Forside

IS-304: 2021

Tittel: Smidig utvikling av utlånssystem

Emnekode	IS-304
Emnenavn	Bacheloroppgave i informasjonssystemer
Emneansvarlig	Hallgeir Nilsen
Veileder	Geir Inge Hausvik
Oppdragsgiver	Egde

Studenter:

Etternavn	Fornavn
Fredriksen	Linn Engedalen
Intaraphasuk	Kantasit
Hedemark	Julie
Erdvik	Magnus
Nergaard	Erik Danielsen
Ramsdal	Marine

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

Forord

Denne rapporten er utarbeidet av studentgruppen Go4IT, og er den avsluttende delen på bachelorstudiet IT og informasjonssystemer ved Universitetet i Agder. I forbindelse med bachelorprosjekt og denne rapporten ønsker vi å takke viktige bidragsytere som har vært vesentlig i gjennomføringen.

Først og fremst vil vi rette en stor takk til Geir Inge Hausvik, som har vært studentgruppens veileder. Under hele prosessen har han vært tilgjengelig for spørsmål, deltatt på styringsgruppemøter samt gitt gruppen gode råd og støtte. Videre vil gruppen takke Hallgeir Nilsen, emneansvarlig i *IS-304 bacheloroppgave i informasjonssystemer*, for at han har skapt en arena for deling og kommunikasjon under Covid-19 pandemien.

Studentgruppen vil takke for samarbeidet med Egde, og ønsker trekke frem og takke en viktig nøkkelperson, Andreas Taraldsen. Han har vært sentral i utviklingsprosjektet gjennom veiledning og tilbakemeldinger, samt satt studentene i kontakt med nødvendige ressurspersoner. I tillegg ønsker vi å takke Robert Quidding og Øyvind Brekkhus Sandåker for gode innspill i utviklingen av systemet. Videre ønsker vi å takke alle i Egde som har stilt sin kunnskap og tid til disposisjon.

Denne rapporten og bachelorprosjektet hadde ikke blitt til uten deres hjelp, tusen takk!

Denne rapporten er utarbeidet av Go4IT

Sted / Dato: Kristiansand / 14.05.2021

Linn E. Fredriksen

Kantasit Intaraphasuk

Julie Hedemark

Magnus Erdvik

Erik D. Nergaard

Marine Ramsdal

Sammendrag

Prosjektgruppen Go4IT har skrevet denne bacheloroppgaven i samarbeid med Egde og Universitetet i Agder. Prosessen startet i oktober 2020 med RefreshIT, hvor UiA la til rette for at samarbeidsbedrifter og prosjektgrupper kunne møtes for nettverksbygging og mingling. Her fikk studenter mulighet til å snakke med potensielle samarbeidsbedrifter, og det var her Go4IT møtte Egde og diskuterte mulighetene for samarbeid. Prosjektarbeidet startet for alvor i januar 2021. Prosjektet gikk ut på å utvikle et utlånssystem for Egdes økende samling av teknisk utstyr, slik at bedriften kunne få en oversikt over hvor utstyret befant seg til enhver tid.

Vi har benyttet oss av Scrum som arbeidsmetodikk. Valget falt naturlig på dette, da gruppen hadde god erfaring med Scrum fra tidligere. Dette er også en metode som tilrettelegger for en agil arbeidsmåte, som gjør at håndtering av endringer underveis forenkles. Dette gjøres ved korte iterasjoner med tydelige delmål og hyppig kontakt med produkteier. På denne måten var vi alltid klare for endringer i krav og prioriteringer for prosjektet. For å kunne forplikte oss til å gi produkteier et system av høy kvalitet, har vi gjennom hele prosessen jobbet med kvalitetssikringstiltak. Disse inkluderte blant annet å sette opp en risiko- og sårbarhetsanalyse (ROS), sette faste standarder og regler for koding og prosjektgjennomføring, samt testing av funksjonalitet og en regelmessig fremvisning av systemet for produkteier.

Gruppen har hatt et stort læringsutbytte under hele prosessen, hvor vi har tilegnet oss verdifull erfaring om hvordan det er å jobbe med reelle kunder. Vi har benyttet oss av kunnskapen vi har tilegnet oss på studiet, men også lært mye nytt. Rent teknisk har vi lært nye programmeringsspråk og systemløsninger, og teoretisk har vi lært mye om kvalitet og kvalitetssikringstiltak. Videre har vi også lært å benytte kunnskapen vi har tilegnet oss på en selvstendig måte ved å hele tiden reflektere over valg som er tatt, og om det eventuelt kunne vært gjort på en bedre måte.

Prosjektet har resultert i en webapplikasjon hvor ansatte i Egde kan booke bedriftens utstyr ved å logge seg inn med sin Office-konto. Systemet forenkler prosessen hvor ansatte skal låne utstyr, og reduserer muligheten for forvirring og rot. Man kan som ansatt se hvor utstyret befinner seg, og om det er ledig eller opptatt i en gitt periode. Det har vært fokus på å fullføre systemkravene med høyest prioritering, men også flere av kravene listet som «Should Have» er implementert, blant annet brukervennlighet. Gruppen er derfor svært stolt over å presentere sluttproduktet, som holder høy standard og lever opp til forventningene produkteier har hatt. Her er linken til ferdig system: https://youtu.be/e923RFv-m2o

Innholdsfortegnelse

1 INTRODUKSJON 7
2 ANALYSE 8
2.1 DATAINNSAMLING 8
2.2 SYSTEMKRAV 8
2.3 BRUKERHISTORIER 9
2.4 RIKT BILDE 11
2.5 ENTITY RELATIONSHIP MODEL 12
2.6 KLASSEDIAGRAM 13
3 DESIGN 14
3.1 ARKITEKTONISK DESIGN 14
3.1.1 Prioritering av kriterier 14
3.1.2 Sekvensdiagram 15
3.2 BRUKERGRENSESNITTDESIGN 16
3.2.1 Skisser 16
3.2.2 Navigasjonskart 17
3.2.3 Designprinsipper 18
3.2.4 Mockups 19
3.2.5 Prototype 21
3.2.6 Systemet 22
4 SPRÅK OG VERKTØY 22
4.1 AZURE DEVOPS 22
4.2 RAMMEVERK – ASP.NET CORE 22
4.3 SYSTEMARKITEKTUR – RAZOR PAGES 23
4.3.1 Backend 23
4.3.2 Frontend 23
4.4 IDE - VISUAL STUDIO 23
4.5 DATABASEVERKTØY 24
4.6 VERSJONSKONTROLL 24
5 IMPLEMENTERING AV UTLÅNSSYSTEM 25
5.1 BACKEND 25
5.1.1 Entity Framework 25
5.1.2 CRUD 26
5.1.3 Razor Pages 26
5.1.5 Docker 26
5.1.6 Microsoft Azure AD 27
5.2 FRONTEND 27
5.3 TESTING 28
6 PROSJEKTGJENNOMFØRING 29
6.1 SCRUM 29
6.1.1 Sprint Planning 30
6.1.2 Daily stand-up 30
6.1.3 Sprint Review 30
6.1.4 Sprint Retrospective 30
6.1.5 Hvorfor passer Scrum i dette prosjektet? 30
6.2 ROLLER I SCRUM 32
6.3 PRODUCT BACKLOG 34
6.4 SPRINTER 35
6.4.1 Pre-Sprint 35
6.4.2 Sprint 1 36
6.4.3 Sprint 2 39

6.4.4 Sprint 3 ... *43*
6.4.5 Sprint 4 ... *45*
6.4.6 Sprint 5 ... *48*
6.5 TIDSSTYRING ... 50
6.5.1 Tilgengelige ressurser ... *50*
6.5.2 Estimering ... *51*
6.5.3 Burn down chart ... *52*
6.5.4 Oversikt ... *53*
6.6 RISIKOHÅNDTERING ... 54
6.6.1 ROS-analyse ... *54*
6.6.2 Kodestandard ... *55*
7 REFLEKSJON OG KONKLUSJON ... 56
7.1 KONKLUSJON ... 56
7.2 REFLEKSJON ... 56
REFERANSER ... 58
VEDLEGG ... 60
VEDLEGG 1 UTTALELSE FRA EGDE ... 60
VEDLEGG 2 EGENVURDERING ... 61
VEDLEGG 3 ANDRE RELEVANTE VEDLEGG ... 63
Vedlegg 3.1 Styringsgruppemøte 1 ... *63*
Vedlegg 3.2 Styringsgruppemøte 2 ... *65*
Vedlegg 3.3 Styringsgruppemøte 3 ... *67*
Vedlegg 3.4 Brukerhistorier ... *69*
Vedlegg 3.5 Versjonskontroll ... *70*
Vedlegg 3.6 Burn down chart for hele prosjektperioden ... *71*
Vedlegg 3.7 Tabeller fra Sprint Retrospective: Hva har gått bra – Sprint 2 ... *73*
Vedlegg 3.8 Tabeller fra Sprint Retrospective: Hva har gått galt – Sprint 2 ... *73*
Vedlegg 3.9 Tabeller fra Sprint Retrospective: Hva har gått bra – Sprint 3 ... *74*
Vedlegg 3.10 Tabeller fra Sprint Retrospective: Hva har gått galt – Sprint 3 ... *74*
Vedlegg 3.11 Tabeller fra Sprint Retrospective: Hva har gått bra – Sprint 4 ... *75*
Vedlegg 3.12 Tabeller fra Sprint Retrospective: Hva har gått galt – Sprint 4 ... *75*
Vedlegg 3.13 Tabeller fra Sprint Retrospective: Hva har gått bra – Sprint 5 ... *76*
Vedlegg 3.14 Ros-analysen med kvalitetssikringstiltak. ... *77*
Vedlegg 3.15 Risikomatrise Sprint 1 ... *78*
Vedlegg 3.16 Risikomatrise Sprint 2-3 ... *78*
Vedlegg 3.17 Risikomatrise Sprint 3-4 ... *79*
Vedlegg 3.18 Risikomatrise Sprint 4-5 ... *79*
Vedlegg 3.19 Kodestandard/Dokumentasjon ... *80*
Vedlegg 3.20 Gruppekontrakt ... *86*

Figurliste

Figur 1: Rikt bilde 11
Figur 2: E/R-modell..... 12
Figur 3: Klassediagram 13
Figur 4: Sekvensdiagram..... 16
Figur 5: Utdrag av skisser. 17
Figur 6: Navigasjonskart. 18
Figur 7: Egdes farger..... 19
Figur 8: Mockup..... 19
Figur 9: Navigasjonsbar 20
Figur 10: Gjenkjennbar posisjon på Book-knappen..... 20
Figur 11: Hover-effekt på knappen 20
Figur 12: Prototype..... 21
Figur 13: E/R-modell, implementert database..... 25
Figur 14: CRUD-prinsipper..... 26
Figur 15: Prosjektets scrum team med interessenter. 32
Figur 16: Tilgjengelige ressurser..... 50
Figur 17: Estimerte, tilgjengelige og registrerte timer 51
Figur 18: ETC..... 52
Figur 19: Burn down chart Sprint 5..... 53
Figur 20: Estimerte, tilgjengelige, BDC og DevOps timer 54

Tabell

Tabell 1: Systemkrav..... 9
Tabell 2: Brukerhistorier. 10
Tabell 3: Prioritering av kriterier..... 15
Tabell 4: Product backlogen..... 34
Tabell 5: Hva har gått bra, Sprint 1 38
Tabell 6: Hva har gått galt, Sprint 1 39
Tabell 7: Hva har gått bra, Sprint 2 42
Tabell 8: Hva har gått galt, Sprint 2 42
Tabell 9: Hva har gått bra, Sprint 3 45
Tabell 10: Hva har gått galt, Sprint 3 45
Tabell 11: Hva har gått bra, Sprint 4. 47
Tabell 12: Hva har gått galt, Sprint 4. 48
Tabell 13: Hva har gått bra, Sprint 5. 50

1 Introduksjon

Våren 2021 leverer prosjektgruppen Go4IT sitt bachelorprosjekt i IT og informasjonssystemer ved Universitet i Agder. Go4IT er en tverrfaglig gruppe bestående av seks studenter med ulik bakgrunn og ferdigheter. Go4IT består av medlemmer som har jobbet sammen hele bachelorløpet, samt noen nye medlemmer. Det har derfor vært viktig for gruppen å bli kjent med hverandres styrker og svakheter, samt lage en bindende gruppekontrakt. Grunnet Covid-19 har arbeidet med prosjektet og rapporten hovedsakelig foregått digitalt. Dette inkluderer samarbeid internt i studentgruppen, møter med eksterne parter, samt veiledninger og forelesninger.

I prosjektets startfase hadde studentene et ønske om å kunne jobbe med et reelt prosjekt fra næringslivet. Med reelt menes det at arbeidet studentene utfører tas i bruk og gir verdi til bedriften. Under Refresh IT 22. oktober 2020, en samling hvor mulige oppdragsgivere presentere ulike bachelorprosjekter for studentene, presenterte Egde at de ønsket et system for å håndtere utlån av utstyr, da de trengte et system hvor ansatte kunne låne utstyr til privat eller profesjonelt bruk. På daværende tidspunkt manglet de et slikt system. Det ble presentert at studentene selv kunne velge løsninger for systemet og presisert at dette er et system de sårt trengte og ville ta i bruk.

Egde er Sørlandets ledende fagmiljø innen digitalisering og teknologi (Egde, u.å, a). Bedriften har i dag 87 medarbeidere fordelt på tre kontorer i Grimstad, Kristiansand og Porsgrunn. Ved å kombinere Egde sin kompetanse innen strategi, brukerverdi og teknologi med kundens erfaring, hjelper Egde kundene med å realisere sine muligheter. Gjennom digitale tjenester og opplevelser gjør Egde en forskjell for virksomheter, mennesker og samfunn (Egde, u.å, b). Bedriften har en rekke kunder fra ulike bransjer som statlige og kommunale sektor, olje- og gass, helse og finans, og forskning. Dette gjør at Egde er i stand til å hente inspirasjon og kunnskap på tvers av bransjen for å løse kundenes problemer (Egde, u.å, a).

Prosjektgruppen mente Egde sin problemstilling virket utfordrende og spennende, og etter presentasjonen møttes Go4IT og Egde, hvor det ble inngått en avtale om at studentgruppen skulle utvikle et system som håndterer utlån av bedriftens utstyr. Egde presenterte følgende mandat for Go4IT:

«Egde trenger et system hvor det registreres nyinnkjøpte eller gammelt teknisk utstyr. Systemet må kunne registrere hvilken ansatt som har hvilket utstyr. Løsningen må fungere på Egdes Grimstad-kontor og Egdes Kristiansand-kontor. Brukeren skal selv ha mulighet til å registrere et utlån.» - Andreas Taraldsen, 23.11.2021.

Rapporten redegjør for sentrale avgjørelser tatt under prosjektperioden, viser studentgruppens utlånssystem, samt beskriver systemutviklingsprosessen. Første del av rapporten omhandler modeller som er utarbeidet i forbindelse med analyse og design, etterfulgt av en presentasjon av det endelige systemet. Videre redegjøres det for valg av rammeverk og implementasjon. Neste del handler om prosjektstyring og gjennomføring, samt risikohåndtering. Avslutningsvis presenteres en konklusjon og en refleksjon rundt arbeidet som er blitt gjort.

2 Analyse

I dette kapittelet presenteres og redegjøres analysen som er gjort for å tilegne seg kunnskap om hvordan utlånssystemet skal fungere. Dette innebærer utarbeidelse av brukerhistorier, med kriterier og argumenter, skisse av rikt bilde, systemkrav, samt Entity Relationship Model og klassediagram.

2.1 Datainnsamling

Ved oppstart av prosjekter er det nødvendig med noen form for data. Dataene som samles inn, blir prosjektets byggesteiner og er bakgrunnen for det som skal utarbeides. Dataene sier noe om hvem som er målgruppen, hvordan systemet kommer til å skulle brukes, samt noen av funksjonalitetene systemet må inneha. Når det kommer til innsamling av data, finnes det flere måter å gjøre dette på. Det vanligste er å oppsøke målgruppen for systemet og enten stille spørsmål i form av intervjuer eller spørreundersøkelser. På denne måten vil man kunne lage en profil basert på gjennomsnittsbrukeren, som kan bidra til at systemet blir best mulig for sluttbrukeren (X. Vasilakopoulou, forelesning, 29. august 2018).

Intervjuer brukes til å samle inn informasjon ved å gjennomføre en samtale med retningslinjer, i et «spørsmål og svar»-format. Det er spesielt viktig å få et inntrykk av intervjuobjektenes meninger og tanker, slik at man best mulig kan gjengi innholdet og forstå personen man intervjuer (Kendall & Kendall, 2002).

Gruppen foreslo for Egde i pre-sprint å gjennomføre intervjuer med de ansatte for å tilegne seg kunnskap og innsikt om hva de synes om dagens måte å låne utstyr på, samt tanker og ønsker for et nytt utlånssystem. Etter en diskusjon med produkteier ble det bestemt at gruppen ikke skulle gjennomføre slike intervjuer da kravspesifikasjonene skulle gi gruppen god nok forståelse av funksjonaliteter bedriften skal ha i sitt utlånssystem.

Dette gir rom for risiko da vi i mindre grad får en åpen dialog med fremtidige sluttbrukere og derfor ikke kan garantere at sluttbruker får den beste løsningen. Likevel ble en slik risiko godt kompensert for ved gjennomføring av sprint review, hvor sluttbruker var til stede, samt oppfordring til åpen dialog med bedriften via produkteier. Videre kunne intervjuer eller tester av systemet være aktuelt på et senere tidspunkt.

Vi mener at en regelmessig oppfølging med åpen diskusjon med alle rollene til stede (se kapittel 6.2 Roller), samt mulighet for å gjennomføre brukertester på sluttbruker etter hvert, kan fungere godt nok selv om det har vært et fravær av intervjuer.

2.2 Systemkrav

Systemkravene i dette prosjektet er utarbeidet av Egde, og er listet opp i Tabell 1. Disse var prioritert av bedriften på forhånd og gruppen har fulgt disse kravene og prioriteringene. På bakgrunn av de forskjellige kravene, har gruppen utarbeidet brukerhistorier (se kapittel 2.3) som inngår i hvert av kravene. Dette er gjort for å kunne konkretisere arbeidsoppgaver, slik at gruppen kan være sikre på at arbeidet som utføres er i henhold til planen og samtidig følger forespørsel fra Egde. Ut ifra brukerhistoriene er det igjen lagt inn underoppgaver som gruppen tar for seg i de forskjellige sprintene. I Tabell 1 er kravene som er merket med «Must Have»,

krav som må være på plass for at systemet kan leveres til produkteier, og er det som defineres som minst brukbare produkt (MBP).

Tabell 1: Systemkrav

Systemkrav	Prioritering	Brukerhistorie
Systemet må være selvbetjent for brukerne	Must Have	1. Som ansatt i Egde, ønsker jeg enkelt å booke bedriftens utstyr ved behov, slik at jeg kan benytte meg av dette til arbeids- eller fritidsformål.
Systemet må ha en form for sikker innlogging/registrering	Must Have	2. Som bruker, vil jeg kunne logge inn i systemet med personlig O365 bruker, slik at systemet er sikkert og ingen utenforstående kan booke utstyret til bedriften.
Systemet må fungere på Egdes hovedkontorer	Must Have	3. Som bruker, vil jeg kunne booke utstyr og levere uavhengig av kontor, slik at jeg kan spare tid og ressurser.
Systemet må lagre informasjon om produkt og bruker	Must Have	4. Som bruker ønsker jeg å kunne se en oversikt over mine bookinger, slik at jeg kan se hvilket utstyr som er booket til hvilken tid.
Systemet bør gi brukeren mulighet til å se hvem andre som har et spesifikt produkt	Should Have	5. Som bruker, ønsker jeg å få oversikt over hvem som har utstyr til enhver tid, slik at jeg slipper å lete og spørre rundt.
Systemet bør ha et enkelt brukergrensesnitt	Should Have	6. Som bruker vil jeg kunne filtrere utstyr etter forskjellige kriterier. 7. Som bruker vil jeg kunne søke etter utstyr ved utstyrsnavn 8. Som bruker ønsker jeg et intuitivt og brukervennlig utlånssystem, slik at booking kan skje så lett som mulig.
Produktet bør gi administrator og ansatte muligheten til å booke et produkt frem i tid	Should Have	9. Som bruker/admin, ønsker jeg å booke utstyr fremover i tid, slik at jeg kan holde av spesifikt utstyr til spesifikke formål.
Administrator for systemet bør kunne legge til, endre og fjerne utstyr og filtreringskategorier.	Should Have	10. Som admin vil jeg kunne legge til, redigere og slette både utstyr, lokasjon og kategori.
Systemet bør ha god performance.	*Should Have*	11. Som produkteier ønsker jeg et system med god performance, for å slippe for mye vedlikehold og oppgraderinger etter overtakelse.

Systemkravene ble så brukt videre til å utarbeide brukerhistorier, som vist i høyre kolonne i Tabell 1. Her tolket gruppen systemkravene basert på skriv fra Egde og muntlig kommunikasjon med produkteier. Dette var verdifullt, da en del kommer frem gjennom samtale og kroppsspråk som kanskje ikke kommer like tydelig frem ved skriftlig kommunikasjon. Hvordan brukerhistoriene ble utarbeidet, kommer frem i kapittel 2.3.

2.3 Brukerhistorier

Da systemkravene var analysert og tolket, ble brukerhistorier utarbeidet. Brukerhistorier er et kort utsagn om en bedrift eller hverdagslivet til sluttbruker i et programvaresystem, som fanger essensielle deler av arbeidet en bruker gjør eller må gjøre med systemet. Mange teknikker som spiller en viktig rolle gjennom agil utvikling, bygger på brukerhistorier. Brukerhistorier benyttes for å styrke kommunikasjon mellom utviklere og sluttbrukere og kan hjelpe til med å konkretisere funksjonaliteter for et system under utvikling (Dimitrijević, Jovanović, & Devedžić, 2015).

Tabell 2 viser et utdrag av de ulike brukerhistoriene som er brukt som utgangspunkt i dette prosjektet. Fullstendig tabell finnes i Vedlegg 3.4 Brukerhistorier. Hver brukerhistorie er basert på et systemkrav som er blitt presentert for prosjektgruppen av samarbeidsbedriften. Alle brukerhistoriene følger samme standard for format: «Som X, ønsker jeg Y, for å oppnå Z» (Cohn, 2004). Videre beskrives hver historie slik at nødvendige funksjonaliteter blir åpenbare, og disse kommer enda tydeligere frem under «kriterier» hvor det listes opp hvilke funksjonaliteter som behøves for å oppnå målet beskrevet i brukerhistorien.

Tabell 2: Brukerhistorier.

ID	Brukerhistorie	Beskrivelse	Kriterier	Argument	Prioritet
1	Som ansatt i Egde, ønsker jeg enkelt å booke bedriftens utstyr ved behov, slik at jeg kan benytte meg av dette til arbeids- eller fritidsformål.	Mulighet for at alle ansatte, på egenhånd, kan benytte seg av systemet for booking av utstyr.	Systemet skal ha en funksjon som lar ansatte booke Egdes utstyr til personlige, eller profesjonelle formål. Når brukeren gjennomfører handlingen og booker et produkt, skal systemet reservere produktet for gitt tidspunkt, oppdatere tilgjengelig beholdning og legge til bookingen i brukerens «Historie».	Denne funksjonen rangeres som et «Must Have» fordi den fanger selve essensen av hva systemet skal kunne gjøre. En aktiv bruker må kunne booke et produkt for at systemet skal kunne regnes som brukbart. Uten denne funksjonen vil ikke systemet være tilfredsstillende for produkteier eller ansatte.	*Must Have*
2	Som bruker, vil jeg kunne logge inn i systemet med personlig O365 bruker, slik at systemet er sikkert og ingen utenforstående kan booke utstyret til bedriften.	Mulighet for at ansatte kan logge inn i systemet. Innlogging via Office365 for å sikre at kun ansatte har tilgang.	Ved ankomst til nettsiden, blir ansatte henvist til innloggingsfunksjon til Office365. Ansatt skriver inn mailadresse og brukernavn og dette verifiseres før brukeren sendes til hjemmesiden for booking. Dersom innloggingen ikke eksisterer, vil ikke brukeren få tilgang til systemet.	Funksjonen for innlogging er et Must Have, fordi dette systemet kun skal kunne benyttes av ansatte hos Egde. Mennesker utenfor bedriften skal ikke ha mulighet til å kunne booke Egdes utstyr.	*Must Have*
3	Som bruker, vil jeg kunne booke utstyr og levere uavhengig av kontor, slik at jeg kan spare tid og ressurser.	Mulighet for å kunne bruke systemet uavhengig av lokasjon.	Systemet skal fungere uavhengig av hvor bookingen skjer. Brukeren går inn på nettsiden og booker utstyr uten at lokasjon er en avgjørende faktor.	Denne funksjonen er prioritert som «Must Have» fordi alle Egdes ansatte skal kunne ha muligheten til å booke utstyr uavhengig om de jobber i Kristiansand eller Grimstad.	*Must Have*
4	Som bruker ønsker jeg å kunne se en oversikt over mine bookinger, slik at jeg kan se hvilket utstyr som er booket til hvilken tid.	Informasjon om brukeren og utstyr lagres i databaser slik at ansatte enkelt kan finne informasjonen de søker etter.	Når brukere logger inn, lagres informasjon om brukeren i systemet slik at historikk om tidligere bookinger enkelt kan uthentes. Når utstyr legges til i beholdningen lagres også informasjon om dette slik at man enkelt kan booke det flere ganger.	Denne funksjonen er et «Must Have» for at det skal være et tilfredsstillende produkt. Dette fordi det gjør systemet mer brukervennlig noe som er definert som et kvalitetskrav av kunden.	*Must Have*

I kolonnen «Argument» i Tabell 2, argumenteres det for hvorfor funksjonene er nødvendige for å oppnå MBP. Brukerhistoriene er prioritert ved hjelp av MoSCoW-metoden. MoSCoW-metoden er en prioriteringsteknikk som kan brukes innen prosjektledelse og programvareutvikling for å sørge for at det er enighet og at forventninger fra stakeholders blir

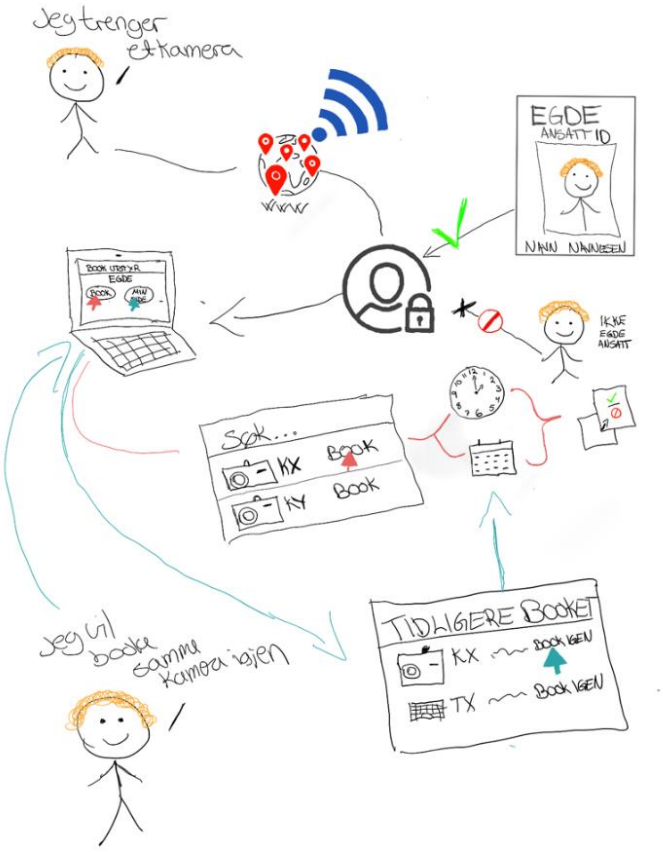
møtt på en tilfredsstillende måte. Alle krav i et prosjekt er viktige, men disse bør prioriteres slik at de viktigste kravene blir fokusert på først (Wikipedia, 2021).

Brukerhistoriene for funksjonene som klassifiseres som «Must Have» er nødvendig for at produktet skal tilfredsstille minstekravene. «Should Have»-kravene er også listet opp i den fullstendige tabellen (Vedlegg 3.4). Dette er funksjonaliteter systemet bør ha, men som ikke er nødvendige. Da brukerhistoriene var ferdig utarbeidet, startet arbeidet med å bruke informasjonen vi hadde samlet og analysert, og sette det i kontekst ved hjelp av figurer og modeller.

2.4 Rikt bilde

En av figurene som ble utarbeidet ut ifra brukerhistoriene i kapittel 2.3, var det rike bildet (Figur 1). Brukerhistorien som lyder følgende: «*Som ansatt i Egde, ønsker jeg enkelt å booke bedriftens utstyr ved behov, slik at jeg kan benytte meg av dette til arbeids- eller fritidsformål.*», er den som skinner sterkest gjennom i bildet, men også brukerhistorien som omhandler innlogging (Tabell 2, ID 2) er integrert i det rike bildet.

Et rikt bilde skal organisere og begrunne all informasjonen en bruker benytter seg av. Rike bilder utarbeides vanligvis i etterkant av intervjuer på arbeidsplassen, slik at man kan få et så helhetlig bilde av situasjonen som mulig. Med et helhetlig bilde blir det dermed lettere å visualisere prosessen senere. Det rike bildet skal vise de primære stakeholders, eller interessenter, for produktet, hvordan de handler og tenker (Monk & Howard, 1998).



Figur 1: Rikt bilde

I vårt tilfelle er det kun sluttbruker som er avbildet, da det er disse som skal benytte seg av systemet og som blir stakeholders. Produkteier vil ikke ha noen egen plass i systemet, men vil fungere som sluttbruker når systemet er implementert. I det rike bildet (Figur 1) har vi ønsket å visualisere brukerreisen av systemet, ved å sammenfatte alle brukerhistoriene inn i ett bilde.

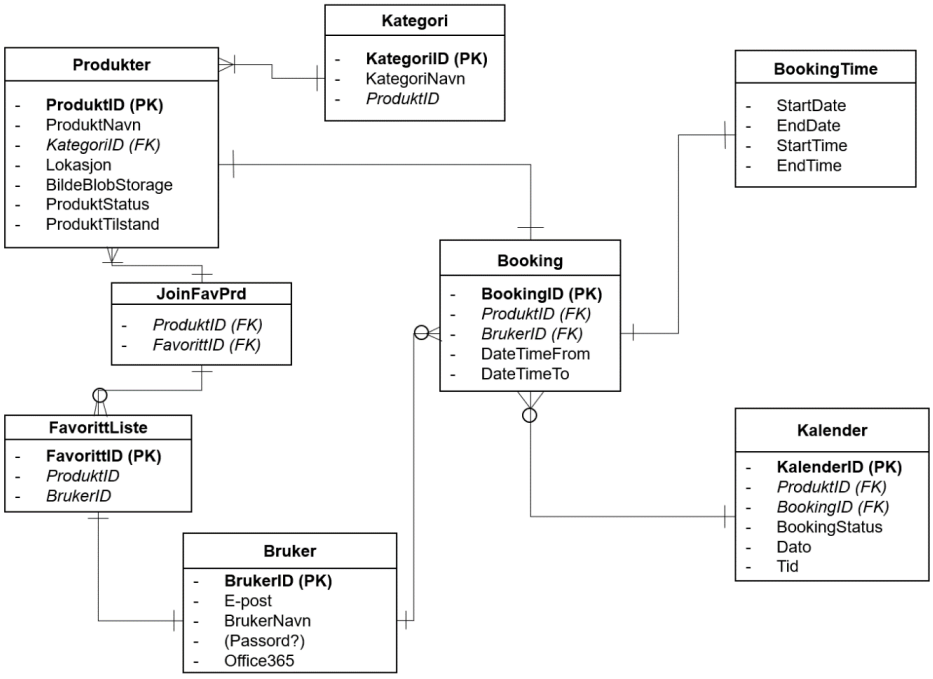
Det rike bildet beskriver hvordan en Egde-ansatt kan booke et kamera. Videre illustrerer pilene rekkefølgen på de neste trinnene i prosessen. Brukeren går inn på websiden til utlånssystemet ved å logge inn, søker etter ønsket kamera, velger dato og klokkeslett og bekrefter bookingen. Videre kan vi se at samme ansatt ønsker å booke det samme kameraet igjen ved en senere anledning, og kan gå rett inn på siden og finne en oversikt over tidligere booket utstyr. Her kan brukeren igjen velge tidspunkt og dato før bookingen bekreftes.

Dette var noe av det første som ble utarbeidet da gruppen mottok kravspesifikasjonene. Det rike bildet hjalp gruppen å visualisere hvordan systemet skulle fungere og hva som var målet med systemet. Dermed kunne vi lettere jobbe målrettet med utarbeidelsen av systemet.

2.5 Entity Relationship Model

Entity Relationship Model (E/R-modell) er en modell som illustrerer strukturen i en database. Modellen består vanligvis av entiteter med tilhørende egenskaper, samt forhold mellom de ulike entitetene (Wikipedia, 2020).

I Figur 2 er entiteter illustrert ved de ulike boksene, med egenskaper som underpunkter. Linjene mellom boksene forklarer relasjonen mellom entitetene. De ulike relasjonene er en til mange, en til en, en til en eller mange, og en til null eller mange forhold. Utarbeidelse av E/R-Modell var en viktig del av prosjektets innledende fase, fordi det gir oversikt over hva systemets database skal bestå av, samt at det gir en veiledning i forhold til videre arbeid av systemet. I tillegg kan vi gjennom utarbeidelsen bestemme hvordan man ønsker å konstruere systemet, for eksempel hvilke filtyper databasen måtte støtte (Lucidchart, u.å).

