Bacheloroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Institutt for datateknologi og informatikk

Jonas Melby
Andreas Danielsen

Miljøturguide

Bacheloroppgave i Dataingeniør
Veileder: Frode Haug
Mai 2020

NTNU
Kunnskap for en bedre verden

Jonas Melby
Andreas Danielsen

Miljøturguide

Bacheloroppgave i Dataingeniør
Veileder: Frode Haug
Mai 2020

Norges teknisk-naturvitenskapelige universitet
Institutt for datateknologi og informatikk

NTNU
Kunnskap for en bedre verden

Sammendrag

Tittel:	Miljøturguide
Dato:	20. Mai 2020
Deltakere:	Jonas Melby Andreas Danielsen
Veileder:	Frode Haug
Oppdragsgiver:	Electric Time Car AS
Kontaktperson:	Dag L Solhaug
Nøkkelord:	Programmingering, Fullstack
Antall sider:	102
Antall vedlegg:	6
Publisering:	Åpen.

Sammendrag:	Vi har utviklet en webapplikasjon som lar brukere søke etter reiser med ulike transportmetoder og få opp miljøutslipp for de ulike reisene. Brukeren vil ha muligheten til å se tidligere reiser og overvåke sitt miljøavtrykk.

Abstract

Title:	Miljøturguide
Date:	20. May 2020
Participants:	Jonas Melby Andreas Danielsen
Supervisor:	Frode Haug
Employer:	Electric Time Car AS
Contact:	Dag L Solhaug
Keywords:	Programming, Fullstack
Number of pages:	102
Number of appendixes:	6
Availability:	Open

Abstract:	We have developed a webapplication where users can search for trips with different transportation modes. They will also receive information regarding emissions for the different trips and transportation modes. The user should be able to see past trips and monitor their environmental footprint.

Forord

Vi vil takke oppdragsgiver Electric Time Car AS og spesielt deres daglige leder Dag L Solhaug som har bistått med veiledning gjennom ukentlige møter samt når vi har hatt behov for det.

Vi vil også takke Frode Haug, vår veileder under prosjektet, for å veilede oss gjennom hele prosjektperioden. Han har vært tilgjengelig ved ukentlige møter, men også ved behov, selv med korte varsler.

Innhold

Figurer

Kodelister

Forkortelser og definisjoner

SSL Secure Sockets Layer

ETC Electric Time Car, oppdragsgiver.

IDE Integrated Development Environment

Spring Security En modul innen Spring-rammeverket som tilbyr både autentisering og autorisering til Java-applikasjoner[1].

HTTP Kommunkasjonsprotokoll som brukes til å overføre HTML-dokumenter mellom tjenere og klienter ved hjelp av en transportprotokoll[2].

HTTPS En kommunikasjonsprotokoll som støtter sikker kommunikasjon over internett. Protokollen sikrer at man kommuniserer med riktig nett-tjeneste og at informasjonen som overføres ikke kan avlyttes eller endres[3].

HTML Hyper Text Markup Language - Et formateringsspråk benyttet for å lage hypertekst-dokumenter på web.[4].

CSS Cascading Style Sheets - Format for styling av html-dokumenter på Internett (World Wide Web), standardisert av W3C. [5].

Spring data JPA En modul innen Spring-rammeverket. Modulen forenkler kommunikasjonen med databasen. [6].

Spring Boot Verktøy for strukturering og oppbygging av applikasjonen. [7].

Thymeleaf En moderne Java mal-engine på serversiden. [8].

Spring MVC En modul innen Spring-rammeverket som er bygget rundt MVC-arkitekturen. [9].

Kapittel 1

Introduksjon

1.1 Omfang

1.1.1 Problemområde

Global oppvarming er et aktuelt tema for tiden. En ser daglig konsekvenser av de jevnt økende temperaturene i atmosfæren. Isbreer smelter, havnivået øker og regnskoger er i ferd med å dø. Dette mener noen er et resultat av utslipp av klimagasser på grunn av menneskelig aktivitet.[1]

1.1.2 Avgrensning

En stor del av de menneskeskapte klimautslippene stammer fra transport, og under denne kategorien kan mange bidra til å redusere utslippene.[2]

Det finnes ulike applikasjoner for ruteplanlegging og reisereservasjoner, men det finnes ingen applikasjoner som i tillegg beregner ditt miljøavtrykk basert på transportmetodene du velger.

Oppgaven fra ETC var orginalt å lage en modul som skulle implementeres i deres allerede eksisterende applikasjon CarAdmin. Denne oppgaven var i hovedsak ment for en gruppe på tre til fire studenter. Ettersom vi er to ble vi anbefalt å avgrense oppgaven av både programansvarlig på skolen og oppdragsgiver. Oppdragsgiver mente det mest krevende med den orginale oppgaven var å sette seg inn i deres system. Avgrensningen blir derfor å utvikle wekapplikasjonen som et enkeltstående system, uten tilknyttning til den eksisterende applikasjonen. Med disse avgrensningene mener oppdragsgiver at vi har gode fotutsetninger for å ha tid til å implementere all funksjonalitet som er beskrevet i oppgaven.

[1] https://www.nationalgeographic.com/environment/global-warming/global-warming-overview/

[2] https://miljostatus.miljodirektoratet.no/tema/klima/norske-utslipp-av-klimagasser/klimagassutslipp-fra-transport/

1.1.3 Oppgavebeskrivelse

Oppgaven vi har fått er å lage en webapplikasjon kalt Miljøturguide. Denne applikasjonen skal gjøre det enkelt for miljøbevisste brukere å se og velge de mest miljøvennlige transportalternativene for en reise.

Webapplikasjonen skal inneholde en reiseplanlegger hvor brukere kan planlegge sine reiser. Her skal brukerne ha mulighet til å søke etter forskjellige reiser ut ifra;

- Startssted
- Stoppested
- Dato og klokkeslett ved reisestart og ankomst.
- Ønsket transportmetode

Ut ifra søkekriteriene skal brukeren få vist de reisealternativene som passer søkene best. For hvert reisealternativ vil det vises hvor lang tid reisen vil ta og hvor lang reisen er i kilometer. Brukeren skal kunne velge mellom de forskjellige alternativene og få se detaljert hvert steg av reisen, samt se visuelt på et kart hvor reisen går. Miljøeffekten av brukerens valg skal også vises, slik at brukeren selv ser hva som spares av miljøutslipp sammenlignet med de andre reisealternativene.

Velgbare transportmidler vil være;

- Bil
- Buss
- Tog
- Gange

Samkjøring skal også kunne organiseres.

Etter at brukeren har valgt sin ønskede reise skal de videresendes til en bekreftelsesside. Her skal brukeren få oppsummert hele reisen. Hvis brukeren bekrefter reisen skal alle reisereservasjoner gjøre automatisk i bakgrunnen, og en faktura skal bli sendt til økonomiansvarlige i organisasjonen.

Basert på valgt transportalternativ dokumenteres spart miljøavtrykk for brukere, avdelinger og organisasjon. Disse dataene skal kunne vises i en totaloversikt for hele organisasjonen. Det er utslipp av karbondioksidgass (Co2) og nitrogenoksidgasser (NOx) som skal dokumenteres.[3]

1.2 Formål

1.2.1 Resultatmål

ETC vil ved prosjektets slutt få levert en webapplikasjon som:

- Lar brukere logge inn for å reservere reiser.

[3] `https://www.nho.no/samarbeid/nox-fondet/artikler/hva-er-nox/` NOx er en fellesbetegnelse for nitrogenoksidene NO og NO2.

- Gjør det mulig for brukeren å se og velge mellom ulike transportmåter, samt tidsbruk for de ulike alternativene.
- Planlegge og avtale samkjøring.
- Lar brukeren overvåke sitt miljøavtrykk for de reserverte reisene.
- Gir brukeren en oversikt over sitt totale miljøutslipp.

1.2.2 Effektmål

Denne oppgaven vil tjene ETC på ulike måter:

- De mottar gratis arbeidskraft for å utføre en oppgave etter deres ønsker.
- De tjener på det tidsmessig og ressursmessig ved å ikke måtte utvikle produktet selv.
- De vil få et produkt som eventuelt kan videreutvikles og implementeres i deres allerede eksisterende løsning Car Admin.

1.2.3 læringsmål

Etter prosjektarbeidet forventes det at vi har lært om/utvidet vår kunnskap om:

- Java/ HTML / Ajax / SQL
- Database - MariaDB
- Server side programmering
- Web/Klientprogrammering

1.3 Målgruppe

1.3.1 Rapporten

Målgruppen for bachelorrapporten vil i hovedsak være IT-studenter og lærere ved NTNU Gjøvik, samt oppdragsgiver.

1.3.2 Oppgaven

Oppgavens målgruppe er oppdragsgiver. Ettersom vi har vært nødt til å avgrense oppgaven, og har laget løsningen som en egen modul, vil vårt produkt kreve mer arbeid for at den skal kunne implementeres i ETC sin eksisterende løsning.

1.4 Egen bakgrunn og kompetanse

1.4.1 Nåværende kompetanse

Gruppen har i løpet av studietiden utviklet generell kompetanse innenfor fagene som er en del av utdanningsløpet. Av disse har fagene som omhandlet programmering, database og systemutvikling vært av nytte i forbindelse med denne oppgaven. Videre har gruppemedlemmene bidratt med kompetanse fra ulike valgfag

som også har vært relevant i forhold til oppgaven. Dette er fag som har gitt kompetanse innen java-utvikling, bruk av APIer og sikkerhetsvurderinger.

1.4.2 Hva må læres?

Ut i fra oppgaveteksten og samtaler med oppdragsgiver forsto vi tidlig at det er en hel del som må læres for å kunne gjennomføre denne oppgaven. Gruppemedlemmene hadde begrenset kunnskap til webapplikasjoner og utviklingen i forbindelse med dette. Det var derfor nødvendig å lære seg HTML, Javascript og Ajax, da kompetansen innen disse teknologiene blant medlemmene var lav. Vi merket også at repetering av SQL i forbindelse med database var nødvendig. Selv om vi fra før hadde generell kompetanse innen Java og SQL var det mye som måtte læres.

1.5 Rammer

1.5.1 Ansvarsforhold og roller

Ettersom vi kun er to på gruppa kommer ingen av oss til å ha noen spesifiserte roller. Vi kommer derimot la en av oss ha hovedansvar for kommunikasjon med veileder og oppdragsgiver, slik at de kun har én person å forholde seg til når det kommer til kommunikasjon.

1.5.2 Rutiner og regler i gruppa

- Forventet arbeidsmengde fra hvert gruppemedlem er 30 timer pr. uke.
- Faste arbeidstimer mandag og tirsdag fra klokken 10 til 16 hvor vi jobber sammen, vi legger opp til mer fleksible arbeidstimer ellers.
- Hvert gruppemedlem skriver ned egen timelogg.
- Aktiv deltagelse i arbeidet, vi forventer at hvert gruppemedlem yter sitt beste, og gjør sitt for at deres oppgaver er gjort innen avtalt tid.
- Gruppens karaktermål for hele Bacheloroppgaven er en C, men vi som ambisjoner å strekke oss opp mot en B.
- Det forventes at gruppemedlemmene tar opp problemer de har med arbeidet, gir tidlig beskjed om de er på etterskudd med arbeid, og evt. komme til enighet om hvordan det skal løses.
- Gruppemedlemene plikter tidlig å informere dersom de ikke er fornøyd med innsatsen til enkelte gruppemedlemme.
- Gruppemedlemene plikter å gi beskjed så tidlig som mulig dersom de ikke kan møte til avtalte tider.

Rutiner ved alvorlige brudd:

- Samtaler mellom gruppedeltakerne om problemet.
- Meld fra om problemene til veileder.

1.6 Øvrige roller

Vår oppdragsgiver, Electric Time Car AS, representert av daglig leder Dag L. Solhaug, bidro hovedsaklig gjennom ukentlige møter med teknisk hjelp og prosessveiledning. Om vi skulle ha problemer var de også tilgjengelig på epost.

Frode Haug fra NTNU Gjøvik har vært en ressurs for gruppen, gjennom hans rolle som veileder. Gruppen har hatt ukentlige møter med han tidlig i arbeidet, hvor han har bistått med prosessveiledning for bacheloroppgaven. Videre har han vært tilgjengelig for oss ved behov, og vi har fått gode råd og veiledning fra han.

1.7 Rapportorganisering

1 - Innledning
Innledningen introduserer oppgaven og rapporten. Den forteller litt om hva som skal gjøres, hvilken kompetanse gruppemedlemmene har, rammer for oppgaven og rapportstrukturen.

2 - Kravspesifikasjon
Kravspesifikasjonen beskriver de funksjonelle og ikke-funksjonelle kravene. Den inneholder også use case og sekvensdiagram for oppgaven.

3 - Design
I dette kapittelet diskuteres teknisk og grafisk design. Hvordan arkitekturen til programvaren er bygget opp, og det grafiske designet av webapplikasjonen.

4 - Implementasjon
Denne delen tar for seg utviklingsprosessen og implementasjonen av utviklingen. Det forklares detaljert hvordan vi har løst hver enkelt del av oppgaven.

5 - Avslutning
I dette kapittelet drøfter vi resultatet av prosjektet, hva som kunne vært gjort annerledes, kritikk av oppgaven og eget arbeid.

Vedlegg
Vedlegg for rapporten.

Kapittel 2

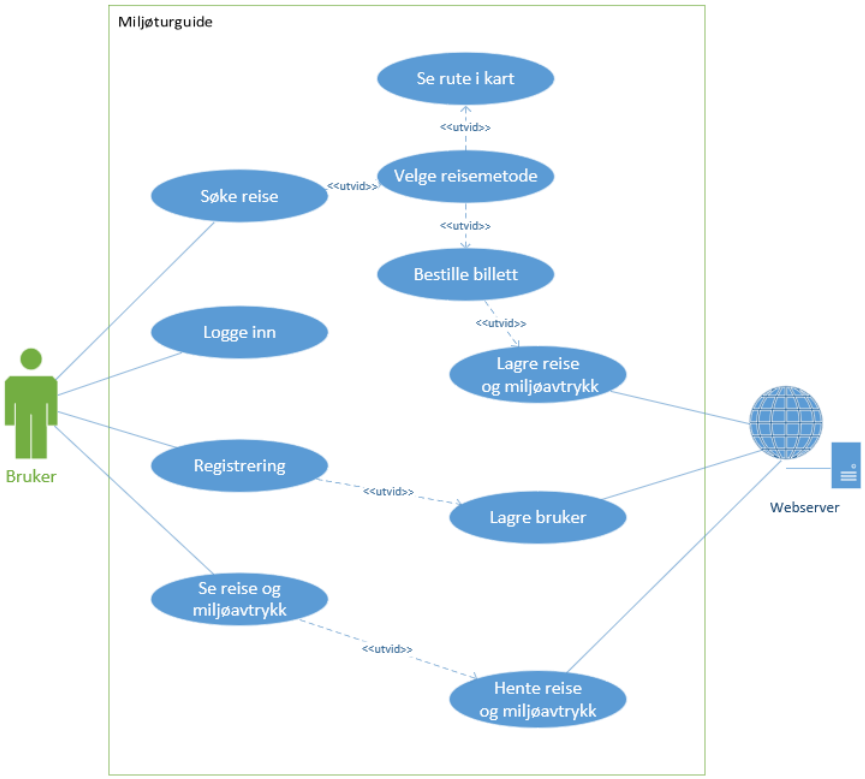
Kravspesifikasjon

Systemutviklingsmodellen vi skal benytte oss av i arbeidet med bachelorprosjektet er en kombinasjon av en inkrementell modell og gjenbruksmodellen. Typen inkrementell modell er ”staged delivery model”. I denne typen av modellen jobber en med et og et inkrement. Dette føles mest hensiktsmessig ettersom vi kun er to på gruppen.

Gjenbruksmodellen benyttes ettersom vi bruker open source rammeverk for loginsystemet og til serverdelen av prosjektet, samt at vi skal benytte oss av flere tredjeparts API-er, bl.a Entur sin API for å få tilgang på informasjon om kollektivtransport i Norge.

2.1 Use case-diagram

I figur 2.1 ser vi use case diagrammet for webapplikasjonen. Use case diagrammet forteller hvilke funksjoner brukeren har tilgang til i webapplikasjonen, og hva som skjer med serveren for disse funksjonene.



Figur 2.1: Use case diagram

2.2 Use cases

Use Case	**Registrering**
Aktør	Bruker
Hensikt	Brukeren registrerer seg i systemet
Beskrivelse	En bruker må registrere seg i systemet for å få tilgang til webapplikasjonens funksjoner.

Use Case	**Logge inn**
Aktør	Bruker
Hensikt	Brukeren logger seg inn i systemet
Beskrivelse	En bruker kan logge seg inn i sytemet med epost og passord etter å ha registrert seg.

Use Case	**Søke reise**
Aktør	Bruker
Hensikt	Brukeren søker etter en reise
Beskrivelse	En bruker velger start-og stoppested, dato, klokkeslett og transportmetode for reisen. Når brukeren trykker på søke-knappen vil de få opp alle de ulike reisealternativene for reisen og informasjon om distanse, tidsbruk og miljøutslipp for hvert valg.

Use Case	**Se rute i kart**
Aktør	Bruker
Hensikt	Brukeren får se valgt rute i kart
Beskrivelse	Etter at brukeren har søkt reise får brukeren se reisen visualisert på et kart.

Use Case	**Velge reisemetode**
Aktør	Bruker
Hensikt	Brukeren velger mellom ulike transportmetoder
Beskrivelse	Etter at brukeren har valgt hvor og når de vil reise, kan de velge mellom ulike transportmetoder. Dette oppdaterer søket og brukeren får opp reisealternativer med valgt transportmetode.

Use Case	**Bestille billett**
Aktør	Bruker
Hensikt	Brukeren bestiller billett basert på valgt reisealternativ
Beskrivelse	Da brukeren har valgt reisealternativ blir brukeren videresendt til en bekreftelsesside for bestilling av billett til valgt reisealternativ.

Use Case	**Se reise og miljøavtrykk**
Aktør	Bruker
Hensikt	Bruker får se sine reiser og miljøavtrykk
Beskrivelse	En bruker kan få se alle sine registrerte reiser og miljøavtrykk pr reise, men også sitt totale miljøavtrykk.

Use Case	**Lagre reise og miljøavtrykk**
Aktør	Server
Hensikt	Server lagrer reisen og miljøavtrykket for reisen i databasen
Beskrivelse	Når en bruker har bekreftet reise og bestilt billett lagres reisen med miljøavtrykket for brukeren i databasen.

Use Case	**Lagre bruker**
Aktør	Server
Hensikt	Server lagrer bruker i databsen
Beskrivelse	Etter en bruker har registrert seg blir brukeren lagret i databasen.

Use Case	**Hente reise og miljøavtrykk**
Aktør	Server
Hensikt	Server henter reise og miljøavtrykket til en bruker fra databasen
Beskrivelse	Når en bruker vil se sine reiser og miljøavtrykk gjennom use caset ”se reise og miljøavtrykk” henter serveren brukerens reiser og miljøavtrykk fra databasen.

2.3 Detaljert use case-beskrivelse

Use Case	**Registrering**
Aktør	Bruker
Hensikt	Brukeren registerer seg i systemet
Pre-betingelse	Brukeren har ingen registrert konto i systemet
Post-betingelse	Ingen
Detaljert hendelsesforløp	1. Brukeren registrerer seg med epost og passord. 2. Systemet bekrefter om informasjonen fra brukeren er gyldig. 3. Systemet lagrer brukerinformasjonen i databasen. 4. Brukeren videresendes til login siden.
Alternative scenarier	Systemet får ikke kontakt med databasen og får ikke lagret brukerinformasjonen.

Use Case	**Se rute i kart**
Aktør	Bruker
Hensikt	Vis ruten til et reisealternativ i kart
Pre-betingelse	Brukeren er innlogget
Post-betingelse	Ingen
Detaljert hendelsesforløp	1. Brukeren søker etter en reise. 2. Systemet viser en liste over reiser for valgt dato, klokkeslett og transportmetode. 3. Brukeren velger en reise fra listen. 4. Systemet viser valgt reiserute i kart.
Alternative scenarier	Systemet får ikke kontakt med karttjenesten, som resulterer i at ruten ikke blir vist.

