UNIVERSITETET I AGDER

Forside

IS-304: 2022

Tittel: Modernisering av Elements Lekdommer webapplikasjon

Emnekode	IS-304
Emnenavn	Bacheloroppgave i informasjonssystemer
Emneansvarlig:	Hallgeir Nilsen
Veileder	Fredrik Werpen
Oppdragsgiver:	Sikri AS

Studenter:

Etternavn	Fornavn
Gauslaa	Martin Elias
Lislevand	Maren Ryder
Martinsen	Andreas Hovland
Monge	Sondre Tråholt
Sundbø	Simen

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 X	NEI___
Kan besvarelsen brukes til undervisningsformål?	JA X	NEI___
Vi bekrefter at alle i gruppa har bidratt til besvarelsen	JA X	NEI___

Forord

Denne rapporten ble skrevet som en obligatorisk del av sjette semester våren 2022 ved institutt for Informasjonssystemer ved UiA, Kristiansand. Besvarelsen tilhører faget IS-304 Bacheloroppgave i informasjonssystemer.

Kristiansand, 16. mai 2022

Vi vil først takke Silje Hansen og Sikri for et fint samarbeid og at vi fikk muligheten til å gjennomføre bachelorprosjektet vårt hos dem. Sikri har gitt råd og veiledning gjennom hele prosjektet, både teknisk og faglig. Vi vil også takk de ansatte hos Sikri for deres mottakelse av oss og deres hjelp med stort og smått.

Vi vil rette en takk til produkteier Fredrik Werpen, samt veilederne våre Marius Holen, Nikolai Holmen, Markus Kolstad og Asbjørn Nordgaard for deres bidrag, støtte og engasjement til dette prosjektet. Takk for verdifull hjelp, anbefalinger og tips gjennom hele prosjektet. Vi har satt stor pris på deres innsats i at vi skal lykkes.

Til slutt vil vi takke veilederen vår ved UiA, Hallgeir Nilsen. Takk for at du alltid har vært tilgjengelig og for alle din verdifulle tilbakemeldinger i løpet av prosjektet, sluttrapporten og ikke minst det siste året på universitetet.

Sammendrag

Som en del av emnet IS-304 Bacheloroppgave i informasjonssystemer, har vi i Sesma IT utført bachelorprosjekt i samarbeid med Sikri AS. Sikri er et nydannet programvareselskap som er norgesledende innen systemer for saksbehandling, dokumentforvaltning og arkivering. De har et fremtidsrettet blikk med fokus på bærekraft og innovasjon.

Grunnen til at valget falt på Sikri var den spennende og utfordrende oppgaven, som ville kreve et vidt spenn av ferdigheter. Gruppemedlemmene har tidligere opparbeidet ulik kompetanse, noe som gjorde at en full-stack oppgave passet oss perfekt. Sikri tar i bruk teknologier som er dagsaktuelle og som gruppen ønsket å bedre sin kompetanse innenfor.

Elements Lekdommer, systemet som i dag håndterer meddommerprosessen, er et utdatert system som skal moderniseres ved bruk av nye teknologier og et forbedret brukergrensesnitt. Gruppens oppgave var dermed å iverksette denne moderniseringen, samt komme med ideer for videre utvikling.

Rapporten tar for seg hvordan gruppen har gjennomført prosjektet, samt hvordan ulike verktøy og rammeverk har blitt anvendt, og hvordan de har bistått i prosessen. Gruppen har jobbet agilt ved hjelp av Scrum-rammeverket, dette ble gjort for å få kontinuerlig tilbakemelding fra produkteier, noe som var med på å sikre kvalitet i prosjektet.

Resultatet av prosjektet er et system der hovedfunksjonalitet har blitt implementert, samt et produkt som er ferdig analysert og designet. Vi la mye vekt på å komme med ideer til et fullverdig system, noe som har resultert i et sett med brukerhistorier som dekker det fullstendige produktet. Sluttproduktet har blitt analysert og utformet med tett oppfølging av produkteier og tekniske veiledere, ettersom produktet skal overleveres til Sikri for videre utvikling.

Innholdsfortegnelse

Figurliste

1.0 Introduksjon

Gjennom det siste semester på studiet It og Informasjonssystemer har gruppen gjennomført bacheloroppgaven i samarbeid med Sikri AS. Prosjektets formål er å ta i bruk kunnskap og lærdom vi har opparbeidet oss fra tidligere emner på studiet. Vår oppgave var å fornye den eksisterende Elements Lekdommer applikasjonen ved bruken av moderne teknologier.

1.1 Sikri

Sikri er en av Norges ledende leverandører av løsninger innenfor saksbehandling og dokumentforvaltning (Sikri, u.å a). De står for populære løsninger som Elements, et saks- og arkivsystem som blir brukt i store deler av offentlig sektor. Løsningene Sikri utvikler skal dekke nye behov, samtidig som de bruker moderne teknologi til å støtte eksisterende forretningsprosesser.

1.2 Produktet i dag

Elements Lekdommer er en landsdekkende webbasert løsning for kommuner i forbindelse med valg av meddommere (Sikri, u.å b). Den tar blant annet for seg valg, ajourhold og fritaksbehandling av meddommere i tingretten, samt meddommere og lagrettemedlemmer i lagretten. Tjenesten ble utviklet for å håndtere problemdomenet, men tjenesten er gammel og noe utdatert når det kommer til teknologi og brukergrensesnitt. Med tiden har flere kommuner gått vekk fra å bruke Elements Lekdommer for valg og rekruttering av meddommere, da den er tungvinn å bruke med et grensesnitt som blant annet innebærer mange unødvendige tastetrykk. I tillegg har tjenesten mangler og forbedringspotensial når det kommer til automatisering og effektivisering av oppgaver tilknyttet rekrutteringen av meddommere. Det er også flere funksjoner i den nåværende tjenesten som ikke blir brukt og oppleves dermed som irritasjonsmomenter.

1.3 Oppgavebeskrivelse

Vår oppgave var å utvikle en ny og forbedret utgave av meddommertjenesten. Dette innebærer alt fra analyse og design til koding av tjenesten. Det var ikke forventet at vi skulle levere et ferdigstilt produkt, da omfanget av dette blir for stort for et bachelorprosjekt. Sikri ønsket derimot at vi kom opp med ideer og løsninger til et komplett system med ulike funksjoner som kan implementeres av Sikri etter endt prosjekt. En grundig analyse- og designjobb ble derfor viktig for å sikre at den nye tjenesten håndterer alle oppgaver tilknyttet prosessen rundt rekruttering og valg av meddommere. Vi skulle også starte utviklingen av meddommertjenesten ved å kode både backend og frontend. Her har vi sammen med Sikri definert et scope for hva som skulle utvikles under bachelorprosjektet. Scopet blir presentert senere i rapporten.

1.4 Problemdomene

En meddommer er en person uten juridisk utdannelse, som deltar i behandlingen av en rettssak sammen med domstolens egne dommere (Norges Domstoler, u.å. a) Kommuner og fylkeskommuner velger hvert fjerde år nye meddommere som skal tjenestegjøre i norske domstoler. Problemdomenet forklarer prosessen og stegene kommuner og fylkeskommuner må gjennom ved valg av nye meddommere og er utgangspunktet for hvordan meddommertjenesten skal håndtere denne prosessen.

Når en ny periode nærmer seg og det er tid å velge nye meddommere, blir kommunene varslet om behovet for meddommere og skjønnsmedlemmer. Det er domstolen som bestemmer hvordan fordelingen skal være basert på antall innbyggere i kommunen, kjønn og hvor mange ganger en meddommer antas å bli trukket ut til å tjenestegjøre. Når kommunene har fått beskjed om hvor mange meddommere de skal velge, begynner de med rekrutteringen. Domstolene krever at utvalgene skal ha en allsidig sammensetning som skal representere hele befolkningen. Kommunene skal oppfordre befolkningen til å verve seg som meddommere.

For å kunne bli meddommer må personen oppfylle visse lovpålagte krav fra domstolen. Disse kravene omhandler blant annet at personen er innenfor en viss alder, har norsk eller nordisk statsborgerskap og det stilles krav til lovlydighet. Det kreves også at kommunene utfører strafferettslig vandelsvurdering av kandidatene for å kontrollere at meddommerkandidatene er skikket til å gjennomføre en verdig rettsforhandling.

Det skal velges meddommere til tre ulike utvalg. De tre utvalgene er lagmannsretten, tingretten og jordskifteretten. Innenfor disse utvalgene skal det være et utvalg for kvinner og ett for menn. I jordskifteretten kreves det at det skal være like mange kvinner og menn. I tillegg skal det fremmes forslag fra kommunen om skjønnsmedlemmer som velges av fylkestinget. Samme person kan ikke være meddommer i både lagmannsretten og tingretten, men alle andre kombinasjoner kan velges.

Valget av meddommere må utføres av kommunestyret. Det utarbeides først et forslag til utvalgene som skal ligge ute til ettersyn i 14 dager før det endelige valget tas av kommunestyret. Når valget skal tas må kommunestyret også håndtere og avgjøre alle forespørsler om fritak som har kommet inn fra kandidatene. Kommunene oppfordres til å sende informasjon til meddommerne som har blitt valgt og hva de har blitt valgt til. Utvalgene skal sendes inn i en egen portal for meddommerutvalg og innrapporteringen skal følge en mal satt av domstolen i et Excel-dokument. Det er ikke mulig å sende inn et utvalg før domstolens behov i antall er oppnådd.

1.5 Det nye produktet

Den nye meddommertjenesten skal med moderne teknologi og et brukervennlig grensesnitt håndtere problemdomenet på en effektiv og intuitiv måte. Tjenesten skal ha funksjonalitet som gjør det mulig å utføre nærmest alle oppgaver tilknyttet rekruttering og valg av meddommere i en og samme tjeneste. Håndteringen av selve rekrutteringsprosessen løses ved at brukeren jobber seg gjennom rekrutteringen ved hjelp av steg-for-steg-løsningen tjenesten tilbyr. Dette gjør prosessen systematisk, effektiv og med få tastetrykk frem og tilbake. Meddommertjenesten kommer også

med en meddommerportal som skal være tilgjengelig for befolkningen der man selv kan melde sin interesse for å være meddommer. Nødvendig personinformasjon blir dermed lagt inn av meddommerkandidaten selv. Dette gjør at kommunene slipper å innhente informasjonen og dermed sparer mye tid. Fra meddommerportalen kan meddommere også sende inn fritakssøknader som kommunene kan behandle direkte i meddommertjenesten. Dermed slipper kommuneansvarlig i systemet å forlate meddommertjenesten for å utføre denne oppgaven.

2.0 Sentrale valg og avgjørelser

I dette kapittelet vil vi presentere sentrale valg og avgjørelser rundt prosjektstyring og teknologier, samt beslutninger tatt gjennom hele prosjektet.

2.1 Forventningsavklaringer

Gjennom dialog med Sikri, ble en rekke forventninger til prosjektet avklart. Disse forventningene omhandlet blant annet utførelse og kvalitet. Ved at gruppen var inneforstått med disse forventningene var det enklere å kunne utføre arbeidet med en sikkerhet om at det som ble gjort stod til Sikris standard.

2.1.1 Forventninger til prosjektstyring

Sikri hadde et ønske om at gruppen skulle jobbe agilt gjennom prosjektet. Ved å organisere arbeidet i perioder på to uker, hvor hver periode avsluttes med et statusmøte med Sikri, ville gruppen kunne få kontinuerlig tilbakemelding på det som har blitt gjort, samt at Sikri enkelt kunne holde følge. Sikri ønsket også at gruppen skulle ta i bruk Azure DevOps for å styre aspekter ved prosjektet, og ordnet dermed bruker og rettigheter i deres interne DevOps miljø. Dette ville gjøre det enkelt for veiledere i Sikri å holde seg oppdatert på hva gruppen gjør, samt at gruppen ville få relevant arbeidserfaring ved å jobbe tett på Sikri.

2.1.2 Teknologi

Sikri utvikler primært nye løsninger ved bruk av React med TypeScript for frontend og .NET med C# for backend. Det var derfor et krav å bruke den samme teknologien, da Sikri tar over utviklingen ved endt prosjekt. Gruppen var inneforstått med hvilke krav Sikri hadde til teknologier etter oppstartsmøte 29. november 2021. Dette gjorde det mulig å sette seg inn i de ulike teknologiene på forhånd, slik at vi møtte opp forberedt til semesterstart.

2.1.3 Kvalitet

Det nye systemet som utarbeides er et stort prosjekt og gruppen får kun være en del av startfasen. Det var derfor viktig at arbeidet som ble gjort gjennom bachelorprosjektet var grundig, slik at Sikri overtar noe av verdi som kan videreutvikles etter prosjektet vårt er ferdig. Sikri spesifiserte tidlig at gruppen måtte fokusere på kvaliteten av det som ble produsert fremfor mengden arbeid som ble levert tilbake. Kvalitet på arbeidet som ble gjennomført var derfor av høy prioritet. Sikri hadde som krav at gruppen skulle ha jevnlige statusmøter med både veiledere og produkteier. Dette ble gjort for å sikre at kvaliteten på utført arbeid er etter produkteiers ønske.

