invitere en motaktør (aktør B). For å gjøre dette kreves det at brukeren angir et brukernavn til den han vil invitere. Vi ser også at vi har kontakt med server.

Denne sørger for å opprette en ny skademelding til brukeren og invitere brukeren. I tillegg har vi en lokal database der vi legger til den nye skademeldingen.

2.4.2 Hent en invitasjon

Sekvensdiagrammet for å hente en invitasjon på aktør B sin enhet, viser den normale hendelsesflyten der en aktør A allerede har invitert aktør B. Her er det kun vist det som skjer på appen.



Figur 7: Sekvensdiagram over hent invitasjon

For å hente en invitasjon kreves det kontakt med server som gjør spørringer mot databasen for å se om denne brukeren har noen invitasjoner. Dersom han har dette blir den nyeste returnert og koblet til skademeldingen.

2.5 Supplementærspesifikasjon

2.5.1 Funksjonalitet

Sikkerhet

Brukere av systemet skal autentisere seg med brukernavn og passord for å få tilgang til systemet. Et ønske fra oppdragsgiver var at autentisering på appen skulle være minst mulig forstyrrende for brukeren. De ønsket derfor at brukernavn og passord skulle lagres lokalt på appen for at brukeren skulle slippe å taste inn dette senere, ved for eksempel henting og sending av data.

2.5.2 Anvendelighet

Språk

Systemet skal være på norsk, men skal utvikles med tanke på at det senere skal kunne oversettes til andre språk.

Konfigurerbarhet

Skademeldingsskjemaet skal være konfigurerbart for bedriftskunder. Dette vil da ha innvirkning på hvilke felter firmabrukerne får anledning til å fylle inn i en skademelding. Det vil si at det skal være mulig å tilpasse skademeldingsskjemaet med de punktene som den skadeansvarlige ønsker at bedriftsbrukeren skal fylle ut. Det skal ikke kunne legges til flere felter i skademeldingsskjemaet utover de som allerede finnes.

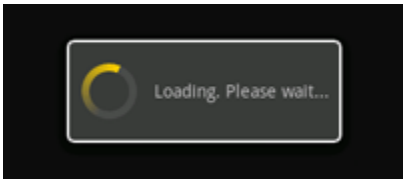
2.5.3 Pålitelighet og ytelse

Oppetid

Oppdragsgiver drifter selv sin egen server og oppgir at forventet oppetid på server er på ca 99 %. Forventet nedetid på server er en dag i løpet av 1 år. Ikke planlagt nedetid vil være mellom 12-18 timer per 1 år.

Responstid

Brukeren skal hele tiden vite hva som skjer i applikasjonen. Dersom det er forventet at en handling kan ta lengre tid enn to sekunder så skal bruker få visuell beskjed om dette ved en indikator.



Figur 8: Eksempel på en beskjed om at systemet arbeider.

Over ser du et eksempel på hvordan en slik indikator kan se ut. I hovedsak vil dette være aktuelt der det er kontakt med server, ved for eksempel invitasjon eller innsending av skademelding. En slik indikator er noe Jakob Nielsen trekker fram i en av sine heuristikker, Visiability of system status [9].

2.5.4 Grensesnitt

Hardware

Serverdelen av systemet skal kjøre på oppdragsgivers servere. Appen skal kunne brukes på både smarttelefoner og nettbrett, men skal spesielt tilpasses smarttelefoner i forhold til skjermstørrelse. Appen kan derfor ikke gjøre bruk av telefonspesifikke funksjoner, slik som for eksempel telefonnummer og SMS. Et minstekrav for enheten som appen skal kjøre på er 500 MHz prosessor og 256 MB minne.

Software

Applikasjonen skal utvikles på en slik måte at den kan kjøres på flere plattformer. De plattformene som er viktige å støtte er de to største Android og iPhone. Men også Windows Phone, som oppdragsgiver forventer blir større etter hvert, skal også kunne støttes.

Webdelen av systemet skal være kjørbar på Internet Explorer 8 og nyere, Chrome og Firefox 4 og nyere.

Design

Oppdragsgiver har en egen designmanual som gir veiledning om hvordan deres produkter skal utformes. Her spesifiseres blant annet utforming av nettside, fargevalg og skrifttyper. Vi vil både på webdelen og appen benytte oss av denne designmanualen for utforming av skjermbildene. I dokumentet er det ikke spesifisert hvordan utforming av en appen bør være. Derfor har vi stått litt friere på denne delen. Men vi har lagt vekt på at elementene skal være gjenkjennbare for brukere som allerede bruker CarAdmin-systemet.

Plattform, utviklingsverktøy, og programmeringsspråk

For serverdelen skal det enten brukes Apache Tomcat og JSP, eller ASP .Net fra Microsoft. Valg av dette vil bli utdypet i kapittel 4. Dersom det velges Tomcat/JSP blir det naturlige valget Eclipse som utviklerverktøy. Dersom ASP .Net velges vil Visual Studio være utviklerverktøy. Lagring på server skal skje i en MySQL-database. Programmeringsspråk vil være avhengig av valg av plattform og utviklerverktøy og behandles også i kapittel 4.

3 Valg og bruk av teknologi

Dette kapittelet tar for seg hvilke teknologier vi har valgt å bruke. Først kommer noen tekniske memoer. I disse vurderer vi ulike teknologier i forhold en spesifikk oppgave eller utfordring som skal løses. Deretter konkluderer vi med et valg av hvilken av de vurderte teknologier vi mener tjener systemet best. Videre tar kapittelet for seg hvilke andre verktøy og teknologier vi har brukt og begrunnelse for hvorfor vi bruker dem.

3.1 Tekniske memoer

De tekniske memoene er vurderinger vi har gjort i forhold til valg av ulike teknologier og løsninger. Disse danner så grunnlaget for hvordan vi har implementert ulike løsninger.

3.1.1 Kryssplattformløsning

Faktorer

Appen skal utvikles for flere plattformer. Her et det viktig at iOS og Android kan støttes, og også Windows Phone. Hvilke muligheter og verktøy finnes for å utvikle en kryssplattform løsning?

Vurdering

Vi har vi sett på ulike rammeverk som kan være aktuelle. Det vi så etter, var rammeverk som gjorde det mulig å skrive koden for appen en gang, får så å distribuere dette til de forskjellige plattformene. Vi har i hovedsak hatt to kandidater; PhoneGap og Appcelerator Titanium.

PhoneGap er et rammeverk som gjør bruk av web-standarder som HTML5, JavaScript og CSS ved utvikling [10]. Denne koden kjøres via et web view som blir lagt i en native-applikasjon. Dette gjør at koden kan være den samme uavhengig av plattform. Gjennom PhoneGap og dets JavaScript API [11] kan vi få vi tilgang til enhetens funksjonaliteter som GPS, WiFi, kamera og lokal lagring. PhoneGap støtter de fleste mobile OS, slik som iOS, Android, Windows Phone, Blackberry, Symbian, Bada og WebOS. Siden koden skrives som en vanlig web-app, kan det meste av koden testes i en vanlig web-browser, og deler kan da også gjenbrukes på web.

PhoneGap er open source og lisensiert under MIT/Apache. En ulempe ved PhoneGap er at det ikke er like raskt som vanlige native-applikasjoner. De gjør for eksempel bruk av CSS for å lage UI, noe som vil kreve mer av maskinvaren enn en rendering av native kontroller. I tillegg får man ikke et utseende som er helt likt det en native app ville ha hatt.

Det andre alternativet for kryssplattformløsning er Appcelerator Titanium . Titanium apper skrives også med JavaScript, men støtter bare Android og iOS. I motsetning til PhoneGap kjøres ikke appen via mobil-browseren, men gjennom en egen oversetter [12]. Måten Appcelerator Titanium fungerer på, er kort sagt at det analyserer og oversetter JavaScript koden til native-

kode (objective-c for iOS, Java for Android). For å utvikle apper med Titanium Mobile SDK, må man bruke Titanium Studio som er et software for utvikling av mobil applikasjoner.

Fordelene med Titanium er at den bruker native-UI slik at den følger utformingen til den enkelte plattformen, noe som kan øke ytelsen. Man kan også få tilgang til all annen native-funksjonalitet på enheten. Bakdelen ved Titanium er at noen av dets API-er er plattformspesifikke, og dette kan gjøre kryssplattform kapasiteten dårligere. Siden appen ikke kjøres gjennom et web view, kan man heller ikke bruke JavaScript biblioteker som for eksempel jQuery etter som de avhenger av at det eksisterer et DOM-objekt.

Av andre alternativer vi har vurdert er Ansca Mobiles Corona [13]. Dette rammeverket har mange av de samme mulighetene som Phonegap og Titanium, men det er mer rettet mot utvikling av spill. Samtidig koster det penger noe som gjør at det vurderes som uaktuelt [14].

Det siste alternativet vi har vurdert er Rhodes [15]. Dette rammeverket baserer seg på HTML og Ruby. Rhodes kompilerer koden til native app-er. Rhodes har GPL lisens for brukere som utvikler open source. For bedrifter er det lisensiert under en kommersiell lisens som betyr at man må betale en avgift for bruken. I tillegg har ingen av oss noen erfaring med Ruby, så dette ble også uaktuelt.

Løsning

Valg av løsning falt på PhoneGap som rammeverk for utvikling av vår app. Grunnen til dette er hovedsakelig at oppdragsgiver ønsket en løsning som skal kunne brukes på Windows Phone i tillegg til Android og iOS. Dette ville ikke vært mulig med Titanium siden det ennå ikke støtter Windows Phone, noe de vil gjøre i framtiden [16]. I tillegg vil utviklingen foregå i programmeringsspråk vi har noe kjennskap til. En annen viktig grunn til at valget falt på PhoneGap, er at vi i tillegg skal utvikle en web-side som skal inneholde mye av den samme funksjonaliteten som applikasjonen. Dette gjør at vi kan gjenbruke kode. Og derfor mener vi at dette er det beste alternativet.

En grunn til at vi ikke ville velge Appcellerator Titanium er at vi har sett flere kilder som nevner problemer med minnehåndteringen til dette rammeverket, og at dette fører til at applikasjonen krasjer[17]. Ved å bruke PhoneGap kan vi også bruke Eclipse for all koding både for web, server og app.

3.1.2 Server og web-teknologi

Faktorer

I tillegg til en app skal systemet skal bestå av en serverdel som appen kan kommunisere med, og en webside. Hvilke teknologier er aktuelle, og hvilken teknologi bør vi velge?

Vurdering

Fra oppdragsgiver har vi fått oppgitt at vi må bruke enten JSP/Java Servlets med Apache Tomcat eller ASP.NET. Derfor vil disse være til vurdering.

Tomcat som utvikles av Apache Software Fundation er en open source implementering av Java Servlet og JavaServer Pages(JSP) og gjør at du kan kjøre Java-baserte webapplikasjoner og JSP-sider[18]. JSP-sider skrives med HTML, og Java-kode innenfor <% %>. Oppdragsgiver bruker selv denne teknologien og er godt kjent med den.

ASP.NET er et webapplikasjonsrammeverk utviklet av Microsoft og har muligheter for å utvikle dynamiske websider, webapplikasjoner og webtjenester[19]. Oppdragsgiver har lite kjennskap til denne teknologien.

Løsning

I hovedsak er begge teknologiene likeverdige i forhold til det vi skal bruke dem til. Men for vår del vil det være fornuftig å gå for Tomcat som serverteknologi. Først og fremst fordi dette gir oss muligheten til å benytte oss av Eclipse som IDE og at vi dermed slipper å forholde oss til Visual Studio i tillegg. Samtidig vil det være enklere for oppdragsgiver å hjelpe oss underveis.

3.1.3 Visuell beskrivelse av skadesituasjonen

Faktorer

I en del av skademeldingen skal det brukeren kunne lage en visuell beskrivelse av skadesituasjonen. Det skal kunne legges til, flytte og endre ferdige objekter av forskjellige typer. Frihåndstegning er ikke noe som er foretrukket. Vi skal utvikle denne delen, og gjøre bruk av biblioteker der dette er nødvendig.

Vurdering

Fordi vi gjør bruk av HTML5-kode gir det oss valget mellom å bruke et canvas-element eller et SVG-element. Disse to elementene er kraftige, men ganske enkle i seg selv. Ved bruk av ekstrabiblioteker er det mulig å få svært mye funksjonalitet ut av dem.

I hovedsak har det vært tre aktuelle JavaScript-biblioteker som har vært aktuelle, alle har støtte for touch og dragAndDrop som vi er avhengig av. Alle benytter seg av MIT-lisens.

oCanvas er et bibliotek som gjør bruk av Canvas-elementet[20]. Dette er veldig enkelt i bruk men er tregt fordi det rendrer hele canvas-en ved endringer.

KineticJS er et annet bibliotek og benytter seg også av Canvas-elementet. Den største fordelen med KineticJS er hastigheten, noe de reklamerer med på sidene sine[21]. Etter ulike tester har vi selv erfart at KineticJS kjører flytende på en smarttelefon med bare 500 MHz prosessor. Med de samme objektene i oCanvas var alt mye mindre responsivt.

Grunnen til dette er at KineticJS benytter seg av flere Canvas oppå hverandre og rendrer kun de delene som endres, mens oCanvas rendrer hele Canvasen hver gang det skjer noe. KineticJS bruker ulike lag, som buffer-layer og backstage-layer for å sikre at hastigheten er god. I tillegg til standard layer-ene kan det også defineres egne layers etter ønske.

Raphaël er en annen type JavaScript-bibliotek som benytter seg av SVG og benytter seg derfor av vektorgrafikk[22]. Dette er interessant i forhold til de forholdsvis enkle tegningene vi skal utforme. Det vil også gjøre det enklere å lage nye objekter i forhold til i de andre bibliotekene der objektene enten må legges inn som bilder, eller hardkodes. Hvert objekt du oppretter i Raphaël blir lagt til i DOM-en som et eget DOM-objekt, dette gjør at DOM-en etter hvert kan bli stor. Noe som kan føre til tregere rendering og dårligere ytelse.

Løsning
Vi benytter oss av KineticJS der brukeren skal lage visuelle tegninger. Hovedgrunnen til dette er at det er svært lettkjørt i forhold til de andre alternativene. Dette gjelder også på smarttelefoner med liten prosessor.

Da vi gjorde valget om å bruke KineticJS var dette et ganske ungt bibliotek. Og siden vi startet har det utviklet seg mye. Hadde vi startet på oppgaven i dag kunne vi spart noe tid ettersom det har fått flere funksjoner som vi har skrevet selv. For eksempel er det kommet en toJSON() funksjon som gjør om alle objektene til en JSON-string. Dette er en funksjon som vi har laget selv.

3.2 Utviklingsverktøy - Eclipse

Både for appen og for serverdelen har vi brukt Eclipse Indigo som utviklingsverktøy. Eclipse er et open source samfunn hvor prosjektene dreier seg om å utvikle en åpen utviklingsplattform bestående av en rekke rammeverk, verktøy og "runtimes" for utvikling, styring og distribusjon av programvare[23]. Eclipse er et mye brukt verktøy, ikke bare i undervisningssammenheng, men også ute blant bedrifter.

I vårt prosjekt har vi brukt verktøy (plug-ins) for web-utvikling, Android utviklingsverktøy, SVN og i tillegg til pakker for Java EE og web-applikasjoner. Disse hjelpeverktøyene inneholder funksjonalitet for å effektivisere arbeidet, blant annet autofullføring av kode, markering av feil i kode, forslag til funksjoner man kan bruke i koden, veivisere for oppsett av prosjekter, og API-er for å bistå med distribusjon, kjøring og testing av applikasjoner. [2]

[2] http://en.wikipedia.org/wiki/Eclipse_%28software%29

De største utfordringene vi har hatt i forhold til Eclipse handler om installasjon av SVN og koblingen til Tomcat-server. Sett bort i fra dette, er vi godt fornøyd med mulighetene Eclipse tilbyr for å hjelpe til med utviklingen av programvare.

Et alternativ til Eclipse som ble vurdert, var NetBeans. Dette er også et IDE for utvikling av programvare i en rekke språk med muligheter for utvidelser på samme måte som Eclipse. Men siden Eclipse er noe begge gruppemedlemmene hadde jobbet en del med i tidligere gjennom utdanningen, ble dette et naturlig valg. I tillegg bruker oppdragsgiver Eclipse til utvikling, noe vi ser på som en stor fordel.

3.3 Versjonskontroll

For versjonsstyring har vi benyttet HiGs SVN-server. SVN har vært et viktig hjelpemiddel for gruppen for å holde orden på koden i prosjektet. Ved å benytte SVN har vi kunne lagre alle filer, og versjoner av filene, på en ekstern server, og på denne måte sikre at overskriving av hverandres kode ikke finner sted.

Et av problemene oppstod når vi skulle laste ned og installere Subversive, som er en SVN-klient. Problemet førte til mye feilsøking og tap av tid og det endte med at et av gruppemedlemmene ble nødt til å laste ned et alternativt plug-in kalt Subclipse.

3.4 PhoneGap

For å bruke PhoneGap til å utvikle applikasjoner for Android må man bruke Eclipse med installerte Android-plugins (Android SDK og ADT) i tillegg til selve PhoneGap-biblioteket. Selve prosjektet i Eclipse opprettes som et nytt Android prosjekt, men man må gjøre noen små endringer for å få det til å kjøre med PhoneGap. Hovedsakelig kobler man en HTML-side som blir kjørt av Java-koden.

App.java er den filen i et Android prosjekt som inneholder startmetoden for applikasjonen. Under er vist hvordan index.html kalles når PhoneGap benyttes.

```
package com.skademelding.app;

import android.os.Bundle;
import com.phonegap.*;/
/**"extends DroidGap" er det som gjør appen til "ikke-native"/PhoneGap app */
public class App extends DroidGap {
  /** Denne funksjonen kjøres når appen starter.
  @Override
  public void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
      /** linjen under loader startsiden for appen */
     super.loadUrl("file:///android_asset/www/index.html");
```

```
    }
}
```

3.5 JavaScript

JavaScript er et kraftig og dynamisk objektorientert språk. Det har vært mye brukt innen web-programmering. De senere årene har bruken av det endret seg ved at vi nå ser at det utvikles hele applikasjoner som baserer seg på JavaScript. Dette er mulig fordi JavaScript-motorene er blitt svært mye raskere de siste årene. Språket har mange avanserte funksjoner, som for eksempel prototyping, arv og namespaces.

For å få en klasse struktur i vår app har vi valgt å benytte oss av en anbefaling fra boken til Alex MacCaw og vil gjøre bruk av prototyping[24]. Her blir det opprettet en funksjon som fungerer som en klasse som det kan legges metoder til. Et eksempel på dette er vist under.

```
function Skademerking()
{
    //Kjører super (konstruktor i Tegnebrett)
    Tegnebrett.apply(this, ["skademerkingsdiv"]);
}

Skademerking.prototype.leggInnStandardObjekter = function(){
    //Denne metoden er en metode i klassen Skademerking
}

//Skademerking arver fra Tegnebrett
extend(Skademerking, Tegnebrett);
```

Vi ser også at denne klassen arver fra Tegnebrettklassen. Det gjøres med metoden `extend()`. Denne metoden er vist under og finnes i filen hjelpemetoder.js.

```
/**
 * Metode for å arve fra en klasse
 * @param obj1 Det objektet som skal få obj2 sine egenskaper.
 * @param obj2 Det objektet som skal arves fra
 */
function extend(obj1, obj2){
     //Her kjøres en løkke for å legge prototypene fra obj2 til obj1
     //slik at det i praksis fungerer som arv, slik det
     //ville gjort i for eksempel Java eller C#
     for (var key in obj2.prototype) {
            if (obj2.prototype.hasOwnProperty(key)) {
                obj1.prototype[key] = obj2.prototype[key];
            }
        }
}
```

3.6 jQuery og jQuery Mobile

jQuery er et JavaScript bibliotek utviklet for å gjøre klientside-scripting av HTML lettere. Dette er det mest populære JavaScript-biblioteket som er i bruk i dag, og statistikken viser at over 55 % av de 10 000 mest besøkte sidene på nettet bruker dette biblioteket[25]. På grunn av dette og at det vil hjelpe oppdragsgiver fordi de kan bruke et kjent verktøy, har vi valgt å benytte oss av det. I tillegg vil vi bruke jQuery Mobile som er en forlenget arm av jQuery og et rammeverk for bruk på mobile enheter. Dette gjør at sideoppsett og GUI kan håndteres på en enkel måte[26].

JQuery Mobile rammeverket gjør det mulig å ha flere sider, kalt en page, i ett html-dokument[3]. Dette kan sees i eksempelet under som er hentet fra index-siden til appen.

```
<html>
<body>
<!-- Start på index-side (den første siden som blir vist når man starter
appen)-->
<div data-role="page" id="index"> <!-- spesifiserer at denne div'en er en
side med id="index"-->

  <div data-role="content" id="loginContent">
 <!-- form for bruker-input  -->
       <div id="login">
           <h4>Eksisterende bruker</h4>
      Brukernavn/e-post
       <input type='text' class="login" name="loginBruker" id="loginBruker"/>
      Passord
       <input type='password' class="login" name="loginPass" id="loginPass"/>
           <a id="loginBrukerKnapp"  data-role='button'>Logg inn</a>
       </div>
       <br/><hr/><br/>
<!-- Link til siden for å registrere ny bruker settes ved å referere til
sidens id (istedet for en url)-->
           <a href="#registrerBruker" data-role="button">Registrer ny
bruker</a>
   </div>
</div> <!-- Slutt på index-siden -->

<!-- Start på "registrer bruker"-siden (siden som ble linket til i #index) --
>
<div data-role="page" id="registrerBruker" >
<!-- form for registering av ny bruker -->
   <div id="reg">
       <h4>Ny bruker</h4>
       E-post
       <input type='text' class="registrer" name="regBruker" id="regBruker"/>
       Passord
       <input type='password' class="registrer" name="regPass" id="regPass"/>
       Gjenta passord
       <input type='password' class="registerer" name="regPass2"
id="regPass2"/>
```

[3] http://jquerymobile.com/demos/1.1.0/docs/pages/page-anatomy.html

```
        <a id="regBrukerKnapp"  data-role='button'>Registrer</a>
        <br/><hr/><br/>
<!-- link tilbake til index-siden -->
        <a href="#index" id="cancelReg" data-role="button">Avbryt</a>
    </div>
</div><!-- slutt på "registrer bruker"-side -->
</body>
</html>
```

jQuery Mobile har den ulempen at det krever en del prosessorkraft. Dette merket vi oss da vi kjørte det på en av testtelefonene våre som har en prosessor på 500 MHz. Her virket appen mindre responsiv enn ved en raskere telefon. Dette ble diskutert med oppdragsgiver som mente at det ikke ville være noe problem, ettersom disse telefonene snart ville bli faset ut. jQuery Mobile var veldig lett å implementere, og siden syntaksen fra jQuery var noe vi var kjent med fra før, gikk det raskt å forstå hvordan man bruker dette rammeverket.

Siden vi hadde denne usikkerheten rundt ytelsen ble også Sencha Touch vurdert som et alternativ. Dette er også et rammeverk for utvikling av mobile applikasjoner som også baserer seg på HTML5, CSS3 og Javascript. Sencha Touch tilbyr mye av det samme som jQuery Mobile, inkludert GUI-komponenter, touch-event håndtering, og AJAX-funksjonalitet. Sencha Touch støtter ikke Windows Phone og var da ikke aktuell. I tillegg er jQuery Mobile godt dokumentert og lett å integrere med PhoneGap.

4 Design

I dette kapittelet skal vi se nærmere på hvordan systemet er bygd opp og hva som er gjort for at systemet skal ha best mulig struktur og oversiktlighet.

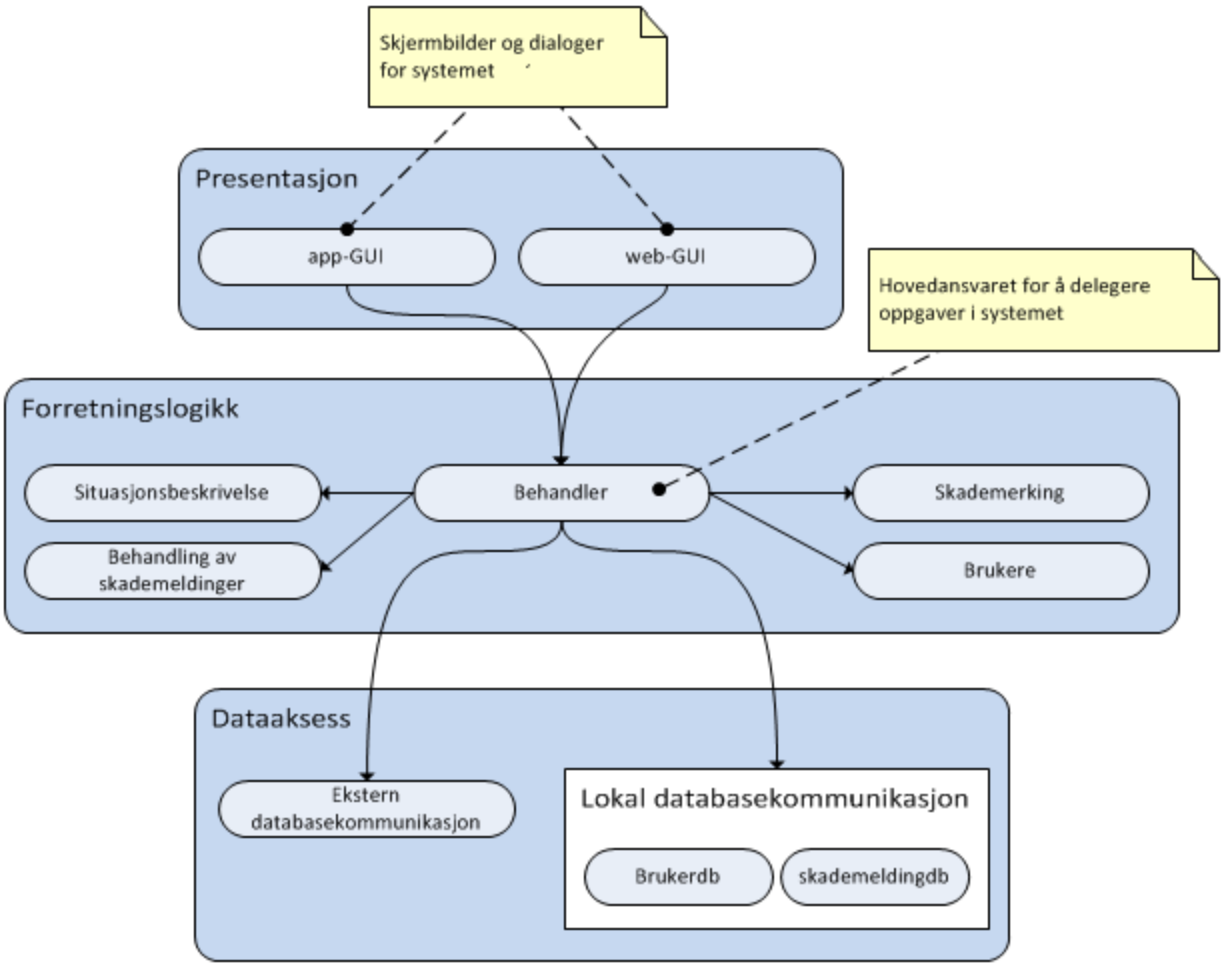
4.1 Logisk oversikt

4.1.1 Lagdeling

Valg av arkitektur vil avhenge av hva systemet skal utføre og hvilke krav som stilles til den funksjonaliteten som skal utvikles. En riktig arkitektur bidrar til å forenkle utvikling og vedlikehold av systemet, samtidig som det gir mindre sjanse for feil ettersom det gir en mer oversiktlig struktur.