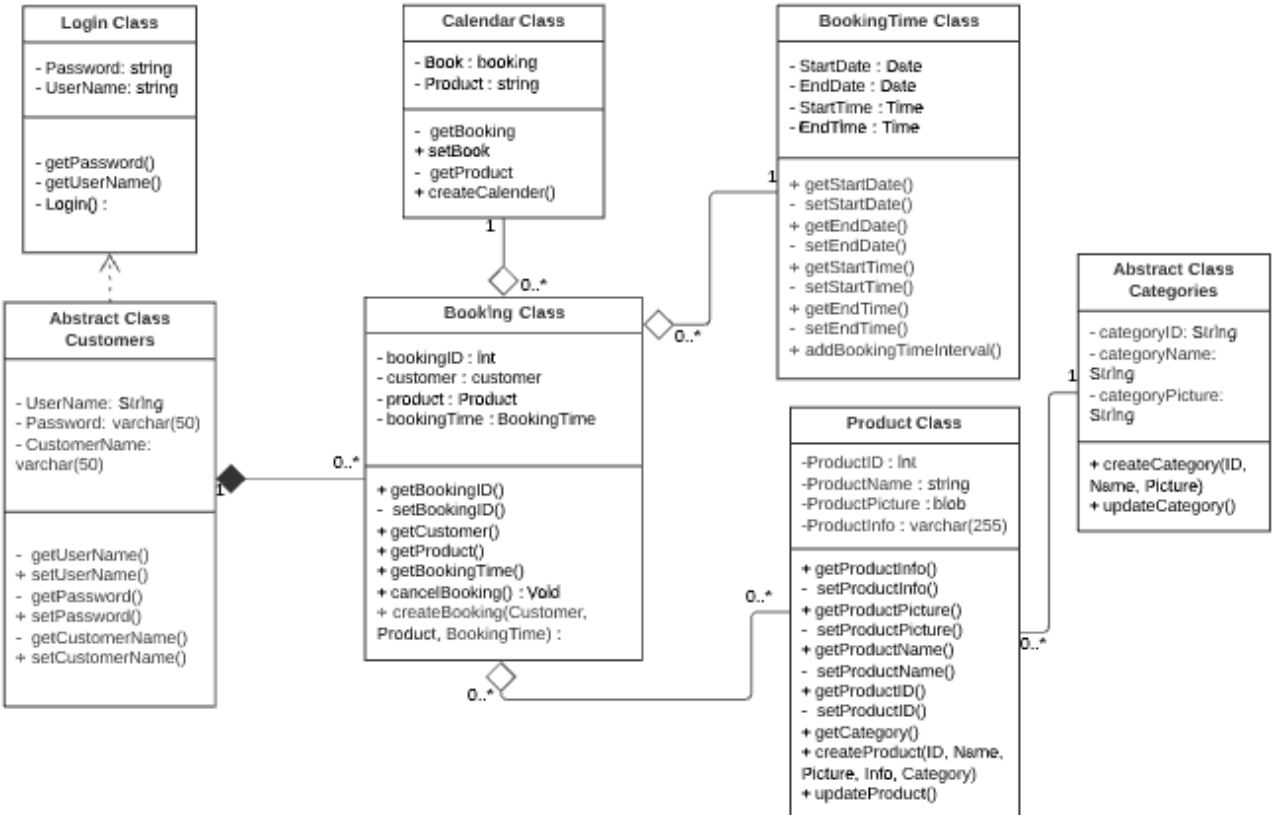


Figur 2: E/R-modell

E/R-modellen, vist ved Figur 2, har følgende entiteter: produkter, bruker, kategori, booking, bookingTime, JoinFavPrd, FavorittListe og kalender. Felter merket med (PK) er primærnøkler og mens felter merket med (FK) er fremmednøkler. E/R-modellen ble videre brukt til å utarbeide klassediagram.

2.6 Klassediagram

Da E/R-modellen var ferdig utarbeidet startet gruppen med utarbeidelse av klassediagrammet. Et klassediagram beskriver en samling av klasser og strukturelle relasjoner, og er en utvidelse av E/R-modellen. Klassediagrammet utarbeides gjerne i analysedelen for å gi videre innsikt i hvordan systemet kan utarbeides senere (Mathiassen, Munk-Madsen, Nielsen & Stage, 2018, s. 255). I Figur 3, er klassene illustrert med bokser, der øverste seksjon av hver boks viser klassens navn, den midterste seksjonen viser klassens felter med tilhørende datatype, og nederste seksjon viser klassens konstruktør og metoder.



Figur 3: Klassediagram

Utarbeidelse av klassediagrammet ga gruppen innsikt i hvordan vi ønsket at systemets elementer skal bygges opp og hvordan de er relatert til hverandre. For eksempel kan vi se klassen «booking» har ansvar for booking av produkter. Den lagrer informasjon om hvem som låner, hva som lånes, og til hvilken tid. Et annet eksempel er forholdet mellom kategori og produkt, der produkter kun kan ha en kategori, mens en kategori kan ha null eller mange produkter. Klassediagrammet er bygget på E/R-modellen, hvor mange av klassene er de representativt tilsvarende entitetene i E/R-modellen.

3 Design

Parallelt med analysedelen jobbet gruppen også med design og designkriterier. Dette var en del av «fundamentet» i Azure DevOps, hvor grunnarbeidet ble gjort før vi startet arbeidet på systemet. Her har vi tatt for oss hvilke kriterier som er viktige for vårt system. Det er utarbeidet sekvensdiagram og navigasjonskart for å forstå gangen i systemet, i tillegg til at vi har utarbeidet mockup basert på individuelle skisser og designkriterier.

3.1 Arkitektonisk design

Et systems arkitektur skal møte både systemkrav og designkriterier, hvor disse setter standarden for hvordan systemet skal fungere. Ved å definere og prioritere de kriteriene vi ønsker av systemet, forenkles utviklingsprosessen ved å eliminere potensiell usikkerhet.

3.1.1 Prioritering av kriterier

En prioriteringsliste sørger for god forståelse for hva som skal til, og hva som er viktig for at et system skal være vellykket. Et godt arkitektonisk design skal ikke inneha store mangler. For utarbeidelsen av Tabell 3 har vi hentet inspirasjon fra Object Oriented Analysis & Design (Mathiassen et al., 2018, s. 187). Prioriteringstabellen viser de forskjellige kriteriene vi ønsker å legge størst vekt på i vårt system, der målet er at det skal være et enkelt og forståelig sluttprodukt. Sluttproduktet skal møte forventninger fra både produkteier og sluttbruker.

Kriteriene rangeres fra «Veldig viktig» til «Irrelevant», hvor rangeringen baserer seg på systemkravene fra produkteier og brukerhistoriene vi har utarbeidet. Den høye rangeringen av *brukervennlighet*, *fleksibilitet* og *forståelighet* grunner først og fremst i systemkravene (Tabell 1), men også i valg av utviklingsrammeverk og språk som redegjøres for i kapittel 4. Brukervennlighet handler om systemets evne til å møte forventningene til brukerne og at det fungerer på de tekniske plattformene bedriften benytter. Med fleksibilitet menes det at systemet skal være driftseffektivt ved å kreve få ressurser ved endringer og vedlikehold (Mathiassen et al., 2018, s. 180). Fleksibilitet er spesielt viktig da systemet skal overleveres til produkteier og muligens videreutvikles av andre utviklere. Disse kriteriene er derfor rangert som «Veldig viktig».

Kriteriene vi har valgt å kategorisere som «viktig» er: *enkelt å forflytte*, *enkelt å teste*, *enkelt å vedlikeholde*, *pålitelig* og *sikkerhet*. Med *enkelt å forflytte* menes det at systemet ikke skal behøve store endringer for å fungere på flere plattformer, eller at selve koden enkelt kan gis videre. Et krav kan for eksempel være at et system skal fungere like godt på PC som på andre bærbare enheter. Utlånssystemet vi utvikler for Egde skal overleveres, og det er dermed viktig at dette kan gjøres på en god og enkel måte for begge parter.

Da systemet kun skal tas i bruk internt, er ikke sikkerhet satt som «Veldig viktig», men heller «Viktig». Vi ønsker å kunne ta i bruk Office365 som innloggingsmetode slik at alle ansatte hos Egde kan ta i bruk tjenesten på en enkel måte ved hjelp av Single-Sign-On (SSO). Ved bruk av SSO vil det være slik at kun ansatte hos Egde vil kunne ta i bruk tjenesten. Systemet skal kun brukes internt, og det vil derfor ikke være nødvendig å legge så stor vekt på effektivitet, da

systemet er såpass enkelt. Ved å ta i bruk et rammeverk som Egde er kjent med fra før, vil ikke vedlikehold kreve mye ressurser.

Tabell 3: Prioritering av kriterier.

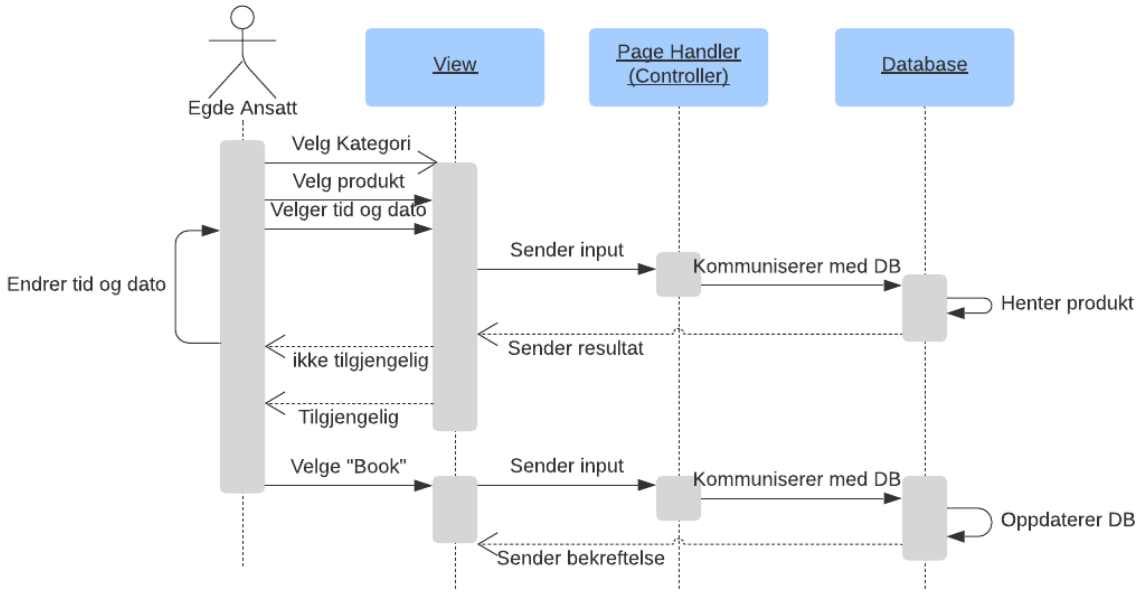
Kriterier	Veldig viktig	Viktig	Mindre viktig	Irrelevant
Brukervennlig	X			
Effektivt			X	
Enkelt å forflytte		X		
Enkelt å teste		X		
Enkelt å vedlikeholde		X		
Fleksibelt	X			
Forståelig	X			
Pålitelig		X		
Sikkerhet		X		

De prioriterte kriteriene la dermed grunnlaget for en rekke av modellene som ble utarbeidet, blant annet sekvensdiagram.

3.1.2 Sekvensdiagram

Da designkriteriene var prioritert, kunne vi sette disse i kontekst ved å utarbeide diagrammer og modeller. Dette startet med sekvensdiagram (Figur 4). Sekvensdiagrammer benyttes for å beskrive forskjellige scenarioer for hvordan systemet kan brukes. Diagrammet kan beskrive deler av, eller hele systemet og logikken bak hvert steg i prosessen i kronologisk rekkefølge. Sekvensdiagrammer modellerer den logiske flyten i systemet i en visuell fremstilling, som kan gjøre det lettere for design og analyse-team å skape en felles forståelse med et backend-team (Ambler, 2004, s. 197). Sekvensdiagrammer viser hvordan objekter i et system, eller klasser i kode, interagerer med hverandre.

De viktigste kriteriene i kapittel 3.1.1 Prioritering av designkriterier, var at systemet skulle være brukervennlig, fleksibelt og forståelig. Kriteriet som omhandlet fleksibilitet, ble hovedfokuset da sekvensdiagrammet skulle utarbeides. Dette ved å si noe om hvordan systemets backend skal håndtere forskjellige input og hva slags output som skal sendes tilbake til bruker. Dette er en forenklet måte å visuelt presentere hva systemet gjør, og er bidragsyter ved utvikling da man kan se hvordan systemet skal respondere. I tillegg er en del av kriteriene som er kategorisert som «Viktig» med her. Blant annet at det skal være enkelt å teste. Dette ved at man enkelt kan se de forskjellige komponentene i systemet og forespørsel/respons gis. Man kan da i ettertid enkelt se hvor man bør teste funksjonalitet.



Figur 4: Sekvensdiagram

I Figur 4 illustreres hvordan en aktør (Egde Ansatt) interagerer med systemet. Boksen merket med «View» er det aktøren kan se (frontend), og de to andre boksene (Page Handler og Database), er det som skjer backend i systemet.

Aktøren i Figur 4 får mulighet til å velge ut produkt og dato (de tre øverste pilene), før det sendes en input som kommuniserer med databasen. Produktet hentes så ut, systemet sender en tilbakemelding basert på tilgjengelighet av produktet, og bookingen bekreftes eller avkreftes av systemet som sendes som output til aktøren. Dersom bookingen avkreftes, ser man at aktøren har mulighet til å endre dato og tid, for å prøve igjen. Om produktet er tilgjengelig kan aktøren velge å booke. Systemet sender deretter input som kommuniserer med databasen. Databasen oppdaterer tilgjengeligheten på produktet, og aktøren får bekreftelse på at produktet er booket. Denne sekvensen kan i stor grad bidra til at man ser en sammenheng i hvordan systemet skal reagere på input, og hvilke outputer som skal sendes tilbake til bruker, basert på om forskjellige kriterier oppfylles eller ikke.

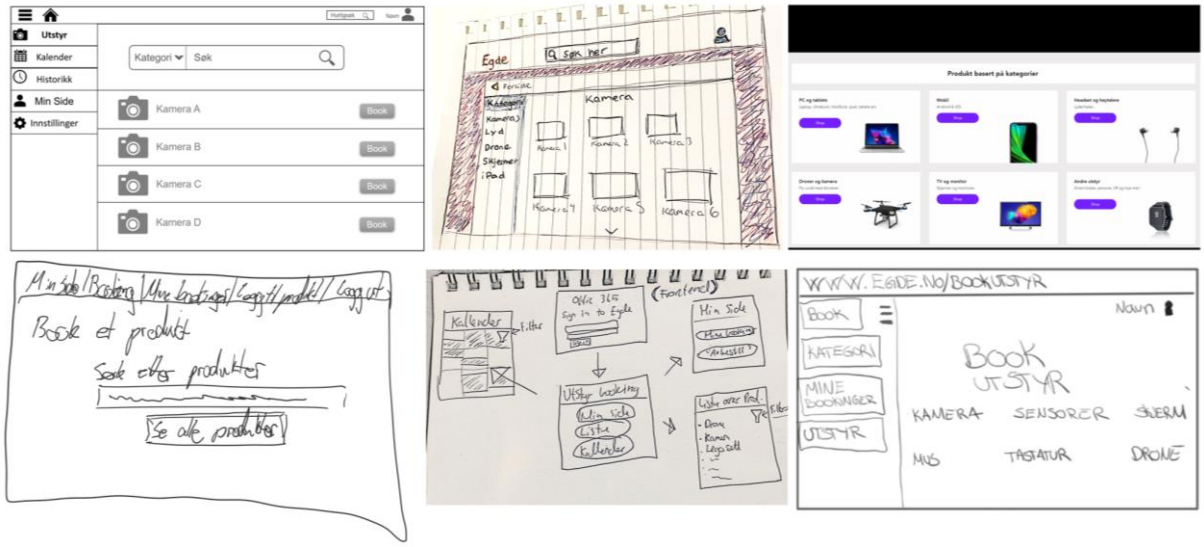
3.2 Brukergrensesnittdesign

Fordi vi utvikler et system for Egde som skal benyttes av deres ansatte, ønsket begge parter fra start at systemet skulle benytte lignende designelementer som på deres nettside. Designvalgene er resultater av ønsker, tilbakemeldinger, standarder fra Egdes merkevarebok, og Benyons (2014) designprinsipper.

3.2.1 Skisser

I innledende fase av prosjektet, startet gruppen med å utarbeide skisser basert på analysen og kravspesifikasjonene (Figur 1). Dette for å skape en felles forståelse og enighet innad i gruppen om hvordan systemet skulle se ut. Denne prosessen startet med at hvert gruppemedlem laget skisser individuelt (Figur 5) på den måten de selv ønsket, deretter diskuterte vi styrker og

svakheter i hverandres skisser. Videre samlet vi de elementene fra hver skisse som fungerte best, og utarbeidet en mockup for systemet (kapittel 3.2.4 Mockups).



Figur 5: Utdrag av skisser.

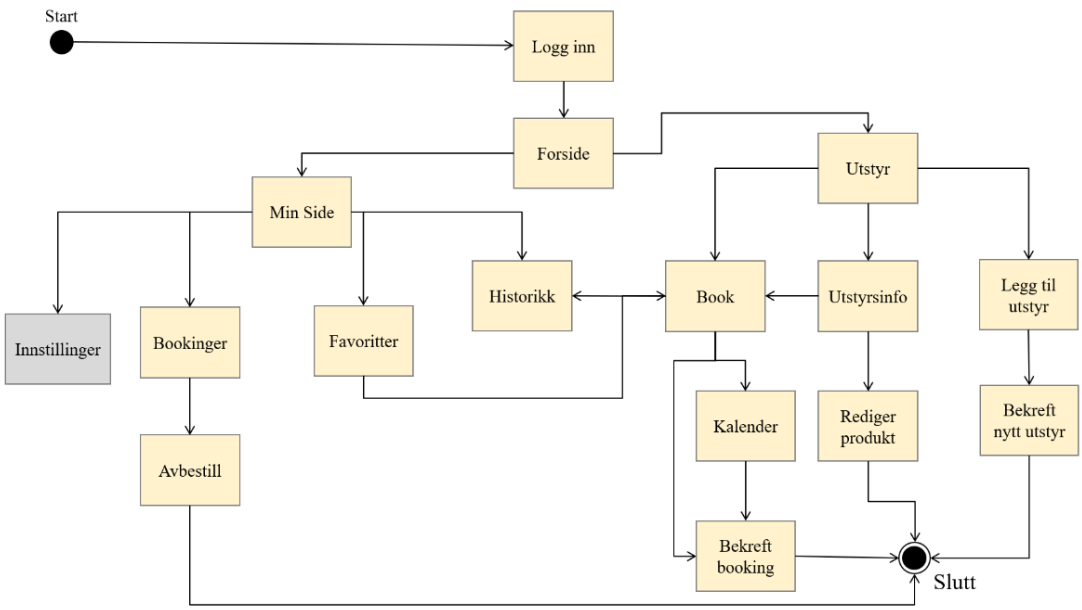
Skissene ble deretter brukt til å utarbeide et navigasjonskart. Navigasjonskartet ble enklere å visualisere når man hadde skisser og en felles forståelse for hvilke sider som skulle være med og hvordan disse skulle henge sammen.

3.2.2 Navigasjonskart

Navigasjon er en viktig del av ethvert system, og et navigasjonskart bidrar til å finne ut av hvordan mennesker navigerer et system eller en applikasjon. Målet er å finne ut av hvordan brukere opplever systemet slik at brukervennlighet er en del av prosessen helt fra start. Dette er et av kriteriene som er kategorisert som «Veldig Viktig» i kapittel 3.1.1. Brukervennlighet kommer frem i navigasjonskartet ved at det har vært fokus på at systemet skal henge sammen på en logisk måte, som skal være intuitiv for sluttbruker. Man skal som bruker forstå hvorfor systemet sender brukeren til de forskjellige sidene fra landingssiden og i en rekkefølge som gir mening. Systemet bør kunne kommunisere en kronologisk historie som brukeren kan delta i, og dette kommer frem i navigasjonskartet ved at man kan se hvilke sider som fører hvor. Det er heller ingen «blindveier» i systemet, slik at en bruker alltid kan komme seg videre eller tilbake.

Hver side i webapplikasjonen representeres med en boks (Figur 6), og pilene sier noe om hvilke sider man kan nå fra hver enkelt side (Benyon, 2014, s. 173).

Det ble tidlig tatt et valg om å inkludere et navigasjonskart, og Figur 6 er et utkast som ble laget i første sprint. Dette skulle være et slags fundament for utvikling av både backend og frontend. Dette fordi det skulle gi en enkel oversikt over hvordan systemet henger sammen og man kan lett se hvordan de forskjellige sidene skal avgrenses.



Figur 6: Navigasjonskart.

Navigasjonskartet ble utarbeidet med grunnlag i det rike bildet som igjen var basert på brukerhistoriene. Det rike bildet ga en visuell fremstilling av hvordan en bruker kunne benytte seg av systemet, noe som gjorde det lettere å utarbeide en systematisk oversikt over navigasjonen. I tillegg ble sekvensdiagrammet brukt for å bidra til økt forståelse for den kronologiske rekkefølgen i systemet. Navigasjonskartet ble så tatt i bruk videre for å utarbeide mockups.

3.2.3 Designprinsipper

Benyons (2014) 12 designprinsipper kan deles inn i tre kategorier: *learnability, effectiveness* og *accommodation.* Learnability handler om tilgang til og kunnskap om systemet. Det skal ikke være vanskelig å ta i bruk systemet, og brukeren skal ikke behøve å sette seg inn i det hver gang det benyttes. Effectiveness handler om systemets enkelthet. Brukeren skal ikke være redd for å gjøre feil og det skal heller ikke være rom for at brukeren kan gjøre feil som kan påvirke systemets evne til å fungere på riktig måte. Accommodation handler om å møte ulike brukeres behov (Benyon, 2014, s. 86-87).

De fem designprinsippene vi har valgt å fokusere på i dette systemet er *navigation, feedback, consistency, familiarity* og *affordance.* Navigation handler at det skal være enkelt for systemets brukere å finne frem til den informasjonen de trenger, samt å navigere mellom systemets ulike sider. Feedback er tilbakemeldinger om brukerens aktivitet og handlinger. Et system som responderer på en brukers handlinger, vil gi økt forståelse av systemet. Consistency handler om at designelementer skal samsvare gjennom hele systemet. Farger, skrifttype, knapper og generell utforming bør samsvare på de ulike sidene. Familiarity vil si å anvende et språk og en utforming som er kjent for brukeren. Gjenkjennelighet øker systemets produktivitet. Affordance handler om egenskapene til et element, og at elementets utseende stemmer overens med de egenskapene det er tiltenkt. Et godt eksempel her er at en knapp ser ut som en knapp (Benyon, 2014, s. 86-87).

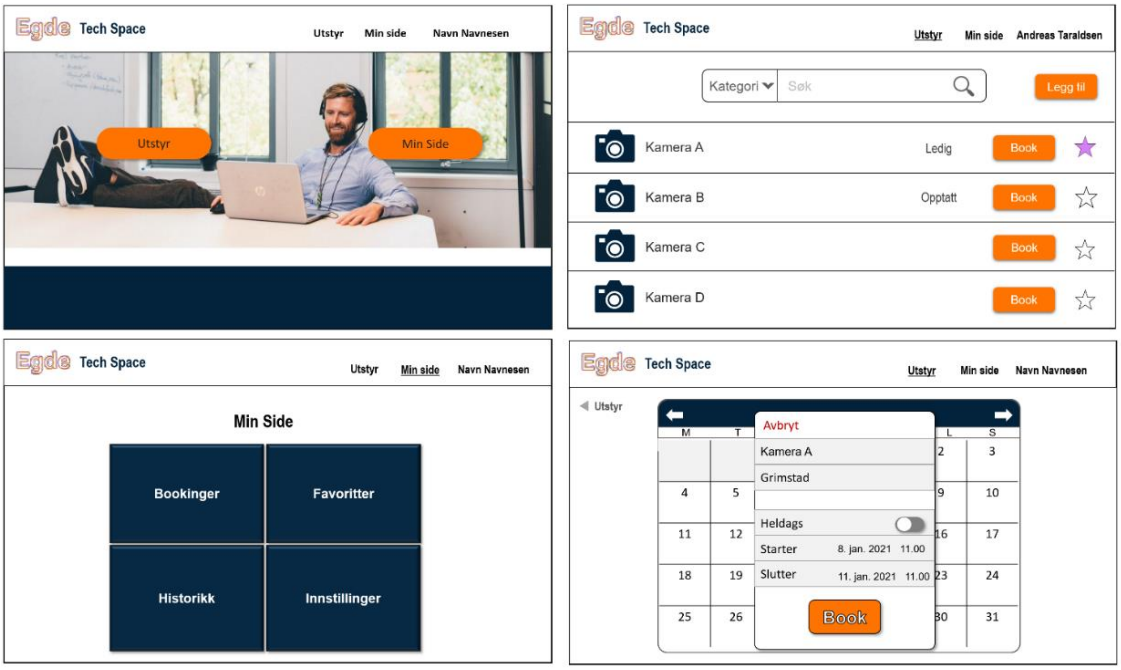
3.2.4 Mockups

Mockups brukes for å reflektere design- og fargevalg, layout, visuell navigasjon og atmosfæren til produktet. En nøye utarbeidet mockup kan bidra til å skape en felles forståelse for produktet slik at både utviklere og produkteier er enige om hva som forventes av produktet. Det gir også mulighet for produkteier å kunne gi tilbakemeldinger tidlig i prosessen, slik at man kan gjøre endringer raskt for å starte med best mulig utgangspunkt når programmeringsfasen igangsettes (Cao, 2020).

Da gruppen skulle utarbeide første mockup (Figur 8), ble dette gjort med brukerhistoriene, navigasjonskartet og designprinsipper/-kriterier i bakhodet. Det ble også tatt inspirasjon fra Egdes egne hjemmesider for å etterligne farger og estetikk, samt atmosfæren Egde ønsker å skape når man besøker nettstedet deres. Gruppen fikk tilsendt en merkevarebok hvor en del av designvalgene allerede var tatt, både med tanke på farger og sideoppsett.

Figur 7: Egdes farger

Figur 7 viser fargene fra Egdes merkevarebok, som vi benyttet oss av ved utforming av systemet. Ved å følge merkevareboken, har vi satt standarden for en rekke designelementer. Hovedfargen på bakgrunn er hvit og primærfarger på eksempelvis knapper er blå og oransje. Bruken av disse fargene tar hensyn til de som er fargeblinde/fargesvake, da de står i kontrast til hverandre. I tillegg har knappene en skriftfarge som står i kontrast til fargen på knappene, og er svart på oransje knapp og hvit på blå knapp (Benyon, 2014, s. 276).

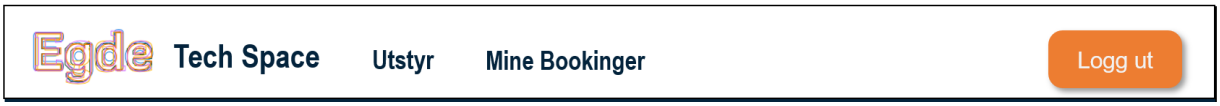


Figur 8: Mockup

Både de arkitektoniske designkriteriene og designprinsippene til Benyon (2014) lå til grunn ved utarbeidelsen av mockups (Figur 8). Brukervennlighet og forståelighet er noe av det viktigste for vårt system.

Vi ønsket at systemet skulle samsvare med kriterier og prinsipper, slik at sluttresultatet stod til forventningene, både for oss, produkteier og sluttbruker. Figur 8, som presenterer en mockup av systemet, illustrerer noen av disse prinsippene og kravene. Brukervennlighet kommer frem ved at knapper er store og tydelige, eksempelvis i ruten øverst til høyre. Knappene er også merket med tekst for tilgjengelighet (eks. nederst til venstre, Figur 8), og har en farge som skiller seg fra bakgrunnen. Dette kan knyttes opp mot Benyons (2014) prinsipper om affordance og consistency ved at knapper og tilhørende handlinger er tydelige, og at den generelle utformingen i systemet følger samme standard.

Et brukervennlig system skal passe brukerens behov og preferanser, samtidig som det skal være enkelt og kjent. Med kjent mener vi å spille på designkomponenter som er vante og hyppig brukt i ulike systemer. Designprinsippet familiarity handler om nettopp dette. For at et system skal være brukervennlig, mener vi det er viktig at brukerne ikke skal bruke lang tid på å lære seg å bruke systemet. Et smart og fint design har lite å si dersom brukeren ikke forstår hvordan systemet skal brukes.



Figur 9: Navigasjonsbar

Vi sørger for familiarity i vårt system ved å aktivt ta i bruk Egdes merkevarebok og deres nettside for inspirasjon. Videre er «Logg ut»-knappen plassert til høyre i navigasjonsbaren, som vist i Figur 9. Prinsippet om navigation fremkommer også i navigasjonsbaren (Figur 9), som illustrerer enkel navigasjon i systemet, hvor man har tilgang til alle systemets hovedsider uavhengig av hvor i systemet en bruker befinner seg.

Figur 10: Gjenkjennbar posisjon på Book-knappen

Funksjonelle knapper som eksempelvis «Book», er plassert til høyre for skriften, slik som illustrert i Figur 10. En slik utforming av brukergrensesnittet vil være gjenkjennelig for brukeren fra lignende webapplikasjoner.

Videre har vi også benyttet prinsippet som omhandler feedback. Et eksempel på dette vil være at knapper endrer tilstand når de markeres, som vist i Figur 11 (Sandnes, 2018, s 50). Vi velger å kalle dette en «hover-effekt». Her ser vi at knappen endres når man holder

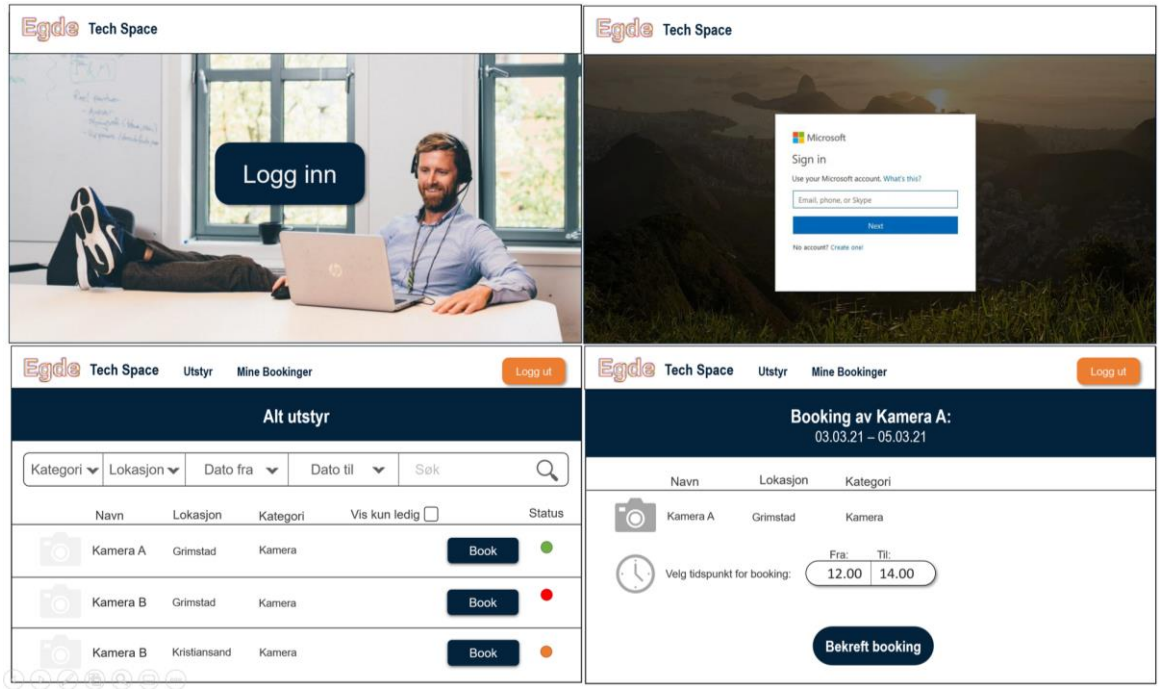
Figur 11: Hover-effekt på knappen

musepekeren over den, noe som gir tilbakemelding til brukeren om at systemet responderer på brukerens handlinger (Benyon, 2014, s. 87).

For å øke forståelighet og brukervennlighet for brukere av et system, er det viktig å være konsekvent i designet. Dette gjelder for både backend og frontend. Vår klikkbare mockup (Figur 8) er konsekvent i både layout, fargevalg, og generell utforming av knapper og tekst. Da gruppen hadde utarbeidet et førsteutkast av systemets visuelle fremstilling i form av mockup (Figur 8), ble denne presentert på Sprint Review 1. Tilbakemeldingene ga oss gode innspill slik at mockupen kunne tilpasses produkteiers og sluttbrukers behov og ønsker. Disse tilbakemeldingene var svært verdifulle for gruppen, særlig med tanke på besparelse av tid og ressurser ved eventuelt dobbeltarbeid.

3.2.5 Prototype

I Sprint 2 ble mockupen i Figur 8 videreutviklet til en klikkbar prototype som vist i Figur 12. Her ble tilbakemeldingene fra Sprint Review 1 tatt i bruk. Det ble blant annet nevnt at «Min Side» (Figur 8, nederst til venstre) var overflødig, og denne ble så fjernet da prototypen ble utarbeidet. Det ble også uttrykt et ønske om å søke etter flere kriterier enn kun kategori, og dette ble også med videre til prototypen (nederst til venstre, Figur 12). Det ble også endret noe på farger av knapper, basert på funksjonalitet. På denne måten fikk knapper som tilførte noe til systemet blå bakgrunn med hvit tekst, og knapper som fjerner noe fra systemet oransje bakgrunn med svart tekst.