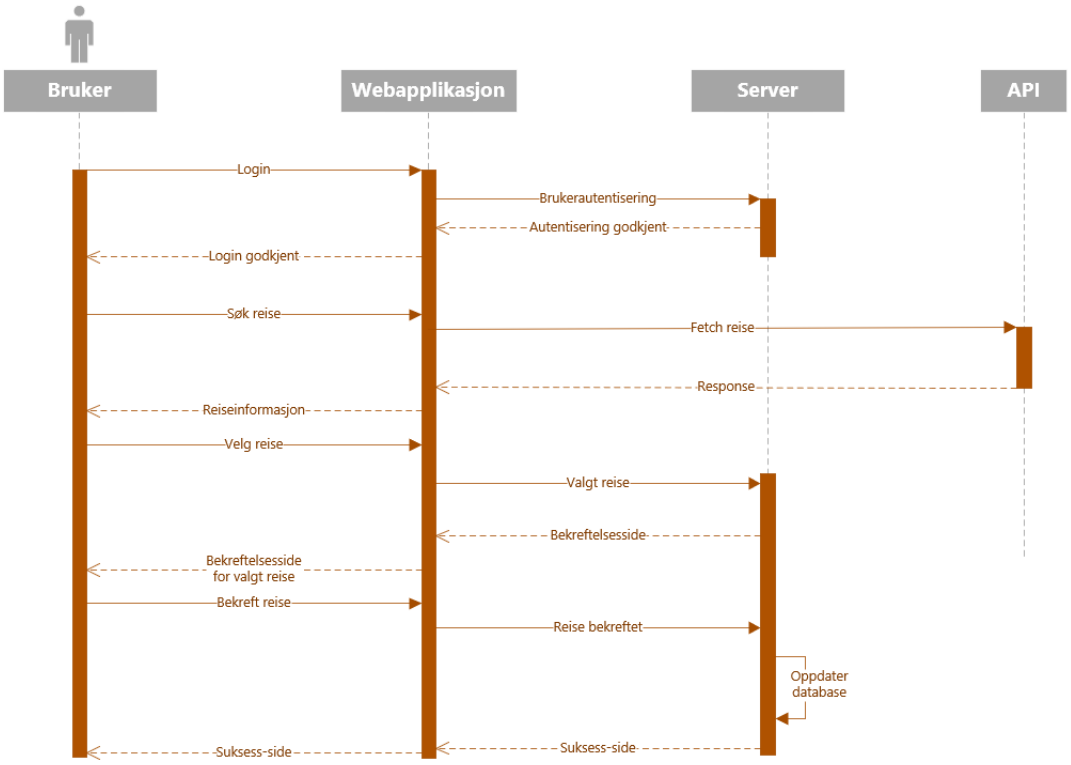
Use Case	**Bestille billett**
Aktør	Bruker
Hensikt	Brukeren bestiller billett for valgt reise
Pre-betingelse	Brukeren er innlogget
Post-betingelse	Ingen
Detaljert hendelsesforløp	1. Brukeren søker etter en reise. 2. Systemet viser en liste over reise for valgt dato, klokkeslett og transportmetode. 3. Brukeren velger en reise fra listen. 4. Systemet sender brukeren til en bekreftelsesside. 5. Brukeren bekrefter eller avbryter kjøpet.
Alternative scenarier	Systemet får ikke kontakt med betalingstjenesten og får ikke kjøpt billett.

Use Case	**Lagre reise og miljøavtrykk**
Aktør	System
Hensikt	Systemet lagrer reise og miljøavtrykk
Pre-betingelse	Brukeren bestiller billett.
Post-betingelse	Ingen
Detaljert hendelsesforløp	1. Brukeren bekrefter valg av reisealternativ. 2. Systemet lagrer informasjon om reisen i databasen.
Alternative scenarier	Systemet får ikke kontakt med databasen og får ikke lagret informasjonen.

Use Case	**Hente reise og miljøavtrykk**
Aktør	System
Hensikt	Systemet returnerer reiser og miljøavtrykk for en bruker
Pre-betingelse	Brukeren klikker på ”Mine Reiser” i nedfallsmenyen.
Post-betingelse	Ingen
Detaljert hendelsesforløp	1. Brukeren klikker på ”Mine Reiser”. 2. Systemet henter reiser og miljøavtrykk for brukeren fra databasen.
Alternative scenarier	Systemet får ikke kontakt med databasen og får ikke hentet informasjonen.

2.4 System sekvensdiagram

Figur 2.2 viser et system sekvensdiagram for et normalt hendelsesforløp for systemet vårt.



Figur 2.2: Sekvensdiagram for normal hendelsesforløp

2.5 Ikke-funksjonelle krav

2.5.1 Brukervennlighet

Det skal være fokus på at webapplikasjonen som utvikles er brukervennlig. En skal ikke trenge mye erfaring med datamaskiner for å forstå hvordan webapplikasjonen benyttes.

- Knapper med funksjonalitet skal ha en hjelpetekst som forklarer knappens funksjon.
- Ikoner og ikonenes funksjonalitet skal være selvforklarende.
- Tekst og ikoner skal ha en størrelse som føles naturlig uten at brukeren må anstrenge seg for å kunne lese teksten.
- Knapper, tekstbokser og ikoner skal være logisk plassert.

- Feilmeldinger skal være forklarende slik at brukeren forstår hva som har forårsaket feilen.

2.5.2 Operasjonelle krav

Vi har ikke blitt gitt noen spesifikke operasjonelle krav, men har tenkt ut noen punkter som er vesentlige for vår webapplikasjon:

- Responstiden på nettverkskall og lasting av informasjon bør ikke overstige tre sekunder.
- Webapplikasjonen skal være enkel å vedlikeholde.

2.5.3 Sikkerhet

2.5.3.1 Systemressurser

Vi har identifisert tre ressursser vi anser som system-kritiske. Angrep som fører til innbrudd og/eller forstyrrelser av disse vil ha varierende konsekvenser for selve systemet, organisasjonen som drifter og vedlikeholder applikasjonen, og brukere av systemet. Disse står beskrevet under, og er listet i Figur 2.3, hvor de er rangert fra lav til høy prioritet vurdert ut i fra konsekvenser angrep på systemressursene kan ha.

- Server :
    - Hardware og software som applikasjonen kjører på. Håndterer tilgang på og utlevering av brukerdata.
- Database :
    - Lagrer all brukerdata som systemet skal håndter
- Brukerdata :
    - All informasjon knyttet til enkelt profiler. Profil informasjon, reiseplaner og personlig identifiserende informasjon.

ID	Navn	Beskrivelse	Prioritet
A1	Server	Hardware og software som applikasjonen kjører på. Gjør bruker autentisering og autorisering, og håndterer tilgangen på og utleveringen av brukerdata.	Høy
A2	Database	Lagrer all brukerdata som systemet skal håndtere	Høy
	Brukerdata		
A3.1	Profildetaljer	Brukernavn og passord.	Høy
A3.2	Reisedata	Data om planlagte turer, tidligere turer.	Middels
A3.3	Personinformasjon	Personlig identifiserende informasjon.	Lav

Figur 2.3: Identifiserte systemressurser

Begrunnelse av prioritering

Vurderingene er gjort ut i fra konsekvenser angrep mot systemet kan ha på CIA-prinsippene om datasikkerhet[1] (Konfidensialitet, Integritet og Tilgjengelighet, på engelsk; Confidentiality, Integrity, Availability) for systemressursene.

A1 Server

Forstyrelser av serveren som systemet kjører på kan føre til at hele systemet blir utilgjengelig. Bryter med CIA-prinsippet om tilgjengelighet, og lengre perioder med utilgjengelighet kan ha store konsekvenser for organisasjonen som drifter og vedlikeholder applikasjonen. Prioriteres derfor høyt.

A2 Database

Angrep mot databasen kan føre til tap av data, uautorisert endring av data og at dataene blir utligjengelige. Bryter med alle de tre CIA-prinsippene, og prioriteres derfor høyt.

A3.1 Profildetaljer

Angrep som gjør at profildetaljer kommer på avveie kan føre til uautorisert tilgang til brukerkontoer. Angripere med tilgang til brukerprofiler vil ha tilgang til all brukerinformasjon og brukerfunksjoner. Angripere kan også låse brukerne ute av sine egne kontoer. Bryter med prinsippene som konfidensialitet og tilgjengelighet. Prioriteres høyt, spesielt med tanke på at noen brukerfunksjoner er knyttet til bestilling og fakturering av reiser.

[1] `https://whatis.techtarget.com/definition/Confidentiality-integrity-and-availability-CIA`

A3.2 Reisedata

Uautorisert tilgang på brukeres reisedata kan fortelle angripere om brukeres reisevaner og reiseplaner. Kan brukes til å finne ut hvor personer kommer til å være på bestemte tidspunkter. Bryter med prinsippet om konfidensialitet. Priotiteres som middels viktig.

A3.3 Personinformasjon

Uautorisert tilgang på brukeres personinformasjon bryter med prinsippet om konfidensialitet. Prioiteres lavt ettersom systemet ikke skal håndtere betydelige mengder personlig identifiserende informasjon per bruker.

2.5.3.2 Tiltak

For å beskytte systemressursene planlegger vi følgende:

- Vi skal benytte oss av Spring Security for autorisering og autentisering. Spring Security lar oss autentisere forespørsler så vi kun gir tilgang til autoriserte brukere. Ved bruk av Spring Security vil vi kryptere passordet til brukerene. Den krypterte versjonen av passordet vil bli lagret i databasen.
- For beskytte data under overføring vil vi også benytte oss av SSL for å gå over fra HTTP til HTTPS.

2.5.4 Lisensiering

Flere API-er vi vil benytte er lisensiert under ”Norwegian Licence for Open Government Data (NLOD)”. Dette gjør at vi kan:

- Kopiere og tilgjengeliggjøre.
- Endre og/eller sette sammen andre datasett.
- Kopiere og tilgjengeliggjøre en endret eller sammensatt versjon.
- Benytte datasettet kommersielt.

På følgende vilkår:

- Vi navngir lisensgiver slik lisensgiver ber om, men ikke på en måte som indikerer at disse har godkjent eller anbefaler vår bruk av datasettet.
- Vi ikke bruker dataene på en måte som fremstår som villedende, og heller ikke fordreier eller uriktig fremstiller dataene[10].

For å benytte API-ene må brukerene også identifisere seg ved å bruke headeren ”ET-Client-Name” når API-kallene gjøres. Vi identifiserer oss med ”NTNU - Miljøturguide Bacheloroppgave”

2.5.5 Kodekvalitet

Det vil være fokus på å utvikle god kode. All kode som skrives skal dokumenteres med java docs og kommentering. Dette hjelper oss med å forstå både hva ikke-egenutviklet kode gjør, men også å huske hva egenutviklet kode gjør. Samtidig vil også ETC enklere kunne sette seg inn i tanken og målet med koden ved leveranse.

2.6 Backlog

- DB-design - Konseptuelt design av DB.
- Server - Registrere bruker - Når en bruker registrerer seg i webapplikasjonen skal brukeren lagres i databasen.
- Web - Registrere bruker - Webside for registrering.
- Server - Hente bruker - Hente informasjon om brukeren.
- Web - Brukerkonto - Webside med brukerinformasjon til en bruker.
- Web - Søke reise - Finne reiser gjennom API-kall fra en destinasjon til en annen for en gitt dato og tid.
- Web - Velge reisemetode - Velge reise fra resultatene av søket.
- Web - Se rute i kart - Få opp ruten til valgt reise i kartet.
- Web - Bestille billett - Webside for bekreftelse av reise og kjøp.
- Server - Lagre reise - Lagre reisen og miljøavtrykket i databasen.
- Web - Se reise og miljøavtrykk - Webside for en brukers reise og miljøavtrykk.
- Web - Innlogging - Webside for innlogging.
- Server - Hente reise og miljøavtrykk - Hente informasjon om en brukers reiser og miljøavtrykk.
- Server - Innlogging - Autentisering av bruker ved login.
- Server - Utslipp - Hente informasjon om transportmetodenes utslipp.

Kapittel 3

Design

3.1 Arkitektur

Applikasjonen vår er bygget med Spring Framework. Spring Framework er et rammeverk som tilbyr et sett med nyttige moduler. Av de ulike modulene som er tilgjengelig har tatt i bruk disse:

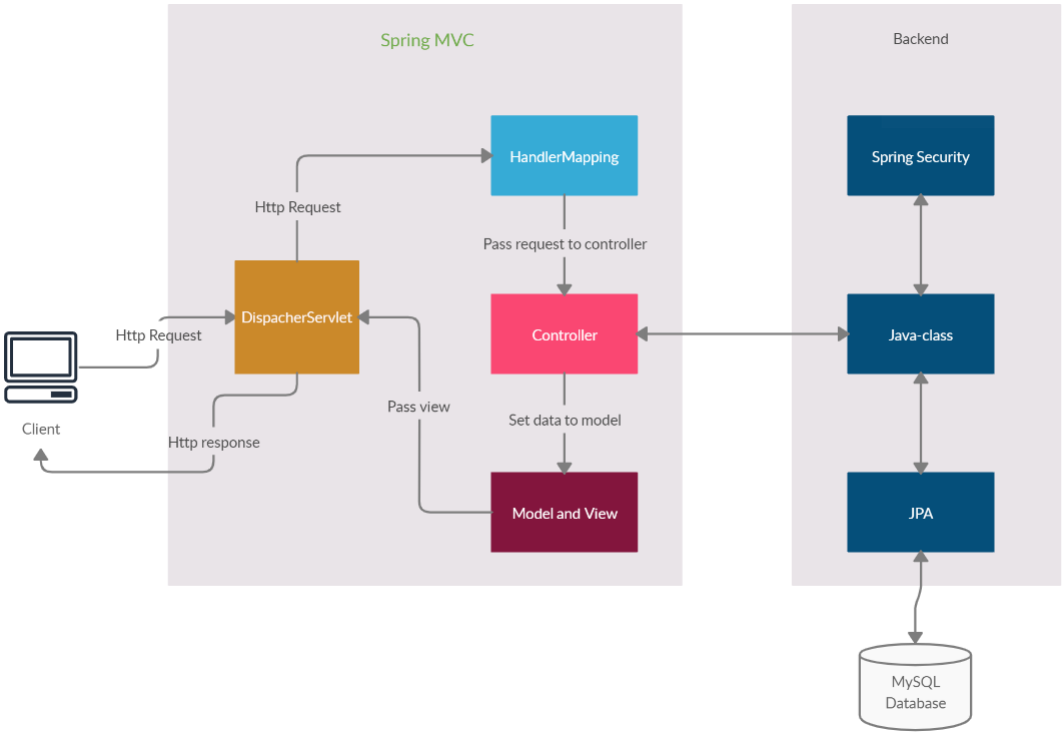
- Spring Boot
- Spring Security
- Spring MVC
- Spring data JPA

Figur 3.1 viser webapplikasjonens overodnede arkitektur. En klient sender en http forespørsel til en dispatcherservlet. Denne sender forespørselen videre til handlermapping. Handlermapping sørger for å sende forespørsel til riktig kontroller. Kontrolleren oppretter objekter av javaklasser etter behov. Disse objektene blir fylt med informasjon enten ved hjelp av ”Thymeleaf”, eller med informasjon hentet fra databasen. Ved bruk av Thymeleaf blir objektene fylt med informasjon fra HTML-siden. Denne informasjonen kan f.eks. være user-input som er nødvendig å lagre. Informasjonen som blir lagret i objektet blir sendt til JPA-repository som er en modul som blir brukt for å oppdatere, eller hente informasjon fra databasen.

Videre setter kontrolleren dataen til modellen. Modellen sender videre et view tilbake til dispatcherservlet, som igjen sender dette til klienten. View er da det klienten ser.

3.1.1 Spring Boot

Spring Boot er et verktøy som lar oss bygge applikasjonen raskt og enkelt. Ved bruk av Spring Boot slapp vi å tenke på implementasjon av server da Tomcat-serveren allerede er ferdig implementert og en del av rammeverket.



Figur 3.1: Arkitektur for webapplikasjon

3.1.2 Spring Security

Spring Security tilbyr funksjonalitet for autentisering og autorisasjon til webapplikasjonen vår. Som nevnt tidligere lar Spring Security oss autentisere forespørsler så vi kun gir tilgang til autoriserte brukere.

3.1.3 Spring MVC

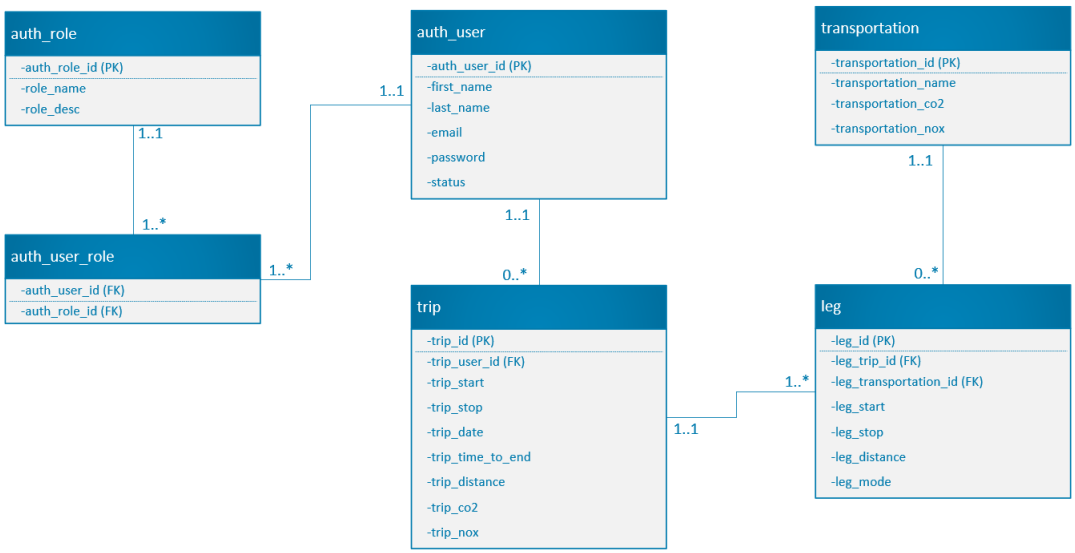
Spring MVC er en modul som er bygget rundt ”Model-View-Controller”-arkitekturen. Spring MVC benytter en DispatcherServlet som tar seg av HTTP-forespørsler og responser. Klienten sender en forespørsel til DispatcherServlet som videresender til ”HandlerMapping” som igjen videresender til riktig Controller. Controlleren manipulerer Modellen og fyller den med data. Modellen oppdaterer View som blir videresendt tilbake til DispatcherServlet som sender responsen tilbake til klienten.

3.1.4 Spring data JPA

Spring data JPA er et verktøy som forenkler kommunikasjonen med database. JPA gjør så vi kan kommunisere med databasen via java-objekter. Dette gjøres ved å lage ”repository-interfaces” og skrive egen kode for å lagre objekter eller

finne objekter i repositoriet. Implementasjonen skjer automatisk ved hjelp av Spring.

3.1.5 Database

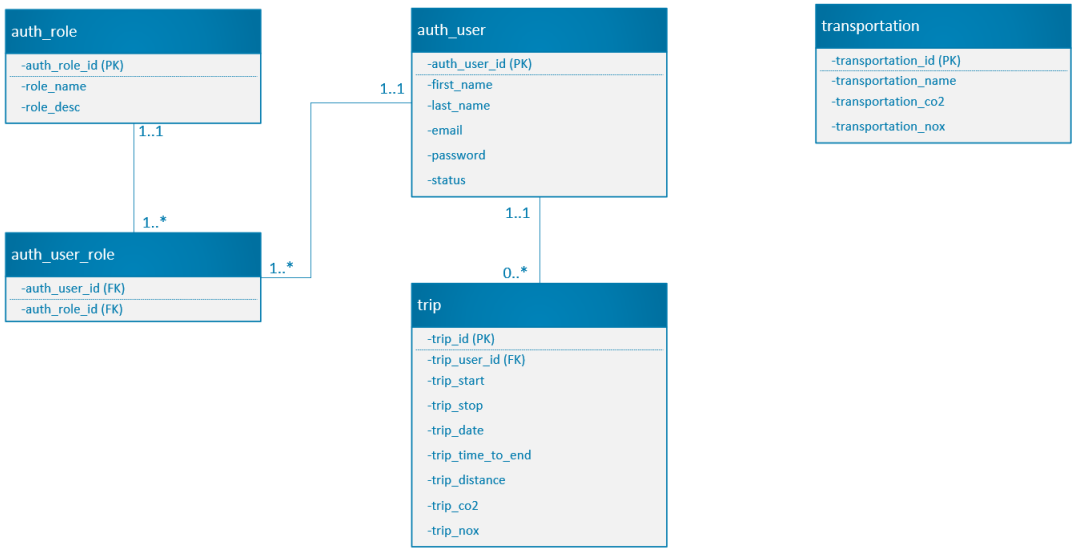


Figur 3.2: Databasedesign

Figur 3.2 viser tiltenkt design for database. Planen var at når en bruker søker og bekrefter en reise, skal reisen lagres i databasen, og hver etappe for den reisen skal lagres i en annen tabell. Grunnet problemer med implementeringen av dette, måtte vi gå bort fra denne løsningen da vi ikke hadde mer tid å bruke på utvikling.