2.2 Prosjektstyring

I dette prosjektet har prosjektstyringen hatt høy prioritet og det er noe vi har hatt fokus på fra start til slutt. Vi vil i dette kapitlet presentere sentrale valg som har blitt gjort relatert til prosjektstyring.

2.2.1 Kontrakter

Vi bestemte oss tidlig for å utforme gruppekontrakt, se vedlegg 1. Hensikten med denne var å forsikre gruppen om at alle og enhver er innforstått med hva som kreves av hvert enkelt medlem for å oppnå ønsket resultat. Kontrakten stiller blant annet

visse krav til oppførsel, pålitelighet og seriøsitet, samt visse strukturelle oppgaver som må gjennomføres hver dag for å sikre kvalitet og kontroll gjennom prosjektet. Det var også et krav fra Sikri at gruppen signerte taushetsavtale for å sikre at konfidensiell informasjon ikke kom på avveie, samt arbeidskontrakt og Code of Conduct.

2.2.2 Scrum

Scrum er et rammeverk ment for å optimalisere produktutvikling og baserer seg på utvikling og leveranser i korte iterasjoner (Scrum, u.å. a). Gruppen vurderte å bruke Scrum som rammeverk for prosjektstyringen av flere grunner. Vi hadde alle god erfaring med Scrum fra tidligere prosjekter og har vært fornøyde med gevinstene dette rammeverket gir. Det å jobbe agilt med et strukturert oppsett gjør det mulig med kontinuerlig tilbakemeldinger og forbedringer. Sikri bruker også Scrum i deres prosjektarbeid. Derfor falt valget på dette rammeverket, da det gjør veiledning og samarbeid lettere.

Sentrale elementer i Scrum som vi valgte å implementere var Scrum Team, Product Backlog, Sprint Backlog, Sprint Planning, Daily Scrum, Sprint Review og Sprint Retrospect. Et Scrum Team består av en gruppe individer som jobber sammen for å gjennomføre prosjekter og levere produkter (Indeed Editorial Team, 2021). Vårt Scrum Team besto av produkteier (Sikri), Scrum Master og utviklingsteamet.

Prosjektarbeidet ble delt opp i sprinter, der hver sprint varte i to uker. Dette resulterte i 8 sprinter totalt, med avslutning av sprintene annenhver torsdag og oppstart av ny sprint påfølgende fredag. Gruppen besluttet dette på bakgrunn av hva Sikri selv praktiserer, da dette ville gjøre samhandlingen med Azure DevOps optimal ettersom at start- og sluttdato for sprintene var fastsatt i programmet.

2.2.3 Rollefordeling

Tidlig i prosjektet ble det bestemt at gruppen ikke skulle ha en overordnet gruppeleder, det var heller ønskelig med en mer dynamisk flyt i gruppen. Etter

samtale med Sikri ble det bestemt at alle gruppemedlemmene, på rundgang, skulle lede statusmøtene med bedriften. Dette ble gjort for at alle skulle få prøve seg som leder og få erfaring om hvordan det er å lede et møte. Det ble som nevnt tidligere valgt ut en Scrum Master. Grunnen til dette var at gruppen ønsket en person som skulle ha oversikt over alle Scrum-elementene. Alle gruppemedlemmene skulle delta på lik linje, men det var ønskelig at én satt seg mer inn i Scrum og hadde den nødvendige kunnskapen for gjennomføring av eventene. Et ønske fra Sikri var en kontaktperson fra gruppen som skulle stå for kommunikasjonen mellom partene og kalle inn til møter. Denne oppgaven ble delegert til Simen som også ble valgt til Scrum Master. Gruppemedlemmene delte seg inn i to team, ett for frontend og ett for backend. Frontend-teamet besto av Andreas, Maren og Sondre, og de fikk ansvar for utviklingen av brukergrensesnitt. Elias og Simen dannet dermed backend-teamet og hadde ansvar for Rest APIen.

2.2.4 Prosjektstyringsverktøy

Programmet Azure DevOps kommer med en rekke fordeler og gir god støtte til agil programvareutvikling (Microsoft, u.å. a). Azure DevOps er et ende til ende program som tar for seg hele livssyklusen til programutvikling. Disse fasene er i hovedsak planlegging, utvikling, levering, operering og monitorering av programvaren.

For prosjektstyringen ble funksjonaliteten i Azure DevOps tatt i bruk, dette inkluderer Boards, Repos og Pipelines (Microsoft, u.å. b). Boards tar for seg alt som har med Scrum å gjøre, samt oppgavehåndtering. Boardet har en struktur som er tilrettelagt bruken av Scrum, her finner man både product backlog og sprint backlog. Hver oppgave som blir registrert til bordet er et Product Backlog Item (PBI) og disse blir koblet sammen med den pågående sprinten. For mer innsyn mot en PBI, kan en få oversikt over tilhørende commits som beskriver arbeidet. Azure DevOps ga gruppen et overblikk over de ferdigstilte og gjenværende oppgavene. Funksjoner i DevOps gjorde det mulig å tildele oppgaver. Dette ga oversikt over hvem som arbeidet med de ulike oppgavene. Repos gir git-hosting, mulighet for review av kode, samt diskusjon og samtale rundt pull requests. Pipelines er basert på begrepene *continuous integration* og *continuous delivery* som går ut på å automatisere merging,

testing og levering av kode (Microsoft, 2022 a). Dette ga oss muligheten til å se endringer i systemet fortløpende.

Google Disk ble tatt i bruk for felles strukturering og oppbevaring av dokumenter. Disk gir mulighet for at hele gruppen kan arbeide effektivt sammen i sanntid via skybaserte tjenester (Google, u.å.). Gruppen har tatt i bruk Dokumenter, Regneark og Presentasjoner til ulike formål. Dette inkluderer blant annet møtereferat, logg, notater, estimeringer, presentasjoner og mer. Google Disk har ikke bare blitt brukt internt i gruppen, noen mapper er også delt med veiledere, prosjektleder og andre i Sikri.

2.2.5 Kommunikasjonsverktøy

Gjennom prosjektet har vi brukt flere kommunikasjonsverktøy for samarbeid internt i gruppen, samt for kommunikasjon med Sikri og veiledere. For kommunikasjon oss gruppemedlemmene imellom, har vi brukt Messenger for generell planlegging. Discord ble brukt for møter og samarbeid de dagene vi jobbet hjemmefra, samt for deling av relevante arbeidsressurser. Møter og kommunikasjon med Sikri foregikk på Microsoft Teams. Grunnlaget for valget av disse kanalene, var deres brukervennlighet og at dette er verktøy vi alle har erfaring med fra tidligere.

2.2.6 Loggbok

Gruppen så det som vesentlig å loggføre viktige hendelser som oppstod underveis i prosjektet. Vi førte disse hendelsene i en loggbok da gruppen har erfaring med dette fra tidligere fag og prosjekter, se figur 1. Ved å ta i bruk loggbok fikk gruppen en strukturert oversikt over hva som har skjedd når under prosjektet, se vedlegg 2. Dersom gruppen ønsket å se tilbake på hva som har skjedd, hadde vi en oversikt som over datoer og de viktigste hendelsene. Dette er noe som har vært til stor hjelp for gruppen ved utforming av Sprint Review og Retrospect.

Torsdag, 27. Januar, 9:15-15:20
Hovedfokus: Brukersamtale

- Sketch
- Brukerhistorier
- Prototype
- Flowchart
- Forberede til brukersamtale

Spesielle forekomster:

Avtalt møte med bruker av Element Lekdommer tirsdag 1 februar

Figur 1) *Utsnitt fra Loggbok*

Figur 1 viser til et eksempel på hvordan gruppen har tatt i bruk loggbok. Hver dag skrev vi ned et hovedfokus for dagen og hva som var blitt jobbet med. Det ble også flere ganger lagt ved spesielle forekomster gruppen kom fram til eller som oppsto den dagen.

2.2.7 Timeestimering og loggføring

Basert på antall studiepoeng er forventet antall arbeidstimer i IS-304 500 timer per student (UiA, 2009). Som resultat av dette vil hver uke inneholde 25 arbeidstimer. Derfor så vi det som nødvendig å få en oversikt over tidsbruken til gruppen. Dette ble gjort ved å estimere alle arbeidsoppgaver planlagt for sprintene. Oppgavene som ble estimert var PBI-ene gruppen tok med fra Product Backlog inn Sprint Backlog. Oppgavene ble delt inn i underoppgaver ikke større enn ett dagsverk og ble estimert ved hjelp av DevOps funksjonen Estimate som er en form for Planning Poker (Microsoft DevLabs, 2022). Hvert gruppemedlem kom med et forslag til antall timer per oppgave og gjennomsnittet ble timeestimatet på oppgaven. I tillegg til å estimere deloppgavene ble det også satt hvem som var ansvarlig for oppgaven, hvem som var deltakere og hvilken prioritet oppgaven hadde i sprinten. Timeestimatet ble brukt til å beregne mengden oppgaver som var tilsvarende tiden gruppen hadde tilgjengelig for hver enkelt sprint. Dette ga gruppen en oversikt over omfanget av sprinten og en mer helhetlig oversikt over hvor langt man var kommet i prosjektet.

Underveis i sprintene, samtidig som loggboken ble skrevet på slutten av dagen, loggførte gruppemedlemmene hvor lang tid som var blitt brukt på de ulike PBI-ene. Det ble også fylt inn i et eget skjema hvor mange timer hver enkelt person jobbet.

Skjema var kategorisert etter forskjellige arbeidsoppgaver, som Rapport, Strukturering, Kode, Design, Møter og Forelesninger, se figur 2. Dette ble gjort for å sikre at forventet arbeidsmengde ble oppnådd og oversikt over hva timene hver dag ble brukt til.

		Generelt	Rapport	Strukturering	Kode	Design	Møter	Selvstudie	Forelesninger	Total tid for dag	Total tid for uke	Sprint	Uke
Tirsdag	11.01.2022	2		3		2				7			
Onsdag	12.01.2022	1				1	1		2	5			
Torsdag	13.01.2022			1		5				6	30	Pre sprint	uke 2
Fredag	14.01.2022	2		1		1			2	6			
Lørdag	15.01.2022									0			
Søndag	16.01.2022									0			
Mandag	17.01.2022	6								6			
Tirsdag	18.01.2022	3	1				2			6			
Onsdag	19.01.2022	3	1						2	6			
Torsdag	20.01.2022	2				3		2		7	31	Sprint 1	uke 3
Fredag	21.01.2022	1				2,5			2,5	6			
Lørdag	22.01.2022									0			
Søndag	23.01.2022									0			

Figur 2) *Personlig skjema for timeføring*

2.2.8 Risikoanalyse

I planleggingsfasen av prosjektet ble det gjennom den initielle kontakten med Sikri avklart at en risikoanalyse ville være til stor hjelp. En risikoanalyse skal gi innsikt til hendelser som kan oppstå i løpet av prosjektet (Aven, 2020). Videre skal analysen forklare hvordan disse hendelsene oppsto og hvilke konsekvenser som medfølger. Gruppen nyttet av denne risikoanalysen ved at den bevisstgjorde hver enkelt over ulike risikoer, slik at gruppen videre kunne unngå eller minimere konsekvensene.

Hver risiko ble satt opp og vurdert etter sannsynlighet og konsekvens fra 1 til 5, se figur 3 og vedlegg 3. Ved å multiplisere disse endte vi opp med en sluttsum som skulle vise overordnet risikonivå, se figur 4. Gruppen ble enige om at desto høyere risikonivået var, skulle bevisstheten rundt risikoen øke i tilsvarende grad. Risikoanalysen viser også til en risikorespons for hver risiko som har gjort gruppen handlingskraftige når ulike risikoer oppsto.

Konsekvens	Katastrofalt	5	5	10	15	20	25
	Mer	4	4	8	12	16	20
	Moderat	3	3	6	9	12	15
	Mindre	2	2	4	6	8	10
	Ubetydelig	1	1	2	3	4	5
			1	2	3	4	5
			Sjeldent	Usannsynlig	Mulig	Sannsynlig	Høyst sannsynlig
	Sannsynlighet						

Figur 3) *Risikoanalyse*

Nr.	Risiko	Sann-synlighet 1-5	Konse-kvens 1-5	Risiko-nivå	Risikorespons	Kommentar
1	Mild sykdom	5	1	5	Akseptere	Hjemmekontor
13	Lav kvalitet på kode	4	4	16	Begrense	Lav kvalitet på kode vil oppstå mest sannsynlig tidlig i prosjektet, men vil løses og være en del av læringsprosessen.