Figur 12: Prototype

Et endringsvalg som ble foretatt senere i designprosessen var hvilke sider som skulle ha en navigasjonsbar. Hvis vi her ser tilbake på systemets mockup, som presenteres ved Figur 8, så kommer det frem at enkelte av sidene inkluderer en navigasjonsbar. Ved det endelige systemet har vi valgt å inkludere navigasjonsbaren på samtlige sider, med unntak av om brukeren ikke

er innlogget. Dette for å fremme prinsippet om navigation og familiarity, da dette ble oppfattet som mest hensiktsmessig, gitt systemets oversiktlighet og effektivitet ved bruk.

Prototypen ble så utgangspunktet for systemet som ble utviklet, både med tanke på frontend-utvikling ved utforming av sider og backend-utvikling.

3.2.6 Systemet

Video av utlånssystemet med forklaring finnes her: https://youtu.be/e923RFv-m2o

4 Språk og verktøy

I dette kapittelet redegjøres det for språk og verktøy. Dette innebærer prosjektstyringsverktøy, utviklingsrammeverk, systemarkitektur, versjonskontroll, IDE og databaseverktøy. Det er tatt et bevisst valg om å ha et eget definert kapittel om språk og verktøy, da dette er en stor del av avgjørelsene som ble tatt i prosjektet.

4.1 Azure DevOps

Som prosjektstyringsverktøy tar vi i bruk Azure DevOps, et samarbeidsverktøy fra Microsoft. Azure DevOps er populært blant utviklere, noe som er hovedgrunnen til at valget falt på dette verktøyet (Microsoft, 2021a). Det kan brukes både i nettleser og som applikasjon, i tillegg til at det er gratis og tilbyr gode tilpasningsmuligheter. Med gode tilpasningsmuligheter menes for eksempel tilleggsverktøy som «Estimate», en utvidelse som kan benyttes til å estimere timer. Dette har vi benyttet oss av fra og med Sprint 1, og vil utdypes senere i kapittel 6.1.1.

I DevOps har man kontinuerlig oversikt over fremgang i sprintene. Dette er en enkel oversikt over alle oppgavene i sprintene, og hvilken status de har. En oppgave kan defineres som *in-progress, review* og *done.* Man kan definere og tildele ansvar, og logge brukt tid på hver oppgave. Gruppen bruker funksjonen «Time Log Summary» for å registrere tid brukt på hver oppgave.

4.2 Rammeverk – ASP.NET Core

For utviklingen av systemet har vi benyttet oss av webapplikasjonsrammeverket ASP.NET Core 3.1. Dette er et populært og godt dokumentert rammeverk som er optimalisert for utvikling av webapplikasjoner. Rammeverket er designet og utviklet av Microsoft (Microsoft, 2020a).

Vi har valgt å benytte oss av dette rammeverket fordi Egde tidlig ga uttrykk for at de har god kompetanse i dette, og at de dermed kunne bistå gruppen dersom utfordringer knyttet til programvareutviklingen skulle oppstå. Gruppemedlemmene viste stor interesse for å lære seg mer om rammeverket, og fant raskt ut av at dette var et passende rammeverk som kunne benyttes for å løse prosjektets oppgaver på en god måte. ASP.NET Core tilbyr mange muligheter for integrasjon med databaseløsninger, for eksempel Entity Framework (EF) som vi har benyttet oss av.

Core-versjonen muliggjør utvikling på tvers av plattformer, som er nyttig da både MacOS og Windows benyttes i gruppa (Halvorsen, 2020, s. 12). Vi benytter oss av versjon 3.1, da vi ble anbefalt dette av Egde. For 3.1 finnes mer dokumentasjon enn ved nyere versjoner, samt at

sannsynligheten for uforutsette programvarefeil ofte er mer tilstedeværende ved nyere versjoner.

4.3 Systemarkitektur – Razor Pages

Razor Pages er et systemarkitektonisk mønster utviklet for ASP.NET Core for å forenkle strukturen og organiseringen av kode i applikasjoner (Smith, 2017). Der MVC (Model-View-Controller) ofte kan resultere i mye mapper og filer med mindre oversikt, tilbyr Razor Pages en struktur der hvert «View» er direkte koblet til dets tilhørende logikk (Microsoft, 2020b). Dette gjøres ved bruk av filutvidelsen «cshtml» som kan tolke både HTML og C# i samme fil for enklere å presentere ønsket informasjon i visningen av en webside. I filstrukturen er disse separert ved at frontend-siden har et filnavn som slutter på «cshtml», mens backend-filen har et filnavn som slutter på «cshtml.cs», slik som illustrert ved Figur 14. Dette er også kalt «Page» for frontend-filen og «Pagemodel» for backend-filen.

Begrunnelsen for at vi har valgt å ta i bruk Razor Pages i vårt system var at flere av gruppemedlemmene tidligere hadde gjennomført nettkurs som tok i bruk dette mønsteret, og det var derfor naturlig å benytte seg av dette i videre prosjekter. I tillegg ble Razor Pages anbefalt for utviklere som var i en tidlig fase av utviklingen, da Razor Pages i teorien er enklere enn andre mønstre (Halvorsen, 2020, s. 15).

4.3.1 Backend

Programmeringsspråket vi har benyttet oss av i dette prosjektet er C#. C# er et objektorientert språk som ligner en del på Java, som er det språket vi er mest kjent med fra tidligere studieår, og som støttes til utvikling i ASP.NET Core (Microsoft, 2021b).

4.3.2 Frontend

Som nevnt har Razor Pages bidratt til forbedret struktur på visning og logikk i applikasjonen. Med en slik modell blir det enklere å utforme sidene, da vi kun trenger å forholde oss til sidens cshtml-fil under selve utviklingen. I ASP.NET Core benytter vi den forhåndsdefinerte filen «_Layout.cshtml», som en mal for prosjektet. Her kan man sette opp en standard for utseende på prosjektet og gjenbruke denne koden. Man kan også velge å sette opp en spesifikk struktur for spesifikke sider (Microsoft, 2020b).

CSS og HTML er det vanligste å benytte seg av, men man kan også ta i bruk Bootstrap som et tillegg. Bootstrap er ferdige layouts som kan lastes ned og implementeres i et prosjekt. Dette gjør det mulig å ta i bruk ferdige løsninger på CSS og HTML som allerede er gode og passer til ønsket formål. Vi har tatt i bruk begge deler, og følger også Egdes merkevarebok slik at blant annet farger, tekst og logo, matcher designet på nettsiden deres.

4.4 IDE - Visual Studio

Vi benytter oss av Microsoft Visual Studio (VS) for utviklingen av vårt system. I nettkursene vi har gjennomført ble det anbefalt å ta i bruk VS, og det er også et av de mest brukte utviklingsmiljøene (IDE) (Carbonnelle, 2020). Det har dermed vært enkelt å finne god dokumentasjon og guider på nett, tips til hvordan man kan importere for eksempel

programvareutvidelser (plug-ins) og pakker, og finne ut av hvordan interaksjon med annen programvare fungerer.

Visual Studio tilbyr integreringsmuligheter med blant annet SQL Server og Git. Det skal derfor være enkelt å knytte applikasjoner opp mot både kontainer og pipelines. Microsoft tilbyr også skytjenester via Azure, og VS kan dermed forenkle prosessen med å flytte systemet opp i en skyløsning ved behov. Visual Studio er tilgjengelig for både Windows og MacOS, og støtter dessuten parprogrammering gjennom «Live Share», slik at flere kan samarbeide om koden.

4.5 Databaseverktøy

Ved bruk av Entity Framework kan et system med en tilknyttet database designes på to ulike måter:

1. Kode først, deretter database
2. Database først, deretter kode

I vårt prosjekt har vi skrevet kode først og deretter opprettet database. Med dette menes at både entiteter, deres egenskaper, koblinger mellom entiteter, samt databasespørringer er blitt skrevet i C# og deretter omstrukturert til databaser ved hjelp av migrasjoner som er en del av Entity Framework. En av fordelene med å utforme systemet på denne måten, har vært at vi enkelt har kunnet endre databasen underveis i utviklingsprosessen. Dette ved å kun endre modellene som utgjør entitetene og deres felter, for deretter å legge til nye migreringer. Databasen endres dermed etter behov, og systemet kan tilpasses etter tilbakemelding fra produkteier. Dette var for eksempel nødvendig da produkteier ønsket at man skulle kunne inkludere et serienummer dersom differensiering av likt utstyr var nødvendig.

4.6 Versjonskontroll

For å opprettholde god oversikt over strukturen i GitHub repository, og for å sikre kvalitet på koden som skal slås sammen med eksisterende kode, opprettet gruppen tidlig en kodestandard som skulle følges ved programvareutvikling. Denne inkluderte blant annet oppsett av git-flyt og ulike «brancher», se Vedlegg nr 3.5 Versjonskontroll.

Ved å benytte «forking» av «repository» har vi fremmet ytterligere kvalitetssikring ved at det kun er git-master som har mulighet til å «merge» kode direkte. På denne måten har vi sørget for at den enkeltes kode har kunnet bli kontrollert før fletting gjennomføres. For at koden til git-master skulle bli kvalitetssikret i like stor grad som resten av gruppen, har andre gruppemedlemmer gått gjennom git-masters kode ved hjelp av skjermdeling i Microsoft Teams, eller ved bruk av verktøy for parprogrammering, gjerne med git-master til stede.

5 Implementering av utlånssystem

Dette kapittelet forklarer hvordan vi har gått frem for å implementere funksjonaliteten til systemet. Dette innebærer utvikling av backend, frontend, og testing av dette. Vi vil også snakke om utfordringer vi har støtt på i utviklingsprosessen.

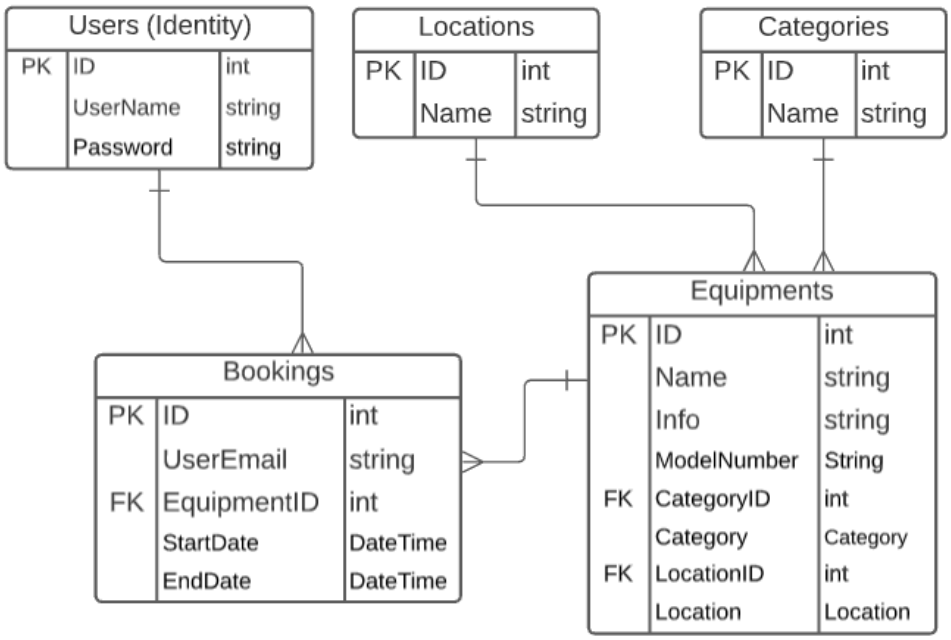
5.1 Backend

I Sprint 2 innledet vi utviklingen med å opprette et ASP.NET 3.1 Razor Pages prosjekt i Visual Studio. Dette tar i bruk en mal som gjør at mye grunnleggende kode blir satt opp for oss. Dette er et enkelt grensesnitt som kan bygges videre på.

5.1.1 Entity Framework

Da vi startet utviklingen av systemet baserte vi først modellene i koden på E/R-modellen og klassediagrammet, se kapittel 2.5 og 2.6. Ved å benytte Entity Framework (EF) har vi i liten grad arbeidet med SQL-kode fordi EF oppretter SQL-kode basert på modeller som via «Migrations» kan benyttes for å opprette, samt endre databasestruktur (Zzzprojects, u.å).

En utfordring knyttet til hvordan modellene var koblet sammen gjorde det vanskelig å hente ønskede verdier fra databasen. Utfordringen omhandlet i hovedsak backend-delen i systemet, der tabellene i databasen ikke var koblet sammen korrekt. Vi valgte derfor å opprette et nytt prosjektet mot slutten av Sprint 3. Dette la videre grunnlag for en omstrukturering klasse- og kodestruktur, som gjorde det lettere å kvalitetssikre kodesegmenter, da disse var mer oversiktlig enn tidligere. Ved dette oppsettet gikk vi frem på tilsvarende måte som tidligere forklart. Her var både oppsett av modeller, klasser og kode forbedret gjennom nytt og forbedret oppsett gjennom refaktorering av kode og ny E/R-modell som illustrert ved Figur 13.

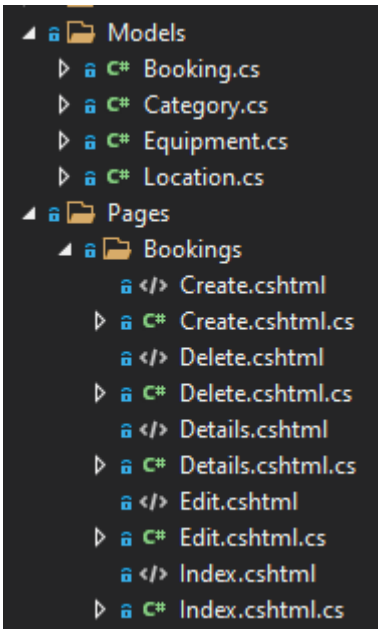


Figur 13: E/R-modell, implementert database.

Endringene i E/R-modellen bestod hovedsakelig av å legge til «lokasjon» og fjerne «kalender» og «favorittListe». I tillegg ble aktuelle felter fra «bookingTime» flyttet til «booking». Tanken med dette var at vi her så det som unødvendig at «bookingTime» stod for seg selv, og at det var mer hensiktsmessig at denne ble lagret med dens respektive booking.

5.1.2 CRUD

Som en del av arkitekturen Razor Pages har man ved dette systemet lagt vekt på prinsippet *CRUD*. CRUD er et akronym for Create, Read, Update og Delete, og er i stor grad benyttet ved utvikling av systemet (Microsoft, 2016). Ved å benytte Razor Pages og CRUD baseres Controller (logikk) og View (visning) på modeller. Følgende vil den enkelte visningen være knyttet til en logikkfil, som utgjør en mappestruktur som både er oversiktlig og enkel å finne frem (Figur 14).



Figur 14: CRUD-prinsipper.

Eksempelvis vil man ved videre utvikling, der man baserer seg på tilsvarende arkitektur, kunne se likheter med det som fremgår av Figur 14. I dette eksempelet illustreres det hvordan CRUD tidligere har blitt benyttet for å opprette visning og logikk for modellen «Booking». Det er verdt å nevne at man ikke vil gjenkjenne alle begrepene under CRUD når prinsippet benyttes i praksis. Som man ser fra figuren vil følgende visning og tilhørende logikk opprettes; Create, Delete, Edit (Update), Index (Read).

5.1.3 Razor Pages

Når vi ønsket å implementere ny funksjonalitet i backend, var det i hovedsak Pagemodel-filen (ref kapittel 4.3) vi arbeidet med. Pagemodel-filen har kontakt med databasen gjennom filen «Context.cs» som ble opprettet som del av «migrations». For eksempel da vi implementerte «Book»-funksjonaliteten til systemet, ble informasjonen om bookingen hentet gjennom Page og sendt til Pagemodel for behandling. I Pagemodel blir det sjekket at verdiene i bookingen er gyldige. Basert på dette vil enten systemet kontakte databasen og legge bookingen til i systemet, eller avvise bookingen og presentere en feilmelding for brukeren.

Denne måten å jobbe på, gjorde at ny funksjonalitet ofte kunne baseres på eksisterende kode, der logikken for den nye funksjonaliteten ble skrevet i Pagemodel-filen. Fordi applikasjonens ulike visninger (Page) ofte hadde store likhetstrekk kunne vi her gjenbruke kode, med større eller mindre tilpasninger. Vi møtte allikevel utfordringer knyttet til dette. For eksempel krevde Pagemodel for booking, der vi ønsket å presentere en kalendervisning, større endringer som ikke kunne knyttes opp mot eksisterende kode.

5.1.5 Docker

Vi ønsket tidlig å undersøke mulighetene for å benytte Microsoft Azures skyløsninger ved videre drift av systemet. Det at vi utviklet på både Windows og MacOS ble en utfordring ved at lokale databaseservere ikke fungerte på MacOS. Det ble dermed forsøkt å bruke Docker-konteinere som en potensiell løsning. Ved å benytte oss av konteinere, kunne prosessene som

omhandlet å dytte systemet opp i en sky bli forenklet. Til tross for god veiledning og hjelp av medstudenter, forelesere ved UiA og partnerbedrift, fungerte heller ikke denne løsningen på MacOS. Fordi prosessen med implementering av kontainere og bruk av skyløsning var tidkrevende, samt at vi manglet ressurser, ble planer om dette nedprioritert.

5.1.6 Microsoft Azure AD

Et av kravene fra produkteier var at systemet skulle ha en form for sikker innlogging. Fra starten, var begge parter enige om at innlogging via SSO for Microsoft-kontoer kunne benyttes. Det viste seg etter hvert, at vi trengte tilgang til produkteiers Azure Portal for oppsett av Azure Active Directory (AAD) for Egdes brukere (Microsoft, 2021c). Med AAD er det kun de som er blitt tildelt tilgang til systemet som kan logge inn. I tillegg skal ulike brukere ha ulike rettigheter i systemet, da vi skiller mellom administrator og vanlig bruker. Dette kunne løses ved å opprette «Grupper» i Azure, men det krevde igjen tilgang til Egdes Azure Portal. Utfordringene knyttet til AAD har dermed vært at vi ikke har hatt tilgang til ressurser.

Grunnet utfordringene nevnt over, har gruppen undersøkt andre muligheter for en sikker innloggingsløsning. Vi har kunnet dra nytte av noen av de fordelene som følger med ASP.NET Core-rammeverket. For innlogging har vi derfor benyttet *Microsoft Identity* for brukerinnlogging, der sikkerheten og konfigurasjonen av innloggingsfunksjonen er bygget inn i rammeverket. Denne fungerer, men alle brukere må registreres direkte i applikasjonen og en løsning med bruk av SSO er dermed foretrukket.

5.2 Frontend

For design av brukergrensesnitt, har vi benyttet oss av systemets tilhørende Razor Pages. Sidespesifikke visninger kan tilpasses i disse filene. For felles designelementer har vi tatt i bruk et eget stylesheet. Designelementer er blitt implementert ved bruk av CSS, HTML og Bootstrap.

I Sprint 3 startet forberedelsene til implementering av design. Vi ønsket å bruke tid på å tilegne oss kunnskap om HTML, CSS og Bootstrap før selve implementeringen startet i Sprint 4. Et par av gruppemedlemmene fikk hovedansvar for frontend-utviklingen av systemet, slik at både læring og implementering kunne skje parallelt med andre arbeidsoppgaver. Vi hadde som mål i denne sprinten å bli helt ferdige med all funksjonalitet i backend, slik at alle designelementene kunne komme på plass før Sprint 5.

I Sprint 4 var fokuset å utarbeide et felles oppsett for alle sidene i systemet. Navigasjonsbar ble lagt til på alle sider, og vi begynte å bruke Bootstrap til å tilpasse de funksjonene vi ønsket oss. Vi benyttet oss av lignende utseende som Egde har på sine nettsider, og farger hentet fra merkevareboken.

I første halvdel av Sprint 5 fokuserte to av gruppemedlemmene utelukkende på ferdigstilling av brukergrensesnittet, med særlig vekt på tilbakemeldinger fra produkteier. For consistency (ref. kapittel 3.2.3) ble alle sidene gått nøye igjennom for å sjekke at alle elementer samsvarte gjennom hele systemet. Vi ønsket at alle knapper som tilførte noe til systemet, eksempelvis «Book» og «Endre», skulle være blå med hvit tekst. Knappene som fjernet noe fra systemet skulle være oransje med svart tekst. Disse knappene har også en hover-effekt (ref. kapittel 3.2.4).

Listevisninger og tekst ble justert slik at alle linjer ble korrekte for å oppnå et symmetrisk inntrykk, da dette gir et ryddig design og ser visuelt bedre ut for brukerne av systemet. Validering av handlinger ble lagt til på handlinger med betydelige endringer. Når en bruker booker et utstyr, får den en beskjed om bookingen er gjennomført eller ikke. Den samme type validering blir brukt dersom en bruker fjerner en aktiv booking.

I utviklingen av systemet har vi kun fokusert på visning på PC. Flere av designelementene er skalerbare, men det er ikke noe vi har lagt stor vekt på i designprosessen. Visning på mobil er derfor ikke optimal.

5.3 Testing

Ved utvikling av systemet, har man underveis måttet teste systemets funksjonalitet og oppførsel. Testene kan utføres manuelt eller gjennom automatiserte tester. Ved utvikling av dette systemet har fokus primært vært rettet mot gjennomføring av manuelle tester.

Samtidig som vi satte i gang utviklingen av systemet i Sprint 2, begynte vi også å fokusere på testing. Dette har ikke vært et krav fra produkteier, men vi har likevel forsøkt å sikre kvaliteten på produktet og ikke minst, lære mer om testing. Vi begynte derfor å undersøke hvordan enhets-testing av systemets enkelte enheter fungerte ved bruk xUnit som testrammeverk (xUnit, u.å). Gruppens medlemmer hadde ingen erfaring med automatiserte tester ved bruk av ASP.NET Core og C# fra før, så dette var nytt for samtlige.

I tillegg til manglende kunnskap innen testing, ble læringsprosessen utfordrende grunnet begrenset ressurs og eksempler på internett. Vår plan med dette var å sørge for at systemets ulike deler fungerte som de skulle til enhver tid. Dette ved hjelp av automatiske tester som skulle settes opp og kjøre gjennom enhetstestene. Det ble stadig mer krevende å utvikle systemets funksjoner enn vi tidligere hadde antatt. På grunn av dette, og at vi hadde tidsfrister til ferdigstilling av funksjonalitet ved ulike sprinter, ble gruppen enige med partnerbedriften om at testing ikke skulle være et stort fokus videre.

Ved videre utvikling ble derfor fokuset rettet mot gjennomføring av manuelle *white-box-tester*. Dette innebærer gjennomgang av koden og testing av funksjonaliteter manuelt, ved for eksempel å registrere data i systemet, og sammenlikne dette med det vi forventer. Manuelle tester er mer tidkrevende enn automatiske tester, men grunnet mangel på kunnskap og nødvendige ressurser, valgte vi å fokusere på manuelle testing slik at produktet vi utviklet holder høy kvalitet.

For å heve systemets kvalitet da all funksjonalitet var ferdigstilt mot slutten av Sprint 4, henvendte gruppen seg til Egde for støtte i forhold til automatisert testing av systemet. Gjennom en sesjon med en av Egdes utviklere fikk vi støtte og bistand til å sette opp noen enhetstester i systemet. Disse skulle så teste om databasen registrerte dataen som var forventet.

Selv om hovedfokuset ikke har vært rettet mot testing ved utvikling av systemet, har vi likevel erfart hvor viktig testing er for å kunne levere et system som lever opp til forventningene til kunden.

6 Prosjektgjennomføring

I denne delen av rapporten vil sentrale avgjørelser, metodikk og relevante begreper anvendt i gjennomføringen av prosjektet redegjøres. Dette innebærer å definere valgt metodikk, og hvordan gruppen har arbeidet med metodikken. Vi vil også definere sentrale begreper i Scrum, samt beskrivelser av de ulike sprintene. Videre vil gruppens backlog presenteres og redegjøres for. Siste del av dette kapittelet omhandler gruppens tidsressurser og hvordan ressursene er allokert gjennom prosjektet.

6.1 Scrum

I dette prosjektet har gruppa valgt å benytte Scrum som arbeidsmetodikk. Dette delkapittelet vil derfor ta for seg teorien rundt Scrum, etterfulgt av en beskrivelse av prosjektets roller og product backlog. Videre vil det beskrives hvordan prosjektgruppa har arbeidet under de ulike sprintene. Til slutt redegjør vi for tidsstyring.

Scrum er et av de mest populære rammeverkene for agilt arbeid (Bibik, 2018, s. 11). Å arbeide agilt eller smidig i Scrum, handler om å jobbe iterativt ved å få hyppige tilbakemeldinger og respondere på disse. En av fordelene med å ta i bruk Scrum er at metodikken oppmuntrer til korte iterasjoner. Disse byr på lavere risiko og tilrettelegger for å tidligere kunne se avkastning på investeringene (McKenna, 2016, s. 27).

For å komme i gang med denne agile arbeidsprosessen er det en rekke ulike elementer som må være på plass. Et av disse elementene er roller. I et prosjektteam er det vanlig å finne *produkteier, scrum master* og *scrum team.* Rollene blir nærmere beskrevet i kapittel 6.2. Neste element er product backlog. En *product backlog* inneholder de spesifikke krav produkteier har til prosjektet. Oppgavene i backlog blir kalt brukerhistorier, og disse skal være formulert fra en brukers perspektiv. Disse brytes ned i gjennomførbare oppgaver og blir prioritert etter forretningsverdi. Det er disse oppgavene scrum team arbeider med under de ulike sprintene (McKenna, 2016, ss. 27-29).

Scrum-prosessen handler om å arbeide i såkalte sprinter, eller iterasjoner. For hver sprint forplikter scrum teamet seg til en gitt arbeidsmengde de mener de klarer å fullføre i løpet av en sprint. Produkteier og scrum teamet blir enige om mål for hver sprint, og oppgavene fra product backlog flyttes til en sprint backlog. Det er vanlig å ha en varighet på 2-4 uker per sprint, med en 24-timers syklus. Hver dag møtes hele prosjektgruppen til et møte. Dette møtet kalles *daily stand-up*. På dette møtet forteller alle hva de har gjort siden sist, hva de skal gjøre og eventuelle utfordringer ved arbeidet. Ved endt sprint kaller scrum teamet inn til sprint review og presenterer sitt arbeid og resultat. Før en ny sprint kan starte må det holdes et sprint retrospective-møte. Dette er et refleksjonsmøte hvor prosjektgruppen tar for seg den foregående sprinten, både hva som var bra og hva som kan forbedres i neste sprint (McKenna, 2016, ss. 31-32).

6.1.1 Sprint Planning

Sprint planning gjennomføres i starten av hver sprint, og er obligatorisk for medlemmene i scrum teamet. Her møtes teamet for planlegging av sprinten. Det skal utvelges en eller flere brukerhistorier som skal gjennomføres i løpet av sprinten. Teamet gjør så om brukerhistoriene til en detaljert liste over arbeidsoppgaver som de forplikter seg til å gjennomføre i løpet av sprinten (Bibik, 2018, s. 20). Produkteier skal delta og forklare målet for sprinten, samt gå igjennom funksjonalitetene med høyest prioritet. Her kommer så produkteier inn og tar på seg rollen som kunde, hvor produkteieren svarer på spørsmål om brukerhistoriene fra kundens perspektiv (McKenna, 2016, s. 92).

Del 2 av sprint planning, er estimering av tidsbruk, heretter kalt estimeringsmøte. Ifølge teorien skal estimeringsmøter inkludere hele teamet. Dette for å utforske teammedlemmers perspektiver der man kan ha ulike oppfatning av innsatsen som kreves for den enkelte oppgave. Videre kan en god estimering bidra til å gi både teamet og produkteier bedre innsikt i nødvendig arbeidsinnsats (Radigan, u.å.).

6.1.2 Daily stand-up

Daily stand-up er et obligatorisk, kort møte som hovedsakelig tar for seg tre spørsmål: *Hva er gjort siden sist? Hva er neste steg på agendaen? Er det noe som hindrer deg fra å utføre jobben?* Et stand-up-møte skal ikke vare lenger enn maksimalt 15 minutter og skal handle om hva hvert enkelt medlem forplikter seg til å gjøre ovenfor teamet sitt. Ved å svare på de ovenstående spørsmålene, gir det resten av scrum teamet en oversikt over hva som er gjort og hva som gjenstår. Hindringer som oppstår skal håndteres av scrum master, slik at scrum teamet kan jobbe uforstyrret med oppgavene sine. Daily stand-up kan være åpent for at hvem som helst involvert i prosjektet kan delta, men kun scrum teamet har lov til å snakke (McKenna, 2016, ss. 87-88). McKenna anbefaler også at man skal stå oppreist under disse møtene da dette angivelig skal øke oksygentilførselen til hjernen med 20 % fremfor å sitte (McKenna, 2016, s.53).

6.1.3 Sprint Review

På slutten av hver sprint holdes det et møte hvor teamet presenterer hva de har produsert i løpet av sprinten og ber om tilbakemeldinger fra interessenter. Dette gjøres av scrum teamet, ikke scrum master. I dette møtet skal også produkteier bygge relasjon med kunden og får belyse verdien systemet vil komme til å få for kunden (McKenna, 2016, s. 31, 182-183).

6.1.4 Sprint Retrospective

Scrum team og scrum master deltar i sprint retrospective, Produkteier kan også delta dersom det er enighet om dette. Prosjektgruppen diskuterer hva som gikk galt, hva som var bra og eventuelle endringer eller forbedringer som kan gjøres til neste sprint (Bibik, 2018, s. 21).

6.1.5 Hvorfor passer Scrum i dette prosjektet?

Dette er et prosjekt hvor et system skal utarbeides i en prosjektgruppe. Prosjektet skal planlegges og estimeres, samt gjennomføres på en ryddig og oversiktlig måte. Vi bestemte oss for å bruke arbeidsmetodikken Scrum. Dette fordi gruppen har kjennskap til denne arbeidsmetodikken fra tidligere, noe som sparte tid ved å unngå å sette seg inn i et nytt

rammeverk for prosjektgjennomføring. Gruppen hadde også hatt positive erfaringer med Scrum, noe som var en betydelig faktor for valget.

I tillegg trives gruppen med å ha kortsiktige mål ved å dele opp perioden i iterasjoner. Dette opplevde gruppen som motiverende ved at man kunne se fremgang tydeligere enn om man ikke skulle brukt flere små iterasjoner for prosjektet. I tillegg har universitetet oppfordret oss på studiet til å ha et tett samarbeid med bedriften gjennom semesteret. Da det ved gjennomføring av prosjektet er en pågående pandemi, ønsket gruppen å sette av tid til å møte produkteier digitalt i slutten av hver sprint/iterasjon. Det er en del fordeler ved å bruke Scrum som arbeidsmetodikk. Følgende er listet opp av Bibik (2018, s. 16):

a. Kontinuerlig forbedring
b. Gjennomsiktighet
c. Tidlig tilpasning
d. Omfavne endring

Med disse punktene menes det for det første, at vi kontinuerlig kan forbedre systemet fra iterasjon til iterasjon, da vi kjører en hel programvaresyklus i hver sprint. Videre betyr gjennomsiktighet at man hele tiden får tilbakemeldinger som man kan vurdere underveis. Disse bidrar til å gi klarhet i hvordan det endelige produktet vil bli. Tidlig tilpasning går mye på det samme, hvor man kan levere ny fungerende funksjonalitet i slutten av hver sprint og få tilbakemeldinger på disse, for så å kunne tilpasse i de forestående sprintene. Siste punkt handler om at man etter hver sprint får muligheten til å reflektere og sammenligne arbeidet man har gjort i løpet av en sprint. Dette kan gi muligheten for å bestemme seg for om man ønsker endring i systemet eller prosjektet i sin helhet (Bibik, 2018, s.16).

Prosjektet i sin helhet taler også for å bruke Scrum. Dette fordi prosjektet bygger på en tett kommunikasjon med kunden, og skal utarbeides med produkteiers krav som hovedprioritet. Derfor er det viktig at produkteier er enig i de valgene som tas. I tillegg er tidsrammen på prosjektet noe som gjør Scrum mer tiltalende å bruke. Dette er et relativt langt prosjekt med store mål, hvor man underveis har hatt krav til å rapportere tilbake til universitetet. Korte iterasjoner har da sørget for jevn fremgang i prosjektet. I tillegg har situasjonen som pågår, med digitale møter og hjemmekontor, gjort at gruppen ikke har fått jobbe på kontoret til Egde. Derfor har de faste møtene med sprint review, hvor representanter fra Egde har vært til stede, sørget for jevn og hyppig kontakt med samarbeidsbedriften.

6.2 Roller i Scrum

I dette avsnittet defineres rollene i Scrum. Hver av rollene har viktige arbeidsoppgaver og et samarbeid mellom disse er det som skal sørge for progresjon og kvalitet, samt gjennomføring av et utviklingsprosjekt. Figur 15 viser rollene som er aktuelle i prosjektet.

Figur 15: Prosjektets scrum team med interessenter.

Den første rollen, scrum team, skal bestå av mennesker som kan skrive, teste, dokumentere og levere fungerende programvare i hver sprint. Det skal ikke være definerte roller da et slikt team skal være tverrfaglig og selvorganiserende. Målet til et scrum team er å produsere verdi (McKenna, 2016, s. 42-45).