Mye brukte arkitekturer i dag er trelagsarkitektur eller Model View Control (MVC). I følge Wikipedia vil den førstnevnte arkitekturen deles opp med et presentasjonslag, et lag for foretningslogikk og et for dataaksess. I en trelagsarkitektur vil kommunikasjonen gå fra det øverste laget, gjennom det midterste til det nederste [27]. Sun beskriver i en artikkel hvordan MVC fungerer, også at det her er delt opp i tre lag. Men her er det et modellag, et presentasjonslag og et kontrollag. Her vil presentasjonen oppdatere kontrolleren, kontrolleren oppdatere modellen, og presentasjonen få oppdateringer fra kontrolleren [28]

I dette systemet har vi valgt å følge oppdragsgivers anbefaling ved valg av arkitektur. Det betyr at systemet er bygd opp rundt en trelagsarktitektur med et lag for presentasjon, et for foretningslogikk og et for dataaksess.



Figur 9: Systemets lagdeling

Presentasjon

Presentasjonslaget håndterer all interaksjon med brukeren og innebærer blant annet navigasjon og utfylling av felter. Dette laget er delt inn i to pakker, GUI for web og GUI for app. Kodemessig vil all HTML-koden og CSS-koden gå under presentasjonslaget.

Forretningslogikk

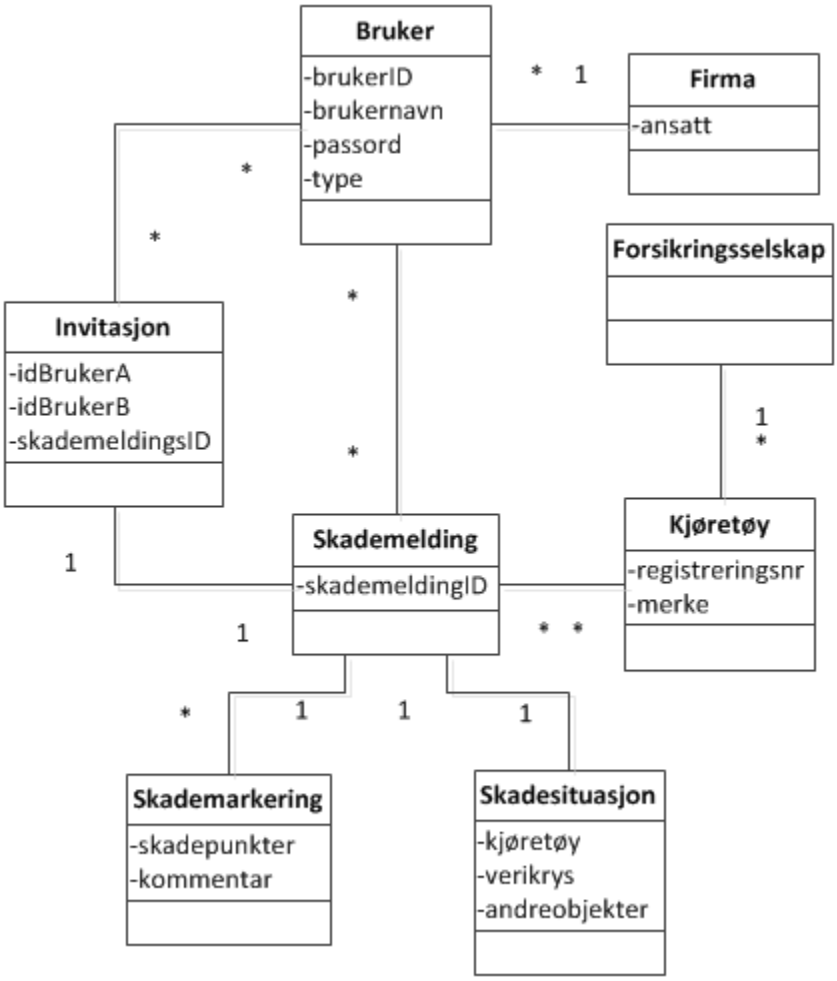
Dette laget kan på mange måter være det viktigste i systemet. Her finnes selve logikken for det som skal utføres. Vi finner en pakke, behandler, som tar seg av koblingen mellom laget over og laget under. Forretningslogikken vil for det meste bestå av JavaScript-kode for appen, for webdelen vil det i tillegg bli benyttet JSP.

Dataaksess

Det siste laget tar for seg all kommunikasjon med databasen og sørger for at dataene blir behandlet slik at dataintegriteten blir ivaretatt. I vårt system er kontakten med server slik at data hentes og lagres. I tillegg er det også lokal lagring av data på appen.

4.1.2 Domenemodell

Domenemodellen under viser konsepter i systemet og forholdet mellom dem.



Figur 10: Systemets domenemodell

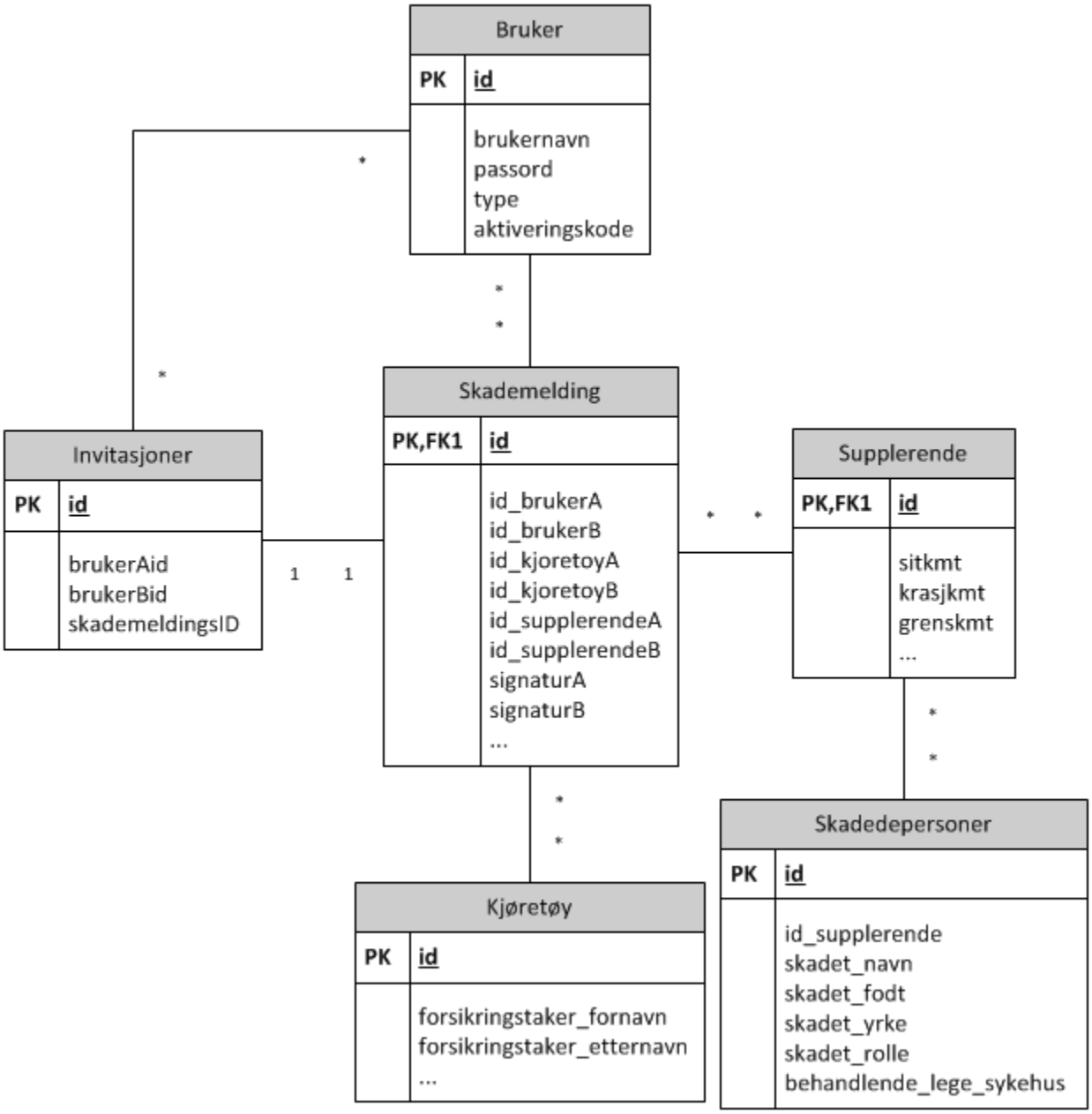
Vi ser også at en skademelding kan høre til mange brukere. I praksis vil dette si en eller to brukere ettersom det kan være to involverte i en skademelding. Siden en skademelding kan finnes både på databasen til appen og på serveren kan det være verdt å merke seg at skademeldingID refererer til IDen som skademeldingen har på serveren.

4.1.3 Lagring av data

Ved lagring av data er det viktig å ta vare på dataintegriteten slik at dataene ikke endres eller ødelegges uten at det var brukerens hensikt[29]. Vi har lagring av data både på appen og på serveren. På appen er det kun en enkel database med to tabeller. En for brukere og en for skademeldinger. Skademeldingene blir her lagret som en JSON, noe som betyr at dersom det

skal hentes noe ut fra en skademelding på appen, må hele skademeldingen hentes ut. Dette er ikke noe problem ettersom lagringen på appen kun fungerer som en mellomlagring slik at vi hele tiden slipper å hente dataene fra server. Dette gjør at vi minsker datatrafikken.

På serveren derimot er dataene lagret i en logisk relasjonsdatabase inndelt i ulike tabeller. Denne er utformet på en slik måte et vi får minst mulig redundans av data.



Figur 11: Databasemodell

I databasemodellen over ser du forholdet til de forskjellige tabellene i databasen. Den består av til sammen seks tabeller, Skademelding, kjøretøy, supplerende, skadedepersoner, invitasjoner og bruker.

Tabellen Skademelding representerer selve skademeldingen. Vi ser at denne er koblet opp mot to brukere, id_brukerA, id_brukerB. For hver skademelding vil det være to koblinger mot en Kjøretøytabellen og mot Supplerende-tabellen, en for hver av brukerne. Kjøretøytabellen er den som inneholder basisinformasjonen til førerne.

En bruker er enten av typen privatbruker, firmabruker eller skadeansvarlig. Alle som registrerer seg på appen blir automatisk privatbruker. Dette betyr at firmabruker og skadeansvarlig må settes opp manuelt av ETC ettersom disse brukerne kommer fra deres database.

Invitasjonstabellen hjelper oss å holde orden på hvilke brukere som er invitert til de forskjellige skademeldingene og ennå ikke har svart på invitasjonen. En slik invitasjon kunne ha vært implementert direkte inn i Skademeldingstabellen, men det vil være hensiktsmessig å bruke en egen tabell fordi det vil være mindre ressurskrevende å kjøre en spørring mot en tabell som vil kun ha noen få rader. Samtidig sikrer vi oss at en bruker ikke blir koblet til en skademelding før han faktisk godtar invitasjonen.

I flere av tabellene er det svært mange felter. Vi har derfor valgt å bare vise de viktigste i databasemodellen. En fullstendig oversikt over alle felter og hvordan databasen er satt opp kan sees i vedlegg D.

Vi har benyttet oss av en forholdsvis lite databaselogikk. Men har laget SQL-setninger som inkluderer JOIN og sorteringer.

4.2 Design av brukergrensesnitt

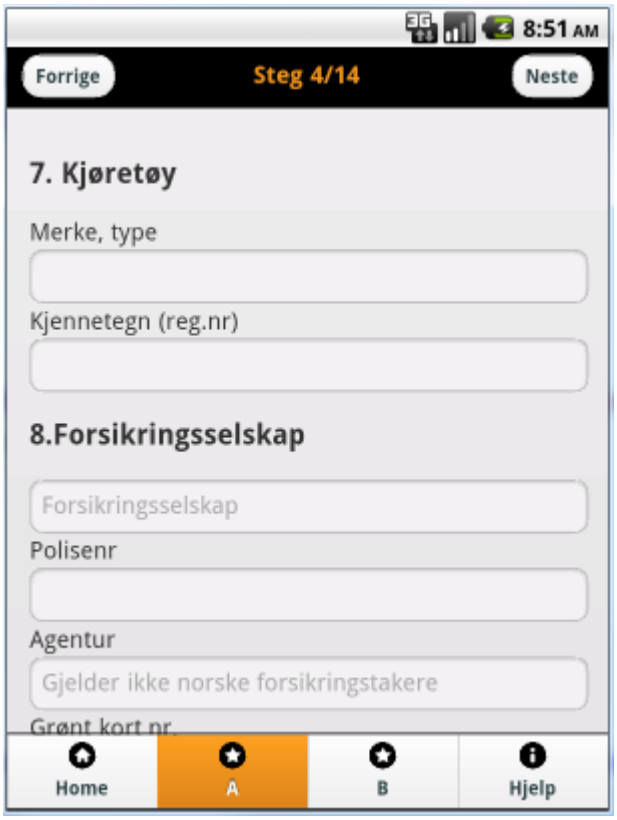
Brukergrensesnittet er designet for å være mest mulig oversiktlig og for at anvendelsen av systemet skal være intuitivt for brukeren. For å gjøre systemet brukervennlig har vi forsøkt å følge Nielsens heuristikker for design av brukergrensesnitt [9].

4.2.1 App

For å utforme brukergrensesnittet på appen har vi brukt de GUI-elementene jQuery Mobile rammeverket tilbyr. Dette rammeverket inneholder en mengde knapper, verktøylinjer og form-elementer som er utformet for å passe til håndholdte enheter med ulik skjermstørrelse. Vi vurderte i starten av prosjektet om vi skulle skrive CSS for brukergrensesnittet selv, slik at

appen skulle ligne mer på CarAdmins grensesnitt. Siden det var en app vi skulle utvikle, mente vi det var et poeng at den også skulle ha brukergrensesnitt som så ut som en app. Dette ble diskutert med oppdragsgiver, som sa seg enig i at jQuery Mobile kunne benyttes.

Måten vi har satt opp skjema-sidene i appen på er at man har en header-meny med to knapper for å navigere seg frem og tilbake. Man kan også navigere mellom sider ved å swipe høyre eller venstre. Headeren viser også en overskrift som sier hvilken side man er på (for eksempel 3/14). Ved å trykke på menyknappen på enheten kan brukeren selv kan velge når denne footer-menyen skal vises eller skjules, dette for å unngå at den tar opp plass på skjermen og kommer i veien for andre elementer på siden. På iPhone og andre telefoner hvor det ikke finnes noen fysisk menyknapp måtte man ha implementert dette på en annen måte, for eksempel ved at brukeren trykker et sted på skjermen. Under ser du et eksempel på hvordan brukergrensesnittet er utformet.



Figur 12: Brukergrensesnitt, header og footer

Footer-menyen inneholder ulike valg man kan gjøre på siden. Disse valgene vil endre seg og vise forskjellige knapper ettersom hvilken side i skjemaet man er på.

Knappene i menyene er utformet slik at de skal være lette å trykke på, og viser en tekst og/eller et ikon for å gi brukeren et hint om hvilken funksjonalitet knappen har. I skjemaet har vi brukt radio-knapper på de punktene hvor brukeren bare skal krysse av for ett valg (se vedlegg B), og checkbokser der brukeren kan krysse av flere alternativ. På denne måten hindres brukeren fra å gjøre feil i utfyllingen.

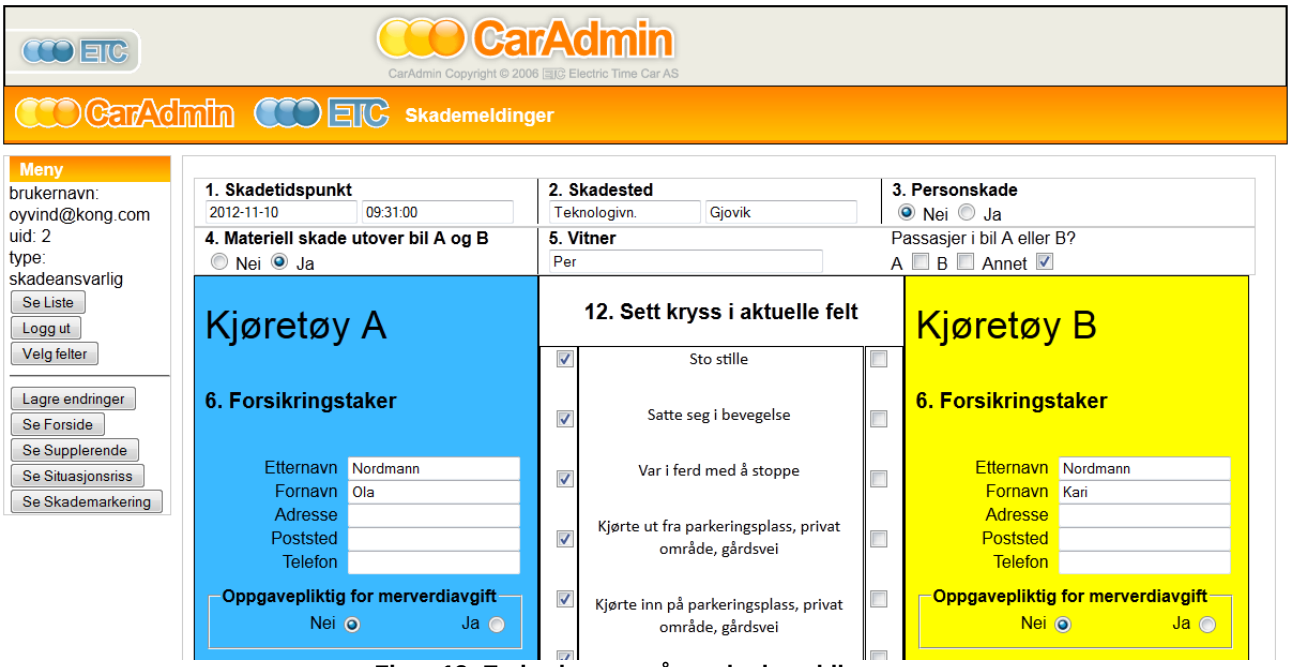
På felter der brukeren må taste inn dato eller tidspunkt har vi brukt et JavaScript-bibliotek, slik at denne informasjonen ikke må skrives inn som tekst. Dette gjør input av dato/tidspunkt intuitivt for brukeren, og sikrer at denne informasjonen blir lagret på riktig format. Bilde av datovelgeren kan sees i vedlegget E.

Når det gjelder tegnemodulen har vi lagt vekt på at tegningene skal være så enkle som mulig slik at det blir lettere for brukeren å orientere seg. Siden tegningen skal foregå på en liten skjerm kan det fort bli vanskelig å holde oversikten hvis det blir for mange detaljer. Bil-objektene i situasjonsrisset er markert med en bokstav (A/B) for å skille mellom de involverte bilene, i tillegg til at de har forskjellig farge(blå/gul). For at brukeren til en hver tid skal vite hvilke objekt som er valgt, vil det siste objektet som ble trykket på bli markert med en egen farge. Det samme gjelder på skademarkeringen.

Farge-temaet vi har brukt i applikasjonen er til dels det samme som brukes i CarAdmin og ETC's nettsider, slik at brukerne skal kjenne seg igjen.

4.2.2 Web

Brukergrensesnittet på web er designet på samme måte som ETC sine nettsider, ved at man har en banner øverst, en meny til venstre og at innholdet vises på midten av siden. Vi har her benyttet oppdragsgivers mal for utforming av nettsider, og brukt noe av de fargekoder og bilder som vi fikk tilsendt av ETC. Tegnemodulen er den samme som på app, bortsett fra menyen som her ligger over selve tegnebrettet.



Figur 13: En bruker ser på en skademelding

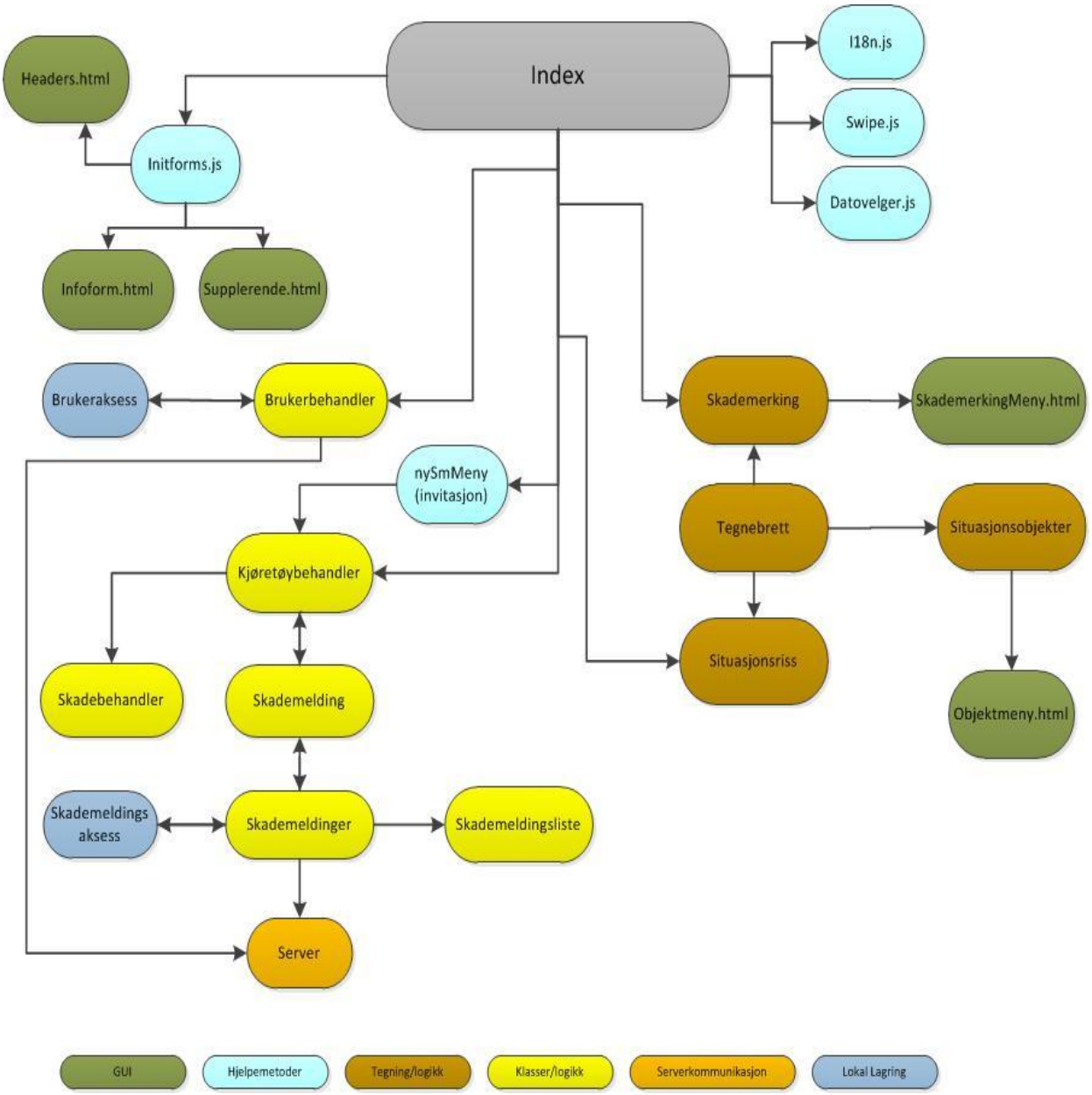
Visningen av skademeldingen på web er utformet for å ligne mest mulig på papir-utgaven av skademeldingen, se vedlegg B. Dette er gjort for at det skal være lettere for brukeren å forstå hva som vises på skjermen, siden dette skjemaet er noe som bør være kjent for bilførere.

5 Implementering

I dette kapittelet skal vi ta for oss hvordan vi har implementert de ulike delene av systemet. Først kommer vi til å se på hovedstrukturen i appen for så å se hvordan funksjonaliteten er implementert. Videre vil vi ta for oss implementasjonen av web-siden for så å komme inn på server-funksjonalitet, som hvordan vi legger inn skademeldinger i databasen.

5.1 Struktur for appen

Figuren under viser forholdet mellom de ulike filene i appen og hvordan dette er strukturert. Boksene i figuren representerer filene og er markert med forskjellige farger i forhold til hva filen brukes til. Pilene viser hvilke filer som er knyttet sammen.



Figur 14: Struktur av appen

De ulike delene vi kommer til å gå nærmer inn på er index, initforms med tilknyttede GUI-filer, Brukerbehandler, Kjøretøybehandler, de to filene for lokal lagring, filene for tegne-modulen, I18N.js, Server, nySmMeny (invitasjon), Skademeldingsliste og Skademeldinger

5.2 Serverkommunikasjon

Ettersom systemet består av forskjellige deler som skal kunne brukes på ulike enheter vil det kreves kommunikasjon mellom disse. Denne kommunikasjonen vil foregå via en server som innehar en database. Dette vil si at appen kommuniserer med server gjennom servlet-er. Disse servlet-ene behandler videre de dataene som mottas og sender tilbake. Mellom webdelen, som består av JSP-filer, og server kommuniseres det med servlets og direkte med databasen.

I utgangspunktet var det et ønske fra oppdragsgiver at kommunikasjonen skulle foregå over HTTPS for å sikre dataene. Det viste seg at det var problematisk i henhold til sertifikater å gjøre dette gjennom PhoneGap[30]. Det ble derfor besluttet å benytte seg av HTTP. I praksis har vi benyttet oss av jQuerys AJAX-metoder for å sende forespørsler til server fra appen. Et eksempel på dette kan sees under.

```
$.ajax({
   type: "GET",
   timeout: 10000, //Hvor lang timout-tiden er
   url : 'http://localhost:8080/skademeldingjsp/invitasjonsjekk?',
   dataType: "json", //Type data som sendes
   success  : function(data, b, c) {
    //Hvis vi mottok svar fra server og alt var OK
   },
   error: function (xhr, ajaxOptions, thrownError){
      //Error-metode
   }
});
```

5.3 Implementasjon av app

5.3.1 Oppstart og initialisering

Index.html inneholder alle sidene (skjermbildene, page-ene) vi trenger i applikasjonen, og siden det er denne filen som blir lastet først når applikasjonen starter er denne en oppstart for hele programmet. Alle JavaScript-filer som brukes er det referanse til Index.html.

Sidene for selve skademeldingsskjemaet er i utgangspunktet tomme når appen startes. Disse sidene får innhold fra de to filene infoform.html og supplerende.html. Disse to filene inneholder alle felter som skademeldingen består av. Headers og footers for alle sidene blir lastet inn fra headers.html. Alt dette skjer ved et kall til `initForms()`.

Vi valgte å laste inn skademeldingsfeltene fra egne filer for at index-fila ikke skulle vært så stor og tung å lese i starten.

5.3.2 Brukerbehandling

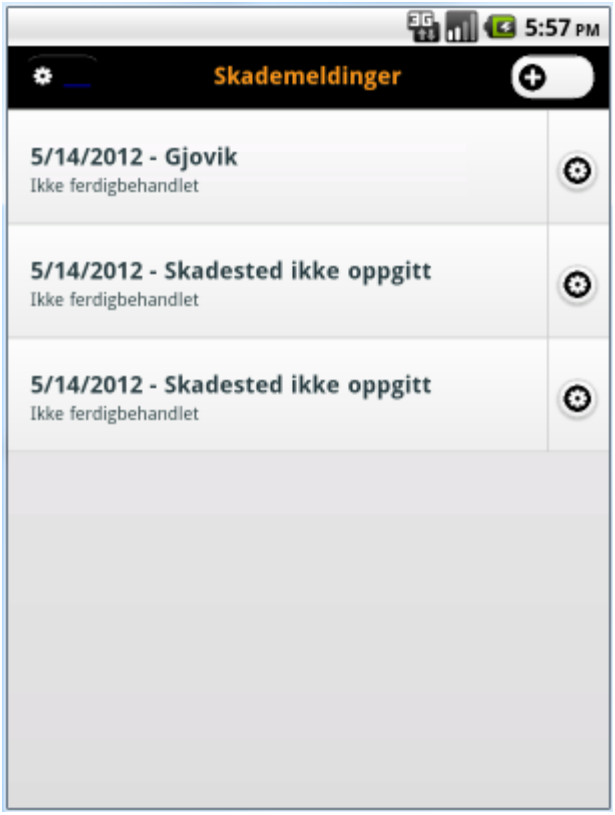
Klassen brukerbehandler tar seg av alt som har med registrering og innlogging av brukeren å gjøre. Ved registrering tar den vare på brukernavn (e-post) og passord fra brukeren, videre sender disse dataene til server som lagrer dette som en ny bruker i databasen. Brukernavnet og passordet blir også lagret lokalt på telefonen slik at brukeren slipper å skrive inn dette hver gang han starter appen, men i stedet blir logget rett inn. Sjekk mot server blir da ikke gjort. Ved utlogging blir denne informasjonen slettet fra lokal database, og man må skrive inn brukernavn og passord eller registrere ny bruker for å få tilgang til appen igjen.

