Utvikling av webapplikasjon for produksjonsstatistikk

IS-304: 2022

Emnekode	IS-304
Emnenavn	Bacheloroppgave i informasjonssystemer
Emneansvarlig:	Hallgeir Nilsen
Veileder	Hallgeir Nilsen
Oppdragsgiver:	Strai Kjøkken

Studenter:

Etternavn	Fornavn
Brekka	Tor Inge
Korsnes	Anders
Land Stokkeland	Espen

Jeg/vi bekrefter at vi ikke siterer eller på annen måte bruker andres arbeid uten at dette er oppgitt, og at alle referanser er oppgitt i litteraturlisten.	JA X	NEI___
Kan besvarelsen brukes til undervisningsformål?	JA X	NEI___
Vi bekrefter at alle i gruppa har bidratt til besvarelsen	JA X	NEI___

Forord

Denne rapporten er en presentasjon av vårt arbeid, våren 2022, som utgjør siste del av bachelorstudiet i IT og informasjonssystemer ved Universitetet i Agder. Vår gruppe har bestått av tre personer, hvor alle har vært med og bidratt til å oppnå et sluttresultat som gjenspeiler den kunnskapen vi har tilegnet oss disse tre årene på studiet. Oppgaven vi har jobbet med har bydd på utvikling og utfordring. Det har vært en oppgave som alle på gruppen har vist stor interesse og motivasjon over, men som samtidig har bydd på en krevende prosess som har krevd vår fulle konsentrasjon og arbeidsinnsats. Grunnet bosituasjonen til medlemmer av gruppa, har arbeidsformen vært i størst grad digital. Dette kommer med sine utfordringer, men vi sitter igjen med en følelse at vi her har gjort vårt ytterste til å få samarbeidet til å fungere. Vi tar med oss gode erfaringer fra dette semesteret, og ser tilbake på en utfordrende og læringsrik periode.

Vi ønsker deretter å rette en stor takk til vår samarbeidsbedrift, Strai Kjøkken. Vi vil takke for et flott samarbeid og for svært god hjelp gjennom hele semesteret i form av god rådgivning, tilgang på fysisk arbeidsplass på hovedkontoret, og innsikt i det vi måtte trenge av informasjon. Denne takken rettes spesielt mot Espen Cederberg, som fra dag én har vært til hjelp for oss og bistått med sin kunnskap og den informasjonen vi har måtte trenge. Dette har vært helt avgjørende for oss i gjennomføringen av prosjektet.

Videre vil vi rette en stor takk til vår veileder, Hallgeir Nilsen. Takk for alle gode råd og tilbakemeldinger vi har fått av deg i semesteret. Dette har vært svært viktig for oss i de periodene som har vært spesielt utfordrende.

Avslutningsvis ønsker vi å takke familie, venner, og kjærester for all støtte dette semesteret. Oppgaven har krevd mye tid og energi, og det har vært helt avgjørende å ha støtten fra dere i grunnen for å holde motivasjon og inspirasjon oppe gjennom hele prosjektet.

Abstrakt

Som en del av bachelorprogrammet i IT og informasjonssystemer ved Universitetet i Agder, har vi som gruppe i løpet av våren 2022 gjennomført et prosjekt hos Strai Kjøkken i Kristiansand. Strai Kjøkken er en av Norges største kjøkkenprodusenter som leverer skreddersydde kjøkken til norske forbrukere. Det store fokuset på innovasjon og fremtidsrettet tenkning gjør at de hele tiden er på utkikk etter måter å forbedre og optimalisere produksjon og bedrift på. Vi som gruppe var på utkikk etter en oppgave som var utfordrende og givende, så da vi fikk tilbud fra Strai Kjøkken ble valget enkelt. Prosjektet startet i januar 2022, og det ferdige produktet skal leveres og presenteres 16. mai og 1. juni. Denne perioden har da gått til å utvikle en webapplikasjon som skal vise kø-tider på skap i produksjonen til Strai Kjøkken. Dette skal være med på å bidra til at Strai Kjøkken på sikt skal kunne optimalisere produksjonstiden og minimere kø-tid på skap som produseres. Gruppen har gjennom våren måtte revurdere og ta hensyn til hvilket sluttprodukt som realistisk sett skal kunne presenteres. Dette har blitt gjort etter flere vurderinger på kompleksiteten av oppgaven og utfordringer som har dukket opp underveis i prosjektet. Dette har ført til at vårt mål er å kunne levere et produkt som Strai Kjøkken skal kunne videreutvikle og på sikt skal kunne optimalisere produksjonslinjen.

Innhold

Figurliste

1. Introduksjon

Gjennom bachelorprosjektet er fokuset å gjennomføre et prosjektarbeid i samarbeid med en bedrift ved hjelp av god planlegging og struktur. Det er viktig med gode beslutninger, risikovurdering, kvalitetssikring, og god prosjektstyring. Vi har derfor i tett samarbeid med Strai Kjøkken jobbet med å utvikle en webapplikasjon som skal være med på å forbedre produksjonstiden av skap, ved å få en oversikt over kø-tid og kunne optimalisere produksjonsrekkefølgen. Det ble gitt noen kriterier på funksjonalitet av webapplikasjonen, men mye frihet til å selv velge hvordan utseende skal være. De viktigste kriteriene var å gi en plattform til de ansatte som jobber på og med produksjonen, hvor de skal kunne få en oversikt over produksjonslinjen. Webapplikasjonen skal kunne videreutvikles av Strai Kjøkken og ha direkte kontakt med databasen som henter og lagrer all data om skap og produksjon. Vi tok i bruk erfaring vi tidligere har opparbeidet oss på studiet, som bruken av en agil prosjektmetodikk, og GitHub i samarbeid med programmering.
Denne rapporten gir en beskrivelse av gjennomføringen av prosjektet våren 2022 og de ulike aspektene ved prosjektplanleggingen, analysen, prosjektgjennomføringen, produktet, og refleksjonen av utfallet.

1.1 Om bedriften

Strai Kjøkken er en kjøkkenprodusent som hadde sin oppstart helt tilbake i 1929 (Strai Kjøkken, u.d.). Det startet som trevarefabrikk og har med tiden utviklet seg til å bli den kjøkkenbedriften den er i dag. Strai Kjøkken har i lang tid hatt stort fokus på innovasjon og fremtidstenking, og investerte i 2006 i ny maskinhall, ny monteringshall og nytt automatisk lakkanlegg (Strai Kjøkken, u.d.). I dag ser vi hele tiden tegn til nytenkning, da bedriften har investert stort i nye automatiske roboter, maskiner på produksjonslinja, og andre automatiserende produkter.

1.2 Om oppgaven

Oppgaven omhandler utvikling av en webapplikasjon der ansatte på Strai Kjøkken vil få tilgang på data tilknyttet produksjons- og køtid på ordre og skap på produksjonslinjen. Applikasjonen skal være en fullverdig webapplikasjon som skal håndtere og behandle data fra en database, og gi nyttig informasjon til de ansatte om relevant statistikk knyttet til en ordres vei fra enkeltelementer og frem til fullverdige skap. Databasen som vi henter data fra er en

samling av svært mye forskjellig data knyttet til produksjonslinjen, som i hovedsak er maskin-generert. Det har derfor vært viktig å finne de relevante dataene som vil gi brukeren av applikasjonen den riktige og viktigste informasjonen. Statistikken som skal vises er derfor særlig knyttet til køtid, da dette er noe Strai Kjøkken ønsker å forbedre og optimalisere i fremtiden. Derfor er det viktig at applikasjonen visualiserer de riktige typene data, og har de funksjonene som skal til for at dette skal være mulig på sikt.

2. Sentrale prosjektvalg

Det har underveis i prosjektet blitt gjort en rekke valg som til sammen har bidratt til vårt sluttprodukt. Dette er valg som både har blitt gjort i forkant av, og underveis i prosjektet. I dette kapittelet skal vi komme nærmere inn på hvilke valg vi gjorde for å få oversikt over kvalitet, metodikk, prosjektstyring, og teknologiske valg som har vært sentralt for oss i dette prosjektet.

2.1 Krav fra Strai Kjøkken

Strai Kjøkken har gitt oss stor frihet til hvordan vi ønsker å løse oppgaven. Særlig når det kommer til valg av prosjektmetodikk og kommunikasjonsverktøy har vi fått frie tøyler. Derimot har vi vært nødt til å tilegne oss ny kunnskap knyttet til oppgaven. Webapplikasjonen har blitt skrevet i rammeverket ASP.NET Core MVC 6 med C# som programmeringsspråk. ASP.NET Core er et rammeverk under teknologien .NET fra Microsoft som er designet for utvikling av webapplikasjoner (Smith, 2022). Rammeverket tar i bruk et designmønster som heter Model-View-Controller, som gir deg god oversikt og gruppering av ulike bruksområder innad i koden (Smith, 2022). Dette språket og rammeverket tar de også i bruk i andre applikasjoner innad i Strai Kjøkken, og vil derfor gjøre at de vil kunne videreutvikle applikasjonen etter levert arbeid. Som IDE (utviklingsmiljø) ble vi anbefalt å bruke JetBrains Rider, og som DBME (databasestyringsmiljø) tar vi i bruk DataGrip som også var anbefalt fra Strai Kjøkken. For oppfølging på fremdrift har vi holdt ukentlige møter med IT-sjef. Dette er noe vi sammen ble enige om, da Strai Kjøkken har hatt god erfaring med det tidligere.

2.2 Kommunikasjonsverktøy

Ettersom gruppen er spredt geografisk, visste vi fra starten av prosjektet at vi kom til å gjennomføre mye av samarbeidet digitalt, var det viktig for oss at alle på gruppen var trygg på kommunikasjonsverktøyet vi skulle ta i bruk. Dette gjaldt både et kommunikasjonsverktøy hvor vi kunne sitte i stemmekanaler og være tilgjengelige for hverandre, og også verktøy hvor dokumenterer og annen dokumentasjon måtte være tilgjengelig og enkelt å gjøre endringer på for alle gruppemedlemmene.

2.2.1 Discord

Discord ble vårt hoved kommunikasjonsverktøy da alle på gruppen var godt kjent med tjenesten fra tidligere. Her har vi opprettet en server som inneholder kanaler hvor vi kan sitte og prate sammen, kanaler hvor vi kan legge ut linker eller annen type hjelpsomme referanser, og daily standup (kapittel 3.3.3) og møtereferater. Dette har gjort det enkelt for alle å være tilgjengelig og oppdatert uansett om man har vært fysisk til stede på Strai Kjøkken sine kontorer, eller har sittet hjemme og jobbet digitalt.
For oss har det også vært viktig å sette en kjernetid hvor alle medlemmene har måtte være tilgjengelig for hverandre. Dette har vi gjort fordi vi ser verdien av å kunne stille hverandre både små og store spørsmål underveis, uten at man skal være nødt til å arrangere spesifikke møter til dette. Det har da vært kort vei til å få svar på det man måtte lure på, og muligheten for skjermdeling har gjort at alle kan ta en nærmere titt på hva det er gruppemedlemmet ville måtte løse.

2.2.2 Microsoft Teams

Selv om Discord fungerer godt til det aller meste, mente vi at det var nødvendig med et verktøy som i større grad var egnet til dokumenthåndtering. Vi falt derfor på Microsoft Teams som verktøyet vi ville bruke til dette. Her har vi mulighet til å opprette, endre og lagre dokumenter med direkte kontakt til alle Microsoft Office-programmer. Vi kan da jobbe på samme dokument og gjøre endringer direkte sammen med de andre gruppemedlemmene på tross av hvor i landet vi har oppholdt oss. Dette verktøyet, i kombinasjon med Discord, har gjort at vi har fungert som gruppe, selv uten å være fysisk til stedet i samme rom.

2.3 Kvalitetssikring

For å sikre at man leverer et produkt oppdragsgiveren vil være fornøyd med er det svært viktig å sikre kvalitet gjennom hele prosjektperioden. Dette gjøres da på flere måter, som vi skal gå nøyere gjennom i de kommende underkapitlene.

2.3.1 Hvordan kan vi sikre kvalitet?

For å kunne sikre kvalitet er vi først nødt til å se på hva kvalitet betyr for vårt produkt, og hvordan vårt produkt kan ha god kvalitet. Kvalitet i seg selv er et begrep som er relativt, ettersom det er våre krav og oppdragsgivers krav som kan utgjøre kvaliteten av produktet. Vi ønsker derfor å definere kvalitet med utgangspunkt i Prosjekt Ledelse 4.utg skrevet av Jan Terje Karlsen. Her beskrives kvalitet slik: «Kvalitetsstyring innebærer at resultatet/produktet skal ha visse egenskaper og ytelser i samsvar med krav og spesifikasjoner.» (Karlsen, 2017, s. 27). Tatt denne definisjonen i betraktning er det derfor krav og spesifikasjoner gitt oppdragsgiver og oss selv som definerer kvaliteten i produktet. Eksempler på slike krav er at webapplikasjonen skal kunne hente og vise køtid uten falske positive resultater, og at koden skal være enkel å vedlikeholde.