Det endelige resultatet ser vi i figur 3.3. Dette ble en del annerledes enn tenkt. Vi fikk ikke til å implementere lagringen av hver enkelt etappe. I stedet henter vi utslippstall for hver transportmetode lagret i databasen ved innlasting av hovedsiden i webapplikasjonen. Disse utslippstallene blir deretter brukt for å beregne utslipp for hver reise ut i fra informasjon vi får fra API-kall.

Når en bruker registrerer seg vil den bli lagret i databasen med fornavn, etternavn, email, passord og status. Samtidig vil tabellen kalt ”authUserRole” bli oppdatert med brukerens id og rolle-id som den ble tildelt. Rolle-id vil være lik det som gir vanlige rettigheter for applikasjonen. Når brukeren søker og bekrefter en reise, vil reisen bli lagret i databasen med informasjon om startsted, sluttsted, dato, tidspunkt, avstand og utslipp for både co2 og nox.



Figur 3.3: Endelig databasedesign

3.1.6 API

Vi benytter oss av API'er fra Entur for å tilgang til nødvendig informasjon om kollektivtransport i Norge. Entur tilbyr flere API'er knyttet til kollektivtransport. I applikasjonen bruker vi to av dem; Journey Planner og Geocoder.

Geocoder

Geocoder API'et gjør det mulig å gjøre søk etter steder innad i Norge, og gjør det enkelt for oss å få tak i nødvendig informasjon om reisedestinasjoner, som koordinater og stedsnavn. Resultateter fra Geocoder søk kan så brukes som søkevariabler i Journey Planner API kallet.

Journey Planner

Journey Planner er et GraphQL[1] API gir oss mulighet til å søke etter reise-og transport alternativer mellom steder i Norge.

I tillegg benytter vi oss av Mapbox som kartløsning.

3.2 Utforming av Webapplikasjon

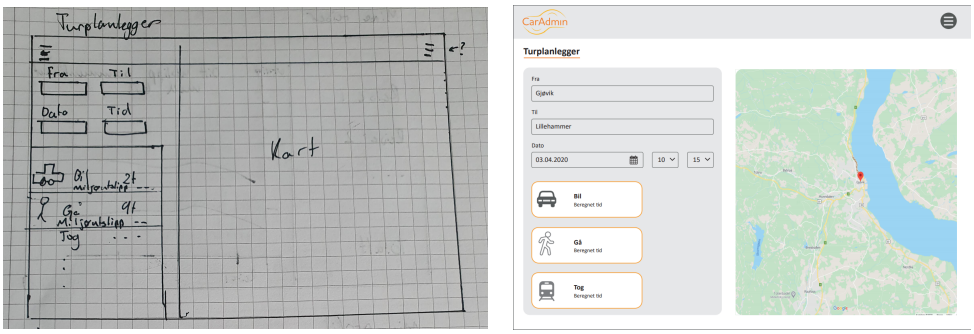
Utformingen og det visuelle designet av webapplikasjonen er basert på tegninger vi fikk fra ETC, som igjen er basert på skisser vi tegnet opp tidlig i prosjektet som

[1]GraphQL er et søkespråk for API'er som gjør det enkelt å gjøre større, mer sammensatte API kall. `https://graphql.org/`

en del av planleggingsfasen.
Vi stod fritt til å selv designe utformingen på nettsidene, og tegningene fra ETC var ment å veilede oss i det visuelle designet, slik at sidene vi utviklet matcher ETC's Car Admin løsning visuelt.
I tilfeller hvor vi ikke hadde konkrete tegninger å gå etter har vi gjort egne besluttninger slik at utformingen best passer resten av designet.

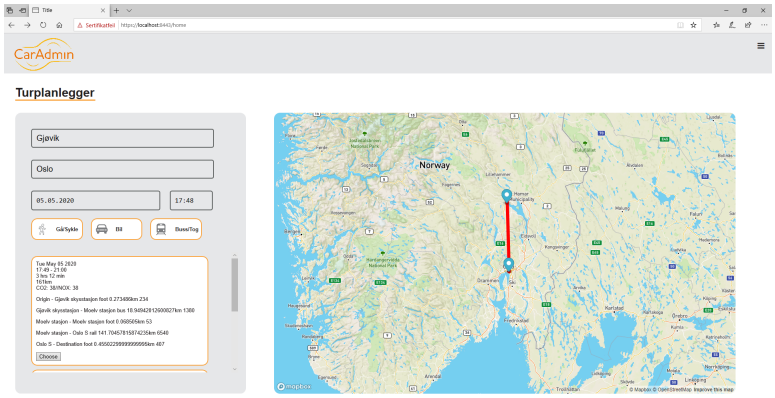
3.2.1 Turplanlegger

Turplanleggeren er utformet slik at brukeren lett skal kunne få visualisert reisen sin. Derfor tar et kart opp en stor del av skjermplassen på høyre side. På venstre side er søkefeltene, hvor brukeren kan søke etter reiser.
Når brukeren holder over søkeknappene endres fargen på kantlinjen fra oransje til grønn. Dette for å tydeliggjøre at disse elementene er knapper. Når knappen trykkes på blir hele knappen grønn. Søkeknappene har et ikon slik at brukeren kan se hvilken transportmetode knappen søker etter.
Når brukeren trykker på en søkeknapp plasseres to markører på kartet som viser start- og stoppested.
Søkeresultatene viser informasjon om dato, klokkeslett, reise tid, reisedistanse og utslippstall(Co2/km og NOx/km).
Feltene som viser søkeresultat kan trykkes på for å utvide feltet og vise alle etappen reisen inneholder. Da tegnes også en linje mellom markørene på kartet.
I det utvidede feltet finner man knappen som velger reisen og tar brukeren videre til bekreftelsessiden.
Vi gjorde en liten endring fra tegningene i implementasjonen vår. Opprinnelig var planen at knappene hvor brukeren velger reisemetoden fyller nedre halvdel av søkefeltet(Figur 3.4 b). Istedet ble de flyttet sammen på samme linje, og er mindre enn på tegningene. Dette ga oss plass til å vise søkeresultatene nederst i søkefeltet, og gjør det mulig for en bruker å kunne endre på reisemetodene, uten å måtte trykke seg tilbake for å få opp søkemetodene igjen(Figur 3.5).



(a) Vår skisse av turplanleggeren **(b)** Tegning fra ETC av turplanleggeren

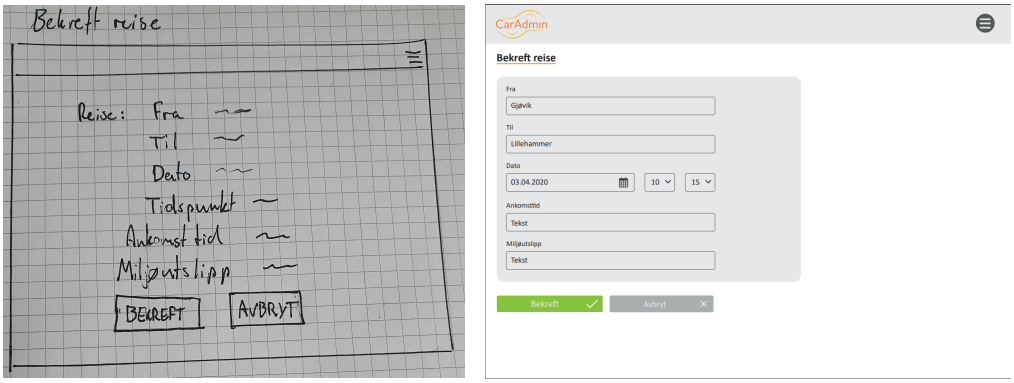
Figur 3.4: Skisse og tegning av turplanleggeren



Figur 3.5: Vår implementasjon av turplanleggeren

3.2.2 Bekreftelsesside

Bekreftelsessiden er ment å være lettlest og oversiktlig. Siden skal oppsummere reisen, og viser den viktigste informasjonen om reisen. Er brukeren fornøyd, trykker de på 'Bekreft' nederst på siden som lagrer reisen.
Implementasjonen (Figur 3.7) gjør noen små endringer fra tegningene (Figur 3.6 b). Selve utformingen er lik, men en ekstra tekstboks med reisedistanse (i kilometer) er lagt til, samt at utslippstallene vises i hver sin boks, en for Co2 og en for NOx.



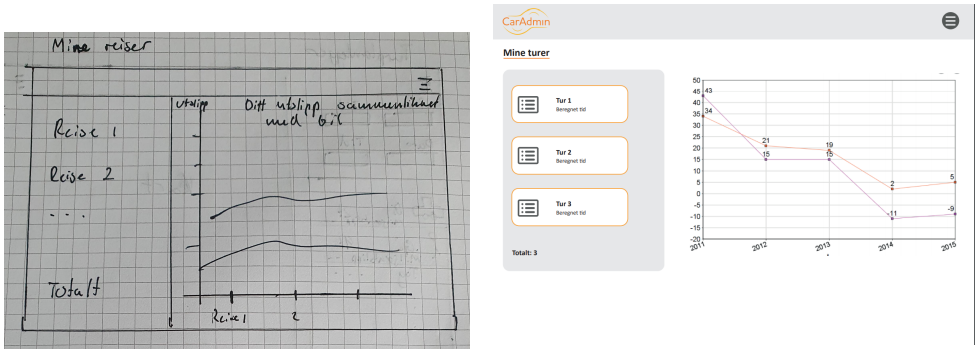
(a) Vår skisse av bekreftelsesside. **(b)** Tegning fra ETC av bekreftelsesside.

Figur 3.6: Skisse og tegning av bekreftelsesside

Figur 3.7: Vår implementasjon av bekreftelsesside.

3.2.3 Reisestatistikk

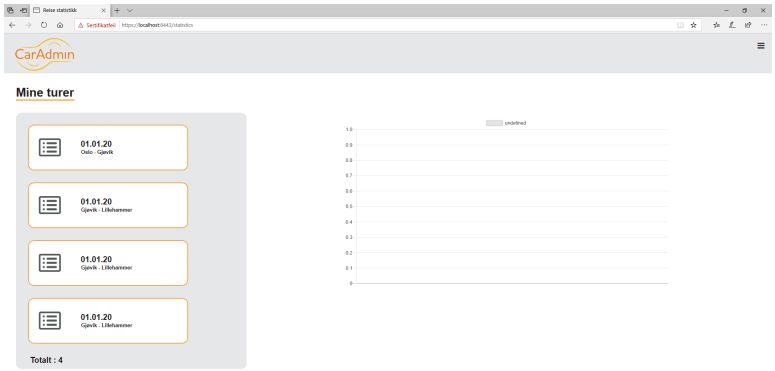
Reisestatistikk viser dokumentasjon av tidligere reiser, og skal gi brukeren mulighet til å se gjennom sine reiser. Oversikt over reisene er listet på venstre side, hver identifiseres med reisedatoen. På høyre side er det en graf som viser utslippstallene for hver etappe av reisen. Grafen viser også hvor hver etappe starter og slutter. Brukeren kan trykke på de forskjellige reisene og få vist oversikt over utslippene for reisen.
Når brukeren trykker på en reise endres bakgrunnsfargen på elementet til en lys gråfarge for å visualisere hvilken reise som vises.
Implementasjonen følger skissene og tegningene fra ETC uten noen endringer i utformingen. Eneste vi endret på i implementasjonen er at hver reise identifiseres med reisedato, og ikke som Reise 1, Reise 2 etc.



(a) Vår skisse av reisestatistikk.

(b) Tegning fra ETC av dokumentasjon av reisestatistikk.

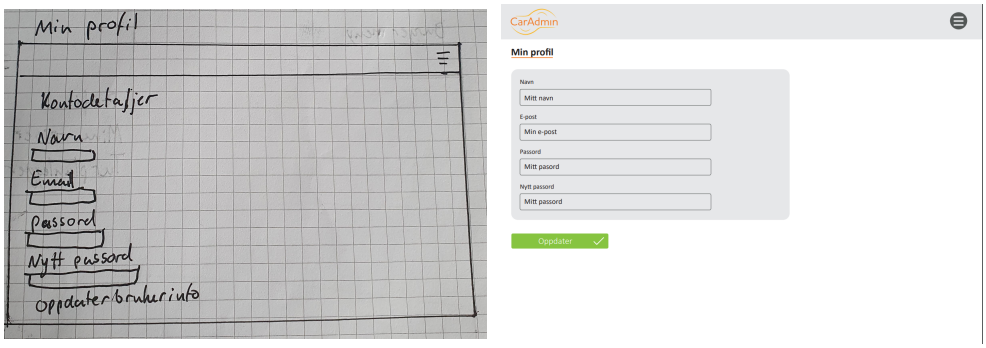
Figur 3.8: Skisse og tegning av reisestatistikk



Figur 3.9: Vår implementasjon av dokumentasjon av tidligere reiser.

3.2.4 Min Profil

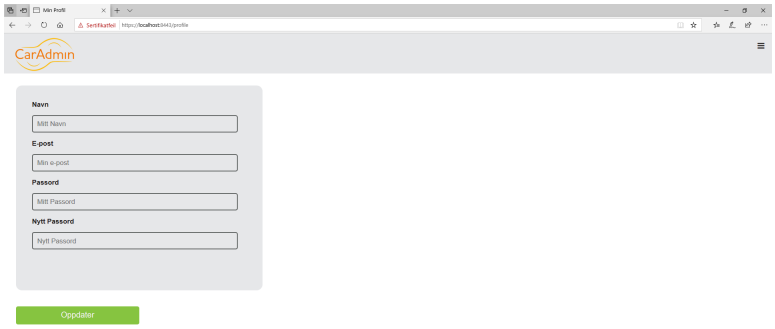
Profilsiden består et skjema slik at brukeren skal kunne endre sine brukerdata. Implementasjonen følger skissene våre og tegningene fra ETC.



(a) Vår skisse profil side.

(b) Tegning fra ETC av profilside.

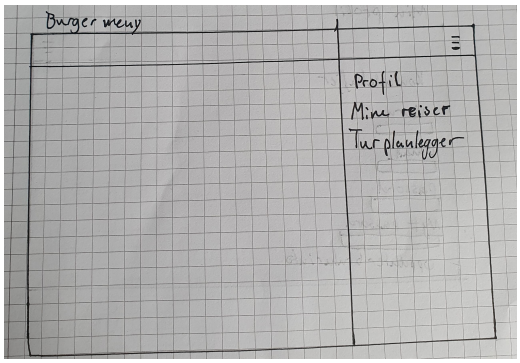
Figur 3.10: Skisse og tegning av profilside



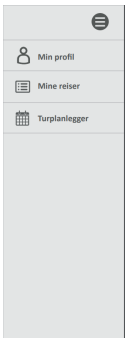
Figur 3.11: Vår implementasjon av profilside.

3.2.5 Navigasjonslinje med nedfallsmeny

Navigasjonslinjen strekker seg over hele toppen av websidene. På venstre side er det en nedfallsmeny som vises når brukeren trykker på en knapp. Implementasjonen og stylingen av navigasjonslinja og nedfallsmenyen følger følger skissene våre og tegningene fra ETC.

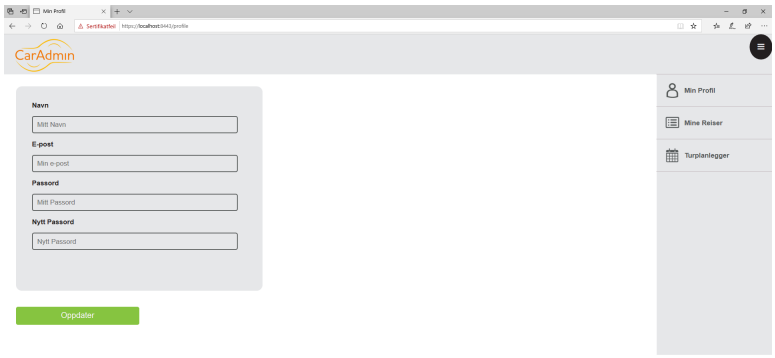


(a) Vår skisse nedfallsmeny.



(b) Tegning fra ETC av nedfallsmeny.

Figur 3.12: Skisse og tegning av nedfallsmeny



Figur 3.13: Vår implementasjon av navigasjonslinje med nedfallsmeny.

Kapittel 4

Implementasjon

4.1 Programmeringsspårk og IDE

Ut i fra oppgavebeskrivelsen var det på forhånd bestemt hvilke programmeringsspråk som skulle benyttes. Java for backend-programmering, HTML, CSS og Javascript for frontend-programmering.

4.1.1 IntelliJ IDEA og Java

Ettersom Java skulle benyttes som programmeringsspråk i backend av systemet vårt valgte vi å benytte oss av IntelliJ IDEA som IDE. Dette gjorde vi ettersom vi allerede hadde erfaring med denne IDE'n fra tidligere emner i studieløpet. I tillegg synes oppdragsgiver at det var en god ide å benytte seg av IntelliJ da de også benytter denne.

4.1.1.1 Maven

I tilegg til å benytte oss av IntelliJ valgte vi å benytte oss av Maven. Maven blir brukt for å fortelle hvordan programmet skal bli bygget opp og hvilke avhengigheter som skal være med.

4.2 Backend og rammeverk

Som nevnt tidligere har vi tatt i bruk Spring rammeverket og flere av modulene dette rammeverket tilbyr. Vi opprettet et Spring prosjekt i IntelliJ som bygget på Spring Boot og Spring Security. Spring Boot initialiserer applikasjonen med en mappestruktur. Dette gjorde det enkelere å komme igang med utviklingen.
Ved bruk av Spring rammeverket benytter en seg av noe kalt "annotations" i programmeringen. Spring bruker "annotations" som et alternativ til XML.

4.2.1 Spring data JPA

For at JPA-modulen i Spring rammeverket skal fungere korrekt må det benyttes spesielle ”annotations” for å få til kommunikasjonen med databasen. I kodeliste 4.1 ser vi et eksempel på bruken av ”annotiations” i forbindelse med JPA-modulen. Klassen ”User” får annotations ”Entity” og ”Table” med navn ”auth_user” hvor navnet er det samme som navnet på tabellen i databasen. Primærnøkkel blir satt med ”Id” som inkrementeres automatisk. Annotation ”Column” forteller navnet på kolonnen i databasen.

Kodeliste 4.1: Spring data JPA klasse

```
package no.ntnu.joname.bachelor.model;

import javax.persistence.*;
import java.util.Set;

@Entity
@Table(name = "auth_user")
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "auth_user_id")
    private int id;

    @ManyToMany(cascade = CascadeType.ALL)
    @JoinTable(name = "auth_user_role",
                joinColumns = @JoinColumn(name = "auth_user_id"),
                inverseJoinColumns = @JoinColumn(name = "auth_role_id"))
    private Set<Role> roles;

}
```

For å kommunisere med databasen lager en et interface som utvider bruken av JpaRepository og benytter ”Repository annotation”. I dette interfacet deklarerer en funksjoner for å kommunisere med databasen. Videre lages det en java-klasse som implementerer dette interfacet. Hvordan dette implementeres ser vi i kodeliste 4.2.