Å lagre passord vil alltid være en utfordring og vil føre til en risiko for at det kommer på avveie. I følge Android Dev Guide bør man unngå dette ved å heller lagre en session som sier at brukeren har autentisert seg [31]. Det er først når den håndholdte enheten kommer i hendene på andre at de kan utnytte dette. Konsekvensen av dette kan bli at andre kan få se de skademeldingene som er fylt ut på telefonen, men dersom passord endres på web vil det ikke kunne bli gjort noen skade i systemet. Dermed blir ikke konsekvensene veldig store. Vi har derfor valgt å følge oppdragsgivers ønske om at brukeren bare skriver inn brukernavn og passord ved første pålogging. For å skille mellom jobb og privat må det opprettes egen konto for privat bruk, der brukeren må logge ut fra den ene og inn på den andre.

5.3.3 Skademeldingsliste

På den første skjermen som vises etter innlogging er det en liste over dine skademeldinger. Hvert element i denne listen har en dato for skadetidspunktet, stedet hvor ulykken foregikk, og status på om skademeldingen er ferdigbehandlet eller ikke. Listen er sortert etter datoen, med de nyeste skademeldingene øverst, selve denne sorteringen skjer i SQL.

Ved å trykke på et element i listen blir skademeldingen vist for brukeren slik at han kan gjøre endringer på den. Elementene inneholder også en id til skademeldingen i den lokale databasen og en link til den første siden i skademeldingsskjemaet. Når brukeren trykker på en skademelding i listen vil det bli gjort et kall til skademeldinger som henter data fra databasen og oppretter et nytt objekt av klassen Skademelding. Deretter vil `kjøretøybehandler.load(skademelding)` bli kjørt. Dataene for den valgte skademeldingen vil bli lastet inn i skjemaet, og første side av skjemaet vil bli vist.



Figur 15: Skjermbilde av skademeldingslisten. Her ser vi at brukeren har lagt til tre skademeldinger.

For å opprette en ny skademelding brukes knappen merket med et plusstegn som finnes øverst til høyre på skjermen. Her blir man tatt videre til menyen men valgene for hvilken type skademelding man vil opprette. Når man har valgt dette vil det kjøres en funksjon som legger til den nye skademeldingen øverst i listen.

For hver skademelding i listen finnes det også en knapp for å sende skademeldingen til server. Data om skademeldingen blir hentet fra den lokale databasen og sendt som JSON til server for videre behandling.

På Android gir det å holde nede på et element ofte muligheter til å gjøre ting med elementet [32]. Vi har valgt å følge disse anbefalinger Google gir. Men ettersom vi bare har en aktivitet som skal gjøres på elementet, kommer det ikke opp en meny, men et spørsmål om å slette.

5.3.4 Skademeldingsskjema

I et skademeldingsskjema er det to sett med felter som skal fylles ut, et for kjøretøy A og et for kjøretøy B. For å spare på plass og for at brukerne skal slippe å bla seg igjennom veldig mange sider for å fullføre skademeldingsskjemaet, har vi valgt å løse dette ved at man kan bytte mellom kjøretøyene med knapper på en footer-meny. Dette gjelder kun hvis skademelding fylles ut for to brukere på en telefon. Dette kan en se i skjermbildene under.

Figur 16: Bytte mellom fører A og B

Koden for denne funksjonaliteten ligger i klassen Kjoretøybehandler. Denne klassen tar seg av alt som har med lagring og innlasting av data i skademeldingen, og inneholder variable (arrays) som tar vare på brukerdataene fra de forskjellige delene av skademeldingsskjemaet. Den holder rede på hvilken bruker som er den aktive, A eller B. Og sørger også for at dataene som blir vist hører til den valgte brukeren. Lagring og lasting av data skjer via en `bytt()`-funksjonen som kjøres hver gang man bytter bruker. Da vil først dataene bli lagret, så blir alle feltene i skademeldingsskjemaet tømt, og deretter blir dataene for den andre brukeren lastet inn og vist.

En utfordring vi fikk underveis var at når brukere skulle skrive skademeldingene på hver sin mobile enhet. Siden vi først implementerte muligheten for å bytte mellom bruker A og B i footeren, og at når man skriver på hver sin enhet skal det ikke være mulig å bytte over til den andre brukeren, ble dette et problem. Dette har vi valgt å løse ved at vi har fjernet motpartens knapp slik at man ikke kan bytte. Fellesinformasjonen og situasjonsriss kan bare fylles inn på bruker A sin telefon. På den måten slapp vi å lage et nytt skjema for individuelle brukere.

5.3.5 Skadesituasjon og skademarkering

I skademeldingsskjemaene skal førerne lage en felles skisse av skadesituasjonen. I tillegg skal de individuelt markere synlige skader på sine kjøretøy.

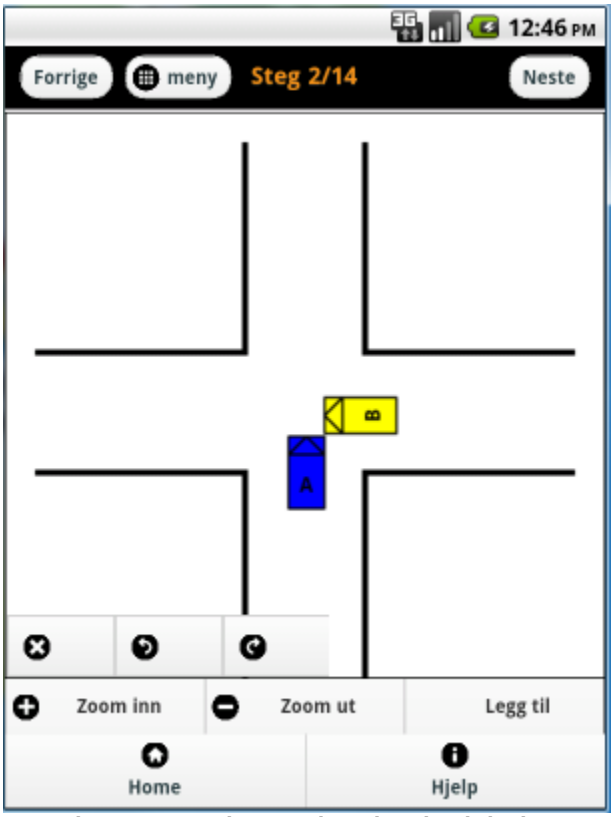
Helt i starten av prosjektperioden ble det diskutert med oppdragsgiver hvordan de ønsket denne løsningen. Først var det snakk om å bruke frihåndstegning. Men dette var noe som raskt var uønsket fordi de så det som problematisk å tegne med frihånd på enheter med små

skjermer. Derfor ble det bestemt at det skulle lages en modul der det kunne behandle ulike objekter. Denne delen ble det satt av mye tid til i prosjektperioden.

Ettersom skadesituasjonsdelen og skademarkeringsdelen er to veldig like funksjoner har vi valgt at disse to delene arver fra en klasse vi har kalt Tegnebrett. Dette har gjort at vi har svært lite kode som er individuell for de to delene.

Skadesituasjon

Når brukeren skal utforme skadesituasjonen kan han velge å legge til objekter, fjerne objekter, rotere objekter og flytte på dem. I tillegg har brukeren mulighet for å zoome inn og ut for å se på detaljer eller for å få oversikt. Når brukeren kommer til denne skjermen vil tegnebrettet zoome seg etter skjermstørrelsen slik at hele tegnebrettet blir synlig. Dette har vi gjort for å ta hensyn til alle de ulike skjermstørrelsene og for at brukeren med en gang skal få oversikt over hele situasjonen.



Figur 17: Tegning av Situasjonsbeskrivelse

Når en bruker legger til et objekt blir dette objektet et valgt objekt. Det vil si at brukeren nå kan rotere eller slette dette. Ved å trykke på et annet objekt blir dette det valgte objektet. Dette gjelder ikke hvis et veikryss blir lagt til. Disse legges alltid nederst av objektene og kan ikke roteres eller slettes, bare byttes ut ved å velge et nytt veikryss.

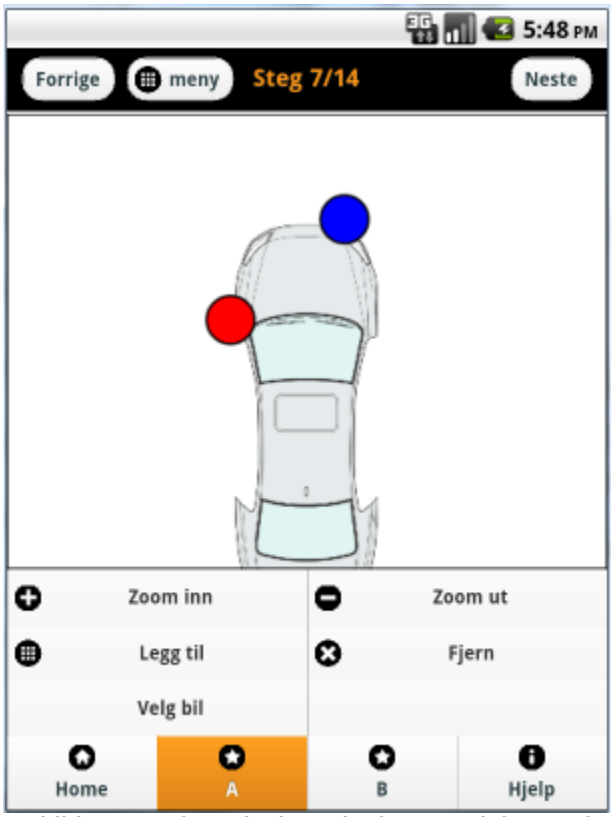
Alle nye objektene som skal lages hardkodes eller legges til fra bildefiler, og legges til i klassen Situasjonsobjekter. Det å hardkode objektene har visse ulemper. Det kan være utfordrende å skrive disse hvis det er komplekse objekter. Men et poeng er at alle objekter skal være så simple som mulig for å at brukeren skal forstå skissene. Under ser vist et eksempel på hvordan en rett veg kodes.

```
/**
 * Oppretter en Shape, en rett veg, som kan legges til
 * tegnebrettet.
 * @function
 * @returns {Shape} et objekt som er utformet som en rett veg.
 */
Situasjonsobjekter.prototype.rettVeg = function() //Rett veg
{
    var that = this;
    var obj = new Kinetic.Shape({
        drawFunc: function(){
            var context = this.getContext();

            //Her starter tegninga av shapen (den rette vegen)
            context.beginPath();
            context.lineWidth = 10;
            context.strokeStyle = "black";

            context.moveTo(0, 0);
            context.lineTo(that.w * 0.9, 0); //w er bredden på tegnebrettet
            context.moveTo(0, that.w * 0.2);
            context.lineTo(that.w * 0.9, that.w * 0.2);
            },
            draggable:false,
            centerOffset: {//Hvor center er på shape-en
                x: that.w * 0.45,
                y: that.w * 0.1
            }
    });

    //Legger til parametere om objektets størrelse og navn
    obj.w = this.w * 0.9;
    obj.h = this.h * 0.2;
    obj.name = "rettveg";
    return obj;

}
```

Skademarkering

I denne delen skal hver av brukerne markere synlige skader på hvert sitt kjøretøy. Dette skjer ved at brukeren får et bilde av en bil og markerer stedet på bilen hvor skadene er, ved at han trykker på skjermen/bildet. Skaden blir markert på bildet i form av en sirkel, og disse objektene blir lagt til på samme måte som i situasjonsriss. Hver gang brukeren markerer en skade vil det komme opp et tekstfelt hvor han kan skrive en kommentar på hva skaden gjelder. For å se på eller endre på kommentarene må man holde inne (2 sek) på markering man ønsker, og tekstfeltet vil dukke opp igjen.

Man har valget mellom bilde 2 forskjellige biler. Bildet av bilen er lagt på den nederste layeren, og bytting mellom bildene fungerer på samme måte som med veikryssene i situasjonsriss.



Figur 18: Skjermbildet over viser skademarkerings-modulen med alle menyvalg.

5.3.6 Lokal database

For lagring på telefonen har vi benyttet oss av PhoneGap sin implementasjon av W3C Web SQL Database [33] som er tilgjengelig gjennom Database-API-et[34]

Vi har to databasefiler, Brukeraksess.js som tar seg av registrering og lagring av brukere, og Skademeldingsaksess.js som tar seg av lagring og lasting av skademeldinger. Ved oppstart av appen vil det bli opprettet en lokal database ved at konstruktøren `database()` blir kjørt, slik som vist under.

```
database = function (){
    //oppretter Db -objekt
    this.db = window.openDatabase("skademelding", "1.0", "SMDB", 1000000);
    //oppretter brukertabell
    this.db.transaction(this.createTable);
}
```

Sql-setning for brukertabellen:

```
tx.executeSql('CREATE TABLE IF NOT EXISTS USER (id INTEGER PRIMARY KEY,
username TEXT UNIQUE, password TEXT , uid INTEGER)');
```

Vi har vi lagt til rette for at det kan registreres og lagres flere brukere i den lokale databasen. For å holde orden på hvilken bruker som sist logget inn har vi benyttet en annen type lagringsmulighet i PhoneGap, nemlig localStorage. Denne lagringsmetoden baserer seg på nøkkel/verdi lagring. Under er vist hvordan det kan settes data i ved bruk av denne metoden:

```
window.localStorage.setItem("sistebruker", uid);
```

Klassen Skademelding representerer en skademelding og er en holder for alle dens data. Klassen inneholder set- og get-metoder for å klargjøre skademeldingsdata. Den har metoden `toJSON()` som omformer skademeldingen til en JSON-string. I tillegg har også en funksjon `fromJSON()` for å laste inn til en skademelding.

Skademeldinger er klassen som sørger for å opprette nye skademeldinger og lagre de i databasen, og hente ut tidligere lagrede skademeldinger fra databasen. Ved oppretting av en ny skademelding vil det legges inn en skademelding i databasen, men denne vil ikke inneholde noen data før brukeren har fylt inn info i skademeldingen.

```
database2.prototype.insertSkademelding = function(data, uid, callback){
    this.db.transaction(function insertDB(tx) {
        var sql ="INSERT INTO SM2 (data, uid) VALUES (?, ?)";
        //Setter inn skademeldingen i databasen
        tx.executeSql(sql, [data, uid], function success(tx, results){
            var insertID = results.insertId;
            callback(insertID); //Gir beskjed om id-en den ble satt inn med.
        });
    });
}
```

En skademelding blir lagret lokalt når brukeren går tilbake til skademeldingslisten. Det vil da bli kjørt en tilsvarende funksjon som over bare ved at SQL-setningen oppdaterer isteden for å sette inn:

```
var sql ="UPDATE SM2 SET data=? WHERE id=" + id;
```

5.3.7 Internasjonalisering

Denne funksjonaliteten er ikke fullstendig implementert, men vi har laget et forslag til hvordan man kan tilpasse appen til forskjellige språk. Siden det i JavaScript ikke finnes noen metode for internasjonalisering og lokalisering slik som i for eksempel Java, har vi løst dette ved å bruke et JavaScript-bibliotek som heter jquery.i18n.js [35]. Dette inneholder metoder for å oversette tekst i HTML-elementer. Det første som gjøres er at man oppretter en ordliste hvor man definerer ulike ord med en tilhørende oversettelse. Under ser du hvordan denne ordlisten er utformet.

```
var i18n_EN = {
        'Skadetidspunkt'        :   'Time of accident',
        'Skadested'             :   'Place of accident',
        'Personskade'           :   'Human injury',
        'Nei'                   :   'No',
        'Ja'                    :   'Yes',
        ...
};
```

Deretter kjøres en kode som finner språkkoden. For eksempel kan browser-objektet gi forkortelsen 'nb', som betyr Norsk Bokmål, og JavaScriptet vil sette den riktige ordlisten som gjeldende.

Når riktig ordliste er satt, kjøres kode for å oversette alle de valgte elementene.

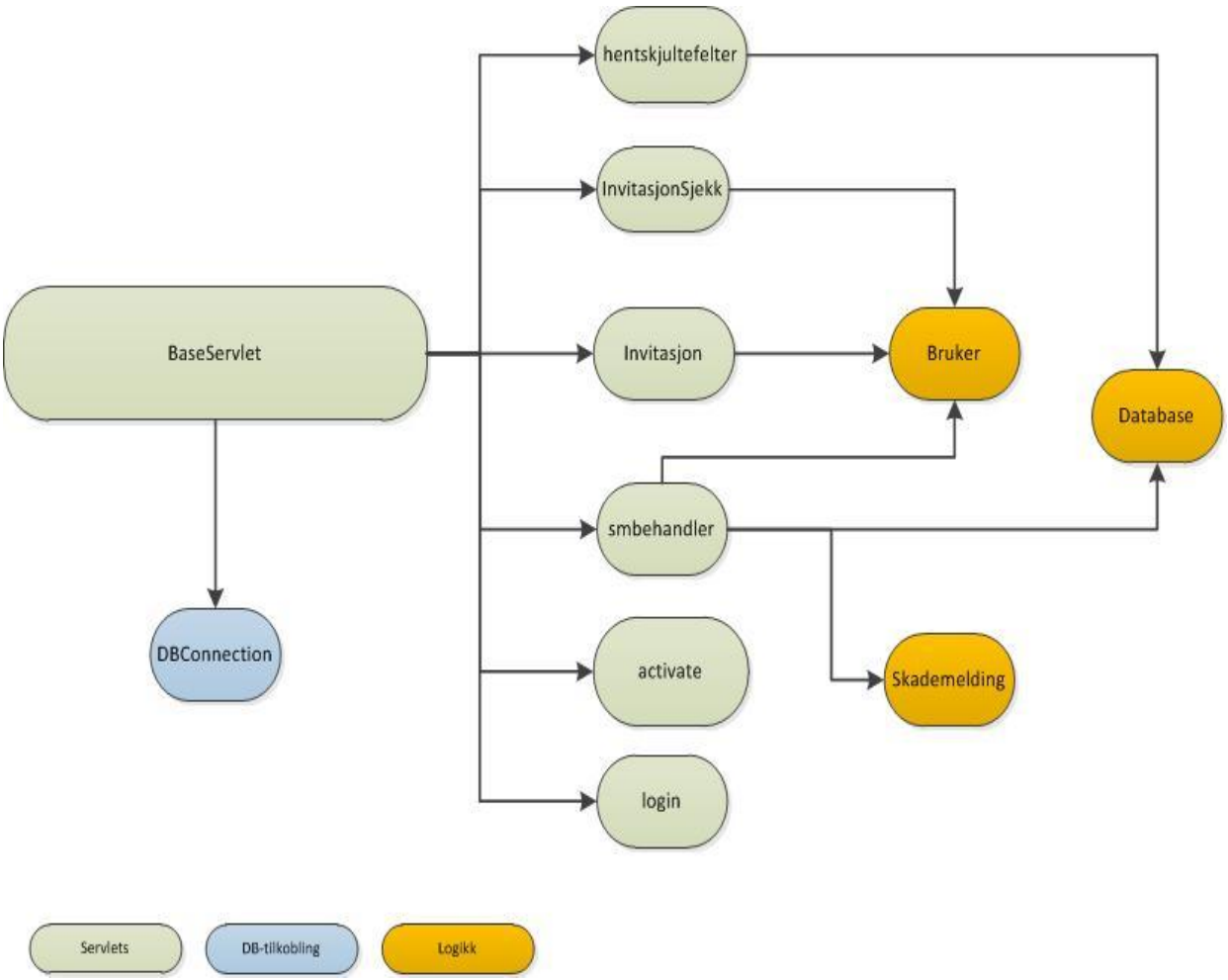
```
/**
 * Funksjonen som oversetter
 * @function
 */
function translate(){
   //Her settes teksten for hver av elementetene som
   //skal oversettes. Dette sjekkes opp mot ordlisten
   //som er spesifisert
   $('#hSkadetidspunkt').text($.i18n._('Skadetidspunkt'));
   $('#hSkadested').text($.i18n._('Skadested'));
   //... her skal alle de andre elementene som skal oversettes være
}
```

Alle elementer som skal oversettes må spesifiseres i denne metoden. Fordi det er mange felter blir dette en svært lang metode. Men når først alle felter er satt så skrives oversettelsen i ordlistene. Vi så også på alternative biblioteker som jQuery.i18n.properties.js og i18next.js, men valgte å bruke jquery.i18n.js på grunn av at anvendelsen var enkel å forstå, og fordi det lett kunne implementeres i vår kode.

5.4 Implementasjon av server og web

5.4.1 Serverstruktur

Som nevnt tidligere har vi utviklet en server for å behandle og lagre data. Denne er bygget opp av ulike servleter og klasser som vi videre vil forklare litt nærmere om.



Figur 19: Struktur av serverdel

```
private String genererSql(JSONArray fraJSON, HashMap<String, String>
                          mappingFelter, String databasetabellNavn)
{
        String felter = ""; //alle kolonnenavn som skal behandles
        String data = ""; //string med spørsmålstegn,

        //går igjennom hele JSONarrayen og henter ut hvert objekt
        for (int i = 0; i < fraJSON.size(); i++) {
            JSONObject smfelt = (JSONObject) fraJSON.get(i);

        //legger til kolonnenavn i "felter", og legger til et spørsmålstegn i
```

```
        "data" for hver kolonne som skal brukes
            felter +=  "," + mappingFelter.get(smfelt.get("name"));
            data += ",?";
        }
        if (felter.length() > 0) felter = felter.substring(1);
        if (data.length() > 0) data = data.substring(1);

        //returnerer den genererte SQL-setningen
        return String.format("INSERT INTO %s (%s) VALUES (%s)",
                                databasetabellNavn, felter, data);
    }
```

Den generert SQL-setning kan da se slik ut:
`"INSERT INTO kjoretoy (forsikringstaker_fornavn, forsikringstaker_etternavn) VALUES (?, ?)".`

For å kjøre den genererte SQL-setningen og sette inn verdier i databasen, benytter vi funksjonen `kjørSQL()`. Denne tar den genererte SQL-setningen og JSON-arrayen som parametere og setter så inn dataene.

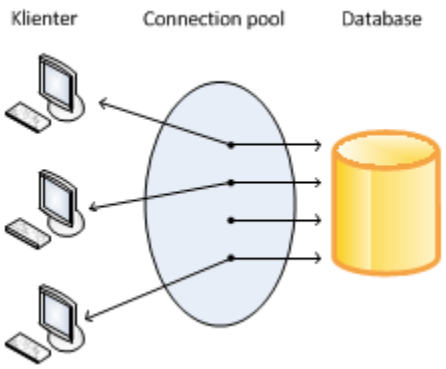
```
private int kjørSQL(String sql, JSONArray jsonData)
    {
        int insertID = -1;
        PreparedStatement stmt = null;
        try
        {   //klargjør spørringen med SQL-setningen
            stmt = con.prepareStatement(sql,
Statement.RETURN_GENERATED_KEYS);
            //går igjennom JSON-arrayen og henter ut verdiene
            for (int i = 0; i < jsonData.size(); i++) {
                JSONObject smfelt = (JSONObject) jsonData.get(i);

                String felt = (String) smfelt.get("name");
                String val = (String)smfelt.get("value");
                //kjører funksjon som setter inn verdien for hver kolonne
                setStmtData(stmt, i, felt, val);
            }

            int num = stmt.executeUpdate();//kjører spørringen

            //Finner ID-en på den som sist ble satt inn
            ResultSet rs = stmt.getGeneratedKeys();
            if (rs.next())
                insertID=rs.getInt(1);
            else
            {
                if (stmt.getUpdateCount() > 0)
                    insertID = -2; //Returnerer at vi har oppdatert en eller
flere rader
            }
        }
        catch (Exception ex) {
           System.out.println("Error - kjørSQL():" + ex.toString());
        }
```

```
        finally {
            DBConnection.closeStmt(stmt);//lukker spørringen
        }
                return insertID; //returnerer Id-en i DB
    }
```

5.4.2 Databasetilkobling

For databasetilkobling har vi valgt å bruke JDBC som driver. JDBC er svært mye brukt low level API og gir oss muligheten, gjennom Java, til å kjøre spørringer mot databasen[36].

Ettersom det er meget ressurskrevende å opprette databasetilkoblinger er det hensiktsmessig å benytte seg av en metode som tar vare på koblingene slik at de kan brukes igjen senere. Dette kalles ofte en connection pool. Disse fungerer på en slik måte at dersom en klasse trenger en tilkobling så spør den connection pool-en om å få en. Dersom det finnes en ledig tilkobling blir denne returnert. Er alle opptatt, oppretter connection pool-en en ny tilkobling og returnerer denne [36]. Når man i klassen ikke lenger trenger koblingen kan man lukke den slik at connection poolen igjen kan gi den til andre.



Figur 20: En connection pool tar vare på tilkoblingene slik at de kan gjenbrukes.

I dag finnes det flere ferdige implementasjoner av slike connection pools. Vi har valgt å benytte oss av Java Naming and Directory Interface (JNDI) for å få til en slik løsning. Det vil i hovedsak si at vi har spesifisert selve databasetilkoblingen i xml-filen WebContent/META-INF/context.xml. Videre er det også spesifisert mappings i filen WebContent/WEB-INF/web.xml

5.4.3 Web

Webdelen av systemet er laget for å gi brukerne mulighet til å hente opp igjen innsendte skademeldinger på PC, utføre eventuelle korreksjoner/endringer, og signere skademeldingen. Webdelen fungerer også som et administrasjonsverktøy for skadeansvarlig, ved at det finnes funksjonalitet for å konfigurere skademeldingsskjemaet (se konfigurerbarhet), og legge til kommentarer på skademeldinger. Brukeren må logge inn med det brukernavn og passord han

har registrert. Når man har logget inn vil man bli vist en liste over de skademeldinger man er berørt av, og en meny til venstre med de valgene man har.

Figur 21: Brukeren sin skademeldingsliste

I listen over skademeldinger har man valget om å signere skademeldingen. Hvis man er fornøyd med informasjonen i skademeldingen trykker man på "tommel opp"-ikonet og skademeldingen blir markert som signert i databasen. Signerte skademeldinger vil markeres i listen, og motparten kan dermed se at du har godkjent dataene i skademeldingen. Etter man har signert en skademelding skal det ikke være mulig å endre på den.

Ved å trykke på datoen til en av skademeldingene i listen, vil man bli vist den valgte skademeldingen.

Når man er inne på en skademelding vil det bli lagt til flere valg i menyen til venstre. Etter eventuelle endringer trykker man på "lagre endringer". Da sendes skjemaet tilbake til server, og databasen oppdateres med de nye dataene. De andre valgene/knappene brukes for å navigere seg mellom de forskjellige delene av skademeldingsskjemaet.

5.4.4 Web-løsningen

Web-løsningen er bygget opp av JSP-sider med HTML og JavaScript. Mye av koden er gjort på samme måte som i appen. Den er utviklet som en dynamisk web-side hvor vi bruker AJAX for å laste inn innholdet. Index.jsp inneholder div-en "content", og det er her all informasjon

brukeren ønsker å se blir lastet inn. Det er også her brukt AJAX for kommunikasjon med server, og innlasting av data i skjemaet skjer slik som det gjør i kjøretøybehandleren på appen.
Tegnemodulen i web-løsningen er ren gjenbruk av koden fra appen. Dette er gjort ved at vi i web-prosjektet i Eclipse har linket til filene i appen som har med tegnemodulen å gjøre.