2.3.2 Sikre kvalitet med oppdragsgiver

Vi har hatt god dialog med oppdragsgiver helt fra starten av prosjektet og det ble tidlig satt krav til hva de ønsket av produktet vi skulle utvikle. Disse kravene har allikevel blitt endret underveis som en naturlig del av at det har oppstått hendelser underveis som har bidratt til at vi enten ikke har hatt en realistisk sannsynlighet for å kunne nå de kravene, eller at det har blitt funnet ting som ikke har vært visst tidligere. Allikevel har det vært krav til en viss kvalitet ettersom Strai Kjøkken har et ønske om å kunne videreutvikle dette produktet etter endt prosjekttid.

Det ble da viktig for oss å sikre kvalitet ved å aller først sikre en tett dialog med oppdragsgiver, slik at de skulle kunne komme med tilbakemeldinger og veiledning, og ellers konkrete tips til endringer underveis i prosjektet. IT-sjefens sterke kompetanse innen fagfeltet bidro i tillegg til at disse møtene i stor grad var en pekepinn og kvalitetssjekk. Vi kunne derfor med jevne mellomrom sikre oss at det vi produserte levde opp til forventningene, og var på riktig vei mot et godt sluttprodukt. Dette har vært en av de viktigste metodene vi har tatt i bruk for sikringen av kvalitet, da det i all hovedsak er Strai Kjøkken som bestemmer om det

vi leverer er bra eller dårlig. Det var derfor svært viktig for oss å opprettholde og ta i bruk denne dialogen så godt som mulig.

2.3.3 Sikre kvalitet innad i gruppa

Ettersom vi visste at vi kom til å jobbe mye adskilt og dermed mesteparten digitalt var det viktig for oss å sikre gode arbeidsrutiner og dermed god kvalitetssikring. Bruken av Discord, Microsoft Teams, Trello, og Github er sentrale valg for nettopp dette.
Ved å ha en kjernetid på Discord hvor alle medlemmene skulle være tilgjengelige for hverandre sikret vi at vi til enhver tid skulle være oppdatert på hverandre sitt arbeid, samt at det var mulig å samarbeide på spesifikke oppgaver. Dette gjorde det enklere å føle seg trygg på valgene man gjorde og få innspill på om man har bommet eller truffet med disse.
Microsoft Teams gjorde det mulig for oss å kvalitetssikre dokumenter som skulle gjøres gjennom hele semesteret. Her kunne vi også jobbe samtidig på dokumenter mens vi samarbeidet på Discord. Dette gjorde planlegging, vurderinger og valg lettere og ga også bedre flyt i prosjektet.
Trello ble brukt for å sikre at både product backlog og de ulike oppgavene som måtte gjøres, ble gjort. Dette kommer vi nærmere inn på i kapittel 3.4.2, men oversikten Trello ga oss var svært viktig for kvalitetssikringen. Det å vite hva hvert gruppemedlem jobbet med, for så å ta en gjennomgang da det var gjort for å sikre at det var riktig gjennomført, gjorde at vi med større trygghet kunne gå videre til neste oppgave. Ved å ta i bruk disse verktøyene og skape kvalitet på de ulike prosessene, sikrer vi også at kvaliteten på produktet blir godt. Derfor var det viktig for oss at dette var godt planlagt og organisert.

2.3.4 Sikre kvalitet i kode

Vi tok i bruk Github for å lagre og dele kode. Github er en plattform som lar utviklere jobbe sammen på den samme koden, uansett hvor man er (github, u.d.). Her opprettet vi et felles repository hvor all kode ble lagret og var tilgjengelig for hvert gruppemedlem. Ved å ta i bruk Github, kunne hvert gruppemedlem jobbe med sin del av koden og deretter merge (slå sammen) sine endringer etter en gjennomgang med gruppemedlemmene.
Github er også en plattform hvor man kan utføre enklere versjonskontroll ved å lage branches med egne commits fra en allerede eksisterende master-branch. Enkelt fortalt betyr dette at man kopierer hele eller deler av hovedkoden, gjør endringer og forbedringer, og til slutt setter den tilbake inn i hovedkoden slik at den blir oppdatert med den siste oppdateringen (atlassian, u.d.). Dette bidrar til at man kan gjøre endringer, og potensielt feile, uten at disse feilene eller

endringene skjer på hovedkoden. Dette bidrar til å gi trygghet og kreativitet til å prøve og feile. For oss var dette svært viktig ettersom hoveddelen av prosjektet handlet om å programmere og utvikle en webapplikasjon.

Det har også vært viktig å dokumentere koden skikkelig. Dette gjøres ved å kommentere over deler av koden, slik at man får en beskrivelse av hva det er den kodesnutten gjør eller hvordan den fungerer i helhet. Ved å være god på å kommentere og dokumentere i koden bidrar man til mindre forvirring blant gruppemedlemmene og dermed også større forståelse og flyt i arbeidet. I tillegg vil god kommentering bidra til at det blir lettere for Strai Kjøkken å jobbe videre med koden i etterkant av prosjektet (Foster, 2020). Vi ønsket også å opprettholde høy cohesion og lav coupling, noe som bruken av MVC-rammeverket bidro til.

```
47      public List<TimeSpan> GetOpAndCykTime(string inputText)
48      {
49          string pattern = @"tid: ((?<operasjonstid>(\d{2}:\d{2})|\d+)s?).+tid: (?<cykeltid>\d{2}:\d{2}|\d+?.?\d+?)s?";
50          Regex r = new Regex(pattern, RegexOptions.IgnoreCase);
51          Match match = r.Match(inputText);
52
53          //vil alltid bare ha 2 entries
54          List<TimeSpan> opAndCykList = new List<TimeSpan>();
55
56          if (match.Success)
57          {
58              string op = match.Groups["operasjonstid"].Value;
59              string cyk = match.Groups["cykeltid"].Value;
60
61              TimeSpan opTime;
62              TimeSpan cykTime;
63
64              //sjekker om operation- eller cykeltid inneholder ':'
65              if(op.Contains(':') && !cyk.Contains(':'))
66              {
67                  string opMin = op.Substring(0, 2);
68                  string opSec = op.Substring(3, 2);
69                  opTime = new TimeSpan(0, 0, Convert.ToInt32(opMin), Convert.ToInt32(opSec));
70                  cykTime = new TimeSpan(0, 0, 0, Convert.ToInt32(Math.Round(Convert.ToDouble(cyk))));
71              }
72              else if(cyk.Contains(':') && !op.Contains(':'))
73              {
74                  string cykMin = cyk.Substring(0, 2);
75                  string cykSec = cyk.Substring(3, 2);
76                  opTime = new TimeSpan(0, 0, 0, Convert.ToInt32(Math.Round(Convert.ToDouble(op))));
77                  cykTime = new TimeSpan(0, 0, Convert.ToInt32(cykMin), Convert.ToInt32(cykSec));
78              }
79              else if(op.Contains(':') && cyk.Contains(':'))
80              {
81                  string opMin = op.Substring(0, 2);
82                  string opSec = op.Substring(3, 2);
83                  opTime = new TimeSpan(0, 0, Convert.ToInt32(opMin), Convert.ToInt32(opSec));
84
85                  string cykMin = cyk.Substring(0, 2);
86                  string cykSec = cyk.Substring(3, 2);
87                  cykTime = new TimeSpan(0, 0, Convert.ToInt32(cykMin), Convert.ToInt32(cykSec));
88              }
89              else
90              {
91                  opTime = new TimeSpan(0, 0, 0, Convert.ToInt32(Math.Round(Convert.ToDouble(op))));
92                  cykTime = new TimeSpan(0, 0, 0, Convert.ToInt32(Math.Round(Convert.ToDouble(cyk))));
93              }
94
95              opAndCykList.Add(opTime);
96              opAndCykList.Add(cykTime);
97          }
98          return opAndCykList;
99      }
100 }
```

Figur 1: Funksjon

Figur 1 viser en funksjon som har ansvar for å hente og riktig formatere operasjons- og cykeltid i en string den blir matet med. Denne stringen kommer i vårt tilfelle fra én hendelse i databasetabellen «ProductionEventLog». Fulgte kodestandarder her er forklarende navngivning av variabler, kommentering og cohesion.

2.4 Risikovurdering og usikkerhet

«Ved vurdering av nye prosjektforslag bør alltid usikkerheten identifiseres og analyseres (Karlsen, 2017, s. 56). I prosjekter med høy usikkerhet betyr det at risikoen for at noe skal gå galt er større. Det er derfor viktig at vi tar høyde for usikkerheten i prosjektet og deretter kalkulerer hvilke typer risikoer vi står ovenfor. For oss handlet usikkerheten i aller størst grad om teknologisk usikkerhet og resursmessig usikkerhet (Karlsen, 2017, s. 56). Dette er usikkerheter som kan forekomme som interne eller eksterne og dette er det vi skal gå nærmere inn på i dette kapittelet.

2.4.1 Intern usikkerhet

Intern usikkerhet kan oppstå fra flere forhold. Dette kan skyldes arbeidsmetoder, tilgjengelig kompetanse, det tekniske konseptet og organisering, som noen eksempler (Karlsen, 2017, s. 415). For oss handlet det om stor usikkerhet i forhold til det å sette seg inn i nytt programmeringsspråk, og det å samarbeide som gruppe. Vi satt derfor av en stor porsjon tid til kompetansebygging, slik at vi i større grad følte oss sikre på prosjektets konsept og ferdighetskrav. Vi fant også tidlig ut at vi måtte ta hensyn til bosituasjon, da vi har to medlemmer som bor langt borte fra campus, og en er spesielt langt borte. Dette medførte til at personen som oftest stilte fysisk på Strai Kjøkken sine kontorer måtte ta på seg en del ansvar knyttet til å få oversikt over produksjonslinja sammen med databasen, og andre analysebaserte oppgaver. Dette gjorde at vi var sårbare for fravær. Det var også viktig å få prioritert besøk på produksjonslinja og kontoret for de to andre medlemmene slik at alle på gruppa hadde en forståelse og oversikt over hva problemdomenet bestod av.

2.4.2 Ekstern usikkerhet

Ekstern usikkerhet handler om forhold man i mindre grad har kontroll over. Dette er forhold som kan påvirke prosjektets gang, men som skapes utenfra (Karlsen, 2017, s. 415). For oss har dette i størst grad handlet om noen få ting. Det har vært Covid, som vi har kjent til at er en usikkerhet vi har vært nødt til å ta i betraktning. Både på grunn av smitte, men også på grunn av mulig oppblomstring og ny nedstengning. En annen ekstern usikkerhet har vært produksjonslinjen på Strai Kjøkken. Denne var nede i store deler av oppstarten til prosjektet, og førte til større forsinkelser. Dette gjorde at vi ble nødt til å arbeide noe annerledes over en periode, og at en større del av prosjektet gikk til kompetansebygging enn hva som først var antatt, da vi ikke fikk tilgang til problemdomenet.

2.5 Andre sentrale valg

Det var helt fra starten av prosjektet en dialog med Strai Kjøkken om å få inn en gruppe med dataingeniører som skulle være med på prosjektet. Denne gruppen skulle i større grad utføre oppgaver som handlet om å installere sensorer på maskiner slik at vi fikk enda mer presise data som vi kunne jobbe med. Prosessen med å få inn en slik gruppe fikk vi være delaktig i, da denne gruppen kom senere i gang med prosjektet. Dette førte til et møte med oss, dataingeniør-gruppa og Strai Kjøkken. Valget vi stod ovenfor var om vi ønsket å stifte samarbeid med dataingeniørene, og om vi så mer verdi enn utfordringer knyttet til dette valget. Det endte til slutt med at vi ga vår ærlige mening til Strai Kjøkken, og kom deretter til en felles beslutning om å ikke ta inn dataingeniørene da usikkerheten var for stor. Grunnen til dette var fordi både vi som gruppe, og tilbakemeldinger fra Strai Kjøkken var at ingeniørgruppen virket lite motiverte i tillegg til at alle medlemmene var klare for å jobbe digitalt og ingen fysisk. Dette førte til at vi ikke valgte å risikere enda en periode med kompetansebygging på vegne av en ny gruppe, samt at vi så risikoen ved at de virket umotiverte for oppgaven og dermed ikke ville komme til å bidra i stor nok grad for at gevinsten skulle veie sterkere enn utfordringene. Denne beslutningen har vært en av våre aller største aktive beslutninger dette semesteret, og vi tror den dag i dag at vi gjorde det riktige.

3. Gjennomføring av prosjektet

For å sikre en god gjennomføring av prosjektet er det viktig å definere et rammeverk for arbeidet. Dette bør gjøres så tidlig som mulig i prosessen, og setter deretter standarden for hvordan vi skal jobbe som gruppe. Gruppen valgte å gå for et agilt prosjektrammeverk og i den kategorien falt vi på rammeverket Scrum som er mye brukt, spesielt i utviklingsmiljøer. Vi kommer til å gå litt mer i dybden om Scrum i kapittel 3.3 Scrum, men ved god utførelse av dette rammeverket sikrer gruppen god struktur, god fremdrift med jevnlige møter, og god refleksjon. For å kunne utføre møter, og opprettholde god kommunikasjon, brukte vi Discord og Microsoft Teams, som nevnt i kapittel 2.2.1 og 2.2.2.