Kodeliste 4.2: Spring data JPA repository

```
package no.ntnu.joname.bachelor.repository;


import no.ntnu.joname.bachelor.model.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface UserRepository extends JpaRepository<User, Integer> {
    User findByEmail(String email);
}
```

4.2.2 Spring Security

Implementasjonen av Spring Security gjorde vi ved hjelp av tutorials. Mye av koden for denne implementasjonen er hentet fra Youtube.[1] Spring Security benyttes for å autorisere forespørsler og sjekke om brukeren har den rette autoriteten. Implementasjonen av Spring Security ser vi i kodeliste 4.3. Som vi kan se fra koden sjekkes det om den som sender forespørselen har rettighetene ”Super_User”, ”Admin_User” eller ”Site_User”. I vårt system benytter vi oss kun av ”Site_User”.

Kodeliste 4.3: Spring security implementasjon

```
@Override
protected void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests()
        // URLs matching for access rights
        .antMatchers("/").permitAll()
        .antMatchers("/login").permitAll()
        .antMatchers("/register").permitAll()
        .antMatchers("/home/**").hasAnyAuthority("SUPER_USER",
         "ADMIN_USER", "SITE_USER")
        .anyRequest().authenticated()
        .and()
        // form login
        .csrf().disable().formLogin()
        .loginPage("/login")
        .failureUrl("/login?error=true")
        .defaultSuccessUrl("/home")
        .usernameParameter("email")
        .passwordParameter("password")
        .and()
        // logout
        .logout()
        .logoutRequestMatcher(new AntPathRequestMatcher("/logout"))
        .logoutSuccessUrl("/").and()
        .exceptionHandling()
        .accessDeniedPage("/access-denied");
}
```

4.2.3 Controller

Controller-klassen er den som styrer hva som blir vist til klienten ved hjelp av Spring MVC. Når klienten gjør en forespørsel blir ”RequestMethod.Get” benyttet og riktig HTML-side lastet inn basert på url. Et eksempel på implementasjonen av hvordan forespørslene håndteres ser vi i kodeliste 4.4. I eksempelet lages et ”ModelAndView”-objekt når forespørselen blir sendt. Videre blir det laget et nytt ”User”-objekt. Dette user-objektet blir lagt til i ModelAndView-objektet.
Ved hjelp av Thymeleaf blir user-objektet fylt med verdier fra user-input fra registreringssiden. Dette ser vi i kodeliste 4.5 hvor en definerer objektet Thymeleaf skal bruke. Dette objektet vil da være ”user” som ble lagt til i ModelAndView-objektet som er et user-objekt. Videre korresponderer ”th:field”

[1] https://www.youtube.com/watch?v=HLSmjZ5vN0w

med de private verdiene til user-objektet, så f.eks. ”name” vil få verdien som brukeren taster inn i teksteltet ”Name”.

Kodeliste 4.4: Controller Get Mapping metode for registreringsside

```
@RequestMapping(value = "/register", method = RequestMethod.GET)
public ModelAndView register() {
        ModelAndView modelAndView = new ModelAndView();
        User user = new User();
        modelAndView.addObject("user", user);
        modelAndView.setViewName("register"); // resources/template/register.html
        return modelAndView;
}
```

Kodeliste 4.5: Thymeleaf eksempel

```
<div class="row␣col-lg-4␣col-lg-offset-4">
    <form autocomplete="off" action="#" th:action="@{/register}"
          th:object="${user}" method="post" class="form-signin" role="form">
        <h3 class="form-signin-heading">Registration Form</h3>
        <div class="form-group">
            <div class="">
                <input type="text" th:field="*{name}" placeholder="Name"
                       class="form-control" />
            </div>
        </div>
```

Knappen på registreringssiden har metoden POST. Når brukeren har fylt inn tekstfeltene på registreringssiden og klikker på knappen for å registrere brukeren vil ”RequestMapping-annotation” med metoden ”RequestMethod.Post” sørge for at ”registerUser”-funksjonen blir kalt. Denne funksjonen sjekker om det allerede finnes en bruker registrert på emailen som er skrevet inn i tekstfeltet. Om dette ikke er tilfellet blir brukeren lagret i databasen.

Kodeliste 4.6: Post mapping metode

```
@RequestMapping(value="/register", method=RequestMethod.POST)
public ModelAndView registerUser(@Valid User user,
        BindingResult bindingresult, ModelMap modelMap) {
        ModelAndView modelAndView = new ModelAndView();
        //Check for the validations
        if(bindingresult.hasErrors()) {
        modelAndView.addObject("successMessage",
        "Please␣correct␣the␣errors␣in␣form!");
        modelMap.addAttribute("bindingResult", bindingresult);
        }
        //We will save the user if no binding errors
        else if(userService.isUserAlreadyPresent(user)) {
        modelAndView.addObject("successMessage", "user␣already␣exists!");
        }
        else {
        userService.saveUser(user);
        modelAndView.addObject("successMessage","User␣registered␣successfully!");
        }
        modelAndView.addObject("user", new User());
        modelAndView.setViewName("register");
        return modelAndView;
}
```

4.2.4 Database

Databasen er opprettet og implementert med SQL. Et eksmpel på en tabell i databasen og implementasjonen av denne ser vi i kodeliste 4.7. Tabellen blir opprettet med en primærnøkkel, og verdier vi ser nødvendig å lagre. Videre har den referanser ved bruk av fremmednøkler fra andre tabeller.

Kodeliste 4.7: Database-implementasjon

```
CREATE TABLE trip (
    trip_id int(11) NOT NULL AUTO_INCREMENT,
    trip_user_id int(11) NOT NULL,
    trip_start varchar(255) NOT NULL,
    trip_stop varchar(255) NOT NULL,
    trip_date varchar(255) NOT NULL,
    trip_time_to_end varchar(255) NOT NULL,
    trip_distance int(11) NOT NULL,
        trip_co2 int(11) NOT NULL,
        trip_nox int(11) NOT NULL
    PRIMARY KEY (trip_id),
    KEY FK_trip_user_id (trip_user_id),
    CONSTRAINT FK_trip_auth_user FOREIGN KEY (trip_user_id)
        REFERENCES auth_user (auth_user_id)

);
```

4.3 Web

4.3.1 Navigasjonsmeny

Navigasjonsmenyen vises på toppen av hver webside, og inneholder CarAdmin logoen og en nedfallsmeny. Menyen inneholder linker til tre av de implementerte sidene; 'Turplanlegger', 'Profilside' og 'Mine reiser'. Navigasjonsmenyen består av et element 'topBar', og inneholder en navigasjonsliste <nav> kalt 'topnav'. Listen inneholder url linker til sidene. Linkene defineres av <a> taggen i HTML, og vi må bruke 'href' attributtet for å spesifisere url adressen. Eksempelvis vil HTML'en som lager linken til statistikk siden være '<a class = "linkContainer" href="/statistics">'. Hvert listeelement inneholder et icon og en beskrivelse.

Kodeliste 4.8: HTML som lager navigasjonsmeny.

```
<div class = "topBar">
        <img src = "https://caradmin.no/wp-content/uploads/2019/08/
caradmin_logo_web.png" style="height:85px" >

        <nav class="topnav">
            <div id="myLinks">
                <a class = "linkContainer" href="/profile">
                    <img src = "/icons/myProfileIcon.png" style=
"height:40px;width:40px;padding:10px;margin:0;">
                    <div class = "link"><b>Min Profil</b></div>
                </a>
                <a class = "linkContainer" href="/statistics">
                    <img src = "/icons/myTripsIcon.png" style=
```

```
"height:40px;width:40px;padding:10px;margin:0;">
                    <div class = "link"><b>Mine Reiser</b></div>
                </a>
                <a class = "linkContainer" href="/home">
                    <img src = "/icons/journeyPlannerIcon.png" style=
"height:40px;width:40px;padding:10px;margin:0;">
                    <div class = "link"><b>Turplanlegger</b></div>
                </a>
            </div>
            <a href="javascript:void(0);" class="icon" onclick="menuExpand()">
                <i class="fa␣fa-bars"></i>
            </a>
        </nav>
    </div>
```

Alle websidene vi har lagd bruker CSS for å styles. Navigasjonsmenyen styles med CSS-filen topBar.css. Kodeliste 4.9 viser hvordan eksempelvis elementene topBar og topnav er stylet.

Kodeliste 4.9: CSS som styler navigasjonsmenyen.

```
.topBar{
    height: 100px;
    background-color: #E6E7E9;
    font-size: 16px;
}

.topBar img{
    margin-left:25px;
    margin-top: 10px;
}

.topnav #myLinks {
    z-index: 5;
    color:#231F20;
    font-family: Gill Sans, sans-serif;
    text-align: center;
    position: absolute;
    background-color: #E6E7E9;
    right:0;
    border-bottom: 2px #231F20;
    width:15%;
    height:700px;
    display: none;
    margin: 1px;
}

.topnav a.icon {
    display: block;
    color:#231F20;
    position: absolute;
    right: 0px;
    top: 0px;
    font-size: 20px;
    padding : 20px;
    margin: auto;
}

.topnav a.icon:hover {
```

```
    border-radius: 50%;
    background-color: #231F20;
    color: #E6E7E9;
}
```

4.3.2 Turplanlegger - Reisealternativer

Turplanleggeren bygger på data fra Journey Planner APIet fra Entur. Journey Planner er er API som bruker GraphQL som søkespråk. Dette gjør det mulig for å gjøre spesifikke søk slik at innholdet av responsen vi får fra APIet kun inneholder de dataene som ønskes. Strukturmessig ligner GraphQL på JSON.
Søket som vi kaller APIet med ser vi i kodeliste 4.11. Søket består av hva vi ønsker å søke etter, trip(), med parametre, og strukturen vi vil ha på responsen fra APIet. Søkeparametrene som brukeren kan søke på er;

- Koordinater til og fra : ${start/stopLngLat.lat} og ${start/stopLngLat.lng}
- Transportmetode : ${mode}.

Kodeliste 4.10: Søke objektet 'query'.

```
query {
        trip(
                from: {
                        coordinates: {
                                latitude: ${startLngLat.lat}
                                longitude: ${startLngLat.lng}
                        }
                }
                to: {
                         coordinates: {
                                        latitude: ${stopLngLat.lat}
                                        longitude: ${stopLngLat.lng}
                        }
                }
                modes:[${mode}]
                numTripPatterns: 5
                walkSpeed: 1.3
                arriveBy: false
                )
####Requested fields
        {
        tripPatterns {
                startTime
                duration
                walkDistance
                distance
                legs {
                        fromPlace{
                                name
                                longitude
                                latitude
                        }
                        toPlace{
                                name
                        }
```

```
                        mode
                        distance
                        duration
                        line{
                                authority {
                                        name
                                }
                                publicCode
                        }
                }
        }
        }
}
```

Journey Planner informasjonen hentes ut med hjelp av to funksjoner. Funksjonen queryFetch() tar hånd om selve API kallet, og tar inn to variabler, selve søkeobjektet 'query' og 'variables'. APIet kalles med Fetch() funksjonen. Metoden er POST, og headerene inneholder informasjon on kallets 'Content-Type' som er 'application/json', og identifikasjon, 'ET-Client-Name": NTNU - Miljøturguide Bacheloroppgave'. I body brukes JSON.stringify for å konverterer søket til en JSON string, og objektet som sendes til APIet er en string. Responsen fra APIet konverteres så fra et JSON objekt til et javascript objekt med .then(res => res.json(). Objektet som vi sitter igjen med returneres til doTripQuery(). DoTripQuery() er funksjonen som inneholder objektet 'query'. Denne funksjonen tar inn parameterne 'startLngLat', 'stopLngLat' og 'mode'. Start-og stopLngLat er koordinatene for start-og stoppested. Mode er transportmetoden brukeren har valgt. I funksjonen kalles queryFetch(), og 'query' sendes med som parameter. Syntaksen på 'query' er ikke valid javascript, og sendes derfor til APIet som en string. Funksjonskallet blir da queryFetch({'query'}).
QueryFetch() returnerer et objektet som inneholder reiseinformasjonen vi trenger. Vi kan da i doTripQuery() gå gjennom returnerte objektet med '.then(data =>'. Her flyttes informasjonen fra det returnerte objektet over til en array kalt tripObject[ ], som blir inneholdende hvert reisealternativ brukeren av applikasjonen kan velge i mellom.

Kodeliste 4.11: Kall til Journey Planner API.

```
function doTripQuery(startLngLat, stopLngLat,mode){
    queryFetch( '...')
        .then(data => {     //  Doing work on the data
            console.log(data.data);
//  Moving the fetched data, tripPatterns, into an object
            for (var i = 0; i < data.data.trip.tripPatterns.length; i++) {
                tripObject[i] = data.data.trip.tripPatterns[i]
            }
//  Do the HTML stuff into its own function,
//      to keep things clean up here
            doHTMLStuffOnTripObject(tripObject,mode)
        })
}

function queryFetch(query, variables){
   return fetch('https://api.entur.io/journey-planner/v2/graphql/', {
```

```
        method: 'POST',
        headers: {"Content-Type": "application/json",
                "ET-Client-Name" : "NTNU - Miljøturguide Bacheloroppgave"},
        body: JSON.stringify({
            query: query,
            variables : variables
        })
    }).then(res => res.json())
}
```

Deretter kaller doTripQuery() på doHTMLStuffOnTripObject(tripObject, mode), som er funksjonen som bygger opp HTML elementene som skal vise informasjonen fra Journey Planner APIet. Denne funksjonen tar inn alle reiseobjektene tripObject, og transportmetoden mode.
Kodeliste 4.12 viser hvordan HTML elementene konstrueres. For hvert reisealternativ tripObject lages det et HTML element som kalles patterns med 'patterns[e] = document.createElement('div')'. Patterns er av HTML typen 'div' og har taggen ' '. Den settes til å tilhører klassen tripPatterns med 'patterns[e].className = 'tripPatterns"; og inneholder to HTML elementer, begge også 'div' elementer; header og trip. Trip skjules fra visning med 'trip[e].style.display = none'"; (Kodeliste 4.12).

Kodeliste 4.12: Bygging av HTML elementer som skal vise reisealternativer.

```
function doHTMLStuffOnTripObject(tripObject, mode){
    var patterns = [tripObject.length];
    var trip = [tripObject.length];
var header;
var choiceButton;

//  Going through each 'tripPattern' in journeyObjects
    for (var i = 0; i < tripObject.length; i++) {
        (function (e) {
        //  Creating html element to display a journeyObject in, called option
            patterns[e] = document.createElement('div');
            patterns[e].class = "collapsible";

        //  Making patterns element part of class  tripPatterns
            patterns[e].className = 'tripPatterns';
        //  Setting element id
            patterns[e].id = "tripPatterns" + e;

                        header = document.createElement('div');

            trip[e] = document.createElement('div');
            trip[e].style.display = "none";

            choiceButton = document.createElement('button');
            choiceButton.textContent = "Choose";
                        ...
        header.innerHTML = '...';

        //  Separate function to create the tab to show the legs of a trip pattern
        doHTMLStuffOnTripLegs(coord, tripObject[e].legs, trip[e], header, e);

                        ...
                }
```

```
        }
}
```

Header elementet inneholder hovedinformasjonen om reisen; dato, klokkeslett, distanse, reisetid og utslippstall. Header elementet fylles med reiseinformasjon ved å sette header.innerHTML lik HTMLinnholdet vi vil ha i elementet(Kodeliste 4.13).

Kodeliste 4.13: Bygging av HTML elementer som skal vise reisealternativer.

```
 header.innerHTML = ‘
            ${startTime}<br>
            ${timeToEnd} <br>

           ${timeOfTravel}<br>
            ${distance + "km"}<br>
            ‘;
```

Trip elementet inneholder informasjon om etappene et reisealternativet består av. Innholdet i trip konstrueres i en egen funksjon, doHTMLStuffOnTripLegs()(Kodeliste 4.14). For hver etappe reisen inneholder lages et HTML element legs med ’const legs = document.createElement(p")’;. Legs er et avsnitts element (’p’ for paragraph), og har HTML tag ’<p>’. Legs sin innerHTML settes til å være tom, ’legs.innerHTML = ’, før innhold festes til elementet med ’legs.append(‘...‘)’. Append() fester nytt innhold bakerst på elementet. Innholdet som festes er informasjon om hvor etappen starter og slutter, transportmetode og reiseavstand. Til slutt festes legs elementet til trip, og for-loopen går videre til neste etappe.

Kodeliste 4.14: Bygging av HTML elementer som skal vise etappene for hvert reisealternativ.

```
function doHTMLStuffOnTripLegs(coordinates, tripLegs, trip, header, e){
    var co2 = 0;
    var nox = 0;

    for(var i = 0; i < tripLegs.length; i++) {
        const legs = document.createElement("p");
        legs.class = "legs";
        legs.id = ’legs’ + i;
        legs.innerHTML = " ";

        ...

        legs.append(‘
            ${tripLegs[i].fromPlace.name + " - " + tripLegs[i].toPlace.name}
            ${tripLegs[i].mode}
            ${tripLegs[i].distance / 1000 + "km " + tripLegs[i].duration}
            ‘);
        coordinates.push([tripLegs[i].fromPlace.longitude,
                tripLegs[i].fromPlace.latitude]);

        trip.append(legs)
    }
    ...
}
```

Header og trip festes så til patterns med .append() funksjonen(Kodeliste 4.15). I tillegg festes knappen choiceButton til trip. Det er denne knappen brukeren trykker på for å bekrefte reisen.

Figur 4.1: Strukturen til hvert patterns element

Kodeliste 4.15: .append().

```
patterns[e].append(header);
trip[e].append(choiceButton);
patterns[e].append(trip[e]);
tripOptions.append(patterns[e]);
```

'Header.addEventListener("click", function () ...)(Kodeliste 4.16);' legger til en funksjonalitet på header elementet i patterns. Når elementet trykkes på så settes trip sin style.display til å være "block" som gjør at 'trip' elementet blir synlig og patterns utvides og gir plass til 'trip' innholdet. I tillegg lagres også informasjon om reisealternativet når header elementet trykkes på i localStorage. Skulle brukeren ønske å gå videre med reisealternativer de har valgt på, vil informasjonen i localStorage være tilgjengelig for bekreftelsessiden når brukeren overføres dit etter at de trykker på bekreft knappen.

Kodeliste 4.16: header.addEventListener()

```
// Make the buttons expand when clicked, showing additional information
header.addEventListener("click", function () {
                var index = "none";
                this.classList.toggle("active");
                if (trip[e].style.display === "block") {
                    trip[e].style.display = "none";
                } else {
                    trip[e].style.display = "block";
                    index = e;
                    mapDirections(coord);
                    console.log(tripObject[e].legs);
                    choiceButton.onclick = function () {
                        location.href =
                        "https://localhost:8443/confirmation?legNumber="
                        + tripObject[e].legs.length;
```

```
                    localStorage.setItem("startPlace", testStart.selectedName);
                    localStorage.setItem("stopPlace", testStop.selectedName);
                    localStorage.setItem("startTime", startTime);
                    localStorage.setItem("timeToEnd", timeToEnd);
                    localStorage.setItem("timeOfTravel", timeOfTravel);
                    localStorage.setItem("distance", distance);
                    var co2 = localStorage.getItem("co2"+e);
                    var nox = localStorage.getItem("nox"+e);
                    localStorage.setItem("co2", co2);
                    localStorage.setItem("nox", nox);

                    localStorage.setItem(
                    "leg", JSON.stringify(tripObject[e].legs)
                    )

                }
            }
            for(var i = 0; i < trip.length; i++) {
                if(i !== index) {
                    trip[i].style.display = "none";
                }
            }
        });
```

4.3.3 Turplanlegger - Utslippstall

Utslippsverdier for hver transportmetode er lagret i databasen, og hentes ut når turplanleggeren lastes, slik at dataene er tilgjengelige i nåværende session når utslippstallene skal regnes ut. Utslippsverdiene lastes i HTML fra databasen.

Kodeliste 4.17: HTML som henter utslippsverdier fra Database

```
<div id = "types" th:object="${transport}" hidden>
    <div class="eachType" th:each="type␣:␣${transport}">
        <li th:id="${type.name}" th:text="${type.name}">
        <li th:class="${type.name}" th:id="${type.name}␣+␣Co2" th:text="${type.co2}">
        <li th:class="${type.name}" th:id="${type.name}␣+␣Nox" th:text="${type.nox}">
    </div>
</div>
```

Javascript henter så tallene fra HTML, som vist i kodeliste 4.18. Utslippstallene for en etappe regnes ut ved å gange utslippstallen med antall kilometer etappen er lang.