Figur 4) *Eksempel på Risikovurderinger*

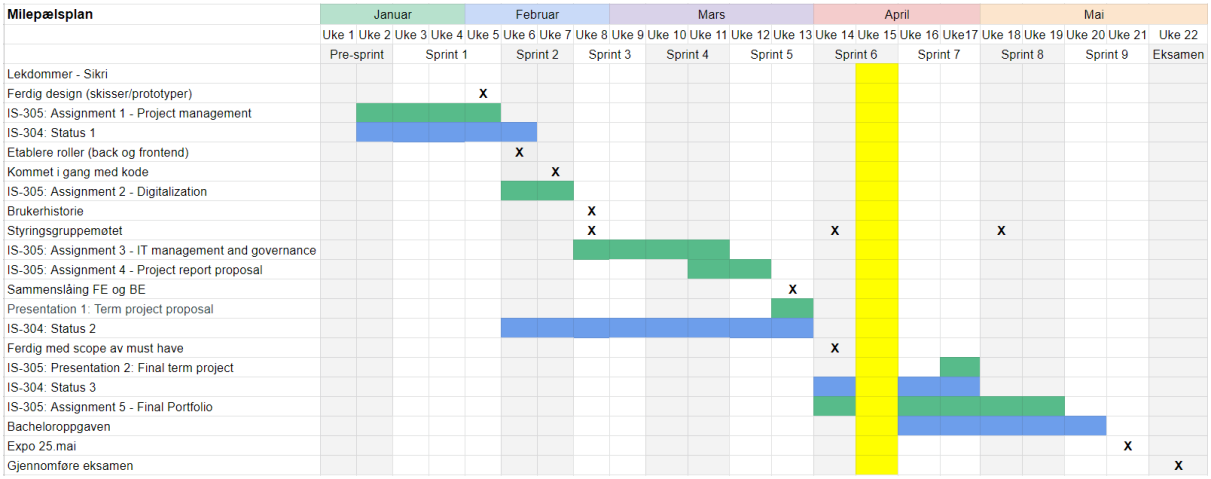
Hvis uforutsette hendelser oppsto under prosjektet, loggførte vi disse hendelsene i en Issue Log, se figur 5. Ved at disse hendelsene ble loggført, kunne vi få oversikt over gjentagende problemer, samtidig som vi kunne se om hendelsene hadde konsekvens for prosjektet.

#	Hendelse	Dato	Kommentar
1	Frafall	12.01	Fredrik fikk ikke deltatt i møte
2	Delvis fravær	17.01	Maren var hos tannlegen
3	Nettverksfeil	18.01	Internettproblemer i starten av møte for Andreas
4	Sykdom	24.01	Sondre er syk
5	Korona	07.01	Andreas og Elias ble kontaktet midt i arbeidsdagen og måtte dra hjem da de er nærkontakter

Figur 5) *Utsnitt Issue Log*

2.2.9 Milepælsplan

Det ble bestemt at gruppen skulle utarbeide en milepælsplan for prosjektet etter ønske fra Sikri. Milepælsplanen tar for seg prosjektets mål og delmål (Karlsen, 2005, s. 279). Den skaper en visuell strukturert oversikt over gruppens oppgaver og milepæler, se figur 6. Dette har resultert i at gruppen har kunnet løfte blikket og få oversikt over prosjektet i sin helhet, istedenfor å kun ha fokus på hver enkelt sprint. De fargede feltene i figuren representerer oppgaver, mens de avkryssede feltene representerer milepæler satt av gruppen. Milepælene ble ikke satt som tidsfrister, men som hva gruppen ønsket å være ferdige med i de ulike fasene. Dette økte motivasjon for gruppemedlemmene, ved at vi hadde noe å strekke oss etter underveis i prosjektet. Milepælsplanen ble presentert under statusmøter med Sikri for å vise til milepæler som var blitt nådd og ha samtale rundt veien videre i forhold til de resterende oppgavene og milepælene.



Figur 6) *Milepælsplan for prosjektet*

2.3 Teknologi

I dette kapittelet vil vi presentere de ulike teknologiene som ble brukt for utvikling av meddommertjenesten, samt begrunnelse for valget av disse. Valgene er basert på anbefalinger fra Sikri med hva de har kompetanse innen og selv tar i bruk. Dette for at Sikri kunne bistå gruppen ved utfordringer som kom til å oppstå, samt enklere overleverelse av systemet ved endt samarbeid.

2.3.1 Frontend

React er et JavaScript-bibliotek for utvikling av brukergrensesnitt som bygger på en komponentbasert programmeringsteknikk (facebook, 2022). React gjør det mulig å designe enkle views for hver tilstand i en applikasjon, og vil effektivt gjengi og oppdatere de riktige komponentene når data endres. Komponentene er løst koblet og uavhengige av hverandre, noe som gjør koden mer forutsigbar, enklere å forstå og enklere å feilsøke.

Typescript er et open-source programmeringsspråk utviklet og vedlikeholdt av Microsoft (TypeScript, u.å. a). Det er en utvidelse av JavaScript og tar i bruk statisk typedefinisjon, i motsetning til JavaScript som bruker dynamisk typedefinisjon. Fordelen med TypeScript og statisk typedefinisjon er at feil i koden fanges opp ved kompilering, noe som reduserer sjansen for bugs (TypeScript, u.å. b).

2.3.2 Backend

.NET 6 rammeverket er en samling av verktøy og teknologier som blir brukt til programvareutvikling utviklet og vedlikeholdt av Microsoft (Microsoft, 2022 b). Rammeverket er open-source og multiplatform som gjør at det kan kjøres på flere forskjellige operativsystemer. .Net rammeverket er stort og har ulike biblioteker for utvikling av web, cloud, mobile og spill applikasjoner. Rammeverket er også flerspråklig og støtter applikasjoner skrevet i C#, F# og Visual Basic.

C# er et objektorientert programmeringsspråk utviklet av Microsoft (Microsoft, 2022 c). Språket er en del av C-familien og har har store likhetstrekk til språk som C, C++ og Java. I motsetning til andre språk i C familien har man i C# muligheten til å definere variabler ubetinget av datatypen. Dette vil si at både tall og karakterer kan bli definert ved bruken av nøkkelordet VAR. C# kompilator vil da tildele den datatypen som passer til det som er blitt definert.

2.3.3 Database

Ved valg av database ble det besluttet at gruppen skulle ta i bruk Microsoft SQL Server(MSSQL). Gruppen hadde ingen formelle krav til hva slags type database som skulle bli tatt i bruk, men ettersom at samtlige gruppemedlemmer har erfaring med relasjonsdatabaser ble det besluttet å ta MSSQL i bruk. MSSQL er også den databaseteknologien Sikri bruker, dermed kunne de bistå med veiledning. En ekstra fordel ved å velge MSSQL er at den kommuniserer godt med .Net (backend) ettersom at begge teknologier eies og utvikles av Microsoft.

2.3.4 Utviklingsmiljø

Ved valg av utviklingsmiljø var det viktig å velge plattformer som samhandler godt med de programmeringsspråkene og rammeverkene vi tok i bruk. For frontend, og React med TypeScript, vurderte vi ulike utviklingsmiljøer basert på flere artikler om plattformer best egnet for TypeScript. Det var bred enighet i disse artiklene om at Visual Studio Code er et ypperlig utviklingsmiljø for TypeScript. VS Code tilbyr innebygd støtte for Node.js-utvikling med JavaScript, TypeScript og en rekke verktøy for webutvikling som JSX/React, HTML, CSS og JSON (Visual Studio Code, u. å. a). I tillegg tilbyr VS Code støtte for en stor mengde nyttige utvidelser som kommer godt med i utviklingen. Alt dette tatt i betrakting, falt valget på Visual Studio Code for utviklingsmiljø for frontend.

Backend-teamet sto også fri til å velge hvilket utviklingsmiljø de ønsket å bruke. Ettersom vi hadde tidligere positive erfaringer med Visual Studio 2022, var dette første utviklingsmiljøet vi vurderte. Likevel, ønsket vi å se på andre alternativer for å evaluere og velge det som passet arbeidet vårt best. Til slutt sto valget mellom Visual Studio 2022 og JetBrains Rider. På nett var Riders ytelse positivt omtalt i en rekke diskusjonsforum, men Visual Studio var omtalt som litt mer brukervennlig. Etter dialog med Sikri der vi ble opplyst om deres erfaringer med bruk av diverse utviklingsmiljø, landet valget på Visual Studio. Tid er en viktig ressurs i vårt prosjekt, dermed gagner det gruppen at vi ikke trenger å sette oss inn i et nytt utviklingsmiljø.

3.0 Kvalitet

Kvalitet og hvordan man sikrer kvalitet er en stor del av prosjektet. Gruppen valgte å gjennomføre en rekke tiltak for å kontinuerlig sikre at kvaliteten ble opprettholdt. Det viktigste for Sikri og gruppen var å holde tett kontakt både med veilederen og produkteier, dette for å sikre at det vi leverer er noe Sikri faktisk kan ta over og fortsette på. Gruppen har i tillegg til dette ytterlige kvalitetssikringskrav som er tett knyttet opp mot ulike faser av systemutviklingen. Disse vil bli presentert i kapitlene under.

3.1 Kommunikasjon

For å oppnå et godt resultat i vår prosjektgjennomførelse, har det vært viktig med god kommunikasjon. Ved å ha god kommunikasjon internt i gruppen og med oppdragsgiver, unngikk vi misforståelser og sikret fremgang.

3.1.1 Kommunikasjon med Sikri

Underveis i prosjektet la gruppen stor vekt på å opprettholde en god kommunikasjon med våre kontakter i Sikri. For å oppnå dette ble det satt en lav terskel for å kontakte våre veiledere. Dersom man satt fast på en oppgave kunne man enkelt sende en melding til veileder og få svar i løpet av kort tid. I tillegg fikk gruppen tildelt eget kontor i Sikris lokaler, noe som gjorde det lett for våre veiledere å prate med oss personlig. Det ble også opprettet kanaler på Teams hvor alle gruppemedlemmene, samt våre kontakter i Sikri hadde tilgang. Disse kanalene ble kategorisert og nyttig informasjon for begge parter ble kommunisert her. Denne uformelle og lave terskelen for kommunikasjon sørget for at gruppen alltid kunne få hjelp der det var nødvendig, og bisto oss med å holde arbeidet i gang. Det ble også bestemt å gjennomføre et statusmøte annenhver uke. Dette ble gjennomført i etterkant av gruppens interne evalueringsmøte som foregikk i slutten av hver gjennomførte Sprint. I disse møtene ble gruppens funn fra evalueringsmøte samt status på arbeid diskutert. Her fikk vi nyttige tilbakemeldinger og en indikasjon på om vi var på rett vei i prosjektet.

3.1.2 Kommunikasjon i gruppen

Gruppen la stor vekt på at den interne kommunikasjonen skulle være bra. I likhet med kommunikasjonen med Sikri ble det lagt en lav terskel for å spørre andre gruppemedlemmer om hjelp. Vi hadde alle ulike egenskaper og forhåndserfaring noe som gjorde at vi kunne spille hverandre gode. For å oppnå dette ble gruppen enig om å møtes fysisk hver dag dersom dette lot seg gjøre. Som nevnt tidligere fikk gruppen mulighet til å sitte i Sikri sine lokaler, noe som var til stor hjelp. Vi opprettet rutiner for oppmøte og holdt disse gjennom hele semesteret. Vi tok også godt i bruk de ulike kommunikasjonsverktøyene i de tilfellene det ikke var mulig å møtes fysisk. Ved sykdom tok gruppen i bruk skjermdeling og kamera i Discord og Teams slik at man kunne være til stede i den grad det var mulig. Disse verktøyene gjorde det også enkelt for gruppen å samarbeide, der man fikk mulighet til å se over hverandres arbeid selv om man ikke satt sammen fysisk.

3.2 Kodestandard

Det ble besluttet å utarbeide en kodestandard før vi startet å programmere for å sikre gjennomgående struktur og kvalitet i koden, se vedlegg 4. Med en felles standard, som ble fulgt av alle på gruppen, kunne vi systematisere koden og gjøre den mer synkronisert. Dette resulterte i at koden ble lesbar for alle som igjen førte til at det ble lettere å samarbeide, oppdage feil og vedlikeholde koden. Det var også et ønske fra Sikri om at vi utarbeidet en kodestandard slik at deres overtakelse av prosjektet skulle gå så smertefritt som mulig.

Ved separat repository for backend og frontend kunne utviklingsteamene utforme én standard for C# og en annen for TypeScript. Kodestandarden ble derfor delt inn i de ulike kategoriene: Generelt, C# og TypeScript, der generell standard blant annet omhandler krav til engelsk språk og kommentarer i koden, se vedlegg 4. Disse standardene omfattet blant annet navngivning for mapper, filer, klasser, variabler og funksjoner, se figur 7.