3.1 Planlegging og analyse

For å gjennomføre et slikt prosjekt som dette, medfører det en del arbeid for å forstå hva som trengs, spesielt gitt at det er et nytt prosjekt. Vi begynte med at Strai sendte en skriftlig formulering av hva oppgaven handler om, og deretter hadde vi møter for å bedre forstå

situasjonen. Det første møtet var i desember og vi fikk en overordnet gjennomgang av situasjonen vi skulle jobbe i forhold til. Over nyttår fikk vi nye møter og åpen tilgang til fabrikken, men siden maskinene sto stille kom informasjonen vi hadde som grunnlag for analyse og planlegging fra møter og videoer. Dette førte til at gruppen planla å bruke tiden på å sette seg inn i teknologien frem til vi fikk tilgang på problemdomenet og reell data. Da maskinene på linjen var oppe og gikk, ble gruppemedlemmet med dette ansvarsområde sittende nede i fabrikken og sammenligne informasjonen i databasen opp mot informasjonen som reelt forekommer i produksjonen. Å forstå hvordan informasjonen i databasen henger sammen tok noe tid, men dette ble veldig nyttig for videre forståelse av problemdomenet.

3.2 Prosjektmetodikk

Vi sto i første rekke mellom agil og waterfall metodikk, men valgte agil metodikk da vi hadde betydelig usikkerhet i prosjektet, og waterfall er mest effektivt dersom man har et veldefinert problem som skal løses. Av de agile metodikkene vi kjenner falt vi på Scrum. Begrunnelsen for denne beslutningen er beskrevet i kapittel 3.3. I etterkant har vi vurdert om vi skulle brukt en variant mer mot scrumban, for mer fleksibilitet (ProdcutPlan, u.d.). Kanban ble droppet da vi så behov for å ha noe mer struktur enn Kanban tilbyr (ProdcutPlan, u.d.), grunnet begrenset med erfaring.

3.3 Scrum

Scrum er et agilt rammeverk egnet for å løse uforutsette og komplekse problemstillinger (Scrum.org, u.d.). Det er lett å ta i bruk Scrum, men svært vanskelig å mestre. Scrum er med vilje ufullstendig, og definerer bare kjernen som trengs for å implementere teorien, og så tilpasser og utvider man etter behov for å fylle ut manglene i rammeverket i hvert enkelt tilfelle (Scrum.org u.å.). Ved anvendelse av Scrum utvikler man et produkt i korte iterasjoner med faste lengder. Disse lengdene er gjerne på 1 måned eller kortere. Hovedmålet er å levere funksjonalitet som man har satt som mål før hver av disse periodene, og etter hver periode går man gjennom om målene ble nådd, hva som kan gjøres annerledes, og hva som blir neste mål (glasspaper, u.d.).

3.3.1 Roller

Ved anvendelse av Scrum er det vanlig med en rollefordeling. De vanligste rollefordelingene er Product Owner, Scrum Master og Developer (Scrum.org, u.d.). Product Owner, eller

produkteier er i dette tilfellet Strai Kjøkken. Dette er vår oppdragsgiver og ham vi skal levere produktet til. En Scrum Master har som oppgave å veilede gruppen, delegere oppgaver og sikre at oppgaver blir gjort, og sikre at prosjektet gjennomføres med de ulike Scrum begivenhetene (Scrum.org, u.d.). For vår gruppe så fant vi det hensiktsmessig å ikke dedikere Scrum Master rollen oss imellom. Dette valget gjorde vi med bakgrunn i at vi er en relativt liten gruppe, med bare tre medlemmer, og at ingen av oss ønsket å ta denne rollen. Vi valgte derfor å la rollebestemmelsen innad i gruppa være flytende, de som ønsket å ta ansvar for noe fikk gjøre det, og de som passet til én spesifikk oppgave fikk naturlig mer ansvar der. Denne løsningen har vi vært godt fornøyde med gjennom hele prosjektet, og mener var helt riktig for oss.

3.3.2 Sprinter

En sprint er hjertet i scrum (Schwaber & Sutherland, 2020). Det er definert som en periode der man jobber med definerte oppgaver som står i Product Backlog (kapittel 3.3.4), helst under en måned, og består av Daily Scrum/Standup, Sprint Planning, Sprint Retrospective og Sprint Review. Vi valgte å dele prosjektet inn i fire hovedsprinter. I tillegg til dette har vi én pre-sprint og én sprint som vil gå fra levering av rapport og frem til muntlig fremføring av prosjektet. Bakgrunnen for dette valget var at fire sprinter ville utgjøre perioden mellom oppstart og innlevering av rapport, og sprintene ville få en tidsperiode på én måned per sprint. På grunn av den omfattende kompetansebyggingen og mengden med ny kunnskap som måtte tilegnes, så vi behovet av én måned per sprint. Sprint 1 var noe lenger enn en sprint er ment å være, men dette ble korrigert fra Sprint 2 og videre. Bakgrunnen for dette avviket var at kompetansebyggingen tok noe lenger tid enn først ansatt, sammen med stopp på produksjonslinjen.

3.3.3 Daily Standup

En Daily Standup er et møte gruppen skal ta på gitte dager for å fortelle hva som er blitt gjort, hva som skal gjøres videre, og hvilke utfordringer som kan komme i veien for fremgang. Dette møtet valgte vi å gjennomføre hver tirsdag og fredag, men vi endret dette senere til hver dag vi møtes, det vil si mandag, tirsdag, torsdag og fredag. Bakgrunnen for å være nøye og å utføre disse møtene såpass regelmessig var for å sikre kvalitet og mulighet for innspill på arbeidsoppgaver. Dette har gitt hvert gruppemedlem en oversikt over de andre sine planer, og

kunne derfra følge hverandres progresjon og komme med innspill på om det trengtes mer korrigering.

Spørsmålene vi var nødt til å svare på var:

1. Hva har du gjort til i dag?
2. Hva skal du gjøre til neste gang?
3. Hvilke utfordringer kan oppstå for din gjennomføring?

3.3.4 Product Backlog

En Product Backlog er en detaljert liste over alt som kan være nødvendig for selve sluttproduktet i et i prosjektet (Schwaber & Sutherland, 2020). Det er den eneste kilden til arbeid som et Scrum team skal gjøre. Produkt eieren er i utgangspunktet ansvarlig for backloggen. Dette dokumentet er dynamisk, og vil utvides kontinuerlig når man ser behov for nye oppgaver. Hendelser som førte til oppdatering av product backlog var som oftest når vi eller oppdragsgiver så behovet for nye funksjoner, eller vi fant feil i koden som måtte fikses. Det første utkastet dekker starten av prosjektet, og for hver sprint som gjøres kan man få flere ting lagt inn i prosjektets Product backlog.
Vår Product backlog er noe kortere og mindre spesifisert enn vanlig, noe vi valgte da det var stor uvisshet i startfasen av prosjektet hvilke konkrete oppgaver vi måtte løse. Teamet har hatt større frihet enn normalt i et scrum prosjekt.

3.3.5 Sprint Backlog

En Sprint backlog består av utvalgte oppgaver fra Product Backlog. Det er en oversikt over hva som skal gjøres i en spesifikk sprint, og skal ikke vikes fra. Hvis man opplever at gjøremålene ikke lenger er relevant, kan produkteier kansellere resten av sprinten og man kan gå videre til en ny runde med sprint planning, for å lage en ny Spring backlog. Enkelt kan man si at en sprint backlog visualiserer av hva som trengs for å nå den enkelte sprintens mål (Schwaber & Sutherland, 2020).

3.3.6 Sprint Planning

En sprint planning er steg en av en sprint. Arbeidet som skal gjøres i løpet av sprinten hentes ut av prosjektets Product Backlog. Produkt eieren viser til hvordan man kan øke verdien og nytten av produktet innad i sprinten. Det er tre hoved spørsmål man går over i løpet av en

sprint planning. «Hvorfor er denne sprinten verdifull?», «Hva kan gjøres i denne sprinten?» og «Hvordan vil arbeidet som er valgt bli gjort?» (Schwaber & Sutherland, 2020). Våre sprint planning møter har hatt noe forskjellig format, ofte noe mindre formelt enn slik som Scrum ønsker at det skal være, men dette er en følge av at første halvdel av prosjektet i stor grad har handlet om informasjonsinnhenting og kompetansebygging. Det har i den sammenheng vært naturlig å tilpasse gjennomføringen til situasjonen som har forekommet.

3.3.7 Sprint Review

I en Sprint Review presenterer Scrum Teamet først hva som er gjort i løpet av sprinten, og man diskuterer deretter over hvilke endringer til planen som burde gjøres videre. Det er nå man gjerne legger inn ny informasjon i Product Backlog (Schwaber & Sutherland, 2020). Hos oss har det tidvis hendt at vi har hatt delvise Sprint Reviews mellom hovedrundene. Disse har da forekommet grunnet tett kontakt med Produkt eieren, og temaer som normalt er forbeholdt Sprint Review har naturlig kommet opp i kommunikasjonen. Dette bidro til at vi fikk en oversikt over potensielle forbedringer som kunne gjøres frem mot neste sprint-periode.

3.3.8 Sprint Retrospekt

Mens et Sprint Review ser på oppgaven, ser et Sprint Retrospekt på kvalitet og effektivitet. Scrum teamet skal her gå over hva som har gått bra, hva som har gått dårlig og hvordan problemer har blitt løst, eventuelt hvorfor problemene ikke er løst. Som et team finner man deretter endringer man kan gjøre for å forbedre effektiviteten til gruppen, og de endringene med mest direkte effekt håndteres så raskt som mulig. Om man finner noe som burde bli en del av Product Backlog, kan det legges inn her. Når Sprint Retrospektet er ferdig, så er sprinten endelig helt ferdig (Schwaber & Sutherland, 2020).

3.3.9 Sprintgjennomføringene

	Sprint planning	Sprint review	Sprint retrospect
Pre-sprint	For å effektivt kunne komme i gang med sprint en, så hadde vi møter og epost utveksling med produkt eieren. Konklusjonen kan oppsummeres som at Sprint 1 sin backlog måtte handle om kompetansebygging. Rammer for tidsbruk og lignende ble planlagt i denne perioden. Det kom tydelig frem at målet med oppgaven er å kartlegge køtid for fire maskiner, og presentere dette visuelt.		
Sprint 1 08.01-14.02	For sprint en var det veldig lite detaljer i sprint planning. Vi la generelle oppgaver for kompetansebygging, slik som CRUD, REST-api, enkelt frontend oppsett og lignende. Prioriteten lå i hovedsak på å lære seg .NET 6. Det ble også lagt inn noe analysearbeid i backlogen. Første punkt for alle sammen var oppsett av miljø på maskinene.	Det ble kjørt forskjellige øvelsesprosjekt, og vi fikk i hovedsak til deler, men kom ikke i mål. Underveis kom det frem at vi burde bruke EntityFramework Core for å gjøre flere av oppgavene når man jobber med .NET 6, og det viste seg at mange guides var i tidlig .NET framework stil. Vi opplevde problemer med å få ting til å kjøre på visse enheter. Det ble i denne perioden laget ER diagrammer for å sikre lik grunnforståelse av problem domenet slik det	Vi fant ut at vi på en bedre måte måtte dele ressurser for kompetansebygging, både gode og dårlige. Det sløses for mye tid om to personer skal gjennom samme dårlige video, når den første kunne advart om problemet med kvaliteten. Som følge av dette gjøres det femover endringer for hvordan vi bruker relevante discord kanaler. Legge til rette for rutiner for at alle møter likt.

		var fremstilt for oss. Produkteieren fremsto fornøyd.	
Sprint 2 15.02-14.03	Produksjonen forventes å være oppe å gå i løpet av denne sprinten, så nå tenkes det at vi deler gruppen litt mer. Espen møter fysisk og kan ta analyse, og Tor Inge og Anders opplever utfordringer med gjevnt fysisk oppmøte og er derfor med digitalt. For denne Sprinten sin backlog prioriteres derfor analyse av problemdomenet, og et brukbart skeleton av applikasjonen.	Vi har nå fått gjort bedre analyse og samlet data som gir realistisk dummy data for frontend. Dette presenteres grafisk og i tabell i applikasjonen. Utfordringer med sykdom når noen er borte og andre er der.	Vi vil prøve å legge til rette for at flere skal kunne hver ting om mulig, for å være mindre sårbare for sykdom i neste periode. Gruppen burde ikke få store problemer om bare en person er borte.
Sprint 3 15.03-13.04	I denne perioden er målet å få direkte data hentet ut fra databasen, skrive kode som behandler denne dataen og få laget enklere modeller for å implementere nødvendig funksjonalitet.	Oppkobling mot Strai sin database hvor vi får hentet ut alle ordrenummer i løpet av en dag lykkes. Databehandling er påbegynt og kommet godt på vei. Vi får noen forbedringer for designet, men opplever utfordringer grunnet bootstrap. Ytterligere analyse gjort, men ikke kommet i mål. Svært mye sykdom. RegEx løsning for uthenting av operasjonstid og syklustid påbegynt, men virker bare av og til. Produkteier viser forståelse for at uforutsette ting kan hende. Produkteier er tydelig på at vi må forenkle ytterligere,	Vi opplever at vi sliter med at ikke alle holder kjernetiden, og må derfor gjøre noe for å fikse dette. Sykdom er sannsynligvis ikke noe stor hindring videre, da det er liten sjanse for mer sykdom etter flere tilfeller med corona og influensa er gjennomgått innad i gruppen.

		og fokusere på å løse det som trengs, og la det som kan være greit vente.	
Sprint 4 19.04-13.05	Vi tenker å lage ny CSS løsning for oss selv, og gjøre store forbedringer til frontend. Rapport skriving flyttes nå høyere i prioritetene, og kjernefunksjonaliteten må nå ferdigstilles i backend delen. Alt dette flyttes derfor over i Sprint Backlog for sprint 4.	Vi har fått implementert en tilleggsfunksjon på henting av ordreinformasjon basert på dato. I tillegg har det skjedd store endringer på webapplikasjonens layout. Views på statistikk, henting av køtid på ordre, og henting av køtid på dato har blitt implementert.	Vi har fått bedre flyt i arbeidet. Og rollefordelingen har hatt stor innvirkning på dette. Det ble gjort en mer konkret rollefordeling her enn tidligere. Utover dette har sprinten blitt noe forkortet mtp utvikling, da bachelorrapport fikk en større fokus enn tidligere.