Kodeliste 4.18: Javascript som henter utslippsverdier fra HTML

```
var car = {
    'name' : 'car',
    'co2' : document.getElementById("CarCo2").innerText,
    'nox' : document.getElementById("CarNox").innerText
};
```

Vår implementasjon tar ikke i bruk faktiske utslippstall, men siden implementasjonen er på plass trengs det bare å bytte ut verdiene som er i databasen nå med mer nøyaktige tall.

4.3.4 Turplanlegger - Kart

Vi benytter oss av Mapbox som kartløsning. Kartet gjør det mulig for oss å grafisk visualisere reisen. Mapbox legges til i HTML.

Kodeliste 4.19: Legge til Mapbox kart i HTML

```
<script src="https://api.mapbox.com/mapbox-gl-js/v1.8.1/mapbox-gl.js"></script>
<link href="https://api.mapbox.com/mapbox-gl-js/v1.8.1/mapbox-gl.css"
rel="stylesheet" />
<script src="https://api.mapbox.com/mapbox-gl-js/
plugins/mapbox-gl-directions/v4.0.2/mapbox-gl-directions.js"></script>

<div id="map"></div>
```

Kartet defineres i javascript med 'var map = new mapboxgl.Map(...)'. Funksjonen tar inn parametre som lar oss definere hvilket HTML element som skal vise kartet, stylingen på kartet, zoom nivå og koordinater kartet skal være sentrert på.
Vi må også identifisere oss med en 'access token'.

Kodeliste 4.20: Definisjon av kart og kartmarkører i javascript

```
var startMarker = new mapboxgl.Marker();
var stopMarker = new mapboxgl.Marker();

mapboxgl.accessToken =
"pk.eyJ1Ijoiam9uYW1lIiwiYSI6ImNrN3Q2ZGNwbDB
1d3YzZnBsYTR5dWNscWkifQ.3S1bd10tLNgLDIH79qBEBg";
var map = new mapboxgl.Map({
    container: 'map',
    style: 'mapbox://styles/mapbox/streets-v11',
    center: [10.757933, 59.911491],
    zoom: 5
});
```

4.3.5 Turplanlegger - Søkefelt

Vi bruker HTML <form> for å implementere søkefeltene. Det er fire felter, to som tar tekst som input (<input type=text»), en som tar dato som input (input type=date") og en som tar tid som input (input type=time"). Implementasjon vises i Kodeliste 4.21.

Kodeliste 4.21: HTML <form>

```
<form class="example" action="action_page.php">
        <input type="text"  id="startLocation" onkeyup="startFunction()"
                placeholder="From" name="search" autocomplete="off">

        <input type="text"  id="stopLocation" onkeyup="stopFunction()"
                placeholder="To" name="search" autocomplete="off">
    </form>

    <form class="timeAndDate" action="action_phage.php">
        <input type="date" id="travelDate">
        <input type="time" id="departureTime">
```

```
    </form>
```

HTML har innebygd funksjonalitet for input type ”date” og ”time”.
I søkeboksene for start-og stoppested har vi implementert autocomplete, som gir brukeren forslag om reisested hver gang brukeren skriver inn en ny bokstav i søkefeltet. Vi bruker Geocoder APIet for å opp reiseforslag, som gjør det mulig å gjøre søk etter reisedestinasjoner innad i Norge.
Funksjonen startFunction() kalles ’onkeyup’, altså når en tast trykkes på. Det gjøres et GET kall til "https://api.entur.io/geocoder/v1/autocomplete?text=’ + input.value + ’lang=en", hvor ’input.value’ er brukerens søk. GET kallet returnerer et JSON objekt som respons. Vi looper over responsen, henter ut informasjon om stedsnavn og koordinater, og flytter denne informasjonen over i et midlertidig objekt. Dette midlertidige objektet pushes så over i arrayen arr[], som vi bruker som kilde for autocompleten. Vi bruker .autocomplete() funksjonen fra jQuery i vår implementasjon. Koden $("#startLocation").autocomplete legger til autocomplete på HTML elementet med id=startLocation. Når et element fra listen velges kalles en funksjon med select : function(event, ui). Lengde-og breddegraden til stedet som velges lagres, slik at det kan sendes med som parametre til ’query’ objektet når Journey Planner APIet skal kalles.

Kodeliste 4.22: Autocomplete implementasjon

```
$("#startLocation").autocomplete({
        source: arr,
        select: function (event, ui) {
            startMarker.setLngLat([ui.item.coordinates.long,
                        ui.item.coordinates.lat])
            // startMarker.addTo(map);
            testStart.selectedName = ui.item.value;
            testStart.selectedLat = ui.item.coordinates.lat;
            testStart.selectedLong = ui.item.coordinates.long;
        }
    });
```

4.3.6 Reisestatistikk

Siden databasen ikke ble fullt implementert bruker statistikksiden et hardkodet objekt ’tripData’ som erstattning. Dette gjorde det mulig for oss å implemtere siden med ønsket funksjonalitet.
Statistikksiden består av en liste med tidligere reiser og en graf som viser utslippsdata for hver etappe en reise inneholder. Listeelementene her bygges opp likt som reisealternativene i turplanleggeren. For hver reise en bruker har reist lages det et HTML element ’trip’ som inneholder reisens dato og start-og stoppested for reisen. Hvert ’trip’ element har også en ’eventListener’ knyttet til seg. Når en reise(’trip’) fra listen trykkes på kalles funksjonen ’displayTripData()’ som oppdaterer grafen med data om reisen(Kodeliste 4.23).

Kodeliste 4.23: eventListener som kaller displayTripData() når reise trykkes på.

```
trip.addEventListener("click", function() {displayTripData(trip)});
```

Vi benytter oss av Chart.js biblioteket for å kunne tegne grafer. Bilblioteket legges til ved å linke til et script i HTML. Biblioteket gir tilgang til <canvas> taggen som definerer en graf i HTML(Kodeliste 4.24).

Kodeliste 4.24: Legge til og definere graf element i HTML

```
<script src="https://cdn.jsdelivr.net/npm/chart.js@2.9.3/dist/Chart.min.js">
</script>

<div id = "content">
    <canvas id="myChart"></canvas>
</div>
```

Grafen inneholder parametre fra reiseobjektet 'tripData'. Chart.js gir oss muligheten til å sette arrayer med informasjon som datapunkter. Chart.js tegner grafen ut i fra disse punktene. Fra reiseobjektet henter vi ut; start-og stoppested og utslippstall. Disse dataene lagres i arrayene 'dataLabelArray' og 'dataArray' . Når en reise trykkes på kalles displayTripData(). Funksjonen tømmer først grafens innhold. Deretter kaller den funksjonen getDataSets() som for hver etappe fyller arrayene 'dataLabelArray' og 'dataArray' med henholdsvis start-og stoppested og utslippstall. Så oppdateres grafen. Data arrayene nullstilles så slik at de er tomme når en ny reise skal vises.

Kodeliste 4.25: Implementering av graf-tegning

```
function displayTripData(trip){
    myChart.reset();
    getDataSets(trip.num);
    myChart.update();

    dataArray.length = 0;
    dataLabelArray.length = 0;
}
function getDataSets(i){

    tripData[i].legs.forEach(function(legs){
        dataArray.push(legs.co2);
        dataLabelArray.push(legs.from + " - " + legs.to);
    })
};
```

Selve grafen implementeres lett med å lage et nytt graf objekt med 'var myChart = new Chart()'. Når objektet initialiseres tar den inn parametre som forteller hvilket HTML objekt grafen tegnes i, og hvordan man vil at grafen skal struktureres.

Kodeliste 4.26: Implementering av graf

```
var ctx = document.getElementById('myChart').getContext('2d');
var myChart = new Chart(ctx, {      //  Chart to display information
    type: 'line',
    data: {
        labels: dataLabelArray,
```

```
        datasets: [{
            data: dataArray,
        }]
    },
    options: {
        responsive: false,
        layout: {
            padding: {
                left: 50,
                right: 0,
                top: 0,
                bottom: 0
            }
        },
        scales: {
            xAxes:[{
                ticks: {
                    beginAtZero: false
                }
            }],
            yAxes: [{
                ticks: {
                    beginAtZero: true
                }
            }]
        }
    }});
```

4.3.7 Reisebekreftelse

Informasjon om reisealternativet brukeren har valgt lagres i localStorage. Kodeliste 4.27 viser hvordan bekreftelsessiden henter denne informasjonen og lagrer den i lokale variable, før informasjonen settes som verdien(.value) til de respektive HTML elementene som viser informasjonen.

Kodeliste 4.27: StartTime fra localStorage til HTML

```
//      I journeyPlanner.js
...
localStorage.setItem("startTime", startTime);
...

//      I confirmation.js
...
var startTime = localStorage.getItem('startTime');
...

...
document.getElementById('startDateField').value = startTime;
...
```

Verdiene til HTML elementene er thymeleaf verdier. HTML'en som fylles er en del av et innsendingskjema <form> som fylles med verdiene fra localStorage, og kan ikke endres på av brukeren. (kodeliste 4.28). Når brukeren trykker på 'bekreft' knappen for å bekrefte reisen vil "RequestMapping-annotation" med

verdi /confirmation og metoden ”RequestMethod.Post” kalle funksjonen confirmationPost() som lagrer reisen i databasen(kodeliste 4.29).

Kodeliste 4.28: HTML skjemaet med startTime thymeleaf verdier

```
<div id="confirmSchema">
    <form autocomplete="off" action="#" th:action="@{/confirmation}"
        th:object="${trip}" method="post" role="form">

                ...

        <p id="startDateLabel"><b>Dato</b></p>
        <div id="startDate" class="form-group">
            <br>
            <input type="text" id="startDateField"
                        th:field="*{tripDate}" class="form-control" readonly>
        </div>

                ...

    </form>
</div>
```

Kodeliste 4.29: Post mapping metode for /confirmation

```
@RequestMapping(value="/confirmation", method=RequestMethod.POST)
    public ModelAndView confirmationPost(@Valid Trip trip) {
        ModelAndView modelAndView = new ModelAndView();
        tripService.saveTrip(trip);
        modelAndView.setViewName("home");
        return modelAndView;
    }
```

Kapittel 5

Avslutning

5.1 Drøftinger

I dette kapittelet vil vi diskutere hvilke utfordringer vi har stått ovenfor underveis i prosjektet, våre resultater og hva som kunne vært gjort annerledes. Målet for prosjektet har vært å opparbeide seg ny kunnskap om prosjektarbeid og ulike teknologier vi tidligere ikke hadde kunnskap til.

5.1.1 Resultater

Den endelige løsningen vår har mangler i forhold til resultatmålene vi satt opp i kapittel 1 og 2. Vår løsning kunne også vært fobedret på mange områder. Vi har ikke hatt tid til å implementere muligheten for samkjøring, fakturering, ei heller ikke muligheten for å se sine tidligere reiser og miljøavtrykk. Grunnen til dette er mangel på tid. Vi brukte for mye tid på deler av utviklingen og havnet på etterskudd i forhold til Gantt-diagrammet vi utarbeidet i fremdriftsplanen. Dette gjorde at vi måtte gå videre med deler av utviklingen og heller se om det fantes tid til forbedring ved senere tidspunkt.
Muligheten for å se sine tidligere reiser og miljøavtrykk mangler i hovedsak kun å hente informasjonen fra databasen. Dette ble ikke gjort da vi ventet på hjelp for å få implementert siste del av databasen. Denne delen kom vi aldri i mål med, så dataen som presenteres på tidligere reiser og miljøavtrykk er kun hard-kodede variabler.
Ved reisebekreftelse vil reisen lagres i database, funksjonalitet for fakturering er ikke implementert. Vi hadde et alternativ til å nedjustere dette punktet fra å opprette en fakturering til å istedet gi brukeren linker til nettsider hvor reisen kunne bestilles, men dette ble heller ikke implementert.
Et mindre punkt som ikke var en del av resultatmålene satt opp i kapittel 1 og 2, men som var en del av vårt design var oppdatering av profildetaljer. Selve profilsiden ble implementert, men mangler funksjonalitet til å oppdatere brukerprofilen.
Ting som kan forbedres er visning av ruter og tidspunkt for reiser. Ved visning av

ruter fikk vi ikke tak i nok koordinater ved bruk av Enturs API ”Journey Planner”. Dette fører til at rutene ikke vises nøyaktig. Tidspunktene for reisene er feil. Denne blir alltid satt til nåværende tidspunkt, og ikke de faktiske tidspunktene for reisene.

5.1.2 Alternativer

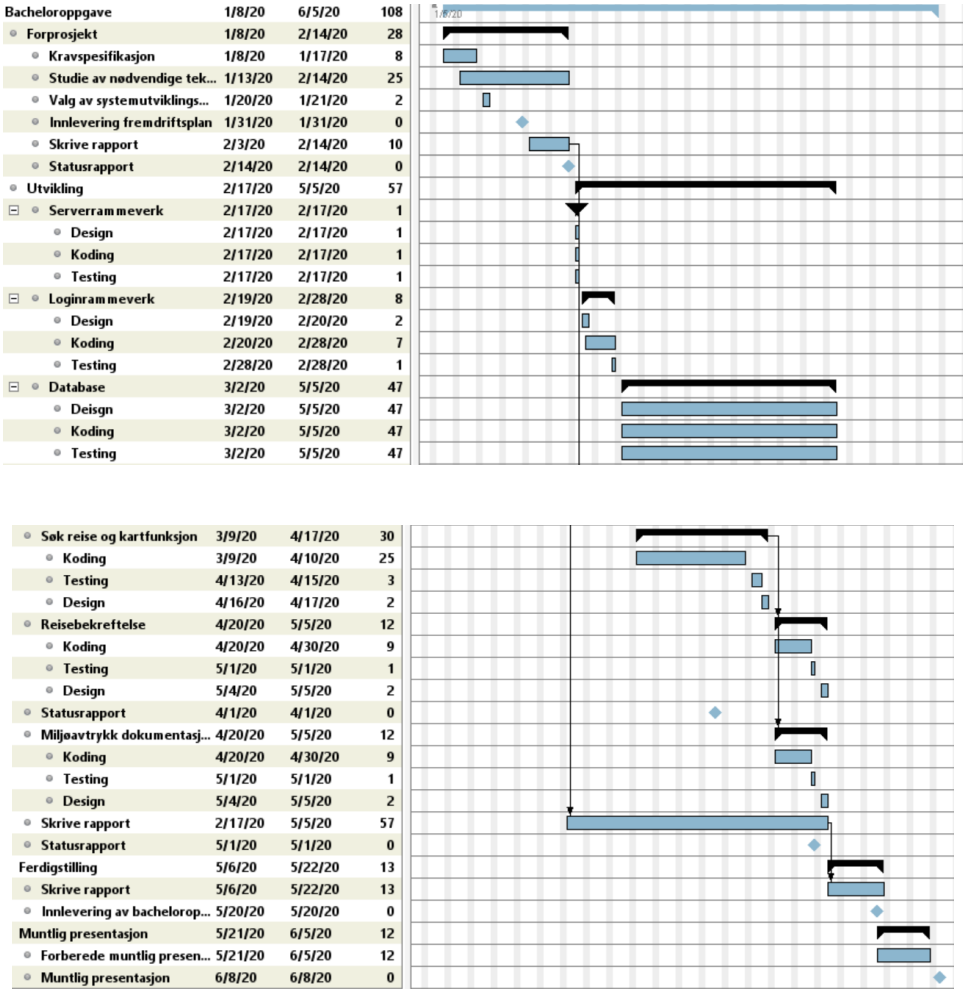
Underveis i prosjektperioden har vi stått ovenfor flere valg og alternativer.
Vi bestemte oss for å følge den inkrementelle systemutviklingsmodellen ”staged delivery model” i prosjektet. Her oppdaget vi tidlig i utviklingen at dette kan ha vært et dårlig valg, og at vi heller burde valgt en smidigere modell. Den inkrementelle modellen virket som et gunstig alternativ tidlig, da vi hadde kravspesifikasjonen tidlig på plass, og anså det som liten risiko for store endringer i denne. Men grunnet mangel på erfaring har vi møtt på utfordringer i ulike deler av utviklingsprosessen. Istedenfor å få fullført hvert inkrement kronologisk, ble vi nødt til å gå videre til andre deler av utviklingen for så å gå tilbake ved et senere tidspunkt. Databaseimplementasjonen var satt opp som et tidlig inkrement i utviklingsplanen. Deler av denne implementasjonen måtte utsettes da vi var usikre på hvilken informasjon vi faktisk fikk tak i via våre API-kall og hva vi trengte å lagre i databasen.
Designet fikk vi beskjed av oppdragsgiver om at en av deres ansatte ville lage for oss. Grunnet sykdom hos den ansvarlige for designet måtte vi også gå videre i utviklingen og rokkere på rekkefølgen av implementasjonen.
Tidlig i prosjektperioden bestemte vi oss for å benytte open-source rammeverk for login-systemet og serveren. Vi vurderte å benytte oss av ”Apache Shiro” for login. Dette alternativet gikk vi bort fra da det var veldig begrenset informasjon og dokumentasjon om hvordan dette skulle implementeres ved bruk av Tomcat som server. Da vi letet etter andre alternativer fant vi Spring-rammeverket som ga oss all funksjonalitet vi var ute etter, samt hadde Tomcat-serveren ferdig implementert i rammeverket.

5.2 Kritikk av oppgaven og eget arbeid

I ettertid av oppgaven ser vi at vi burde bedt om hjelp tidligere da vi sto fast på ulike deler av utviklingen. Dette kunne gjort at vi ble ferdig med deler av systemet tidligere, så vi hadde hatt mer tid til utviklingen av delene vi ikke fikk fullført.
En årsak som gjorde alt litt mer problematisk var covid-19 viruset. Den spesielle situasjonen som oppsto i forbindelse med viruset gjorde at det ble lite kommunkasjon med oppdragsgiver en periode. Dette resulterte i mindre veiledning enn vi mottok tidligere i prosjektperioden, som kanskje gjorde at vi brukte for lang tid på de ulike delene av utviklingen.
En mangel ved løsningen vår er organisert testing og verifisering av programmvaren. Vi har åpenbart testet ut funksjonaliteten mens vi har jobbet,

og det vi har pushet til git repositoriet våres har vært fungerende kode. Likevel kunne dette vært utvidet til organisert testing, slik at løsningen blir mer robust og gjør vedlikeholdet i produksjon lettere. Dette gjelder både for verifisering av koden og testing av sikkerheten til løsingen. Mangel på feilhåndtering i løsningen er også verdt å nevne som et punkt som bør tas hånd om ved videre arbeid.

5.2.1 Gantt-diagram



Bacheloroppgave	1/8/20	6/5/20	108
Forprosjekt	1/8/20	2/14/20	28
Kravspesifikasjon	1/8/20	1/17/20	8
Studie av nødvendige tek...	1/13/20	2/14/20	25
Valg av systemutviklings...	1/20/20	1/21/20	2
Innlevering fremdriftsplan	1/31/20	1/31/20	0
Skrive rapport	2/3/20	2/14/20	10
Statusrapport	2/14/20	2/14/20	0
Utvikling	2/17/20	5/5/20	57
Serverrammeverk	2/17/20	2/17/20	1
Design	2/17/20	2/17/20	1
Koding	2/17/20	2/17/20	1
Testing	2/17/20	2/17/20	1
Loginrammeverk	2/19/20	2/28/20	8
Design	2/19/20	2/20/20	2
Koding	2/20/20	2/28/20	7
Testing	2/28/20	2/28/20	1
Database	3/2/20	5/5/20	47
Deisgn	3/2/20	5/5/20	47
Koding	3/2/20	5/5/20	47
Testing	3/2/20	5/5/20	47
Søk reise og kartfunksjon	3/9/20	4/17/20	30
Koding	3/9/20	4/10/20	25
Testing	4/13/20	4/15/20	3
Design	4/16/20	4/17/20	2
Reisebekreftelse	4/20/20	5/5/20	12
Koding	4/20/20	4/30/20	9
Testing	5/1/20	5/1/20	1
Design	5/4/20	5/5/20	2
Statusrapport	4/1/20	4/1/20	0
Miljøavtrykk dokumentasj...	4/20/20	5/5/20	12
Koding	4/20/20	4/30/20	9
Testing	5/1/20	5/1/20	1
Design	5/4/20	5/5/20	2
Skrive rapport	2/17/20	5/5/20	57
Statusrapport	5/1/20	5/1/20	0
Ferdigstilling	5/6/20	5/22/20	13
Skrive rapport	5/6/20	5/22/20	13
Innlevering av bacheloro...	5/20/20	5/20/20	0
Muntlig presentasjon	5/21/20	6/5/20	12
Forberede muntlig presen...	5/21/20	6/5/20	12
Muntlig presentasjon	6/8/20	6/8/20	0

Figur 5.1: Det faktiske gantt-diagrammet

Figur 5.1 viser det faktiske gantt-diagrammet med tidsbruken for de ulike delene. Dette diagrammet er som forventet annerledes enn gantt-diagrammet som ble utarbeidet i planleggingsfasen, se vedlegg B. Som nevnt tidligere i rapporten måtte vi rokkere om på implementasjonen av ulike deler. Vi har også arbeidet med noen av inkrementene parallelt, noe en ikke skal gjøre ved bruk av den inkrementelle modellen ”staged delivery model”.