PascalCase: NewObject;

camelCase: newObject;

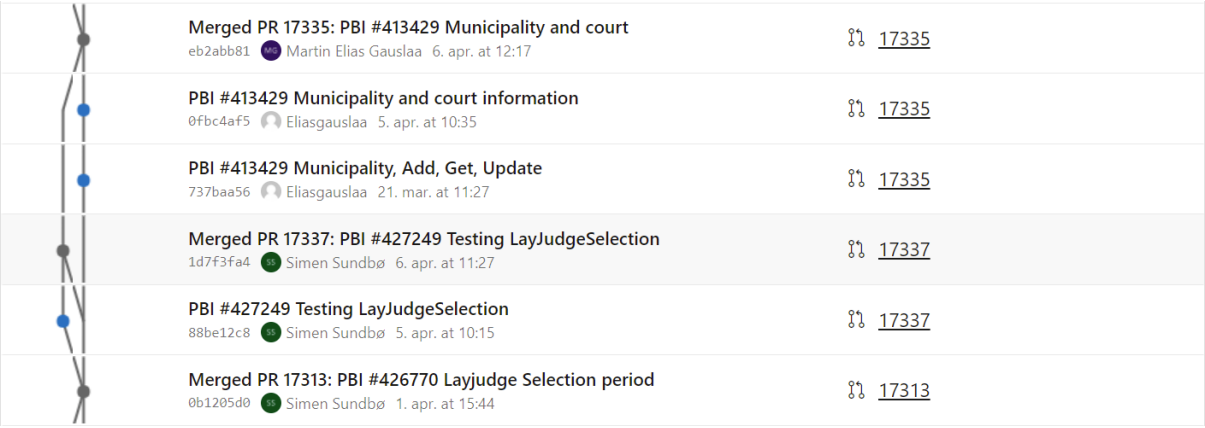
Klasse navn(PascalCase) = ExampleClass

Lokal variabel(camelCase) = localVariable

Figur 7) *Utdrag fra Kodestandard*

3.3 Versjonskontroll

Versjonskontroll er en viktig del av systemutviklingen og handler om å spore og administrere endringer i programvarekoden (Atlassian, u. å.). Git er det mest brukte versjonskontrollsystemet og sporer endringer som gjøres i filene slik at man har oversikt over hva som har blitt gjort, se figur 8 (Git, u.å.). Git gjør også samarbeid lettere, da arbeidet til flere personer kan slås sammen til en hovedkilde. Gruppen har sammen med Sikri valgt å bruke Git som versjonskontrollsystem. Med repositoriene våre hostet i Azure Repos med Git integrert, var dette et åpenbart valg. Vi besluttet å utarbeide en Git-guide med en felles standard for hvordan vi skulle jobbe med Git, se vedlegg 5.

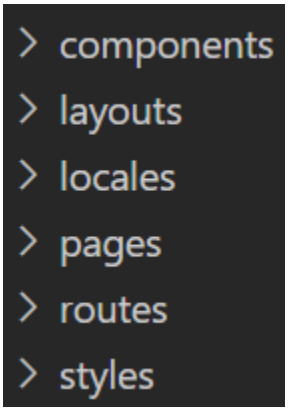


Figur 8) *Utdrag av Commits fra Azure DevOps*

Sikri tok på seg ansvaret med å opprette separate repositories for backend og frontend slik at dette var klart til å klones til våre maskiner. Vi fikk også en innføring i hvordan Azure DevOps brukes i samhandling med Git for å sikre at vi fulgte Sikris praksis rundt dette. Denne praksisen går ut på å bruke brukerhistoriene og deres tilhørende product backlog items (PBIer) ved opprettelse av en ny feature-branch. Feature-branchen er en kopi av main-branchen hvor vi jobbet med nye funksjoner til disse var klare til å integreres med main. Én PBI tilsvarer én feature-branch. Sikri ga oss en standard for navngiving av branches og commits, som blant annet omhandlet å inkludere PBI-nummeret i navnet. På denne måten hadde branch og PBI en direkte kobling og vi kunne enkelt holde styr på hva som ble jobbet med og hvilke PBIer som var ferdig.

Så fort en PBI var ferdigstilt ble den pushet opp til repoet i DevOps. For å kunne si at PBIen var ferdig brukte vi akseptansekriteriene til brukerhistorien for å kontrollere at disse var oppfylt. Videre ble det opprettet en pull request slik at veileder og andre i teamet kunne se over koden før feature-branchen ble slått sammen (merget) med main-branchen. Våre veiledere i Sikri hadde her en viktig rolle med å se til at det som var produsert var av god nok kvalitet. Veilederne ble satt til "required reviewer" som medførte at vi ikke kunne merge før den var godkjent. Kommentarfunksjonen i Azure Repos ble brukt aktivt for å gi tilbakemeldinger på hva som burde endres i koden før merging med main-branchen. Når tilbakemeldingene var fikset opp i, var branchen klar til å bli merget.

3.4 Systemarkitektur

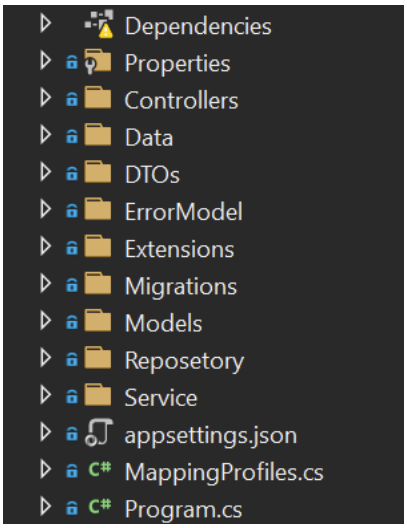


Figur 9) *Mappestruktur til frontend*

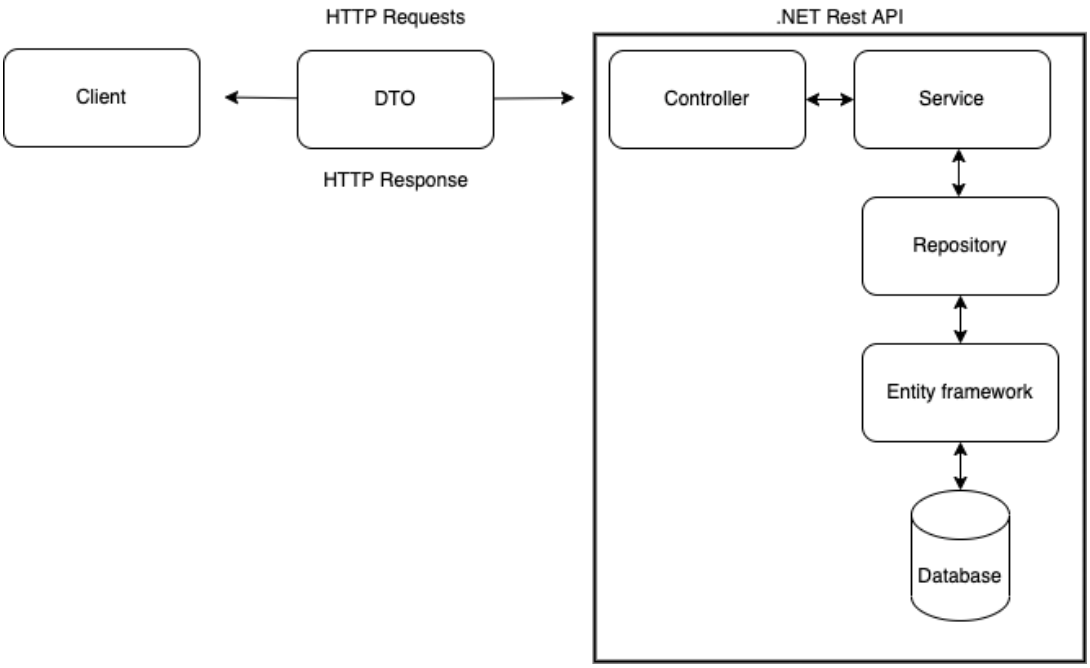
For filstrukturen i frontend hentet vi inspirasjon fra Navdeep Singh Gill's "Folder Structure for React Project" (Gill, 2021), og tilpasset det til vårt prosjekt. /**components** inneholder alle komponentene som brukes i ulike filer, se figur 9. /**layouts**-mappen inneholder navigasjonsbaren som vises på alle sidene i applikasjonen. I **/locales**-mappen ligger lokaliseringsfiler for ulike språk som applikasjonen skal støtte med oversettelser av ord og setninger. **/pages**-mappen inneholder filer for de ulike sidene i applikasjonen der komponenter blir importert til filene og brukergrensesnittet blir til. I **/routes**-mappen ligger AppRouter-filen som definerer URL-adressene og dirigerer brukeren til rett side. Til slutt har vi en **/styles**-mappe som inneholder styling som gjelder for hele systemet.

Backendens filstruktur er basert på det kjente og anerkjente designmønsteret Repository-Service pattern, se figur 10 og 11 (Jones, 2021). Dette er et

designmønster som deler business laget av applikasjonen i to separate lag. Det nederste laget er **Repository** som tar for seg all innhenting og lagring av data. Dette er det eneste laget i applikasjonen som er direkte knyttet til databasen. Det er også viktig at hvert **Repository** kun har ansvar for en enkelt modells klasse som befinner seg **/Models**-mappen. Modellene er dem som holder på dataen og representerer et objekt, for eksempel en meddommer. Laget som ligger over **Repository** er **Service**-laget. **Service**-laget er avhengig av **Repository**-laget og får dette injisert som et objekt. **Service**-laget får tilgang til ett eller flere **Repository**-lag for å kombinere og fremstille komplekse datamodeller. Det ytterste laget er **Controllers** og dette fungerer som et endepunkt som tar imot HTTP-forespørsler. **Controllers** har igjen tilgang på **Service-**lagene den trenger for å hente ut riktig data som brukeren spør etter.



Figur 10) *Mappestruktur til backend*



Figur 11) *Visualisering av backend systemarkitektur*

3.5 Parprogrammering

Parprogrammering er når to personer jobber sammen på en og samme oppgave på en maskin (Simek, 2021). Dette har blitt praktisert både når vi sitter fysisk sammen og når vi har hatt hjemmekontor. Backend-teamet har blant annet tatt i bruk funksjonen Live Share for å skrive og redigere kode mens en annen observerer i sanntid (Visual Studio Code, u.å. b). Det at vi har valgt å praktisere parprogrammering har gjort at vi lettere aktivt kunne dele kunnskap opparbeidet fra tidligere erfaringer eller underveis i prosjektet. Vi lærte av hverandre og gjorde det enklere å følge kodestandardene. Med et ekstra sett med øyne kunne vi oppdage feil tidligere i prosessen istedenfor at de dukker opp når koden var ferdig under evaluering av pull request.

4.0 Prosjektgjennomføring

For at prosjektet skulle bli gjennomført på en måte som var mest mulig effektiv og produktiv, har det vært viktig for gruppen å fokusere på kvaliteten av gjennomførelse. Denne delen av rapporten tar for seg gruppens tilnærming til Scrum, analyse, design og utviklingsfasen.

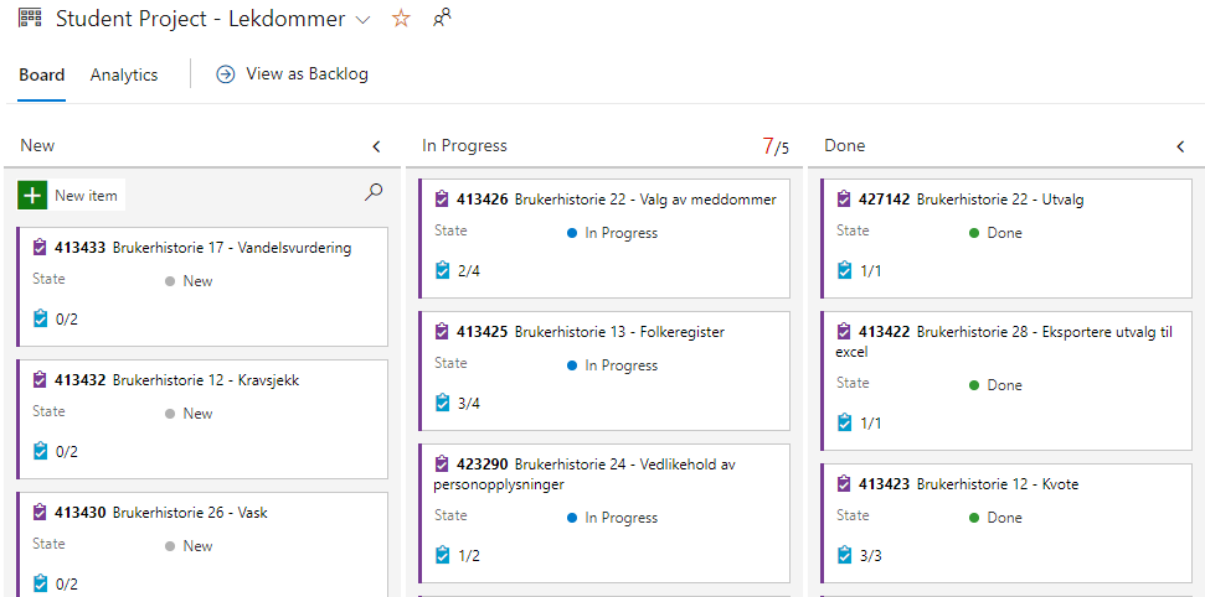
4.1 Gjennomføring av Scrum eventer

I denne delen vil det bli gjennomgått hvordan gruppen har tatt i bruk de ulike Scrum-elementene, samt hvordan de har hjulpet oss med prosjektgjennomføringen. Gruppen har tatt i bruk Azure DevOps som verktøy for gjennomførelse og strukturering av disse elementene. Det som vil bli gjennomgått er Product Backlog, Sprint Backlog, Sprint Planning, Daily Scrum, Sprint Review og Sprint Retrospekt.