Ved innlogging vil det bli opprettet en ny session for brukeren, hvor det blir tatt vare på brukernavn og passord. Ved at denne informasjonen ligger lagret i sessionen, sikrer man at brukeren er logget inn, og at man slipper å taste inn brukernavn og passord hver gang man skal hente/lagre i databasen. Det vil også sjekkes hvilken type bruker(privat, firma, skadeansvarlig) som har logget inn, og gis ulike rettigheter/valgmuligheter basert på dette.

Måten data blir hentet ut fra databasen på, er at det fra index-siden sendes en ajax-forespørsel til en annen JSP-side hvor det blir opprettet et objekt av klassen Database. Dette er en Java-klasse som brukes til kommunikasjon med databasen. Deretter hentes de data som er etterspurt, og sendes tilbake til AJAX-funksjonen.

Se vedlegg C for eksempelkode på en JSP-side.

5.5 Konfigurerbart skademeldingsskjema

Konfigurerbarhet handler om hvilke felter som skal være synlige i en skademelding og hvilke som skal være skjulte slik at brukeren ikke kan se dem og dermed slippe å fylle dem ut. Denne funksjonaliteten vil kun være tilgjengelig for brukere som er ansatt i et firma. Det betyr at privatbrukere alltid vil måtte fylle ut alle felter.

Alle firmabrukere har en skadeansvarlig over seg som har ansvaret for å velge hvilke felter som ikke skal være synlige og hvilke som skal være skjulte. Dette gjøres på web og er implementert slik at de felter som velges av skadeansvarlig er de som ikke skal være synlige for firmabrukeren.

Figur 22: En skadeansvarlig kan velge de felter en skademelding skal inneholde for firmabrukere.

For eksempel kan en skadeansvarlig velge at alle brukere han har under seg ikke trenger å fylle inn om de driver en forretning eller andre opplysninger. Ved lagring av feltlisten blir den lagret i brukertabellen. Og her kan den enten bli lagret direkte hos den skadeansvarlige eller hos alle firmabrukere som har den skadeansvarlige over seg. Hvis vi hadde valgt den siste løsningen ville vi mistet denne listen dersom den skadeansvarlige ikke har noen brukere under seg. Vi har derfor valg å lagre feltlisten direkte på den skadeansvarlige.

Når en firmabruker åpner appen og logger inn vil systemet kontakte en servlet som spør hvilke felter som denne brukeren ikke skal se. Spørringen må da sjekke hvilke felter den skadeansvarlige har valgt at ikke skal vises.

5.6 Invitasjon

Ved oppretting av ny skademelding får brukeren en meny der bruke kan velge å invitere en motpart eller hente en invitasjon, se skjermbilde i vedlegg E.

Inviter motpart

For å kunne utføre en invitasjon kreves det at enheten som inviterer er koblet til Internett. For at motparten skal få koble seg på denne invitasjonen må også denne enheten ha internettilkobling. Det var diskutert om det var hensiktsmessig å koble skademeldingene sammen med en kode som skrives inn hos motparten for så koble de sammen når brukerne kommer på nett. Men da er det ingen garanti for at brukerne faktisk gjør dette.

Når en bruker velger "inviter motpart" vil det komme opp et tekstfelt hvor han skriver inn motpartens brukernavn og deretter velger send. Funksjonen

`sendInvite(motpartBrukernavn, brukernavnA)` fra Server.js vil da kjøres. Den sender et jQuery AJAX-kall til server hvor brukernavnene sendes med. På server sjekkes det om dette brukernavnet finnes. Dersom det finnes opprettes en invitasjon og en ny skademelding i databasen. Invitasjonen vil da inneholde ID-en til begge brukerne og ID-en til skademeldingen. Skademeldingen som opprettes i databasen inneholder ID-en til bruker A, men informasjon om bruker B vil ikke bli lagt inn før han har akseptert invitasjonen. Dette er gjort for å ikke fylle databasen med data hvis bruker B ikke godtar invitasjon.

Dersom det skjedde en feil, for eksempel at brukernavnet til motparten ikke finnes, vil server sende tilbake et error-resultat, som gjør at det blir vist en melding på skjermen til brukeren. Av andre feil som kan oppstå er en invitasjonsError som betyr at Databasen ikke klarte å opprette en ny rad i Invitasjonstabellen, for eksempel hvis databasen er nede. Eller det kan være at det er to like brukernavn, altså at brukeren prøver å invitere seg selv (likeBrukerIDer).

Hent invitasjon

Motparten kan så hente en invitasjon til sin enhet. Da vil funksjonen `getInvite(brukernavn)` kjøres. Denne gjør et tilsvarende jQuery AJAX-kall for å hente invitasjonen. Server sjekker i databasen om det er en ny invitasjon. Hvis det finnes en invitasjon, sender server tilbake et resultat med “nyInvitasjon” og id-en på skademeldingen i databasen. Brukeren får beskjed om at det finnes en ny invitasjon og blir tatt videre til skademeldingsskjemaet.

På server vil bruker B’s ID legges inn i skademeldingen som ble opprettet av bruker A, og deretter vil invitasjonen slettes fra databasen.

Vi vurderte også å finne en metode for å automatisk gi motparten melding om at det var mottatt en ny invitasjon, ved for eksempel en push-løsning. Etter å ha diskutert dette med oppdragsgiver ble vi enige om at det var en grei løsning at motparten selv må sjekke om det finnes en invitasjon.

6 Testing

Testingen har hjulpet oss til å luke bort feil i systemet. Dersom vi i en test har funnet feil har disse blitt rettet opp. Dette gjør at vi kan sikre oss at systemet fungerer som forventet. Men testing er ingen bevis for at systemet er feilfritt. I en bok skrevet av Henrik Kniberg som tar for seg Scrum beskrives testing som den vanskeligste oppgaven, kanskje ikke i Scrum, men innen programvareutvikling generelt [2]. Testingen vil avhenge av type programvare, hvem brukerne er, hyppighet på utgivelser og så videre. Kniberg mener det er fornuftig å ha egne testere med i Scrum-teamene. Dette er noe vi ikke har mulighet til og vi må derfor selv fungere som disse testerne. Hvordan har vi så utført testing?

6.1 White box testing

Vi har for det første benyttet oss av ulike tekniske tester. Disse har foregått kontinuerlig gjennom utviklingen. For det første har vi testet systemet på våre utviklingsmaskiner og videre har vi brukt fysiske smarttelefoner for testing av appen. Den første formen for testing er selve debugging av koden. Her hjelper Eclipse oss med en egen debug-modus for Java-koden. For JavaScript har vi benyttet oss av FireBug som finnes til nettleseren Firefox og utviklerverktøyet i Chrome. I tillegg har vi kjørt appen med USB-debugging på telefonen, og ved feil under kjøring vil disse skrives ut til konsollen i Eclipse.

I deler av systemet der det har vært vanskelig å kode har vi benyttet oss av såkalt white-box testing.

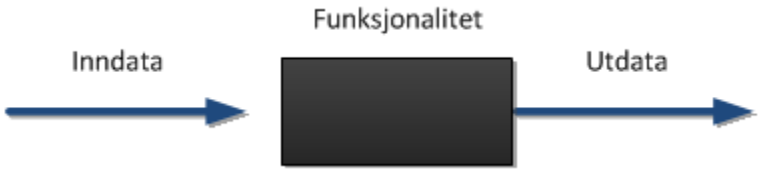
Figur 23: White box diagram

White box testingen går ut på, slik som J. M. Myers forklarer i sin bok, at å ta oss frem gjennom programkoden og følge med på hva som skjer internt i komponenten[37]. Dette krever at testeren har tilgang til programkoden. Her går vi altså direkte inn i koden for å se at de ulike variabel-verdiene stemmer overens med det som er forventet. Det er viktig at slike tester fungerer, ikke bare i de verdiene vi forventer, men også i de verdiene vi ikke forventer. I tillegg til de forventede dataene lister T. E. Hasle opp hvilke ulike inndataverdier som bør være med i testene: Grensedata (maks/min-verdier), data utenfor grensene, null-verdi og negative verdier[1]. Disse verdiene er viktig å teste fordi de representerer verdier som ofte kan føre til feil.

Denne metoden har gjort at vi letter kan finne frem til feil i koden ved at debug-verktøyene gir oss en indikasjon på hvor i koden feilen oppstod, og dermed kan rette den opp.

6.2 Black box testing

Det har også vært gjort en annen type tester på systemet. Disse testene har vært utført av personer som ikke har kjennskap til hva som skjer i programkoden. Denne metoden kalles black-box testing. Her er testeren ikke interessert i hva som skjer inni systemet og dets struktur, men prøver å finne noe som ikke samsvarer med spesifikasjonene[37]. Dette skjer ved å se på utdataene i forhold til det som var forventet.



Figur 24: Black box diagram

I løpet av prosjektperioden har oppdragsgiver kontinuerlig testet de ulike funksjonalitetene i systemet. Dette er blitt gjort for å få tilbakemelding på om funksjonaliteten er slik de hadde tenkt seg, og det har ved flere tilfeller resultert i at vi har måtte gjøre endringer.

Et problem som ble oppdaget ved denne testingen, er at den funksjonen som gjør at brukeren kan velge skadetidspunkt gav datoen ut på feil format i forhold til det som krevdes.

Noe av det første vi utviklet var selve skjemaet (formalia) for appen. Dette var delt opp i 2 sider; 1 for fremsiden og en for baksiden. Brukeren måtte da scrolle seg nedover på skjermen for å se alle feltene. Dag testet dette, og mente det var for mye data på en side. Når brukeren må scrolle igjennom så mye data, vil det føre til at det ble vanskelig å holde seg orientert om hvor de forskjellige opplysningene skal tastes inn. Dag ønsket at vi omstrukturerte skjemaet slik at man kunne bla seg frem og tilbake mellom de ulike delene av skjemaet. Ved å løse det på den måten blir det minimalt med scrolling, og skjemaet blir mer oversiktlig.

En annen ting vi har måtte endre på som følge av testing, er menyen i “situasjonsriss”. Denne var først implementert sammen med footer-menyen, og Dag mente at dette tok for stor plass på skjermen. Han ville derfor at vi skulle dele opp denne menyen slik at valg som hadde med objekter i tegningen å gjøre lå for seg selv (roter, slett), og at footer-menyen skulle inneholde valg som hadde med selve tegnebrettet å gjøre (zoom, legg til). På den måten kan man velge å lukke footer-menyen for å få mer plass til å behandle objektene man har lagt til.

På et annet sprintmøte ble skademarkerings-modulen testet av Øyvind F. Da kom det frem at han hadde glemt å si at det måtte være mulighet for å legge inn kommentarer på de markerte punktene, og at implementasjon av dette var noe vi burde prioritere. Dette resulterte i at vi opprettet et nytt element i produktkøen, og utviklet en løsning for kommentarer på skademarkering i neste sprint.

7 Avslutning

I dette siste kapittelet skal vi oppsummere og trekke noen linjer i forhold til det vi har gjort. Vi skal først diskutere resultatene av oppgaven, før vi sier litt om hva som gjenstår med videre arbeid på systemet. Videre evaluerer vi gruppas arbeid før vi kommer med noen personlige tanker rundt prosjektet. Helt til slutt kommer en konklusjon av prosjektet.

7.1 Diskusjon av resultater

Vårt system består av en applikasjon for håndholdte enheter, et web-grensesnitt og i tillegg en serverdel. En stor del av arbeidet har gått med til å utvikle appen. Siden en skademelding består av veldig mange felter har det tatt mye tid å få satt opp feltene og delt dem inn i sider, samt å få laget SQL-spørringer som kan sette de ulike feltene inn i databasen på server.

Vi har etter beste evne prøvd å få systemet til å bli så ferdig som mulig slik at systemet kan bli tatt i bruk innen ett år hos oppdragsgiver, slik som målet deres er. Vi har også valgt å bruke verktøy som er kjente og godt dokumenterte, slik som Eclipse, jQuery og Tomcat. Dette vil gjøre at de som skal fortsette utviklingen og vedlikeholde systemet raskt kan få oversikt og komme i gang. Vi har også laget en enkel bruksanvisning om hvordan komme i gang med utvikling av systemet, se vedlegg L.

I starten gikk det med en del tid til kartlegging av hva oppgaven gikk ut på. Det var noe uklarhet rundt hvilke funksjonaliteter vi faktisk skulle lage. Men på tross av dette startet vi raskt med utvikling. Dette var noe oppdragsgiver satt pris på.

Et av hovedfokusene her har vært den visuelle beskrivelsen av skadesituasjonen som brukerne skal tegne opp. Dette er noe oppdragsgiver ønsket at skulle være godt implementert og er noe vi selv er godt fornøyd med. Det kunne allikevel ha vært lagt til flere ulike objekter. Men vi har prioritert å kun lage noen siden det senere vil gå raskt å legge til de som er aktuelle. Vi så at tegning av skadesituasjonen og skademarkering trengte mange av de samme funksjonene. Ved å la disse klassene arve fra en Tegnebrett-klasse har vi kunne gjenbruke mye kode.

På server kobler Servlet-ene opp til databasen gjennom en Connection pool. Dette gjør at vi får en serverdel som krever mindre ressurser og kan dermed håndtere flere forespørsler.

I forhold til arbeidet med rapporten ser vi nå i ettertid at vi burde ha kommet skikkelig i gang med den tidligere. Vi var i starten fokusert på utviklinga, noe som gjorde at vi satt av for lite tid til skriving. Først etter et skrivekurs som ble arrangert tok dette arbeidet seg opp. Det vil si at vi har brukt mye av tiden på slutten til rapportskriving.

Selv om vi er fornøyd med resultatet er vi ikke helt i mål. Det gjenstår fortsatt en del arbeid som må gjøres før det kan lanseres. Men vi har nå lagt grunnlaget for at skademeldinger i framtiden kan bli skrevet og sendt inn elektronisk til forsikringsselskap.

7.2 Videre arbeid

Ut over det arbeidet vi har gjort er det ennå en del som gjenstår før det kan lanseres for brukerne til oppdragsgiver. For det første må inngås avtaler med forsikringsselskap om hvilke måte de kan motta og på hvilken form de vil motta data. Dette er en del som har gått utenfor oppgaven.

Videre er det ennå ikke implementert en full synkronisering mellom de ulike delene av systemet. I hovedsak er det nå bare mulig å sende skademeldinger til server for så å behandle dem videre på web.

Vi har i systemet lagt mest vekt på at appen. Dette har sammenheng med at oppdragsgiver ennå ikke er inni dette markedet og da ønsket at denne delen skulle være mest mulig ferdig. For dem vil det være mye enklere å ta tak i arbeid som må gjøres på webdelen. Derfor er det også på denne delen det gjenstår mest arbeid.

Det kan i tillegg vurderes å implementere en PUSH-løsning for invitasjoner slik at disse kommer direkte til de som blir invitert.

7.3 Evaluering av gruppas arbeid

Gruppa som har arbeidet med dette prosjektet består av to personer på bachelorstudiet Programvareutvikling ved Høgskolen i Gjøvik. Vi har arbeidet jevnt hele prosjektperioden, men de siste ukene har dette intensivert seg. Etter hver arbeidsdag har vi skrevet logg for å kunne dokumentere hva vi til en hver tid har arbeidet med, se vedlegg I. Vi har sett at dette har vært en stor oppgave og at det har vært svært vanskelig å ferdigstille hele systemet.

7.3.1 Organisering

I prosjektplanen, se vedlegg F, planla vi at vi skulle bytte på å være gruppeledere. Vi gjorde det slik at vi byttet ved hver sprint slik at begge skulle få følelsen av å ha et større ansvar for spriten. Dette mener vi har fungert veldig godt for oss i dette prosjektet. Men vi ser at dette ikke er en realitet i arbeidslivet siden det kan føre til problemer ved at man ikke vet hvem man skal forholde seg til, til en hver tid.

Bruken av Scrum har fungert godt. Med to ukers sprinter har vi holdt arbeidstempoet oppe samtidig som vi har fått gjort ganske mye arbeid i hver sprint. For oss har det vært viktig å hatt jevnlig kontakt med oppdragsgiver, også mellom sprintene. Dette har hjulpet oss til å holde oss på rett spor gjennom sprinten.

7.3.2 Arbeidsfordeling

Vi har gjennom hele prosjektperioden arbeidet mye sammen som gruppe. Noen av oppgavene har vi fordelt mellom oss, mens andre har vi arbeidet tett sammen med. Ved starten av hver sprint fordelte vi ulike oppgaver som måtte utføres de neste to ukene. Fordi vi begge var nye på mye av teknologien vi har brukt, har vi sett at det å kunne samarbeide og hjelpe hverandre har vært svært verdifullt. Bruken av de daglige møtene har gjort at vi hele tiden har vært oppdatert på hva den andre sine oppgaver har vært.

Underveis har vi begge både skrevet på denne rapporten i tillegg til utviklingen. Vi har her fordelt noen av delene i rapporten som skulle skrives. Men hovedstrukturen, modeller og viktige punkter har vi diskutert nøye i fellesskap.

7.4 Subjektiv opplevelse av prosjektet

Arve

Min personlige opplevelse av prosjektet har vært veldig positiv. Jeg mener selv vi har arbeidet godt som gruppe og lært veldig mye. En av grunnene til at jeg ønsket denne oppgaven var ikke det at jeg så denne som den letteste, men heller at jeg her kunne lære og utvikle meg mest mulig som utvikler. Det at jeg har måtte satt meg inn i flere nye teknologier har vært svært lærerikt, og også svært utfordrende. Spesielt når det har vært så mange forskjellige, Tomcat, JSP, Servlets, PhoneGap og til dels JavaScript. Det er ofte lett å se i ettertid hva man burde ha gjort annerledes når man begir seg ut på en ny teknologi. Å utvikle en app for håndholdte enheter har vært svært inspirerende og har åpnet veien videre i forhold til videre utvikling av nye apper.

Øyvind

Dette prosjektet har vært en bra opplevelse og det har vært veldig lærerikt å jobbe med en så stor oppgave. Å jobbe sammen med en reell kunde/oppdragsgiver har vært veldig interessant, og jeg føler dette er en god erfaring å ta med seg videre. Prosjektet har bydd på mange utfordringer, og det har vært mange nye teknologier å sette seg inn i. Selv om det til tider har oppstått problemer i utviklingen, har følelsen man sitter igjen med etter å ha funnet en løsning vært desto bedre. Noe jeg syns har vært spesielt gøy er å få erfaring på utvikling av mobile applikasjoner, og jeg føler at dette er noe jeg virkelig kunne tenke meg å jobbe med. Jeg synes at samarbeidet i gruppen har vært veldig bra, men føler at vi kanskje burde ha vært tre

personer, med tanke på størrelsen av oppgaven. Vi kunne ha kommet enda mer i mål med utviklingen hvis vi hadde hatt med en person til, men jeg alt i alt er jeg veldig fornøyd med resultatet av vårt arbeid. Dette prosjektet har gitt meg en mye bredere forståelse for systemutvikling, og jeg har fått programmeringserfaring som jeg mener er relevant i forhold til arbeidslivet.

7.5 Konklusjon

Gjennom dette bachelorprosjektet har vi fått kjenne på hvordan det er å jobbe med et større systemutviklingsprosjekt. Dette har vært et interessant og ikke minst et lærerikt prosjekt, som også har gitt oss et innblikk i hvordan systemutvikling foregår i det virkelige liv ved at vi har hatt kontakt med en faktisk oppdragsgiver. I løpet av dette halvåret har vi fått benytte den lærdommen vi har tilegnet oss i løpet av vår tid på HiG, og fått se hvordan mange av studiets ulike temaer brukes under ett.

Selv om det ville blitt vanskelig å løse denne oppgaven uten relevante forkunnskaper om systemutvikling, har vi også måtte lære oss veldig mye vi ikke hadde erfaring med fra før. Noe som har vært spesielt utfordrende er å finne ut av hvordan programmering mot håndholdte enheter gjøres, og hvilke muligheter man har i forhold til utvikling av kryssplattform-applikasjoner. Vi har også måtte sette oss inn i en rekke rammeverk innenfor JavaScript, samt lære oss serverteknologier som JSP og servlets.

En annen nyttig erfaring vi har fått i løpet av dette prosjektet er bruk av Scrum. Vi hadde teoretisk kunnskap om dette rammeverket fra før, men har nå igjennom denne oppgaven fått benyttet det i praksis. Selv om noen av aspektene ved Scrum ikke ble implementert grunnet størrelsen på prosjektgruppen, føler vi at dette er en god erfaring å ta med seg videre.

Programvaren er ikke klar for å slippes ut til kunder enda, men vi mener at arbeidet vi har gjort er et godt utgangspunkt for videre utvikling. Vi er godt fornøyd med resultatet av prosjektet, og håper oppdragsgiver vil få nytte av det vi har utviklet.

8 Litteraturliste

1. Hasle T. E. Systemutvikling Applikasjoner og databaser. Oslo:Cappelen Damm AS; 2008

2. Kniberg H. Scrum and XP from the Trenches. United States of America:C4Media Inc; 2007

3. Sutherland J., Schwaber K. The Scrum Papers: Nuts, Bolts, and Origins of an Agile Process. Cambridge;

4. Simula, Estimering i smidige prosjekter: http://simula.no/research/se/publications/Simula.SE.391/simula_pdf_file

5. Fog Creek Software. Trello sin hjemmeside [Internettside]. New York City: Fog Creek Software; 2002 [sitert 22.05.2012]. Tilgjengelig fra: http://trello.com

6. Sutherland J. Scrum Handbook, Somerville:Scrum Training Institute Press: 2010

7. Nazzaro W. F., Suscheck C. New to User Stories? [Internettside]. Indianapolis:ScrumAlliance; 2010 [sitert 22.05.2012]. Tilgjengelig fra: http://www.scrumalliance.org/articles/169-new-to-user-stories

8. Bell D. UML basics: The sequence diagram [Internettside]. United States: IBM Corporation; 2004 [sitert 22.05.2012]. Tilgjengelig fra: http://www.ibm.com/developerworks/rational/library/3101.html

9. Nielsen J. Ten Usability Heuristics [Internettside]. California [sitert 16.05.2012] Tilgjengelig fra: http://www.useit.com/papers/heuristic/heuristic_list.html

10. Adobe Systems Inc. Phonegap [Internettside]. United States: Adobe Systems Inc; [sitert 22.05.2012]. Tilgjengelig fra: http://phonegap.com

11. Adobe Systems Inc. Phonegap hjemmeside [Internettside]. United States: Adobe Systems Inc; [sitert 22.05.2012]. Tilgjengelig fra: http://docs.phonegap.com/en/1.7.0/index.html

12. Haynie J. How Does Appcelerator Titanium Mobile Work? United States:stack exchange inc.; [sitert 22.05.2012]. Tilgjengelig fra: http://stackoverflow.com/questions/2444001/how-does-appcelerator-titanium-mobile-work

13. Ansca Inc. Ansca Mobile Conoras hjemmeside [Internettside]. Palo Alto:Ansca Inc; [sitert 22.05.2012]. Tilgjengelig fra: http://www.anscamobile.com/corona/

14. Ansca Inc. Ansca Mobile Conoras hjemmeside [Internettside]. Palo Alto:Ansca Inc; [sitert 22.05.2012]. Tilgjengelig fra: http://www.anscamobile.com/pricing/?ref=home

15. Rhomobile Inc. Rhomoblie sin hjemmeside [Internettside]. California:Rhomobile Inc; [sitert 22.05.2012]. Tilgjengelig fra: http://www.rhomobile.com/

16. Appcelerator Inc. Comparing Titanium and PhoneGap [Internettside]. California:Appcelerator Inc; [sitert 22.05.2012]. Tilgjengelig fra: http://developer.appcelerator.com/blog/2012/05/comparing-titanium-and-phonegap.html

17. Dallera A. Why you should stay away from Appcelerator's Titanium [Internettside]. [sitert 22.05.2012]. Tilgjengelig fra: http://usingimho.wordpress.com/2011/06/14/why-you-should-stay-away-from-appcelerators-titanium/

18. The Apache Software Foundation. Tomcat sin hjemmeside [Internettside]. The Apache Software Foundation; [sitert 22.05.2012]. Tilgjengelig fra: http://tomcat.apache.org/

19. Microsoft Corporation. ASP.NET sin offisielle hjemmeside [Internettside]. Microsoft Corporation; [sitert 22.05.2012]. Tilgjengelig fra: http://www.asp.net/

20. Koggdal J. oCanvas sin offisielle hjemmeside [Internettside]. oCanvas [sitert 22.05.2012]. Tilgjengelig fra: http://ocanvas.org/

21. Rowell E. KineticJS sin offisielle hjemmeside [Internettside]. KineticJS [sitert 22.05.2012]. Tilgjengelig fra: http://kineticjs.com/

22. Baranovskiy D. Raphaël sin offisielle hjemmeside [Internettside]. Raphaël [sitert 22.05.2012]. Tilgjengelig fra: http://raphaeljs.com/

23. Eclipse Fundation. Eclipse sin offisielle hjemmeside [Internettside]. Eclipse Fundation [sitert 22.05.2012]. Tilgjengelig fra: http://www.eclipse.org/org/

24. MacCaw A. JavaScript Web Applications. Sebastopol:O'Reilly Media, Inc.; 2011

25. BuiltWith trends. jQuery Usage Trends [Internettside]. BuiltWith trends [sitert 22.05.2012]. Tilgjengelig fra:http://trends.builtwith.com/javascript/JQuery

26. The jQuery Foundation. jQuery Mobile sin offisielle hjemmeside [Internettside]. The jQuery Foundation. [sitert 22.05.2012]. Tilgjengelig fra: http://jquerymobile.com/

27. Wikipedia. Multitier architecture [Internettside]. United States: Wikipedia; 2012 [sitert 22.05.2012]. Tilgjengelig fra: http://en.wikipedia.org/wiki/Multitier_architecture

28. Sun Microsystems, Inc. Java BluePrints Model-View-Controller [Internettside]. United States: Sun Microsystems, Inc; 2002 [sitert 22.05.2012]. Tilgjengelig fra: http://java.sun.com/blueprints/patterns/MVC-detailed.html

29. Withman M. E., Mattord H. J. Principles of Information Security, Thomson Course Technology: 2009 (3. utgave)

30. Saunders A. PhoneGap and iPhone Development [Internettside]. United States: Clearly Innovative; [sitert 22.05.2012]. Tilgjengelig fra: http://blog.clearlyinnovative.com/post/1012434483/phonegap-and-iphone-development

31. Android developers: Designing for security [Internettside]. United States; 2012 [sitert 22.05.2012] Tilgjengelig fra: http://developer.android.com/guide/practices/security.html#UserData

32. Android developers: User Interface: Menus [Internettside]. United States; 2012 [sitert 22.05.2012] Tilgjengelig fra: http://developer.android.com/guide/topics/ui/menus.html

33. Hickson I. Web SQL Database [Internettside]. W3C. [sitert 22.05.2012]. Tilgjengelig fra: http://dev.w3.org/html5/webdatabase/

34. Adobe Systems Inc. Database [Internettside]. United States: Adobe Systems Inc; [sitert 22.05.2012]. Tilgjengelig fra: http://docs.phonegap.com/en/1.7.0/cordova_storage_storage.md.html#Database

35. Recursive Design. jQuery i18n Plugin [Internettside]. Australia:Recursive Design; [sitert 22.05.2012]. Tilgjengelig fra: http://recursive-design.com/projects/jquery-i18n/

36. Williamson A.R. Java Servlets by Example. Greenwich:Manning Publications; 1999

37. Myers G. J. The Art of Software Testing .New Jersey:Word Association; 2012

9 Vedlegg

Vedlegg A: Terminologiliste

- Scrum: Rammeverk for utvikling av programvare
- Sprint: En periode på normal 2-6 uker der en viss mengde funksjonalitet skal implementeres.
- Sprintmøte: Starten på hver sprint. Prioritering og estimering av funksjonalitet som skal implementeres i den kommende sprinten.
- Demomøte: Møte i slutten av hver sprint der funksjonaliteten som er implementert vises fram.
- Retrospektivemøte: Avslutter hver sprint. Tilbakeblikk for å finne ut hvilke feil som ble gjort og hvordan det kan forbedres.
- Produktkø: Består av funksjonaliteten som skal implementeres i systemet
- Sprintkø: Består av de delene av systemet som skal utføres i en sprint
- PhoneGap: Rammeverk for å utvikle app-er for kryssplattform
- jQuery: JavaScript-bibliotek for å traversere HTML-dokumenter, ha kontroll på event-handling, animasjoner og AJAX-funksjoner.
- jQuery Mobile: JavaScript bibliotek for enklere utvikling av app-er.
- Tomcat: En Open Souce implementasjon av Java Servlet og JavaServer Pages teknologier.
- Produkteier: Eieren av systemet som utvikles. Skal prioritere hvilken funksjonalitet som skal utvikles.
- Scrum-team: Gruppen av personer som arbeider for å levere funksjonaliteten.
- Event: Metode som registrerer og utfører en handling når noe bestemt oppstår i systemet. For eksempel klikk på en knapp.
- Serialisere: Kode et objekt, eller felter, til et format som kan lagres, ofte til bits eller en string.
- Native app: En app utviklet i ett bestemt programmeringsspråk, for å kjøre på bestemt plattform/OS. Eks: Android-apps kodes i Android. iPhone apps kodes i Objective-C.