3.4 Endringer underveis

Underveis i prosjektet har det vært stadige endringer, og disse endringene har vært et resultat av at vi har sett at ting ikke har gått eller virket slik som vi først har antatt. Dette er enten endringer på arbeidsmetode, verktøy, arbeidstidspunkt eller lignende. Alle disse tilpassingene har vært svært viktig for oss som gruppe og for gjennomføringen av prosjektet.

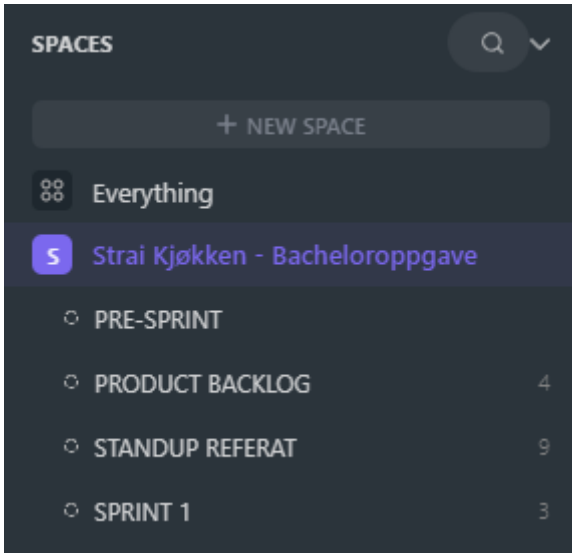
3.4.1 Daily Standup

I oppstarten av prosjektet ble vi enige om å ha standup to ganger i uka. Disse to gangene skulle lande på tirsdager og fredager. Vi antok at det ville være nok til å få innblikk i hvordan ting gikk, og om det var noen endringer som måtte gjøres for å korrigere. Dette forandret vi på i starten av april, da vi så et behov for å øke antallet og ha daily standup samtlige arbeidsdager. Bakgrunnen for denne beslutningen var at vi så et behov for å i større grad melde ifra om hva som var gjort og hva planen videre var. Dette er også i tråd med rammeverket til Scrum og daily Scrum (Schwaber & Sutherland, 2020). Med tanke på fordelingen av oppgaver og det samspillet vi har vært nødt til å ha ga dette valget stor mening. Dette førte til mer kontroll for alle gruppemedlemmene og et enda tettere samarbeid. I tillegg fikk vi mulighet til å anpasse oss hverandre i enda større grad ved forskjellig forståelse av hva som måtte gjøres.

3.4.2 Fra ClickUp til Trello

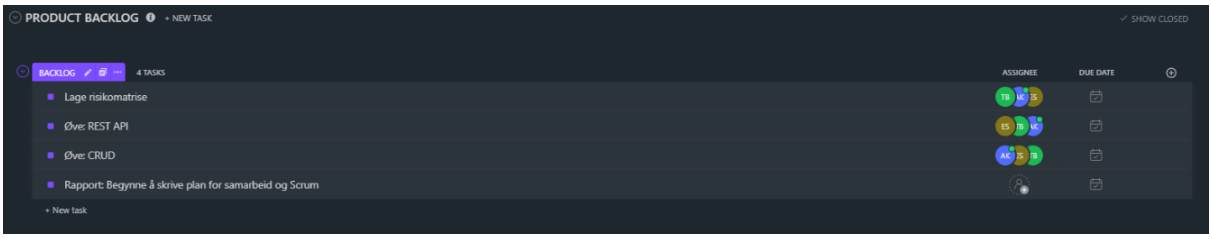
Frem til litt ut i mars brukte vi Clickup som hovedverktøy for planlegging. Dette viste seg tungvint, da funksjonaliteten vi så behov for viste seg å være bak betalingsmur og verktøyet virket noe rotete til vårt behov.

Figur 2 viser Clickup sin fordeling av ulike «Spaces». Dette var kanaler man opprettet, hvor man inni hver kanal kunne opprette oppgaver, gjøremål og lignende. Vi mente det ble for mange ulike kanaler å forholde seg til, noe som førte til at vi ved enkelte situasjoner lot være å legge inn oppgaver eller hendelser.



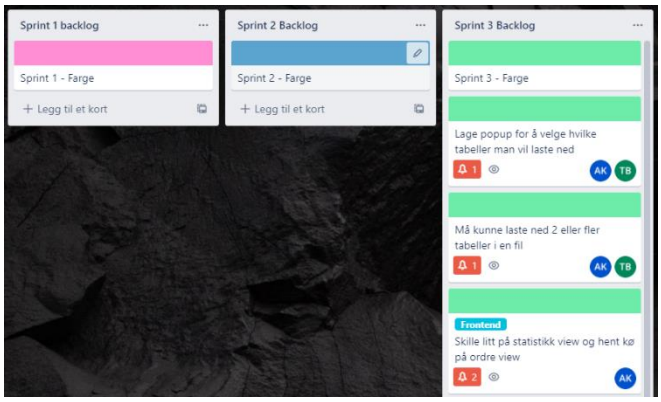
Figur 2: Clickup - Spaces

Figur 3 viser et eksempel på hvordan vi la inne oppgaver i backlog. Også her mente vi at oppsette ble for rotete og uoversiktlig, noe som til slutt endte med å bli grunnen til at vi endret verktøy for prosjektstyringen.



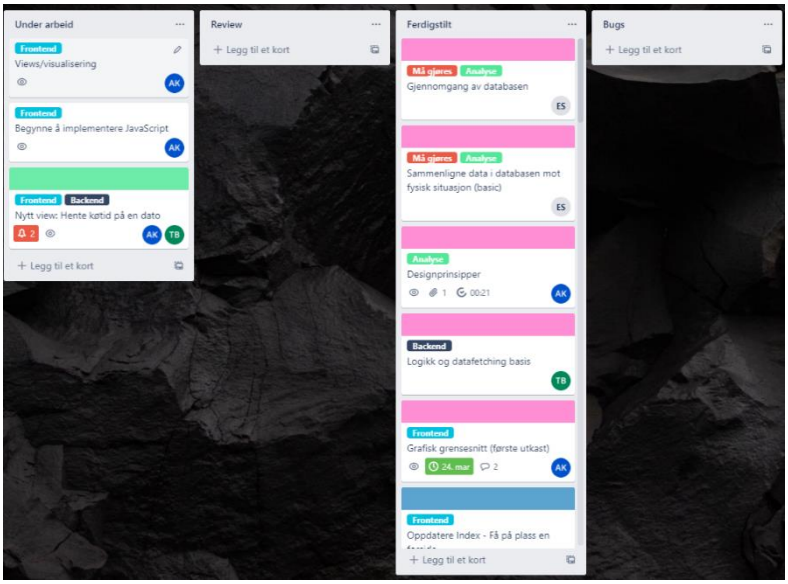
Figur 3: Clickup - Product Backlog

Vi valgte derfor å se etter andre verktøy og landet til slutt på Trello. Trello er et enkelt-å-bruke verktøy for organisering av prosjekter og delegering og kontroll på oppgaver innad i et prosjekt (Arun, 2021) Dette var grunnen til at vi valgte å ta i bruk Trello, da grunnen til byttet i hovedsak var et for komplisert verktøy.



Figur 4: Trello - Sprinter

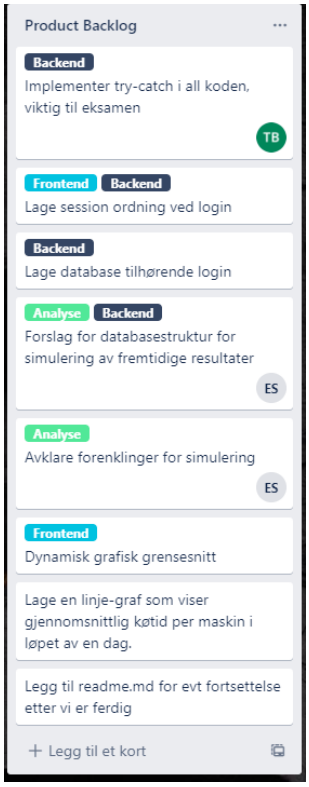
Figur 4 viser organiseringen av de ulike Sprintene. Inne i Trello ble sprintene fargekodet for å ha kontroll på de ferdigstilte oppgavene og i hvilke sprinter disse oppgavene ble ferdigstilt.



Figur 5: Trello - Sortering av oppgaver

Figur 5 viser sorteringen av oppgaver. Så fort en oppgave ble påbegynt ble den flyttet til «Under arbeid». Her lå den helt til oppgaven var ferdig og deretter flyttet til «Review». Når et gruppemedlem hadde gått over oppgaven sammen med den personen som i utgangspunktet løste det ble det flyttet til «Ferdigstilt» og personen kunne ta på seg en ny oppgave. Hver oppgave ble fargekodet etter hvilke sprint den tilhørte, ytterliggere fargekodet etter hvilken oppgavetype den hadde (Backend, Frontend, Analyse osv.), og tilegnet personen som jobbet på den med initialer.

Figur 6 viser oversikten over Product Backlog og de oppgavene vi visste vi måtte løse. Disse ville gjennomgå den overnevnte prosedyren avhengig av hvilken sprint den ble påbegynt.



Figur 6: Trello - Product Backlog

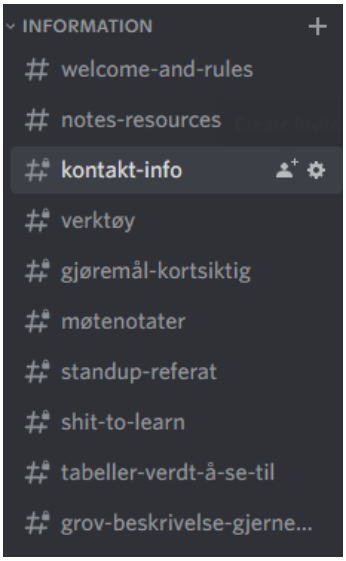
3.4.3 Kjernetid

Kjernetid er noe som jevnt over hele prosjektet har vært oppe i diskusjon hos gruppen. I oppstarten gjorde vi en avtale på å ha kjernetid fra 09:00 til 16:00 da vi så behovet for at alle var til stedet som om det var en vanlig arbeidsdag. Dette gjorde vi flere endringer på, hvor vi så satte kjernetiden fra 09:00 til 15:00, før vi endte på 10:00 til 14:00. Begrunnelsen til dette valget var at to av gruppemedlemmene har hatt en livsstil som har vært noe uforutsigbar. Det å sette en kortere kjernetid, hvor sannsynligheten for at alle er til stede er større, har derfor vært svært gunstig for samarbeidet i gruppen. Vi så en økning i samarbeid og en økning i aktivitet i hele gruppen etter at dette ble

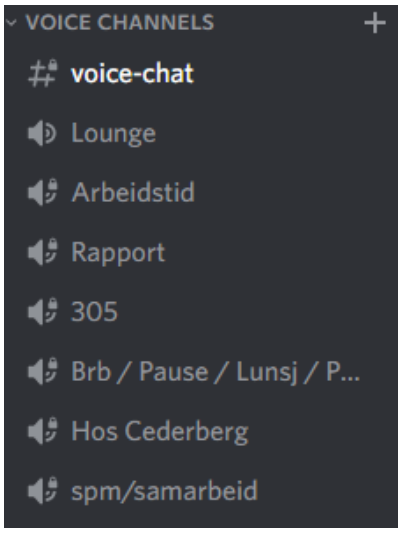
gjort. Det at vi i løpet av en fire timers kjernetid rekker å ha standup, og så gå over alt vi trenger hverandres hjelp og input på, for så å kunne jobbe alene i fred den resterende tiden, har også vist seg å være en betydelig fordel når det kommer til flyt i prosjektet.