4.1.1 Product Backlog

Product Backlog er en dynamisk liste over alle de forskjellige oppgavene som inngår i prosjektet (Scrum, u.å. b). Denne legger grunnlaget for det som skal jobbes med i løpet av prosjektet og blir brukt for å forme en Sprint Backlog hver sprint,

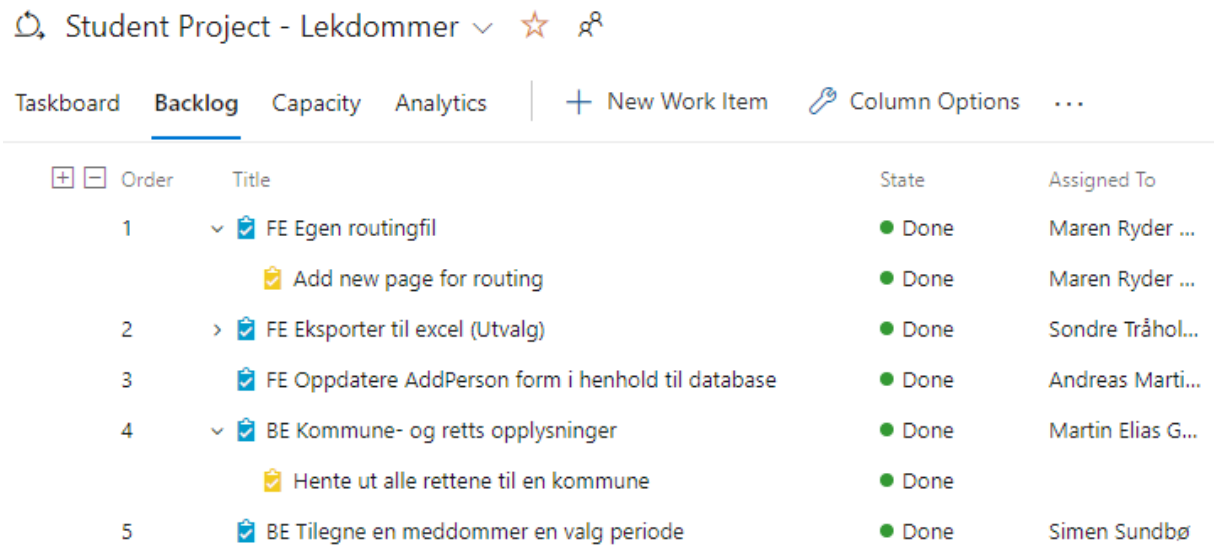
Product Backlog ble utarbeidet og prioritert ut i fra brukerhistoriene som har blitt utarbeidet, se figur 12. Hver brukerhistorie ble en egen item i Product Backlog og arbeidsoppgaver ble definert i hver item. Ting som oppsto underveis ble lagt inn i brukerhistoriene og Product Backlog ble oppdatert deretter. En god og detaljert Product Backlog gjorde at det var lett å holde oversikt over hvilke deler av prosjektet som gjensto til en hver tid.



Figur 12) *Utsnitt av Product Backlog fra Azure DevOps*

4.1.2 Sprint Backlog

Sprint Backlog er en plan for den kommende sprinten, som blir oppdatert etterhvert som arbeidet ble gjort i sprinten, se figur 13 (Scrum, u. å. c). Her hentes de oppgavene som bestemmes i Sprint Planning ut fra Product Backlog og brytes ned i mer definerte underoppgaver. Hver Backlog item har sin egen Sprint Goal, hvor man definerer hvilke kriterier som må oppfylles for at man kan se på den som ferdig. Disse kriteriene tok i vårt tilfelle utgangspunkt i akseptansekriteriene i brukerhistoriene.



Figur 13) *Utkast av Sprint Backlog fra Azure DevOps*

4.1.3 Sprint Planning

Sprint Planning er en event innenfor Scrum rammeverket som handler om å planlegge den kommende sprinten (Scrum, u. å. d). Her er alle gruppemedlemmer tilstede og diskuterer hvilke mål som skal nås i sprinten. Deretter tas aktuelle oppgaver fra Product Backlog som skal bearbeides og legges inn i Sprint Backlog, samt at estimert tidsbruk ble diskutert.

Gruppen la stor vekt på å gjennomføre Sprint Planning så nøye som mulig, ettersom dette la grunnlaget for den kommende sprinten. Hver Sprint Planning ble gjennomført som gruppe og hva som skulle jobbes med ble diskutert i plenum, se vedlegg 6. Gruppen estimerte hvor lang tid oppgavene ville ta, samt hvordan de prioriteres sammenliknet med de andre oppgavene i sprinten. Det å ha klare forutsetninger og mål hjalp med motivasjon, samt at det ble lettere å holde oversikt over gjenstående arbeid i sprinten.

4.1.4 Daily Scrum

Daily Scrum er et 15 minutter langt møte som holdes hver dag, oftest samme tid og sted, hvor hvert gruppemedlem presenterer sin plan for dagen, samt eventuelle hindringer som har oppstått i arbeidet (Scrum, u. å. e).

Daily Scrum ble gjennomført av gruppen hver dag klokken 9:15. Dette gjorde at gruppen fikk oversikt over hvordan hver enkelt lå an med de pågående oppgavene, samt at hindringer ble adressert slik at ingen ble hengende etter. Vi valgte å ikke gjennomføre Daily Scrum slik den er beskrevet i teorien, men heller en løsere variant som ikke tok like lang tid. Gruppen diskuterte overordnede mål for dagen og skrev disse ned. Dette tok som oftest under fem minutter å gjennomføre. På denne måten ble alle gruppemedlemmer bevisst på hva som jobbes med og hvem som gjør hva. I tillegg ble hver dags hovedfokus og spesielle forekomster notert ned i loggboken, se vedlegg 2. Dette sammen med gjennomgangen på starten av dagen ble vår Daily Scrum. Daily Scrum har vært med på å sikre fremgang i prosjektet. Her forsikret vi oss at hvert gruppemedlem hadde oppgaver å jobbe med, samt at gruppen var samkjørte på hvordan gruppen lå an i den pågående sprinten.

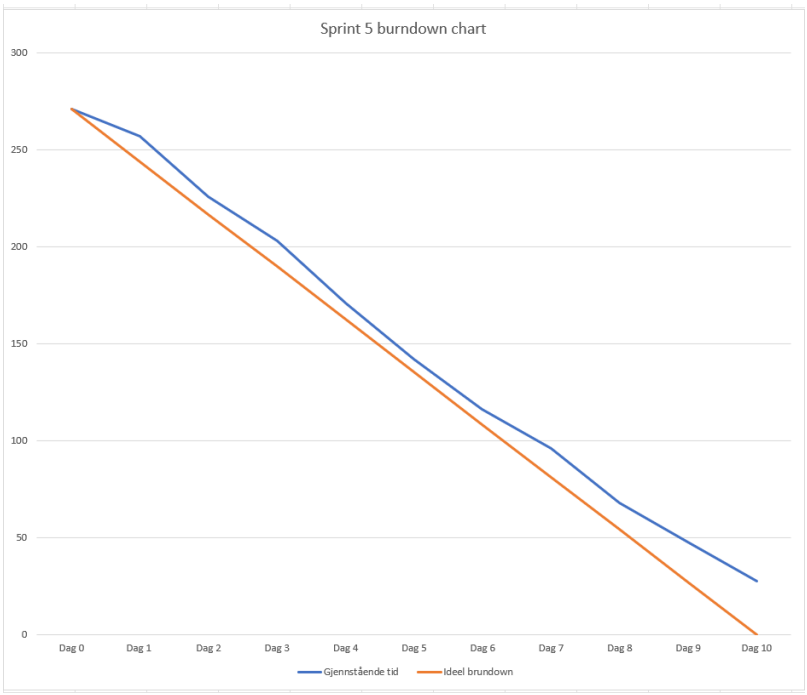
4.1.5 Sprint Review og Retrospect

Sprint Review og Retrospect er eventer som går ut på å evaluere den fullførte sprinten (Scrum, u. å. f) (Scrum, u.å. g). Review har fokus på produktet og arbeidsoppgavene som ble tildelt i Sprint Planning, mens Retrospekt ser på hvordan arbeidet har gått og har fokus på effektivitet og beslutninger som ble tatt.

Gruppen valgte å slå sammen disse eventene til et samlet event. Dette ble gjort ettersom produkteier ønsket innsikt i våre refleksjoner rundt hvordan vi jobbet i tillegg til hvordan produktet hadde utviklet seg. For å spare tid i disse møtene med produkteier gikk vi da gjennom dette i et samlet møte. Annenhver torsdag gjennomførte gruppen intern Review og Retrospect. Førstkommende tirsdag etter at en sprint ble avsluttet, inviterte vi våre kontakter i Sikri til et møte hvor funnene i Review og Retrospect ble gjennomgått. Disse eventene ble lagt stor vekt på ettersom

vi så på det som nødvendig for å sikre god fremdrift, samtidig som Sikri ble oppdatert på fremgangen på arbeidet. Hver Sprint Retrospect gikk gruppen gjennom hvilke arbeidsoppgaver som ble gjort og presenterte dette for hverandre, se vedlegg 7. Deretter ble det tatt en evaluering av hvordan arbeidet har gått.

For å visualisere hvordan gruppen hadde jobbet i hver sprint, ble det laget et Burndown Chart, se figur 14. Dette er en grafisk fremstilling av hvor mye tid som gjenstår av arbeid sammenliknet med hvor mye tid som gjenstår i sprinten. Denne ga oss en oversikt over om vi ble hengende etter og var med på å gi en indikasjon om noe måtte endres. I hver sprints evaluering ble det først gjennomgått hva som har gått bra og hvilke valg som ble tatt som vi vil fortsette med i neste sprint, deretter hva som kunne blitt gjort bedre og som bør endres før videre arbeid. Eksempler på ting som gikk bra og som ble tatt med videre var de rutinene vi satt for arbeid i Sprint 1, hvor vi ble enige om å møte fysisk så ofte som mulig og sitte sammen til klokken 15:00 hver dag. Når det gjelder ting som kunne blitt gjort bedre så vi tidlig at vi måtte tenke mer igjennom hvordan vi gjennomførte estimeringen av arbeidsoppgaver. Dette var noe vi fant ut av tidlig og som ble tydelig forbedret i de kommende sprintene.



Figur 14) *Burndown Chart fra Sprint 5*

4.2 Fase 1 - Analyse og design

I dette kapittelet vil vi gå inn på den første fasen i prosjektet som omhandler analyse og design. Det vil bli presentert problemdomene, systemkrav, brukerhistorier, Scetches, navigasjonskart, designprinsipper, hi-fi pototype, brukersamtale og databasedesign.

4.2.1 Forstå problemdomenet

For å kunne designe og utvikle meddommertjenesten, var det vesentlig at vi satt oss inn i og forstod problemdomenet den nye tjenesten skulle håndtere. Vi brukte derfor mye tid i starten av prosjektet på å lese oss opp på meddommerprosessen og hvilke krav domstolen hadde til kommuner ved valg og rekruttering av nye meddommere. Nyttige ressurser vi tok i bruk var brukerhåndboken til Elements Lekdommer og Norges Domstolers guide for valg av meddommere (Norges Domstoler, u. å. b). Denne forklarte alle aspektene rundt valg, krav til meddommere, frister m.m. på en systematisk og forståelig måte og ga oss nødvendig innsikt og kunnskap om problemdomenet.

4.2.2 Systemkrav

I semesterets første møte ble gruppen blant annet bedt av Sikri om å utforme systemkrav. Brukerhåndboken sammen med problemdomenet ble brukt for å definere de ulike systemkravene for det nye systemet. Etter litt frem og tilbake, med tanke på utformingen av dokumentet, endte vi opp med en tredelt tabell og en fritekst, se figur 15 og vedlegg 8. Systemkravene ble utformet for å definere noen retningslinjer før utformingen av systemet ble påbegynt. Utformingen av systemkrav gjorde av vi måtte sette oss inn i hva systemet var, hvem det skal være for, hvilke teknologier som måtte læres og hva det nye systemet til slutt skulle dekke.

Nr.	Systemkrav	Tilhørende brukerhistorie
1	Meddommerportal	1, 2, 3, 4
2	Fritaksbehandling	5, 6
3	Sikker innlogging	7,8

Figur 15) *Utsnitt fra Systemkrav med nummeringsfelt, kort beskrivelse og tilhørende brukerhistorier.*

4.2.3 Brukerhistorier

En brukerhistorie er et konkret og generelt scenario som beskriver et ønske sett fra brukerens system (Rehkopf, u.å.). Brukerhistoriene kan bli brukt til å hjelpe utviklere å forstå hvordan produktet skal gi verdi til brukeren. Disse ble utformet i dette formatet; "*Som* [rolle], *ønsker jeg* [å oppnå/ha mulighet til], *slik at jeg kan* [oppleve fordel]". Hver respektive brukerhistorie har akseptansekriterier, disse kriteriene beskriver hva som må oppnås før brukerhistorien er komplett, se figur 16. Brukerreisen tar for seg hvordan en bruker skal oppleve scenariet i det nye systemet, slik at vi som utviklere på et dypere nivå forstår fremgangsmåten.