Vedlegg B: Europeisk skademeldingsskjema

Skademelding - motorvogn

Er ingen skylderkjennelse, men klargjøring av skadeforløpet med henblikk på raskere oppgjør.

Undertegnes av begge førerne

1. Skadedato	Kl.	2. Skadested (Gate/gatekryss, veinr.)	Kommune.	3. Personskade Nei ☐ Ja ☐ ★
4. Materiell skade utover bil A og B Nei ☐ Ja ☐ ★		5. Vitner Navn, adresse og telefon. (Notér om vitnet var passasjer i A eller B.)		

Kjøretøy A	A	12. Sett kryss i aktuelle felt	B	Kjøretøy B
6. Forsikringstaker Etternavn (Store bokstaver) Fornavn Adresse Poststed Telefon Oppgavepliktig for merverdiavgift Nei ☐ Ja ☐	☐	1 Sto stille 1	☐	6. Forsikringstaker Etternavn (Store bokstaver) Fornavn Adresse Poststed Telefon Oppgavepliktig for merverdiavgift Nei ☐ Ja ☐
7. Kjøretøy Merke, type Kjennetegn (reg.nr.)	☐	2 Satte seg i bevegelse 2	☐	7. Kjøretøy Merke, type Kjennetegn (reg.nr.)
8. Forsikringsselskap Polisenr. Agentur (Gjelder ikke **norske forsikringstakere**) Grønt kort nr. (Skade utenfor Norden) Grønt kort gyldig til Kaskoforsikring Nei ☐ Ja ☐	☐	3 Var i ferd med å stoppe 3	☐	8. Forsikringsselskap Polisenr. Agentur (Gjelder ikke **norske forsikringstakere**) Grønt kort nr. (Skade utenfor Norden) Grønt kort gyldig til Kaskoforsikring Nei ☐ Ja ☐
9. Fører Etternavn (Store bokstaver) Fornavn Adresse Førerkort nr. Klasse Utstedt av Førerkort gyldig til Førers alder Kjønn K ☐ M ☐	☐	4 Kjørte **ut** fra parkeringsplass, privat område, gårdsvei 4	☐	9. Fører Etternavn (Store bokstaver) Fornavn Adresse Førerkort nr. Klasse Utstedt av Førerkort gyldig til Førers alder Kjønn K ☐ M ☐
	☐	5 Kjørte **inn** på parkeringsplass, privat område, gårdsvei 5	☐	
	☐	6 Kjørte **inn** i en rundkjøring 6	☐	
	☐	7 Kjørte **i** en rundkjøring 7	☐	
	☐	8 Kjørte på bakfra i samme retning og samme kjørefelt 8	☐	
	☐	9 Kjørte i samme retning i annet kjørefelt 9	☐	
	☐	10 Skiftet kjørefelt 10	☐	
	☐	11 Kjørte forbi 11	☐	
	☐	12 Svingte til **høyre** 12	☐	
	☐	13 Svingte til **venstre** 13	☐	
	☐	14 Rygget 14	☐	
	☐	15 Kom inn på del av veien bestemt for trafikk i motsatt retning 15	☐	
	☐	16 Kom fra høyre (i et kryss) 16	☐	
	☐	17 Fulgte ikke varsel om vikeplikt 17	☐	
	☐	Antall avkryssede felt	☐	

13. Situasjonsriss

Tegn opp: 1. Gater, veier. - 2. Kjøretøyene A og B's retning før kollisjonen. 3. A og B's posisjoner i kollisjonsøyeblikket. - 4. Trafikkskilt - 5. Gate/veinavn. - 6. Veibredde.

10. Markér med pil (➧) det første berørings-punkt ved kollisjonen	11. Synlige skader	14. Tilleggsopplysninger (Ytterligere opplysninger skrives på baksiden)	15. Førernes underskrifter A B	14. Tilleggsopplysninger (Ytterligere opplysninger skrives på baksiden)	10. Markér med pil (➧) det første berørings-punkt ved kollisjonen	11. Synlige skader

★ *Gi nærmere opplysninger om personskade og materiell skade på baksiden - spørsmål 32 og 33.*

Det må ikke foretas rettelser eller tilføyelser på denne siden etter at blankettene er undertegnet og skilt fra hverandre.

Se forsikringstakerens redegjørelse på baksiden.

Vedlegg C: Eksempelkode

Dette vedlegget inneholder eksempel på kode som vi selv har skrevet. Noe er hele filer og noe er utdrag fra filer.

Konfigurerbarefelter

Dette kodeeksempelet er hentet fra filen Database.java på server og er metodene for å sette og hente de feltene som skal vises/skjules for en firmabruker.

```
/**
 * Setter konfigurerbare felter (felter som skal skjules)
 * for alle brukere som har uid som skadeansvarlig
 * @param uid id-en til den skadeansvarlige
 * @param feltArray en string på formen felt1,felt2,felt3 som skal lagres
 * @return true hvis ok ellers false
 */
public boolean setKonfigurerbarhet(int uid, String feltArray)
{
    boolean res = false;
    PreparedStatement stmt = null;
    try
    {
        String sql ="UPDATE brukere SET skjultefelter='%s' WHERE id=%s";
        sql=String.format(sql, feltArray, uid);

        stmt = con.prepareStatement(sql);
        stmt.executeUpdate();

        res = true;
    }
    catch (Exception e) {
          System.out.println("Error - Database.setKonfigurerbarhet(): "+
e.getMessage());
          e.printStackTrace();
    }
    finally {
        DBConnection.closeStmt(stmt);
        DBConnection.closeConnection(con);
    }

    return res;
}

/**
 * Finner de feltene som skal skjules fra en skademelding
 * @param forSkadeansvarlig om uid er en skadeansvarlig eller ikke
 * @param uid ID til skadeansvarlig
 * @return En string på formen felt1,felt2,felt3 som angir de feltene
 * som ikke skal vises i skademeldingen
 */
public String getKonfigurerbarhet(int uid, boolean forSkadeansvarlig)
{
    String felter = "";
    PreparedStatement stmt = null;
```

```
    String sql = String.format("SELECT b2.skjultefelter FROM brukere b1 " +
            "INNER JOIN brukere b2 ON b1.idAnsvarlig=b2.id " +
            "WHERE b1.id=%s;", uid);
    if (forSkadeansvarlig)
        sql = "SELECT skjultefelter FROM brukere WHERE id=" + uid;

    try
    {
        stmt = con.prepareStatement(sql);
        ResultSet res =  stmt.executeQuery();

        if (res.next())
            felter = res.getString(1);

    }
    catch (Exception e) {
          System.out.println("Error - Database.getKonfigurerbarhet(): "+
e.getMessage());
          e.printStackTrace();
    }
    finally {
        DBConnection.closeStmt(stmt);
    }

    return felter;
}
```

Signer skademelding

Eksempelet er hentet fra filen signerskademelding.jsp og lar en bruker signere en skademelding.

```
<%@page language ="java" import="skademelding.dataaksess.*" %>
<%
String skademeldingIDparam = (String)  request.getParameter("smid");
String acceptSigneringparam = (String)  request.getParameter("accept");
String uidParam = (String) session.getAttribute("uid");

if (uidParam == null || skademeldingIDparam == null)
    out.println("Du må logge på for å ha tilgang til denne siden!");
else
{
    int skademeldingID = Integer.parseInt(skademeldingIDparam);
     int uid = Integer.parseInt(uidParam);

    boolean akseptert = false;
    if (acceptSigneringparam.equals("true"))
        akseptert = true;
```

```
    //Kjører kall mot databasen for å signere skademeldingen.
    Database db = new Database();
    boolean res = db.signerSkademelding(skademeldingID, uid, akseptert);

    if (res)
        out.println("1"); //Skademeldingen ble signert
    else
        out.println("0"); //Skademeldingen ble IKKE signert

}
%>
```

Tegnebrett

Dette eksempelet viser koden i Tegnebrett.js som brukes av både appen og web.

```
/**
 * Klassen oppretter et tegnebrett det kan legges til objekter på.
 * @class
 * @param {string} div id-en til den div-tagen som tegnebrettet skal ligge i.
 * @param {string} enhet app eller web
 */
function Tegnebrett(div, enhet)
{
    this.enhet = enhet; //
    this.zoom = 1.0;
    this.size = {w :1000, h:1000}; //Størrelsen på tegnebrettet (ikke stagen)

    this.objekter = new Situasjonsobjekter(this.size.w, this.size.h);
    this.kryssvalgt = null; //Holder orden på den veien/krysset som er i bruk
    this.verdenRektangel = null; //Rektangelet som er selve brettet der
tegningen er
    this.merketObjekt = null; //Det objektet som sist er trykket på/lagt til

    var stageSize = this._getNyStageSize();
    this.stage = new Kinetic.Stage(div, stageSize.w, stageSize.h);
    this.layer = new Kinetic.Layer();
    this.layer.draggable(true);
    this.stage.add(this.layer);
    this._events();

    this.addVerden();
    this._resize();

    this.roterHoyreKnappID = "";
    this.roterVenstreKnappID = "";
    this.fjernKnappID = "";
    this.zoomUtKnappID = "";
    this.zoomInnKnappID = "";

    this.tooltip =new Kinetic.Text({
        text: "...",
        fontFamily: "Calibri",
        fontSize: 40,
        padding: 5,
```

```
        visible: true,
        fill: "black",
        alpha: 0.75,
        textFill: "white"
    });

    this.layer.add(this.tooltip);
}

//#################################################################
//# PUBLIC METODER ################################################
//#################################################################

/**
 * Legger til et vanlig objekt på stagen ut fra type-parameteren
 * @function
 * @param {string} type Angir hvilken type objekt som skal legges til
 * @param {int} x Hvor objektet skal plasseres horisontalt
 * @param {int} y Hvor objektet skal plasseres vertikalt
 * @param {int} r Rotasjonen på objektet

 */
Tegnebrett.prototype.add = function(type, x, y, r, text)
{
    var nyttObjekt = null;
    switch (type) {


    //Veger/vegkryss
    case "tkryss":
        this.addKryss(this.objekter.tkryss());
        break;
    case "4kryss":
        this.addKryss(this.objekter.firekryss());
        break;
    case "rettveg":
        this.addKryss(this.objekter.rettVeg());
        break;

    //Kjøretøy
    case "personbil":
        this.addObjekt(this.objekter.personbil(null));
        break;
    case "personbilA":
        this.addObjekt(this.objekter.personbil("A"));
        break;
    case "personbilB":
        this.addObjekt(this.objekter.personbil("B"));
        break;
    case "lastebil":
        this.addObjekt(this.objekter.lastebil(null));
        break;
    case "lastebilA":
        this.addObjekt(this.objekter.lastebil("A"));
        break;
    case "lastebilB":
        this.addObjekt(this.objekter.lastebil("B"));
```

```
        break;

    case "bildePersonbil":
        //this.addObjekt(this.objekter.personbil(null));
        this.objekter.bil1(callbackBildeReady);  //callbackBildeReady-
funksjonen er bare laget for skademerking!!!
        break;

    case "bildeVarebil":
        this.objekter.varebil(callbackBildeReady);//callbackBildeReady-
funksjonen er bare laget for skademerking!!!
        break;
    case "skademerke":
        this.addObjekt(this.objekter.skademerke(text));
        break;
    default:
        break;
    }

    //Flytter objektet til rett plassering
    //hvis parameterne x og y er spesifisert
    if (x != undefined && y != undefined && this.merketObjekt != null)
    {
         this.merketObjekt.setPosition(x,y);
         this.layer.draw();
    }

    //Setter rotasjon på objektet
    if (r != undefined && this.merketObjekt != null)
    {
        this.merketObjekt.setCenterOffset(this.merketObjekt.w / 2 * this.zoom
, this.merketObjekt.h / 2 * this.zoom);
        this.merketObjekt.rotate(r);
        this.layer.draw();
    }

    //Lukker dialogboksen hvis den er åpen
    $('.ui-dialog').dialog('close');

}

/**
 * Bytter ut det objektet (vegkryss) som ligger helt nederst på Stage-en.
 * @function
 * @param objekt Det objektet som skal bli det nye bakgrunnsobjektet
(vegkrysset)
 */
Tegnebrett.prototype.addKryss = function(objekt){
    //this._addEvent(objekt);

    if (this.kryssvalgt != null) this.layer.remove(this.kryssvalgt);
    this.kryssvalgt = null;

    this.layer.add(objekt);
    this.kryssvalgt = objekt;
    //Flytter objektet til midten av layeret
    objekt.setPosition((this.size.w / 2 - objekt.w / 2) * this.zoom,
```

```
(this.size.h / 2 - objekt.h / 2) * this.zoom);
    objekt.setZIndex(1);
    this.layer.draw();

}

/**
 * Legger til selve objektet(shapen) på stage-en
 * og flytter det til sentrum av skjermen
 * @function
 * @param {Node} objekt Shapen som skal legges til på Stage-en
 */
Tegnebrett.prototype.addObjekt = function(objekt){
    this._addEvent(objekt);

    this.layer.add(objekt);

    //#Flytter objektet til sentrum av skjermen
    var size = this._getNyStageSize();
    var x = 0;
    var y = 0;

    if (size.w > this.size.w * this.zoom)
        x = (this.size.w / 2 - objekt.w / 2) * this.zoom - this.layer.x;
    else
        x = (size.w / 2 - objekt.w / 2) - this.layer.x;

    if (size.h > this.size.h * this.zoom)
        y = (this.size.h / 2 - objekt.h / 2) * this.zoom - this.layer.y;
    else
        y = (size.h / 2 - objekt.h / 2) - this.layer.y;
    objekt.setPosition(x,y);
    //###

    this.layer.draw();
    this._setMerketObjekt(objekt);
}


/**
 * Returnerer et array med alle punkter på layeret
 * @function
 * @returns {Array} Returnerer et array der hvert element har x, y og
r(rotasjon)
 */
Tegnebrett.prototype.getAllePunkter = function(){
    var punkter = {};

    var children = this.layer.getChildren();
    for (var a in children)
    {
        //Legger alle til utenom verdenRektangelet
        if (children[a] != this.verdenRektangel)
        {
            var textt = children[a].text;
            if(textt == undefined)
                textt = "";
```

```
            var xy = children[a].getPosition();
            var pt = {
                    name : children[a].name,
                    x : xy.x / this.zoom,
                    y : xy.y / this.zoom,
                    r : children[a].getRotation(),
                    text : textt
            };
            punkter[a] = pt;
        }
    }

    return punkter;

}

/**
 * Fjerner alle objekter på tegnebrettet og legger inn de som er spesifisert
 * i punkterJSON
 * @function
 * @param punkterJSON et array med punter (name, x, y) som er json-stringifya
 */
Tegnebrett.prototype.lastInnObjekter = function(punkterJSON){
    this.tooltip.hide();
    console.log(punkterJSON);
    //fjerner alle objekter sånn at vi kan legge på de som skal lastes inn
    this.fjernAlleObjekter();
    this._resize();//Resizer tegnebrettet sånn at zoom-en blir etter
størrelsen på skjermen

    if (punkterJSON == undefined || punkterJSON == null || punkterJSON == ""
|| punkterJSON == "{}")
    {
        //Denne metoden må defineres i alle klasser
(Situasjonsriss/Skademerking) som arver fra denne (Tegnebrett).
        this.leggInnStandardObjekter();
        return;
    }

    var punkterArray = jQuery.parseJSON(punkterJSON); //Oversetter fra JSON-
string til et Array

    for (var a in punkterArray)
    {
        this.add(punkterArray[a].name, punkterArray[a].x * this.zoom,
punkterArray[a].y * this.zoom, punkterArray[a].r, punkterArray[a].text)
    }

}

/**
 * Fjerner alle objekter utenom verdenRektangel
 * @function
 */
Tegnebrett.prototype.fjernAlleObjekter = function(){
    //Løper gjennom alle noder på layeret og fjerner alle utenom
verdenrektangelet
```

```
    //Fjerner også kryssvalgt
    var antall = this.layer.children.length;//Antall objekter
    for (var n=antall-1; n >= 0; n--) {
        if (this.layer.children[n] != this.verdenRektangel &&
this.layer.children[n] != this.tooltip)
            this.layer.remove(this.layer.children[n]);
     }

    this.kryssvalgt = null;
    this.layer.draw(); //Oppdaterer GUI
}

/**
 * Legger til, på Stage-en, den shape-en der alt skal tegnes oppå.
 * @function
 */
Tegnebrett.prototype.addVerden = function()
{
    var firkant = new Kinetic.Rect({
        x: 0,
        y: 0,
        width: 1000,
        height: 1000,
        fill: "white",//"#FFE7F1", //"#00D2FF",
        stroke: "black",
        strokeWidth: 4
    });

    this.layer.add(firkant);
    this.layer.draw();
    this.verdenRektangel = firkant;
}

/**
 * Sletter det objektet(shape-en) som er merket det objektet(shape-en) fra
Stage-en
 * @function
 */
Tegnebrett.prototype.slettMerketObjekt = function(){
    if (this.merketObjekt == null)
        return;

    this.layer.remove(this.merketObjekt);
    this.tooltip.hide();
    this.layer.draw();


    var children = this.layer.getChildren();
    if (children.length > 2) //2 fordi det ikke skal gå ann å fjerne objektet
i bakgrunnen (bilen/veikrysset)
        this._setMerketObjekt(children[children.length-1]);    //Setter den
øverste på layeret til merket
}

/**
 * Zoomer inn tegnebrettet
 * @function
```

```
 */
Tegnebrett.prototype.zoominn = function(){
    this.stage.setScale(this.zoom = this.zoom + 0.1);
    this._setCenterOffsetForAlleObjekter();//Gjør at objektene holder seg
riktig i forhold til hverandre
    this.stage.draw();

    //Disabler zoom inn knapp når du har zoomet veldig langt inn (verden sin
bredde > 3000)
    if (this.size.w * this.zoom > 3000)
        $(this.zoomInnKnappID).button('disable');

    $(this.zoomUtKnappID).button('enable'); //Forsikrer oss om at zoom ut
knappen er enablet
}

/**
 * Zoomer ut tegnebrettet
 * @function
 */
Tegnebrett.prototype.zoomut = function(){
    this.stage.setScale(this.zoom = this.zoom - 0.1);
    this._setCenterOffsetForAlleObjekter();//Gjør at objektene holder seg
riktig i forhold til hverandre
    this.stage.draw();

    //Disabler zoom ut knapp når hele brettet er synlig på skjermen
    var winSize = this._getNyStageSize();
    if (this.size.w * this.zoom <= winSize.w && this.size.h * this.zoom <=
winSize.h)
        $(this.zoomUtKnappID).button('disable');
    $(this.zoomInnKnappID).button('enable'); //forsikrer oss om at zoom inn
knappen er enablet
}

/**
 * Setter offset for alle objekter til sentrum av dem.
 * Dette er viktig å gjøre før man skal zoome slik at de
 * vil holde seg rett i forhold til hverandre.
 * @function
 */
Tegnebrett.prototype._setCenterOffsetForAlleObjekter = function(){
    this.layer.setCenterOffset(this.size / 2 * this.zoom , this.size / 2 *
this.zoom);

    var children = this.layer.getChildren();
    for (var a in children)
        children[a].setCenterOffset(children[a].w / 2 * this.zoom ,
children[a].h / 2 * this.zoom);
}

/**
 * Roterer et objekt til venstre eller høyre
 * @function
 * @param {bool} venstre true for å rotere til venstre, ellerst høyre
 */
Tegnebrett.prototype.roter = function(venstre){
```

```
    if (this.merketObjekt == null)
        return;

    var th = Math.PI / 10;
    if (venstre == true)
        th = th * -1;

    this.merketObjekt.setCenterOffset(this.merketObjekt.w / 2 * this.zoom ,
this.merketObjekt.h / 2 * this.zoom);
    this.merketObjekt.rotate(th);
    this.layer.draw();
}

//#################################################################
//# PRIVATE METODER ###############################################
//#################################################################

/**
 * Returnerer størrelsen på skjermen
 * @function
 * @returns {Punkt (w,h)}
 */
Tegnebrett.prototype._getNyStageSize = function(){

    if (this.enhet == "web")
    {
        return {
            w : 500,
            h : 500
        };
    }
    else
    {
        return {
            w : window.innerWidth, // - 20,
            h : window.innerHeight - 45
        };
    }
}

/**
 * Setter et objekt som merket. Handlinger som senere gjøres (feks: fjern)
vil
 * bli gjort på dette objektet
 * @function
 * @param {Node} objekt Objektet(Shapen) som skal bli det valgte/merkede
objektet
 */
Tegnebrett.prototype._setMerketObjekt = function(objekt){
    if(objekt == this.tooltip)
        return;
    if(this.merketObjekt != null)
    this.merketObjekt.setFill(this.merketObjekt.color);

    this.merketObjekt = objekt;
    this.merketObjekt.setFill("blue");
    this.layer.draw();
```

```
    //Skjuler/viser aktuelle menyknapper
    if (objekt == null)
    {
        try {
            //Skjuler knappene
            $(this.roterHoyreKnappID).button('disable');
            $(this.roterVenstreKnappID).button('enable');
            $(this.fjernKnappID).button('disable');
        } catch (e) {
        }

    }
    else
    {
        if (objekt.text == undefined || objekt.text =="")
            this.tooltip.hide();

        try {
            //Viser knappene
            $(this.roterHoyreKnappID).button('enable');
            $(this.roterVenstreKnappID).button('enable');
            $(this.fjernKnappID).button('enable');
        } catch (e) {
        }
    }

}

/**
 * Legger til standard-events som trengs på et objekt
 * @function
 * @param {Shape} shapeObjekt
 */
Tegnebrett.prototype._addEvent = function(shapeObjekt){
    var that = this;
    var timer = null;
    shapeObjekt.draggable(true);

    shapeObjekt.on("mousedown touchstart", function(evt){
        var that2 = this;

        // Viser en dialogboks med kommentaren
        timer = window.setInterval(function() {
            console.log(shapeObjekt.text);

                $('<div>').simpledialog2({
                     mode: 'blank',
                        headerText: 'Kommentar:',
                        headerClose: true,
                        dialogAllow: true,
                        animate: false,
                        callbackClose: function(){
                                  that2.text = $('textarea#comment').val();
                        },
                        blankContent :
                          "<textarea id='comment' rows='5'
```

```
style='resize:none'>"+that2.text+"</textarea>"+
                            "<a rel='close' id='close' data-role='button'
href='#'>Close</a>"
                    });
                window.clearInterval(timer); //Stoppre timeren
               }, 2000);

        that.layer.draggable(false);
        that._setMerketObjekt(this);


    });

    shapeObjekt.on("dragmove touchmove", function(evt){
        window.clearInterval(timer);
    });

    shapeObjekt.on("dragend touchend", function(evt){
        that.layer.draggable(true);
    });
}

/**
 * Legger til events som trengs til stage-en
 * @function
 */
Tegnebrett.prototype._events = function()
{
    var that = this;

    //Når vi endrer orientation på telefonen resizer og skalerere
    $(window).resize(function() {
        that._resize();
    });

    //Fjerner menyen når du trykker på stagen/skjermen
    this.stage.on("mousedown touchstart", function(evt){
        that.toggleMenyKanpp(true);
    });

    //Sånn at vi bare kan dra opp/ned når verden er zoomet langt ut
    this.layer.on("dragstart", function(evt){

        var winSize = that._getNyStageSize();
        if (that.size.w * that.zoom <= winSize.w)
            that.layer.draggableX(false);
        else
            that.layer.draggableX(true);

        if (that.size.h * that.zoom <= winSize.h)
            that.layer.draggableY(false);
        else
            that.layer.draggableY(true);
    });

    //Etter at vi har dradd på noe så må vi tegne opp på nytt
    this.layer.on("mousemove touchmove dragend", function(evt){
```

```
        var w = that.stage.width - (that.verdenRektangel.getWidth() *
that.zoom);
        var h = that.stage.height - (that.verdenRektangel.getHeight() *
that.zoom);

        var needDraw = false;

        if (this.x < w)
        {
            this.x = w;
            needDraw = true;
        }

        if (this.x > 0)
        {
            this.x = 0;
            needDraw = true;
        }

        if (this.y < h)
        {
            this.y =h;
            needDraw = true;
        }

        if (this.y > 0)
        {
            this.y = 0;
            needDraw = true;
        }

        if (needDraw)
            this.draw();
    });


}

/**
 * Oppdaterer størrelsen på Stage-en i forhold til skjermen
 * @function
 */
Tegnebrett.prototype._resize = function(){
    var nyStageSize = this._getNyStageSize();

    this.stage.setSize(nyStageSize.w,nyStageSize.h);

    //Skalerer sånn at brettet/verden er like bredt som bredden på skjermen
    var w = (nyStageSize.w) / this.size.w;
    this.stage.setScale(this.zoom = w);
    this._setCenterOffsetForAlleObjekter();//Gjør at objektene holder seg
riktig i forhold til hverandre
    this.stage.draw();

    if (situasjonsriss != null)
    {
        if (nyStageSize.w > nyStageSize.h)
```

```
            situasjonsriss.menyOverstHoyre();
        else
            situasjonsriss.menyNederstVenstre();
    }

}
```

Vedlegg D: SQL-kommandoer

Dette vedlegget inneholder er SQL-kommandoer for databasen på serveren.