3.4.4 Kanaler i Discord

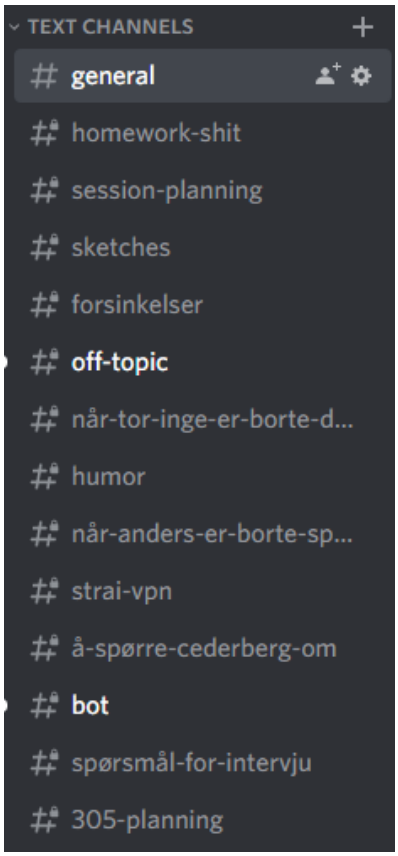
Discord har som nevnt tidligere vært hovedsamarbeidsverktøy og det stedet vi har vært koblet til i kjernetiden. I oppstarten brukte vi én stemmekanal for interaksjon mellom hverandre, hvor man kunne stille spørsmål og være tilgjengelig for besvarelse og hjelp. Et problem vi så oppstod fra dette var at vi fort kunne gli ut til temaer som ikke nødvendigvis omhandlet problemstillingen, eller at én av gruppemedlemmene ble forstyrret av samtalen til de to andre. Vi valgte derfor å løse dette med å i enda større grad ha dedikerte stemmekanaler for den situasjonen man satt i. Vi opprettet derfor en kanal hvor man kunne invitere det gruppemedlemmet man trengte hjelp eller samarbeid fra. Dette førte til at den i gruppen som ikke behøvde å hjelpe eller å diskutere kunne sitte i fred i hovedkanalen, mens de to andre byttet kanal. På den måten skapte vi større ro for de som satt i arbeid, samt muligheten for å diskutere og samarbeide når det måtte til. Ellers har vi strukturert discord serveren vår med dedikerte tekst kanaler til forskjellig informasjon, alt fra forskjellige referater, til nyttige ressurser og beskjeder eller generell prat om hva enn som virker relevant. Det å ha denne kommunikasjonen sortert gjør det veldig lett for oss å hente



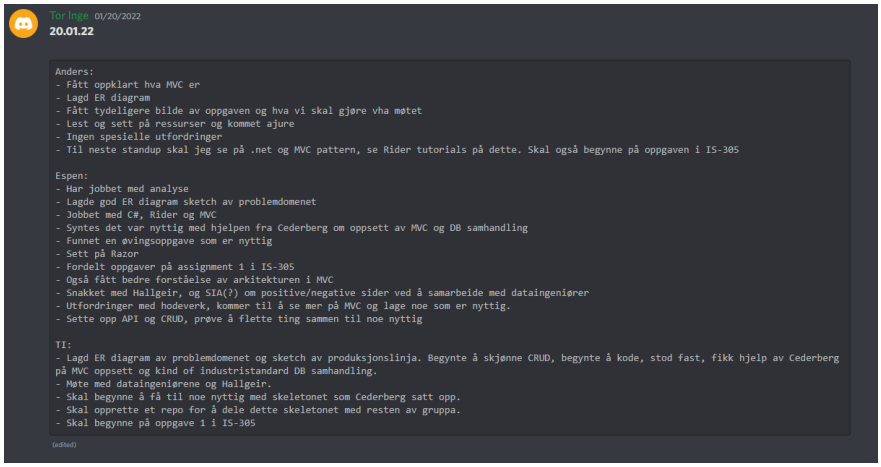
Figur 9: Information



Figur 7: Discord - Stemmekanaler



Figur 8: Discord - Tekstkanaler



Figur 10: Daily Standup

opp hva som er snakket om tidligere, og dele informasjon på en mer effektiv og god måte enn vi ellers hadde klart. Figur 8 viser de ulike tekstkanalene vi hadde i prosjektet. «General» ble brukt til normal chatting mellom gruppemedlemmene. Dette kunne være alt fra spørsmål eller generell informasjon. Vi valgte også å ha kanaler dedikert til ting utenfor prosjektet og også humor. Dette bidro til å senke skuldrene og opprettholde god gruppekjemi. Figur 10 viser hvordan vi logget standup referater. Dette er gjort inne på Discord hvor hvert medlem gikk gjennom de ulike spørsmålene som er nevn i kapittel 3.3.3.

3.5 Programmeringsprosessen

Dette bachelorprosjektet har gitt oss mye verdifull kunnskap og forståelse når det kommer til programmering. Vi fikk lære og bruke et rammeverk og programmeringsspråk som vi tidligere ikke hadde kjennskap til, og sitter igjen med en ny forståelse av hvordan en webapplikasjon er bygd opp. I dette prosjektet var altså kompetansebygging en stor og naturlig del av programmeringsprosessen. Læringen foregikk for det meste ved å bruke tutorials på YouTube og andre ressurser på nettet til å lage øvingsprosjekter. Vår kontaktperson hos Strai, satt i tillegg tidlig opp en sentral tilleggsfunksjon for objekt-databaseorganisering med navn EF Core, som er et rammeverk som tillater oss å jobbe med en database (Microsoft, 2021). Da vi hadde lært og forstått hvordan en webapplikasjon i .NET settes opp og er organisert, et skjelett for applikasjonen var implementert, og rammene for datahenting (som var kjerneoppgaven i dette prosjektet) var satt, var programmeringsprosessen preget av struktur og flyt. Denne flyten over tid når man jobber flere personer på samme skylagrede filer er ikke gitt, men ved hjelp av versjonskontroll og kommunikasjon mellom front- og back-end utvikler ble programmeringsarbeidet mer effektivt og strukturert.

3.5.1 Struktur og kodestandard

Vi har gjennom tidligere utviklingsprosesser gjort i løpet av studiegangen og inspirasjon fra andre prosjekter og ressurser på nettet dannet oss et bilde av hva som kjennetegnes som god kodestandard og er viktig å tenke på under en programmeringsprosess. Med dette bildet i bakhodet har vi fulgt prinsipper og standarder som resulterte i en webapplikasjon som er forholdsvis lett å forstå og bygge videre på ved en eventuell overtakelse.

Vi har fra begynnelsen av programmeringsprosessen hatt fokus på at alle medlemmer i gruppa forstår all koden, og hvis dette ikke har vært mulig grunnet fordeling av arbeid, sørget for å kommentere kode som nevnt i kap. 2.3.4. Tiden det vil ta for eventuelt nye utviklere å sette seg inn i koden mener vi er betraktelig redusert som følge av at vi har lagt inn kommentarer som beskriver en kodesnutts konkrete funksjon.

Vi har gjennom hele programmeringsprosessen, med få unntak, skrevet kode ut ifra konseptene om cohesion og coupling. Man vil ifølge programmeringsportalen geeksforgeeks.org (geeksforgeeks, 2021) ha høy grad av cohesion (at komponenter i koden har én jobb, og kun gjør denne jobben) og lav grad av coupling (at graden av avhengighet mellom komponenter i koden er lav). Som beskrevet nedenfor kunne vi ha modularisert koden enda mer, for å oppnå høyere grad av cohesion, men vi mener at den naturlige organiseringen av et MVC prosjekt med modeller, view-modeller, views og kontrollere, i seg selv legger et godt grunnlag. I tillegg er alle modeller bevisst skrevet til å kun inneholde de attributter som er relevant for *den* modellen. De aller fleste funksjoner står alene under metodeerklæringer og graden av mulighet for gjenbruk er høy som følge av denne relativt høye graden av cohesion. Denne gjenbruken gjøres blant annet mulig via «getters» og «setters», og tilgangsmodifikatorer (eng: access modifiers).

Etter større implementeringer av logikk eller funksjonalitet har vi også, så ofte vi har hatt anledning til, tildelt hverandre review-ansvar – altså at et medlem i gruppa går over et annet medlems kode og vurderer løsningen. Hvis slike code reviews har blitt gjennomført har det ikke blitt fulgt noen mal eller kriterier for kodestandard. Grunnen til det er at vi, som nevnt nedenfor, nedprioriterte strukturerte code reviews. Vi har også stolt på hverandres løsninger hvis f.eks. en funksjonalitet åpenbart har virket og ikke vært spesielt komplekse, og i slike tilfeller droppet code review. Under store deler av programmeringsprosessen har fokuset vært på fremgang, et fokus Strai Kjøkken har vært veldig enig i, og de har dermed aldri tatt opp eller spurt om temaer rundt kodestandard. De har heller aldri gitt oss noen krav for kodestandard. Siden dette prosjektet aldri hadde uoversiktlig mye kildekode ble code review på en måte gjort indirekte ved at vi stadig så på hverandres kode, men da med hensikten å

forstå. Vi ser for oss at strukturert code-review-gjennomføring ville vært mer relevant i en større applikasjon med flere utviklere.

Vi ser også for oss at det samme gjelder for testing, både i form av unit tester og End-to-End tester (UI/UX tester). Vi har helt fra starten, for hver nye implementasjon, vært ivrige på å sjekke om den nye implementeringen funker, og til tider dermed bygget og kjørt appen flere titalls ganger daglig. Den hyppige byggingen sammen med bruk av debugger har gjort at vi til enhver tid har hatt kontroll på hvilken kode som funker som den skal, og hvilken som ikke gjør det. Strukturert skriving av unit tester og E2E tester har dermed blitt nedprioritert, selv om det er en av kjerneprosessene i dagens fremgangsmåte for utviklingsprosjekter.
En refaktorisering av koden kunne hatt noe for seg, men ble aldri tid til å gjennomføre da vi måtte utnytte alle arbeidstimer til å løse oppgaver eller problemer og videreutvikle applikasjonen. For enda høyere grad av lesbarhet og struktur kunne vi ha vært enda flinkere på å modularisere funksjoner, eller med andre ord øke graden av cohesion, og minske graden av coupling. Spesielt en metode i klassen DataFetchService inneholder mange funksjoner for å behandle resultatet av en datahenting som kunne blitt modularisert og overført til klassen DataProcessService. Grunnen til at vi aldri modulariserte denne metoden er nok at det ikke var noen andre steder i koden som trengte eller brukte noen av disse funksjonene og var forståbart for oss. Her burde vi ha tenkt på eventuelt fremtidig arbeid med applikasjonen der disse funksjonene kunne vært relevant å ha for seg selv, og økt graden av cohesion.

3.5.2 Programmeringsspråk og rammeverk

Strai Kjøkken valgte å kreve at vi skulle utvikle appen i ASP.NET CORE med MVC (Model-View-Controller) pattern. For front-end har vi brukt Razor pages, som er en teknologi som muliggjør C# og HTML markup i samme fil (filendelse: .cshtml). I tillegg er det blitt implementert JavaScript for dynamiske elementer og CSS for styling. Databasen vi brukte var en Microsoft SQL Server database, som allerede var organisert og hadde mye data da vi begynte å bruke den. Istedenfor å skrive «prepared statements» i SQL lagret i string-variabler satte Strai Kjøkken opp EF Core (Entity Framework Core). EF Core er en teknologi under .NET miljøet til Microsoft som sammen med klassebiblioteket ApplicationDbContext muliggjør modellering og organisering av data som C# objekter og sørger for sømløs datahenting og -behandling. I tillegg har vi brukt den innebygde debuggeren i Rider flittig, for å stegvis gå gjennom hva som til enhver tid skjer i applikasjonen når vi har stått fast på errors.

I mange situasjoner har det vært nyttig å visuelt se hva variabler og objekter inneholder for å oppdage hvor i applikasjonen det stopper opp ved en runtime-error.
Da vi startet på å implementere rammene for datahenting i applikasjonen begynte vi, grunnet mangel på kunnskap om .NET, med å skrive «prepared statements» og manuell databasekommunikasjon. Hvis vi ikke hadde fått tipset om EF Core av Cederberg hadde vi brukt mye unødvendig tid i ren kodeskriving, hatt betydelig duplisering av kode, og videre en unødvendig lang kildekode med dårlig lesbarhet.

3.5.3 Roller

Under denne programmeringsprosessen var vi en utvikler på backend og en på frontend. Det var flere grunner til at vi valgte å fordele prosjektarbeidet, som nevnt i kapittel 5.2.4, men spesielt for rollefordelingen i programmeringsprosessen var at produktiviteten i en rolle ble drevet av interesse for rollen. Gruppemedlemmet som ble tildelt frontend har mye erfaring og sterk interesse for visuell utforming og UI/UX design, samtidig som backend-utvikler har erfaring med og sterk interesse for objektorientert programmering av logikk og modellering. Rammene for hvert ansvarsområdene var i vårt tilfelle tydelige og enkle å forholde seg til. En tommelfingerregel vi i underbevisstheten brukte var at hvis implementeringen er synlig for brukeren er det frontend arbeid, og hvis den er motsatt er det backend arbeid. Dette er også den typiske fordelingen av back- og front-end arbeid (kilde), en fordeling vi altså utnyttet oss av. Bindeleddet mellom frontend og backend var i vårt tilfelle alle klassene under namespacet «ViewModels». Backend-utvikler skriver logikk som ved en datauthenting mater objekter av ViewModel-klassene med dynamisk data som frontend-utvikler igjen kan jobbe med i et view.
Samarbeidet mellom front- og backend-utvikler har vært avhengig av god kommunikasjon og bruk av versjonskontroll for å lykkes. De har jevnt oppdatert hverandre på nye implementeringer, avtalt endringer eller oppstart på funksjonalitet og tatt opp spørsmål om hverandres tankegang rundt problemstillinger. I ettertid reflekterer de over at det er en svært god erfaring å sitte på etter endt studie å ha opplevd et strukturert samarbeid mellom front- og back-end-utvikler.