Ifølge McKenna er det spesielt fem verdier som er viktige å ha i et scrum team: forpliktelse, respekt, mot, åpenhet og fokus. Teamet må derfor kunne forplikte seg til krav som er satt til hver enkelt oppgave, ha respekt for produkteiers input, ha mot til å si ifra dersom produkteier øker arbeidsmengden i for stor grad, og holde fokus i gjennomføringen av oppgaver. For at et scrum team skal fungere godt sammen, er det viktig med åpenhet og gjennomsiktighet i prosjektet (Mckenna, 2016, s. 43).

Figur 15 illustrerer at fire av seks gruppemedlemmer utgjør prosjektets scrum team. Det er viktig å påpeke at alle seks medlemmer har vært med i utarbeidelsen av rapporten og utlånssystemet. Som nevnt innledningsvis er Go4IT en tverrfaglig gruppe med studenter, og dette har gruppen brukt til sin fordel, blant annet ved å gjennomføre ulike workshoper for å dele kompetansen og sikre fremgang og kvalitet i prosjektet. Videre har scrum teamet forpliktet seg til en rekke regler i form av en gruppekontrakt, hvor respekt, forventinger og konsekvenser er avklart (Vedlegg 3.20). Scrum teamet har jobbet agilt i prosjektets fem sprinter, samt lyttet til tilbakemeldinger og foretatt endringer basert på tilbakemeldingene fra produkteier, sluttbruker og veileder. Under samtlige sprint review og styringsgruppemøter har det kommet forslag til ny funksjonalitet til utlånssystemet. Da har scrum teamet og produkteier sammen diskutert hvordan de skal prioriteres eller om funksjonaliteten er nødvending å implementere.

Den andre rollen er scrum master, som har i oppgave å gjøre det som kreves for at scrum teamet skal komme i mål. Blant annet skal en scrum master ta ansvar for å håndtere eventuelle

hindringer og skal sette teamet først. Scrum master må sikre at teamet anvender det agile rammeverket på rett måte, som kan inkludere å lære bort agile prinsipper til teamet (McKenna, 2016, ss. 46-54).

Scrum master skal også tilrettelegge og avholde Scrum-hendelser, inkludert planlegging av møter. Scrum master bør hele tiden motivere for nært samarbeid på tvers av roller og funksjoner, identifisere flaskehalser og mangel på gjennomsiktighet. Scrum master skal sørge for høy grad av fremdrift og ytelse. I tillegg skal den beskytte teamet fra eksterne avbrytelser eller forstyrrelser, slik at teamet kan fokusere fullt ut på den pågående sprinten. En Scrum master skal også ha gode nok ferdigheter rundt mennesker til å kunne håndtere eventuelle konflikter og dårlige dager, samt være oppmuntrende ovenfor scrum teamet (McKenna, 2016, ss. 46-54).

Fra Figur 15 ser man at Go4IT har to scrum mastere. Hovedgrunnen til at gruppen har valgt å ha to scrum mastere er fordi gruppen mener det kan være en fordel å ikke være alene med ansvaret. Dette ved at det åpnet for en arena hvor utfordringer og administrative avgjørelser kunne diskuteres, samt legge planer for prosjektgjennomføring. I løpet av prosjektet har scrum masterne sendt ut møteinnkallelser til aktuelle personer, skrevet agendaer og referater, og holdt tre ukentlige stand-up-møter. Videre har scrum masterne forsøkt å holde motivasjonen innad i gruppen oppe. Scrum masterne har avklart med Egde om hvilke ressurser de kan gjøre tilgjengelig for dette prosjektet, samt fulgt opp administrative oppgaver. Scrum masterne har i tillegg vært en del av scrum teamet.

Den tredje rollen i prosjektet er produkteieren. Produkteieren fungerer som et knutepunkt mellom scrum team, kunden og bedriften. Produkteieren kommuniserer med kunder for å definere krav til systemet, og har ansvar for å utarbeide en backlog som scrum teamet kan jobbe videre med for å kunne utvikle fungerende programvare. Produkteier har også fokus på å maksimere avkastningen på investeringen, og må hele tiden tenke på både bedriften og kunden. Dette betyr at hvert valg som tas veies opp mot hvordan det vil påvirke bedriften, samt hvordan det vil skape verdi for kunden. Produkteieren har autoritet til å akseptere eller avslå fullført arbeid og kan endre retningen på prosjektet underveis (McKenna, 2016, s. 39).

Den fjerde rollen er sluttbrukeren, og er den som har behov for systemet. Med bedriftskunnskap vil sluttbrukeren kunne definere produktet ved å fortelle scrum teamet hva som forventes (Gloger, 2010, s. 14). Fra Figur 15 ser man at sluttbrukeren er ansatt i Egde. Sluttbrukeren har vært med på sprint review/styringsgruppemøter. I disse møtene har sluttbrukeren fått teste systemet samt gitt sine tilbakemeldinger. Sluttbrukeren har gitt verdifulle tilbakemeldinger på systemets funksjonalitet og utforming. Dette gjorde at gruppen kunne jobbe konstruktivt med tilbakemeldingene som ble gitt.

Den siste rollen fra Figur 15 er veileder. Veileders rolle i prosjektet er å veilede studentene og å delta på styringsgruppemøter. Veileder har stilt en rekke viktig spørsmål under sprint møter, eksempelvis om Egde hadde et designdokument, noe som resulterte i at gruppen fikk tilsendt Egdes merkevarebok, og dermed var en rekke avgjørelser tatt i forbindelse med design og utforming. I tillegg har veilederen vært tilgjengelig for spørsmål og gått igjennom rapporten med gruppen.

6.3 Product backlog

En product backlog er utarbeidet etter kravspesifikasjonene gruppen mottok 14. desember 2020 fra produkteier. Tabell 4 viser prioriterte kravspesifikasjoner (systemkrav-ID 1-8), kategorisert som «Must Have» eller «Should Have» (se kap 2.2). Gruppen har supplert med systemkrav-ID (SID) 9, etter samtaler med produkteier og SID 48 (hentet fra DevOps) for å kunne danne fundamentet for systemutviklingen. I tillegg har gruppen lagt inn SID I, II og III med en egen prioritering om å *bli ferdig*. Disse er inkludert i product backlog, fordi elementene må være ferdig for at vi kan definere bacheloroppgaven som ferdig.

Videre har gruppen lagt inn kolonnen «Status» i Tabell 4, som viser stjerner. En helt gul stjerne representerer at systemkravet er fullført. En halvt gul, eller grå stjerne, representerer at arbeidet er i gang, eller ikke påbegynt. Kolonnen «Gjennomføring i %» viser hvor mange prosent som gjenstår før brukerhistorien er fullført. Videre viser kolonnene «Estimert», «Timer brukt» og «ETC» (Estimated Time to Complete), sine respektive verdier. Tallene i disse kolonnene er en summering av de mindre oppgavene. En brukerhistorie er 100 % når alle underoppgaver er fullført og ETC er satt til null. Dersom alle brukerhistoriene tilknyttet et systemkrav er fullført, endres statusen slik at stjernen blir helt gul, og kravet anses som ferdig.

Tabell 4: Product backlogen

SID	Systemkrav	Status	Prioritering	BID	Brukerhistorie	Gjennom-føring i %	Estimert	Timer brukt	ETC
1	Systemet må være selvbetjent for brukerne	☆	Must have	1	Som ansatt i Egde, ønsker jeg enkelt å booke bedriftens utstyr ved behov, slik at jeg kan benytte meg av dette til arbeids- eller fritidsformål.	100	189	146	0
2	Systemet må ha en form for sikker innlogging/registrering	☆	Must Have	2	Som bruker, vil jeg kunne logge inn i systemet med personlig O365 bruker, slik at systemet er sikkert og ingen utenforstående kan booke utstyret til bedriften.	95	88	82,5	7
3	Systemet må fungere på Egdes hovedkontorer	☆	Must Have	3	Som bruker, vil jeg kunne booke utstyr og levere uavhengig av kontor, slik at jeg kan spare tid og ressurser.	87	60	57	8
4	Systemet må lagre informasjon om produkt og bruker	☆	Must Have	4	Som bruker ønsker jeg å kunne se en oversikt over mine bookinger, slik at jeg kan se hvilket utstyr som er booket til hvilken tid.	100	52	67	0
48	*Fundament*	☆	*Must have*		Analyse og design modeller	100	114	169,25	0
5	Produktet bør gi brukeren mulighet til å se hvem andre som har et spesifikt produkt	☆	Should Have	5	Som bruker, ønsker jeg å få oversikt over hvem som har utstyr til enhver tid, slik at jeg slipper å lete og spørre rundt.	100	60	20,5	0
				6	Som bruker vil jeg kunne filtrere utstyr etter forskjellige kriterier.	100	237,5	97,75	0
				7	Som bruker vil jeg kunne søke etter utstyr ved utstyrsnavn	100	64	60	0
6	Produktet bør ha et enkelt brukergrensesnitt	☆	Should Have	8	Som bruker ønsker jeg et intuitivt og brukervennlig utlånssystem, slik at booking kan skje så lett som mulig.	100	78	87,5	0
7	Produktet bør gi administrator og ansatte muligheten til å booke et produkt frem i tid	☆	Should Have	9	Som bruker/admin, ønsker jeg å booke utstyr fremover i tid, slik at jeg kan holde av spesifikt utstyr til spesifikke formål.	100	120	148,75	0
8	Administrator for systemet bør kunne legge til, endre og fjerne utstyr og filtreringskategorier.	☆	Should Have	10	Som Admin vil jeg legge/redigere/slette utstyr/lokasjon/kategori	100	27	14	0
9	*Systemet bør ha god performance.*	☆	*Should have*	*11*	Som produkteier ønsker jeg et system med god performance, for å slippe for mye vedlikehold og oppgraderinger etter overtakelse.	100	176	132,5	0
Nødvendige elemter for ferdigstilling av bacheloroppgaven									
I	Annet	☆	Bli ferdig		timer, brukerhistorier, research, adminoppgaver ovs.	100	335	315,5	0
II	Rapport	☆	Bli ferdig		Skrive, lese, rett tilbakemeldinger og korrekutrlese	100	360,5	288	0
III	Møter/Teambuilding	☆	Bli ferdig		Scrum møter, gruppemøter og veiledingsmøter + teambuidling.	100	352	292,25	0

Et krav som defineres som et «Must Have», er et obligatorisk krav som må være på plass for at produktet kan leveres og bli akseptert. Dersom noen av kravene i «Must Have»-kategorien mangler, kan det føre til at leveransen ikke blir godkjent. «Should Have»-prioriteringen er ikke obligatorisk, men skal inkluderes i en endelig levering om mulig for å få et bedre produkt (Varma, 2015). Tabell 4 viser at brukerhistoriene-ID (BID) 2 og 3 ikke er fullført, begge er prioritet som «Must Have» og er ifølge litteraturen obligatorisk. Dette kunne vært grunnlag for å ikke få godkjent leveransen.

For å fullføre BID 2, trenger gruppen ressurser i Azure. Systemet er tilrettelagt for at bedriften kan legge inn SSO, men studentgruppen har ikke rettigheter til å gjøre dette. Gruppen har estimert at det er sju timer igjen for å lukke brukerhistoriene. For å fullføre BID 3, må systemet legge til muligheten for at en ansatt kan endre lokasjon på utstyret de har booket. Eksempelvis, hvis en ansatt booker et kamera i Kristiansand og leverer det i Grimstad, er det per dags dato bare administrator som kan endre lokasjonen. Dette kan føre til feil ved at utstyr kan befinne seg på annen lokasjon enn det som er oppgitt i systemet.

Gruppen har fått leveransen godkjent selv som BID 2 og 3 ikke er 100 % ferdig. Grunnen til dette er at produkteier ikke har gitt oss de nødvendige ressursene eller endret prioriteringen.

6.4 Sprinter

I dette kapitlet presenteres de fem sprintene prosjektgruppen har gjennomført i løpet av semesteret. Her presenteres alle de forskjellige møtene i sprintene og hvordan disse er gjennomført.

6.4.1 Pre-Sprint

4. januar-8. januar

Mål for Pre-Sprint: Avtale, kartlegge og bli enige om hvordan vi ønsker å arbeide med utlånssystemet, både internt i Go4IT og med Egde.

I forberedelsesfasen ble det brukt tid på å legge selve fundamentet for prosjektarbeidet. For å sikre god arbeidsflyt gikk vi gjennom kalenderen til alle gruppemedlemmer slik at vi kunne legge en felles arbeidsplan. Vi ønsket tidlig å bestemme oss for rammeverk for prosjektarbeidet, og dette kom også på plass i forberedelsesfasen. Det ble også utarbeidet en gruppekontrakt hvor regler og eventuelle konsekvenser ble stadfestet, samt forventninger til innsats fra medlemmene. En gruppekontrakt vil kunne forhindre konflikt innad og setter standarden for ønsket arbeidsmiljø og kvalitet i prosjektet. Videre ble det utarbeidet et forslag til en product backlog basert på systemkrav fra Egde, og det ble diskutert ulike metoder for gjennomføring slik at alle var forberedt til Sprint 1. Det ble også diskutert hvilke verktøy som skulle benyttes for prosjektstyring, og valget falt på Azure DevOps. Med fundamentet på plass sparte vi verdifull tid da Sprint 1 startet.

6.4.2 Sprint 1

11. januar – 29. januar

Målet for Sprint 1: Ferdigstille analyse, design og prototype samt tilegne oss grunnleggende kunnskap i ASP.NET Core og Razor Pages.

Sprint Planning 1

Arbeidet i Sprint 1 startet med sprint planning, et planleggingsmøte hvor kravspesifikasjoner fra produkteier blir gjennomgått og satt inn i en product backlog. Det ble i tillegg diskutert hvilke andre oppgaver som er viktige å ha med i en backlog, og disse valgte vi å kategorisere som *Fundament*. Under Fundament inngår alle oppgaver som tilhører temaene analyse, design og prototype, blant annet utarbeidelse av navigasjonskart, E/R-modell og skisser. På denne måten sørger vi for en oversiktlig og grundig arbeidsplan.

For å sikre kvalitet og håndtere utfordringer i prosjektet, utarbeidet vi en risiko- og sårbarhetsanalyse med tilhørende kvalitetssikringstiltak. Som tidligere nevnt må vi tilegne oss kunnskap om et nytt webutviklingsrammeverk. Da ASP.NET Core og Razor Pages er helt nytt for alle prosjektmedlemmer, er det hensiktsmessig å planlegge timer for å tilegne oss kunnskap innenfor disse områdene.

Etter at utkastet til backlog var ferdig, startet arbeidet med å lage Sprint Backlog 1. Utvalgte oppgaver ble brutt ned i mindre oppgaver. Å bryte ned oppgaver gjør at vi i større grad kan definere arbeidet som må gjøres. I tillegg er det enklere å estimere tid på små oppgaver. For Sprint 1 valgte vi blant annet å utarbeide rike bilder, skrive brukerhistorier, lage navigasjonskart og ROS-analyse, samt å lære oss det nye programmeringsspråket og utviklingsrammeverket.

Produkteier hadde dessverre ikke mulighet til å delta på Sprint Planning 1, men var med på et annet møte der vi gikk gjennom backlog og dens innhold. Vi gikk gjennom oppgaver og prioriteringer med produkteier og fikk diskutert ting som var uklart. Blant annet ble det nevnt at seksjonene som omhandlet design og analyse var for vage, og disse måtte derfor defineres tydeligere og brytes bedre ned. Dette gjorde teamet ved å sette opp konkrete oppgaver innenfor de forskjellige oppgavene, blant annet å utarbeide rikt bilde, og dermed var backlog fullført for Sprint 1. Produkteier valgte så å godkjenne disse oppgavene slik at teamet kunne starte arbeidet.

Estimeringsmøte 1

Da del 1 av sprint planning var fullført, hvor oppgaver var definert, gikk gruppen over på del 2 av møtet hvor vi skulle estimere tidsbruk per oppgave. For tidsestimering av arbeidsoppgaver tok vi i bruk tilleggsverktøyet «Estimate» i Azure DevOps. Med dette verktøyet kunne alle medlemmer delta på en digital planleggingspoker. På tur blir én og én oppgave estimert ved at hvert prosjektmedlem velger antall timer de tror det tar å fullføre oppgaven. Det antallet timer som flest svarer blir dermed estimatet for oppgaven. Dette var første gangen vi estimerte tid på denne måten, men vi syns at dette var en god måte å gjøre det på. Da alle oppgavene hadde fått et tidsestimat diskuterte vi hvem som skulle være ansvarlig for gjennomføringen av oppgaven.

Daily stand-up 1

Hver mandag, onsdag og fredag kl. 09.15 har gruppen 15 minutter til å gjennomføre stand-up. I Sprint 1 har prosjektgruppen gjennomført daily stand-up i henhold til teorien. Hvert medlem fortalte hva som var gjort siden sist, og hva planen var for dagen. I tillegg ble det tatt opp eventuelle hindringer dersom det var noen. Møtet ble holdt innenfor tidsrammen på 15 minutter. I motsetning til hva teorien anbefaler, ble disse møtene avholdt digitalt og samtlige team-medlemmer ble dermed sittende foran skjermen i dette møtet, fremfor å stå oppreist. Gruppen valgte å kun gjennomføre daily stand-up annenhver dag, da gruppen var enige om at det holdt å møtes tre ganger i uka for å oppsummere hva som var gjort siden sist. Dette for at alle kunne ha mer arbeid å vise til, slik at det føltes mer motiverende. En annen faktor er det andre faget vi har hatt i løpet av semesteret som ikke er relevant for daily stand-up. Derfor ønsket vi i gruppen å ha rom til å jobbe med det andre faget noen dager, og fortsatt kunne vise til relevant arbeid på stand-up.

Sprint Review 1

Første sprint review ble holdt med produkteier, sluttbruker og veileder til stede. Dette møtet ble også definert som det første styringsgruppemøte til Go4IT (se Vedlegg 3.1). Møtet startet med en gjennomgang av backloggen og hvordan teamet administrerer arbeidsoppgaver i løpet av sprinten. Teamet fortalte også om programmeringsspråket som var valgt for utvikling av systemet, og fikk positive tilbakemeldinger fra produkteier og sluttbruker på dette.

Her fikk scrum team fremvist et førsteutkast av en klikkbar prototype (Figur 8) for systemet som skulle utarbeides, med grunnlag i kravspesifikasjoner, brukerhistorier og andre analyse-elementer (delkapittel 2.2 og 2.3) som var gjennomført i løpet av sprinten. Deretter var det satt av tid til diskusjon og tilbakemeldinger fra produkteier, sluttbruker og veileder.

Møtet ble i stor grad avholdt i henhold til teorien. Slik teorien tilsier, ble møtet holdt i slutten av sprinten, der aktuelle deltakere var til stede og det ble fremvist hva som var gjort i løpet av sprinten. Det var mindre deltakelse av produkteier enn det teorien tilsier, og det ble ikke solgt inn til kunden/sluttbruker. Dette ble gjort fordi produkteier også er sluttbruker, samt at kravspesifikasjonene er utarbeidet i samarbeid med sluttbruker slik at kunden allerede er klar over hva verdien av produktet vil bli. Scrum master var ordstyrer i møtet.

Sprint Retrospective 1

Etter endt sprint review var det satt av tid til at scrum team kunne ha et retrospective-møte. Her gikk teamet systematisk gjennom de tre spørsmålene som er sentrale for et retrospective-møte. Dette startet med hva teamet var fornøyde med. Her diskuterte vi fordelene med daily stand-up og alle synes dette var en verdifull måte å oppdatere hverandre på, samt forplikte seg til nye arbeidsoppgaver, spesielt når vi ikke har hatt mulighet til å jobbe sammen fysisk. I tillegg var det bra at vi hadde kommet i gang med prosjektet tidlig i semesteret.

Videre ble det diskutert at den tekniske fremgangen var svært god, hvor dette hovedsakelig gikk ut på å sette seg inn i det nye programmeringsspråket, og å utarbeide de designelementene som var satt opp i backloggen. Prosjektgruppen trivdes godt med hvordan arbeidsoppgavene hadde blitt inndelt og synes det var lett og oversiktlig å jobbe med. I tillegg var gruppen fornøyde med

Azure DevOps som verktøy for å ha oversikt over backloggen og timeregistrering og ønsket å gå videre med dette. Til slutt diskuterte vi gruppens medlemmer på individnivå i sprinten, og alle virket veldig positive til hverandres innsats og aktivitetsnivå i prosjektet så langt.

Deretter diskuterte vi hva som ville være forbedringspotensial til neste sprint. Her kom det blant annet opp at sprinten kan ha vært for lang, eller at vi har hatt for få arbeidsoppgaver. Dette med bakgrunn i at vi tidlig ble ferdig med oppgaver og ikke brukte like mange timer som var estimert. Det ble også diskutert hvorvidt vi skulle starte å legge tidsfrist på de forskjellige underoppgavene i backloggen for å holde arbeidet mer strukturert. I tillegg snakket vi om å utarbeide en policy for hva vi kunne jobbe med når man var ferdig med arbeidsoppgavene for sprinten slik at så lite tid som mulig skulle gå til spille. Her ble det diskutert at vi kunne ta opp disse problemstillingene i løpet av daily stand-up slik at hvert medlem kunne få tettere oppfølging, og for å holde fremgangen i prosjektet så gjennomsiktig som mulig.

Go4IT brukte ROS-analysen (Vedlegg 3.14) i sprint retrospective for å få en oversikt over hvordan vi benyttet forebyggende arbeid og kvalitetssikringstiltak. Tabell 5 viser de viktigste elementene som gikk bra i Sprint 1, i motsetning til Tabell 6 som viser hva som ikke gikk bra og hvor vi må iverksette tiltaksplanen. Fargekoden i venstre kolonne er hentet fra risikomatrisen, og sier noe om hvor sannsynlig det er at en hendelse inntreffer, og hvor stor konsekvensen potensielt kan bli (Vedlegg 3.15).

Tabell 5: Hva har gått bra, Sprint 1

ID	**Hva har gått bra?**	**Forebyggendetiltak fra ROS-analysen**	**Beskrivelse**	**Løst ✓ / ikke løst X**
C3 Gruppemedlemmer møter ikke opp uten gyldig fravær	Stand-up møtet	Utsendelse av møteinnkallelse i god tid	Det ble laget serie innkallelser til stand-up mandag, onsdag og fredag kl 09.15 på Microsoft Teams. Dette gjør at alle har satt av tiden i sine kalendere. Dette gjelder for øvrig de andre møtene i sprinten.	✓
E1 Samarbeidsbedrift svarer ikke på mail / annen kontakt fra gruppen **E2** Samarbeidsbedrift har ikke til å møte gruppen.	Samarbeid med Egde	Avtaler møter i god tid / tidsfrist for svar /aksept Gruppen er fleksibel for møtetidspunkt	Vi har gitt Egde forsalg til møtetidspunkter. Go4IT har vært fleksible med sin timeplan som har resultert i at alle møtene ble gjennomført.	✓

Tabell 6: Hva har gått galt, Sprint 1

ID	Hva har gått galt?	Tiltaksplan fra ROS-analysen	Beskrivelse	Løst ✓ / ikke løst X
C1 Gruppemedlemmer vet ikke hva de skal gjøre.	For få oppgaver i sprint Backloggen	Delegere oppgaver ved sprint planning.	Vi hadde lagt inn for få oppgaver i sprinten, og ble ferdig med disse oppgavene før sprinten var ferdig. Når det ikke var flere oppgaver igjen, viste vi ikke helt hva vi skulle gjøre.	**X**
C5 Gruppemedlemmer leverer ikke forventede resultater.	Noen analyse og design elementer ble utarbeidet uten teori.	Andre må gjennomgå arbeidet på nytt.	Det var noen oppgaver i sprinten som ikke ble gjennomført etter gitt standard. Da vil si at den ikke følger teorien. Dette ble løst gjennom at flere gikk sammen og fullførte oppgaven.	✓
G2 Covid-19 begrenser muligheter for fysiske samarbeid i gruppen.	Kun digitalt samarbeid	Følge med på retningslinjer samt motivere hverandre digitalt.	Gruppen mente det ikke ville vært forsvarlig å møtes samt at retningslinjene sa at man ikke skulle samles mer enn fem personer.	**X**
A3 Personer misforstår bruk av teori i praksis.	Klassediagram	Gruppen går gjennom aktuell teori for å avdekke feil, samt oppklare misforståelser.	Klassediagrammet ble utarbeidet etter hva gruppen mente var den beste løsningen, og ikke etter teorien. Dette ble løst gjennom samarbeid.	✓
A4 Personer er lite kildekritiske.	Startet rett på en oppgave	Gruppen går gjennom aktuell teori for å avdekke feil, samt oppklare misforståelser	Gruppen startet rett på de ulike oppgaven, og var dermed ikke kildekritiske i starten. Dette ble løst gjennom samtaler og gjennomgang av andres arbeid.	✓

6.4.3 Sprint 2

2. februar – 22. februar

Mål for Sprint 2: Få all grunnleggende funksjonalitet på plass (MBP).

Sprint Planning 2

Sprint 2 startet med et planleggingsmøte for at prosjektgruppen kunne forplikte seg til et sett med arbeidsoppgaver. Dette ble gjort digitalt over Microsoft Teams, og backloggen ble utarbeidet i Azure DevOps ettersom gruppen var fornøyd med dette som verktøy for planlegging. I denne sprinten var planen å bruke design- og analyseelementene som ble utarbeidet i Sprint 1, til å få på plass de grunnleggende tekniske funksjonalitetene. Gruppen lagde oppgaver i Azure DevOps, en til hver side i den klikkbare prototypen. En av oppgavene het blant annet «Min Side», og oppgaven her var å bli ferdig med denne siden i systemet.

Noe vi tok med oss fra Sprint 1, var at vi ønsket å delegere flere arbeidsoppgaver i Sprint 2. Dette ble gjort som en del av tiltaksplanen fra ROS-analysen for å hindre at medlemmene i prosjektgruppen ikke skulle vite hva de skulle gjøre. Derfor valgte vi å øke arbeidsmengden vi skulle forplikte oss til i Sprint 2, sammenlignet med det vi hadde satt opp i Sprint 1.

Estimeringsmøte 2

I Sprint Review 1 meldte sluttbruker sin interesse for å være med i Sprint Planning 2. Hovedgrunnen til at han ønsket å være med var for å sikre at prosjektgruppen valgte ut riktig oppgaver, samt at han ville vise gruppen hvordan Egde estimerte timer. Sammen med sluttbruker gikk gruppen gjennom de planlagte oppgavene for Sprint 2. Sluttbruker mente at vi godt kunne starte med estimeringen på de planlagte oppgavene.

Vi fikk en innføring i hvordan Egde estimerer vanskelighetsgrad og tidsbruk for hver oppgave. De bruker et poengsystem for å definere ressursbruk på hver av oppgavene i en backlog. Eksempelvis krever en oppgave med 5 poeng mindre ressurser enn en oppgave med 30 poeng. De starter med å stemme på en poengsum for første oppgave i backloggen, blir så enige om hvilken vanskelighetsgrad oppgaven har, og låser så estimatet. Den første oppgaven blir så utgangspunktet for estimeringen av de resterende oppgavene.

Det ble holdt en diskusjon om hvorvidt Go4IT skulle bruke samme estimeringsmåte som Egde eller ikke. Gruppen ble enige om å fortsette å estimere ved bruk av timer, fremfor et poengsystem, hovedsakelig med tanke på burn down chart (ref. kapittel 6.5.3). I tillegg var timer noe alle forstod seg på, mens poeng var det en risiko for å mistolke og dermed ikke få et konkret nok mål på innsatsen lagt i hver oppgave, da ulike individer lettere kan tolke dette forskjellig. Resten av estimeringsmøtet gikk vi gjennom hver oppgave og estimerte tidsbruk. Dersom noen estimerte veldig mange eller veldig få timer i forhold til resten av gruppen, åpnet det for diskusjon slik at alle kunne begrunne valget av antall timer. Deretter ble vi enige om hva som var den beste estimeringen. Etter alle oppgavene var estimert, ble ansvar fordelt for de ulike oppgavene i Sprint 2.

Daily stand-up 2

I Sprint 2 fortsatte teamet å kjøre daily stand-up hver mandag, onsdag og fredag kl. 09.15. Disse møtene ble knutepunktet for kontakt i en digital arbeidshverdag, og i denne sprinten gled derfor rammene litt ut. I stedet for å følge teorien og rammeverket til punkt og prikke, ble det tatt opp diskusjoner, demoer og andre ting som ikke hører til i en daily stand-up. Dette førte til at møtene ble lenger enn 15 minutter, og problemer som ble tatt opp ble også diskutert for å finne løsninger. Som teorien tilsier, skal det kun være et møte hvor man forteller hva som er gjort, hva man forplikter seg til å gjøre den aktuelle arbeidsdagen, samt hindringer som kan gjøre utførelsen av arbeidsoppgaver vanskelig. I henhold til teorien, skal det ikke skje problemløsning i en daily stand-up. Allikevel fungerte dette til en viss grad for gruppen, da det har bidratt til å gjenåpne dialog om problemet senere slik at man kan gå enda dypere inn i potensiell problemløsing.

Sprint Review 2

I dette møtet var scrum team, scrum master og produkteier til stede. Sprint Review 2 startet med å presentere backlog og gjennomførte arbeidsoppgaver for produkteier. Deretter ble det fremvist en demo av det foreløpige systemet. Dette ble gjort av scrum teamet. Produkteier kom med tilbakemelding og var meget fornøyd med fremgangen prosjektgruppen hadde hatt siden sist. Vi fikk en del tilbakemeldinger vi kunne ta med oss inn i neste sprint, i tillegg til litt idéer vi kunne jobbe videre med. I tillegg ønsket produkteier et fokus på administrasjonsrettigheter,

noe vi hadde fått beskjed om å ikke ha fokus på frem til dette møtet. Derfor måtte gruppen omstille seg noe i forbindelse med det endrede fokuset. Det ble også åpnet for spørsmål og diskusjon, og gruppen fikk vite mer om omfanget av systemet.

Møtet var kort og konkret, og prosjektgruppen kunne så gå videre inn i Sprint Retrospective 2.

Sprint Retrospective 2

I dette møtet ble igjen spørsmålene fra teorien sentrale. Vi diskuterte det som hadde gått bra, forbedringspotensial, samt utfordringer og hindringer. Tiltakene som ble iverksatt i starten av sprinten ble også diskutert.

Her kom det frem at gruppen var fornøyde med at det var flere arbeidsoppgaver, og derfor mindre tid til usikkerhet rundt hva vi skulle bruke tiden på. Dette førte til økt progresjon i prosjektet, som føltes positivt for scrum teamet. Det ble også avholdt en workshop i starten av sprinten, slik at de som hadde brukt tid på å lære seg det nye programmeringsspråket kunne lære det bort til resten av gruppen. Det kom på plass en del funksjonalitet i forbindelse med workshopen og økt arbeidsmengde, noe som skapte resultater og dermed motivasjon i gruppen. Videre var prosjektgruppen fortsatt godt fornøyd med daily stand-up, og ønsket å fortsette med dette tre dager i uken. Se Tabell 7 for oversikt over elementer fra ROS-analysen som gikk bra (se Vedlegg 3.7 for fullstendig tabell).

Det ble også diskutert hvor det var forbedringspotensial til de neste sprintene. Her ble det tatt opp at motivasjonsnivået var for lavt. Gruppemedlemmene snakket åpent om tap av motivasjon og for lavt kunnskapsnivå til å gjennomføre oppgaver de hadde fått ansvar for. Mye av grunnen til dette var blant annet at oppgavene ikke var brutt ned i små nok oppgaver, og derfor ble noe oppfattet som litt vagt. Dette var noe av den samme problematikken som dukket opp i Sprint 1, og selv om det hadde blitt en forbedring, var det fortsatt en vei å gå.

Enkelte medlemmer var derfor usikker på hvordan de skulle angripe oppgavene. Dette førte til at oppgaver ikke ble gjort på en tilfredsstillende måte, eller i det hele tatt. Her måtte vi derfor vurdere om det måtte settes inn ytterligere tiltak for å øke motivasjon og arbeidsinnsats. Videre diskuterte gruppen problematikk rundt å spørre om hjelp. Det var hendelser hvor gruppemedlemmer ba andre om hjelp, men dette ble ikke fulgt opp av noen av partene. Derfor ble det usikkerhet rundt dette, noe som førte til lavere grad av åpenhet om kommunikasjon i løpet av sprinten. Her diskuterte vi da muligheten for å sette inn en policy som skulle gjøre terskelen om å spørre om hjelp lavere.

Til slutt diskuterte vi hvordan sprint review hadde gått. Det var noe uenighet i hvordan gjennomgangen av systemet hadde gått for seg, og vi bestemte oss for å sette opp en generalprøve før sprint review fremover for å skape enighet om gjennomgangen av systemet. Som en del av dette punktet ble det også tatt opp at store valg måtte diskuteres med hele scrum teamet før det ble presentert på review for produkteier. Det ble diskutert om det kunne skje noen endringer for å optimalisere bruken av Azure DevOps som verktøy, blant annet å ha rekkefølger på oppgaver, samt noe omstrukturering. Scrum teamet diskuterte også mangel på oppdatering og bruk av ROS-analysen. Dette kan ha vært noe av grunnen til at gruppen hadde en del problematikk. Dersom de forebyggende tiltakene hadde blitt brukt mer aktivt gjennom

sprinten, kunne mye av problematikken vært unngått. Se Tabell 8 for oversikt over elementer fra ROS-analysen som gikk galt (se Vedlegg 3.8 for fullstendig tabell).

Det ble også tatt opp hvilke hindringer vi hadde hatt i løpet av Sprint 2. Noe som ble diskutert her var at oppgaver gjerne tok lenger tid enn det som var estimert fra Estimeringsmøte 2. En annen av de største utfordringene vi hadde hatt, var problemer med applikasjonen Docker. I tillegg har en pågående hindring vært Covid-19 som har gjort samarbeid mer utfordrende, noe som kom godt frem i Sprint 2.