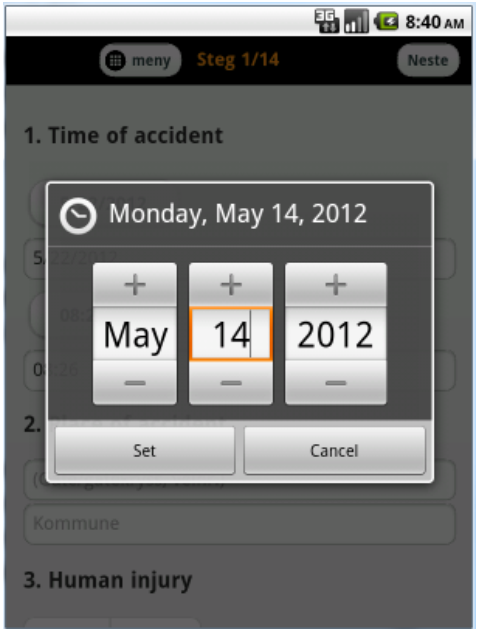
```
CREATE DATABASE if not exists skademeldingdb;

CREATE TABLE skademelding (
    id INTEGER NOT NULL AUTO_INCREMENT,
    id_brukerA INTEGER NULL DEFAULT -1,
    id_brukerB INTEGER NOT NULL DEFAULT -1,
    id_kjoretoyA INTEGER NOT NULL,
    id_kjoretoyB INTEGER NOT NULL,
    id_supplerendeA INTEGER NOT NULL,
      id_supplerendeB INTEGER NOT NULL,

    signaturA TINYINT(1) NOT NULL DEFAULT 0,
      signaturB TINYINT(1) NOT NULL DEFAULT 0,

    skadedato DATE,
      tidspunkt TIME,
    skadested VARCHAR(45) NOT NULL DEFAULT '',
    kommune VARCHAR(45) NOT NULL DEFAULT '',
    personskade TINYINT(1),
    annen_materiell_skade VARCHAR(45) NOT NULL DEFAULT '',
    vitner VARCHAR(120) NOT NULL DEFAULT '',
      vitner_pass VARCHAR(120) NOT NULL DEFAULT '',
      situasjonsriss VARCHAR(256) NOT NULL DEFAULT '',
      kommentar VARCHAR(500) NOT NULL DEFAULT '',

    PRIMARY KEY (`id`)
);

CREATE TABLE kjoretoy (
    id INTEGER NOT NULL AUTO_INCREMENT,

    forsikringstaker_fornavn VARCHAR(45) NOT NULL DEFAULT '',
    forsikringstaker_etternavn VARCHAR(45) NOT NULL DEFAULT '',
    forsikringstaker_adresse VARCHAR(45) NOT NULL DEFAULT '',
    forsikringstaker_poststed VARCHAR(45) NOT NULL DEFAULT '',
    forsikringstaker_telefon VARCHAR(45) NOT NULL DEFAULT '',
    forsikringstaker_mvapliktig TINYINT(1) NOT NULL DEFAULT 0,

    kjoretoy_merke VARCHAR(45) NOT NULL DEFAULT '',
    Kjoretoy_regnr VARCHAR(45) NOT NULL DEFAULT '',

    forsikringsselskap_navn VARCHAR(45) NOT NULL DEFAULT '',
    forsikringsselskap_polisenr VARCHAR(45) NOT NULL DEFAULT '',
    forsikringsselskap_agentur VARCHAR(45) NOT NULL DEFAULT '',
    forsikringsselskap_grontkort VARCHAR(45) NOT NULL DEFAULT '',
    forsikringsselskap_grontkort_gyldig_til DATE,

    forer_etternavn VARCHAR(45) NOT NULL DEFAULT '',
    forer_fornavn VARCHAR(45) NOT NULL DEFAULT '',
    forer_adresse VARCHAR(45) NOT NULL DEFAULT '',
    forer_forerkortnr VARCHAR(45) NOT NULL DEFAULT '',
```

```
    forer_forerkort_klasse VARCHAR(45) NOT NULL DEFAULT '',
    forer_forerkort_utstedt_av VARCHAR(45) NOT NULL DEFAULT '',
    forer_forerkort_gyldig_til DATE,
    forer_alder INTEGER,
    forer_kjonn enum ('k', 'm') NOT NULL DEFAULT 'k',

    forste_beroringspunkt VARCHAR(45) NOT NULL DEFAULT '',
    synlige_skader VARCHAR(45) NOT NULL DEFAULT '',

    tilleggsopplysninger VARCHAR(120) NOT NULL DEFAULT '',
      forsikringsselskap_kasko TINYINT(1) NOT NULL DEFAULT 0,

    v1 TINYINT(1) NOT NULL DEFAULT 0,
    v2 TINYINT(1) NOT NULL DEFAULT 0,
    v3 TINYINT(1) NOT NULL DEFAULT 0,
    v4 TINYINT(1) NOT NULL DEFAULT 0,
    v5 TINYINT(1) NOT NULL DEFAULT 0,
    v6 TINYINT(1) NOT NULL DEFAULT 0,
    v7 TINYINT(1) NOT NULL DEFAULT 0,
    v8 TINYINT(1) NOT NULL DEFAULT 0,
    v9 TINYINT(1) NOT NULL DEFAULT 0,
    v10 TINYINT(1) NOT NULL DEFAULT 0,
    v11 TINYINT(1) NOT NULL DEFAULT 0,
    v12 TINYINT(1) NOT NULL DEFAULT 0,
    v13 TINYINT(1) NOT NULL DEFAULT 0,
    v14 TINYINT(1) NOT NULL DEFAULT 0,
    v15 TINYINT(1) NOT NULL DEFAULT 0,
    v16 TINYINT(1) NOT NULL DEFAULT 0,
    v17 TINYINT(1) NOT NULL DEFAULT 0,

    skademerking VARCHAR(256) NOT NULL DEFAULT '',

    PRIMARY KEY (`id`)
);

CREATE TABLE supplerende (
    id INTEGER NOT NULL AUTO_INCREMENT,
  sitkmt VARCHAR(45) NOT NULL DEFAULT '',
  krasjkmt VARCHAR(45) NOT NULL DEFAULT '',
  grenskmt VARCHAR(45) NOT NULL DEFAULT '',
  vei VARCHAR(45) NOT NULL DEFAULT '',
  fore VARCHAR(45) NOT NULL DEFAULT '',
  vaer VARCHAR(45) NOT NULL DEFAULT '',
  temp VARCHAR(45) NOT NULL DEFAULT '',
  lys VARCHAR(45) NOT NULL DEFAULT '',
  utstyr_brukt VARCHAR(45) NOT NULL DEFAULT '',
  kjetting BOOL DEFAULT false,
  vinterdekk BOOL DEFAULT false,
  piggdekk BOOL DEFAULT false,
  bilbelte BOOL DEFAULT false,
  signal_lys BOOL DEFAULT false,
  signal_horn BOOL DEFAULT false,
  signal_blinklys BOOL DEFAULT false,
  ovelseskjoring BOOL,
  tillatelse BOOL,
  leasing BOOL,
  ansvarlig VARCHAR(45) NOT NULL DEFAULT '',
```

```
  alkohol bool,
  tyveri bool,
  km_stand VARCHAR(45) NOT NULL DEFAULT '',
  andre_opplysninger VARCHAR(256) NOT NULL DEFAULT '',
  politi BOOL,
  verksted BOOL,
  verkstednavn VARCHAR(45) NOT NULL DEFAULT '',
  politikammer VARCHAR(60) NOT NULL DEFAULT '',
  annen_materiell_skade VARCHAR(128) NOT NULL DEFAULT '',
  eier_skadet_materiell VARCHAR(128) NOT NULL DEFAULT '',


  PRIMARY KEY (`id`)
);

CREATE TABLE skadedepersoner (
    id INTEGER NOT NULL AUTO_INCREMENT,
    id_supplerende INTEGER,
  skadet_navn VARCHAR(45) NOT NULL DEFAULT '',
  skadet_fodt DATE,
  skadet_yrke VARCHAR(45) NOT NULL DEFAULT '',
  skadet_rolle VARCHAR(45) NOT NULL DEFAULT '',
  behandlende_lege_sykehus VARCHAR(45),

    PRIMARY KEY (`id`)
);

CREATE TABLE brukere (
    id INTEGER NOT NULL AUTO_INCREMENT,
    idAnsvarlig INTEGER NOT NULL DEFAULT -1,
    brukernavn VARCHAR(45) NOT NULL UNIQUE,
    passord VARCHAR(32) NOT NULL,
    aktiveringskode VARCHAR(45) NOT NULL,
    type ENUM('privat', 'firmabruker', 'skadeansvarlig') NOT NULL,
    skjultefelter VARCHAR(500) NOT NULL DEFAULT '',

    PRIMARY KEY (`id`)
);

CREATE TABLE invitasjoner (
    id INTEGER NOT NULL AUTO_INCREMENT,
    brukerAid INTEGER NOT NULL,
    brukerBid INTEGER NOT NULL,
    skademeldingsID INTEGER NOT NULL,

    PRIMARY KEY (`id`)
);
```

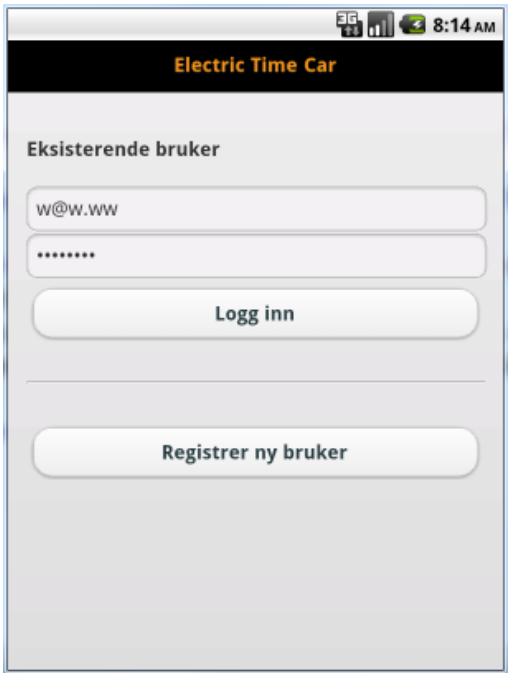
Vedlegg E: Skjermbilder

Skjermbilde av datovelger:



Ny skademelding gir mulighet til å velge to involverte kjøretøy, egenskade, inviter motpart.eller hent invitasjon:

Innloggingsskjerm



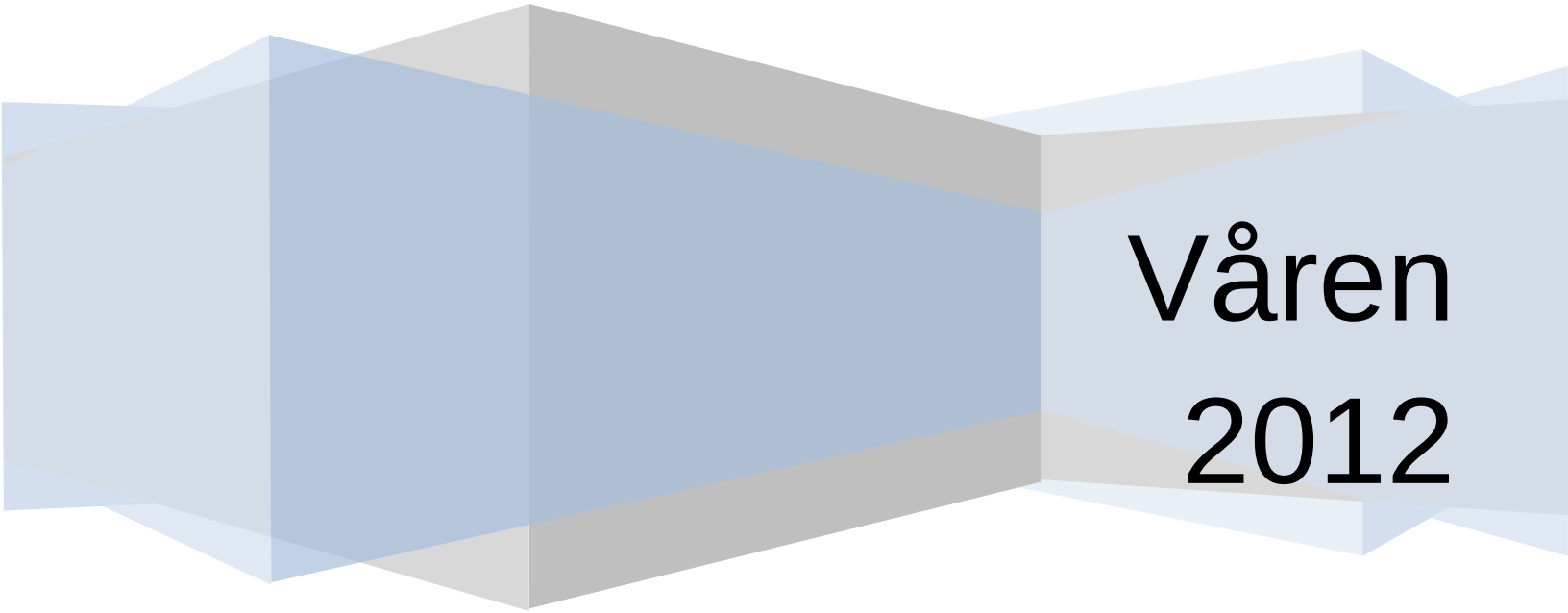
Vedlegg F: Prosjektplan

I dette vedlegget finner du prosjektplanen (forprosjektet). Innholdsfortegnelsens sidetall er ikke rett fordi vi følger denne rapportens sidetall.

Prosjektplan

Mobil skademeldingsmodul

Arve Eidjord
Øyvind Kongsengen



Innhold

1. MÅL OG RAMMER

1.1. Bakgrunn

Vi ser stadig at ny teknologi gir nye muligheter og større effektivitet. Mye av det som før ble behandlet på papir kan i dag gjøres digitalt, for eksempel via en PC eller en smarttelefon.
ETC, som er leverandør av løsninger til bilhold ønsker en slik digitalisering ved rapportering av skademeldinger på biler. Dette må i dag gjøres ved utfylling av skjemaer på papir. For bedrifter og i offentlig sektor der det finnes bilparker vil det være en utfordrende jobb å hente inn slike skademeldinger og sikre at de blir skrevet dersom det oppstår bulker eller småskader på bilene. Slike skader koster bileiere store summer årlig.

Det er derfor ønskelig å utforme et system som kan gjøre skademeldingsprosessen enklere ved at all nødvendig informasjon kan sendes fra smarttelefonen på stedet der uhellet oppstår, til godkjenning fra eier av bilen. For så at skademeldingen kan sendes videre til forsikringsselskapet.

1.2. Prosjektmål

Effektmål

Med den ferdige løsningen er den ønskede effekten å få flere bilførere til å sende inn skademeldingsskjemaer dersom de er ute for et uhell, samt at en slik prosess skal være en forenkling både for bilførerne og de som administrerer bilene ved en eventuell bilpark. Dette vil også føre til effektivisering og besparelse av tid og penger.

Resultatmål

Målet med oppgaven er å utvikle et system for behandling av skademeldinger. Dette systemet skal bestå av en applikasjon for mobile enheter og et webgrensesnitt som skal ta seg av etterbehandling av skademeldingen. Når oppgaven er ferdig skal ETC kunne tilby løsningen til sine kunder.

Læringsmål

Etter prosjektet skal vi ha tilegnet oss ny kunnskap om programmering for mobile enheter og spesielt hvordan dette kan løses for kryssplattform. Lære hvordan denne kan kommunisere med en sentral server og database, og i tillegg ha en bredere erfaring innenfor programmering av webgrensesnitt.

1.3. Rammer

- Løsningen skal utvikles eller klargjøres for kryssplattform.
- Løsningen skal ha et billedlig rapporteringsgrensesnitt.
- Etterbehandleren(webgrensesnittet) skal kunne integreres med oppdragsgivers eksisterende system, CarAdmin.

2. OMFANG

2.1. Oppgavebeskrivelse

Vår oppgave er å lage et system for behandling av skademeldinger for kjøretøy. Den består i hovedsak av to deler.

Det skal utvikles en applikasjon for bruk på mobile enheter (smartphones/tablets). Det er tiltenkt både privatbrukere og bedriftskunder og skal være et alternativ til papirutgaven som brukes som standard i dag. Dette betyr at man ved en ulykke, skal kunne registrere alle nødvendige opplysninger om de involverte, bilinformasjon, skadeomfanget, og stedet hvor ulykken skjedde, slik at det til sammen utgjør et komplett skademelingsskjema. Formalia som navn og bilinformasjon eller lignende, skal hentes fra sentral server. Den mobile applikasjonen skal ha et ”billedlig” rapporteringsgrensesnitt, og etter endt skademeldingsutfylling, sendes data til sentral server for etterbehandling, eventuell korreksjoner og attestasjon før den videre sendes til forsikringsselskapet, verkstedet, etc. Skjemaet skal være konfigurerbart slik at det kan tilpasses kundens behov, dette gjelder spesielt for bedrifter hvor alle poster som finnes i et vanlig skadeskjema kanskje ikke vil være nødvendig.

Det skal også utvikles et webgrensesnitt for etterbehandlingen, og dette skal være et selvstendig grensesnitt som videre skal kunne brukes i tredjepartsløsninger. Bruker skal her kunne logge seg inn, hente ut igjen innsendte opplysninger/data, og gjøre eventuelle endringer før skjemaet blir godkjent for innsending til forsikringsselskap. Oppdragsgiver vil legge premisser for dette grensesnittet slik at modulen kan integreres i det eksisterende CarAdmin systemet.

I tillegg til å utvikle løsningen, skal vi også diskutere de valgene vi gjør underveis og vurdere de opp mot andre mulige løsninger. Slik at vi ved prosjektets slutt også har en dokumentasjon på arbeidsprosessen, satt i et faglig perspektiv.

Valg av plattform er ikke fastlagt, men det skal legges vekt på at det legges til rette for kryssplattform.

2.2. Avgrensning

- Løsningen skal ikke ta høyde for hvordan forsikringsselskapene skal motta dataene og hvordan disse må se ut.

3. PROSJEKTORGANISERING

3.1. Ansvarsforhold og roller

Vår oppdragsgiver er Electric Time Car AS med kontaktperson Dag Solhaug. ETC vil bistå med gjennomføringen av prosjektet. Vår veileder på Høgskolen i Gjøvik er Tom Røise, og vil bistå ved teoretiske spørsmål og gi råd i forhold til oppgaven.

Gruppens kontaktperson er Øyvind M. Kongsengen. Han er ansvarlig for kommunikasjon med oppdragsgiver, veileder og evt andre eksterne kilder. Han vil også arrangere møter med oppdragsgiver eller veileder ved behov (utenom de faste møtetidspunktene). Samtidig har Øyvind ansvaret for å føre felleslogg for gruppen. Denne loggen vil inneholde info om hva vi (sammen) har jobbet med i løpet av hver økt. I tillegg vil det føres individuell logg for hver av gruppemedlemmene.

Arve Eidjord har hovedansvar for å sette opp en webside for prosjektet. Begge medlemmene vil redigere innholdet på siden. I tillegg har han ansvaret for å skrive notater fra møter, noe som også vil deles på mellom gruppemedlemmene.
Det vil underveis i prosjektet bli gjort en vurdering på hvordan vi skal fordele arbeidsoppgavene som har med utvikling å gjøre.

3.2. Rutiner og regler i gruppa

Rutiner

- Fast arbeidstid vil være kl 0900 til 1600, 4 dager i uken. I denne perioden skal gruppen sitte sammen og arbeide med oppgaven. Dette vil i hovedsak foregå på skolen eller på et annet egnet sted.

- Møter med veileder/oppdragsgiver vil bli gjort i henhold til avtale. Alle gruppemedlemmer skal delta på møtene, hvis det ikke finnes en gyldig grunn for fravær (sykdom etc.).

- Det vil bli satt av 10 minutter på starten av hver arbeidsdag hvor gruppen gjennomfører en daily scrum møte, og eventuelt diskuterer andre forhold som ses på som kritisk for videre arbeid.

Grupperegler

1. Alle medlemmer skal møte presist til avtalte tidspunkter. Ved sykdom eller annet som gjør det umulig å stille opp skal det gis beskjed så raskt som mulig til resten av gruppen.

2. Prosjektledertittelen vil gå på rundgang mellom gruppemedlemmene. Byttet vil skje ved starten av hver sprint.

3. Ved store gjentatte forsømmelser av avtalte oppgaver kan gruppemedlemmer avskjediges etter samtale og anbefaling av veileder.

4. Avtaler kan kun inngås/underskrives dersom alle gruppemedlemmer samtykker.

5. Ved uenigheter i forhold til oppgaven som påvirker arbeidets fremgang, vil veileder/oppdragsgiver kontaktes. Hvis det etter dette fortsatt er uenighet, vil prosjektleder ha rett til å nedlegge veto.

6. Ved eventuelle kostnader, som for eksempel kjøp av litteratur, vil disse deles likt mellom gruppemedlemmene dersom disse samtykker til at dette er relevant for oppgaven. Kvittering må fremlegges.

7. Det skal arbeides med prosjektet fire dager i uken, der den ukentlige arbeidstiden er 28 timer.. Arbeid utover dette vurderes underveis. Hvis gruppen befinner seg i en situasjon hvor det er fare for at planlagt arbeid ikke vil bli fullført innen tidsfristen, kan prosjektleder pålegge gruppen å jobbe utenom oppsatt arbeidstid.

8. Gruppemedlemmene skal føre individuell logg for alle arbeidstimer, samt en kommentar på hva som er gjort.

4. PLANLEGGING, OPPFØLGING OG RAPPORTERING

4.1. Hovedinndeling av prosjektet

Ved valg av systemutviklingsmetode konkluderte vi raskt med at vi ønsket å benytte en av de smidige metodene til fordel for de tradisjonelle som for eksempel fossefallsmodellen. En av grunnene til dette er at det med stor sannsynlighet vil komme endringer i krav og design underveis. Dette gjør at prosjektet vil ha større sjanse for å lykkes ved å bruke en smidig metode.

Av de smidige metodene som er mest aktuelle har vi eXtreme Programming(XP), FDD, Lean og Scrum. I første omgang var XP av interesse. Men for vår del kan denne metoden bli et problem fordi kunden når som helst kan komme med krav. Vi har en tidsfrist der prosjektet må leveres derfor vil det være vanskelig å kombinere med en slik tankegang. Samtidig gjør XP bruk av parprogrammering. Dette skal blant annet hjelpe utviklerne med å få større oversikt over prosjektet. For oss som bare er to stykker vil ikke dette være noe problem. Samtidig vil det være mer effektiv å arbeide slik der gruppen er større.

Lean gir i hovedsak en liste med best-practices for software development og er derfor ikke et rent rammeverk for hvordan man gjennomfører prosjektet. Siden vi ikke har mye erfaring fra slike prosjekter ser vi dette som svært negativt.

FDD har i grunn gode prinsipper og har elementer av seg som likner på Scrum, slik som for eksempel feature-list/product backlog, plan-by-feature/sprint backlog, build-by-feature/sprint. Men siden vi har en større forståelse for Scrum, valgte vi å ikke bruke FDD i vårt prosjekt.

Scrum er den metoden vi mener vil passe oss best i dette prosjektet. Den gir oss noen retningslinjer for hvordan prosjektet skal organiseres samtidig som den ikke er en metode som er stor og krevende å sette seg inn i. I tillegg er det den metoden vi har fått størst teoretisk innføring i på høgskolen. Siden vi bare er to på gruppa og fordi vi har begrenset med tid kommer vi bare til å benytte oss av deler av Scrum. ETC har også tidligere arbeidet med bachelor-grupper som har benyttet Scrum.

Iterasjoner og møter

Når det gjelder iterasjoner, har vi kommet frem til at vi velger å arbeide med 2-ukers sprint-perioder. Vi mener at 1 uke blir for lite arbeidstid, og at vi ikke vil komme i gang før sprinten er ferdig. Ved å velge sprint varighet på 3 uker, føler vi at vi vil miste kontroll og fokus på når arbeidet skal leveres, samt at størrelsen på sprint backloggen kan bli vanskelig å planlegge/prioritere. I tillegg vil det bli for få sprinter i dette prosjektet som

har fast leveringstidspunkt (slutten av mai). Vi mener derfor at å arbeide med 2-ukers sprint vil passe best for oss, da vil vi få nok tid til å fullføre oppgavene og det vil bli lettere å prioritere elementene i sprint backlog-en.

Utarbeiding av sprint backlog vil vi gjøre i samarbeid med ETC, ved at vi ut i fra product backlog prioriterer hvilke mål/krav som bør gjennomføres først, og velger ut fra de høyest prioriterte oppgavene hva som skal gjøres i løpet av sprinten.

Vi vil gjennomføre Sprint retrospective møter ved slutten av hver sprint. Dette vil gjøre at vi får diskutert eventuelle problemer vi har hatt i løpet av sprinten, og på den måte kan lære av dette for å forbedre det neste sprint.

Ved å ha Sprint review møter etter hver sprint, vil dette være en god mulighet for oss å få feedback fra ETC i forhold til arbeidet vi har gjort i foregående sprint. På den måten får vi kartlagt eventuelle endringer som må gjøres, og ikke minst vil vi få en pekepinn på hvor godt vi har imøtekommet de krav ETC har til programvaren.

Vi vil også benytte oss av burndown chart både for det totale prosjektforløpet og for hver sprint. Dette vil gi oss en oversikt over gjenstående arbeid kontra gjenstående tid i prosjektperioden, samt hjelpe oss med riktig disponering av tiden.

Roller

ETC er produkteier, og vil også representere sluttbrukeren av vårt produkt. ETC vil være inkludert i utviklingsprosessen, og vil i samarbeid med oss utarbeide og prioritere hvilken funksjonalitet programvaren skal inneholde. Vi vil hovedsakelig ha kontakt med Øyvind Flatval som er systemutvikler hos ETC, men Dag Solhaug vil også delta innimellom.

Vi er scrum-teamet, og skal kode og implementere løsningen for ETC. Med tanke på at vi bare er 2 personer, vil vi tilpasse scrum-rammeverket slik det best passer for oss.

Vi ser ikke nytten av å ha en ren ScrumMaster siden vi er et så lite team, men dersom det trengs en ScrumMaster vil prosjektlederen fungere som dette.

4.2. Plan for statusmøter og beslutningspunkter

Vi vil med jevne mellomrom ha møter med veileder for å diskutere prosjektets status og fremgang.

Ettersom vi har valgt Scrum som metode vil de daglige sprintmøtene gi oss en viktig pekepinn i forhold til fremgangen i prosjektet. Her vil vi også fange opp ulike problemer vi står overfor og som kanskje også må diskuteres med veileder eller oppdragsgiver.

Ved avslutning av hver sprint vil vi ha et møte med ETC som deles opp i to, hvor vi i første del tar for oss arbeidet vi har utført (sprint review, med evt demo av arbeid), og i andre del planlegger vi neste sprint (sprint backlog).

I hovedsak er det på møtene med oppdragsgiver at de store beslutningene skal tas. Her skal vi også gå gjennom Product backloggen og videre hva som skal være med i Sprint backloggen

5. ORGANISERING AV KVALITETSSIKRING

5.1. Dokumentasjon, standardbruk og kildekode

Alle dokumenter som skrives skal enten skrives i Google docs eller i Microsoft Word og lagres på doc/docx-format.

All dokumentasjon av kildekode vil bli gjort fortløpende i form av in-line kommentarer. Det vil også vurderes bruk av en “api-generator” avhengig av hvilke programmeringsspråk som velges.

Ved valg av utviklingsverktøy vil vi se på de mest kjente og utbredte for å se hvilke som passe oss best i denne oppgaven. Det vil legges spesiell vekt på at dette verktøyet er framtidsrettet og moderne. Det endelige valget av utviklingsverktøy vil begrunnes i hovedrapporten. Men siden ETC har mindre erfaring med utvikling på Mac vil dette også være mindre aktuelt for oss. Derfor vil valget av utviklingsverktøy i hovedsak stå mellom Eclipse eller Microsoft Visual Studio.

Før møter med ETC skal det sendes ut nødvendig informasjon om samtaleemnene slik at alle kan møte forberedt. Etter møter med ETC og veileder skal det skrives møtereferater.

5.2. Konfigurasjonsstyring

Vi vil benytte oss av Subversion for å sikre at versjonskontroll på kildekoden blir ivaretatt. Denne tjenesten drifter og setter Høgskolen i Gjøvik opp for oss. I tillegg vil vi benytte oss av Google Docs for samskriving av dokumenter og Dropbox for deling av dokumenter underveis.

5.3. Risikoanalyse (identifisere, analysere, tiltak, oppfølging)

Vi har valgt å gjøre en risikoanalyse for å identifisere faktorer som kan påvirke oss som prosjektgruppe i gjennomføringen av Bacheloroppgaven. Vi har vurdert hvor sannsynlig det er at en risiko forekommer, og i hvilken grad dette vil påvirke utviklingen av applikasjonen. Vi har også foreslått tiltak til hvordan vi mener konsekvensen av de høyeste risikoene kan forebygges.