Hver brukerhistorie ble prioritert etter nødvendighet innenfor systemet. For å oversiktlig kunne prioritere brukerhistoriene kategoriserte vi disse etter MoSCoW-modellen (Benyon, 2019, s. 148). MoSCoW-kravene for vårt prosjekt tar utgangspunkt i disse kravene:

- **Must have:** Minimum funksjonalitet som trengs for å håndtere problemdomenet.
- **Should have:** Funksjonalitet som bidrar til en bedret brukeropplevelse.
- **Could have:** Tilleggsfunksjonalitet som ikke er nødvendig men som kan føre til et mer komplett system.
- **Won't have:** Ideer som ikke vil bli implementert, men som blir tatt med videre.

Nr.	Funksjon	Brukerhistorie	Vilkår for aksept	Akseptansekriterie	Argument	Prioritet
1	Meddommerportal	Som bruker ønsker jeg en portal for potensielle meddommere hvor de kan lese informasjon om meddommerrollen, melde sin interesse og søke om fritak.	En nettside hvor personer kan melde sin interesse for å være meddommer ved å fylle ut et skjema med personopplysninger, samt å søke om fritak. Dette gjør at informasjon som mobilnummer og e-post ikke trenger å legges inn av brukeren.	1. Går inn på informasjonssiden 2. Får valgene; "informasjon om meddommerrollen", "bli meddommer" og "søk om fritak"	"Should have" fordi en meddommerportal vil forenkle og effektivisere mange prosesser i rekrutteringen	Should have
2	Meddommersøknad	Som potensiell meddommer ønsker jeg å kunne søke som meddommer gjennom en egen portal	Et skjema som er tilgjengelig på en offentlig side for personer som ønsker å bli meddommere	1. Navigerer seg til meddommerportalen. 2. Fyller inn skjema med nødvendig informasjon 3. Sender inn til behandling.	"Must have" for at borgere skal kunne søke selv.	Must have
3	Fritakssøknad, meddommer	Som meddommer ønsker jeg å kunne sende inn fritakssøknad fra meddommerportalen ved å fylle inn et standardisert skjema slik at jeg vet hva som må med i søknaden	Systemet skal integrere funksjonalitet for innsending av fritakssøknader.	1. Meddommer trykker på "Fritakssøknad" 2. Logger inn med personinformasjon(BankID) 3. Fyller ut skjema med nødvendig informasjon 4. Trykker på "send".	"Could have" da dette er en funksjon som vil gjøre prosessen mer effektiv, men ikke nødvendig for at systemet skal fungere.	Could have
6	Unik link til fritaksskjema	Som bruker ønsker jeg at borgere får en unik link til fritaksskjema slik at jeg unngår å få fritakssøknader fra tilfeldige borgere som ikke er valgt som meddommer.	En unik link til fritaksskjema, som er vedlagt i meldingen om at en borger har blitt valgt som meddommer.		"Should have" fordi et åpent fritaksskjema som ligger tilgjengelig for alle kan bli misbrukt.	Won't have

Figur 16) *Utsnitt fra brukerhistoriene*

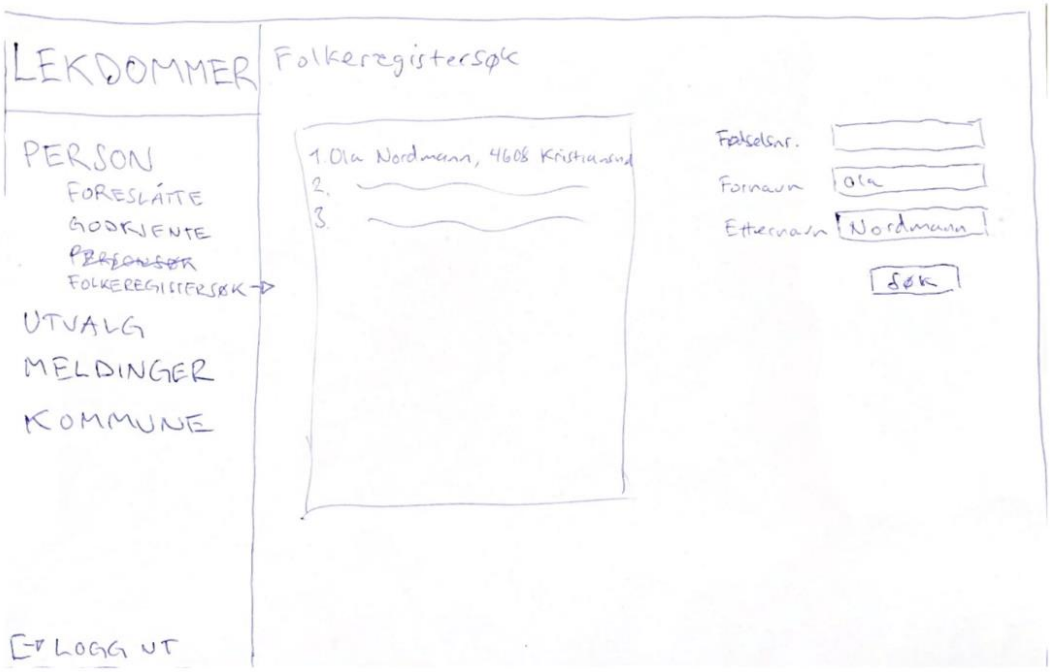
Før brukerhistoriene kunne bli opprettet, måtte gruppen bruke tid på å innhente informasjon om meddommertjenesten gjennom den eksisterende brukerhåndboken, samt informasjonen som var tilgjengelig på domstol.no. Videre ble det opprettet grunnleggende brukerhistorier og gjennom kommunikasjon og oppfølging av Sikri ble disse brukerhistoriene forbedret, samtidig som nye ble opprettet.

Brukerhistoriene spiller ikke bare en stor rolle i planleggingsfasen, men også videre i arbeidsfasen. Disse ble utgangspunktet for både design og utvikling av produkt. Hensikten var at alle brukerhistoriene skal dekke hele omfanget av systemets behov, sett fra brukerens perspektiv. Ved hjelp av brukerhistoriene unngår vi unødvendig problemstrukturering, ved å oversiktlig dekke brukerens behov.

4.2.4 Sketches

I løpet av Pre-Sprinten kom gruppen til enighet om at den beste måten å iverksette analyse- og designfasen på, var å utvikle sketcher. Sketcher er enkle tegninger som er hjelpsomme ved at de legger til rette for at man enkelt kan visualisere tanker og ideer (Benyon, 2019 s. 184). Hver sketch skulle ta utgangspunkt i hva hvert gruppemedlem tenkte var nødvendig i et nytt system, se figur 17. Dette ble basert på informasjon hentet fra brukerhistorier og systemkrav. Hensikten bak de respektive sketchene var å kombinere det beste fra de ulike resultatene, slik at man kunne skape en felles visjon for hvordan et nytt produkt vil se ut. Fordelen ved å la hvert

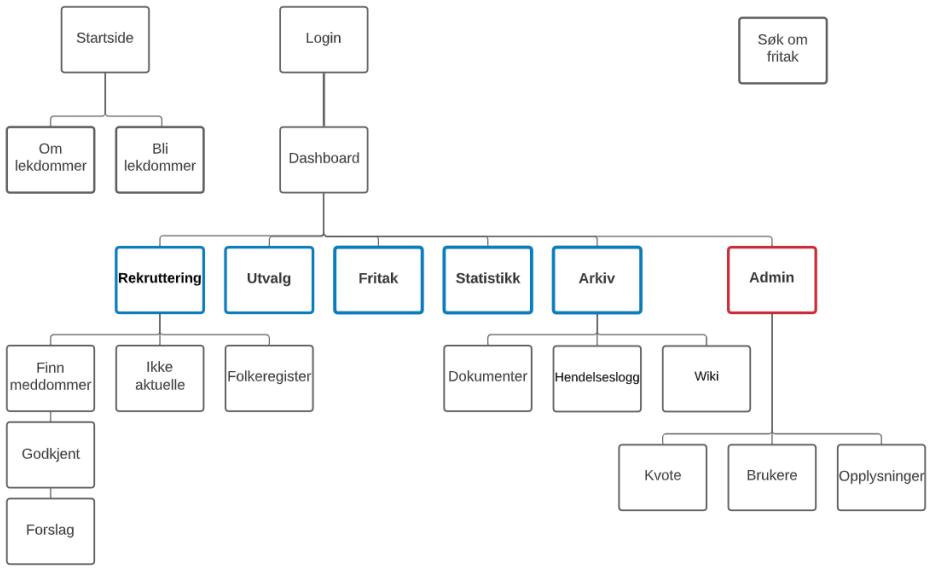
gruppemedlem sketche for seg selv, var at man fikk frem fem ulike synspunkter som sto fritt fra påvirkning av andre på gruppen. Dette ble samtidig en slags idemyldring der vi tilrettela for innovasjon og forbedringer på eksisterende aspekter. Ved at gruppen lagde sketcher til systemet, kunne disse bli tatt i bruk for en raskere utforming av prototypen.



Figur 17) *Eksempel på Sketch laget av gruppen*

4.2.5 Navigasjonskart

For å skape bedre oversikt over det nye systemets utforming, konstruerte gruppen et navigasjonskart i løpet av Sprint 1. Et navigasjonskart fokuserer på hvordan en bruker kan navigere seg gjennom de ulike sidene av et system (Benyon, 2019, s. 191). Navigasjonskartet ble opprettet så fort gruppen hadde kommet til enighet om hvilke sider som kom til å være nødvendige i et endelig produkt. Fordelen med å lage dette kartet var å gi gruppen mer oversikt og enkelt avdekke mangler. Ved siden av dette kunne navigasjonskartet videre beskrive vår visjon overfor produkteier, se figur 18.



Figur 18) *Navigasjonskart*

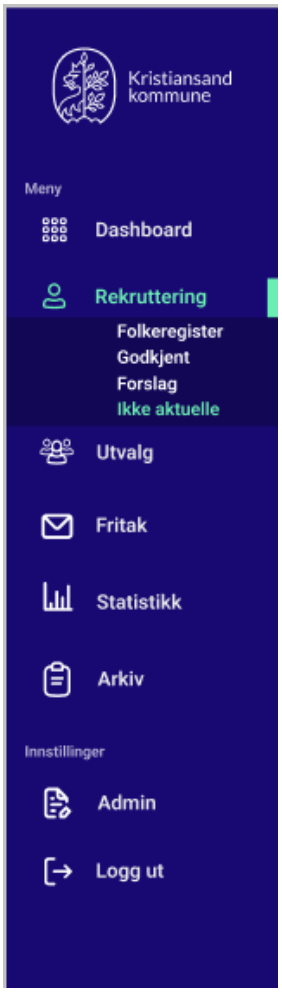
4.2.6 Designprinsipper

WCAG-retningslinjene ble tatt i betraktning da gruppen utformet designet av den nye meddommertjenesten. WCAG er en samling med retningslinjer på hvordan man utformer og designer applikasjoner slik at all data er tilgjengelig for alle uavhengig av funksjonsevne (Henry, 2022). Retningslinjene til WCAG dekker store deler av design og brukervennlighet.

Gruppen har valgt å fokusere på kravene rettet mot design og brukeropplevelse. Dette er krav som valg av farger, titler på sider, plassering, tekststørrelse, input felter og logisk flyt på siden. I første omgang la gruppen fokuset på rekrutteringsprosessen av meddommere. Dette skulle bli en stegvis prosess som tar brukeren gjennom systemet i en logisk rekkefølge, se figur 19. Farger, størrelser og plasseringer av ulike elementer har blitt tatt i betraktning i henhold til WCAG kravene som stilles.

Figur 19) *Stegvis rekrutteringsprosess*

Gjennom studieløpet har gruppen tatt i bruk Benyon's designprinsipper når det kommer til å utforme brukervennlige systemer. Benyons Designprinsipper er 12 ulike retningslinjer og konsepter som man kan ta i betraktning når man evaluerer et systems utforming og design (Benyon, 2019, s. 114) Ettersom meddommertjenesten er et system som benyttes for å gjennomføre praktiske oppgaver effektivt, er designprinsipper som *Navigation* essensielle. Dette designprinsippet omfatter hvordan systemet støtter brukeren i å navigere seg gjennom systemet. Det nye systemet oppnår dette ved å blant annet ha en steg-basert prosess for å tilegne personer meddommerrollen, se figur 19. Fordelen ved dette er at brukeren føler han/hun har kontroll og vet hva og hvordan dette skal gjennomføres. Designprinsippet *Consistency* skal sørge for konsekvent utforming av funksjoner. Ved at undersidene til hovedsidene i navigasjonsbaren alltid blir synlige under, og ikke avviker og for eksempel blir presentert til høyre, gjør at det blir opprettet en standard for bruk, se figur 20. Navigasjonsbaren tar også i bruk designprinsippet *Visibility,* som viser til hvilke funksjoner som er tilgjengelig. Den viser brukeren hvilken side brukeren befinner seg på, og hvilke man kan navigere seg til. Ved å ha en tydelig og oversiktlig navigasjonsbar kan brukeren navigere seg til ønsket side fra hvor som helst i systemet. Dette går under designprinsippet *Navigation*.