5.3 Videre arbeid

For at løsningen skal kunne tas i bruk er det en del som bør endres på. Løsningen vår bør forbedres på de punktene nevnt tidligere. Arbeidet vi ikke rakk bør også implementeres. Det bør lages en løsning for betaling når en reise

er bekreftet. Statistikksiden bør hente informasjonen som er lagret i databasen. Videre bør også organisasjon og avdeling som brukeren har en tilknytning til implementeres.
Ettersom løsningen vår er laget som en enkeltstående modul, må løsningen tilpasses ETC sine systemer før det kan tas i bruk.

5.4 Evaluering av gruppas arbeid

5.4.1 Innledning

Dette var første gang vi arbeidet sammen på gruppe. Vi kjente ikke hverandre fra tidligere selv om vi begge går i samme klasse. Vi har som gruppe ikke hatt noen spesielle uenigheter underveis i prosjektet.
Totalt sett har det under prosjektarbeidet blitt brukt mer tid på utviklingen enn rapportskriving. I ettertid har vi innsett at vi burde ha brukt mer tid på rapportskrivingen tidligere.

5.4.2 Organisering

Prosjektorganiseringen har vi ikke gjort endringer på ut ifra hva vi skrev i fremdriftsplanen, vedlegg A. Ettersom vi kun har vært to på gruppa så vi ikke behovet for å ha noen spesifiserte roller, men i løpet av prosjektperioden har det automatisk oppstått behov for en ”leder” som har tatt ansvar for å planlegge møter, hva som skal gjøres og fordele oppgaver. Vi har også latt en person ha ansvaret for all kommunikasjonen med oppdragsgiver og veileder.
De første månedene av prosjektperioden var det faste rutiner for møter med oppdragsgiver og veileder, samt faste arbeidsdager på skolen. Grunnet en svært spesiell situasjon i forbindelse med covid-19 viruset måtte disse faste rutinene forkastes. Istedenfor faste rutiner for møter og arbeidsdager ble dette byttet ut med arbeid hjemmefra. Dette skapte noen uforutsette utfordringer.

5.4.3 Fordeling av arbeidet

Fordelingen av arbeidet har gått greit. Jonas har hatt ansvar for backend-programmering, mens Andreas har hatt ansvaret for design-implementasjonen. Videre har begge hatt ansvar for frontend-programmering utenom design. Dette ble en naturlig måte å fordele arbeidet på, da det ville kreve ekstra tid for en person å måtte sette seg inn i alle deler av arbeidet. Dette vil si at gruppemedlemmene har ulik kunnskapsnivå innenfor ulike deler av systemet. Vi planla å benytte oss av Trello som prosjektorganiseringsverktøy, men var ikke flinke til å benytte oss av denne under utviklingen.

5.4.4 Prosjekt som arbeidsform

Vi har tidligere vært innom prosjektarbeid i løpet av studietiden. Allikevel var det en ny erfaring med et så omfattende prosjekt som dette. Ved prosjektstart følte vi oss ikke klare for et så stort prosjekt. I ettertid av prosjektet ser vi at prosessen har vært veldig lærerikt.
Prosjektprosessen har gitt oss erfaring om hvordan det er å jobbe sammen i et større prosjekt, hvor en må forholde seg til en ekstern oppdragsgiver. Denne erfaringen er noe som vil komme godt med i arbeidslivet.

5.5 Konklusjon

Selv om løsningen vår ikke er komplett og har mangler er vi delvis fornøyde med prosjektet. Prosjektet har vært utfordrende, men ekstremt givende læringsmessig. Kunnskap vi har oppnådd gjennom studietiden ved NTNU i Gjøvik har vært av stort utbytte i prosjektoppgaven.
Da vi startet i Januar var det veldig lite kunnskap om frontend-programmering eller bruk av eksisterende rammeverk. Vi har i løpet av prosjektperioden lært mye om Javascript, CSS og HTML-programmering, samt det å benytte seg av eksisterende rammeverk for utviklingen av et produkt. Vi har også fått utvidet vår kunnskap om backend-programmering.

Referanser

[1] *Spring Security*, https://spring.io/projects/spring-security, [Online; accessed 29-April-2020].

[2] *HTTP*, https://snl.no/HTTP, [Online; accessed 29-April-2020].

[3] *HTTPS*, https://snl.no/HTTPS, [Online; accessed 29-April-2020].

[4] *HTML*, https://snl.no/HTML, [Online; accessed 19-May-2020].

[5] *CSS*, https://snl.no/CSS, [Online; accessed 19-May-2020].

[6] *Spring data JPA*, https://spring.io/projects/spring-data-jpa, [Online; accessed 20-May-2020].

[7] *Spring Boot*, https://spring.io/projects/spring-boot, [Online; accessed 20-May-2020].

[8] *Thymeleaf*, https://www.thymeleaf.org/, [Online; accessed 20-May-2020].

[9] *Spring Web MVC*, https://docs.spring.io/spring/docs/current/spring-framework-reference/web.html, [Online; accessed 20-May-2020].

[10] *Oppsummering av vilkårene i Norsk lisens for offentlige data (NLOD)*, https://data.norge.no/nlod/no, [Online; accessed 29-April-2020].

Vedlegg A

Fremdriftsplan

Fremdriftsplan, Bacheloroppgave

Jonas Melby, Andreas Danielsen

Januar 2020

Contents

1 Mål og rammer

1.1 Bakgrunn

Vi har fått oppgave fra ETC, Electric Time Car AS, å utvikle en Miljøturguide. Dette er en webapplikasjon hvor brukere kan reservere reiser med både buss, tog, fly, etc. for en valgt dato og tidspunkt. Brukeren vil da få opp de ulike alternativene med informasjon om selve reisen og miljøeffekten av valget.

Denne fremdriftsplanen vil være et hjelpemiddel for oss under arbeidet med Bacheloroppgaven, spesielt med tanke på at dette er en ny erfaring for oss begge. Å ha en fremdriftsplan vil hjelpe oss med å overholde tidsfrister, beholde oversikt og nå våre mål.

1.2 Prosjektmål

1.2.1 Effektmål

Denne oppgaven vil tjene ETC på ulike måter:

- De mottar gratis arbeidskraft for å utføre en oppgave etter deres ønsker.
- De tjener på det tidsmessig og ressursmessig ved å ikke måtte utvikle produktet selv.
- De vil få et produkt som eventuelt kan implementeres i deres allerede eksisterende løsning Car Admin.

1.2.2 Resultatmål

ETC vil ved prosjektets slutt få levert en webapplikasjon som:

- Lar brukere logge inn for å reservere reiser.
- Gjør det mulig for brukeren å se og velge mellom ulike transportmåter/samkjøring, samt tidsbruk for de ulike alternativene.
- Lar brukeren overvåke sitt miljøavtrykk for de reserverte reisene.

1.2.3 Læringsmål

Etter prosjektarbeidet forventes det at vi har lært om/utvidet vår kunnskap om:

- Java/ HTML / Ajax / SQL
- Database - MariaDB
- Server side programmering
- Web/Klientprogrammering

1.3 Rammer

1.3.1 Praktiske rammer

Arbeidet med bacheloroppgaven vil vare fra semesterstart den 8. januar 2020, frem til presentasjonene av Bacheloroppgavene 8, 9, og 10. juni.

Vi vil under prosjektperioden arbeide sammen på skolen, men også individuelt på både skolen og hjemme.

1.3.2 Tekniske rammer

Begge benytter seg av Windows 10 som operativsystem. Utviklingsmiljøet som vi kommer å benytte oss av vil være IntelliJ. IntelliJ har vi erfaring med fra tidligere, og anvedelsen av denne var også et ønske fra oppdragsgiver.

Fra oppdragsgiver settes det krav om at webapplikasjonen skal utvikles i språkene Java/HTML/Ajax/SQL, og at vi skal bruke MariaDB som databasesystem. Oppdragsgiver har også foreslått at vi bruker Apache Tomcat som server-rammeverk.

1.3.3 Føringer

Vi har satt som regel at hver person skal minst arbeide 30 timer hver uke, men at dette må kunne justeres fortløpende dersom det er behov for det for å nå våre mål.

2 Omfang

2.1 Fagområde/Problemområde

Global oppvarming er et aktuelt tema for tiden. En ser daglig konsekvenser av de jevnt økende temperaturene i atmosfæren. Isbreer smelter, havnivået øker og regnskoger er i ferd med å dø. Dette mener noen er et resultat av utslipp av klimagasser på grunn av menneskelig aktivitet[1].

2.2 Avgrensning/Problemavgrensning

En stor del av de menneskeskapte klimautslippene stammer fra transport[2], og under denne kategorien kan mange bidra til å redusere utslippene.

Det finnes ulike applikasjoner for ruteplanlegging og reservasjoner, men det finnes ingen applikasjoner som i tillegg beregner ditt miljøavtrykk basert på transportmetoden du velger.

Oppgaven fra ECT var orginalt å lage en modul som skulle implementeres i deres allerede eksisterende applikasjon CarAdmin. Denne oppgaven var i hovedsak ment for en gruppe på tre til fire studenter. Ettersom vi er to ble vi anbefalt å avgrense oppgaven av både programansvarlig på skolen og oppdragsgiver. Oppdragsgiver mente det mest krevende med den orginale oppgaven var å sette seg

inn i deres system. Avgrensningen blir derfor å utvikle wekapplikasjonen som et enkeltstående system, uten tilknyttning til den eksisterende applikasjonen.

2.3 Oppgavebeskrivelse/Problemstilling

Webapplikasjonen vi har fått i oppgave å lage er en Miljøturguide. Denne applikasjonen skal gjøre det enkelt for miljøbevisste brukere å se og velge de mest miljøvennlige transportalternativene for en reise.

Webapplikasjonen skal inneholde en reiseplanlegger hvor brukere kan planlegge sine reiser. Her skal brukerne ha mulighet til å søke etter forskjellige reiser ut ifra; ønskede steder å reise til og fra og ønsket dato og klokkeslett for reisestart og ankomst, eller kun ankomstfrist.

Ut ifra søkekriteriene vil brukeren få vist de transportalternativene som passer søkene best. For hvert transportalternativ vil det vises hvor lang tid reisen vil ta, og hvor langt man reiser. Brukeren skal kunne velge mellom de forskjellige alternativene og få se detaljert hvert steg av reisen, samt se visuelt på et kart hvor reisen går. Miljøeffekten av brukerens valg skal også vises, slik at brukeren selv ser hva som spares av miljøutslipp sammenlignet med de andre transportalternativene.

Velgbare transportmidler vil være bl.a. bil, buss, tog, ferge, fly, sykkel, eller gange. Samkjøring skal også kunne organiseres.

Etter at brukeren har valgt sin ønskede reise, videresendes brukeren så til en bekreftelsesside. Her får brukeren oppsummert hele reisen. Hvis brukeren bekrefter reisen skal alle nødvendige reisereservasjonen gjøre automatisk i bakgrunnen, og en faktura blir sendt til økonomiansvarlige i organisasjonen.

Basert på valgt transportalternativ dokumenteres spart miljøavtrykk for brukere, avdelinger og organisasjon. Disse dataene skal kunne vises i en totaloversikt for hele organisasjonen.

3 Prosjektorganisering

3.1 Ansvarsforhold og roller

Ettersom vi kun er to på gruppa kommer ingen av oss til å ha noen spesifiserte roller. Vi kommer derimot la en av oss ha hovedansvar for kommunikasjon med veileder og oppdragsgiver, slik at de kun har én person og forholde seg til når det kommer til kommunikasjon over epost.

3.2 Rutiner og regler i gruppa

- Forventet arbeidsmengde fra hvert gruppemedlem på 30 timer pr. uke.
- Faste arbeidstimer mandag og tirsdag fra klokken 10 til 16, vi legger opp til mer fleksible arbeidstimer ellers.
- Hvert gruppemedlem skriver ned egen timelogg.

- Aktiv deltagelse i arbeidet, vi forventer at hvert gruppemedlem yter sitt beste, og gjør sitt for at deres oppgaver er gjort innen avtalt tid.
- Gruppens karaktermål for hele Bacheloroppgaven er en C, men vi som ambisjoner å strekke oss opp mot en B.
- Det forventes at gruppemedlemmene tar opp problemer de har med arbeidet, gi tidlig beskjed om "henger etter", og evt. enighet om at "dette er greit"/kan taes inn igjen.
- Gruppemedlemene plikter tidlig å informere dersom de ikke er fornøyd med innsatsen til enkelte gruppemedlemme.
- Gruppemedlemene plikter å gi beskjed så tidlig som mulig dersom de ikke kan møte til avtalte tider.

Rutiner ved alvorlige brudd:

- Samtaler mellom alle gruppedeltakerne om problemet.
- Meld fra om problemene til veileder.

4 Planlegging, oppfølging og rapportering

4.1 Hovedinndeling av prosjektet

Bakgrunn for valg av systemutviklingsmodell

Valget av systemutviklingsmodell er basert på følgende punkter:

- Det er et fast tidspunkt for levering av bacheloroppgaven.
- Vi er en gruppe på kun to studenter.
- Vi har lite erfaring med større prosjekter.
- Det virker lite sannsynlig at kravspesifikasjonen kommer til å endre seg bemerkelig under prosjektet.
- Ukentlige møter med oppdragsgiver.
- Det vil være høye krav til dokumentasjon med tanke på at dette er arbeid i forbindelse med bacheloroppgave.

Systemutviklingsmodellen vi vil benytte oss av vil være en kombinasjon av den inkrementelle modellen og gjenbruksmodellen med noen aspekter fra Kanban. Typen inkrementell modell vil være "staged delivery model". I denne typen av modellen jobber en med en og en inkrement før en kan gå videre. Dette føles mest hensiktsmessig ettersom vi kun er to på gruppa.

Gjenbruksmodellen blir benyttet ettersom vi skal bruke open source rammeverk for loginsystemet og til serverdelen av prosjektet, samt at vi planlegger

benytte oss av flere tredjeparts API-er, bl.a Entur sin API for å få tilgang på informasjon om kollektivtransport i Norge.

Vi utvikler programvaren inkrementelt fordi den gir oss en ønsket fleksibilitet under utviklingsfasen. Applikasjonen vil med det utvikles gradvis og i faser, og vi vil tidlig ha en fungerende versjon av programvaren. I hver fase kan vi dermed ferdigstille design, utvikle og teste de nye funksjonalitetene. Vi vil da etter hver fase ha en ny versjon av applikasjonen. En inkrementel utvikling gir oss også muligheten til å planlegge ukentlig hva som skal måtte kodes, og vi kan enklere dele opp arbeid oss to imellom.

Vi kommer også til å bruke et Kanban-Brett under utviklings fasen for å gjøre det enklere for oss å fordele oppgaver, og organisere og planlegge videre arbeid iløpet av utviklingen.

Andre utviklingsmodeller vi vurderte

Fossefallsmodellen er en av modellene vi vurderte å benytte oss av. Etter møte med oppdragsgiver fikk vi på plass kravspesifikasjonen relativt tidlig. Det var enighet om hva som skulle gjøres fra begge parter. Vi anså det derfor som lite sannsynlig at kravspesifikasjonen ville endre seg noe nevneverdig iløpet av prosjektet. Grunnen til at vi valgte å se bort fra fossefallsmodellen var at den tilbyr lite fleksibilitet, noe som er ønskelig for oss da vi ønsker å ha fungerende versjoner av applikasjonen underveis i arbeidet.

Scrum var en av modellene vi så bort i fra relativt tidlig. Ettersom bacheloroppgaven er begrenset i en kort periode, følte vi det var unødvendig med denne typen utviklingsmodell hvor det skal utvikles prototype hver andre uke.

4.2 Plan for statusmøter og beslutningspunkter i perioden

Vi har avtalt møter med både oppdragsgiver og veileder på faste tidspunkt under prosjektperioden. Dette hjelper oss å holde riktig kurs gjennom prosjektperioden dersom problemer skulle oppstå. Innad i gruppa har vi avtalt statusmøter, der vi informerer hverandre om progresjon på individuelt arbeid, mandager klokken 10.00. Naturlige beslutningspunkter vil være etter hver endt fase.

Vi vil også ha statusmøter med veileder etter 15. Februar, 1. April og 1. Mai.

5 Organisering av kvalitetssikring

5.1 Dokumentasjon, standardbruk og kildekode

I likhet med fossefallsmodellen stiller den inkrementelle utviklingsmodellen høye krav til dokumentasjon og planlegging. Dette fordi det er nødvendig å ha en klar og full definisjon på det som skal gjøres før det kan brytes ned og bygges inkrementelt.

Alle møter med både veileder og oppdragsgiver skal dokumenteres med møtereferat. Møtereferatene legges i gruppens felles Discord-kanal.

Under utviklingsfasen vil standard javadoc benyttes for dokumentasjon av kode. Dette skal skrives på engelsk, som er standardspråket som benyttes for javadoc. All kildekode skal også skrives på engelsk. Klasser, funksjoner og variabler skal kommenteres.

5.2 Konfigurasjonsstyring

Vi skal benytte oss av Git og Bitbucket for versjonkontroll av programvaren under prosjektet. Dette gjør vi slik at vi begge kan ha tilgang til hele kildekoden og for å gjøre det lettere for oss å håndtere versjonsforskjeller i kildekode oss imellom.

Vi har også en felles Discord-kanal som vi bruker til kommunikasjon når vi ikke jobber sammen. Her deler vi notater, møtereferater og andre dokumenter som vi kommer til å bruke og jobbe med iløpet av prosjektperioden.

5.3 Risikoanalyse

Under ser vi vår vurdering av risikoanalyse og hvordan vi har rangert sannsynligheten, konsekvensen og summen av de to. Figur 1 viser skalaen som er summen av sannsynligheten og konsekvensen for en risiko.

Sannsynlighet	**<-----Risikonivå----->**			
Svært Høy	Tolerabelt	Problematisk	Kritisk	Kritisk
Høy	Tolerabelt	Tolerabelt	Problematisk	Kritisk
Nokså Lav	Liten	Tolerabelt	Tolerabelt	Problematisk
Lav	Liten	Liten	Tolerabelt	Tolerabelt
	Liten	Tolerabelt	Problematisk	Kritisk

Figure 1: Rangering av risiko.

Nr	Risiko	Sannsynlighet	Konsekvens	Sum
1	Feilestemering av tidsbruk for oppgaver	Høy	Problematisk	Problematisk
2	Langtidssykdom	Lav	Kritisk	Tolerabelt
3	Store endringer i kravspek	Nokså lav	Problematisk	Tolerabelt
4	Dårlig kommunikasjon med oppdragsgiver	Lav	Problematisk	Tolerabelt
5	Dårlig kommunikasjon med veileder	Lav	Problematisk	Tolerabelt
6	Mangel på teknisk kunnskap	Høy	Problematisk	Problematisk
7	Tap av data	Lav	Kritisk	Tolerabelt

5.4 Plan for håndtering av de viktigste risikoene

Tiltak

Vi ser på det som kun nødvendig å innføre tiltak på de risikoene som vi har rangert som problematisk eller høyere.

Nr. 1
Risikoen for feilestimering av tidsbruk for de ulike oppgavene ser vi på som en høy sannsynlighet for å inntreffe, med problematiske konsekvenser. Dette kan være konsekvenser som at vi ikke får fullført oppgaver, og i værste fall ikke har grunnlag nok for en fullstendig bacheloroppgave. For å håndtere dette har vi regelmessige møter med oppdragisgiver som kan bistå med den hjelpen vi trenger for å komme oss videre. Om det ikke skulle være nok har vi også muligheten til å begrense omfanget på noen deloppgaver.