Definisjon av sannsynlighet

Sannsynlighet	**Hvor ofte kan dette forekomme?**
Minimal	Ikke sannsynlig at dette noen gang vil forekomme

Liten	Ikke forventet, men kan forekomme
Moderat	Kan forekomme innimellom
Høy	Vil sannsynligvis forekomme
Meget høy	Vil med sikkerhet forekomme

Definisjon av konsekvens

Konsekvens	**Hvordan påvirker dette prosjektet?**
Ubetydelig	Vil ikke påvirke gjennomføringen prosjektet
Liten	Kan føre til forsinkelser i arbeidet
Moderat	Kan føre til at enkelte deler av prosjektet ikke blir ferdigstilt
Stor	Kan føre til stor negativ konsekvens for prosjektets resultat
Kritisk	Kan føre til at prosjektet blir avsluttet

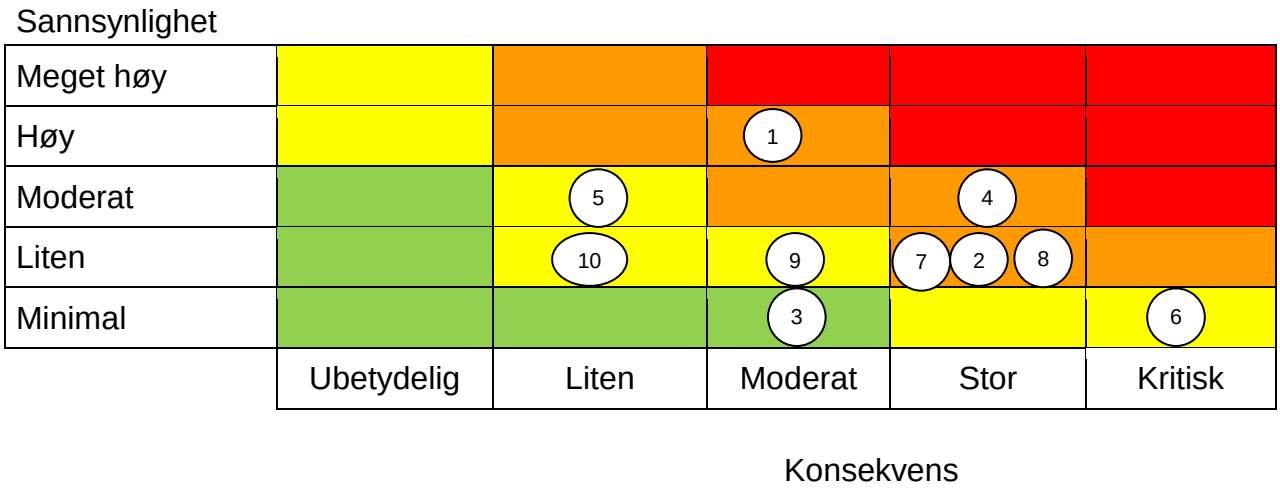
Risikoer relatert til prosjektet

1. Feil fordeling/estimering av tid
2. Langtids sykdom. En av gruppemedlemmene blir syk over lengre tid
3. Uenighet mellom gruppen og veileder/oppdragsgiver som påvirker fremgangen i prosjektet
4. For liten kompetanse i forhold oppgavens krav
5. Veileder/oppdragsgiver er ikke tilgjengelig ved behov
6. Tap av data
7. Feil bruk av tid. Vi bruker for mye tid på et problem, slik at det påvirker fremgangen i prosjektet

8. Store endringer i krav

9. Valgt utviklings verktøy er ikke tilstrekkelig for å løse oppgaven

10. Forsikringsselskap godkjenner ikke applikasjonen som en gyldig metode for innsending av skademelding



Forebyggende tiltak

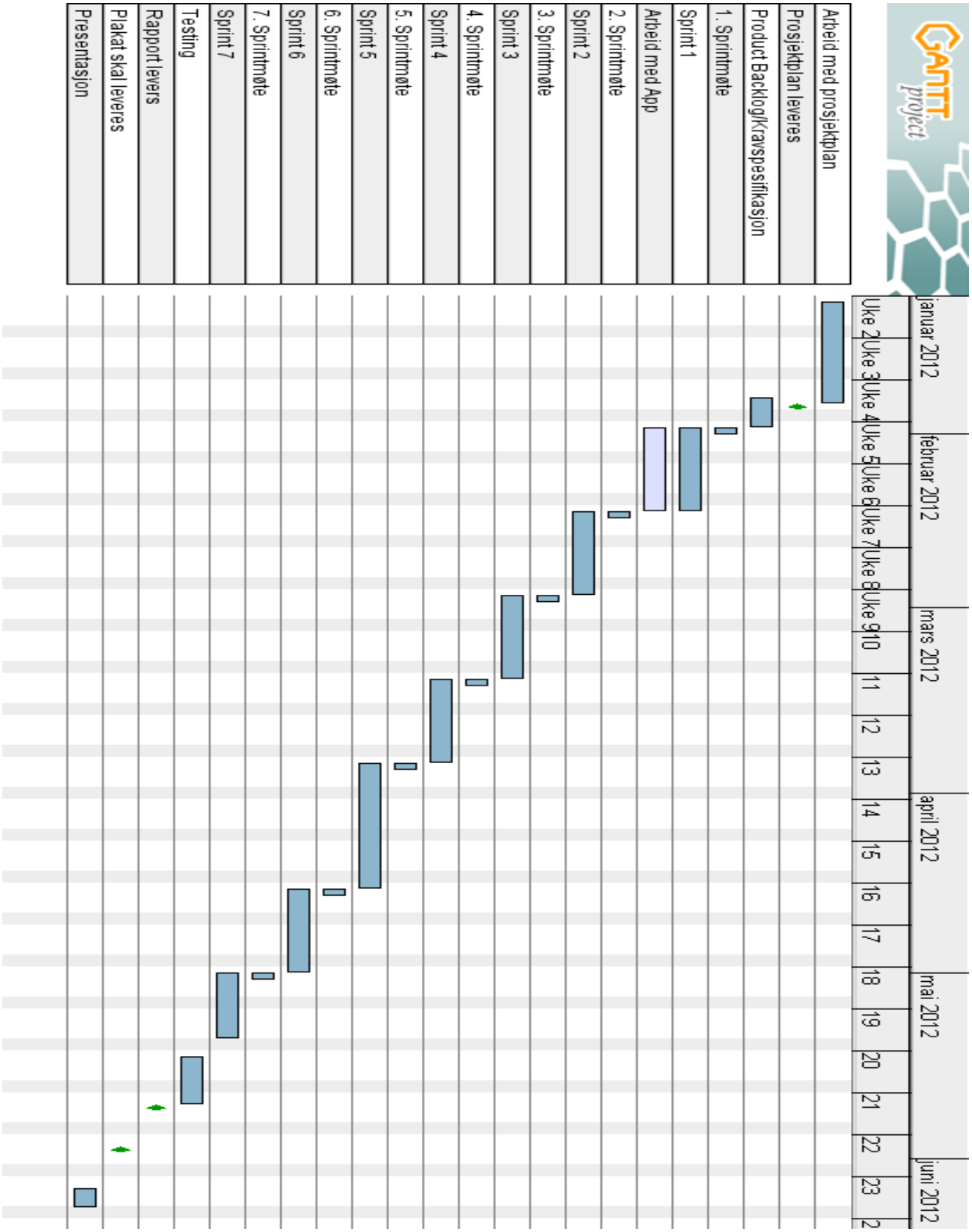
1. Fordeling/estimering av tid
   - Bruk av systemutviklingsmodell
   - Ganttskjema
   - Møter med oppdragsgiver og veileder

4. For liten kompetanse
   - Bistand fra oppdragsgiver og veileder
   - Gjøre seg kjent med relevant dokumentasjon

7. Feil bruk av tid
   - Holde fokus på helheten av oppgaven, og ikke henge seg opp i mindre problemer

8. Store endringer i krav
   - God planlegging i samarbeid med oppdragsgiver

6. PLAN FOR GJENNOMFØRING



7. KILDER

Utviklingsmodeller

- Wikipedia. eXtreme Programming [online] URL: http://en.wikipedia.org/wiki/Extreme_programming (26.01.12)
- Wikipedia. eXtreme Programming practices [online] URL: http://en.wikipedia.org/wiki/Extreme_programming_practices (26.01.12)
- Wikipedia. Feature-driven development [online] URL: http://en.wikipedia.org/wiki/Feature-driven_development (26.01.12)
- Wikipedia. Lean Software development [online] URL: http://en.wikipedia.org/wiki/Lean_software_development (26.01.12)

Risikoanalyse

- ISACA. Risk IT Practitioner Guide [online] URL: http://www.scribd.com/doc/48561562/Risk-IT-Practitioner-Guide (26.01.12)

Tidligere bacheloroppgaver

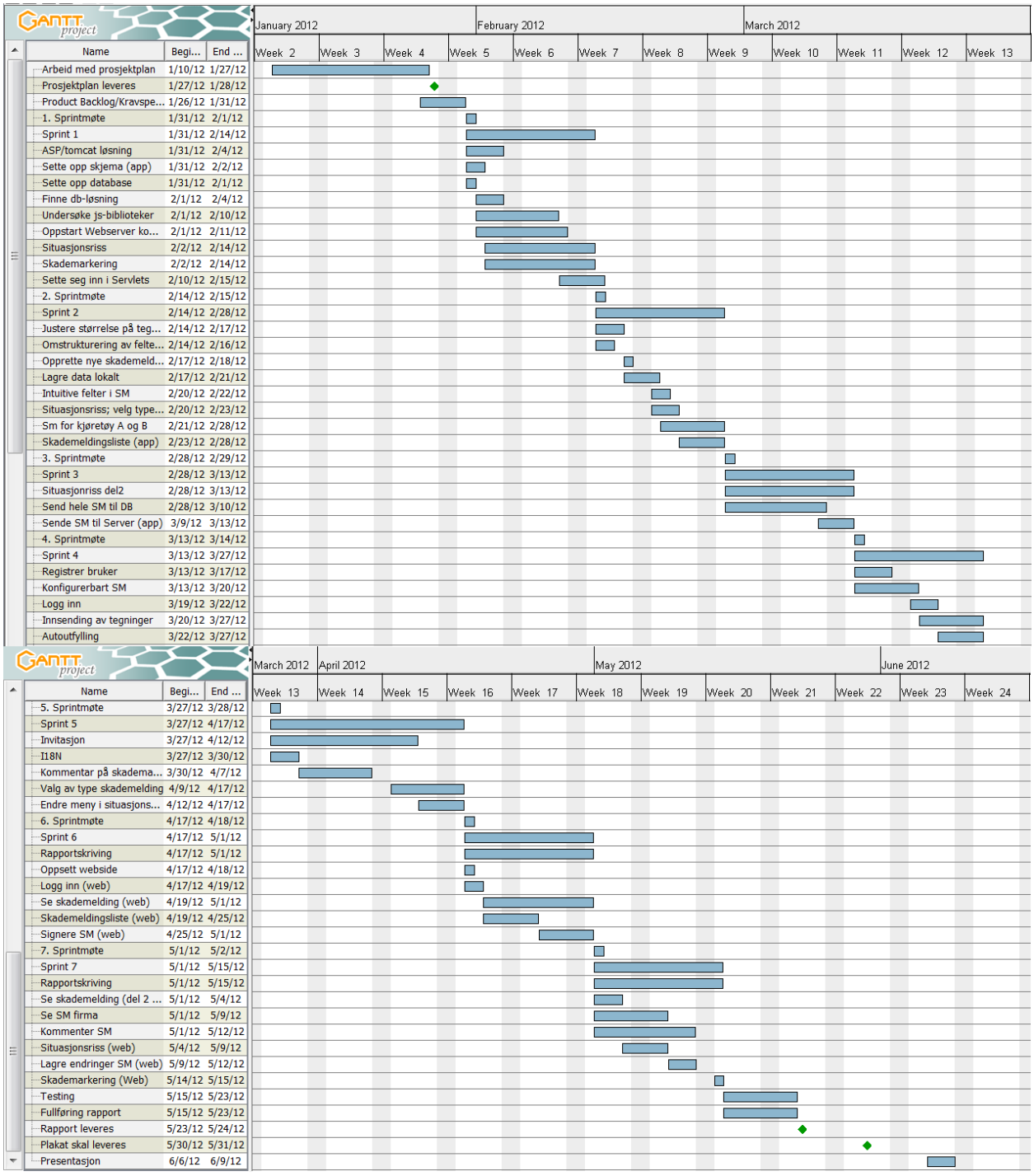
- Ruteplanlegger og flåtestyring i CarAdmin (2010)
- DriSMo - the driving quality application (2011)

Annet

- Informasjon fra ETC ved Dag Solhaug og Øyvind Flatval
- Egne mappearbeider fra emnet Objektorientert systemutvikling

Vedlegg G: Ganttskjema

Det opprinnelige ganttskjemaet finnes i prosjektplanen. Dette ganttskjemaet viser hvor arbeidsforløpet ble til slutt.



Vedlegg H: Sprintkø

Sprint 1

ID	Navn	Type	Prioritet	Estimat (timer)	Notat
4	Egenskade formalia	App	1	7	Lage skjema for 1 person
5	Egenskade tilleggsopplysninger	App	2	7	Lage skjema for supplerende opplysninger
6	Synlige skader	App	3	1	Markere synlige skader på “tegning”
14	Sette opp database	Server	4	7	Sette opp database for skademeldinger
24	Webserver kommunikasjon	Server	5	21	Enkel kommunikasjon mellom server og DB
7	Situasjonsriss	App	6	14	Tegning av skadesituasjon
8	Første berøringspunkt	App	7	14	Markere første berøringspunkt på “tegning”

Sprint 2

ID	Navn	Type	Prioritet	Estimat (timer)	Notat
28	Justere størrelse på tegning	App	4	14	Justere størrelse på tegning i forhold til skjermstørrelsen til Mobiltelefon
29	Intuitive felter i skademelding	App	3	7	DateTime-picker og lignende
31	Omstrukturering av felter i SM	App	1	14	Skademelding i flere skjemer. Tegning først. (slik at ikke alt ligger på samme side)
32	Opprette nye skademeldinger	Server	6	7	Tømme felter i skjema, lagre hvis ikke lagres
33	Lagre data lokalt	Server	5	21	Lagre en skademelding lokalt på telefonen
34	SM for kjøretøy A og B	App	2	14	Skrive inn skademelding for 2 brukere på samme telefon
35	Situasjonsriss, velge type	App	8	14	Stor/liten bil, MC etc..

	kjøretøy				
36	Skademelings liste	App	7	14	Liste over innsendte SM og SM som ikke ennå er sendt

Sprint 3

ID	Navn	Type	Prioritet	Estimat (timer)	Notat
7	Situasjonsriss del 2	App	3	45	Fullføre situasjonsriss-modulen
30	Sende hele SM til DB	Server	1	30	Legge inn all data fra skademelding inn i database
11	Sende SM til server	App	2	5	Sende utfylt SM til server

Sprint 4

ID	Navn	Type	Prioritet	Estimat (timer)	Notat
1	Registrere bruker	App	2	21	Enkel registrering (brukernavn/passord)
2	Logg inn	App	3	14	Logg inn på app
26	Konfigurerbart SM	App	4	24	Skjule felter man ikke vil ha med
29	Innsending av tegninger	Server	1	14	Lagre tegninger(punkter) på server
10	Autoutfylling	App	5	5	Hente lagret info om brukeren fra siste skademeling lagret på telefonen

Sprint 5

ID	Navn	Type	Prioritet	Estimat (timer)	Notat
9	Invitasjon	App	1	40	Invitere motpart til å skrive skademelding (2 telefoner)
27	Klargjøring for internasjonalisering	App	3	10	I18N
43	Kommentar på skademarkering	App	2	14	Kunne legge inn kommentar på skademarkeringer
41	Endre meny i situasjonsriss	App	4	14	Meny kommer opp når man klikker

					på et objekt
38	Valg av type skademelding	App	5	4	Egenskade, invitasjon, 2 brukere på samme tlf

Sprint 6

ID	Navn	Type	Prioritet	Estimat (timer)	Notat
46	Vis skademelding (forside)	Web	3	21	Laste inn skademelding på web
43	Skademeldingsliste	Web	4	21	Vis liste over skademeldinger for innlogget bruker
23	Signering av SM	Web	5	14	Signer Sm
3	Logg inn	Web	2	7	Logge inn på web (rettigheter/type bruker)
22	Sett opp web-side	Web	1	7	Oppsett av web-siden (menyer, css, etc)

Sprint 7

ID	Navn	Type	Prioritet	Estimat (timer)	Notat
44	Situasjonsriss	Web	4	5	Situasjonsriss på web (for endring av tegning)
46	Vis skademelding (supplerende)	Web	2	7	laste inn supplerende oppl.
42	Skadeansvarlig: se skademeldinger for firma	Web	3	7	Se en SM for et firma
48	Skadeansvarlig: kommenter skademelding	Web	5	7	Legge til kommentar på skademelding for firma
16	Lagre endringer SM	Web	1	14	Lagre endringer gjort på web i db
45	Skademarkering	Web	6	5	Skademerking på web

Vedlegg I: Logg

Vi har gjennom hele arbeidsperioden skrevet felleslogg og individuell logg. Fellesloggen viser det arbeidet vi har arbeidet sammen som gruppe, mens den individuelle loggen viser arbeidet som er utført utenfor dette.

Oversikt over tidsbruk i timer

	Felles	Individuell	Totalt
Arve	416,5	90,5	507
Øyvind	416,5	43	459,5

Felleslogg

Dato	Timer	Arbeid
Uke 2	**8**	
11.01.2012	4	Første dag i arbeidet med bacheloroppgaven. Vi diskuterte hvordan vi skulle legge opp løpet og utarbeidet grupperegler.
12.01.2012	4	I dag har vi arbeidet med prosjektplanene.
Uke 3	**21**	
16.01.2012	5	Har fortsatt arbeidet med prosjektplanen. Diskutert rundt valg av utviklingsmetode og hatt møte med veileder for å få klarhet i hvordan prosjektplanen skulle bygges opp.
17.01.2012	5,5	Første møte oppdragsgiver. Vi ble introdusert for oppgaven, hva de så for seg og hva de ønsket seg. Vi fortsatte med å arbeide med utarbeidelsen av en risikoanalyse, ganttskjema og fikk også testet PhoneGap.
18.01.2012	6	Arbeidet videre med prosjektplanen. Det ble skrevet mer utfyllende om risikoanalyse og utviklingsmodell. Har også fortsatt med å teste PhoneGap.
19.01.2012	4,5	Har sett på Titanium (alternativ til PhoneGap) og i tillegg sett hvilke løsninger som finnes angående tegning (oCanvas, Kineticjs)
Uke 4	**22,5**	
23.01.2012	4,5	I dag har vi brukt dagen til å se på tidligere bacheloroppgaver og diskutert hvordan et skademeldingsskjema burde utformes på appen.
24.01.2012	7	Møte med oppdragsgiver så på hva som kunne brukes av det som hadde vært gjort høsten 2011 i prosjektet i Objektorientert Systemutvikling. Møte med veileder. Arbeidet videre med prosjektplanen.
25.01.2012	6	Arbeidet med prosjektplanen. Og opprettet et kode-prosjekt for appen.
26.01.2012	5	Gjort ferdig og levert prosjektplan. Fått satt opp SVN etter noe problemer

		som viste seg å være grunnet feil rettigheter fra IT-tjenesten. Har videre testet jQuery mobile og om dette er et godt alternativ til vår app.
Uke 5	**23**	
30.01.2012	3	Vi har prøvd å få oversikt over hva som skal være med i rapporten. Og har begynt å skrive litt. Opprettet en product backlog.
31.01.2012	7	Første sprintmøte med oppdragsgiver. Satt opp web-side. Begynt på programmering av appen (oppsett av GUI og noe grunnkode)
01.02.2012	7	Vi har funnet ut at jQuery mobile kan føre til at appen blir treg på eldre telefoner. Dette er noe vi skal ta opp med oppdragsgiver. Har også hatt møte med veileder.
02.02.2012	7	I dag har vi brukt mye av dagen til å sette opp og test tomcat og msqldatabase
Uke 6	**21**	
06.02.2012	7	I dag har vi jobbet litt todelt. Arve har jobbet med kommunikasjon mellom app og server, mens Øyvind har jobbet med tegning(Kineticjs) av synlige skader.
07.02.2012	7	I dag hadde vi møtet med oppdragsgiver og tok opp problemet med jQuery. De mente at dette ikke var noe problem siden disse telefonene uansett snart ville fases ut. Jobbet videre med tegnemodul og og legge inn data i databasen.
08.02.2012	7	I dag har vi hatt møte med veileder. Og arbeidet videre i sprinten. Vi hadde noe problemer med Tomcat sånn at vi ikke fikk gjort så mye som vi ønsket.
09.02.2012		Se individuell logg
Uke 7	**28**	
13.02.2012	7	Har arbeidet med appen og noe på server og gjort oss klar til første demomøte i morgen
14.02.2012	7	Demomøte og andre sprintmøte. Arbeidet med å få inn litt smartere felter i skademeldingene og har hatt noen problemer med css som ikke vil lastes
15.02.2012	7	Har måtte omstrukturere noe av koden i dag. Arve har satt seg inn i Kineticjs og arbeidet med skalering av canvasen.
16.02.2012	7	Har implementert sånn at man kan velge mellom kjøretøy A og B og fortsatt arbeidet med skalering av canvasen
Uke 8	**28**	
20.02.2012	7	Fortsatt der vi slapp forrige uke. Innlasting av skademeldinger.
21.02.2012	7	Møte med oppdragsgiver. Jobbet med lagring av skademeldinger.
22.02.2012	7	Jobbet med use case i rapporten, har hatt møte med tom. og arbeidet med

		lagring av data lokalt på appen.
23.02.2012	7	I dag har vi arbeidet med å få opp selve skademeldingslisten, få vist kommentar på skademerkingen og arbeidet med lagring av skademeldinger på telefonen.
Uke 9	**26**	
27.02.2012	6,5	Jobbet med lokal database, kjøretøybehandler og klargjort appen for demoen i morgen.
28.02.2012	7	Møte med opprdagsgiver. Har lest oss opp på Servlets og hatt noe problemer angående oppsett av dette prosjektet.
29.02.2012	6,5	Møte med veileder. Jobbet med database og innsending av data.
01.03.2012	7	Har jobbet med sending av data, parsing til JSON og SQL-generering, og innelgging av objekter på tegnebrettet.
Uke 10	**28**	
05.03.2012	7	I dag har vi brukt dagen til å skrive på rapporten. Vi har blant annet laget noen tekniske memoer og use-case diagram.
06.03.2012	7	Har fortsatt med innlegging av objekter på tegnebrettet og en del servletkoding inkludert databaselogikk
07.03.2012	7	Lynkurs i rapportskriving +++
08.03.2012	7	Blitt ferdig med databasebiten, arbeidet med zoom av tegnebrettet, og fikset sånn at det nå funker å legge til skadede personer.
Uke 11	**28**	
12.03.2012	7	Gjort ferdig det som gjennsto av sprint 3 og gjort oss klart til demoen i morgen
13.03.2012	7	Demo og sprintmøte 4. Starte på registrering av brukere og arbeidet med lagring av tegningene.
14.03.2012	7	Møte med veileder. Vi har arbeidet med sending av tagningene til server og fikset feil i skadede personer. Har også jobbet videre med registrering av brukere.
15.03.2012	7	I dag har vi fått ferdig registrering av brukere og opptimalisering av database connection
Uke 12	**27**	
19.03.2012	7	Arbeidet med konfigurerbarhet av feltene. Brukte en del tid til å få til visning og skjuling av knapper.
20.03.2012	6	Møte med oppdragsgiver. Ble enige om at data skal hentes fra tidligere skademeldinger. Vi skal altså ikke lage egen brukerprofil. På grunn av konfigurerbarheten har vi måtte omstrukturere en del av feltene i

		skademeldingen i dag.
21.03.2012	7	Gjorde ferdig konfigurerbarhet. Møte med veileder. Har videre fikset en feil i innloading av radioknapper i skademeldingene. Har også gjort ferdig sending av mail/aktivering av bruker
22.03.2012	7	Har omstrukturert koden litt slik at skademerking og skadesituasjon arver mer fra Tegnebrettklassen. Har også brukt en del av dagen til testing og feilretting på appen.
Uke 13	**27**	
26.03.2012	7	Har klargjort appen for demoen i morgen og fikset en del småfeil. Har også utforsket JSDoc.
27.03.2012	6	Demomøte og sprintmøte der veileder var til stede. Etter møtet klargjorde vi taskboardet og sprintbaklogen før vi arbeidet vi videre med rapporten (domenemodell, databasemodell og supplementærspesifikasjon)
28.03.2012	7	Måtte gjøre en del endringer i forlhold til kommentering av skademerking. Kommenterte så en del av koden som ennå ikke var kommentert.
29.03.2012	7	Har fikset noe på kommentar til skademarkeing i henhold til hva ETC ønsket.
Uke 15	**7**	
10.04.2012	7	I dag har vi arbeidet med invitasjonsdelen. Serverdelen er nesten ferdig, har startet på app-delen, der UI-et er ferdig men mangler noe funksjonalitet.
11.04.2012		Se individuell logg
12.04.2012		Se individuell logg
Uke 16	**28**	
16.04.2012	7	Testing og fiksing av invitasjon og klartgjort før demo.
17.04.2012	7	Sprint og demomøte
18.04.2012	7	I dag har vi hatt møte med veileder. Etter dette arbeidet vi med webdelen av systemet før vi tok en økt med rapportskriving.
19.04.2012	7	Vi brukte dagen i dag til å skrive på rapporten å arbeide med webdelen.
Uke 17	**-**	
23.04.2012		Se individuell logg
24.04.2012		Se individuell logg
25.04.2012		Se individuell logg
26.04.2012		Se individuell logg
Uke 18	**20**	
30.04.2012	7	Dagen i dag har gått med til å arbeide med ulike deler på web, dette inkluderer visning og signering av skademeldinger. I tillegg har det gått med

		noe tid til å arbeide med rapporten.
02.05.2012	7	Vi brukte det meste av dagen i dag til å utvikle på webdelen, og noe til rapporten.
03.05.2012	6	I dag har vi hatt demomøte og sprintmøte med oppdragsgiver. Etter møtet planla vi sprinten videre sprinten og arbeidet videre med rapporten.
Uke 19	**28**	
07.05.2012	7	Brukte dagen til å skrive på rapporten og arbeide med webdelen.
08.05.2012	7	I dag har vi også brukt dagen til å arbeide med rapporten.
09.05.2012	7	Vi kodet en del på webdelen i dag.
10.05.2012	7	Vi brukte første delen av dagen til å fikse noen ting på appen og noe på web. Så hadde vi møte med veileder. Etter dette arbeidet vi med rapporten. I hovedsak implementasjonsdelen.
Uke 20	**28**	
14.05.2012	7	Arbeidet med raporten og fikset noen feil på webdelen
15.05.2012	7	Rapport, implementasjonsdel, strukturerte om litt på serverdelen.
16.05.2012	7	Hovedfokus på rapportskriving
18.05.2012	7	Brukt dagen i dag til å skrive på rapporten. Implementasjonsdelen og Avslutningsdelen
Uke 21	**18**	
21.05.2012	3	Skriving på rapporten.
22.05.2012	11	Fullføring av rapport
23.05.2012	4	Levering av bacheloroppgaven kl 12.00!

Individuell logg for Øyvind

Dato	Timer	Arbeid
Uke 5	**3**	
30.01.2012	3	Konfigurert test-telefon og testet den med phonegap
Uke 6	**5**	
09.02.2012	5	JQM navigation-bar,eksportering av bilde (tegning), insert html-rader (skadede personer)
Uke 15	**7**	
11.04.2012	5	Jobbet hjemmefra pga sykdom. klargjort for internasjonalisering
12.04.2012	2	Fortsatte på i18n. fortsatt syk så det ble lite arbeid.