Figur 20) *Navigasjonsbar fra prototypen*

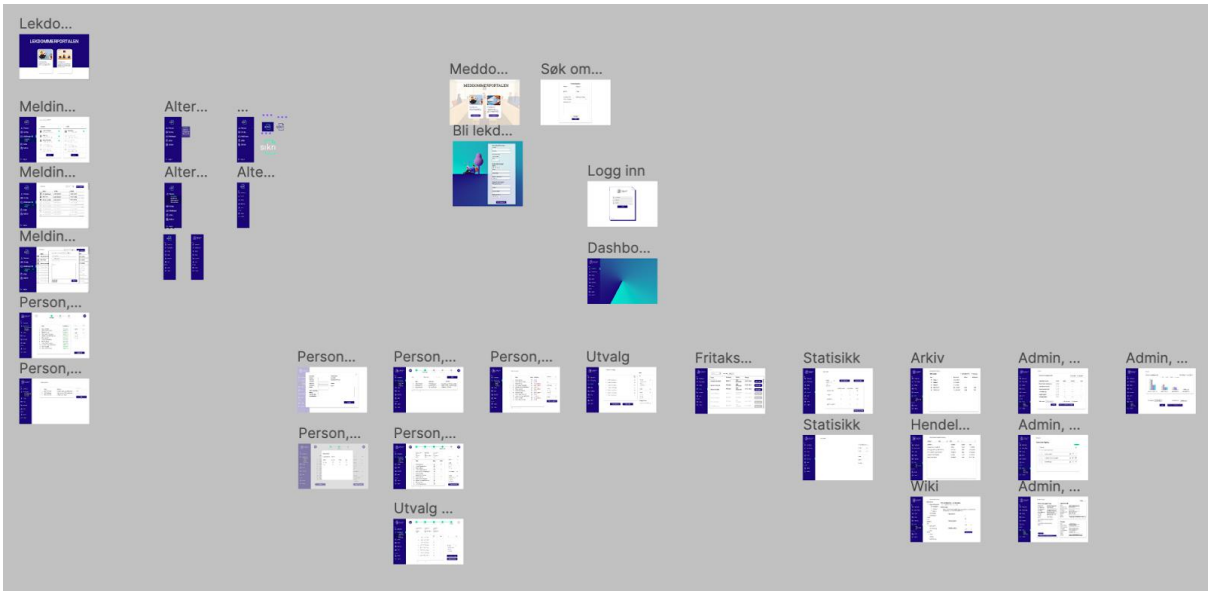
4.2.7 Hi-fi prototype

For utvikling av prototypen valgte vi å ta i bruk Figma, noe som gjorde det mulig for gruppen å jobbe side om side i designprosessen, se figur 21. Prototypen viser et komplett system med funksjonalitet som dekker alle brukerhistoriene, og med utforming tilsvarende designet gruppen ble enige om under skisseringen. Vi har valgt å presentere prototypen med en video som viser alle sidene i meddommertjenesten. Vedlagt ligger link til både videoen og den klikkbare prototypen.

Link til oversikt av prototypen: Prototype

Link til klikkbar gjennomgang av prototypen: Klikkbar prototype

Link til video: https://www.youtube.com/watch?v=57_Rnvvo6gU



Figur 21) *Oversikt over Prototypen utarbeidet i Figma*

4.2.8 Brukersamtale

I Sprint 1 fikk vi tilbud om å gjennomføre en brukersamtale med en bruker av det eksisterende systemet. Denne personen var også med på å utvikle den eksisterende meddommertjenesten og hadde derfor god kunnskap om problemdomenet. Gruppen utviklet en rekke spørsmål for å innhente så mye nødvendig informasjon som mulig, se vedlegg 9. Spørsmålene gikk primært ut på hva de største og mest tidkrevende vondtene var ved den eksisterende løsningen, samt hva han ville ønsket inkludert i en ny løsning. Ved siden av spørsmålene ønsket vi også å gi brukeren fritt spillerom til å demonstrere sin bruk av systemet, slik at gruppen fikk et dypere innblikk i selve meddommerprosessen og hvordan den gjennomføres i dag.

De viktigste funnene fra brukersamtalen var først og fremst hvor innsiktsrik gjennomgangen av det eksisterende systemet var. Ved å høre på hva brukeren hadde å si, fikk gruppen avdekket mangler ved vårt første utkast av den nye meddommertjenesten. Eksempelvis er funksjonalitet for å liste ut alle meddommere

som ligger lagret lokalt i systemet, samt viktigheten av statistikk i utvalgene noen av de tingene som ble presisert. Etter gjennomført brukersamtale diskuterte gruppen også potensiale for automatisering i meddommerprossessen og hvorvidt dette kunne implementeres. I diskusjon med Sikri etter denne brukersamtalen kom forslaget fram om en “steg-basert rekrutteringsprosess” der man på en systematisk måte kan hente ut en person og videre plassere personen i et utvalg.

4.2.9 Databasedesign

For å finne ut av relasjonene mellom de ulike enhetene i det framtidige systemet, konstruerte gruppen et Entity Relationship Diagram (ERD), se vedlegg 10. Et slikt diagram skal representere samlingen av data i tabeller og hvordan de relateres til hverandre gjennom felles verdier (Hoffer, 2019, s.46). Dette diagrammet har vært en visuell representasjon for både gruppen og Sikri, som har gjort flyten av data innenfor systemet enklere å forstå, samt lagt grunnlag for utforming av komponenter.

4.3 Scope

Meddommertjenesten er et prosjekt av såpass stort omfang at det måtte skaleres ned til et gjennomførbart bachelorprosjekt. Omfanget var dermed viktig å etablere før utviklingsfasen, slik at det ikke går unødvendig tid til oppgaver av lav prioritet for Sikri. Gruppen diskuterte seg frem til et omfang, sammen med veileder og produkteier, som var gjennomførbart. Først ble alle brukerhistoriene som ikke var av prioritet Must have fjernet fra gruppens scope for prosjektet. Dette var sider som kunne gi bedre brukeropplevelse, men som ikke er nødvendig for at systemet fungerer. Dette inkluderte blant annet sidene for arkiv og statistikk. Deretter ble brukerhistoriene Sikri allerede har standarder på, som for eksempel innlogging, fjernet fra scopet. Sikri og gruppen diskuterte seg frem til at Must have for gruppas scope skulle inkludere prosessen for rekruttering av en meddommer. De resterende Must Have brukerhistoriene for systemet ble prioritert på nytt for vårt scope. Dermed har vi to ulike omfang, ett for det endelige systemet, samt ett for vårt prosjekt.

4.4 Gjennomføring av sprinter - Fase 1

Den første fasen av prosjektet, som inneholdt en Pre-sprint, Sprint 1 og Sprint 2, ble brukt til å bli kjent med problemdomenet, analysere det og sette i gang med design. Mye av tiden ble satt av til utforming av brukerhistorier, ettersom dette la grunnlaget for arbeidet fremover. Endeproduktet i denne fasen var scopet som skulle legge grunnlaget for hva gruppen skulle utvikle. Det ble gjennomført en brukertest med en representant fra Arendal Kommune, som var kjent med det eksisterende systemet. Her ble prototype vist frem og tilbakemeldinger ble diskutert. Etter dette møte fikk gruppen bekreftet at vi var på god vei, og etter samtaler med Sikri ble vi enige om at vi kunne sette i gang med utvikling av systemet.

I den første fasen la vi mye av grunnlaget for hvordan vi skulle strukturere arbeidet vårt fremover. Vi etablerte gode rutiner, møttes hver dag fysisk dersom annet ikke oppsto og satt konsentrert med arbeid frem til klokken 15. Faste tidspunkter for møter med Sikri ble etablert og vi opprettholdt hyppig og løs kontakt. Av ting som kunne bli gjort bedre var det største punktet timeestimering. Hver Sprint settes i gang med en Sprint Planning hvor alle gjøremål ble estimert. Vi fant fort ut at vi bommet på denne estimeringen og tenkte ut tiltak for neste Sprint Planning. Dette var noe vi så komme og som ville bli bedre etterhvert, noe vi fikk bekreftet i de neste sprintene hvor vi traff bedre. I startfasen av prosjektet hadde ikke gruppen de nødvendige rettighetene for å ta i bruk Azure DevOps optimalt. Dette førte til at vi måtte ty til andre alternativer for prosjektstyringen. Valget falt først på ClickUp, et program med mange av de samme funksjonalitetene for prosjekstyring som DevOps, samtidig som vi tok i bruk regneark i Disk for å notere ned eksempelvis timeestimeringen. Omsider fikk gruppen de rettighetene som trengtes og prosjektstyringen ble migrert over til DevOps.

4.5 Fase 2 - Utviklingsfasen

I avsnittet som følger vil vi vise til implementering, API-requests og unit testing, samt hvordan disse har vært med på å samhandle front- og backend.

4.5.1 Implementering

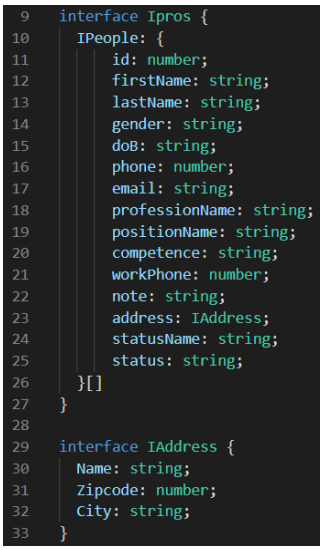
Gruppen har gjennomført et prosjekt hvor de tildelte teknologiene for det meste har vært nytt. Dette var noe vi visste på forhånd, men med relevant erfaring fra lignende teknologier og mål om ny kompetanse var dette en ønskelig utfordring. Det har blitt opparbeidet kompetanse innen prosjektstrukturering gjennom studieløpet, dette gjennom blant annet verktøyene trello, github og monday. Azure DevOps har blitt brukt for å styre dette prosjektet og har funksjoner for mer enn bare strukturering av selve prosjektet. I tillegg til dette ga det oss oversikt over alt sprint-relatert, samt koderepository, commits og pull-requests for koden. Under et oppstartsmøte for frontend og backend med veilederne fra Sikri fikk vi tildelt dokumenter med flere ressurser. Vedlagt var linker til typescript handbook og cheatsheet, samt andre installasjoner vi kunne ta i bruk som ESlint, Prettier og Material UI.

Når det kommer til kodespråk fikk vi tilsendt flere relevante ressurser allerede før jul. Vi kunne anvende den nye teknologien med en gang istedenfor å bruke unødvendig tid på å finne relevante dokumenter og kurs. Ressursene ble tatt i bruk og ga oss et grunnlag for videre arbeid. De første ukene av semesteret gikk til mye planlegging, samt en prototype for systemet. Underveis i denne prosessen satt vi av tid til å bli kjent med de ulike kodespråkene. Ved hjelp av disse kursene hadde vi et bedre utgangspunkt når vi startet den aktive kodingen i slutten av Sprint 2. Det var fortsatt mye som måtte læres underveis i prosessen, men det var lettere å vite hvordan man skulle gå frem for å løse problemer som oppsto.

4.5.1.1 Typescript React

Gruppen opprettet prosjektet ved bruk av kommandoen `npx create react app my-app —-template typescript` (Create React App, 2022). Her fikk vi et skjelett klart til bruk, med alt av nødvendig oppsett for å kunne starte et prosjekt lokalt på datamaskinen.

Som nevnt tidligere er TypeScript en utvidelse av JavaScript og kommer med typesjekking. JavaScript er kjent for å være vanskelig å debugge og kommer ofte med kryptiske tilbakemeldinger som for eksempel; «Uncaught TypeError». Ettersom at TypeScript kommer med typesjekking vil slike feil fanges opp før applikasjonen kjører og dette er noe som har vært til stor hjelp for gruppen. For å definere objekter har vi tatt i bruk interfaces som vist i figur 22. Interfaces har vært en enkel og strukturert måte for å definere hva slags data som objektet skal inneholde.



***Figur 22**) Eksempel på interface brukt i koden*

En annen fordel med React er muligheten for gjenbruk av komponenter som har samme funksjonalitet (ReactJS, u.å.). Vi har en egen mappe i **components-**mappen for gjenbrukbare komponenter. Et par eksempler på gjenbrukbare komponenter som har blitt brukt flere steder i systemet, er ulike inputfelter og knapper. Disse komponentene tar seg av mye av logikken for komponentene, samt styling. Vi trenger derfor kun å fokusere på det som er unikt for der komponenten brukes. Med gjenbrukbare komponenter unngår vi mye kodeduplisering, noe som gjør feilsøking enklere og sikrer et konsekvent design. Dette bygger på konseptet om løs kobling og høy cohesion. Hver komponent skal ha ansvar for kun en oppgave slik at en endring i en komponent ikke skal føre til å måtte gjøre endringer i en annen komponent, samtidig som det skal være høy samhandling mellom komponentene (geekesforgeeks, 2021). Endrer vi for eksempel utseende på en knapp som brukes flere steder, holder det å gjøre endringen i bare én fil.