3.5.4 Versjonskontroll

Vi har brukt Git sammen med GitHub for versjonskontroll, Git for lokal versjonskontroll og GitHub for versjonskontroll av skylagret kode. Alle commits har blitt pushet til egne

branches, som deretter er gått over av andre medlemmer før de er merget inn i Development-branchen vår. Periodevis når vi har hatt oppdateringer i Development som er betydelige forbedringer og helt stabile har vi igjen merget Development inn i Master, og squashet commits for å få et så ryddig og oversiktlig Git-tre som mulig, slik at vi ikke risikerer å miste noe. Når vi har gjort feil har vi dermed lett kunnet rulle tilbake feilene og korrigere dem, uten å forårsake skade på Master-branchen. Ved tilfeller der to gruppemedlemmer uheldigvis har redigert samme kode i samme fil, kanskje i flere filer, har ikke Github kunnet gjennomføre automatisk merge (sammenfletting av to branches). I slike tilfeller har en av oss måttet pulle de to versjonene som har konflikt til to ulike branches lokalt, manuelt flette sammen koden slik den er ment og så pushe den rette koden opp til base-branchen på Github (development).

4. Produktet

I dette kapittelet tar vi for oss to sider av produktet – Backend og Frontend. Dette gjør vi for å vise viktige valg og viktige deler av koden og utviklingen. Til slutt blir produktet presenter i en video hvor et av gruppemedlemmene går gjennom applikasjonen på video med forklaring med fortellerstemme.

4.1 Backend

I dette kapittelet presenteres sentrale backend-beslutninger og gjennomføringer. Ordet backend referer til det som foregår «under panseret» på applikasjonen, eller med andre ord logikken og prosessene som ikke er synlig for en sluttbruker (TechTerms, u.d.). I denne applikasjonen består backend-delen av datahenting ved kommunikasjon med database, formatering og sortering av data, abstrakt modellering og tilgjengeliggjøring av data for frontend.

4.1.1 ApplicationDbContext og EF Core

Som beskrevet i kapittel 3.5.2 brukte vi EF Core og biblioteket ApplicationDbContext for å hente ut og behandle data. EF Core er en Nuget-pakke som ikke krever noe koding, og er kun ansvarlig for å kommunisere med databasen. ApplicationDbContext er ansvarlig for å sette dataen i kontekst logikk-messig i applikasjonen. Dette gjør den ved å ta i bruk C# klasser under namespacet «entities» (beskrevet i delkapittel 4.1.2). I figur 11 ser vi begynnelsen på en metode som står for å hente og formatere data basert visse kriterier. ApplicationDbContext brukes her for å kjøre en spørring mot databasen ved hjelp av de speilede C#-entitetene. Resultatet av spørringen lagres i en variabel som igjen brukes til å opprette ett objekt av modelltypen Event for hver rad den har hentet ut. Kriteriene er at dato for hendelsen (Eventet) som skal hentes er den samme som brukeren ber om (kommer inn som parameter), og at ProductionId i samme hendelse ikke er lik null (dvs. at hendelsen skal inneholde et skap som har gått igjennom produksjonen).

```
0+1 usages   toringebrekka2
public ProductionEventList FetchEventsWithDate(DateTime date)
{
    //henter en join mellom ProductionEventLog, ProductionEventType og Production der TimeStamp i førstnevnte er lik 'date'
    var productionEvents :List<ProductionEventLog> = ApplicationDbContext.ProductionEventLog // DbSet<ProductionEventLog>
        .Include( navigationPropertyPath: p :ProductionEventLog => p.ProductionTypes) // IIncludableQueryable<ProductionEventLog,...>
        .Include( navigationPropertyPath: p :ProductionEventLog => p.Production) // IIncludableQueryable<ProductionEventLog,...>
        .Where(p :ProductionEventLog => p.TimeStamp.Date == date.Date && p.ProductionId != null) // IQueryable<ProductionEventLog>
        .OrderBy(p :ProductionEventLog => p.ProductionId).ToList();

    List<Event> prodEvents = productionEvents.Select(p :ProductionEventLog => new Event
        {
            Id = p.Id,
            TimeStamp = p.TimeStamp,
            ExtraInfo = p.ExtraInfo,
            ProductionId = (int) p.ProductionId!,
            ProductionType = p.ProductionType,
            EventType = p.EventType
        }) // IEnumerable<Event>
        .ToList();
```

Figur 11: ApplicationDbContext

4.1.2 Entities

```
namespace Straisimulator.Data.Entities;

5 usages  Tor Inge Brekka +2
public class ProductionEventLog
{
    [Key]
    2 usages
    public int Id { get; set; }
    3 usages
    public DateTime TimeStamp { get; set; }
    2 usages
    public string ExtraInfo { get; set; }
    6 usages
    public int? ProductionId { get; set; }

    [ForeignKey(nameof(ProductionId))]
    3 usages
    public Production Production { get; set; }

    3 usages
    public int ProductionType { get; set; }
    [ForeignKey(nameof(ProductionType))]
    2 usages
    public ProductionTypes ProductionTypes { get; set; }

    3 usages
    public int EventType { get; set; }

    [ForeignKey(nameof(EventType))]
    public ProductionEventTypes ProductionEventType { get; set; }
}
```

Figur 12: Entities

Figur 12 viser entiteten ProductionEventLog som er en speiling av databasetabellen med samme navn. Alle tabeller vi bruker i databasen må ha en C#-entitet med samme navn og properties for å kunne bruke ApplicationDbContext. ApplicationDbContext bruker disse entitetene til å hente og lagre data.

For å speile kolonneattributter som primary key og foreign key har vi brukt biblioteket DataAnnotations som kommer med C#. Her kan det nevnes at foreign key-forhold i databasen er gjenspeilet ved å deklarere properties med andre entiteter som typer. Eksempler på dette i figur 12 er propertiene Production og ProductionEventTypes som har andre entiteter som typer. Dette er altså samme foreign key-forhold som i databasen.

4.1.3 Modellering

```
15 usages  toringebrekka2 +1
public class ProductionEventList
{
    3 usages
    public string OrderId { get; set; }
    2 usages
    public DateTime Date { get; set; }
    21 usages
    public List<Event> ProductionEvents { get; set; }
    14 usages
    public List<Event> DrillingEvents { get; set; }
    14 usages
    public List<Event> Fitting1Events { get; set; }
    14 usages
    public List<Event> Fitting2Events { get; set; }
    14 usages
    public List<Event> AssemblyEvents { get; set; }
    4 usages
    public List<Event> OtherOrUndefinedEvents { get; set; }
    3 usages
    public TimeSpan TotalOrderCykelTime { get; set; }
    4 usages
    public TimeSpan TotalOrderQue { get; set; }
    4 usages
    public TimeSpan TotalDayQue { get; set; }
    5 usages
    public TimeSpan TotalDrillingCykelTime { get; set; }
    5 usages
    public TimeSpan TotalFitting1CykelTime { get; set; }
    5 usages
    public TimeSpan TotalFitting2CykelTime { get; set; }
    5 usages
    public TimeSpan TotalAssemblyCykelTime { get; set; }
    10 usages
    public TimeSpan TotalDrillingQue { get; set; }
```

Figur 13: Modellering

I en webapplikasjon skrevet med MVC-pattern er modellene ansvarlig for å representere abstrakte eller faktiske modeller med den hensikt å forenkle logikken i applikasjonen. Modellene kan representere hendelser, objekter, subjekter eller abstrakte komponenter av logikken. Vi har skrevet både abstrakte modeller, som vist i figur 13, og faktiske modeller (f.eks. Event). Modeller som den man ser i figuren er kun til for å samle og sortere flere objekter av modellen Event på ett sted etter en datauthenting. Et objekt av denne modellen brukes i Views via ViewModels. Hvis man ser på den lille grå teksten over hver property er objekter av denne klassen svært hyppig brukt.

4.1.4 Controller

```
anderskorsnes
public IActionResult Statistikk()
{
    return View();
}
anderskorsnes
public IActionResult StatistikkRes(string orderId)
{
    ProductionEventList productionEvents = _dataFetchService.FetchProductionEvents(orderId);
    if (productionEvents.ProductionEvents.Count == 0)
    {
        return View("HentEventLogError");
    }
    else
    {
        StatistikkResViewModel model = new StatistikkResViewModel();
        model.ProductionEventList = productionEvents;
        model.ProductionEventList.OrderId = orderId;
        return View(model);
    }
}
```

Figur 14: Controller

I en MVC-applikasjon er kontrollere bindeleddet mellom Views og Models. Metodene i kontrolleren vår HomeController sørger for å returnere det viewet brukeren ber om på nettsiden. Disse metodene blir altså trigget av brukeraktivitet og kan også sette i gang definerte prosesser i tillegg til å returnere et view. I figur 14 ser man to av metodene i kontrolleren vår som er ansvarlig for statistikk-viewet. Den ene metoden sørger for å vise viewet der brukeren blir bedt om å skrive inn et ordrenummer. Den andre metoden sørger for å indirekte hente og prosessere data basert på parameteret den får av det første viewet og deretter vise resultat-viewet. Dette er mulig gjennom å bruke visse attributter i html-deklarasjonen <form>.

4.2 Frontend

I dette kapittelet presenteres sentrale frontend-beslutninger og gjennomføringer. Ordet frontend refererer til alt på en hjemmeside som brukeren samhandler med (TechTerms, u.d.). Dette vil si designet på grensesnittet og programmeringen som får grensesnittet til å fungere (TechTerms, u.d.).

4.2.1 HTML

Vi har tatt i bruk HTML som markup-språk for å organisere alle de ulike elementene i de forskjellige views vi tar i bruk. Hensiktet med dette er å få kontroll på de ulike elementene slik at man kan ta i bruk CSS på en fornuftig måte, og få lagt på stiler på de spesifikke elementene man ønsker. Figur 15 viser et eksempel på en måte å organisere ulike elementer inn i HTML-tags. Det brukes klassenavn og/eller ID for å kunne henvise til disse elementene i CSS-filen, hvor man legger inn hvordan stil man ønsker å tillegge dette elementet.

```
<div id="features">
  <div class="Wrapper">
    <ul>
      <div class="container-2">
        <a asp-area="" asp-controller="Home" asp-action="Simulator" id="animated-word">Simulering</a>
      </div>
      <div class="container-2">
        <a asp-area="" asp-controller="Home" asp-action="Statistikk" id="animated-word">Statistikk</a>
      </div>
      <div class="container-2">
        <a asp-area="" asp-controller="Home" asp-action="HentEventLog" id="animated-word">Hent køtid i ordre</a>
      </div>
      <div class="clear"></div>
    </ul>
  </div>

</div>
```

Figur 15: HTML

4.2.2 JavaScript

Det har blitt implementert noe JavaScript-kode i presenteringen av diagrammer i webapplikasjonen. Bakgrunnen for dette valget er at det finnes flere gode Chart-biblioteker tilgjengelige som man kan ta i bruk, som er skrevet i JavaScript. Figur 16 viser et eksempel på et kakediagram vi presenterer i webapplikasjonen.

```
<script type="text/javascript">
      google.charts.load('current', {'packages':['corechart']});
      google.charts.setOnLoadCallback(drawChart);

      function drawChart() {

        var data = google.visualization.arrayToDataTable([
          ['Maskin', 'Køtid'],
          ['Borring (sekunder)',     outDrilling],
          ['Evert 1 (sekunder)',      outFitting1],
          ['Evert 2 (sekunder)',  outFitting2],
          ['Pressa (sekunder)', outAssembly]
        ]);

        var options = {
         colors: ['#ffd166', '#89c2d9', '#2a6f97', '#013a63']
        };

        var chart = new google.visualization.PieChart(document.getElementById('piechart'));

        chart.draw(data, options);

      }
    </script>
```

Figur 16: JavaScript

For å kunne presentere denne typen data gjennom JavaScript ble vi derimot nødt til å konvertere C# variabler over til JavaScript-format. Dette ble gjort med kodesnutten vist i figur 17. På denne måten fikk vi tilgang til dataen som opprinnelig var i en C# variabel og dataen kunne brukes i de ulike diagrammene. Dette var helt avgjørende for å få dynamiske diagrammer som kunne vise data i sanntid.

```
// lager JavaScript variabler på de ulike model-variablene
var jsDrilling = '@Model.ProductionEventList.TotalDrillingQue';
var jsFitting1 = '@Model.ProductionEventList.TotalFitting1Que';
var jsFitting2 = '@Model.ProductionEventList.TotalFitting2Que';
var jsAssembly = '@Model.ProductionEventList.TotalAssemblyQue';
var jsTotalQueTime = '@Model.ProductionEventList.TotalOrderQue';

var jsDrillingCyk = '@Model.ProductionEventList.TotalDrillingCykelTime';
var jsFitting1Cyk = '@Model.ProductionEventList.TotalFitting1CykelTime';
var jsFitting2Cyk = '@Model.ProductionEventList.TotalFitting2CykelTime';
var jsAssemblyCyk = '@Model.ProductionEventList.TotalAssemblyCykelTime';
```

Figur 17: JavaScript - Variable

4.2.3 CSS

I starten av prosjektet falt CSS-stilvalget til Bootstrap. Bootstrap er et CSS-rammeverk som er laget for utvikling av responsive nettsider (w3schools, u.d.). Dette gikk vi derimot bort fra og valgte å utvikle CSS-stilen fra bunn av. Grunnen til dette var at vi ønsket å ha 100% kontroll på alle stilvalg og en enklere tilgjengelighet om vi ønsket å forandre på noe i fremtiden.