Uke 17	**6**	
23.04.2012	4	Hjemme grunnet sykdom, jobbet litt med rapport
24.04.2012	0	Legebesøk, Hjemme grunnet halsbetennelse
25.04.2012	2	Hjemme grunnet halsbetennelse, jobbet litt med rapport
26.04.2012	0	Hjemme grunnet halsbetennelse
Uke 18	**4**	
01.05.2012	4	Jobbet med web-del, uthenting av skademelding fra server
Uke 21	**14**	
18.05.2012	4	Jobbet med implenetasjonsdelen av rapporten
19.05.2012	5	Rapport, brukergrensesnitt, implementasjon
20.05.2012	5	Rapport, avsluttningsdel
Uke 22	**4**	
21.05.2012	4	Lest over og finpusset rapporten, skrevet sammendrag

Individuell logg for Arve

Dato	Timer	Arbeid
Uke 9		
27.02.2012	3,5	Jobbet videre med GUI-lista, lista over skademeldingene
Uke 17	**28**	
23.04.2012	7	Skrev en del på rapporten i dag. Fant noen kilder vi kunne bruke.
24.04.2012	7	Omstrukturerte noe kode på web
25.04.2012	7	Drev med feilretting av webdelen
26.04.2012	7	Hadde møte alene med veileder i dag. Gikk gjennom første delen av rapporten.
Uke 18		
30.04.2012	2	Skrevet videre på rapporten.
Uke 19	**24**	
08.05.2012	3	Brukte noe tid på å fikse noe på lagring av data på serverer..
09.05.2012	2	Jobbet med lagring av data på server.
10.05.2012	4	Fikset noe på konfigurerbarhet og arbeidet litt med rapporten.
11.05.2012	4	Rapportjobbing.
12.05.2012	7	Jobbet med rapporten. Har utvidet noe på kravspesifikasjon

13.05.2012	4	Har i dag brukt litt tid på å skrive om testing av systemet.
Uke 20	**29**	
14.05.2012	3	Skrev på implementasjonsdelen og fikset litt på appen
15.05.2012	2	Så over en del i rapporten. Endret på en del formuleringer.
16.05.2012	4	Det er rapporten som er i fokus for tiden. Så gjennom deler av designdelen.
18.05.2012	4	Jobbet med avslutningsdelen av rapporten.
19.05.2012	6	Gikk gjennom store deler av rapporten, endret formuleringer og rettet feil.
Uke 22		
21.05.2012	4	Eksamen. Finpussing på rapporten
22.05.2012		

Vedlegg J: Møtereferater

Møtereferat 31.01.2012 - ETC

Sted: ETC
Deltakere: Arve Eidjord, Øyvind Kongsengen, Øyvind Flatval
Tidspunkt: 09:00 31.01.2012

Tema for møtet

- Kommentar på prosjektplanen
- Database, server, webserver og kommunikasjon med app
- Sprintplanlegging

Referat
Øyvind Flatval hadde ennå ikke fått tatt en titt på prosjektplanen, men mente at det mest sansynlig ikke var noe serlig å setter fingeren på

Vi står fritt til å velge på hvilken form dataene skal sendes på: XML, json...
På serveren kan vi også velge hvilken teknologi vi vil. For eksempel jsp, asp.net. Helst ikke php som kommunikasjon siden dette blir tungvindt.

Øyvind Flatval mente at vi burde går for en trelagsarkitektur når vi skulle utvikle for å dele opp i presentasjon, logikk, dataaksess.

Sprintplanleggingen gikk raskt for seg. Vi hadde allerede satt opp en product backlog. Bestemte at vi skulle begynne å arbeide med app-en og noe server.

Møtereferat 01.02.2012 - Veileder

Sted: Tom's kontor
Deltakere: Arve Eidjord, Øyvind Kongsengen, Tom Røise
Tidspunkt: 13:00 01.02.2012

Tema for møtet

- Prosjektplan
- Status for arbeid

Referat
Tom savnet noe mer rundt **struktur** av arbeidet og særlig de aktivitetene som er utenom sprintene. Aktivitetene som feks testing av verktøy bør fastlegges, i blant annet gantt-skjema. Dette gjør at vi får det systematisert.

Vi er i gang med første sprint. Tom mente at vi burde legge prestisje i å **gjøre ting helt ferdig**. De delene vi har begynt på må gjøres ferdig, men det går bra å ikke få begynt på noen deler.

Møtereferat 07.02.2012 - ETC

Sted: ETC
Deltakere: Arve Eidjord, Øyvind Kongsengen, Øyvind Flatval
Tidspunkt: 09:00 07.02.2012

Tema for møtet

- jQuery mobile hastighet
- Bruk av biblioteker - lisenser
- Status for fremgang

Referat
Vi ønsket en avklaring rundt bruken av jQuery moblie siden den har noe **ytelsesproblemer** på eldre maskinvare. VI viste derfor en app som kjørte på en eldre og en nyere telefon. Øyvind Flatval mente at problemene ikke var så store og at det dermed ikke var noe problem å ta i bruk biblioteket. Mobiltelefoner blir stadig raskere og de trege vil snart ikke lenger være i bruk. ETC har også vært interessert i jQuery-bruk selv.

jQuery og Kinetic js er biblioteker som har **MIT-lisens**. Vi luftet dette for Øyvind Flatval. ETC hadde ingen invendinger mot å bruke disse, men han skulle sjekke det opp for å være helt sikker

Ø. Flatval ønsket å vite litt om **fremgangen** i prosjektet. Vi mente at vi hadde kommet et godt stykke på vei og omtrent i rute. Ø. Faltval sa han fikk et positivt inntrykk.

Vi snakket også litt om **product backlogen** og kom fram til at vi kanskje burde stykke opp eller spesifisere de ulike elementene mer til neste sprint.

Skademeldingen skal være **konfigurerbar** på en slik måte at felter kan skjules, ikke legge flere til enn det som det utvikles for. Dette gjør at databasen vår kan ha felter likt det i skademeldingen, (kontra å lagre som xml/json i database)

Vi har også valgt tomcat isteden for en asp-løsning.

Møtereferat 08.02.2012 - Veileder

(Ingen møte med Tom 15.02.2012)

Sted: Tom's kontor
Deltakere: Arve Eidjord, Øyvind Kongsengen, Tom Røise
Tidspunkt: 13:00 08.02.2012

Tema for møtet

- Rapporten
- Status for arbeid

Referat
Vi lurte på om kravspesifikasjonsdelen (feks use case-ene) skulle lages ferdige for så å ikke endres. Slik at dersom det ble forandringer så måtte det dokumenteres senere en annen plass i rapporten. Tom mente at siden vi bruker en iterativ prosess(Scrcum) så vare det ikke noe problem å gjøre endringer underveis. Vi kunne godt vise prosessen og hvilke endringer som ble gjort, ikke alle og alt, men noen. Siste utgave av feks use case i rapporten. Vise hvilke endringer som er gjort. Men Scrum måtte ikke bli en unskyldning for å ikke planlegge.

Vi bekreftet overfor Tom at vi hadde gått for PhoneGap, jQuery mobile og at vi hadde hatt en samtale rundt jQery mobile.

Tom mente også at vi burde legge inn alle ting vi kommer på i Product Backlogen, som for eksempel: ta bilder. Selv om de ikke skal gjøres før om lenge.

Møtereferat 14.02.2012 - ETC

Sted: ETC
Deltakere: Arve Eidjord, Øyvind Kongsengen, Dag Solhaug
vind FlatvalTidspunkt: 09:00 14.02.2012

Tema for møtet

- Demo
- Sprintplanlegging
- Datakommunikasjon

Referat
Vi presenterte det vi hadde gjort med appen. Dag mente vi hadde komme godt i gang, men hadde noe tips rundt strukturering av skjermene. Første berøringspunkt (SM pkt 10) burde komme som det første man valgte. Listen over felter burde ikke være så lang. Den burde deles inn i flere skjermer som man kunne scrolle høyre/venstre. Tegningen over berøringspunkt burde også kunne inneholde tekst mente Dag. Bilde av biler kunne vi få ved å sende mail til Øyvind Flatval.

Vi ble enige om å ha vekten på app-en også denne sprinten. Dag mente det var lurt å få på plass lokal lagring av data.

Om datakommunikasjon burde vi sende så lite data som mulig, kun de feltene der det er skrevet inn noe. Json er bra. Hvis man skulle sende ennå mindre kunne man sendt en kommaseparert string.

Møtereferat 21.02.2012 - ETC

Sted: ETC
Deltakere: Arve Eidjord, Øyvind Kongsengen, Øyvind Flatval
Tidspunkt: 09:00 21.02.2012

Tema for møtet

- Status
- Bilbilder
- Header/Footer

Referat
Ettersom Øyvind Flatval (ØF) ikke kunne være med på forrige sprintmøtet gav vi ham en demo av noe av det vi hadde gjort. Ettersom vi er midt i en sprint gjørt det at vi ikke kunne vise alt.

Vi snakket litt om bilder som skulle brukes på avmerking av skader. Vi skulle bruke de ØF hadde sendt oss på mail (som PDF) og redigere de slik at dørene ble lagt inntil bilen.

Header og Footer(som inneholder valg av bil A eller B) forsvinner mens du scroller i jQuery. Vi kunne godt gjøre det sånn at footeren kommer fra med trykk på menyknappen eller tap på skjermen. Den trenger altså ikke alltid å vises.

Møtereferat 22.02.2012 - Veileder

(Vi hadde ingen møte med Tom 15.02.2012)

Sted: Tom's kontor
Deltakere: Arve Eidjord, Øyvind Kongsengen, Tom Røise
Tidspunkt: 13:00 22.02.2012

Tema for møtet

- Status for arbeid
- Use case-diagram
- Annet

Referat
Vi forklarte litt rundt status for arbeidet.Tom mente at vi burde være lojale mot sprint backlogen og levere det vi bestemte oss for. Ikke bare hver av itemene 70 % ferdig. Vi bør ikke ta på oss mer en det vi klarte å levere forrige runde.

Vi lurte på om det burde være to use case diagram, ett for web og ett for applikasjonen. Tom mente at det burde være ett der alle use casene var.

Tom mente at læringsmålene er de viktigste å oppnå, altså den prosessen og det vi lærer. Derfor er refleksjonen viktig i rapporten. Vi bør etter hvert bruke mer tid på rapporten enn utviklingen.

Møtereferat 28.02.2012 - ETC

Sted: ETC
Deltakere: Arve Eidjord, Øyvind Kongsengen, Øyvind Flatval (ØF)
Tidspunkt: 09:00 28.02.2012

Tema for møtet

- Demo
- Sprintmøte

Referat
Vi startet møtet med å vise det vi hadde arbeidet med i sprint 2. ØF synes det så bra ut og hadde bare små kommentarer. For eksempel kunne det hende at vi i framtiden skulle ha en annen hovedmeny (med to knapper: ny skademelding og en med historikk), og at tittelene i listen over SM kunne være bilskilt og skadested (pluss dato)

Vi ble i sprintmøtet enige om å arbeide med to store items: Sende fra appen til server (og inn i database) og lage situasjonsrissdelen.

Møtereferat 29.02.2012 - Veileder

Sted: Tom's kontor

Deltakere: Arve Eidjord, Øyvind Kongsengen, Tom Røise
Tidspunkt: 13:00 29.02.2012

Tema for møtet
- Status for arbeid
- Rapport

Referat
Vi snakket litt om hva vi hadde gjort. Vi er si sprint 3. Sprintmøtet gikk bra. To store items (situasjonsriss og sending av data)

Tom lurte på om vi kunne sende han det vi hadde gjort på rapporten innen 15. mars. Noe vi sa vi skulle.

Møtereferat 13.03.2012 - ETC, demo/sprintmøte

Sted: ETC
Deltakere: Arve Eidjord, Øyvind Kongsengen, Øyvind Flatval (ØF)
Tidspunkt: 09:00 28.02.2012

Tema for møtet
- Demo
- Sprintmøte

Referat
Vi startet møtet med å vise det vi hadde arbeidet med i sprint 3. ØF fikk prøve de funksjonene vi hadde implementert og var fornøyd med at vi hadde fokusert på hovedfunksjonaliteten og ikke brukt masse tid på ting utenom oppgaven.

Sprintmøtet gikk greit for seg. Etter litt diskusjon med ØF ble konfigurasjonsdelen av SM og registrering/innlogging prioritert opp. Det er da i hovedsak dette vi skal jobbe med, samt implementering av å kunne legge til flere vitner.

Møtereferat 14.03.2012 - Veileder

Sted: Tom's kontor
Deltakere: Arve Eidjord, Øyvind Kongsengen, Tom Røise
Tidspunkt: 13:00 14.03.2012

Tema for møtet
- Status for arbeid
- Rapport

Referat
Vi ble enige om å levere et tidlig utkast av rapporten før kl 9 fredag 16.03.2012.
Vi ble også enige om at Tom skulle være med på neste sprintmøte.

Tom mente også at vi ikke måtte gå for å gjøre alt, men kanskje bare integrere en del med for eksempel ETC sine systemer.

Møtereferat 20.03.2012 - ETC

Sted: ETC
Deltakere: Arve Eidjord, Øyvind Kongsengen, Øyvind Flatval (ØF)
Tidspunkt: 09:00 20.03.2012

Tema for møtet
- Status
- Innlogging/registrering
- Konfigurerbare felter

Referat
Vi fortalte at vi var godt i gang med sprinten og at vi hadde mer en nok å gjøre fram til neste uke.

Angåense innlogging og registrering luftet ØF noen tanker om hvilke måter dette skjer på hos andre. For eksempel hos Amazon: Som ikke har "trykk her for å registrer deg". Du kan bare skrive inn et brukernavn og passord i innloggingsfeltene. Hvis brukeren ikke finnes så kommer du til registreringsformen.

E-posten til brukerne må bekreftes ved at det sendes ut en e-post med en link (som har en unik hash) som de må trykke på for å bekrefte. ØF sa vi kunne velge om bruker måtte lage passord når han hadde trykket på denne linken, eller når han registrerer seg på app-en

ØF kom også på at han hadde glemt å si at **kommentarer** på "markering av skade på bil" må være med!

Angående konfigurerbare felter synes ØF det hørtes fornuftig ut at vi hadde en liste over de felter som skulle skjules.

Når en ny SM opprettes er det fornuftig at data fra den forrige SM-en som ble opprettet blir lastet inn (i den nye SM) mente ØF. Dette sa vi oss enige i.

Vi fortalte også at tom ønsket å være med på neste møtet(sprintmøte), noe ØF ikke hadde noe problemer med. Også Dag kommer mest sannsynlig til å være med på dette møtet.

Møtereferat 21.03.2012 - Veileder

Sted: Tom's kontor
Deltakere: Arve Eidjord, Øyvind Kongsengen, Tom Røise
Tidspunkt: 13:00 21.03.2012

Tema for møtet
- Status for arbeid
- Rapport

Referat
Tom mente at det vi hadde gjort i rapporten til nå var ganske tynt. Han oppfordret oss til å jobbe mer aktivt med denne.

Vi burde også dele opp use caset "Skriv skademelding" i flere deler siden dette er så mye større enn de andre use case-ene. Enten gjøre det i use case-diagrammet eller vise det i et detalsjert use case. Og i use case-ene er det viktig at vi ikke skriver hvordan løsningen skal være, men hva "kravene" er.

Tom poengterte at det som var påkrevd av oppdragsgiver skulle plasseres i kravspesifikasjonen. Noe som vil si at for eksempel de teknologiene vi har brukt og valg av de kommer i designdelen.

Han savnet også en domenemodell.

Møtereferat 27.03.2012 - ETC

Sted: ETC
Deltakere: Arve Eidjord, Øyvind Kongsengen, Øyvind Flatval (ØF), Dag Solstad?, Tom Røise
Tidspunkt: 09:00 27.03.2012

Tema for møtet

- Test av app
- Demo av app
- Sprintmøte

Referat
Vi startet møtet med at Dag kunne få prøvekjørt app-en. Han kommenterte en ting som vi senere på møte valgte å ta med i den forestående sprinten.

Vi viste det vi hadde jobbet med. Registrering, logg inn, konfigurerbare felter.

På sprintmøtete planla vi hva vi skulle gjøre neste sprint. Vi ble blant annet enige om å arbeide med inviasjonsdelen. Her ble det gjort noen endringer i forhold til det use case-et vi hadde satt opp. Dag mente at det burde likne mer på slik som invitasjon foregår på Wordfeud. For vår del vil dette si gjennom å oppgi en e-post-adresse. Og det ble det fastsatt at man skulle gå ut ifra at brukere måtte være koblet til Internett for å kunne invitere. Også motparten måtte være koblet til Internett.

Det ble også avklart at "Registrer firmabrukere" ikke skulle være en del av use case-ene ettersom disse brukerne kun kan opprettes av ETC, i det systemet som allerede finnes.

Møtereferat 17.04.2012 - ETC (Sprintmøte)

Sted: ETC
Deltakere: Arve Eidjord, Øyvind Kongsengen, Øyvind Flatval (ØF), Dag Solstad?, Tom Røise
Tidspunkt: 09:00 17.04.2012

Tema for møtet

- Demo av app
- Sprintmøte

Referat
ØF mente at vi ikke rekker integrasjonen med CarAdmin.
Ble enige om å arbeide med Webdelen denne sprinten.

Møtereferat 18.04.2012 - Veileder

Sted: Tom's kontor
Deltakere: Arve Eidjord, Øyvind Kongsengen, Tom Røise

Tidspunkt: 13:00 18.04.2012

Tema for møtet
- Status for arbeid
- Rapport

Referat
Vi lurte på om vi kunne få en kommentar på **rapporten**. Vi ble enige om å sende den til tom før mandags morgen sånn at vi kunne få en kommentar til onsdag. Samtidig med rapporten skulle vi også sende med en **statusrapport**.

Tom forklarte at vi burde passe på at rapporten var vitenskaplig nok. Dette er noe HiG ønsker i større grad. Dette kan være ved å gjøre vurderinger og diskusjoner og referere til ulike kilder.

På møtet i går (17.04.2012) med oppdragsgiver kom det fram at vi ikke kom til å rekke integrasjonen med CarAdmin. Vi lurte på hvordan dette kunne løses i rapporten. Tom mente at vi bare måtte forklare de valgene vi har gjort underveis og at det i hovedsak er oppdragsgiver som vet hva som vil være det beste å gjøre i dette tilfellet. Men vi kunne lage et API eller lignende slik at det ble lagt til rette for integrering senere.

Det er også viktig i rapporten å ikke unnskylde seg for det vi ikke fikk gjort mente Tom. Viktigere å få vist faktisk hva vi har fått gjort også legge en plan for hva som kan gjøres videre.

Det kan også være viktig å få fram at tegnebiten var mer kompleks en det som HiG mente at den var. Siden de mente at det bare var å bruke noen ferdigbiblioteker. Dette var noe oppdragsgiver ønsket at ble brukt en del tid på. Få bort tanken om at det bare er en direkte overføring fra penn til digital skademelding.

Møtereferat 26.04.2012 - Veileder

Sted: Tom's kontor
Deltakere: Arve Eidjord, Tom Røise
Tidspunkt: 11:00 26.04.2012

Tema for møtet
- Rapport

Referat
Vi gikk gjennom det vi hadde skrevet i rapporten. Tom mente at vi burde skrive mer utfyllende. For eksempel i 1.1.1 problemområde. Vi burde bruke figurere for å sprite opp rapporten. Ikke overdreven bruk, men fornuftig bruk.

Burde få bort betegnelser som "for enkelhtesskyld". Det er ikke for enkelhetsskyld. Det har en annen grunn.

I de detalsjerte use casene burde vi fylle inn mer om hva som skjer når man tegner skadesituasjonen feks.

Lagdelingsmodellen (3.2.1) må man gjøre mer ut av.

Domenemodellen må også gjøres mer ut av
I databasemodellen bør det komme fram ulike nøkler, id-er viktige felter. (trenger ikke å ta med alle feltene siden det er så mange)

I memoene må vi støtte oss på kilder. Memoet om server-web teknologi kunne vært litt mer utfyllende.

I supplementærspesifikasjon kan vi noen plasser skrive om det var ETC som bestemte at det skulle være sånn eller om det er noe vi har kommet med. Kanskje vi kan fjerne Oppetid eller ihvertfall påpeke at dette er noe som ligger utenfor oppgaven. Er jo ETC som drifter serveren. Vi må heller ikke gjenta oss. Så Skademelding og inviasjon kan fjernes fra supplementær spesifikasjon siden de også er use case

Øyvind var syk.

Vedlegg K: Prosjektavtale

Under følger prosjektavtalen inngått mellom oss, HiG og ETC.

HØGSKOLEN I GJØVIK

PROSJEKTAVTALE

mellom Høgskolen i Gjøvik (HiG) (utdanningsinstitusjon), Electric Time Car AS (ETC) (oppdragsgiver), og Øyvind M. Kongsengen og Arve Eidjord (student(er))

Avtalen angir avtalepartenes plikter vedrørende gjennomføring av prosjektet og rettigheter til anvendelse av de resultater som prosjektet frembringer:

1. Studenten(e) skal gjennomføre prosjektet i perioden fra 10.01.2012 til 23.05.2012.

   Studentene skal i denne perioden følge en oppsatt fremdriftsplan der HiG yter veiledning. Oppdragsgiver yter avtalt prosjektbistand til fastsatte tider. Oppdragsgiver stiller til rådighet kunnskap og materiale som er nødvendig for å få gjennomført prosjektet. Det forutsettes at de gitte problemstillinger det arbeides med er aktuelle og på et nivå tilpasset studentenes faglige kunnskaper. Oppdragsgiver plikter på forespørsel fra HiG å gi en vurdering av prosjektet vederlagsfritt.

2. Kostnadene ved gjennomføringen av prosjektet dekkes på følgende måte:
   - Oppdragsgiver dekker selv gjennomføring av prosjektet når det gjelder f.eks. materiell, telefon/fax, reiser og nødvendig overnatting på steder langt fra HiG. Studentene dekker utgifter for trykking og ferdigstillelse av den skriftlige besvarelsen vedrørende prosjektet.
   - Eiendomsretten til eventuell prototyp tilfaller den som har betalt komponenter og materiell mv. som er brukt til prototypen. Dersom det er nødvendig med større og/eller spesielle investeringer for å få gjennomført prosjektet, må det gjøres en egen avtale mellom partene om eventuell kostnadsfordeling og eiendomsrett.

3. HiG står ikke som garantist for at det oppdragsgiver har bestilt fungerer etter hensikten, ei heller at prosjektet blir fullført. Prosjektet må anses som en eksamensrelatert oppgave som blir bedømt av faglærer/veileder og sensor. Likevel er det en forpliktelse for utøverne av prosjektet å fullføre dette til avtalte spesifikasjoner, funksjonsnivå og tider.

4. Den totale besvarelsen med tegninger, modeller og apparatur så vel som programlisting, kildekode, disketter, taper mv. som inngår som del av eller vedlegg til besvarelsen, gis det en kopi av til HiG, som vederlagsfritt kan benyttes til undervisnings- og forskningsformål. Besvarelsen, eller vedlegg til den, må ikke nyttes av HiG til andre formål, og ikke overlates til utenforstående uten etter avtale med de øvrige parter i denne avtalen. Dette gjelder også firmaer hvor ansatte ved HiG og/eller studenter har interesser.

   Besvarelser med karakter C eller bedre registreres og plasseres i skolens bibliotek. Det legges også ut en elektronisk prosjektbesvarelse uten vedlegg på bibliotekets del av skolens internett-sider. Dette avhenger av at studentene skriver under på en egen avtale hvor de gir biblioteket tillatelse til at deres hovedprosjekt blir gjort tilgjengelig i papir og nettutgave (jfr. Lov om opphavsrett). Oppdragsgiver og veileder godtar slik offentliggjøring når de signerer denne prosjektavtalen, og må evt. gi skriftlig melding til studenter og dekan om de i løpet av prosjektet endrer syn på slik offentliggjøring.

5. Besvarelsens spesifikasjoner og resultat kan anvendes i oppdragsgivers egen virksomhet. Gjør studenten(e) i sin besvarelse, eller under arbeidet med den, en patentbar oppfinnelse, gjelder i forholdet mellom oppdragsgiver og student(er) bestemmelsene i Lov om retten til oppfinnelser av 17. april 1970, §§ 4-10.

6. Ut over den offentliggjøring som er nevnt i punkt 4 har studenten(e) ikke rett til å publisere sin besvarelse, det være seg helt eller delvis eller som del i annet arbeide, uten samtykke fra oppdragsgiver. Tilsvarende samtykke må foreligge i forholdet mellom student(er) og faglærer/veileder for det materialet som faglærer/veileder stiller til disposisjon.

7. Studenten(e) leverer oppgavebesvarelsen med vedlegg (pdf) i Fronter. I tillegg leveres et eksemplar til oppdragsgiver.

8. Denne avtalen utferdiges med et eksemplar til hver av partene. På vegne av HiG er det dekan/prodekan som godkjenner avtalen.

9. I det enkelte tilfelle kan det inngås egen avtale mellom oppdragsgiver, student(er) og HiG som nærmere regulerer forhold vedrørende bl.a. eiendomsrett, videre bruk, konfidensialitet, kostnadsdekning og økonomisk utnyttelse av resultatene.

   Dersom oppdragsgiver og student(er) ønsker en videre eller ny avtale, skjer dette uten HiG som partner.

10. Når HiG også opptrer som oppdragsgiver trer HiG inn i kontrakten både som utdanningsinstitusjon og som oppdragsgiver.

11. Eventuell uenighet vedrørende forståelse av denne avtale løses ved forhandlinger avtalepartene i mellom. Dersom det ikke oppnås enighet, er partene enige om at tvisten løses av voldgift, etter bestemmelsene i tvistemålsloven av 13.8.1915 nr. 6, kapittel 32.

12. Deltakende personer ved prosjektgjennomføringen:

HiGs veileder (navn): Tom Røise

Oppdragsgivers kontaktperson (navn): Dag L. Solhaug

Student(er) (signatur): _____ dato 24.01.2012

_____ dato 24.01.2012

_____ dato _____

_____ dato _____

Oppdragsgiver (signatur): _____ dato 24.01.2012

IMT Dekan/prodekan (signatur): _____ dato 15/2-2012

Versjon Januar 2011hb

Vedlegg L: Oppsett av utviklingsmiljø

Dette vedlegget inneholder en veiledning for å komme i gang med videreutvikling og vedlikehold av systemet.

App

1. Installer Eclipse EE
2. Last ned og installer Android SDK fra http://developer.android.com/sdk/index.html. For hjelp til installasjonen se: http://developer.android.com/sdk/installing.html
3. Last ned og installer PhongeGap fra http://phonegap.com. Følg veiledningen på http://docs.phonegap.com/en/1.7.0/guide_getting-started_android_index.md.html#Getting%20Started%20with%20Android
4. Legg til kildekoden til et nytt Android/PhoneGap-prosjekt.
5. Kjør appen på telefonen/AVDen eller prøv i en webbrowser

Server og web

1. Last ned og installer Tomcat 7.0 fra http://tomcat.apache.org/download-70.cgi. Se også for installasjonshjelp http://tomcat.apache.org/tomcat-7.0-doc/setup.html
2. Sett opp Tomcat-server i Eclipse.
3. Opprett et nytt dynamisk web prosjekt i Eclipse
    a. Legg til kildekoden
    b. I mappen web\linkedfiles\tegning skal det linkes fem filer fra appen:
        i. kinetic.js
        ii. situasjonsrissobjekter.js
        iii. situasjonsriss.js
        iv. skademering.js
        v. tegnebrett.js
4. Start server
5. Åpne index.jsp i en webbrowser og å sjekke at alt fungerer