Tabell 7: Hva har gått bra, Sprint 2

ID	Hva har gått bra?	Forebyggendetiltak fra ROS-analysen	Beskrivelse	Løst ✓ / ikke løst X
G2 Covid-19 begrenser muligheter for fysiske samarbeid i gruppen.	Digitalt samarbeid.	Opprettholde jevnlig kontakt digitalt. Følge retningslinjer for smittevernstiltak.	Gruppen har jobbet godt digitalt, til tross for at vi ikke har fått møttes fysisk.	✓
B7 Merge-konflikt i repository.	Gruppemedlemmene har vært flinke til å pulle ved arbeidsdagens start, samt følge rutiner som avtalt.	Alltid pull ved arbeidsdagens start.	Dette har skjedd, men siden gruppen har på plass rutiner som gjør at koden gjennomgås før den blir merget med main har ikke dette blitt et problem.	✓

Tabell 8: Hva har gått galt, Sprint 2

ID	Hva har gått galt?	Tiltaksplan fra ROS-analysen	Beskrivelse	Løst ✓ / ikke løst X
C1 Gruppemedlemmer vet ikke hva de skal gjøre.	Oppgavefordeling, og nedbryting av oppgaver.	Delegere oppgaver ved sprint planning.	Oppgavene som var satt opp i Sprint 2, var ikke detaljerte nok/små nok. Dette skapte forvirring og usikkerhet. Gruppemedlemmene har fått for store oppgaver som de ikke har nok kunnskap til å gjennomføre.	**X**
C5 Gruppemedlemmer leverer ikke forventede resultater.	Ikke nok kunnskap til å gjennomføre oppgaver.	Andre må gjennomgå arbeidet på nytt.	Henger sammen med C1. Gruppen leverte ikke forventet resultat, da de ikke hadde kunnskap til å gjøre oppgaver som ikke var godt nok definert i backloggen. Alle var innstilt på å forbedre dette til neste sprint.	**X**

6.4.4 Sprint 3

23. februar – 19. mars

Mål for Sprint 3: Lage ferdig all logikk og ha et førsteutkast av rapporten klart for veileder.

Sprint Planning 3

Sprint planning for tredje sprint ble gjennomført 23. februar. Her var det kun scrum team og scrum master som deltok. Etter vurderingene som ble gjort i sprint retrospective fra andre sprint, ble oppgavene i backloggen delt opp i mindre og mer detaljerte oppgaver i enda større grad enn tidligere, på en måte som motiverte alle allerede i planleggingsfasen av sprinten. Det ble også bedre tilrettelagt for at fordelingen av oppgaver skulle passe med den som hadde ansvar for oppgaven. Dette som en del av tiltaksplanen som var satt i ROS-analysen. Som en konsekvens av lavere motivasjon i forrige sprint, ble denne iverksatt. Tiltaksplanen sier følgende ved tap av motivasjon: «*Tildele arbeid (så langt det lar seg gjøre) som medlemmene synes er spennende*» (ID C7, Vedlegg 3.14). Dette ble i stor grad fokuset da arbeidsoppgaver ble brutt ned og fordelt i Sprint Planning 3.

Sprint planning avviker her fra teorien ved at produkteier ikke var til stede under møtet. Denne avgjørelsen ble tatt med bakgrunn i sprint review hvor det kom frem og ble diskutert med produkteier hva som var viktig å få på plass i neste sprint, samt at produkteier har noe begrenset tid. I tillegg er det avvik fra teorien hvor det skal velges ut en brukerhistorie som skal fullføres. Dette ble ikke gjort slik det fremstilles i teorien, men det ble heller valgt ut mindre oppgaver fra brukerhistoriene som skulle fullføres innen sprinten var ferdig. Dette er gjort bevisst da teamet har visse begrensninger i ressurser og kunnskap, noe som fører til at visse elementer må prioriteres fremfor andre.

Estimeringsmøte 3

Etter sprint planning gikk teamet over i et estimeringsmøte. Dette ble gjennomført på samme måte som i Sprint 1 og Sprint 2, da dette har vist seg å fungere godt for prosjektgruppen.

Daily stand-up 3

Daily stand-up ble hovedsakelig gjennomført på samme måte som i de foregående sprintene. Scrum master måtte likevel ta på seg ansvaret for å stramme inn på hva som var tillatt og ikke, og klarte å guide gruppen over i en struktur som stemmer overens med teorien. Dette innebar blant annet å holde problemløsing utenfor møtet, og holde tidsrammen på 15 minutter.

Sprint Review 3

Til stede i tredje sprint review var scrum team, produkteier, veileder fra UiA og sluttbruker. Møtet ble holdt på samme måte som i Sprint 2, hvor gruppen først viste frem backloggen. Her fikk møtedeltakerne se hvilke arbeidsoppgaver som var gjennomført i løpet av sprinten, i tillegg til en kort forklaring av backloggen og endringer som var gjort siden sist. Systemet slik det var til nå ble også presentert, og utviklerne fikk vise frem ferdig funksjonalitet. Til slutt ble det diskutert hvor fokuset skulle ligge fremover. Det ble avklart at grunnleggende funksjonalitet nå var på plass, og det som gjenstod var design, med fokus på brukerreise, samt fiksing av

programvarefeil og andre småfeil. Alle som deltok på møtet var fornøyde med fremgangen, og gruppen gikk videre inn i retrospective-møtet.

Sprint Retrospective 3

Sprint retrospective ble også avholdt på samme måte som tidligere, hvor møtet var sentralisert rundt de tre hovedspørsmålene fra teorien.

Først gikk prosjektgruppen gjennom det som var bra med Sprint 3. Gruppemedlemmene hadde en følelse av økt motivasjon som resultat av at mye i sprinten hadde gått bra. Vi hadde vært flinke til å registrere timer, som igjen ga scrum master økt motivasjon ved at hun slapp å minne medlemmene på dette. Programmeringsdelen av prosjektet var også på god vei og vi hadde fått stor fremgang her med å få på plass all grunnleggende funksjonalitet, noe som var målet med sprinten. Fra tidligere hadde vi tatt med oss bedre måter å dele inn arbeidsoppgaver, noe som gjorde at det var lite eller ingen usikkerhet rundt hva som var forventet av hver enkelt person og konkrete arbeidsoppgaver, som igjen førte til økt motivasjon, bedre humør og stor progresjon i alle deler av prosjektet. Vi hadde også vært mye flinkere til å bruke ROS-analysen aktivt gjennom sprinten. Til slutt diskuterte vi fordelene med at vi hadde sendt ut møteinnkallelse i god tid denne gangen i motsetning til Sprint 2 som resulterte i at flere deltok på møtet.

Neste spørsmål omhandlet forbedringspotensial for neste sprint. Det var færre ting som dukket opp enn tidligere, noe som tydet på at sprinten generelt sett hadde gått veldig bra. Vi diskuterte blant annet det med brukerreise som ble tatt opp på sprint review. Dette var noe vi måtte ha mye mer fokus på i neste sprint, og gruppen bestemte seg derfor for å avtale et møte med prosjekteier for å samkjøre dette med systemet. Det var også forbedringspotensial for brukerhistoriene. Det var et ønske om å dele disse opp i mindre historier slik at man lettere kunne tilpasse oppgaver til historier. Dette fordi systemet har mange funksjoner, men ikke alle dekkes av brukerhistoriene. Til slutt var det også ønske fra scrum master om å i større grad skrive ned nytt tidsestimat i Azure DevOps når man har jobbet med en oppgave, slik at det lettere kan visualiseres i en graf på et senere tidspunkt.

Avslutningsvis gikk vi gjennom eventuelle utfordringer eller hindringer gruppen hadde hatt i løpet av sprinten. Her diskuterte vi at kunnskapsnivået rundt Git var noe lavt, som gjorde at merging ble noe problematisk. I tillegg hadde det vært noe problemer med kodingen og å få alt ferdig med backend og logikken i systemet, noe som hang sammen med litt mangel på teknisk kompetanse. I tillegg var Docker et pågående problem, men som vi valgte å legge til side da det har blitt satt inn mange tiltak for å løse situasjonen allerede. Det var også noen hindringer rundt testing, og spesielt mot database, samt at Mac-brukere i gruppa fortsatt har problemer med å kjøre systemet. Generelt gikk det igjen at det var noe tekniske problemer, men mest mangel på kunnskap når systemet etter hvert krever litt mer enn det gjorde i starten. Tabell 9 viser et utdrag av den fullstendige tabellen som sier noe om hva som gikk bra i Sprint 3. Det samme gjelder Tabell 10 som viser et utdrag av hva som gikk galt i Sprint 3 (se Vedlegg 3.9 og 3.10 for fullstendige tabeller).

Tabell 9: Hva har gått bra, Sprint 3

ID	Hva har gått bra	Forebyggendetiltak fra ROS-analysen	Beskrivelse	Løst ✓ / ikke løst X
A11 Glemme å bruke ROS-analysen.	Bruk av ROS-analysen.	Sette ROS på agendaen i sprint retrospective	Gruppen brukte ROS-analysen jevnt under spriten.	✓
C11 Gruppemedlemmer logger ikke timer.	Timeregistrering i Azure og Microsoft Teams.	Påminnelse i stand-up møte	Gruppemedlemmer vært flinke til å logge timer. Scrum master har sendt noen påminnelser til gruppen i felles chatten.	✓

Tabell 10: Hva har gått galt, Sprint 3

ID	Hva har gått galt?	Tiltaksplan fra ROS-analysen	Beskrivelse	Løst ✓ / ikke løst X
B11 Docker fungerer ikke.	Docker fungerer ikke på MacOS.	Kontakte universitetet og forklare situasjonen. Arbeide videre med den status man sist var kjent med.	Gruppen har kontakt Egde og UiA. Vi har fått noe hjelp, men problemet er fortsatt til stede. Docker legges på is inntil videre.	X
D5 Har ikke nok kunnskap om database til å løse en oppgave.	Programmet vil ikke kjøre med Docker på MacOS.	Få hjelp av Egde.	Gruppen hadde møte med Egde, og fikk løst problemet. Vi fikk løst noe av problemet og kom oss litt videre. Programmet kjører ikke med Docker på MacOS	✓/X

6.4.5 Sprint 4

24. mars – 23. april

Mål: Bli ferdig med system og, så langt det lar seg gjøre, rapport.

Sprint Planning 4

Sprint Planning 4 ble gjennomført 24. mars med alle medlemmene i prosjektgruppen. Her diskuterte vi først hva målet med sprinten var, og deretter delte vi oss inn i tre grupper hvor hver gruppe hadde sine ansvarsområder. Disse var hovedsakelig backend, frontend og rapport/administrative oppgaver. Vi satt da i grupper i 45 minutter, før vi samlet gruppen igjen og gikk nøye gjennom hvert element i backloggen og hvem som skulle være ansvarlig for oppgaven. Her kunne vi da diskutere dersom det var uklarheter eller uenigheter.

Etter ca. 90 minutter var alle fornøyde med hvordan backloggen så ut og vi gikk over i estimeringsmøte. Møtet avviker fra teorien til Scrum, både ved at gruppen valgte å deles opp i mindre grupper, men også ved at produkteier ikke var til stede. Dette fordi gruppen har funnet ut at dette er den mest effektive måten å gjennomføre sprint planning på når det holdes digitalt.

Dette gjør også at alle medlemmene deltar aktivt hele veien og får en ansvarsfølelse og et eierskap til produktet, noe som fører til økt motivasjon.

Estimeringsmøte 4

Etter en nøye gjennomgang av backloggen i Sprint Planning 4, kunne vi ganske raskt estimere timer på de forskjellige oppgavene. Dette ble gjort på samme måte som i de foregående sprintene, hvor alle estimerte hvor mange timer en oppgave kom til å ta, før vi diskuterte sammen og ble enige. Dette var gruppen nå såpass gode på at møtet kun tok 30 minutter, noe som var en stor forbedring fra forrige sprint hvor det ble brukt opp mot 2 timer. Dette kan tyde på at gruppen har gjort en bedre jobb på å bryte ned oppgavene enn før.

Daily Stand-up 4

Daily stand-up for fjerde sprint var nå såpass godt innarbeidet i gruppens rutine, at det ble avholdt på samme måte hver gang i likhet med hvordan det ble gjennomført i Sprint 3.

Sprint Review 4

Under fjerde sprint review, var alle interessenter til stede. Dette inkluderte hele scrum teamet, veileder fra UiA, produkteier og sluttbruker. Review-møtet ble avholdt på samme måte som tidligere, hvor gruppen presenterte backloggen for sprinten, samt systemets nye funksjonalitet. Deretter åpnet det for diskusjon og tilbakemeldinger som ble notert ned av gruppen. Det var en del ting vi kunne ta med oss videre, blant annet litt detaljer i systemet, som skrivefeil, farger og lignende, som vi enkelt kunne fikse i neste sprint. Til slutt fortalte gruppen om planen for neste sprint, og det ble tatt opp utfordringer gruppen hadde hatt.

Sprint Retrospective 4

For fjerde retrospective, bestemte gruppen seg for første fysiske møte. Dette ble gjort i henhold til smittevern på offentlig plass. Her spiste vi lunsj sammen før retrospective startet. Som vanlig, ble de tre spørsmålene som står i teorien stilt. Dette startet med hva vi i gruppen hadde vært fornøyd med i Sprint 4. En av tingene som gikk igjen på dette punktet, var samarbeid i gruppa. Her diskuterte vi at det var høyt nivå av kommunikasjon gjennom hele sprinten og tett samarbeid. Dette førte så til at møtene var av høy kvalitet, gruppemedlemmene følte seg motiverte og det var generelt god flyt gjennom hele sprinten. Tabell 11 viser et utdrag av hva som gikk bra i Sprint 4 (se Vedlegg 3.11 for fullstendig tabell).

I tillegg var det enighet blant medlemmene om at det aldri var noen tvil om hva man skulle jobbe med, som tyder på godt beskrevet arbeidsoppgaver og bra arbeidsfordeling. Dette sørget så for god progresjon i sprinten, både i systemutviklingen av frontend og backend, men også i rapportskrivingen. Dette tyder på at tiltak som ble iverksatt etter Sprint 3 har fungert. Disse gikk hovedsakelig på utviklerdelen av prosjektet, hvor ting ikke fungerte eller at man ikke hadde nok kunnskap om nødvendige programmeringsoppgaver. Ettersom at gruppen nå var fornøyde med backend, tyder dette på at disse problemene var løst. Dette ved at gruppen har hatt kontakt og avtalt møter med Egde for å få bistand, samt at gruppemedlemmer har brukt tid på å sette seg inn i og gjøre research om det som trengs.

Det var også noe uenighet om hvordan programmeringen ble gjort i forrige sprint hvor noen gruppemedlemmer jobbet mer individuelt enn ønsket, som førte til problematikk for andre som ønsket samarbeid og samkjøring når det kom til kodestandarder og programmering. Dette ble løst ved å ha et «oppvaskmøte» hvor de det gjaldt fikk sagt det de hadde på hjertet, som løste situasjonen og forbedret situasjonen enormt til Sprint 4, hvor samarbeid og kommunikasjon har vært på topp.

Scrum master stilte så gruppen spørsmål om hva som kunne forbedres til neste og siste sprint. Et av gruppemedlemmene innrømte innledningsvis at han gjerne ville bli bedre på å registrere timene sine. Videre snakket vi om at det var estimert for mange timer for sprinten, og at vi ikke hadde brukt så mye tid som vi hadde forpliktet oss til. Det hadde heller ikke vært så mye fokus på ROS-analysen i Sprint 4, sannsynligvis fordi sprinten gikk såpass sømløst, at man ikke tenkte over dette på samme måte som tidligere. Allikevel ønsker vi å sørge for at vi alltid skal innom denne et par ganger i løpet av hver sprint slik at både analysen og matrisen er oppdatert.

Det ble også nevnt at vi ønsket høyere grad av samarbeid når det gjaldt rapport og rapportskriving, i form av digitale samlinger. Avslutningsvis kom det frem at flere ønsket at arbeidsoppgavene i DevOps skulle ha hver sin ansvarsperson. I utviklingsdelen, ønsket gruppen å bli flinkere til å legge inn kommentarer i koden. Tabell 12 viser et utdrag av hva som gikk galt i Sprint 4 (se Vedlegg 3.12 for fullstendig tabell).

Utfordringene i Sprint 4 var varierte. Utenfor gruppens kontroll var blant annet at Egde var trege på å svare når vi hadde tekniske spørsmål som var viktig å få på plass for å kunne gå videre i prosjektet. Dette ble diskutert med produkteier på Sprint Review 4, og det ble lovet fortgang i prosessen. Det hadde også vært mye press fra et annet fag i slutten av Sprint 4, noe som tok opp en god del av gruppens tid. Rent teknisk hadde ansvarspersonene fra frontend hatt problemer med Bootstrap og endring av skrifttype. Utfordringen knyttet til for mange estimerte timer, kunne ha en sammenheng med at gruppen kun registrerer effektive timer brukt, fordi vi har jobbet på hjemmekontor. Dersom gruppen hadde møtt opp fysisk på kontoret til Egde, ville man kanskje medregnet alle timer fra man ankommer kontoret til man reiser hjem.

Tabell 11: Hva har gått bra, Sprint 4.

ID	**Hva har gått bra?**	**Forebyggende tiltak fra ROS-analysen**	**Beskrivelse**	**Løst ✓ / ikke løst X**
A5 Personer deltar ikke i rapportskriving.	Deltakelse i rapportskriving.	Delegere oppgaver ved sprint planning.	Alle medlemmene i gruppa har deltatt i utarbeidelse av rapporten.	✓
B4 Ikke i stand til å programmere nødvendige funksjoner.	Kunnskapsnivå i programmering.	Sette seg inn i nødvendig programmering og rammeverk.	Etter hjelp utenfra i forrige sprint, økte kunnskapsnivået i gruppa og vi var i stand til å programmere nødvendige funksjoner.	✓

Tabell 12: Hva har gått galt, Sprint 4.

ID	Hva har gått galt?	Tiltaksplan fra ROS-analysen	Beskrivelse	Løst ✓ / ikke løst X
A11 Glemme å bruke ROS-analysen	Glemt å bruke ROS jevnlig.	Innkallelse til ROS-møte sendes ut.	Sprinten gikk såpass sømløst at ROS ble glemt litt bort. Gruppen ble klar over dette og sendte ut en innkallelse til et ROS møte i starten av Sprint 5.	✓
I2 Ikke nok kunnskap om testing.	Kunnskapsnivået på testing litt for lavt, kom i gang litt sent.	Spørre om hjelp via digitale plattformer.	Medlemmene kunne ikke nok om testing til å gjennomføre tilfredsstillende tester på systemet. Ikke vært tilgjengelige ressurser for hjelp. Ble satt opp møte med eksterne i Sprint 5.	X

6.4.6 Sprint 5

26. april – 7. mai

Mål: Ferdigstille rapport og system.

Sprint Planning 5

I Sprint Planning 5 deltok alle medlemmene i prosjektgruppen. I motsetning til Sprint Planning 4, holdt gruppen seg samlet gjennom hele møtet. Dette var forberedelse til siste sprint, og alle ønsket å diskutere det som var viktig og sette opp oppgaver sammen. Gruppen startet med å sette opp oppgaver som skulle med i backend-delen. Videre tok vi for oss frontend og til slutt rapporten.

Fra forrige sprint tok vi med oss et par elementer fra ROS-analysen for forbedring. Det ble lagt inn et ROS-møte i sprint backloggen, og alle fikk innkallelse til dette. Videre la vi til rette for å prioritere testing i høyere grad enn før. Avslutningsvis ble det diskutert hvordan vi kunne håndtere mangel på svar og/eller sene svar fra bedriften. Her kom gruppen frem til at det ikke var stort vi kunne gjøre, men ettersom grunnleggende funksjonalitet allerede var på plass, vurderte vi at konsekvensene av dette var såpass små at det ikke ville ha stor betydning for sluttresultatet. Derfor ble planen å fortsette som før med å purre på mail eller søke hjelp utenfra.

På slutten av sprint review, fordelte gruppen ansvar for hver enkelt oppgave i Azure DevOps. Dette ble gjort da det var uttrykt et ønske om dette i tidligere sprint. Da dette var gjort, var Sprint Planning 5 fullført.

Estimeringsmøte 5

Etter sprint planning, gikk gruppen over til å estimere hvor mange timer hver oppgave kom til å ta. Ettersom at det dukket opp noe tekniske feil i verktøyet DevOps, ble dette gjort på en litt annen måte enn før. Vi brukte ikke plug-in for tidsestimering slik som før, men estimerte manuelt for hver enkelt oppgave. Dette fungerte greit, og gruppen ble fornøyd med resultatet. Det ble estimert totalt 290 timer av 300 timer tilgjengelig for sprinten.

Daily stand-up 5

Daily stand-up i Sprint 5 ble i stor grad avholdt på samme måte som tidligere sprinter. Dersom møtene skled litt ut, var scrum master rask med å innhente samtalen igjen og minnet om at det var mulighet for å ta diskusjonen videre etter at stand-up var avrundet. Dette fungerte fint, og mange møter hadde slike diskusjoner etter stand-up, som kan ha bidratt til at kommunikasjonen var bra i Sprint 5.

Sprint Review 5

I siste sprint review, deltok alle interessenter i prosjektet. Scrum teamet presenterte det ferdige systemet for produkteier, sluttbruker og veileder. Møtet fulgte agendaen som var utstedt på forhånd. Møtet startet med en gjennomgang av backloggen, etterfulgt av en presentasjon av systemet. Det var åpent for å komme med tilbakemeldinger. Alle interessentene var svært fornøyd med hvordan systemet hadde blitt. Spesielt fornøyde var de med utseendet på systemet. Scrum teamet hadde notert ned tilbakemeldinger fra forrige review, og gjort de endringene som ble utpekt. Frontend var nå gjort på en måte som både produkteier, sluttbruker og scrum teamet var fornøyd med. Til slutt takket alle parter for samarbeidet, og det ble avtalt dato for overlevering av produktet.

Sprint Retrospective 5

Siste sprint retrospective ble avholdt digitalt over Microsoft Teams. Her diskuterte gruppen hva som hadde vært bra, hva som kunne forbedres og eventuelle utfordringer eller andre ting som dukket opp. Innledningsvis diskuterte gruppen hva som hadde vært bra med sprinten, og hvert medlem fikk mulighet til å si hva de synes. Generelt var gruppa svært fornøyd med både samarbeid og kommunikasjon, i likhet med Sprint 4. Videre synes alle medlemmene at frontend nå var blitt svært bra, og var stolte av det som ble presentert på Sprint Review 5. Det ble også diskutert at sprinten hadde vært mye kortere enn de foregående sprintene. Gruppen opplevde dette som positivt, da det oppmuntret til enda mer jevn arbeidsflyt enn tidligere og virket motiverende. Gruppemedlemmene satt igjen med et generelt inntrykk av at sprinten hadde vært effektiv, motiverende og produktiv. I tillegg synes gruppen at innsatsen på ROS-analysen var bra, hvor vi hadde hatt et detaljert møte og diskutert analysen slik den var, samt lagt til ting som eventuelt manglet. Tabell 13 viser et utdrag av hva som gikk bra i Sprint 5 (se Vedlegg 3.13 for fullstendig tabell).

Da spørsmålet om forbedringspotensial ble presentert, var gruppen i tvil om hvordan det var best å svare på spørsmålet. Dette fordi det var siste sprint, og det var usikkerhet rundt om vi skulle diskutere hva som kunne vært gjort annerledes i en hypotetisk neste sprint, eller om vi skulle snakke litt åpent om hva vi hadde lært gjennom prosessen. Vi kom frem til at vi ønsket å snakke om hva vi ville gjort dersom det hadde vært en ny sprint videre i prosjektet. Her var tilbakemeldingene at det kunne vært lurt å fortsette med korte sprinter på to uker, fremfor tre uker. Gruppen hadde heller ikke klart å knekke koden helt med oppgaver og timer, og alle trenger å øve mer på dette. Vi diskuterte oss frem til at det kunne vært fordelaktig å følge teorien i en større grad, til tross for at gruppen generelt har vært fornøyde med den versjonen vi har arbeidet med. På bakgrunn av dette, er det ikke utarbeidet en tabell for hva som har gått galt i Sprint 5.

Til slutt diskuterte prosjektgruppen hvilke utfordringer vi hadde stått ovenfor i sprinten. Scrum master hadde hatt utfordringer med timer og hvordan best få orden på disse og visualisere for rapportens del. Vi snakket også litt om hvordan vi skulle definere oppgaver som ferdig i Azure DevOps, ettersom at dette var den siste sprinten. Mye små detaljer skulle fortsatt fikses litt på, og derfor synes noen av medlemmene at dette var noe utfordrende.

Tabell 13: Hva har gått bra, Sprint 5.

ID	Hva har gått bra?	Forebyggendetiltak fra ROS-analysen	Beskrivelse	Løst ✓ / ikke løst X
A11 Glemme å bruke ROS-analysen	Gruppen har husket å bruke ROS-analysen gjennom sprinten.	Sette ROS på agendaen i sprint retrospective.	Etter end Sprint 4, hvor ROS-analysen var glemt litt bort, ble det satt opp et ROS-møte i Sprint Retrospective 4. Dette ble gjennomført i Sprint 5, og ROS var dermed integrert i sprinten igjen.	✓
A5 Personer deltar ikke i rapportskriving.	Deltakelse i rapportskriving.	Delegere oppgaver ved sprint planning.	Alle medlemmene i gruppa har deltatt i utarbeidelse av rapporten.	✓

6.5 Tidsstyring

Dette delkapittelet forklarer hvordan vi har styrt tiden i prosjektet. Dette innebærer å gjøre rede for gruppens tilgengelige ressurser og hvordan gruppen har estimert og reestimert timer. Vi vil også vise hvordan vi har registrert timer i burn down chart, samt vise en oversikt over alle planlagte og faktiske timer brukt.

6.5.1 Tilgengelige ressurser

For å finne tilgengelige ressurser startet gruppen med å se på hvor mange arbeidsdager vi hadde til rådighet før innleveringsfristen den 14.05.2021. Vi startet arbeidet i pre-sprinten 04.01.2021, og ved å trekke fra helger og påsken satt vi igjen med 89 arbeidsdager. Det neste vi gjorde var å se hvor mange timer det forventes at man skal arbeide hver uke. Dette prosjektet gir 20 studiepoeng, som har en forventet arbeidsmengde som tilsvarer 26 timer i uken per student. Gruppen forventer derfor at gruppemedlemmene skal arbeide 5 timer i snitt, hver dag. Figur 16 viser hvordan gruppen har regnet ut antall tilgjengelige timer.

antall timer hver dag * antall dager i prosjektet * antall gruppemedlemmer = tilgjengelige ressurser

5 * 89 * 6 = 2670

Figur 16: Tilgjengelige ressurser

6.5.2 Estimering

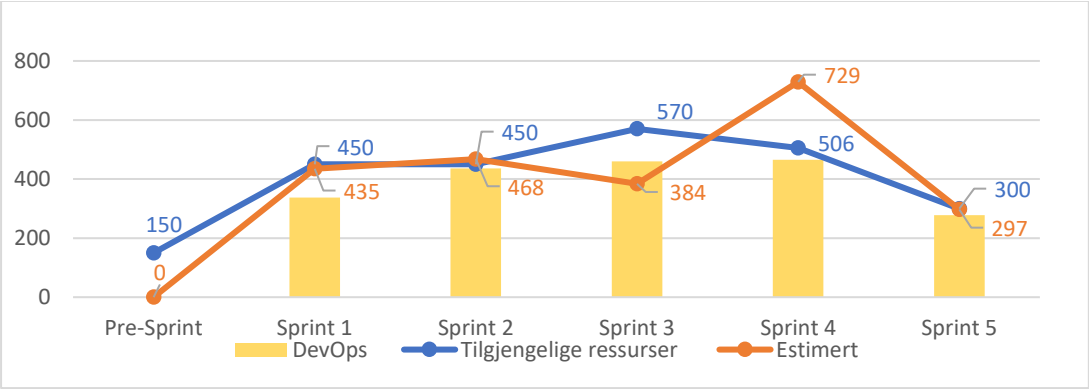
Som tidligere nevnt har gruppen etter sprint planning gjennomført estimeringsmøter. I estimeringsmøtet har gruppen spilt planleggingspoker ved hjelp av Fibonacci-sekvensen 1, 2, 3, 5, 8, 13, 20, 40, 100 (Roseman, 2019). Vanligvis representeres sekvensen som innsats, men gruppen har benyttet en hybrid løsning der vi har angitt sekvensen i timer. Estimeringen er gjort i DevOps ved hjelp av utvidelsen «Estimate».

Ifølge Mahnič & Hovelja (2012, s. 2087) er planleggingspoker en estimeringsteknikk som bygger på prinsippet om at flere hoder er bedre enn ett. De ansvarlige for å gjennomføre estimeringen er scrum teamet, og prosessen starter med at produkteier forklarer kravene til de ulike oppgavene. Videre diskuterer scrum teamet kravene i hver brukerhistoriene, hvor de skal ha mulighet til å stille spørsmål til produkteier. Da alle brukerhistoriene var nøye diskutert skulle teamet skrive ned eller velge et kort med forhåndsdefinerte verdier. Alle estimatene ble avslørt på samme tidspunkt, dette for å sikre at gruppemedlemmene ikke påvirkes av de andres meninger. Ved store avvik i estimeringen diskuteres dette ved forklaring av hvorfor verdien er valgt (Mahnič & Hovelja, 2012, s. 2087).

Tamrakar & Jørgensen (2012, s. 232) fant gjennom sin studie at utviklingsteam kan være mer treffsikre i estimater med bruk av Fibionaccitall enn ved bruk av lineære tallrekker. Videre påpeker studiet at man kan bli dratt mot et lavere estimat på oppgaver. Dette forekommer særlig i situasjoner hvor det er tendens til optimisme og der usikkerheten rundt estimatet er høyt. Det å ha god kunnskap om oppgavene kan redusere risikoen for underestimering. Dette medfører at de første iterasjonene i Scrum har størst risiko for feilestimering ved bruk av Fibonaccitall.

Gruppen startet med estimering av timer i Sprint 1, og gjentok samme prosess gjennom de øvrige sprintene. For å kunne estimere timer måtte alle i scrum teamet benytte «Estimate» i Azure DevOps for den aktuelle Sprinten. Når dette var gjort kunne gruppen starte å spille planleggingspoker.

Oppgavene i sprint backlog ble videre gjennomgått i tur og orden av scrum teamet ved at hvert enkelt medlem valgte det antall timer i Fibonacci-sekvensen de mente at oppgaven ville behøve for å fullføres. Da alle i gruppen hadde estimert hvor mange timer oppgaven ville ta, ble alle estimatene synlig for resten av gruppen. Store avvik i estimeringen måtte forklares og diskuteres. Figur 17 viser estimerte, tilgjengelige og registrerte timer for de ulike sprintene.



Figur 17: Estimerte, tilgjengelige og registrerte timer

Figur 17 viser at gruppen estimerte tett opp mot grensen for tilgjengelige timer i Sprint 1, 2 og 5. I disse sprintene har antall registrerte timer, illustrert med de gule stolpene i Figur 17, vært nært opp mot de estimerte timene. I Sprint 3 har gruppen estimert i underkant av 200 timer under ledig kapasitet. En av grunnene til dette kan være redusert motivasjon etter Sprint 2, samt manglende kunnskap om gjennomføring av oppgaver. Målet i Sprint 3 var å få på plass all logikk og ha ferdig et førsteutkast av rapporten. Under estimeringsmøtet til Sprint 3 var gruppen optimistiske med tanke på å nå sprintens mål. Dette fordi vi mente at utviklingen ville gå raskere gitt kompetansen tilegnet ved tidligere sprinter. Derfra fikk gruppen et kollektivt tankesett om at oppgavene ville ta mindre tid enn de faktisk gjorde. Dette er ifølge Tamrakar & Jørgensen (2012) vanlig når man blir litt overivrig eller har manglende kunnskap om oppgavene og hva de krever.

I Sprint 4 ble det derimot estimert i overkant 200 timer mer enn hva gruppen hadde kapasitet til. Gruppen hadde underestimert i Sprint 3, og forstod at oppgavene i denne sprinten ville kreve mer tid. Målet for sprinten var å gjøre seg ferdig med utlånssystemet. I motsetning til ved Sprint 3, fikk gruppen et kollektivt tankesett om at hver oppgave behøvde flere timer, som resulterte i overestimering.

I tillegg til å estimere timer på hver oppgave har gruppen estimert gjenstående tid ved påbegynte oppgaver. Definisjon av ETC er tiden som kreves for å fullføre en oppgave (Upland, u.å). Dette har vært en uformell forpliktelse gjennom prosjektet. I de ulike sprintene har gruppen delt seg inn i mindre grupper som igjen har hatt ansvaret for å fullføre oppgaver i de respektive sprintene. Innad i gruppene ble alle enige om hvor mye tid det er igjen på de ulike oppgavene ved å skrive kommentarer og ny ETC i «Time log» i DevOps. På denne måten har alle involverte i oppgaven oversikt over hvor lang tid det er igjen av en oppgave, se Figur 18.