Nr. 6
Konsekvensene av mangel på teknisk kunnskap kan være at vi bruker for mye tid på å finne ut hvordan vi skal løse oppgavene, som fører til at vi havner på etterskudd. For å redusere konsekvensen har vi allerede begynt å sette oss inn i ulike rammeverk som skal benyttes i oppgaven. Videre kan både oppdragsgiver og fagfolk som befinner seg på skolen være til hjelp om problemene blir store.

Kommentarer

Nr. 2
Langtidssykdom, sykdom over tre dager, anser vi som lav sannsynlighet for skal inntreffe. Dersom det skulle inntreffe har vi muligheten for å jobbe hjemmefra med kommunikasjon via vår Discord-kanal.

Nr. 3
Etter møter og diskusjoner med oppdragsgiver fikk vi på plass en veldefinert kravspesifikasjon relativt tidlig. Det virker som det var stor enighet om det som skulle gjøres, og at vi hadde fått på plass det meste. Ut i fra dette har vi rangert sannsynligheten for endring i kravspesifikasjonen som nokså lav.

Nr. 4
Vi har avtalt faste møter med oppdragsgiver. Så lenge vi overholder møteavtalene med oppdragsgiver og er åpne om eventuelle problemer vi måtte møte på under prosjektperioden, anser vi det som lite sannsynlig at dårlig kommunikasjon med oppdragsgiver blir er problem. I tillegg befinner oppdragsgiver seg nære skolen, som også er en faktor som reduserer sannsynligheten for dårlig kommunkasjon.

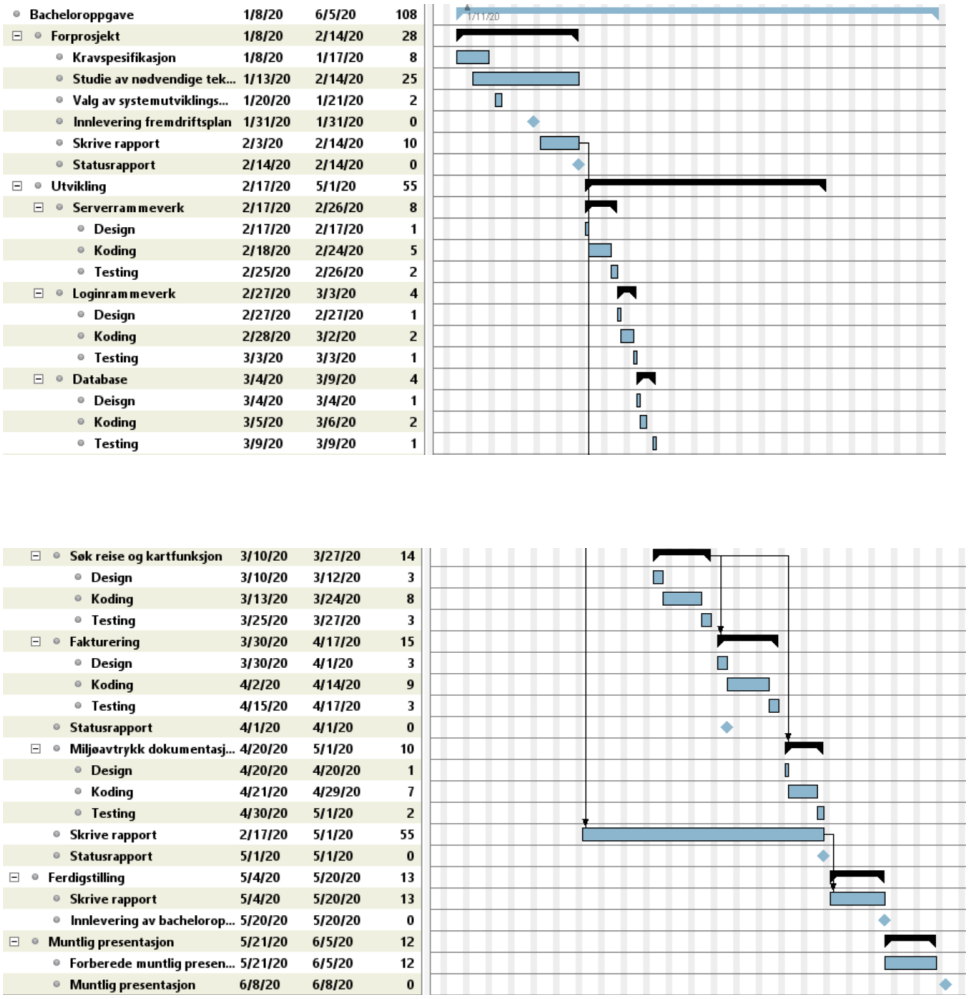
Nr. 5
Som med oppdragsgiver, har vi avtalt faste møter med veileder. Med tanke på

at veileder er stasjonert på skolen, og vi har faste møtetidspunkt er sannsynligheten for dårlig kommunkasjon lav.

Nr. 7
Tap av data ville vært kritisk om skulle skje. Ettersom vi benytter oss av Git og Bitbucket hvor vi "commiter" koden relativt ofte vil tap av data være lite sannsynlig og bli begrenset til det en hadde lokalt siden forrige "push".

6 Plan for gjennomføring

6.1 Gantt-Skjema



Bacheloroppgave	1/8/20	6/5/20	108
Forprosjekt	1/8/20	2/14/20	28
Kravspesifikasjon	1/8/20	1/17/20	8
Studie av nødvendige tek...	1/13/20	2/14/20	25
Valg av systemutviklings...	1/20/20	1/21/20	2
Innlevering fremdriftsplan	1/31/20	1/31/20	0
Skrive rapport	2/3/20	2/14/20	10
Statusrapport	2/14/20	2/14/20	0
Utvikling	2/17/20	5/1/20	55
Serverrammeverk	2/17/20	2/26/20	8
Design	2/17/20	2/17/20	1
Koding	2/18/20	2/24/20	5
Testing	2/25/20	2/26/20	2
Loginrammeverk	2/27/20	3/3/20	4
Design	2/27/20	2/27/20	1
Koding	2/28/20	3/2/20	2
Testing	3/3/20	3/3/20	1
Database	3/4/20	3/9/20	4
Deisgn	3/4/20	3/4/20	1
Koding	3/5/20	3/6/20	2
Testing	3/9/20	3/9/20	1
Søk reise og kartfunksjon	3/10/20	3/27/20	14
Design	3/10/20	3/12/20	3
Koding	3/13/20	3/24/20	8
Testing	3/25/20	3/27/20	3
Fakturering	3/30/20	4/17/20	15
Design	3/30/20	4/1/20	3
Koding	4/2/20	4/14/20	9
Testing	4/15/20	4/17/20	3
Statusrapport	4/1/20	4/1/20	0
Miljøavtrykk dokumentasj...	4/20/20	5/1/20	10
Design	4/20/20	4/20/20	1
Koding	4/21/20	4/29/20	7
Testing	4/30/20	5/1/20	2
Skrive rapport	2/17/20	5/1/20	55
Statusrapport	5/1/20	5/1/20	0
Ferdigstilling	5/4/20	5/20/20	13
Skrive rapport	5/4/20	5/20/20	13
Innlevering av bacheloro...	5/20/20	5/20/20	0
Muntlig presentasjon	5/21/20	6/5/20	12
Forberede muntlig presen...	5/21/20	6/5/20	12
Muntlig presentasjon	6/8/20	6/8/20	0

Figure 2: Gantt-diagram for prosjektperioden 01.08.20 - 06.08.20

6.2 Liste over aktiviteter

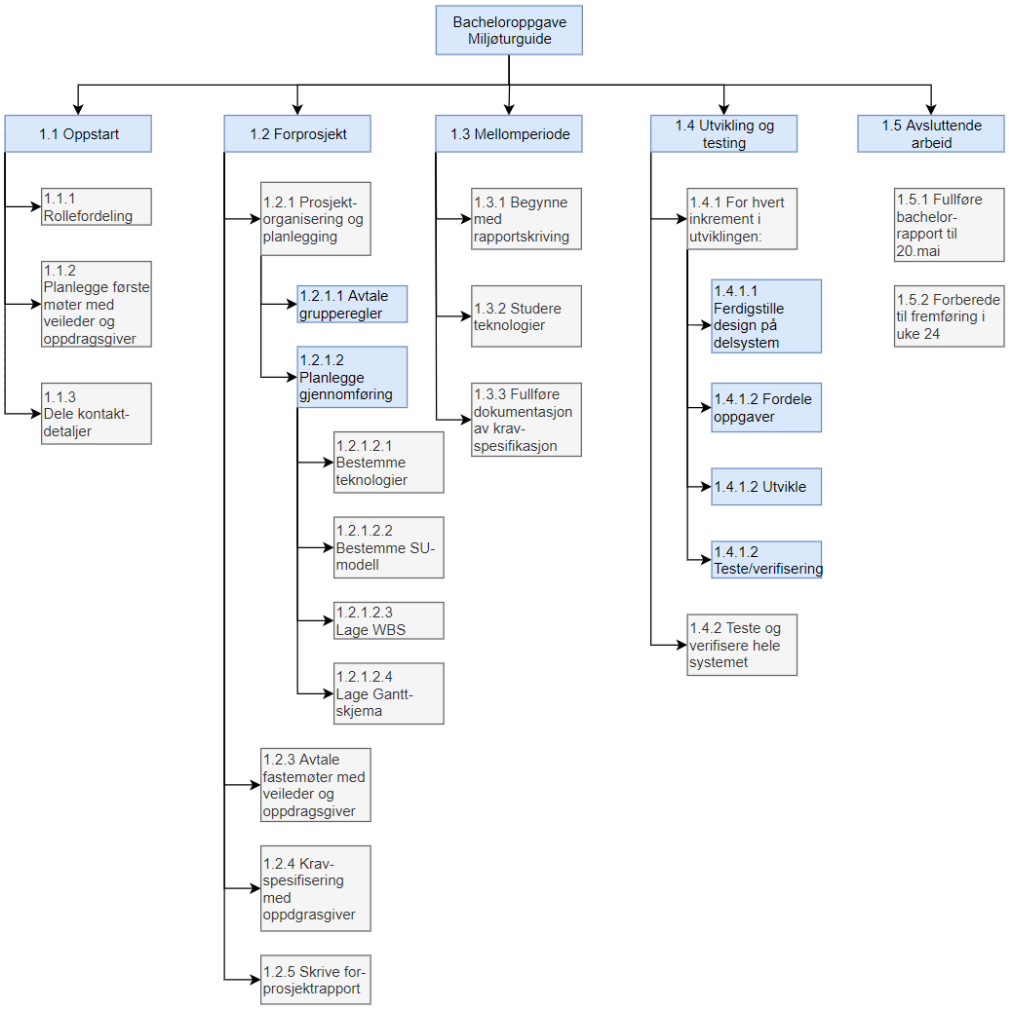


Figure 3: Work Breakdown Structure. Figuren viser planlagte arbeidsoppgaver gjennom bacheloroppgaven, delt inn i fem faser.

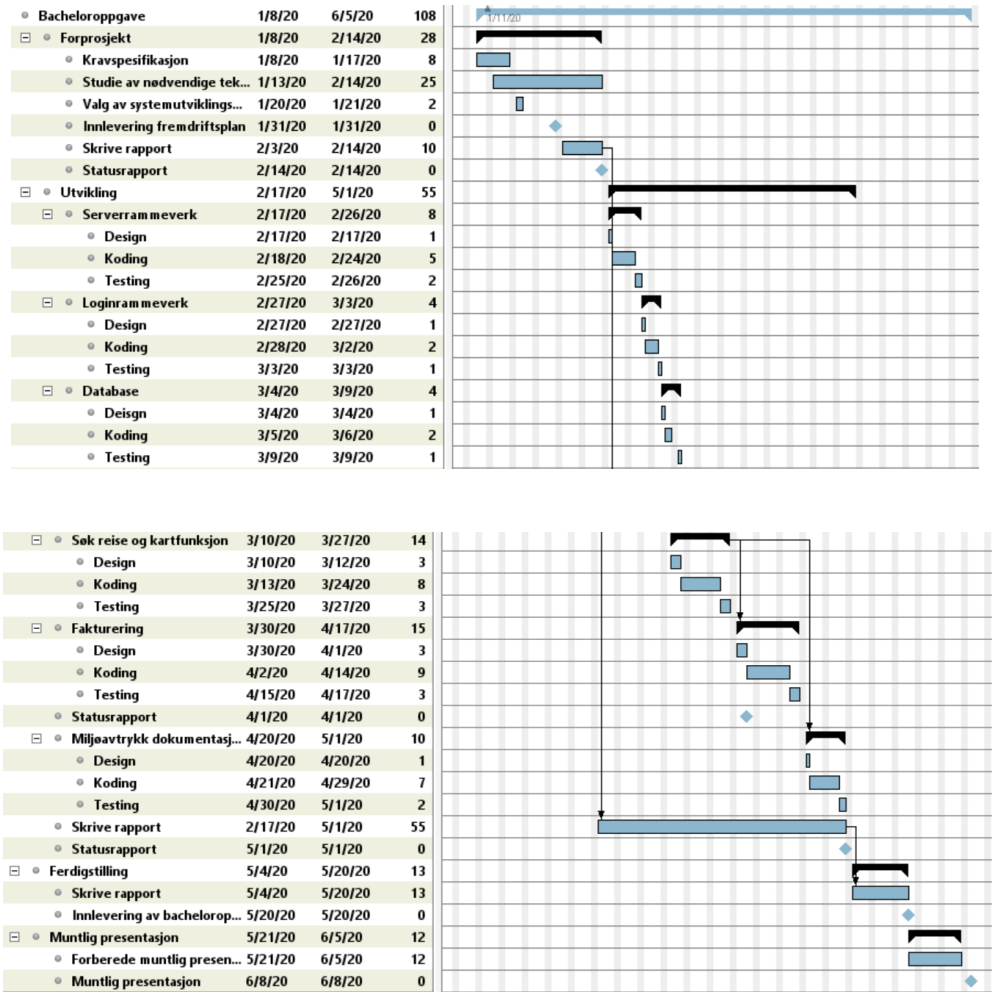
References

[1] *What is global warming, explained.* https://www.nationalgeographic.com/environment/global-warming/global-warming-overview/. [Online; accessed 14-January-2020].

[2] *Klimagassutslipp fra transport.* https://miljostatus.miljodirektoratet.no/tema/klima/norske-utslipp-av-klimagasser/klimagassutslipp-fra-transport/. [Online; accessed 20-January-2020].

Vedlegg B

Prosjektplanen



Bacheloroppgave	1/8/20	6/5/20	108
Forprosjekt	1/8/20	2/14/20	28
Kravspesifikasjon	1/8/20	1/17/20	8
Studie av nødvendige tek...	1/13/20	2/14/20	25
Valg av systemutviklings...	1/20/20	1/21/20	2
Innlevering fremdriftsplan	1/31/20	1/31/20	0
Skrive rapport	2/3/20	2/14/20	10
Statusrapport	2/14/20	2/14/20	0
Utvikling	2/17/20	5/1/20	55
Serverrammeverk	2/17/20	2/26/20	8
Design	2/17/20	2/17/20	1
Koding	2/18/20	2/24/20	5
Testing	2/25/20	2/26/20	2
Loginrammeverk	2/27/20	3/3/20	4
Design	2/27/20	2/27/20	1
Koding	2/28/20	3/2/20	2
Testing	3/3/20	3/3/20	1
Database	3/4/20	3/9/20	4
Deisgn	3/4/20	3/4/20	1
Koding	3/5/20	3/6/20	2
Testing	3/9/20	3/9/20	1
Søk reise og kartfunksjon	3/10/20	3/27/20	14
Design	3/10/20	3/12/20	3
Koding	3/13/20	3/24/20	8
Testing	3/25/20	3/27/20	3
Fakturering	3/30/20	4/17/20	15
Design	3/30/20	4/1/20	3
Koding	4/2/20	4/14/20	9
Testing	4/15/20	4/17/20	3
Statusrapport	4/1/20	4/1/20	0
Miljøavtrykk dokumentasj...	4/20/20	5/1/20	10
Design	4/20/20	4/20/20	1
Koding	4/21/20	4/29/20	7
Testing	4/30/20	5/1/20	2
Skrive rapport	2/17/20	5/1/20	55
Statusrapport	5/1/20	5/1/20	0
Ferdigstilling	5/4/20	5/20/20	13
Skrive rapport	5/4/20	5/20/20	13
Innlevering av bachelorop...	5/20/20	5/20/20	0
Muntlig presentasjon	5/21/20	6/5/20	12
Forberede muntlig presen...	5/21/20	6/5/20	12
Muntlig presentasjon	6/8/20	6/8/20	0

Figur B.1: Gantt-diagram for prosjektperioden 01.08.20 - 06.08.20

Vedlegg C

Kontrakt

NTNU

Norges teknisk-naturvitenskapelige universitet

Vår dato Vår referanse

Prosjektavtale

mellom NTNU Fakultet for informasjonsteknologi og elektroteknikk (IE) på Gjøvik (utdanningsinstitusjon), og

ETC, Electric Time Car #5

(oppdragsgiver), og

Jonas Melby

Andreas Danielsen

(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 08/1-20 til 20/5-20.

   Studentene skal i denne perioden følge en oppsatt fremdriftsplan der NTNU IE på Gjøvik 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 NTNU 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, reiser og nødvendig overnatting på steder langt fra NTNU i Gjøvik. Studentene dekker utgifter for ferdigstillelse av prosjektmateriell.
   - 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. NTNU IE på Gjøvik 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 intern og ekstern sensor. Likevel er det en forpliktelse for utøverne av prosjektet å fullføre dette til avtalte spesifikasjoner, funksjonsnivå og tider.

4. Alle bestått bacheloroppgaver som ikke er klausulert og hvor forfatteren(e) har gitt sitt samtykke til publisering, kan gjøres tilgjengelig via NTNUs institusjonelle arkiv NTNU Open.

Tilgjengeliggjøring i det åpne arkivet forutsetter avtale om delvis overdragelse av opphavsrett, se «avtale om publisering» (jfr Lov om opphavsrett). Oppdragsgiver og veileder godtar slik offentliggjøring når de signerer denne prosjektavtalen, og må evt. gi skriftlig melding til studenter og instituttleder/fagenhetsleder om de i løpet av prosjektet endrer syn på slik offentliggjøring.

Den totale besvarelsen med tegninger, modeller og apparatur så vel som programlisting, kildekode mv. som inngår som del av eller vedlegg til besvarelsen, kan vederlagsfritt benyttes til undervisnings- og forskningsformål. Besvarelsen, eller vedlegg til den, må ikke nyttes av NTNU 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 NTNU og/eller studenter har interesser.

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 NTNUs elektroniske eksamenssystem. I tillegg leveres ett eksemplar til oppdragsgiver.

8. Denne avtalen utferdiges med ett eksemplar til hver av partene. På vegne av NTNU, IE er det instituttleder/faggruppeleder som godkjenner avtalen.

9. I det enkelte tilfelle kan det inngås egen avtale mellom oppdragsgiver, student(er) og NTNU som regulerer nærmere 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 med oppdragsgiver, skjer dette uten NTNU som partner.

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

11. Eventuell uenighet vedrørende forståelse av denne avtale løses ved forhandlinger avtalepartene imellom. 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:

NTNUs veileder (navn): Frode Haug

Oppdragsgivers kontaktperson (navn): Dag I Solhaug

Student(er) (signatur): Jonas Ulleberg dato 14/1-20

Andreas Danielsen dato 14/1-20

_____ dato _____

_____ dato _____

Oppdragsgiver (signatur): Dag I Solhaug dato 13/1-20

Signert avtale leveres digitalt i Blackboard, rom for bacheloroppgaven.
Godkjennes digitalt av instituttleder/faggruppeleder.

Om papirversjon med signatur er ønskelig, må papirversjon leveres til instituttet i tillegg.
Plass for evt sign:

Instituttleder/faggruppeleder (signatur): _____ dato _____

Vedlegg D

Statusrapporter

Statusrapport1

Jonas Melby og Andreas Danielsen

February 2020

1 Status

1.1 Planlegging

Fremdriftsplanen ble vi ferdige med og leverte innen tidsfristen, og fikk denne godkjent. Føler vi var i rute da denne ble arbeidet med.