Figur 18 viser et eksempel på stiloppsettet til en av hovedknappene på hjemmesiden. Det var viktig med god kommentering slik at man enkelt kunne komme frem til det elementet man ønsket å endre på. Dette gjorde at man også kunne søke opp den plasseringen koden lå på.

```
/* Knapper index */

#animated-word {
  margin-top: 50px;
  margin-bottom: 50px;
  padding: 25px;
  letter-spacing: 0.2em;
  font-weight: 600;
  font-size: 28px;
  text-align: center;
  color: #444;
  cursor: pointer;
  max-width: 95%;
  width: 70%;
  outline: 1px solid;
  outline-color: #465460;
  outline-offset: 40px;
  transition: all 300ms cubic-bezier(0.2, 0, 0, 0.8);
}
```

Figur 18: CSS - Stiloppsett

Et annet fokusområde var å gjøre alle sider dynamiske og responsive. Dette vil si at alle sidene til webapplikasjonen kan brukes og er tilgjengelige uansett størrelse på vinduet til nettleseren. Dette ser vi et eksempel på i figur 19, hvor det er lagt inn at navigasjonsbaren til nettsiden skal kollapse inn til en dropdown-meny så fort bredden til vinduet er under 900 piksler. Dette gjør at man får tilgang til alle

```
@media screen and (max-width: 900px) {
  .navbar a  {
    display: none;
  }
```

Figur 19: CSS - Resizing

linker i navigasjonbaren uavhengig av hvilken plattform man er på, og uten at linkene trenger å ta mer plass enn nødvendig med mindre man skal ha tilgang til dem. Dette er illustrert i figur 20 og figur 21.

Figur 20: Navigasjonsbar

Figur 21: Navigasjonsbar - Lukket

4.3 Presentasjon av produkt

Link til gjennomgang av webapplikasjonen: https://youtu.be/rS5Wo4IHNsQ

5. Refleksjon

Det har vært en prosjektgjennomføring proppfull av utfordringer, endringer, uforutsette hendelser, og gevinster. Vi skal i dette kapittelet reflektere over prosjektet under ett, hvordan vi har håndtert uforutsette situasjoner og utfordringer, hvordan vi har endret oss og tilpasset oss, og hvilke gevinster og læringsutbytte vi sitter igjen med i dag.

5.1 Gjennomføring

Det har vært et prosjekt med mange utfordringer, som vi kommer nærmere inn på i neste underkapittel, men også en hel del med endringer og løsninger som har vist seg å være gode for oss. Det vi sitter igjen med er at vi har lært mye, både om temaer som omhandler prosjektet, men også om oss selv og om hverandre. Prosjektgjennomføringen har vært preget av å hele tiden måtte justere og finne ut hvilken metode som vil fungere best, eller hva det er vi gjør som fører til at det ikke går så mye på skinner som vi kanskje har ønsket. Dette har

gjort at vi hele tiden har vært på utkikk etter en forbedret måte å jobbe på, som igjen har ført til at vi i dag sitter med en følelse av å i stor grad ha klart å optimalisere samarbeidet oss imellom. Dette gjenspeiler seg over på produktet vi har klart å komme frem til. Hvor vi på et tidspunkt var redde for hva vi kom til å klare å levere, endte opp med at vi klarte det, sammen med stolthet over å kunne levere noe av verdi. Det er allikevel funksjonalitet vi gjerne skulle ha rukket å få implementert og ferdigstilt, men det produktet vi har i dag er noe vi kan se tilbake på og være godt fornøyde med.

5.2 Utfordringer

Det har underveis i prosjektet dukket om en rekke utfordringer i form av uforutsette hendelser, bosituasjon og generelle utfordringer som satt oss i situasjoner hvor vi var nødt til å gjøre noen valg for å prøve å forbedre veien videre, eller sette oss tilbake i kurs. Vi ønsker gjennom dette kapittelet å reflektere nærmere på noen av de mest avgjørende utfordringene som har formet hendelsesløpet til prosjektet.

5.2.1 Sykdom og livssituasjon

En av de største tidstyvene i dette prosjektet har vært ulike sykdomsfravær og ellers andre hendelser på privatlivet til gruppemedlemmene. Disse hendelsene førte til store avbrudd i progresjon i store deler av spesielt februar. Februar var en svært kritisk periode for oss, da det var denne måneden produksjonslinja var oppe og kjørte igjen, og vi i teorien kunne kommet i gang med utviklingen. Det at vi ble preget av mye fravær og sykdom førte derfor til at vi kom en god del senere i gang enn hva vi hadde ønsket og sett for oss i forkant. Denne perioden har preget resten av semesteret og gjorde at vi måtte se på hvordan vi kunne gjøre endringer for å øke effektivitet og progresjon videre i prosjektet. Det er helt klart den enkelthendelsen/enkeltperioden som har preget vårt gruppearbeid og prosjektarbeid i størst grad. Allikevel er det en utfordring vi ble nødt til å håndtere på en så god måte som mulig, da dette var noe som ikke var i vår makt og ingenting vi kunne gjøre med da det skjedde. Selv om utfallet av dette ble tapt tid og mindre progresjon, tror vi også at det førte til en del endringer i hvordan vi burde fordele arbeidsoppgavene, hvordan vi skulle effektivisere gruppearbeidet, og hvilke mål som var realistiske for fremtiden.

5.2.2 Stopp i produksjonslinjen

Vi visste fra ganske tidlig av at produksjonslinjen ville være nede i en periode da det skulle skje oppgraderinger av maskin. Denne perioden ble allikevel forlenget og medbragte seg en større usikkerhet etter en rekke forsinkelser på ferdigstillingen. Vi var på dette tidspunktet avhengig av å få sett på produksjonslinjen, samtidig som at databasen vi skulle få tilgang på var avhengig av data derfra. Måten vi valgte å håndtere denne perioden på var å utvide kompetansebyggingen. Vi så det som hensiktsmessig å prøve å være så klare som mulig så fort produksjonslinjen var i gang igjen, noe som gjorde at vi ventet med utviklingen av applikasjonen og heller forberedte oss i de ferdighetene vi så for oss ville være mest relevante. Vi tror at forsinkelsen av produksjonslinjen var gunstig for vår utvikling, da vi i retrospekt ser at vi ikke var klare for å begynne med utvikling av webapplikasjon på det tidspunktet produksjonslinjen egentlig skulle stå klar. Dette ble derfor en hendelse som vi prøvde å dra nytte av, og hadde det ikke vært for at vi ble truffet av en lengre periode med sykdom (nevnt i forrige kapittel) så tror vi dette i all hovedsak ville gitt oss et positivt utfall.

5.2.3 Bosituasjon

Vår bosituasjon har definitivt vært med på å forme utførelsen av prosjektet. Vi har to gruppemedlemmer som har måtte jobbe mye hjemmefra, hvor én har jobbet så å si 100% digitalt ved unntak av fysiske oppmøter som har vært helt kritiske for å få en totalforståelse som har vært nødvendig for arbeidet. Det at det i hovedsak har vært én person som har kunnet stille fysisk på kontorene på Strai Kjøkken, har ført til en større utfordring ved det å løse små problemstillinger, være tilgjengelig for Strai Kjøkken, kunne være nede på fabrikken osv. Vi ble som nevnt tidligere derfor tvunget til å finne så gode digitale løsninger som mulig, og i stor grad føler vi at vi har lyktes med dette. Det har allikevel vært situasjoner vi har havnet i som vi tror vi kunne vært foruten om vi hadde vært i en situasjon hvor alle kunne deltatt fysisk til enhver tid. Dette er situasjoner hvor vi har misforstått hverandre, ikke vært tilgjengelige for samarbeid, eller har måttet omprioritere/nedprioritere prosjektarbeidet for situasjoner som har oppstått hjemme. Det er allikevel veldig viktig å poengtere at dette er en situasjon vi som grupper føler vi vokste inn i, og samarbeidet, effektiviteten, og kommunikasjonen ble bare bedre og bedre utover i prosjektet. Det vi ville gjort annerledes var å i større grad være realistiske med tanke på antall fysiske oppmøter vi totalt sett som gruppe ville være i stand til å delta på. Denne forventningen av hvor mye vi trodde vi kunne delta fysisk var med på å skape unødvendig frustrerende øyeblikk. Så større planlegging og mer realistiske mål er to hovedpunkter vi ville gjort annerledes i et annet prosjekt.

5.2.4 Arbeidsfordeling og prosjektstyring

Det var tidlig enighet om at alle skulle lære seg litt om alt. Tanken med dette var at ved å involvere alle i alle deler av prosjektet var veien kort for å diskutere og få hjelp om man satt fast på noe eller ønsket en diskusjon om en problemstilling. Denne arbeidsfordelingen hadde vi også i selve prosjektgjennomføringen. Her skulle alle stille likeverdige og bidra med det som kom naturlig i enhver situasjon. Var det arbeidsoppgaver som gjentok seg, delegerte vi dette ved å rullere på hvem som gjorde hva fra gang til gang. Denne måten å jobbe på har fungert i stor grad, men særlig når kompleksiteten i prosjektet var så lav som mulig. Utover i prosjektet har særlig arbeidsoppgavene i utviklingen blitt fordelt og vi har fått våre respektive roller. Grunnlaget for denne endringer var at vi så et behov får å i større grad øke produktiviteten. Ved å ha en klar rollefordeling på oppgaver som var viktige for at vi skulle kunne levere et godt produkt oppnådde vi nettopp dette med produktivitet og vi så en større arbeidsflyt på grunn av dette. Denne måten å jobbe på har hatt dog sine svakheter. Spesielt ser vi på strukturen i prosjektgjennomføringen som et klart svakhetstegn, da ingen på gruppa er spesielt gode til å strukturere og gjennomføre alt av planlegging og utførelse på en 100% strukturert måte. Dette sees spesielt på hvor konsekvente vi har vært med å dokumentere, holde tider, og andre lignende oppgaver hvor vi kunne fått mer oversikt og bedre kontroll ved å ha en person som fungerte som enten en «Scrum Master» eller en prosjektleder. Vi tror at ved å være tydeligere på rollefordelingene helt fra starten av, kunne vi i større grad opprettholdt en bedre flyt gjennom hele prosjektet og i større grad hatt kontroll på de «administrative» oppgavene et prosjekt medbringer.

5.2.5 Programvare og utviklingsmiljø

Vi måtte tilbringe en stor porsjon av oppstarten og generelt hele prosjektperioden til å tilegne oss ny kunnskap innen både verktøy som brukes, programmeringsspråk og programvare. Dette har helt klart vært med på å forme gjennomføringen av prosjektet og hvordan sluttproduktet ble. Det har gått svært mange timer til å forstå problemer som har vært vanskeligere å løse på grunn av at vi ikke har nok kunnskap innen for eksempel C# eller JavaScript. Ser vi på hele prosjektet under ett er det lite av problemdomenet vi har hatt mye kunnskap til fra før, og det har hele tiden vært fokus på å tilegne seg den nye kunnskapen som har måtte trengtes for å løse den problemstillingen man står i. Denne siden av prosjektet har

både vært utfordrende, men også motiverende og lærerikt når man har løst et problem som har tatt lengre tid å løse.

5.2.6 Kompetansebyggingen

Kompetansebyggingen var svært viktig for vår oppstart og forståelse av problemdomenet. Allikevel ser vi store rom for forbedring på dette feltet, og om vi hadde kunnet gjort det igjen så ville vi gjort flere endringer her. Spesielt gjelder det det å finne de mest relevante kildene, og den riktige kunnskapen man ønsker å tilegne seg. Mye tid har gått til å lære seg noe eller tilegne seg en kunnskap vi i retrospekt ikke fikk bruk for. Samtidig er kompetansebygging et krevende felt, da det er lett å gå i diverse feller. Det var flere utfordringer – Hvordan vite at det man gikk igjennom kom til å bli relevant? Når kunne man si seg ferdig med kompetansebygging? Skal man drive med kompetansebygging gjennom hele prosjektet? Dette var spørsmål vi stilte oss selv, og ser i retrospekt at vi kunne vært enda mer strukturerte i dette arbeidet slik at vi sammen som en gruppe kunne ta beslutninger.
Det vi her kunne gjort annerledes ville vært og satt en enda tydeligere plan for hva det var vi ønsket å komme frem til som sluttprodukt. Ved å vie mer tid til en grundig planlegging tror vi at vi kunne spart oss for mye tid på kompetansebyggingen, og en enda mer konkret forståelse av hva vi måtte kunne, mye tidligere i prosjektet. Samtidig er det viktig å presisere at kompetansebyggingen var viktig for oss, da vi som nevnt tidligere, ikke var kjent med språk og rammeverk.

5.3 Læringsutbytte

Bachelorprosjektet har gitt oss læringsutbytte som strekker seg lengre enn bare tekniske ferdigheter. Vi skal ta for oss hvilke tekniske ferdigheter vi har tilegnet oss, men det er i større grad samarbeidsferdigheter, medmenneskelige ferdigheter, og prosjektgjennomføringsferdigheter som står igjen hos oss av læringsutbytte vi vil få stor nytte av videre inn mot arbeidslivet.

5.3.1 Tekniske ferdigheter

Samtlige gruppemedlemmer har underveis i prosjektarbeidet tilegnet seg en rekke tekniske ferdigheter. Det å utvikle en fullverdig webapplikasjon i nytt programmeringsspråk, med en rekke nye verktøy fører til stort læringsutbytte. Gruppen har lært seg programmeringsspråket C#, med ASP.NET og MVC som rammeverk. I tillegg til dette har det blitt brukt JavaScript,

som også er nytt for gruppen. Å få kunnskap i bruken av både C# og JavaScript er svært gunstig for arbeidslivet da det er språk som er hyppig brukt. Bruken av Trello og dens muligheter for å strukturere prosjektet og fordele oppgaver er også en kunnskap vi verdsetter, da vi så verdien av å struktur og effektivitet dette medførte.