Gått gjennom implementasjon av Azure AD med Magnus og pushe branch med fungerende AD. Ny ETC : 12
Fikk tips om å endre identity-type, så har arbeidet med multitenant
Forsøker å endre routing for AD sign-in/-out. Har ikke fått det til. Ny ETC: 10
Jobber med endpoints i kode for å hente inn dagens index-side til forside heller enn azure ad som kommer opp først nå. Ny ETC: 9
Etc 5

Figur 18: ETC

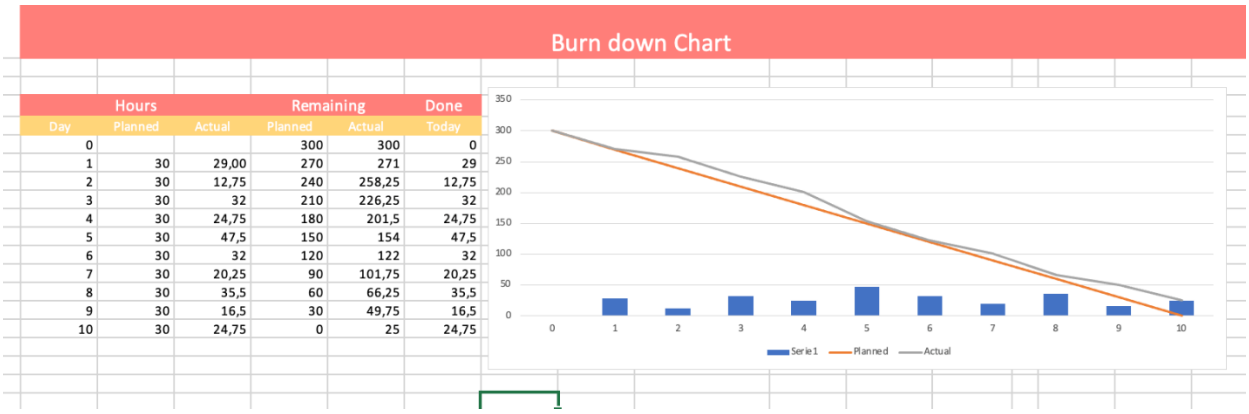
Dette har fungert godt da gruppen har tillit til hverandre, og medlemmene har vært flinke til å spørre om hjelp ved utfordringer eller manglende kunnskap. I tillegg oppdaterer vi hverandre om utfordringer i stand-up møtene. Dette igjen har ført til at gruppen har vært trygg på at vi klarer å bli ferdig i tide og levere avtalt produkt.

6.5.3 Burn down chart

Gruppen registrerte arbeidstimer i *burn down chart* (BDC) i Excel etter endt arbeidsdag. Hensikten med å bruke et BDC er å kunne gi informasjon om hvor mye tid som gjenstår i en gitt periode på en enkel måte. Det er tre typer BDC som er vanlig å bruke i Scrum. Det første dokumenterer fremgangen i sprintene, den andre viser fremgangen til utgivelsen og den tredje viser samlet prosjektfremdrift (Cervone, 2011, s. 21). Gruppen har brukt to typer BDC, en for

hver sprint og en for hele prosjektperioden. Før hver sprint har vi satt opp et BDC, hvor det er lagt inn antall dager i sprinten, og hvor mange timer gruppen har tilgjengelig basert på Figur 19.

Figur 19 viser BDC for Sprint 5. Timene som var planlagt i Sprint 5 er de tilgjengelige ressursene (forventet arbeidsmengde * 6 studenter * 10 dager) som er 300 timer, dette er fordelt på ti arbeidsdager, som tilsvarer 2 uker. De faktiske timene kommer fra en summering av alle gruppemedlemmenes timer per dag.



Burn down Chart

	Hours		Remaining		Done
Day	Planned	Actual	Planned	Actual	Today
0			300	300	0
1	30	29,00	270	271	29
2	30	12,75	240	258,25	12,75
3	30	32	210	226,25	32
4	30	24,75	180	201,5	24,75
5	30	47,5	150	154	47,5
6	30	32	120	122	32
7	30	20,25	90	101,75	20,25
8	30	35,5	60	66,25	35,5
9	30	16,5	30	49,75	16,5
10	30	24,75	0	25	24,75

Figur 19: Burn down chart Sprint 5

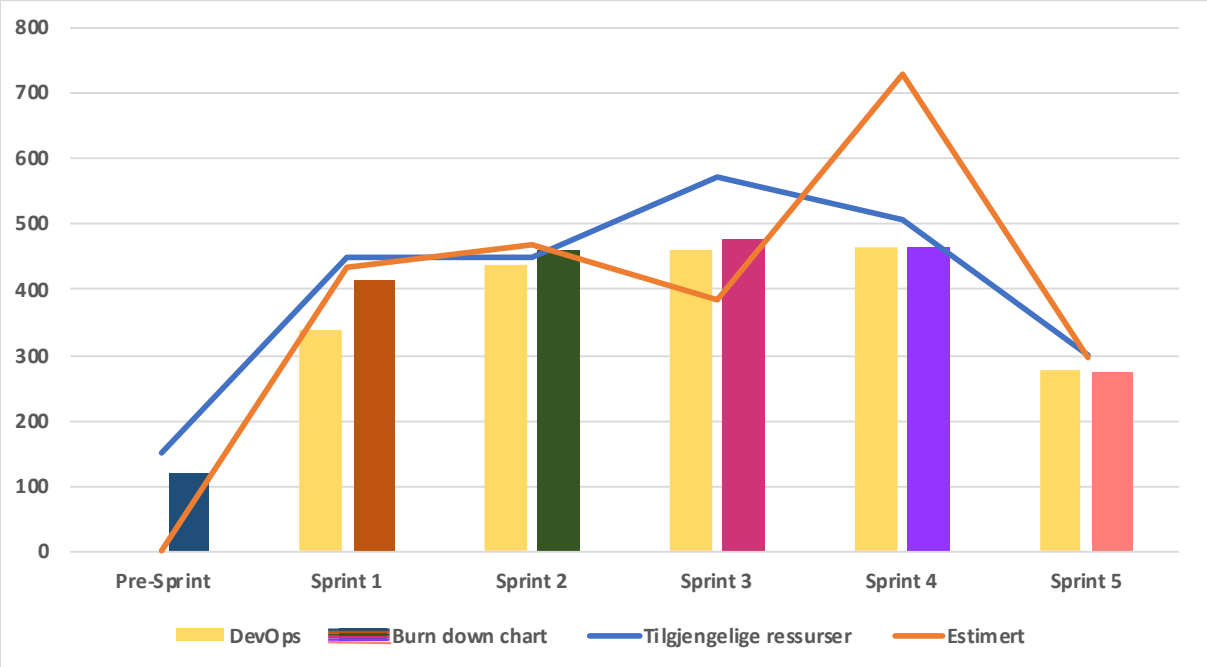
Den oransje linjen i grafen fra Figur 19 representere de planlagte timene, det vil si de tilgjengelige ressursene gruppen har hver dag i sprinten. Den grå linjen forteller hvor mange timer gruppen faktisk har jobbet og de blå stolpene representer tiden brukt for å løse oppgaver i sprint backlog. I Sprint 5 var hovedfokuset å gjøre ferdig frontend og skrive rapport. Figur 19 viser at den grå linjen ligger på linje eller litt over den oransje linjen, det vil si at gruppen har jobbet akkurat antall timer eller under antall timer som forventes for å fullføre sprinten. I Vedlegg 3.6, er den andre typen BDC som viser samlet oversikt over hele prosjektperioden. Grafen for hele prosjektperioden viser og at den grå linjen ligger på linje eller over den oransje linjen. I etterkant av Sprint 5 har gruppen diskutert hvorfor vi har jobbet færre timer enn forventet. Vi stilte oss spørsmål om vi ikke har arbeidet nok, eller om vi spart tid i forbindelse med digitalt arbeid, eller om det har vært andre årsaker. Gruppen mener at en grunn kan være mer digitalt arbeid hjemmefra som videre er et resultat av Covid-19.

Dette medførte at gruppen kun har registrert effektivt arbeid på hver oppgave i de ulike sprintene. Noen gruppemedlemmer har fortalt at de tidlig benyttet stoppeklokke for å være helt nøyaktige. I en normal hverdag ville sannsynligvis situasjonen vært annerledes, hvor tiden også ville gått til uformelle diskusjoner, samarbeid og kaffeprat. Dermed ville sannsynligvis timeantallet vært høyere.

6.5.4 Oversikt

Figur 20 viser alle timer registrert i DevOps, BDC, estimert og tilgjengelig. I pre-sprint hadde vi ikke valgt prosjektstyringsverktøy og følgende er det ikke registrert timer i DevOps. I Sprint 1 og 2 hadde gruppen noen avvik mellom registrerte timer i DevOps og BDC. Grunnen til dette er fordi gruppen deltok på forelesninger som ikke var en del av sprint backlog. I de øvrige

sprintene er timeantallet veldig likt mellom arbeidstimer registrert i DevOps og BDC. De øvrige avvikene i sprintene kan være et resultat av feil registrering fra medlemmene.



Figur 20: Estimerte, tilgjengelige, BDC og DevOps timer

Totalt i hele prosjektperioden har vi vært fornøyde måten vi håndterte tidsstyring. Vi har hele veien, selv med noen avvik, hatt kontroll på hva som må gjøres, hva som gjenstår og hva vi er ferdige med. Dette har vært en trygghet i perioder hvor det har vært hektisk med andre forpliktelser gruppens medlemmer har.

6.6 Risikohåndtering

I dette kapitlet vil kodestandard og risiko- og sårbarhetsanalyse presenteres som en del av prosjektets kvalitetssikringstiltak.

6.6.1 ROS-analyse

I en helhetlig risiko- og sårbarhetsanalyse (ROS), identifiseres uønskede hendelser som kan komme til å skje, og man beskriver risiko og sårbarhet forbundet med disse. Herunder vil de tre viktigste begrepene forbundet med ROS-analyse defineres: risiko, sannsynlighet og sårbarhet. Risiko er en vurdering av en hendelse som kan skje, hva konsekvensene kan bli og usikkerhet knyttet til dette. Sannsynlighet brukes som et mål på hvor trolig det er at en bestemt hendelse vil inntreffe, og sårbarhet utrykker de problemer et system kan få med å fungere når det utsettes for en uønsket hendelse, samt de problemer systemet får med å gjenoppta sin virksomhet etter at hendelsen har inntruffet (DSB, 2009, s. 15).

ROS-analysen (Vedlegg 3.14) som har blitt brukt av scrum teamet gjennom semesteret, ble først utarbeidet i Sprint 1. Videre i de påfølgende sprintene ble den oppdatert ved å legge til nye elementer, eller redigere eksisterende elementer ved behov. Analysen er blitt hyppig brukt, med unntak av Sprint 2, da rutinene ved oppdatering av analysen ikke var innarbeidet i

prosjektgruppens rutiner. Dette ble vi heldigvis bevisst på innen Sprint 2 var over, og analysen ble følgende benyttet i de neste sprintene.

Aktiv bruk av ROS-analysen er gjort som en del av kvalitetssikring, da forebygging av risiko kan bidra til at produktet og prosjektgjennomføringen holder høy standard. Følgende sikrer det også progresjon og høy kvalitet på leveranse underveis. Dette fordi vi hadde både forebyggende planer, samt tiltaksplaner slik at vi kunne sørge for at gruppen holdt seg på rett spor gjennom hele prosjektet. I hver sprint ble ROS-analysen tatt frem og gjennomgått, hvor alle elementers sannsynlighet og risiko ble vurdert på nytt. Ved behov ble det også lagt til nye hendelser. Vi var i tillegg opptatt av å se hvilke elementer vi hadde hatt nytte av og som gjorde at hver enkelt sprint gikk mer sømløst (se tabeller i kapittel 6.4).

ROS-analysen i Vedlegg 3.14 er delt inn i ulike kategorier der hver er definert med unik kategori-ID. Grunnen til dette er at vi ville samle alle risikoer som omhandler samme tema. Risikoer innenfor rapport og teori er kategorisert som A. B er utvikling, C er gruppefeil, D er database, E er samarbeid med bedriften, F er samarbeidsbedriften, G er andre feil, H er eksamen og I er testing. ID-ene sammen med sannsynligheten og konsekvensen ble lagt inn i risikomatrisen.

I hver sprint oppdaterte gruppen risikomatrisen (Vedlegg 3.15, 3.16, 3.17 og 3.18) etter justeringen i ROS-analysen. Matrisen beskriver sannsynligheten for at en hendelse kan inntreffe, samt hvor farlige konsekvensene av hendelsen potensielt blir. Matrisen viser en enkel oversikt over alle ID-er til de respektive risikoene. Eksempelvis var C7: «Gruppemedlemmer mister motivasjonen» (Vedlegg 3.15) satt til svært sannsynlig og utfallet ville være katastrofalt. Grunnen til at C7 ble karakterisert som en stor risiko var at gruppen var redd for at arbeidet ville stagnere, eller at noen gruppemedlemmer ikke ville bidra som avtalt. En annen grunn var pandemien, som begrenset oss til kun digitalt arbeid. Gjennom sprintene ble risikoen for dette redusert og utfallet ble mindre farlig. Dette er fordi gruppen har oppretthold god kommunikasjon gjennom hele prosjektperioden. I tillegg har gruppen motivert hverandre gjennom faglige interesser og løftet hverandre frem.

Et annet eksempel er B5: «At gruppemedlemmer ikke arbeider i seneste systemversjon». I Sprint 1 ble denne satt som lite sannsynlig og konsekvensen ble betraktet som lite farlig. I løpet av Sprint 2-4 ble sannsynligheten endret til meget, fordi systemet er i stadig utvikling og det arbeides med flere komponenter samtidig. Selv om sannsynligheten ble endret er fortsatt konsekvensen satt til lite farlig da gruppen har gode rutiner for sikkerhetskopiering. B5 har forekommet i Sprint 5, hvor sannsynligheten ble endret til svært høy, likevel gikk dette bra da gruppen kunne hente tapt data fra sikkerhetskopi.

6.6.2 Kodestandard

For å sikre høy kvalitet på koden som skulle leveres, ble det utarbeidet en kodestandard som skulle følges av alle gruppens medlemmer ved utvikling (Vedlegg 3.19). Dette er nyttig både for prosjektgruppen, men også for produkteier som skal overta produktet.

Ved at prosjektgruppen følger samme kodestandard, kan man minimere risikoen for at kodens kvalitet reduseres. Dette inkludere blant annet duplikasjon av kode, kobling, etc. Det blir også lettere for gruppemedlemmer å se over og kontrollere at andres kode fungerer som forventet.

Man kan enkelt gjennomgå kode som sparer gruppen for tid og ressurser. Egde vil også kunne spare ressurser ved bruk av kodestandarden. På denne måten blir koden blir lettere å vedlikeholde, og det vil også forenkle muligheten for å videreutvikle koden ved behov uten overforbruk av ressurser.

7 Refleksjon og konklusjon

Avslutningsvis vil gruppen konkludere og reflektere over prosjektperioden.

7.1 Konklusjon

Hovedmålet i dette prosjektet var å utvikle et reelt system. Dette har blitt gjort ved å utvikle en nettside hvor ansatte i Egde kan låne bedriftens utstyr, enten til profesjonelle eller private formål. Systemet oppfyller de viktigste kravene Egde presenterte for prosjektgruppen. Der «Must Haves» ikke er fullstendig fullført, har dette hatt med tilgang til ressurser å gjøre. Allikevel er alt lagt til rette for at produktet enkelt kan tilpasses og videreutvikles på disse områdene. Vi ønsker dermed å definere systemet som fullstendig, og at det oppfyller alle krav som det er blitt enighet med produkteier om. Disse har vært noe fleksible, som igjen understreker at den smidige arbeidsmetodikken som ble brukt var rett valg. Med fleksibel mener vi for eksempel at kravet om sikker innlogging har blitt noe modifisert, da vi ikke fikk de ressursene som var nødvendig for å kunne oppnå dette målet. Det ble derfor en enighet med produkteier at grunnarbeidet skulle gjøres, men at bedriften selv skulle implementere dette ved overtakelse.

I tillegg har prosjektgruppen valgt å fullføre brukerhistorier prioritert som «Shoud Haves». Dette for å imøtekomme så mange som mulig av forventningene til bedriften og oss selv. Det ble dermed størst mulig læringsutbytte for oss som studenter, og en bedrift som var svært fornøyde med innsatsen og produktet som skulle leveres. Dette systemet vil forenkle hverdagen til de ansatte hos Egde, og også hjelpe bedriften selv å holde oversikt over utstyr.

7.2 Refleksjon

Gjennom prosjektperioden har gruppen hatt mange opp- og nedturer. Alt i alt har det vært en lærerik prosess hvor gruppemedlemmene har fått føle på ansvaret om å skulle levere et ekte produkt til en reell bedrift. Dette er en prosess som åpnet opp for nettverksbygging, og potensielle fremtidige jobbmuligheter for studentene. Vi har fått muligheten til å føle på hvordan det er å bli ansatt av en bedrift og å skulle levere et produkt som må leve opp til forventninger. Gruppen har fått føle på mange utfordrende elementer. Det har blant annet vært utfordring med det tekniske, hvor ting ikke har fungert som det skal av forskjellige grunner. Det har også vært sosiale utfordringer, både det å være ny i en ellers tett sammenknyttet gruppe, og utfordringer knyttet til digital arbeidsplass og hjemmekontor.

Når det kommer til prosjektstyring, har vi hatt mye å lære. Til tross for at mye av det vi har jobbet med er kjent, har vi ikke mye erfaring annet enn et par fag fra tidligere år som har hatt noen like elementer. Dette har blant annet vært en utfordring ved strukturering av prosjektet, inndeling av oppgaver, estimering av timer og fordeling av ansvarsområder. Tidsestimering er noe vi har jobbet mye med, og selv om vi ikke er i mål, har vi i større grad fått god erfaring

med dette. Det vil fortsatt kreve mye arbeid før vi blir skikkelig gode på å estimere timer, men dette er en pågående læringsprosess som kun blir bedre av erfaring, noe dette prosjektet har gitt mye av.

Videre har det våren 2021 vært noen utfordringer knyttet til fysisk samarbeid. Det har vært en pågående pandemi som har forhindret gruppen fra å møtes internt for samarbeid, i tillegg til at vi ikke har hatt mulighet til å arbeide på kontorene til samarbeidsbedriften Egde. Dette har til tider vært tungt for gruppen, da alle hadde et ønske om å kunne jobbe sammen på kontoret for å få gode rutiner og i tillegg kunne aktivt bruke utstyrsrommet som systemet er basert på. Det har likevel vært et stort læringsutbytte knyttet til denne utfordringen. Selv om det til tider har vært lite motiverende å sitte isolert for seg selv, har gruppen gjort det beste ut av situasjonen ved å jobbe sammen over Microsoft Teams, både hele gruppen, men også i mindre grupper. Vi har lært at man i like stor grad kan jobbe effektivt sammen, uavhengig av lokasjon. Det eneste som virkelig har vært essensielt, var en fungerende datamaskin og god nettverkstilkobling.

Referanser

Ambler, *S*. (2004). *The Object Primer: Agile Model-Driven Development with UML 2.0*. Cambridge University Press.

Benyon, D. (2014). *Designing Interactive Systems: A comprehensive guide to HCI, UX and interaction design.* (3. utg.). Harlow: Pearson Education Limited

Bibik, I. (2018). *How to kill the Scrum Monster: Quick Start to Agile Scrum Methodology and the Scrum Master Role*. Montréal: Apress. DOI 10.1007/978-1-4842-3691-8

Carbonnelle, P. (2020). TOP IDE Top Integrated Development Environment index. Hentet fra https://pypl.github.io/IDE.html

Cao, J. (2020, 3. Februar). What is a Mockup: The Final Layer of UI Design. Hentet fra: https://www.uxpin.com/studio/blog/what-is-a-mockup-the-final-layer-of-ui-design/

Cervone, H.F. (2011), Understanding agile project management methods using Scrum, *OCLC Systems & Services: International digital library perspectives*, Vol. 27 No. 1, ss. 18-22. https://doi.org/10.1108/10650751111106528

Cohn, M. (2004). *User Stories Applied*: *For Agile Software Development.* Michigan: Addison-Weasley

Dimitrijević, S., Jovanović, J., Devedžić, V. (2015). A comparative study of software tools for user story management . *Information and Software Technology*. **57**: 352-368. doi:10.1016/j.infsof.2014.05.012

DSB. (2009). Veileder for kommunale risiko- og sårbarhetsanalyser. Lokalisert 06.02, 2014, fra: https://www.dsb.no/globalassets/dokumenter/veiledere-handboker-og-informasjonsmateriell/veiledere/veileder-til-helhetlig-risiko-og-sarbarhetsanalyse-i-kommunen.pdf

Egde. (u.å)a. Vi skaper bærekraftige endringer som gjør en forskjell. Hentet 16. mars 2021 fra: https://egde.no/om-oss/

Egde. (u.å,)b. Vi hjelper våre kunder med å realisere sine muligheter. Hentet 16. mars 2021 fra:

https://egde.no/hva-vi-gjor/

Gloger. V. B. (2010). *Your Scrum Checklist: Scrum Hard Facts.* InfoQ.com Hentet fra: https://www.infoq.com/minibooks/scrum-checklists/

Halvorsen, H. P (2020). Web Programming ASP.NET Core. Hentet fra: https://www.halvorsen.blog/documents/programming/csharp/textbook/aspnet/Web%20Programming%20-%20ASP.NET%20Core.pdf

Kendall, K. E. & Kendall, J. E. (2002). *System Analysis and Design.* (5. utg.). Camden: Pearson

Lucidchart. (u.å). What is an Entity Relationship Diagram (ERD)? Hentet 11. mai 2021 fra: https://www.lucidchart.com/pages/er-diagrams#section_2

Mahnič, V., & Hovelja, T. (2012). On using planning poker for estimating user stories. *Journal of Systems and Software*, 85(9), 2086-2095.

Mathiassen, L., Munk-Madsen, A., Nielsen, P. A. & Stage, J. (2018). *Object Oriented Analysis & Design*.(2. utg.). Metodica ApS: Hadsund

McKenna, D. (2016). *The Art of Scrum: How Scrum Masters Bind Dev Teams and Unleash Agility*. Aliquippa: Apress. DOI 10.1007/978-1-4842-2277-5

Microsoft (2016, 30. oktober). CRUD (Create, Read, Update, Delete). Hentet fra https://docs.microsoft.com/en-us/iis-administration/api/crud

Microsoft a. (2020). Choose between ASP.NET 4.x and ASP.NET Core. Hentet fra https://docs.microsoft.com/en-us/aspnet/core/fundamentals/choose-aspnet-framework?view=aspnetcore-5.0

Microsoft b. (2020). Introduction to Razor Pages in ASP.NET Core. (2020). Hentet fra https://docs.microsoft.com/en-us/aspnet/core/razor-pages/?view=aspnetcore-5.0&tabs=visual-studio

Microsoft a. (2021). DevOps-løsninger | Microsoft Azure. Hentet fra https://azure.microsoft.com/nb-no/solutions/devops/

Microsoft b. (2021). A Tour of C# - C# Guide. Hentet fra https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/

Microsoft c. (2021). Azure portal overview - Azure portal. Hentet fra https://docs.microsoft.com/en-us/azure/azure-portal/azure-portal-overview

Monk, A., & Howard, S. (1998). Methods & tools: the rich picture: a tool for reasoning about work context. *Interactions*. 5(2), 21-30. https://doi.org/10.1145/274430.274434

Radigan, D. (u.å.). Story points and estimation. Hentet 13. mai 2021 fra https://www.atlassian.com/agile/project-management/estimation

Roseman, I. (2019, 4. januar). Microsoft's Azure devops - Planning poker estimation tool. Hentet fra: https://www.rosemansolutions.com/wordpress/2019/01/microsofts-azure-devops-planning-poker-estimation-tool/

Sandnes, F.E. (2018) Universell utforming av IKT-systemer. Brukergrensesnitt for alle (2. utg). Oslo: Universitetsforlaget

Smith, S. (2017, september) Simpler ASP.NET MVC Apps with Razor Pages. MSDN Magazine Issues. Hentet fra https://docs.microsoft.com/en-us/archive/msdn-magazine/2017/september/asp-net-core-simpler-asp-net-mvc-apps-with-razor-pages

Tamrakar, R., & Jørgensen, M. (2012). Does the use of Fibonacci numbers in Planning Poker affect effort estimates?. *16th International Conference on Evaluation & Assessment in Software Engineering.* IET, 228 – 232. DOI 10.1049/ic.2012.0030

Upland. (u.å) Estimated Time To Complete (ETC). Hetet 11. mai 2021 fra: https://uplandsoftware.com/psa/resources/glossary/estimated-time-to-complete-etc/

Varma, T. (2015). *Agile Product Development*. New York. Apress. DOI https://doi.org/10.1007/978-1-4842-1067-3

Wikipedia. (2020, 25. november). E/R-modell. Hentet fra https://no.wikipedia.org/wiki/E/R-modell

Wikipedia. (2021, 5. mai). MoSCoW method. Hentet fra https://en.wikipedia.org/wiki/MoSCoW_method#cite_note-1

xUnit.(u.å) About xUnit.net. Hentet 28.04.2021 fra: https://xunit.net/

Zzzprojects (u.å) Entity Framework Migration. Hentet 8. Mai 2021 fra: https://entityframework.net/migration

Vedlegg

Vedlegg 1 Uttalelse fra Egde

Egde

Uttalese fra Egde

Høsten 2020 arrangerte Universitetet i Agder RefreshIT på Radisson Blue Caledonien Hotel i Kristiansand. Her møtte flere lokale aktører opp for å presentere seg selv for studentene, for så å legge frem hvorfor studentene skulle samarbeide med de representative selskapene. Egde ble gjennom arrangementet en populær aktør hvor mange studentgrupper hadde lyst å samarbeide med oss. Gjennom en intervjurunde og en intern prat falt Egdes valg på Go4IT bestående av Linn, Erik, Marine, Julie, Magnus og Kantasit.

Egde er en bedrift i vekst og dette har resultert i et behov for å holde kontroll på teknisk utstyr bedriften eier. Dette problemet har Go4IT løst ved å utvikle et utlånsystem for Egde.

Ved prosjektoppstart ble det kjent at studentgruppen ønsket å holde en så profesjonell og realistisk dialog og samhandling som mulig. Derfor har Andreas Taraldsen (Egde) vært produkteier og Robert Quidding (Egde) vært sluttbruker. Egde har gjennom prosjektet støttet studentgruppen med teknisk ekspertise innen skyplattform, test, Docker og annen teknologi for å hjelpe studentene på veien.

Gruppen har fremstått veldig profesjonelle, med strukturerte sprint review møter hvor Julie har styrt møtene på en god og effektiv måte. Det har aldri vært tvil om hva backloggen inneholder og hvilke krav som brukerhistoriene inneholder. Rollene innad i gruppen har vært enkle å forholde seg til og de har gjennomført prosjektet på en måte Egde ser seg igjen i.

Go4IT har vist en god teknisk forståelse og nysgjerrighet ovenfor fremmed teknologi. De fremstår som meget sterke på det tekniske. Løsningen gruppen har utviklet inneholder flere ulike teknologier som de har måtte sette seg inn i og lære.

Det var alltid et håp om at gruppen kunne benytte seg av Egdes kontorlokaler for arbeid, og på den måten lære mer om bedriften og hvordan vi arbeider. Dessverre har situasjonen med COVID-19 og restriksjoner gjort dette umulig. Likevel har kommunikasjonen vært effektiv på både epost og Microsoft Teams.

Egde er en profesjonell aktør som verdsetter kvalitet i arbeid og metodikk. Her har Go4IT levert som ønsket og Egde er stolte av å kunne være en del av denne bacheloroppgaven med studentene.

Grimstad, 12.04.2021

På vegne av Egde

Andreas Taraldsen

Vedlegg 2 Egenvurdering

Go4IT

Som nevnt innledningsvis har Go4IT både nye og gamle gruppemedlemmer. Det har vært viktig å bli godt kjent med hverandre samt opprettholde god kommunikasjon gjennom hele prosjektet. Gruppen hadde et felles mål om å levere et fungerende system og utarbeide en fullsteding rapport. Medlemmene har høye ambisjoner og stiller krav om alle i gruppen stille forbedret, delta må møter, tar ansvar for oppgaver osv. I tillegg har alle medlemmene jobbet med alle deler av dette prosjektet, deltatt på Scrum-møter samt motivert og hjulpet hverandre. Vi har ulike interesseområder innen systemutvikling, likevel har gruppens medlemmer bidratt i alle deler av prosjektet.

Linn Engedalen Fredriksen

I dette prosjektet har jeg hatt rollen som scrum master sammen med Julie, og hatt ansvaret for tidsstyring, risikovurdering samt prosjektplanlegging og -gjennomføring. Jeg har booket inn til diverse scrum-møter, hatt dialog med Egde, skrevet referater og bistått der det trengtes. Jeg har både jobbet med frontend og backend av systemet, men mitt hovedansvar har hele veien vært prosjektstyring. Dette er en rolle jeg har trives svært godt, siden fagområdet interessere meg.

Kantasit Intaraphasuk

Jeg har vært en del av scrum teamet og hatt det overordnede ansvaret for frontend sammen med Marine. I tillegg har jeg sørget for at systemet har et godt design og er brukervennlig, i samsvar med både designprinsippene, krav fra Egde og interne valg. Jeg har tilegnet meg betydelig kunnskap innenfor ASP.NET Core og Razor Pages som har kommet til nytte under utviklingen av backend i systemet.

Julie Hedemark

Min rolle i dette prosjektet, har hovedsakelig omhandlet den administrative delen med spesielt fokus på prosjektgjennomføring og prosjektstyring. I samarbeid med Linn, påtok jeg meg rollen som scrum master, og ansvaret som tilhørte denne rollen. Dette kunne eksempelvis være å sette opp og kalle inn til møter, både internt og eksternt, men også å finne løsninger på eventuelle utfordringer som dukket opp. Jeg bidro også til programmering i startfasen av prosjektet, samt hjalp til senere ved frontend-utvikling.

Magnus Erdvik

I dette prosjektet har jeg hatt ansvar for backend delen av systemet med Erik, samt hatt ansvar for GitHub Repository (git-master). Jeg har brukt mye tid på å sette meg inn i ASP.NET Core, Razor Pages og Entity Framework. Denne kunnskapen ble brukt til å bygge en solid grunnmur til systemet vårt, samt være til hjelp for andre i gruppen ved tekniske spørsmål. I løpet av prosjektet har jeg funnet ut at backend utvikling er noe jeg liker, og at det er noe jeg ønsker å jobbe med videre. Det har også vært en god erfaring å jobbe med en tverrfaglig gruppe der alle har ulike styrker. Jeg er svært fornøyd med innsatsen til gruppen og det ferdige produktet.

Erik Danielsen Nergaard

Jeg har i dette prosjektet vært en del av scrum teamet, hvor jeg har hatt ansvaret for implementering av backend med Magnus. Dette har krevd at jeg har måttet tilegne med kunnskap om C# og ASP.NET. Videre har jeg deltatt i utarbeidelsen av prosjektets kodestandard og kvalitetssikring av git-masters kode etter behov. Videre har jeg arbeidet mye med oppsett for skyløsninger og sikker innlogging ved bruk av SSO. Jeg har også bistått ved oppsett av modeller ved analyse og design.

Marine Ramsdal

I dette prosjektet har jeg vært en del av scrum teamet, med hovedfokus på frontend som mitt ansvarsområde. Her har jeg arbeidet i tett med Kantasit, hvor vi har utviklet utseende på systemet slik at det skulle stemme overens med det som var avtalt. I tillegg har jeg bidratt i backend-utvikling i startfasen av prosjektet, og bidratt der det har vært behov for bistand. Blant annet har jeg hatt hovedansvaret for å korrekturlese rapporten i sluttfasen av prosjektperioden.

Vedlegg 3 Andre relevante vedlegg.

Vedlegg 3.1 Styringsgruppemøte 1

Styringsgruppemøte 1

Innkallelse:

Sted: Digitalmøte på Microsoft Teams.
Dato: 28.01.2021
Tid: 12:00-13:00
Agenda:

- Sprint review
- Status på prosjektet
- Demo
- Plan for neste sprint

Invitert:

- Prosjektgruppen Go4IT
- Produkteier
- Sluttbruker
- Veileder

Sprint Review 1

Oppmøtte:

Go4IT		Egde		UiA	
Navn	**Rolle**	**Navn**	**Rolle**	**Navn**	**Rolle**
Julie	Scrum master	Andreas	Produkteier	Geir Inge	Veileder
Linn	Scrum master	Robert	Sluttbruker		
Erik	Scrum team				
Magnus	Scrum team				
Kantasit	Scrum team				
Marine	Scrum team				
Erik					

Overordnet status på prosjektet

Julie gikk igjennom sprint backlog med oppgaver som har blitt gjort i løpe av Sprint 1. Her viste hun hvordan Go4IT administrerer oppgaver samt hvordan vi registrerer timer. Oppgaver som har hatt størst fokus i denne sprinten er å lage en klikkbar prototype av systemet samt lære oss ASP.NET Andre oppgaver som har blitt gjennomført i Sprint 1 er rikt bilde, mockups, klassediagram, skisser, navigasjonsdiagram, ER-diagram, sekvensdiagram, ROS-analyse med kvalitetssikringstiltak, brukerhistorier, prioritering av kriterier.

Demo av systemet

Julie viste frem den klikkbare prototypen. Prototypen var sendt ut til produkteier og sluttbruker i forkant av møtet.

Etter demoen fikk gruppen følgende tilbakemeldinger:

- Andreas: Det må være unik ID på alle produkter som skal legges til.
    - Go4IT svarer: Planen er at ID-en til hvert produkt skal bli autogenerert.
- Geir Inge: Hvem skal ha ansvar:
    - Andreas svar: På dette tidspunktet skal systemet utvikles uten at noen skal ha ansvar for det. Vi ønsker ikke å låse systemet til en ansvarlig.
- Andreas: Navigasjonsbaren på toppen – denne trenger dere ikke. Videre vil dere fokuserer på administrasjonsrettigheter på et senere tidspunkt.