1.2 Organisering og ansvarsområder

Organisering av gruppens arbeid har vært greit. Vi har fått avtalt faste møter med oppdragsgiver og veileder hver uke. Vi snakket litt om ansvarsområder, men ettersom vi kun er to på gruppa har vi sett på det som mest gunstig å ikke fordele ansvaret noe særlig.

1.3 Klargjøring av problemstilling

Problemstillingen var tidlig avklart med oppdragsgiver. Det eneste som har endret seg nevneverdig er noen at noen problemområdene er blitt mer spesifisert og tydeliggjort siden de første møtene med oppdragsgiver.

1.4 Løsningsmetode

Vi begynner med kodingen uke 8. Uke 6 og 7 har gått til å forberede oss på kodingen ved å se nærmere på teknologiene og språkene vi ikke er så godt kjent med fra før av.

1.5 Rapportskriving

Vi er har ikke startet med rapportskrivningen, annet en de avsnittene fra forprosjektrapporten som skal inn i rapporten.

2 Oppsummering av status

Vil si vi ligger godt an i forhold til oppgaven og fremdriftsplanen vi har laget. Føler vi har grei kontroll på det som skal gjøres i løpet av de neste ukene, og at vi ligger godt an i forhold til det.

3 Trusler og problemer

Vi opplevde forrige uke at sykdom kommer veldig plutselig. Beste vi kan gjøre med det er å informere hverandre skulle dette skje, og holde hverandre oppdatert, slik at oppgavene som skal gjøres blir gjort, og at ting ikke utsettes.

4 Motivasjon

Gruppens samarbeid har gått fint hittil. Har hatt en god kommunikasjon, og vært flinke til å gi beskjed dersom det skulle være noe problemer eller at en ikke kan møte til avtalte tider.

5 Veilederkontakt

Kontakt med veileder er veldig bra. Har møter hver uke hvor vi kan stille spørsmål om det skulle være noe vi lurer på. Har også fått inntrykk av at det bare er å komme innom veileder utenom møtene om det er noe vi trenger hjelp med.

Statusrapport 2

joname

April 2020

1 Status

1.1 Organisering og ansvarsområder

Organisering av gruppens arbeid har vært greit. I utviklingsfasen har vi fått fordelt oppgaver som vi jobber med parallelt.

1.2 Klargjøring av problemstilling

Problemstillingen har ikke endret seg siden forrige statusrapport.

1.3 Løsningsmetode

Siden sist statusrapport har meste parten av tiden gått med til programmering. Her føler vi at vi har kommet oss et godt stykke i oppgaven, men med fortsatt mye igjen å gjøre. I forhold til gantt-diagrammet vi lagde i fremdriftsplanen ligger vi på etterskudd, men det var forventet.

1.4 Rapportskriving

Som nevnt har meste parten av tiden gått til programmering, men de siste ukene har det vært fokus på rapportskriving også. I tillegg skriver vi en kladd som beskriver hva vi opplever underveis i oppgaven som vi kan bruke senere i selve rapporten.

2 Oppsummering av status

Vil si vi ligger greit an i forhold til oppgaven. Vi har havnet litt på etterskudd med tanke på gantt-diagrammet, men føler ikke det er av betydning enda. Gantt-diagrammet er også kun en beregning vi gjorde tidlig i prosjektet med begrenset kunnskap om denne type arbeid.

3 Trusler og problemer

I denne spesielle perioden vi befinner oss i grunnet covid-19 har den vanlige arbeidsdagen blitt annerledes. Begge gjør det vi kan for å unngå smitte. Det har oppstått en del problemer i forbindelse med viruset. Gruppens samarbeid fungerer greit, men merker at det er lettere å samarbeide effektivt når man kan møtes fysisk. Den største utfordringen vi har møtt på er at de ukentlige møtene vi har hatt med oppdragsgiver har utgått. Heldigvis har vi fått avtalt møte via teams i fremtiden.

4 Motivasjon

Gruppens samarbeid har blitt påvirket av covid-19. Kommunikasjonen fungerer fint, men føler det er vanskeligere å planlegge like effektivt når man ikke møtes fysisk.

5 Veilederkontakt

Kontakt med veileder har vært veldig bra. Da vi har vært i utviklingsfasen av prosjektet har vi ikke hatt behov for veiledning den siste perioden. Men føler vi får raske svar via mail når det er noe vi lurer på.

Statusrapport 2

Jonas Melby og Andreas Danielsen

Mai 2020

1 Status

1.1 Organisering og ansvarsområder

Organisering av gruppens arbeid har gått relativt greit. Vi har den siste tiden jobbet med forskjellige ting i forbindelse med oppgaven parallelt, og kommet langt på hver av våre tildelte oppgaver.

1.2 Klargjøring av problemstilling

Problemstillingen har ikke endret seg siden forrige statusrapport.

1.3 Løsningsmetode

Siden forrige statusrapport har tiden hovedsakelig blitt brukt til programmering, men også en del tid er brukt til rapportskriving. Med tanke på utviklingen føler vi alltid at det er ting som kan forbedres, men med tanke på tiden vi har igjen vil vi ikke kunne endre så mye mer på det vi har.

1.4 Rapportskriving

Rapporten har vi mye igjen å skrive på, så den siste tiden vi har igjen vil nesten all tid bli brukt på rapportskriving. Dette er nødvendig for at vi skal komme i mål med rapporten.

2 Oppsummering av status

Vi kunne ligget bedre an med oppgaven. Rapportskrivingen burde vi ha fokusert mer på tidligere enn vi har gjort, men med en god innsats den siste tiden vil vi nok komme i mål. Som nevnt føler vi alltid at det er noe å endre på med tanke på utviklingen, men ser oss nødt til å si oss fornøyd med det vi har.

3 Trusler og problemer

Som nevnt i forrige statusrapport er det egentlig de samme trusler og problemer med tanke på covid-19. Dette har ført til en spesiell situasjon som har gjort at vi har måtte endre på den vanlige arbeidsdagen. Det største problemet har vært at de ukentlige møtene med oppdragsgiver uteble en periode. Heldigvis har vi hatt ukentlige møter via teams de siste ukene.

4 Motivasjon

Gruppens samarbeid har blitt påvirket av covid-19. Kommunikasjonen fungerer fint, men føler det er vanskeligere å planlegge like effektivt når man ikke møtes fysisk.

5 Veilederkontakt

Siden forrige statusrapport har vi ikke hatt så mye kontakt med veileder, men dette skyldes at vi ikke har hatt behov for det da vi i hovedsak har arbeidet med programmering. Føler likevel vi blir godt ivaretatt da veileder tar kontakt med oss for å høre hvordan vi ligger an. Vi har også fått god hjelp og tilbakemelding på rapporten da vi har sendt inn denne.

Vedlegg E

Møtereferat

E.1 Møtereferat veileder

Veiledningsmøter

Veiledningsmøte 09.01:

Gå på ooprog for å sjekke grupperegler.
Fremdriftsplan alt fra 10-30 sider
Omfag er veldig viktig i fremdriftsplanen - skal også brukes i hovedrapporten.
Gantskjema: Tidsbruk på programmering - deloppgaver

- Statusrapport til Frode.
- 15 Februar
- 1 April
- 1 mai

Avtale faste møter med arbeidsgiver, helst 1 gang i uka.
Få på plass info om oppgaven.
Mye info på blackboard.
Side 25 i lysbildene fra lynkurs.
Kanskje ikke vits med scrum - kritisk til utviklingsmodell.
Kravspekk er veldig viktig - få på plass tidlig.

Prosjektmål : Effektål, resultatmål og læringsmål. Effekt: Hva er oppdragsgiverens fortjeneste.
Resultat: hva blir levert. Læringsmål: hva har vi lært.
Rammer: praktisk, teknisk og føringer. Praktisk : vi jobber der og der,
Teknisk: OS , maskiner osv. Føringer: andre ting,kan bare jobbe så og så mye.
Timelogg: Før inn i regneark.

Veiledningstimer Mandager klokken 13.
Førstkommende Tirsdag klokka 13.

Sende til Frode minst 24 timer før.

Rapport senere: Sammenlikne gantskjema fra fremdriftsplan mot den virkelige tidsbruken

MØTEVARIGHET : 30 min

Veiledningsmøte 14.01:

Få ned det som faktisk skal gjøres - diskutere med oppdragsgiver utviklingsmodell.
Viktig å dokumentere alt vi bruker som ikke er vårt!

Veiledningsmøte 20.01:

Mulig å benytte seg av fossefall deler av prosjektet, og benytte seg av inkrementell etter hvert.
Statusmøter = statusmøter med veileder og oppdragsgiver.
Beslutningspunkter = viktige ting som kan gjøre at andre ting blir utsatt.

Dokumentasjon, standardbruk og kildekode:

- Høre med oppdragsgiver om dette.

Konfigurasjonsstyring: git/github - få tak i hverandres kode.

Gantt : Vanlig med 5 dager. Ikke ha mer enn 2 uker - da er det vanskelig å måle om man er i rute

Veiledningsmøte 27.01:

Gantt-diagrammet skal ikke endres på underveis, men lage en kopi og sammenlikne hva vi har lært.
Trenger kanskje ikke spesifisere hvilke rammeverk vi skal bruke enda.

E.2 Møtereferat oppdragsgiver

Møter med oppdragsgiver

Møte 13.01:

Avgrenser oppgaven - fristiller oss fra systemet.
Vi skal lage løsningen på en javaplatform - benytter MariaDB, tomcat, css html ajax - kan være lurt med et lett javarammeverk for login.
Reservasjonsmodul -
EnTur samler alle offentlige tilbud -
Finnes en tilsvarende i Sverige.
Få designet til CarAdmin
Miljøregnskap - Definere en bil for så og så mye avtrykk - velge ut noen biler, fossilbil og elbil, fossilbil = 100, elbil = 30 (skala), (må begrunne tall), -> gange skalatall med det faktiske avtrykket
Ha med grafer og nøkkeltall - min besvarelse
API fra EnTur
Sette opp miljøet vi skal utvikle i, dokumentere - hva slags bibliotek , skissere sidene vi skal ha - hva skal vi bygge.
Kravspekk - først - hva skal lages - detaljspesifisere i logisk forstand - hvilke sider må lages (GUI, Rappotside, reservasjonsside)

- Lage usecase
- Trenger funksjoner som gjør det og det for usecase
- Backlog på funksjoner

Faste møter med oppdragsgiver fredager - 10.00.

MØTEVARIGHET: 45 min

Møte 17.01:

Reiser mye - samling så det ikke blir for mye data.
Må ha innlogging.
Viktig å drøfte teknologivalg.
Gjennomføre faktisk bestilling på vår side. - Finne et reiseselskap som kan ta imot data og gjennomføre bestilling.
Kan avgrense ved å videresende bestilling til reiseselskap - oppgaven blir bedre om vi får til betaling på vår side.

SU:
Oppdragsgiver benytter seg av inkrementell utviklingsmodell.
Hva avhenger av hverandre? Prøve å unngå å gjøre ting opp igjen.
Skrive i oppgava at målet er å lære utvikling. Utviklingsmodell læring er mindre viktig - skrive dette.

Møte 24.01:

Spring apache - se på det.
Lurt å ha flere å velge mellom, så vi kan argumentere med hvorfor vi velger den og den. Dette viser at vi har mer kunnskap om ulike ting.
Standard java doc. Git. Funksjonsdokumenterer. Brukerveiledning.
Sånn vil vi skrive koden, sånn vil vi dokumentere - bestemme dette selv.
Sende til oppdragsgiver senest onsdag for tilbakemelding.

Møte 31.01:

Tilbakemelding rapport:

- Navn på inkrementell metode? - Inkrementell er ganske generellt.
- Risikoanalyse: Hva kan gå galt? Få med hva som er konsekvensen av f.eks. feilestimering av tidsbruk.
- Ha med et punkt mellom 1 og 4 i risikoanalysen : Korreksjon av oppgaven.
    - Eks misforståelse eller resultatet ikke tilfredsstiller oppgavebeskrivelsen eller funker i det virkelige liv.

Til neste uke:
Vi bør gå gjennom how to do med javascript, html og css.
Javascript kjører på tomcat.
Hva har vi gjort sist uke? Hva skal vi gjøre neste uke - til neste møte med oppdragsgiver.

Møte 07.02:

Java: rettighetsstyre. Neste uke: rammeverk.

Møte 14.02:

Forretningslogikk i java
Presentasjonslaget i html og javascript

Møte 21.02:

Sjekke ut spring
Forskjellen på tomcat og jetty er oppsettet
Tomcat begynner å bli gammelt.
Kan bruke jetty i utviklingsprosessen.

Om 3 uker : må ha tegnskisser om hvordan vi skal ha det.

Vedlegg F

Timelogg

F.1 Timelogg Jonas

Timelogg Bacheloroppgave – Jonas Melby

Dato	Tidspunkt	Timer	Utført arbeid
09.01.2020	10.15 - 11.45	1,5 timer	Veilednings time, grupperegler
13.01.2020	13.00 - 15.00	2 timer	Møte med oppdragsgiver, kravspek
13.01.2020	19.00 - 20.30	1,5 timer	Skrevet om Mål og rammer på fremdriftsplan, samt sett på tidligere bacheloroppaver
14.01.2020	10.00 - 13.15	3,25 timer	Arbeidet med fremdriftsplan og hatt veiledningstime
17.01.2020	10.15 - 10.45	0.5 timer	Møte med oppdragsgiver
20.01.2020	10.00 - 15.00	5 timer	Møte med veileder, research om systemutviklingsmodell, endret fremdriftsplan
21.01.2020	10.30 - 12.30	2 timer	Begynt å skrive om utviklingsmodell
24.01.2020	10.00 - 14.00	4 timer	Møte med oppdragsgiver, fortsatt med framdriftsplan
28.01.2020	10.45 - 14.00	3,25 timer	Jobbet med fremdriftsplan, risikoanalyse
29.01.2020	12.30 - 15.30	3 timer	Jobbet med fremdriftsplan, gantt-diagram
04.02.2020	14.00 - 16.00	2 timer	Læring av html, javascript og css
04.02.2020	18.00 - 19.00	1 time	Læring av html, javascript og css
05.02.2020	16.00 - 18.00	2 timer	Læring av html, javascript og css
06.02.2020	15.00 - 17.00	2 timer	Læring av html, javascript og css
10.02.2020	11.15 - 13.15	2 timer	Research av apache tomcat og shiro.
11.02.2020	12.00 - 14.00	2 timer	Skrevet statusrapport og gjort research
18.02.2020	11.15 - 14.15	3 timer	Fått implementert serverrammeverk og begynt på innlogging.
20.02.2020	14.00 - 18.00	4 timer	Shiroundersøking
25.02.2020	11.15 - 13.15	2 timer	Spring implementasjon
25.02.2020	15.00 - 17.00	2 timer	Spring implementasjon
25.02.2020	18.00 - 22.00	4 timer	Spring implementasjon / SSL
28.02.2020	11.30 - 15.30	4 timer	Login/registrering /database implementasjon
29.02.2020	15.00 - 18.00	3 timer	Login/registrering /database implementasjon - ferdig
02.03.2020	13.00 - 18.00	5 timer	Programmering
03.03.2020	12.00 - 16.00	4 timer	Programmering

	10.00 -	5	
04.03.2020	15.00	timer	Programmering
	11.00 -	5	
05.03.2020	16.00	timer	Programmering
	11.30 -	2	
10.03.2020	13.30	timer	HTML problemer
	16.00 -	4	
10.03.2020	20.00	timer	Fikset html, css og js-problemer.
	13.00 -	3	
11.03.2020	16.00	timer	Forsøkt å få til geocoder API
	16.00 -	5	
12.03.2020	21.00	timer	Fikset geocoder API med autocomplete.
	14.00 -	3	
14.03.2020	17.00	timer	Fortsatt med geocoder for å lagre valg fra autocomplete.
	14.00 -	5	
15.03.2020	19.00	timer	Implementering av mapbox API og markers basert på søk.
	17.00 -	2	Skrevet kladd om hva jeg har gjort i det siste,
16.03.2020	19.00	timer	og skrevet litt på bachelorrapporten.
	16.00 -	3	
17.03.2020	19.00	timer	Laget felter for dato og tidspunkt for avreise med mer
	17.00 -	3	
19.03.2020	20.00	timer	Rapportskriving og diverse arbeid i html/js.
	19.00 -		
23.03.2020	20.00	1 time	Rapportskriving
	12.00 -	2	
24.03.2020	14.00	timer	Rapportskriving og litt programmering.
	17.00 -		
25.03.2020	18.00	1 time	
	20.00 -	1,5	
26.03.2020	21.30	timer	Rapportskriving - use case diagram
	14.00 -		
31.03.2020	15.00	1 time	Planlegging
	11.30 -	6	Statusrapport, rapportskriving
01.04.2020	17.30	timer	(use case /use case diagram) og litt programmering
	12.00 -	4	
02.04.2020	16.00	timer	Rapportskriving
	13.00 -	6	
03.04.2020	19.00	timer	Rapportskriving, planlegging / programmering
	14.00 -	6	
04.04.2020	20.00	timer	Programmering
	13.00 -	4	
05.04.2020	1700	timer	Programmering
	12.00 -	10	Programmering - vise kun en reise om gangen –
15.04.2020	22.00	timer	reise i kart (ikke ferdig)
	13.00 -	4	
16.04.2020	17.00	timer	Programmering
	14.00 -	6	
17.04.2020	20.00	timer	Programmering
	15.00 -	2	
20.04.2020	17.00	timer	Programmering
	16.00 -	6	
21.04.2020	22.00	timer	Programmering - database

	13.00 -	5	
22.04.2020	18.00	timer	Programmering - database
	12.00 -	3	
24.04.2020	15.00	timer	Programmering - database
	13.00 -	6	
27.04.2020	19.00	timer	Programmering - database
	12.30 -	4	
28.04.2020	16.30	timer	Programmering - database
	12.30 -	6	
29.04.2020	18.30	timer	Rapportskriving - kravspesifikasjon
	12.00 -	4	
30.04.2020	16.00	timer	Rapportskriving / programmering - database
	12.00 -	3	
01.05.2020	15.00	timer	Programmering - database
	14.30 -	7	
04.05.2020	21.30	timer	Programmering - database /rapportskriving - sekvensdiagram
	14.30 -	7	
05.05.2020	21.30	timer	Rapportskriving - database - uml
	14.00 -	5	
06.05.2020	19.00	timer	Rapportskriving
	12.00 -	4	
07.05.2020	16.00	timer	Rapportskriving
	11.30 -	6	
08.05.2020	17.30	timer	Rapportskriving
	12.00 -	7	
11.05.2020	19.00	timer	Rapportskriving
	10.00 -	3	
12.05.2020	13.00	timer	Rapportskriving
	13.30 -	2	
14.05.2020	15.30	timer	Rapportskriving
	19.00 -	4	
15.05.2020	23.00	timer	Rapportskriving
	17.00 -	6	
17.05.2020	23.00	timer	Rapportskriving
	12.00 -	10	
18.05.2020	22.00	timer	Rapportskriving
	11.00 -	13	
19.05.2020	23.59	timer	Rapportskriving

F.2 Timelogg Andreas

01/09/2020

1 1/2 time - Første møte med Frode - Diskutere og bli enige om grupperegler

January 13, 2020

01/13/2020

2 timer - Første møte etter jul med oppdragsgiver, avtalte nytt møte til fredag hvor vi kommer med første utkast til design - Skissere design + starte opp arbeid med forprosjekt rapporten

January 14, 2020

01/14/2020

3 timer - Fortsatte med prosjektplan - begynnelse av kravspesifikajon, smått i gang. - 2. Møte med frode

February 4, 2020

02/04/2020

Kanskje jeg skal fortsette med denne.... 2 timer - Java studie

February 5, 2020

02/05/2020

3 timer - Java

February 7, 2020

02/07/2020

I går : 3 timer - Java

February 11, 2020

02/11/2020

4 timer - Java

Ikke ført timelogg mellom 02/11 - 19/05.

19/05/2020

14 timer - Rapportskriving

20/05/2020

xx timer - Rapportskriving

Miljøturguide

NTNU

Kunnskap for en bedre verden