5.3.2 Samarbeidsferdigheter og prosjektgjennomføringsferdigheter

Det er særdeles viktig med et godt samarbeid for å kunne arbeide godt og effektivt gjennom hele bachelorprosjektet. Dette har vært et stort fokus fra første dag ettersom vi visste viktigheten av god kommunikasjon og godt team-arbeid for å gjennomføre på en så god måte som mulig. Bruken av Discord for kommunikasjon, Microsoft Teams for dokumenthåndtering, Zoom og Microsoft Teams for møter, og Trello for struktur og oppgavehåndtering har vært helt essensielt for at samarbeidet har skullet fungere i det hele tatt. Vi visste godt viktigheten av dette før vi startet, men potensialet for hvor mye det kan påvirket et gruppearbeid i negativ eller positiv forstand var noe uvisst.

Det vi har lært gjennom dette prosjektet er hvordan hverandre fungerer som personer, hvordan vi arbeider best, hvordan vi kan samarbeide på en god måte, og hvordan vi skal kommunisere. Dette har vært felt hvor vi i ettertid ser at vi har gått gjennom mange iterasjon for å komme frem til den type samarbeid vi har hatt den siste tiden. Ved å hele tiden være bevisst på hvordan samarbeidet går, hva det er vi har gjort galt og hva vi har gjort riktig, har vi kunnet korrigere oss selv og hverandre for å bli den beste versjonen av oss selv i en gruppe. Det vi derfor så den siste delen av prosjektet var at vi hadde blitt mye mer effektive i både kommunikasjon og forståelse av hverandre og hva som måtte bli gjort. Det tok kortere tid å løse et problem, og samhandlingen og delegeringen av arbeid var mer sømløs enn i starten av prosjektet. Grunnen til dette var at vi i enda større grad enn tidligere i prosjektet var en tydeligere klarhet i kommunikasjonen og fordelingen. På dette tidspunktet hadde vi også større ferdigheter knyttet til utviklingen, så det gikk raskere å komme frem til en løsning. I tillegg var forståelsen vår om oppgaven såpass stor at vi i større grad visste hva som måtte til for å løse respektive oppgaver, noe som skapte bedre flyt i arbeidet.
Denne forståelsen og utviklingen av samarbeidet tror vi at vi vil få stor nytte av i fremtiden. Vi tror at det å være igjennom et prosjektarbeid med utfordringer av mange ulike sorter har gjort oss mer egna til å håndtere prosjektarbeid på en god måte i fremtiden.

5.3.3 Gruppeevaluering

Gruppen er godt fornøyd med det totale samarbeidet. Vi hadde fra før prosjektet lite eller ingen erfaring med å samarbeide sammen, og har gjennom dette semesteret skapt en god gruppedynamikk med forståelse for hverandre. Vi har vært åpne for hverandre, ærlige fra start, og villige til å bli bedre kjent og forstå hvordan vi fungerer som enkeltindivider. Dette har underveis i prosjektet gjort at vi har fått bedre og bedre samarbeid, og et godt samhold i gruppa. Vi kjenner også på en unik blanding av ulike karakterer og interesser, som har gjort at vi har fått en dynamikk som har vært annerledes enn det vi tidligere er vant til. Denne dynamikken tror vi har vært med på å gi oss noe unikt til produktet og prosjektet vi har vært igjennom, noe vi tar med oss videre etter endt prosjektperiode.

5.3.4 Egenevaluering

Tor Inge Brekka

Jeg har i dette utviklingsprosjektet ledet programmeringsarbeidet av webapplikasjonen. Rollen min har hovedsakelig vært backend-utvikler, men jeg har prøvd å bidra på alle områder i appen for å ferdigstille et godt produkt. Hovedansvaret mitt har innebåret å skrive kode for datauthenting, -sortering og -behandling, skrive og organisere abstrakte C# modeller, skrive logikk, og generelt sikre god struktur i en MVC-applikasjon. Jeg har også bidratt så godt jeg har kunnet i gruppa med rådgivning, forklaring og teknisk hjelp når det kommer til programmering og versjonskontroll. Ved merge-konflikter i sistnevnte, altså en merge mellom to gruppemedlemmers kode som ikke kan gjennomføres automatisk av Github, hadde jeg ansvar for å utføre manuelle merges.

Jeg syntes dette bachelorprosjektet har vært et veldig gøy utviklingsprosjekt, men også, kanskje mest av alt, svært lærerikt. Personlig synes jeg det i stor grad var positivt at vi ble utfordret på å lære et helt nytt rammeverk og programmeringsspråk. Jeg har måttet lære og forstå MVC struktur i ASP.NET miljøet, bruken av Entity Framework Core sammen med ApplicationDbContext-biblioteket for datahenting og -sortering, C# modellering og debugging. Jeg har også måttet utvide kunnskapene mine om Git og versjonskontroll, og webutvikling da vi innførte bruken av Razor pages. Samtidig vil jeg poengtere at Anders programmerte websidene, med blant annet alt av stil- og utseendearbeid. Likevel var det viktig at vi til en viss grad forsto hverandres behov i applikasjonen og dermed overlappet kompetansebyggingen litt.

Til slutt vil jeg nevne at jeg er glad for at Espen Cederberg fant tid til rådgivning og veiledning på programmeringsbiten. Hans tips under kompetansebyggingen førte til at jeg fikk lære en moderne og effektiv måte å bruke .NET og MVC, altså tipset om EF Core og utvidelsen av MVC ved å i tillegg bruke ViewModels. Denne kunnskapen ser jeg på som høyst relevant og nyttig i arbeidslivet.

Anders Korsnes

Jeg har i dette utviklingsprosjektet hatt ansvar for frontend-utviklingen av webapplikasjonen. Dette innebærer design og utforming av knapper, layout, og visning av ulike grafer og tabeller. Jeg har utformet CSS-filen fra bunnen av og gjort sidene responsive og fungerende uansett vindustørrelse og potensiell plattform applikasjonen blir vist på. I tillegg har jeg utformet designet på utseendet til webapplikasjonen i sin helhet.
Jeg har i mindre grad bidratt til programmering av backend, men har i noen få tilfeller vært med på noen løsninger der også.
I tillegg til dette har jeg utformet bachelorrapporten og jobbet mye med rapportskriving der. Her har jeg i tillegg bidratt med rettskrivning og bruk av kilder. Jeg har også hatt ansvaret med å ta opp og redigere de ulike statusrapportene som har vært obligatoriske gjennom semesteret. På møter med både mentor og oppdragsgiver har jeg fokusert på å være aktiv og plukket opp det jeg har trengt for å kunne forstå arbeidet videre så godt som mulig.
Jeg har jobbet jevnt og trutt gjennom hele semesteret og prøvd så godt jeg har kunnet med å bidra til å motivere gruppen. Jeg har lært mye, og hatt en bratt læringskurve på det meste av mine hovedansvar. Så langt det har godt så har jeg fokusert på å være tilgjengelig for samarbeid og spørsmål. Dette innebærer at jeg har vært på Discord og sittet i stemmekanalene. Grunnet min bosituasjon så har dette vært viktig for å opprettholde god kommunikasjon hele veien. Jeg har opplevd at jeg har vært en bidragsyter til prosjektet, og jeg sitter igjen med at jeg har gitt prosjektet noe ekstra på mine fokusområder.

Espen Land Stokkeland

I dette prosjektet har jeg blitt litt en alt mulig mann. Jeg har samarbeidet noe med Anders og Tor Inge på deres hovedområder (Frontend og Backend), men ellers har jeg gjort mye av det andre som et prosjekt består av.
Analysen er det i hovedsak jeg som har stått for. Fra å skrive forslag til hvordan koden skal fungere logisk, til kommunikasjon med oppdragsgiver, eller gjennomgang av database og organisering av data. Arbeidet direkte med problemdomenet har jeg stått for alene. De to

andre bor langt borte fra vår oppdragsgiver, og jeg er derfor den eneste som har kunnet møte fysisk i noen betydelig grad. Reelt sett har det blitt fra 2 til 4 dager i uken med fysisk oppmøte, men oftest 4, da onsdager har gått til IS-305 og noen dager har vært på hjemmekontor ved for eksempel sykdom, møter med forelesere, intervjuer eller lignende. RegEx som brukes har jeg laget, og jeg har fått mye av ansvaret relatert til SCRUM (standups osv).
Jeg har jobbet med rapporten, etter Anders hadde laget en struktur for den.
I løpet av hele semesteret har jeg i så stor grad som mulig vært fysisk hos oppdragsgiver, og vært koblet til på stemmekanaler på Discord i kjernetiden (generelt sett mer). Ifølge prosjekt dagboken min har snittet vært i overkant av 7 timer om dagen. Dette inkluderer da ikke rapport arbeid eller kompetansebygging utenom arbeidstiden.
I starten av semesteret tok jeg litt styringen blant medlemmene i dialogen med oppdragsgiveren og veileder. Dette da delvis grunnet fysisk oppmøte. Utover i semesteret har jeg prøvd å ha en bedre balanse, for å få med alle medlemmene sine innspill i like stor grad.
Det har vært mye å ta seg til, fra å lære seg å programmere med .NET 6 og C# til å få bedre kontroll på algoritmer.
Jeg har lært mye, og det som kanskje er viktigst av alt er verdien av god kommunikasjon og hvor mye det betyr å raskt justere hvordan noe gjøres når noe ikke virker helt som det skal. Når vi jobber agilt skal vi ha jevnlige iterasjoner, og det har tidligere vært for lett å knytte dette opp mot produktet, men ikke selve prosessene.
Jeg opplever at mitt bidrag har vært veldig viktig for progresjonen og resultatet i prosjektet. Ellers vil jeg nevne at jeg ikke vet hvor stor andel av det jeg har gjort som jeg hadde klart uten støtte fra andre utenfor gruppen, deriblant Espen Cederberg, andre ved Strai Kjøkken og professorer ved UiA.

Litteraturliste

Arun, R. (2021, September 16). *simplilearn*. Hentet fra What is Trello and How to Use It: https://www.simplilearn.com/tutorials/project-management-tutorial/what-is-trello

atlassian. (u.d.). *atlassian.com*. Hentet fra Getting Git right, with tutorials, news and tips: https://www.atlassian.com/git

Foster, E. (2020, November 9). *guides.lib.berkeley.edu*. Hentet fra How to Write a Good Documentation: Home: https://guides.lib.berkeley.edu/how-to-write-good-documentation

geeksforgeeks. (2021, November 6). *geeksforgeeks.org*. Hentet fra Software Engineering | Coupling and Cohesion: https://www.geeksforgeeks.org/software-engineering-coupling-and-cohesion/

github. (u.d.). *docs.github.com*. Hentet fra Hello World: https://docs.github.com/en/get-started/quickstart/hello-world

glasspaper. (u.d.). *glasspaper.no*. Hentet fra Hva er egentlig Scrum?: https://www.glasspaper.no/artikkel/hva-er-egentlig-scrum/

Karlsen, J. (2017). Prosjekt Ledelse. I J. Karlsen, *PROSJEKT LEDELSE - fra intitiering til gevinstrealisering* (s. 27). Oslo: Universitetsforlaget.

Microsoft. (2021, Mai 25). *docs.microsoft.com*. Hentet fra Entity Framework Core: https://docs.microsoft.com/en-us/ef/core/

opensoruce.com. (u.d.). *opensource.com*. Hentet fra What is open source?: https://opensource.com/resources/what-open-source

ProdcutPlan. (u.d.). *productplan.com*. Hentet fra Scrumban: https://www.productplan.com/glossary/scrumban/#:~:text=Scrumban%20is%20a%20project%20management,agile%2C%20efficient%2C%20and%20productive.

Schwaber, K., & Sutherland, J. (2020). *scrumguides.org*. Hentet fra The 2020 Scrum Guide: https://scrumguides.org/scrum-guide.html#sprint-review

Scrum.org. (u.d.). *Scrum.org*. Hentet fra What is Scrum?: https://www.scrum.org/resources/what-is-scrum

Scrum.org. (u.d.). *Scrum.org*. Hentet fra What is a Scrum Master?: https://www.scrum.org/resources/what-is-a-scrum-master

Scrum.org. (u.d.). *Scrum.org*. Hentet fra What is Scrum?: https://www.scrum.org/resources/what-is-scrum

Smith, S. (2022, Mars 3). *docs.microsoft.com*. Hentet fra Overview of ASP.NET Core MVC: https://docs.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-6.0

Strai Kjøkken. (u.d.). *Strai Kjøkken*. Hentet fra Om Strai Kjøkken: https://www.strai.no/om-strai/

Strai Kjøkken. (u.d.). *Strai Kjøkken*. Hentet fra Vår historie: https://www.strai.no/om-strai/historien/

TechTerms. (u.d.). *techterms.com*. Hentet fra Frontend: https://techterms.com/definition/frontend

TechTerms. (u.d.). *techterms.com*. Hentet fra Backend: https://techterms.com/definition/backend

w3schools. (u.d.). *w3schools.com*. Hentet fra What is Bootstrap?: https://www.w3schools.com/whatis/whatis_bootstrap.asp