- Robert: Synes det ser greit ut, noe rart med plassering av knapper, men dette kan vi endre på et senere tidspunkt. Skille f.eks. kamera av samme type med ID
- Geir Inge: Har dere i Egde en standard som prosjektgruppen burde følge?
    - Andreas svar: Vi har noen grunn prinsipper med bruk av farger og tekst, jeg skal sende dette til prosjektgruppen.

Møtet fortsatte med en diskusjon rundt ASP.NET Core.
Go4IT har valgt å bruke ASP. NET Core. Produkteier er fornøyd med dette, men kom med et tips at gruppen ikke skal «legacy». Robert mente det er lurt å ikke velges det nyeste da det er enklere å finne god dokumentasjon på nett med de litt elder versjoner. Videre forteller Erik at vi har planlagt og bruker Razor Page og docker. Egde stilte seg veldig positive dette.
Side del av møte ble en diskusjon rundt kalenderfunksjonen da Geir Inge spurte om Egde ville ha kalenderoverføring, slik at produktet til lagt i Outlook-kalenderen. Dette er noe Egde ikke ønsker da de vil at systemet skal lage lite støy og være enkelt å bruke.

Videre plan
Det ble nevnt at Go4IT skal kjøre sprint retrospective etter endt møte og Sprint Planning 2 førstkommende uke. Her ønsket Robert å være med på planleggings- og estimeringsmøte for å sikre at gruppen gjorde dette på en god måte. Gruppen fortalte at vi ønsket å fokusere å bygge grunnsteiner, og få på plass en del funksjonalitet i Sprint 2.
Aksjonspunkter
1. Andreas sender designdokument til Go4IT (så fort som mulig)
2. Julie sender møteinnkallelse til Robert på Sprint Planning 2 (så fort som mulig)

Tilleggsinformasjon:
Gruppen har ikke sendt ut referat til møtedeltakerne da det er en mange aksjonspunkter og det var klarhet i møtet. Vi kommer til å sende referat fra de andre styringsgruppemøtene.

Vedlegg 3.2 Styringsgruppemøte 2

Styringsgruppemøte 2

Innkallelse:

Sted: Digitalmøte på Microsoft Teams.
Dato: 19.03.2021
Tid: 14:00-15:00
Agenda:

- Sprint review
- Status på prosjektet
- Demo
- Plan for neste sprint

Invitert:

- Prosjektgruppen Go4IT
- Produkteier
- Sluttbruker
- Veileder

Sprint Review 3

Oppmøtte:

Go4IT		Egde		UiA	
Navn	**Rolle**	**Navn**	**Rolle**	**Navn**	**Rolle**
Julie	Scrum master	Andreas	Produkteier	Geir Inge	Veileder
Linn	Scrum master	Robert *	Sluttbruker		
Erik	Scrum team				
Magnus	Scrum team				
Kantasit	Scrum team				
Marine	Scrum team				
Erik	Scrum team				

*25 min forsinket.

Overordnet status på prosjektet

Backlog til prosjektgruppen ble presentert av Kantasit, dette for å gi innsikt i hva som var fullført i sprinten og hva som eventuelt manglet

Demo av systemet

Magnus ferdig funksjonalitet til nå i systemet. Han gikk gjennom følgende:

- innlogging
- forskjell på admin- og brukervisning i systemet
- hvordan legge til nytt utstyr, redigere utstyr og slette utstyr
- booking av produkter

Tilbakemeldinger på systemet og backlog:

Andreas: synes det ser veldig bra ut nå, grunnleggende funksjonalitet er på plass. Kan fokusere mer på frontend og design, samt brukerreise fremover.

Robert: synes det ser veldig greit ut, noe pirk på utseende til systemet.

Kort diskusjon om testing, bør gjøres for gruppens egen del samt for å sikre kvalitet. Plan videre:

Videre plan

Fremover skal gruppen fokusere på frontend (brukerreise, design osv.), fikse programvarefeil, og generelt gjøre systemet så ferdig som mulig innen Sprint 4 er over (23. april). Muligens prøve å hoste systemet og dytte det opp i skyløsning.

Aksjonspunkter

1. Andreas understreker at gruppen kan ta kontakt underveis dersom det skulle dukke opp utfordringer/problemer/spørsmål.
2. Geir Inge får tilsendt førsteutkast på rapport mandag 22. mars.
3. Andreas skal høre om gruppen kan få noen ressurser i Azure.

Vedlegg 3.3 Styringsgruppemøte 3

Styringsgruppemøte 3

Innkallelse:

Sted: Digitalmøte på Microsoft Teams.
Dato: 23.04.2021
Tid: 10:00-11:00
Agenda:

- Overordnet status
  - Backlog, oppgaver og timer brukt
  - Evt feedback på backlog
- Demo av systemet per nå
  - Feedback på systemet, innspill/kommentarer
- Plan for neste sprint
  - Hva blir fokus i neste sprint?
  - Feedback/kommentarer
  - Evt diksjon
- Annet (utfordringer, m.m.)

Invitert:

- Prosjektgruppen Go4IT
- Produkteier
- Sluttbruker
- Veileder

Sprint Review 4

Oppmøtte:

Go4IT		Egde		UiA	
Navn	**Rolle**	**Navn**	**Rolle**	**Navn**	**Rolle**
Julie	Scrum master	Andreas	Produkteier	Geir Inge	Veileder
Linn	Scrum master	Robert	Sluttbruker		
Erik	Scrum team				
Magnus	Scrum team				
Kantasit	Scrum team				
Marine	Scrum team				
Erik	Scrum team				

Overordnet status på prosjektet

Backlog til prosjektgruppen ble presentert av Kantasit, for å gi en innsikt i hvordan gruppen lå an med oppgaver og hva som var fullført i sprinten siden sist

Demo av systemet

Systemet, slik det er nå, presentert av Magnus. Han viste frem følgende som var nytt siden sist:

- Frontend
  - Nye farger fra Egdes merkevarebok
  - Endring av tekststørrelse
  - Litt endret stil/layout på siden
- Utstyrsliste
  - Lagt til modellnummer/serienummer

    - Fjernet «info»
- Booking
    - Endret layout/utseende
    - Etter booking: sendes nå tilbake til utstyrslisten som ble diskutert sist.
- Historiske bookinger kan bookes på nytt.
- Fjernet minutt/sekundviser fra dato: ser ryddigere ut.

Tilbakemeldinger på systemet og backlog:

- Ikke fokuser mer på Bootstrap, har tatt for mye tid. Bruk alternativer til Bootstrap.
- Legg tiden der det trykker mest.
- Går fremover, ser ganske bra ut nå. Ikke nødvendig med mer funksjonalitet.
- Knappen «Fjern» har samme design som knappen for «Book». Skaper forvirring, vurder å
- endre fargekode på knappen. F.eks. en av Egdes andre farger.
- Veldig bra at det er brukt tid på backend i forberedelse til overlevering. Settes pris på av Egde.
- Trengs en opprydding i skrift og språk i systemet, samt justere slik at alt ting på linje
- (allignment).
- Luke bort feil, fjerne alt som ikke skal være synlig for brukeren (userview/adminview).
- Kolonnen i utstyrslisten som heter Modell|S/N må få nytt navn, ser rotete ut.
- Søk opp universell utforming og se om deres system er i tråd med dette.
- Husk å teste for litt absurde ting og hva slags påvirkning det har på systemet.

Videre plan

Bli fullstendig ferdig med system og rapport. Siste sprint, innlevering av rapport 14. mai. Dette blir hovedfokuset.

Tips fra Geir Inge: Husk å få med alt av kriterier og argumentasjon bak de ulike elementene. Kan si at kunden ønsker systemet slik, men likevel argumentere for hva som ville vært best ifølge teorien. Reflektere på ulikheter.

Aksjonspunkter

1. Linn sender informasjon om Egde sin uttalelse i rapporten
2. Erik kontakter Robert angående testing
3. Julie sender rapporten til Geir Inge
4. Geir Inge gir gruppen feedback hva som kan gjøres bedre i rapporten

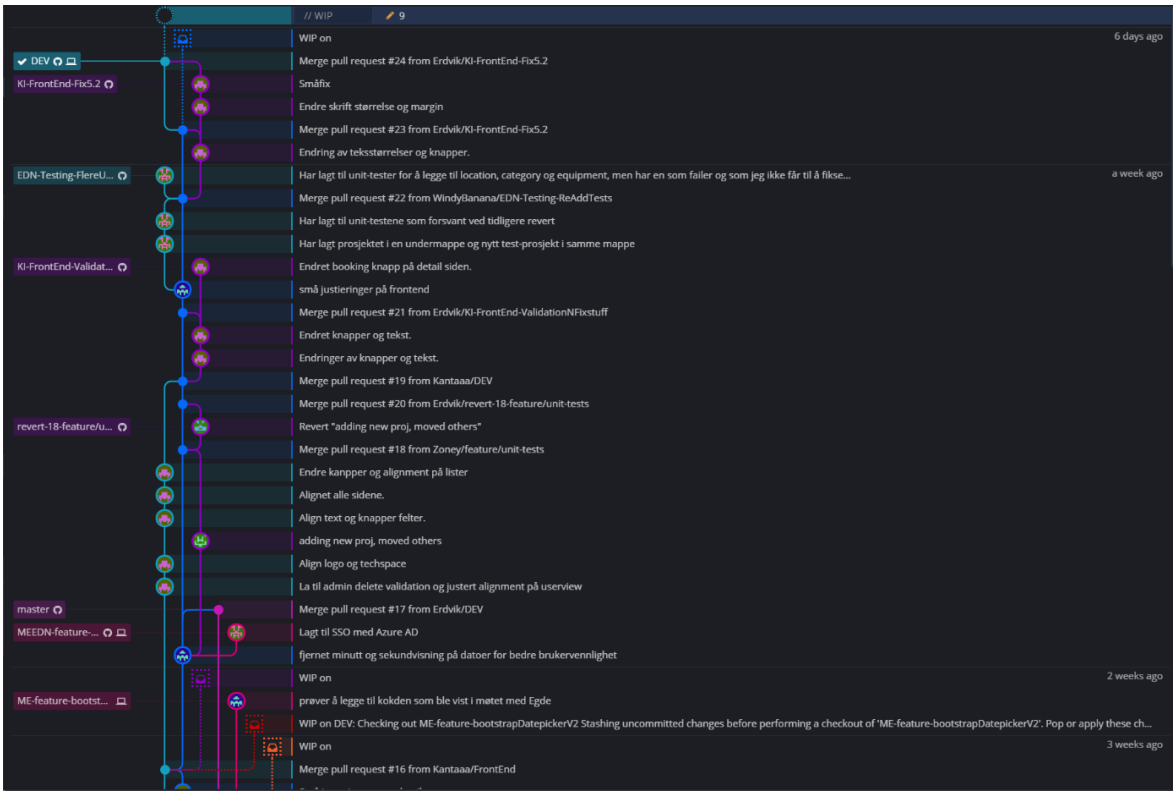
Vedlegg 3.4 Brukerhistorier

ID	Brukerhistorie	Beskrivelse	Kriterier	Argument	Prioritet
5	Som bruker, ønsker jeg å få oversikt over hvem som har utstyr til enhver tid, slik at jeg slipper å lete og spørre rundt.	Mulighet for enkeltpersoner i bedriften å finne ut hvor utstyr befinner seg til enhver tid.	Systemet skal ha en funksjon som gir oversikt over hvor utstyr befinner seg. Bruker forespør informasjon om et produkt, systemet svarer forespørselen ved å sende brukeren til infosiden til utstyret hvor det presenteres hvor det befinner seg. Det skal fremkomme om utstyret er på kontoret (Grimstad/Kristiansand), eller hos en ansatt.	Denne funksjonen klassifiseres som «Should Have» fordi det er kjekt for brukere å slippe og lete etter utstyr, men ikke nødvendig for at systemet skal kunne tjene sitt formål.	*Should Have*
6	Som bruker vil jeg kunne filtrere utstyr etter forskjellige kriterier når jeg skal booke, slik at jeg enkelt finner produktet jeg søker etter.	Mulighet for å legge inn filter ved søk på utstyr i utstyrslisten på nettsiden. Kriteriene kan inkludere lokasjon, kategori, osv.	Systemet bør ha en funksjon som tillater brukeren å legge inn kriterier når de søker på utstyr på nettsiden. Brukeren kan velge kategori og lokasjon, og systemet vil da filtrere bort alt utstyr som ikke stemmer med kriteriene som er valgt.	En funksjon som er prioritert som «Should Have». Dette fordi en slik funksjon ikke er nødvendig for at systemet skal kunne kjøre, men som gjør brukeropplevelsen mye bedre for sluttbruker da det forenkler prosessen med booking.	*Should Have*
7	Som bruker vil jeg kunne søke etter utstyr ved utstyrsnavn, slik at jeg enkelt kan finne et spesifikt produkt.	Mulighet for å søke med fritekst i utstyrslisten på nettsiden. Bruker kan søke etter navnet på produktet.	Når bruker er logget inn, og har navigert seg til utlånssiden, kan vedkommende skrive inn utstyrets navn i søkefeltet. Systemet vil da oppdatere listen med utstyr fortløpende slik at utstyret som vises matcher ordet i søkefeltet, og systemet vil filtrere bort alt utstyr som ikke stemmer overens med søkeordet.	Funksjonen for søk på utstyrsnavn, klassifiseres som «Should Have» fordi den gjør systemet mer brukervennlig for bruker, men det er ikke en kritisk funksjon for at systemet skal fungere i henhold til kravspesifikasjonene.	*Should Have*
8	Som bruker ønsker jeg et intuitivt og brukervennlig system, slik at booking kan skje så lett som mulig.	Et bruker-vennlig system som oppfyller krav fra Egde, samt universell utforming for tilgjengelighet og generell enkelhet.	Når en bruker er innlogget, vil det ikke være tvil om hvordan systemet fungerer, inkludert navigasjon for å oppnå mål, hvor det går an å trykke og ikke, systemet gir feedback dersom noe går galt, og når det er suksess.	Et viktig punkt som er prioritert som «Should Have». Systemet bør følge både designkrav fra kunden, samt oppfylle kravene fra universell utforming for at systemet skal være så tilgjenglig og intuitivt som mulig.	*Should Have*
9	Som bruker/admin, ønsker jeg å booke utstyr fremover i tid, slik at jeg kan holde av spesifikt utstyr til spesifikke formål.	Mulighet for å leie utstyr for fremtiden, ikke bare i sanntid.	Når bruker er innlogget, og har navigert seg til ønsket utstyr, presenterer systemet mulighet for å velge ønsket dato og klokkeslett. Bruker velger så fremtidig dato/tid for låneperioden og bekrefter bookingen. Systemet lagrer så informasjonen på den aktuelle brukeren, legger bookingen inn i «Mine bookinger», og reserverer produktet for gitt dato/tid.	Funksjonen for booking fremover i tid, er klassifisert som «Should hHve». Dette fordi den inngår i en kravspesifikasjon med samme klassifisering (se Tabell 1: Systemkrav). Denne funksjonen er ikke kritisk for at systemet skal kunne betegnes som et «minst brukbart produkt» i henhold til kravspesifikasjonene.	*Should Have*
10	Som admin vil jeg kunne legge til, redigere og slette både utstyr, lokasjon og kategori.	En adminrettighet som tillater å endre på utstyr, lokasjon og kategori dersom nødvendig.	Admin kan etter innlogging, legge til nytt utstyr, fjerne gammelt utstyr, samt redigere utstyr ved nødvendighet. I tillegg kan admin endre på alternativene for filtrering slik at utstyrslisten alltid er oppdatert.	En funksjon som prioriteres som «Should Have», da kriterier fra kunde spesifiserer at Admin-rettigheter ikke skal prioriteres som «Must Have». Funksjonen er gjennomførbar via andre måter, og er derfor ikke en nødvendighet for at systemet skal fungere som definert MBP.	*Should Have*

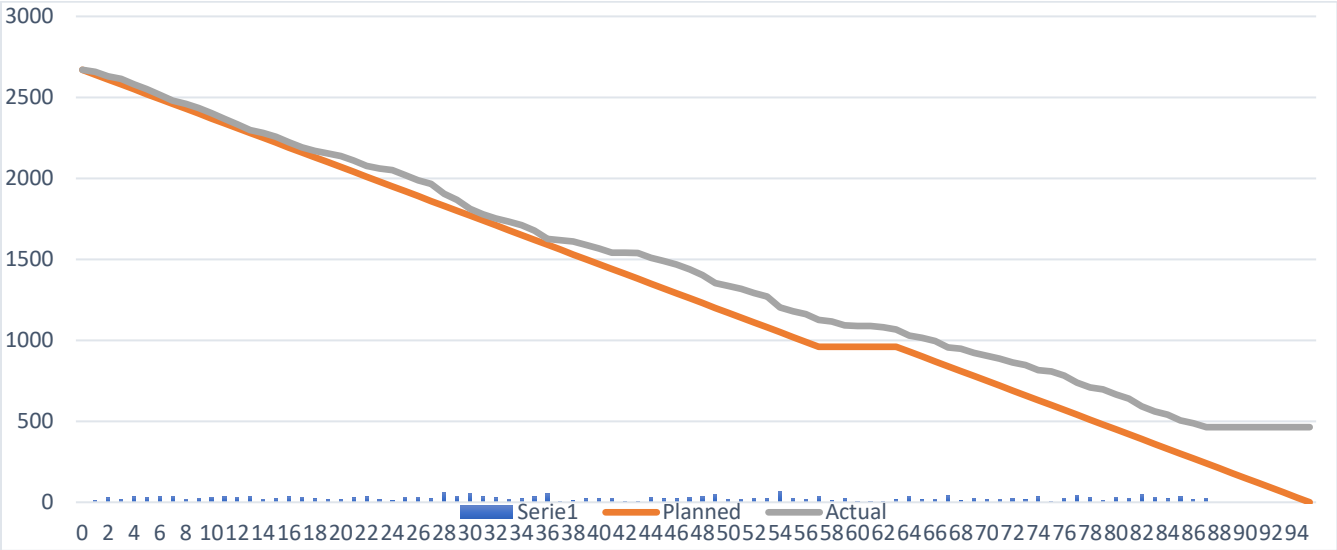
11	Som produkteier ønsker jeg et system med best mulig ytelse, slik at bedriften slipper for mye tilpasninger og vedlikehold etter overtakelse.	Et system med ryddig kode, hvor det er tatt smarte valg for å øke ytelsen.	Et økonomisk og effektivt system som besparer kunden for mye vedlikehold ved og etter overtakelse. Koden skal følge kodestandarder, samt fundamentale kodeprinsipper som øker ytelse.	Funksjonen prioriteres som «Should Have» da et system med rotete kode kan fungere, men det bør være på plass for å øke responstiden i systemet, samt unngå for mye vedlikehold for kunden.	*Should Have*

Vedlegg 3.5 Versjonskontroll

Prosjektet og utviklingens Git-flyt illustreres i bildet under og kan beskrives på følgende måte: masterbranch er prosjektets hovedbranch. DEV-branch er basert på master og er branchen som utvikling av ny backend og frontend vil baseres på. Ved utvikling av en ny funksjonalitet opprettes det en branch for denne basert på DEV. Når koden for angitt funksjonalitet er ferdigstilt og kontrollert av annen utvikler, merges denne til DEV. Fletting av DEV til master forekommer, men det er sjeldent. Dette har skjedd ved de anledninger der samtlige i gruppen har gitt sitt samtykke for at funksjonaliteten er godkjent i henhold til produkteiers krav og gruppens planer for videre utvikling. Videre «rebases» DEV og utvikling fortsetter deretter fra DEV. Fiksing av feil og mindre feilrettinger har vært basert på DEV og navngitt etter dens hensikt slik at den enklere kunne spores i ettertid om nødvendig.



Vedlegg 3.6 Burn down chart for hele prosjektperioden



	Hours		Remaining		Done
Day	Planned	Actual	Planned	Actual	Today
0			2670	2670	0
1	30	12	2640	2658	12
2	30	27,75	2610	2630,25	27,75
3	30	17	2580	2613,25	17
4	30	33	2550	2580,25	33
5	30	29,25	2520	2551	29,25
6	30	34,75	2490	2516,25	34,75
7	30	36,75	2460	2479,5	36,75
8	30	20	2430	2459,5	20
9	30	25	2400	2434,5	25
10	30	32,5	2370	2402	32,5
11	30	35,75	2340	2366,25	35,75
12	30	32	2310	2334,25	32
13	30	35,75	2280	2298,5	35,75
14	30	19,25	2250	2279,25	19,25
15	30	23,25	2220	2256	23,25
16	30	32,75	2190	2223,25	32,75
17	30	32	2160	2191,25	32
18	30	21	2130	2170,25	21
19	30	16,5	2100	2153,75	16,5
20	30	16,5	2070	2137,25	16,5
21	30	27	2040	2110,25	27
22	30	33,25	2010	2077	33,25
23	30	16,75	1980	2060,25	16,75
24	30	9,25	1950	2051	9,25
25	30	32,25	1920	2018,75	32,25
26	30	31,25	1890	1987,5	31,25
27	30	21,25	1860	1966,25	21,25
28	30	61	1830	1905,25	61
29	30	38,5	1800	1866,75	38,5
30	30	55,25	1770	1811,5	55,25
31	30	32,75	1740	1778,75	32,75
32	30	26,75	1710	1752	26,75
33	30	19,5	1680	1732,5	19,5
34	30	21,25	1650	1711,25	21,25
35	30	33,25	1620	1678	33,25
36	30	52,75	1590	1625,25	52,75
37	30	6,25	1560	1619	6,25
38	30	8,75	1530	1610,25	8,75
39	30	21,75	1500	1588,5	21,75
40	30	22,25	1470	1566,25	22,25
41	30	25	1440	1541,25	25
42	30	1	1410	1540,25	1
43	30	1,25	1380	1539	1,25
44	30	29,25	1350	1509,75	29,25
45	30	20,25	1320	1489,5	20,25
46	30	21	1290	1468,5	21
47	30	30	1260	1438,5	30
48	30	35,75	1230	1402,75	35,75
49	30	50	1200	1352,75	50
50	30	17	1170	1335,75	17
51	30	19	1140	1316,75	19
52	30	24,5	1110	1292,25	24,5
53	30	22,75	1080	1269,5	22,75
54	30	67,25	1050	1202,25	67,25
55	30	23,5	1020	1178,75	23,5
56	30	17,75	990	1161	17,75
57	30	35,75	960	1125,25	35,75
58	0	10	960	1115,25	10
59	0	23	960	1092,25	23
60	0	3,75	960	1088,5	3,75
61	0	1	960	1087,5	1
62	0	6,75	960	1080,75	6,75
63	0	14,5	960	1066,25	14,5
64	30	36,75	930	1029,5	36,75
65	30	14,5	900	1015	14,5
66	30	19,5	870	995,5	19,5
67	30	39,75	840	955,75	39,75
68	30	8,25	810	947,5	8,25
69	30	24,5	780	923	24,5
70	30	17,5	750	905,5	17,5
71	30	19	720	886,5	19
72	30	24,35	690	862,15	24,35
73	30	14,5	660	847,65	14,5
74	30	32,75	630	814,9	32,75
75	30	7,5	600	807,4	7,5
76	30	24,75	570	782,65	24,75
77	30	44,5	540	738,15	44,5
78	30	29	510	709,15	29
79	30	12,75	480	696,4	12,75
80	30	32	450	664,4	32
81	30	24,75	420	639,65	24,75
82	30	47,5	390	592,15	47,5
83	30	32	360	560,15	32
84	30	20,25	330	539,9	20,25
85	30	35	300	504,9	35
86	30	16,5	270	488,4	16,5
87	30	24,7	240	463,7	24,7
88	30	0	210	463,7	0
89	30	0	180	463,7	0
90	30	0	150	463,7	0
91	30	0	120	463,7	0
92	30	0	90	463,7	0
93	30	0	60	463,7	0
94	30	0	30	463,7	0
95	30	0	0	463,7	0

Day	Totalt
0	0
1	12
2	27,75
3	17
4	33
5	29,25
6	34,75
7	36,75
8	20
9	25
10	32,5
11	35,75
12	32
13	35,75
14	19,25
15	23,25
16	32,75
17	32
18	21
19	16,5
20	16,5
21	27
22	33,25
23	16,75
24	9,25
25	32,25
26	31,25
27	21,25
28	61
29	38,5
30	55,25
31	32,75
32	26,75
33	19,5
34	21,25
35	33,25
36	52,75
37	6,25
38	8,75
39	21,75
40	22,25
41	25,00
42	1,00
43	1,25
44	29,25
45	20,25
46	21,00
47	30,00
48	35,75
49	50,00
50	17,00
51	19,00
52	24,50
53	22,75
54	67,25
55	23,50
56	17,75
57	35,75
58	10,00
59	23,00
60	3,75
61	1,00
62	6,75
63	14,50
64	36,75
65	14,50
66	19,50
67	39,75
68	8,25
69	24,50
70	17,50
71	19,00
72	24,35
73	14,50
74	32,75
75	7,50
76	24,75
77	44,50
78	29,00
79	12,75
80	32,00
81	24,75
82	47,50
83	32,00
84	20,25
85	35,00
86	16,50
87	24,70
88	
89	
90	
91	
92	
93	
94	
95	

Vedlegg 3.7 Tabeller fra Sprint Retrospective: Hva har gått bra – Sprint 2

ID	Hva har gått bra?	Forebyggendetiltak fra ROS-analysen	Beskrivelse	Løst ✓ / ikke løst X
G2 Covid-19 begrenser muligheter for fysiske samarbeid i gruppen.	Digitalt samarbeid.	Opprettholde jevnlig kontakt digitalt. Følge retningslinjer for smittevernstiltak.	Gruppen har jobbet godt digitalt, til tross for at vi ikke har fått møttes fysisk.	✓
B7 Merge-konflikt i repository.	Gruppemedlemmene har vært flinke til å pulle ved arbeidsdagens start, samt følge rutiner som avtalt.	Alltid pull ved arbeidsdagens start.	Dette har skjedd, men siden gruppen har på plass rutiner som gjør at koden gjennomgås før den blir merget med main har ikke dette blitt et problem.	✓
E1 Samarbeidsbedrift svarer ikke på mail / annen kontakt fra gruppen	Kommunikasjon med Egde.	Avtaler møter i god tid / tidsfrist for svar /aksept	Egde svarer raskt på henvendelser, både på møteinnkallelser og annet.	✓
E2 Samarbeidsbedrift har ikke tid til å møte gruppen.	Møtedeltakelse fra Egde.	Gruppen er fleksibel på møtetidspunkt	Scrum master sender ut innkallelser i god tid. Egde deltar i alle møtene som er forespurt og satt opp fra gruppens side.	✓

Vedlegg 3.8 Tabeller fra Sprint Retrospective: Hva har gått galt – Sprint 2

ID	Hva har gått galt?	Tiltaksplan fra ROS-analysen	Beskrivelse	Løst ✓ / ikke løst X
C1 Gruppemedlemmer vet ikke hva de skal gjøre.	Oppgavefordeling, og nedbryting av oppgaver.	Delegere oppgaver ved sprint planning.	Oppgavene som var satt opp i Sprint 2, var ikke detaljerte nok/små nok. Dette skapte forvirring og usikkerhet. Gruppemedlemmene har fått for store oppgaver som de ikke har nok kunnskap til å gjennomføre.	X
C5 Gruppemedlemmer leverer ikke forventede resultater.	Ikke nok kunnskap til å gjennomføre oppgaver.	Andre må gjennomgå arbeidet på nytt.	Henger sammen med C1. Gruppen leverte ikke forventet resultat, da de ikke hadde kunnskap til å gjøre oppgaver som ikke var godt nok definert i backlog. Alle var innstilt på å forbedre dette til neste sprint.	X
C6 Gruppemedlemmer tar ikke solidarisk ansvar	Kommunikasjon	Signert gruppekontrakt som spesifiserer konsekvenser	Digitalt samarbeid senket kvalitet på kommunikasjonen. Gruppemedlemmer venter for lenge med å si ifra når de ikke var i stand å gjennomføre en oppgave. Dårlig kommunikasjon førte til dårlig samarbeid og lite ansvarsfølelse for oppgaven.	X
C7 Gruppemedlemmer mister motivasjon.	Oppgavefordeling, og nedbryting av oppgaver.	Tildele arbeider (så langt det lar seg gjøre) som medlemmene synes er spennende.	Henger sammen med C1 og C5. Motivasjon gikk tapt ved usikkerhet og forvirring, samt dårlig kommunikasjon og kun digitalt samarbeid.	X/✓
B4 Ikke i stand til å programmere nødvendige funksjoner.	Kunnskapsnivået for programmering.	Kontakte andre gruppemedlemmer/produkteier/veileder for bistand.	Ble fikset under sprinten, Janis Gailis, Egde og medstudenter ble kontaktet for å prøve å fikse problemet.	✓

Vedlegg 3.9 Tabeller fra Sprint Retrospective: Hva har gått bra – Sprint 3

ID	Hva har gått bra	Forebyggendetiltak fra ROS-analysen	Beskrivelse	Løst ✓ / ikke løst X
A11 Glemme å bruke ROS-analysen.	Bruk av ROS-analysen.	Sette ROS på agendaen i sprint retrospective	Gruppen brukte ROS-analysen jevnt under spriten.	✓
C11 Gruppemedlemmer logger ikke timer.	Timeregistrering i Azure og Microsoft Teams.	Påminnelse i stand-up-møtet	Gruppemedlemmer vært flinke til å logge timer. Scrum master har sendt noen påminnelser til gruppen i felles chatten.	✓
C3 Gruppemedlemmer møter ikke opp uten gyldig fravær.	Alle aktuelle parter har deltatt på alle aktuelle møter	Utsendelse av møteinnkallelse i god tid	Scrum master har sendt ut innkallelser i god til de aktuelle møtedeltakerne.	✓
E5 Samarbeidsbedrift og studentgruppen har ulike forståelse av problemstilling og/eller løsningsforslag.	Produkteier deltar aktivt under sprint review.	Produkteier er med i sprint review	Egde og prosjektgruppen har vært enig om mye/alt og har hatt en felles forståelse for hva som er forventet av systemet. Dette har vi fått gjennom diskusjon og avklaringer i sprint review.	✓
C9 Gruppemedlemmer er redd for å si ifra om konflikter.	Gruppemedlem sa ifra når noe ikke gikk slik som ønsket.	Gruppemedlemmene oppfordres til å sine meninger	Gjennom samtaler i mindre grupper turte medlemmene i gruppen å løfte sine meninger/konflikter i plenum og sammen kunne vi fikse problemene. Gruppen hadde et oppvaskmøte for å legge konflikten bak seg.	✓
C1 Gruppemedlemmer vet ikke hva de skal gjøre.	Oppgavene ble brutt ned i mer konkrete arbeidsoppgaver.	Gjennomgang ved stand-up	Vi hadde fokus på dette under sprint planning og klarte å gjøre oppgavene mer tydelige.	✓

Vedlegg 3.10 Tabeller fra Sprint Retrospective: Hva har gått galt – Sprint 3

ID	Hva har gått galt?	Tiltaksplan fra ROS-analysen	Beskrivelse	Løst ✓ / ikke løst X
B11 Docker fungerer ikke.	Docker fungerer ikke på MacOS.	Kontakte universitetet og forklare situasjonen. Arbeide videre med den status man sist var kjent med.	Gruppen har kontakt Egde og UiA. Vi har fått noe hjelp, men problemet er fortsatt til stede. Docker legges på is inntil videre.	**X**
D5 Har ikke nok kunnskap om database til å løse en oppgave.	Programmet vil ikke kjøre med Docker på MacOS.	Få hjelp av Egde.	Gruppen hadde møte med Egde, og fikk løst problemet. Vi fikk løst noe av problemet og kom oss litt videre. Programmet kjører ikke med Docker på MacOS	**✓/X**
B4 Ikke i stand til å programmere nødvendige funksjoner.	Ikke nok kompetanse til å løse gitte oppgaver.	Sette seg inn i nødvendig programmering og rammeverk.	Gruppen måtte øke kunnskapsnivået for å kunne løse oppgavene. Det gikk litt tid til å lese seg opp og finne ut hvordan man kunne løse oppgavene.	✓
B10 Folk programmerer sånn de selv ønsker / følger ikke gitt standard.	Dårlig samarbeid mtp. programmering	Utarbeide kodestandard Oppvaskmøte (forebyggende C9).	Vi var ikke samkjørte nok og hadde ikke gode nok rutiner for programmering. Forskjellig kunnskapsnivå gjorde at noen medlemmer ikke visste hva de skulle gjøre eller følte seg litt til overs. Vi hadde dermed et oppvaskmøte.	**✓/X**

Vedlegg 3.11 Tabeller fra Sprint Retrospective: Hva har gått bra – Sprint 4

ID	Hva har gått bra?	Forebyggende tiltak fra ROS-analysen	Beskrivelse	Løst ✓ / ikke løst X
A5 Personer deltar ikke i rapportskriving.	Deltakelse i rapportskriving.	Delegere oppgaver ved sprint planning.	Alle medlemmene i gruppa har deltatt i utarbeidelse av rapporten.	✓
B4 Ikke i stand til å programmere nødvendige funksjoner.	Kunnskapsnivå i programmering.	Sette seg inn i nødvendig programmering og rammeverk.	Etter hjelp utenfra i forrige sprint, økte kunnskapsnivået i gruppa og vi var i stand til å programmere nødvendige funksjoner.	✓
B8 Personer deltar ikke på programmering/ databaseutforming	Deltakelse av medlemmer i utviklerdelen av prosjektet.	Delegere oppgaver ved sprint planning.	Alle medlemmene har bidratt ved utvikling av systemets backend og frontend.	✓
C1 Gruppemedlemmer vet ikke hva de skal gjøre.	Fordeling av arbeidsoppgaver og nedbryting av oppgaver.	Gjennomgang ved stand-up.	Etter tidligere sprinter, var det ekstra fokus på å eliminere usikkerhet rundt hva medlemmene skulle jobbe med, i form av å nøye bryte opp oppgaver, samt beskrive disse godt.	✓
C2 Gruppemedlemmer arbeider ikke.	Arbeidsfordeling og nedbryting av oppgaver, samt beskrivelse av oppgaver. Stand-up.	Delegere oppgaver ved sprint planning.	Alle i gruppen visste hva de skulle gjøre, både på grunn av en nøye utarbeidet sprint backlog, og gode møter hvor man forpliktet seg til å gjennomføre arbeid, samt kunne diskutere usikkerheter.	✓
C7 Gruppemedlemmer mister motivasjon.	Kommunikasjon og oppgavefordeling.	God ledelse/styring av gruppa.	Kommunikasjonen var mer transparent i Sprint 4, som åpnet for ærlige diskusjoner. Dette gjorde samholdet i gruppe bedre og økte motivasjonen. I tillegg gjorde godt definerte arbeidsoppgaver arbeidet lettere og mer belønnende.	✓

Vedlegg 3.12 Tabeller fra Sprint Retrospective: Hva har gått galt – Sprint 4

ID	Hva har gått galt?	Tiltaksplan fra ROS-analysen	Beskrivelse	Løst ✓ / ikke løst X
A11 Glemme å bruke ROS-analysen	Glemt å bruke ROS jevnlig.	Innkallelse til ROS-møte sendes ut.	Sprinten gikk såpass sømløst at ROS ble glemt litt bort. Gruppen ble klar over dette og sendte ut en innkallelse til et ROS møte i starten av Sprint 5.	✓
I2 Ikke nok kunnskap om testing.	Kunnskapsnivået på testing litt for lavt, kom i gang litt sent.	Spørre om hjelp via digitale plattformer.	Medlemmene kunne ikke nok om testing til å gjennomføre tilfredsstillende tester på systemet. Ikke vært tilgjengelige ressurser for hjelp. Ble satt opp møte med eksterne i Sprint 5.	X
E1 Samarbeidsbedrift svarer ikke på mail / annen kontakt fra gruppen.	Kommunikasjon med samarbeidsbedriften har ikke vært tilfredsstillende.	Kontakt emneansvarlig og/eller veileder for bistand.	Gjelder ikke produkteier. Gruppen har ikke fått tilgang til nøkkelpersoner som innehar nødvendig kunnskap for videre progresjon.	X
E2 Samarbeidsbedrift har ikke tid til å møte gruppen.	Samarbeidsbedriften har ikke hatt tid/ressurser til å møte gruppen.	Kontakt emneansvarlig og/eller veileder for bistand.	Gjelder ikke produkteier. Gruppen har ikke fått tilgang til nøkkelpersoner som innehar nødvendig kunnskap for videre progresjon.	X

Vedlegg 3.13 Tabeller fra Sprint Retrospective: Hva har gått bra – Sprint 5

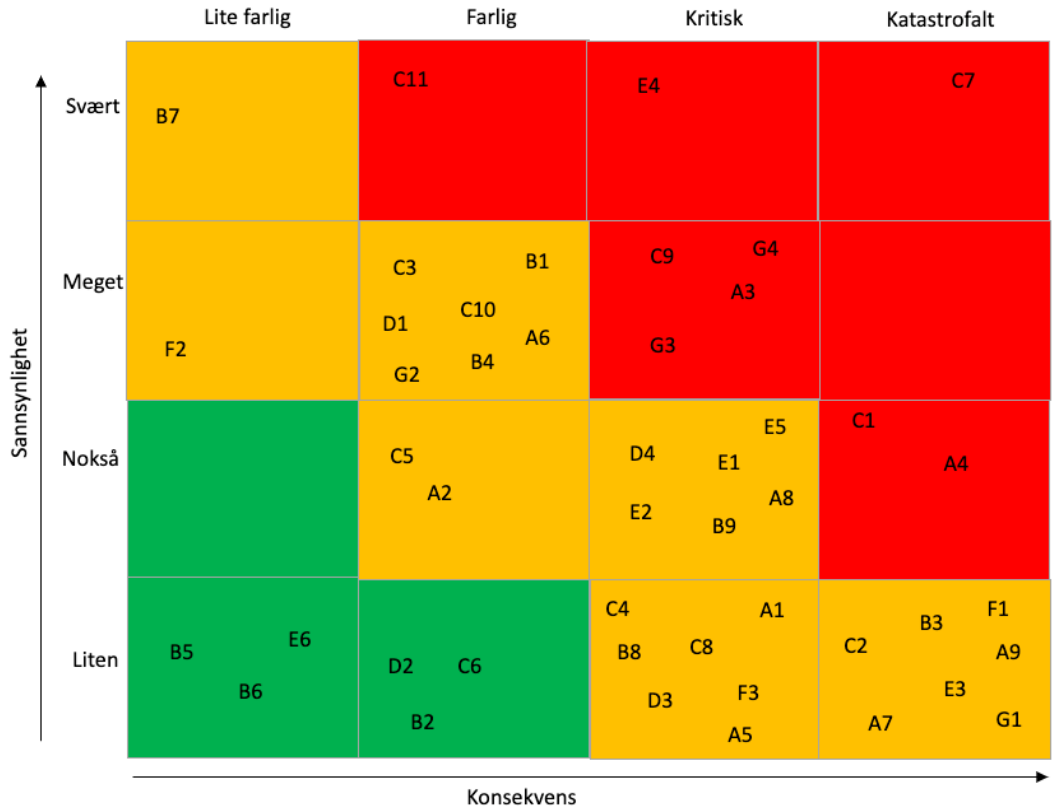
ID	Hva har gått bra?	Forebyggendetiltak fra ROS-analysen	Beskrivelse	Løst ✓ / ikke løst X
A11 Glemme å bruke ROS-analysen	Gruppen har husket å bruke ROS-analysen gjennom sprinten.	Sette ROS på agendaen i sprint retrospective	Etter end Sprint 4, hvor ROS-analysen var glemt litt bort, ble det satt opp et ROS-møte i Sprint Retrospective 4. Dette ble gjennomført i Sprint 5, og ROS var dermed integrert i sprinten igjen.	✓
A5 Personer deltar ikke i rapportskriving.	Deltakelse i rapportskriving.	Delegere oppgaver ved sprint planning.	Alle medlemmene i gruppa har deltatt i utarbeidelse av rapporten.	✓
C2 Gruppemedlemmer arbeider ikke.	Arbeidsfordeling og nedbryting av oppgaver, samt beskrivelse av oppgaver. Stand-up.	Delegere oppgaver ved sprint planning.	Alle i gruppen visste hva de skulle gjøre, både på grunn av en nøye utarbeidet sprint backlog, og gode møter hvor man forpliktet seg til å gjennomføre arbeid, samt kunne diskutere usikkerheter.	✓
C1 Gruppemedlemmer vet ikke hva de skal gjøre	Arbeidsfordelingen har vært god, konkrete arbeidsoppgaver.	Gjennomgang ved stand-up.	Alle i gruppen visste hva de skulle gjøre av samme grunner som nevnt i beskrivelsen for C2.	✓
C7 Gruppemedlemmer mister motivasjon.	Kommunikasjon og oppgavefordeling.	God ledelse/styring av gruppa.	Kommunikasjonen var mer transparent i Sprint 4, som åpnet for ærlige diskusjoner. Dette gjorde samholdet i gruppe bedre og økte motivasjonen. I tillegg gjorde godt definerte arbeidsoppgaver arbeidet lettere og mer belønnende.	✓

Vedlegg 3.14 Ros-analysen med kvalitetssikringstiltak.

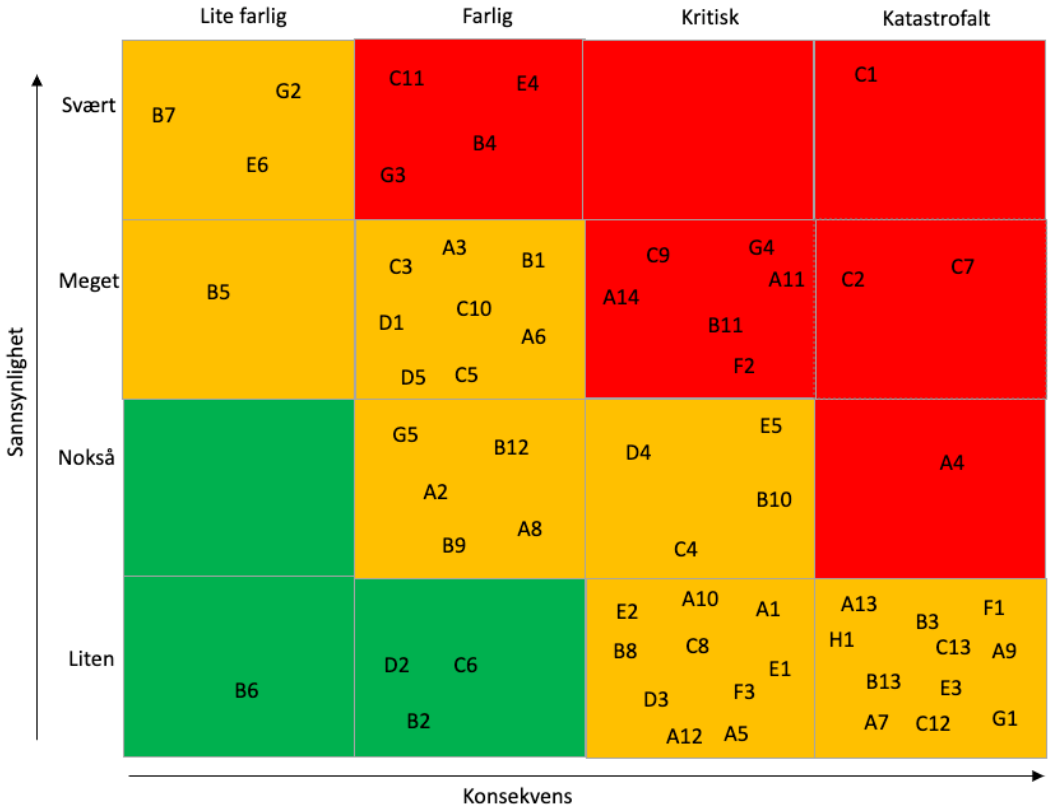
Link til fullstendig tabell: https://docs.google.com/spreadsheets/d/12exrXP-g20lIJpnBSoCeSDq13KM0iQdA6gjk2L9_Y5s/edit?usp=sharing

ID	Hva kan gå galt?	Sannsynlighet for forekomst	Konsekvens for prosjektet
A01	Person sletter hele rapporten	Liten	Farlig
A02	Person sletter deler av rapporten	Meget	Farlig
A03	Personer misforstår bruk av teori i praksis	Meget	Farlig
A04	Personer er lite kildekritiske	Liten	Katastrofalt
A05	Personer deltar ikke i rapportskriving	Liten	Katastrofalt
A06	Personer gjør fundamentale endringer i andres arbeid uten samtykke	Liten	Kritisk
A07	Rapporten blir ikke ferdig til gitt frist (14.mai 2021)	Meget	Katastrofalt
A08	Hele eller deler av rapporten er under forventet standard etter endt Sprint 4.	Nokså	Farlig
A09	Har levert arbeid som er plagiert.	Liten	Katastrofalt
A10	Personer plagierer andres arbeid.	Nokså	Kritisk
A11	Glemme å bruke ROS-analysen	Meget	Kritisk
A12	Veileder gir feilaktig eller ufullstendig tilbakemelding	Liten	Kritisk
A13	Ikke leverer rapport	Liten	Katastrofalt
A14	Har ikke nok kunnskap til å løse en oppgave	Meget	Kritisk
A15	Gruppemedlemmer logger feil timer	Svært	Kritisk
B01	Oppkobling/ kommunikasjon med internett, flyttet til G05	Meget	Farlig
B02	At uvedkommende får tilgang til online-server jf F04	Liten	Katastrofalt
B03	Repository slettes	Liten	Katastrofalt
B04	Ikke i stand til å programmere nødvendige funksjoner	Svært	Lite
B05	At et gruppemedlem ikke arbeider i seneste systemversjon	Svært	Lite
B06	Ikke kontakt med repository	Liten	Lite
B07	Merge-konflikt i repository	Svært	Lite
B08	Personer deltar ikke på programmering/ databaseutforming	Liten	Kritisk
B09	Systemet mangler nødvendig funksjonalitet ved endt prosjektperiode	Liten	Kritisk
B10	Folk programmere sånn de selv ønsker / følger ikke gitt standard	Nokså	Kritisk
B11	Docker fungerer ikke jf. f04	Svært	Farlig
B12	Oppgaver som er avhengige av hverandre blir gjort i feil rekkefølge	Liten	Farlig
B13	Ingen i gruppa får kontakt med repository	Liten	Katastrofalt
B14	Mangler kunnskap om front end-utvikling	Meget	Kritisk
B15	Brukervenneligheten i front end er under forventning	Nokså	Kritisk
C01	Gruppemedlemmer vet ikke hva de skal gjøre	Liten	Katastrofalt
C02	Gruppemedlemmer arbeider ikke	Liten	Katastrofalt
C03	Gruppemedlemmer møter ikke opp uten gyldig fravær	Liten	Farlig
C04	Gruppemedlemmer leverer ikke tildelt arbeid	Nokså	Kritisk
C05	Gruppemedlemmer leverer ikke forventede resultater	Nokså	Farlig
C06	Gruppemedlemmer tar ikke solidarisk ansvar	Liten	Farlig
C07	Gruppemedlemmer mister motivasjon	Nokså	Farlig
C08	Gruppemedlemmer viser ikke hensyn til andre	Liten	Kritisk
C09	Gruppemedlemmer er redd for å si ifra om sosiale konflikter	Liten	Farlig

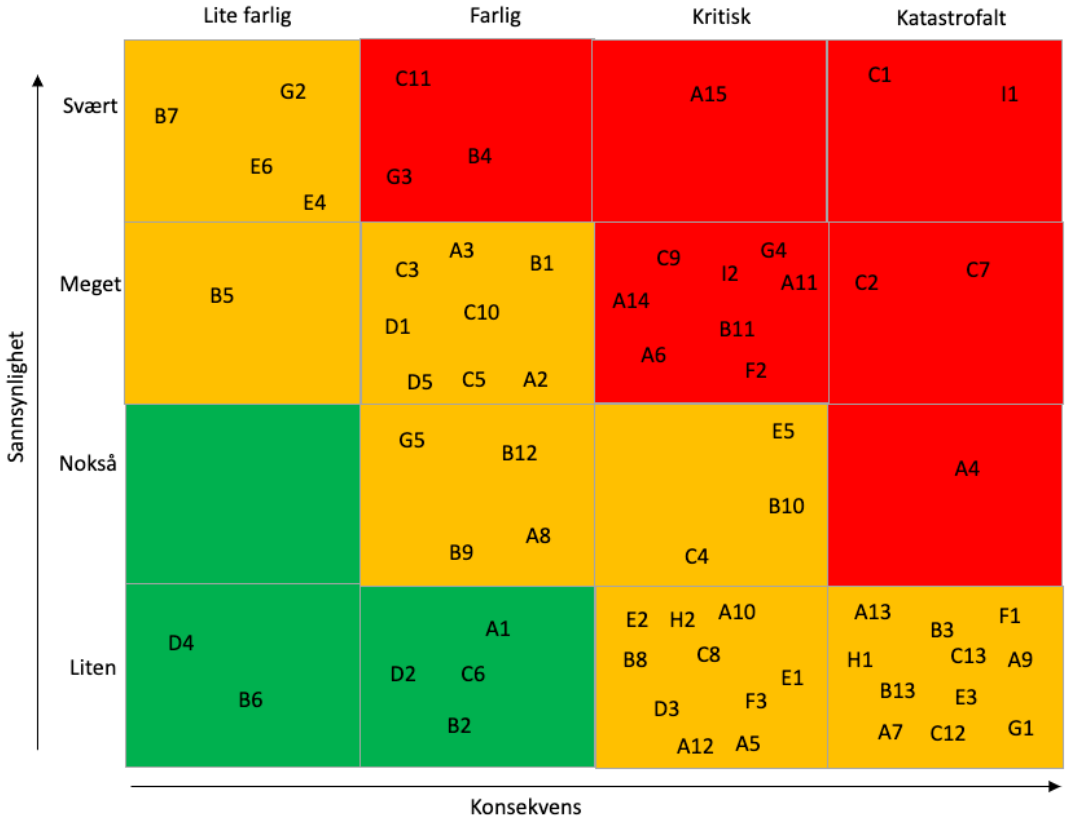
Vedlegg 3.15 Risikomatrise Sprint 1



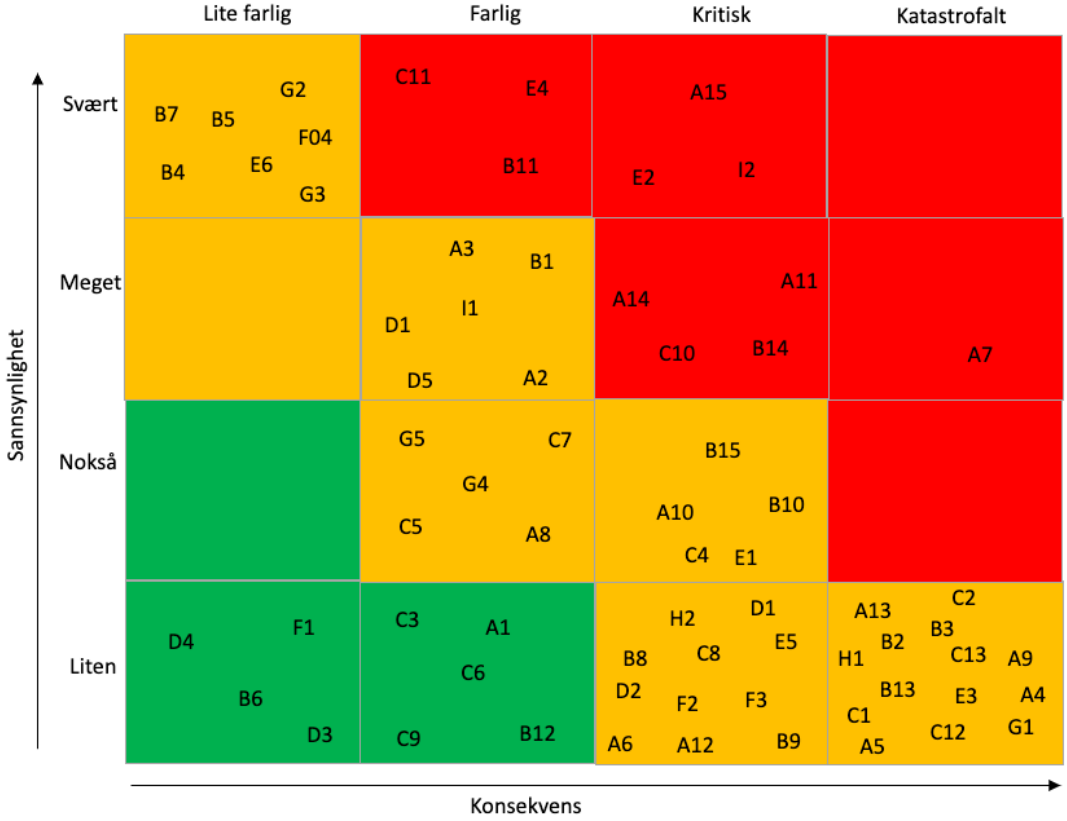
Vedlegg 3.16 Risikomatrise Sprint 2-3



Vedlegg 3.17 Risikomatrise Sprint 3-4



Vedlegg 3.18 Risikomatrise Sprint 4-5



Vedlegg 3.19 Kodestandard/Dokumentasjon

GO 4IT

Dokumentasjon for overlevering
av utlånssystem

[Versjon 1.2]

12.05.2021

Versjon	Dato	Endringer	Forfatter	Kontrollert av
1.0	08.02.2021	Opprettet dokument og fylt inn overordnet informasjon	Erik N., Kantasit I. og Magnus E.	Magnus E.
1.1	05.05.2021	Påfylling av informasjon fra utviklingen.	Erik N.	Kantasit I.
1.2	12.05.2021	Formatering og ferdigstilling for rapportlevering	Erik N., Kantasit. I. og Linn F.	Erik N., Kantasit. I., Linn F. og Magnus E.

Innholdsfortegnelse

1 Generelt **3**

2 Versjonskontroll **3**

2.1 Kvalitetssikringstiltak *3*

2.2 Kommentering av kode *3*

2.3 Git-flyt *3*

3 Systemarkitektur **4**

4 Database **5**

5 Testing **5**

6 Forslag til videre utvikling **5**

1 Generelt

Denne kodestandarden er utarbeidet av Go4IT, som er en studentgruppe bestående av seks studenter ved Universitetet i Agder, som våren 2021 fullfører sine bachelorgrader innen IT og informasjonssystemer.

Kodestandarden har til hensikt å være en veiledende forklaring til den eller de som skal videreutvikle systemet. Videre tar kodestandarden for seg hvilke komponenter systemet består av i dag (oppdatert 12.05.2021), som dets rammeverk og arkitektur, databasekobling (ORM), i tillegg til prinsipper gruppen har basert seg på ved utvikling av systemet slik det fremstår i dag.

Språk og verktøy som er anvendt:

- Rammeverk: ASP.NET Core 3.1
- Arkitektur: Razor Pages
- Objektrelasjonell kartlegging (eng. ORM): Entity Framework

2 Versjonskontroll

Go4IT har brukt GitHub som versjonskontroll. Repository er per i dag navngitt: «EgdeBookingSystemV2» og er tilgjengelig ved url:
https://github.com/Erdvik/EgdeBookingSystemV2

2.1 Kvalitetssikringstiltak

Siden det er flere utviklere som samarbeider om utviklingen av dette systemet, er det viktig at samtlige følger de samme rutinene for kvalitetssikring. Ved merging/ pull-request av kode fra en branch til en annen, skal koden, så fremt det ikke foreligger noe unntak, gjennomgås av minst én annen utvikler før merge gjennomføres. Unntaket er kun dersom det er deg selv som har opprettet den branchen du merger til. Ved alle andre tilfeller skal koden gjennomgås av annen utvikler.

2.2 Kommentering av kode

Ved utvikling av ny funksjonalitet eller ved endring av eksisterende funksjonalitet, skal koden kommenteres. I kommentaren skal man kort beskrive den enkelte kodeseksjons hensikt om koden er ny, og hva som er endret, samt ny hensikt dersom man endrer eksisterende kode. Ved endring av eksisterende kode skal man i tillegg sørge for at gamle kommentarer fjernes dersom disse ikke lenger er aktuelle.

2.3 Git-flyt

Prosjektets hovedbranch er benevnt: «master». «Master» innehar siste fungerende versjon av systemet. Dette er ansett som prosjektets back-up og følgende skal man ikke merge andre brancher med master med mindre koden er gjennomgått av minst to andre utviklere og man har testet og kontrollert at systemet fungerer til sin hensikt.

Videre har man benyttet branchen «DEV», som er basert på «master», som aktuell branch ved utvikling av systemet. Fra «DEV» skal man, både ved endring av eksisterende kode, samt ved utvikling av ny, lage en ny branch som navngis i tre deler på følgende måte:

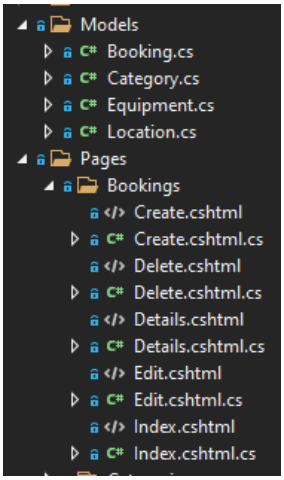
1. Hvem oppretter branchen (Initialer): eks. «NN» (Navn Navnesen)
2. Hva er hensikten med branchen: eks. «Feature»
3. Spesifikasjon av hva som skal gjøres: eks. «LeggTilUtstyr»

Ved oppsett av branch vil dette se slik ut: NN-Feature-LeggTilUtstyr

3 Systemarkitektur

I tillegg til arkitekturen ved Razor Pages har man ved dette systemet lagt vekt på prinsippet CRUD. CRUD er et akronym for Create, Read, Update og Delete, og er i stor grad benyttet ved utvikling av systemet. Ved å benytte Razor Pages og CRUD baseres controller og view på en enkelt modell. Følgende vil det enkelte view være knyttet til en controller, som utgjør en mappestruktur som både er oversiktlig og enkel å finne frem i.

Eksempelvis vil man ved videre utvikling der man baserer seg tilsvarende arkitektur, kunne se likheter med det som fremgår av Figur 1.



Figur 1: CRUD prinsipper

I eksempelet illustreres det hvordan CRUD tidligere har blitt benyttet for å opprette views og controllere for modellen «Booking».

4 Database

Databasen som benyttes i dag (oppdatert 12.05.2021) er en lokal MSSQL-server. Entity Framework (EF) er benyttet for objektrelasjonell kartlegger. I systemet er den enkelte modell grunnlaget for en entitet i databasen. Videre vil hver modell håndtere entitetens felter og egenskaper, samt hvilke entiteter som er koblet sammen og hvordan.

Ved videreutvikling av dette systemets database er noe kunnskap om bruk av EF nødvendig.

5 Testing

Testing av systemet har hovedsakelig blitt gjennomført gjennom manuelle white-box-tester. Her har vi som utviklere gjennomgått koden og testet de ulike funksjonaliteter manuelt ved for eksempel å registrere data i systemet, og sammenlikne dette med det vi forventer. Manuelle tester er mer tidkrevende enn automatiske tester, men grunnet mangel på kunnskap og nødvendige ressurser, valgte vi å fokusere på manuelle testing slik at produktet vi utviklet holder høy kvalitet.

For å heve systemets kvalitet da all funksjonalitet var ferdigstilt, fikk gruppen støtte av Egde i forhold til automatisert testing for deler av systemet.

6 Forslag til videre utvikling

Vi anbefaler produkteier å opprette automatiske tester for resten av systemets funksjoner. Systemet slik det fremstår i dag er som nevnt tidligere hovedsakelig testet gjennom manuelle tester, selv om enkelte unit-tester er lagt til for modellene.

Andreas T. har mot slutten av prosjektet vist interesse for å kunne gruppere utstyr og med det kunne booke/låne en samling av utstyr samtidig. For eksempel kunne grupperingen «podcast» inkludert et sett med mikrofoner og annet relevant utstyr, slik at hvert av disse produktene ikke måtte bookes enkeltvis slik de må i dag. Dette ble ikke prioritert, men kan legges til.

Grunnet manglende ressurser er systemet avhengig av en lokal host-maskin. Systemet er tilrettelagt for at det skal fungere live i «skyen» og dette er noe Egde har mulighet til å gjøre om de ønsker det.

Per dagsdato (12. mai 2021) eksisterer det en bug (feature) i systemet som gruppen er klar over, men som ikke er prioritert grunnet tidsbegrensninger og at bug-en ikke er ødeleggende for systemet. Bug-en forekommer ved feilbooking når man forsøker å booke bakover i tid. Da vises ikke lenger majoriteten av informasjonen om det aktuelle utstyret for brukeren, kun utstyrs-ID forblir synlig.

Utviklingen har hatt fokus på desktop-versjonen av systemet, dermed er ikke mobilvisningen av applikasjonen ideell. Selv om mye av systemet er skalerbart, må noen elementer optimaliseres for visning på mobil.

Ved brukergrensesnittdesign er Egdes merkevarebok brukt aktivt. Vi benyttet oss av farger og utforming på knapper lik som på deres nettside. Vi fikk tilsendt den skrifttypen som Egde bruker,

og ønsket opprinnelig å bruke denne på utlånssystemet. Dette fikk vi dessverre ikke til. Dette er noe Egde selv kan velge å legge inn på et senere tidspunkt, dersom det er ønskelig.

I systemet er det kun Admin som har rettigheter til å endre lokasjonen til et utstyr. Så om en bruker henter et utstyr fra Kristiansand og leverer dette i Grimstad vil ikke dette bli oppdatert i systemet med mindre Admin er kontaktet og gjør endringen manuelt. Dette kan løses på flere måter, for eksempel at brukere har rettighet til å oppdatere lokasjonen til et produkt ved endt utlånstid (og blir oppfordret til dette av systemet).

Vedlegg 3.20 Gruppekontrakt

IS-304 Bacheloroppgave i informasjonssystemer

GRUPPEKONTRAKT

GO 4IT

Gruppemedlemmer og kontaktinformasjon

Navn	E-post	Telefonnummer
Linn E. Fredriksen	[redacted]	[redacted]
Marine Ramsdal	[redacted]	[redacted]
Julie Hedemark	[redacted]	[redacted]
Magnus Erdvik	[redacted]	[redacted]
Kantasit Intaraphasuk	[redacted]	[redacted]
Erik D. Nergaard	[redacted]	[redacted]

Om dette dokumentet

Dette dokumentet er en juridisk bindene kontrakt signert av alle gruppens medlemmer, og er gyldig inntil siste eksamen er gjennomført i det aktuelle faget. Alle gruppens medlemmer har kommet til en enighet om å gjøre sitt ytterste for å sikre god kvalitet på innleveringer og generelt høy innsats i faget.

Artikkel I: Møter og gruppearbeid

1.1 Alle gruppemedlemmer plikter å møte til avtalt tid, gjelder både fysiske og digitale møter.

1.2 Alle gruppemedlemmer plikter å bidra med en arbeidsmengde i henhold til antall studiepoeng.

1.3 Alle gruppemedlemmer plikter å være til stede under alle møter, med mindre det foreligger en gyldig grunn til fravær.

1.4 Alle gruppemedlemmer plikter å stille med nødvendig utstyr til å gjennomføre gruppearbeidet. Gruppemedlemmene plikter til å holde seg oppdatert på aktuelle arbeidsverktøy.

1.5 Hvert teammedlem pliktes til å registrere den tid hver enkelt har brukt på en avtalt arbeidsoppgave. Tiden skal registreres i et system som avtales mellom teamets medlemmer. Krav til registrering vil være; dato, timer brukt, og hva man har arbeidet med.

1.6 Gruppemedlemmer plikter å delta på møter og gruppearbeid innen 15 minutter fra avtalt tid. Dersom et gruppemedlem ikke møter opp innen tidsgrensen, vil medlemmet få en anmerkning. Dersom et gruppemedlem får tre anmerkninger må den aktuelle personen lage kake, pizza, eller annet som bidrar til et godt gruppemiljø.

1.7 Alvorlige og gjentatte brudd på tidsfrister samt fravær uten gyldig grunn vil resultere i anmerkinger. Dersom et gruppemedlem får tre anmerkninger, vil den aktuelle personen bli innkalt til en samtale for å drøfte oppførselens konsekvenser. Hovedregelen er at prosjektleder gjennomfører samtalen med mindre annet foreligger. Gjelder avviket prosjektleder selv, skal muntlig advarsel gis av resterende gruppemedlemmer.

1.8 Linn og Julie er fra dags dato (05.01.2021) fungerende prosjektledere. Prosjektlederne har som hovedansvar å sørge for godt samarbeid, i tillegg til ansvar for innlevering av obligatoriske oppgaver.

1.9 Dersom et flertall av gruppemedlemmene uttrykker misnøye med prosjektleder, skal dette begrunnes og tas opp med prosjektleder snarest. Prosjektleder forplikter seg til å høre ut de meninger som kommer frem og rette seg etter disse innen rimelighetens grenser.

1.10 Dersom flertallet stemmer for at prosjektleder fratrer sin rolle, skal ny leder utnevnes som følge av avstemming foretatt av samtlige gruppemedlemmer, også fratredende leder.

Artikkel II: Arbeidskvalitet

2.1 Alle gruppemedlemmer plikter å gjøre seg kjent med og følge universitetets retningslinjer for kildehåndtering (APA6th).

2.2 Dersom et gruppemedlem ønsker å gjøre betydelige endringer i prosjektinnhold opprettet av et annet gruppemedlem eller medlemmer, må den eller de gruppemedlemmene informeres.

2.3 Gruppemedlemmene er ansvarlig for sluttproduktet, og alle medlemmene må samtykke til innholdet som leveres. Dette innebærer at man skal gjennomgå hverandres arbeid og ikke være redd for å gi tilbakemelding på andre medlemmers arbeid, samt stille spørsmål ved måten for gjennomføring og innholdet som presenteres.

2.4 Gruppearbeidet skal deles likt mellom gruppemedlemmene.

2.5 Frister/ avtalte arbeidsoppgaver skal overholdes, avvik skal meldes til resterende medlemmer av team ved første mulige anledning.

2.6 Medlemmer som ikke mestrer /forstår omfanget av egen arbeidsoppgave, skal snarest gi beskjed til andre teammedlemmer om hva de trenger hjelp til.

Artikkel III: Gruppemiljø

3.1. Alle gruppemedlemmer skal respekteres og lyttes til ved uttrykk av meninger og ideer. Alle gruppemedlemmer skal videre inkluderes og alle gruppemedlemmer skal ta vare på hverandre.

3.2 Oppførsel som inkluderer fysisk og psykisk trakassering eller vold, vil ikke tolereres og kan resultere i utestengelse fra gruppen.

Signatur

Linn E. Fredriksen — 06.01.2021/Kristiansand

Linn E. Fredriksen — Dato/Sted

Marine Ramsdal — 06.01.2021/Kristiansand

Marine Ramsdal — Dato/Sted

Julie Hedemark — 06.01.2021/Kristiansand

Julie Hedemark — Dato/Sted

Magnus Erdvik — 06.01.2021Kristiansand

Magnus Erdvik — Dato/Sted

Kantasit I. — 06.01.2021/Kristiansand

Kantasit Intaraphasuk — Dato/Sted

Erik D. Nergaard — 06.01.2021/Kristiansand

Erik D. Nergaard — Dato/Sted