4.5.1.2 ESlint

Eslint er en utvidelse som ble kontinuerlig brukt gjennom utviklingen. Gruppen vurderte også å ta i bruk prettier, en utvidelse som optimaliserer formatet på koden, mens ESlint analyserer koden statisk for å raskt avdekke feil (Prettier, u.å.) (ESlint, u.å.). Etter komplikasjoner med samarbeid mellom utvidelsene bestemte vi oss for å

kun ta i bruk ESlint videre. Gruppemedlemmene på frontend har i løpet at prosjektet opparbeidet rutine for bruk av ESlint. Etter hver fullførte feature branch ble det kjørt ESlint på alle filene involvert. Flere av feilene ble automatisk fikset, mens andre måtte gjøres manuelt. Først etter at alle filene er sjekket kan man pushe opp branchen og lage pull request. Dette var med på å sikre kvalitet ved at all koden er satt opp etter samme utformingsstandard.

4.5.1.3 Material-UI

Gruppen ble introdusert for Material-UI (MUI), et React UI rammeverk, av Sikri i startfasen av prosjektet (MUI, u.å.). Gruppen har brukt dette for utforming av ulike komponenter. Ved en ny oppgave har vi undersøkt om det vi trenger finnes her før vi eventuelt lager det selv. Selv om det kan være krevende å tilpasse de ulike komponentene til vårt prosjekt, har det spart oss mye tid. Ved at vi har tatt i bruk disse komponentene sikrer vi kode av god kvalitet. Gruppen har tatt utgangspunkt fra MUI i alt fra navigasjonsbar og tabeller til knapper og kort. Det er en informativ nettside som ikke kun viser kode, men også en visuell fremvisning av komponenten og link til codesandbox for enkel lokal utforsking av endring i koden. CodeSandbox er en nettredaktør for rask webutvikling(CodeSandbox, u.å.). Dette er funksjoner som var behjelpelige når vi skulle finne en komponent og integrere den i systemet. Ved å teste eventuelle endringer i codesandbox først slipper vi å sette inn ufullstendig og feil kode. Vi har blant annet tatt det i bruk for å raskere kunne se hva de ulike endringer i koden vil gjøre, da man slipper å måtte lagre endringer i koden og refreshe nettleseren. Når vi har hentet kode ut fra MUI, er det ikke alltid vi har bruk for alt. CodeSandbox har da blitt tatt i bruk for å lettere forstå koden. Ved å fjerne deler av kode og se hva som skjer med produktet har vi fått økt kompetanse for hvordan de ulike delene av koden fungerer.

4.5.1.4 Styling

Det ble i utgangspunktet tatt en besluttning å ikke fokusere på styling, men utover i prosjektet dukket det opp litt styling ved at vi for eksempel tok i bruk MUI. Rundt halvveis i prosjektet diskuterte vi dette med Sikri og hadde et møte om hvordan Sikri

utformer CSS strukturering. Her ble vi introdusert for LESS, en utvidelse for CSS som lar utvikleren definere variabler i CSS kode, noe som viste seg å være vanskeligere å få implementert enn først antatt (Less, u.å.). Vi tok derfor en avgjørelse sammen med Sikri om å ta i bruk Sass istedenfor. Sass hjelper til med å skrive ren, enkel og mindre CSS (Sass, u.å.). Det har blitt lagd egne filer for styling til komponentene. Det har også blir lagd egen mappe for overordna CSS-filer som kan gjenbrukes.

4.5.1.5 React-i18next

Underveis i prosjektet ble det poengtert at produktet, tiltenkt offentlig sektor, må være tilgjengelig på minimum bokmål og nynorsk. Gruppen ble bedt om å sette seg inn i, og implementere React-i18next i koden. React-i18next er et lokaliseringsrammeverk for React basert på i18next (React-i18next, u.å.). Koden blir skrevet på engelsk, selv det som skal bli presentert for brukerne på ønsket språk. Gruppen har valgt å fokusere på oversettelse til bokmål og har laget et eget dokument for oversettelsesprosessen. Det at gruppen har implementert lokalisering for endring av språk, har gjort at Sikri ved senere anledning kun trenger å lage en ny oversettelsesfil fra engelsk til nynorsk og implementere denne. Dette uten å måtte endre store deler av den eksisterende koden. Bruken av react-i18next har videre opprettholdt kodestandarden om å skrive all kode på engelsk, ved at f.eks labels i koden blir oversatt.

4.5.2 API-requests

For HTTP-requests mellom frontend og backend valgte vi å ta i bruk Axios-biblioteket over fetch() API. Axios og fetch()-metoden gjør begge nytten når det kommer til å utføre request og response, men med Axios medfølger det innebygde funksjoner som gjør jobben litt enklere med færre linjer kode. Axios konverterer blant annet data fra en POST-request automatisk til JSON-format. Med fetch() må man kalle på en ekstra metode for å formatere data fra serveren til JSON (Kelhini, 2022). Axios ble brukt til å utføre GET, POST og PUT-requests for å hente ut, legge inn og endre data i databasen. For å kunne utføre spørringene var det viktig at Interface i frontend og

DTO i backend var av samme format. Propertynavn i Interface må tilsvare tilhørende navn i DTO. Dersom disse hadde vært ulike ville ikke spørringen gått gjennom.

4.5.2.1 Postman

Postman er en API-plattform for testing og iterering av APIer (Postman. u.å). Dette var et nyttig verktøy for å teste APIen ved å sende HTTP-requests, slik at vi så om koden utførte de gitte oppgavene. I utgangspunktet brukte vi det lignende verktøyet Swagger, men gikk over til Postman ettersom dette var et bedre illustreringsverktøy under statusmøtene.

4.5.2.2 Automapper

Automapper er et utvidelse for .Net plattformen som løser problemet når man skal mappe data fra et objekt til et annet (AutoMapper, u.å). Det er en liten og fleksibel utvidelse som kommer med funksjoner som gjør mapping både lettere og mer leselig for oss som skriver koden, se figur 23.

```
var layjudges = _mapper.Map<List<LayJudge>>(LayJudgeDTO);
```

Figur 23) *Eksempel på mapping i koden*

Automapperen krever lite konfigurasjon etter at den er lagt til i prosjektet. For at mapperen skal fungere krever den ulike profiler som forteller hvilke objekter som kan mappes til hverandre. Alle disse profilene ble lagt inn i en egen fil som fikk navnet MappingProfiles. I de forskjellige profilene kunne vi definere ulike regler som mappingen følger, et eksempel på dette er vist i figuren 24.

```
0 references
public MappingProfiles()
{
    //CreateMap<Source, dest>

    CreateMap<People, PeopleDTO>()
        .ForMember(dest => dest.DoB, opt => opt.MapFrom(src => src.DoB.ToShortDateString()))
        .ReverseMap();
```

Figur 24) *Eksempel av MappingProfiles*

Bruk av AutoMapper var ikke et krav fra Sikri, men gruppen valgte likevel å ta den i bruk ettersom at noen av oss hadde tidligere positive erfaring med dette. Utvidelsen har spart oss for mye tid ettersom vi ikke lenger manuelt trenger å tildele data til et

nytt objekt, men vi har likevel støtt på problemer. Selv om utvidelsen skal gjøre det lettere er det likevel noe nytt man må lære og sette seg inn i. Debugging har vist seg å være vanskelig, dette skyldes ofte feil i mapping profilene, men det er ikke alltid helt klart akkurat hvor feilen ligger. Her har ressurser som StackOverflow vært til stor hjelp, ettersom at AutoMapper er vidt brukt blant .Net utviklere.

4.5.2.3 Global exception handler

Å håndtere uforutsette feil er svært viktig for oss, ettersom vi skulle opprettholde brukeropplevelsen og kvaliteten av APIen. Dersom en feil dukker opp mens applikasjonen kjører kastes extension, som stopper applikasjonen for å kjøre videre. Når uforventede tilstander oppsto brukte vi try-catch blokker i koden som fange opp feil og behandlet dem slik at applikasjonen ikke krasjet. Global exception handling fungerer på samme prinsipp, men med denne tilnærmingen slapp vi å pakke inn alle funksjonene med exception handling. Ved å implementere dette ble koden mer lesbar og unntaksmessige tilstander ble flyttet inn i en separat fil.

4.5.2.4 Object-relation mapper

Det ble også tatt en beslutning om at gruppen skulle ta i bruk en Object-relation mapper (ORM) for å lettere kunne kommunisere med databasen. En ORM er et abstraksjonslag som tar bort mye av SQL logikken i koden og kobler sammen klasser i koden med tabeller i databasen (Liang, 2021). Manuelle SQL spørringer kan ta tid å skrive, det kan fort bli store og komplekse spørringer hvor sikkerhetsfeil lett kan komme med i spørringen. Ved bruk av en ORM forhindrer man at slike feil kommer med. Vi fikk en anbefaling fra Sikri om å ta i bruke enten Entity Framework eller Dapper som ORM i vårt prosjekt. Etter å ha evaluert fordeler og ulemper med hver enkelt av dem valgte vi å gå for Entity Framework. Entity Framework kommer med mange fordeler, en av disse er at man lettere kan sette deg inn i det uten forkunnskaper (ParTech, 2020). Entity Framework kommer også med funksjoner som lettere lar utvikleren generere databasen ut fra modell klassene i APIen. Denne tilnærmingen blir kalt for Code first og egnet seg bra til vårt prosjekt ettersom det ikke hadde en eksisterende database.

4.5.3 Unit Testing

Ved å gjennomføre tester, verifiserer man at individuelle enheter utfører oppgaven sin korrekt (Kumar, 2018). Dette er essensielt for å minimere antall bugs og videre kvalitetssikre at koden som blir levert, står til forventningene til Sikri.

xUnit er et test-verktøy innenfor .Net rammeverket. Backend-teamet veide opp dette verktøyet mot et annet populært verktøy som heter NUnit. Etter denne vurderingen, sammen med dialog med veileder på Sikri, valgte vi å bruke xUnit. Dette valgte vi for å kunne få hjelp av Sikri om det skulle oppstå behov, ettersom dette er verktøyet Sikri har kjennskap til. Figur 25 viser til oppsettet av en Unit Test skrevet ved hjelp av verktøyet xUnit.

```
[Fact]
0 references
public async Task UpdatePeople_ExistingPerson_ReturnsUpdatedPerson()
{
    var updatePeople = await _peopleRepository.Update(1, TestPeopleData.UpdatePersonData());

    Assert.IsType<People>(updatePeople);
    Assert.Equal("SimenUpdate", updatePeople.FirstName);
}
```

Figur 25) *Eksempel på Unit Test skrevet i xUnit*

For å teste enhetene uten å trenge en kobling opp mot databasen ble det implementert en InMemory Database som forestiller et objekt, se figur 26 (Barbettini, 2016). Fordelen ved dette er at vi har unngått å påvirke den eksisterende databasen ved å gjennomføre tester.

```
0 references
public PeopleRepositoryTests()
{
    var builder = new DbContextOptionsBuilder<DatabaseContext>();
    builder.UseInMemoryDatabase("TestDatabase");

    _context = new DatabaseContext(builder.Options);

    _peopleRepository = new PeopleRepository(_context);

    _context.Database.EnsureDeleted();
    _context.Database.EnsureCreated();

    SeedData(_context);
}
```

Figur 26) *Eksempel på tilkobling til InMemory Database*

4.6 Gjennomføring av sprinter - Fase 2

Fase 2 av prosjektet ble brukt til implementering og utvikling av systemet som ble analysert og designet i fase 1. Denne fasen strekker seg fra slutten av Sprint 2 helt til starten av Sprint 7. Slutten av Sprint 2 ble brukt til å utvikle et ER-diagram som la til grunn strukturen i systemet, før vi fikk på plass en template for frontend. Videre ble relevante verktøy og utvidelser lastet ned samtidig som vi satt av tid til selvstudie, for å bli kjent med disse verktøyene. Hovedfokus for frontend-teamet i Sprint 3 var å få på plass de ulike sidene i systemet, slik at funksjonalitet kunne implementeres senere. Backend-teamet jobbet med funksjonalitet for å legge inn og liste ut personer. Neste sprint gikk til å implementere noe av den funksjonaliteten som ble lagt til rette for i forrige sprint. Hovedfokus for backend-team var å få på plass hoved funksjonaliteten for håndtering av personer, meddommere og kvoter. Sprint 5 ble brukt til å videre implementere den funksjonaliteten som ble lagt til rette for tidligere, samt koble sammen frontend og backend. Frontend-teamet tok også en gjennomgang av filstruktur samt fikk implementert i18next etter anbefaling fra Sikri. Backend-teamet valgte å omstrukturere ERD som ble utarbeidet i fase 1, samt se til at den faktiske databasestrukturen passet over ens med ERD. Sprint 6 ble brukt til å ferdigstille systemet slik at Sikri kunne implementere pipelines.

Gjennom hele fase 2 opprettholdt vi de gode arbeidsrutinene som ble satt i fase 1. Selv gjennom en periode med sykdom og covid-19 klarte vi å opprettholde arbeidsflyt og rutiner. Den gode kontakten med Sikri fortsatte og den lave terskelen for å spørre om hjelp ble opprettholdt. Dette gjorde at vi alltid fikk den veiledningen vi trengte, slik at vi ikke satt fast på enkelte oppgaver. Vi så merkbar fremgang i timeestimeringen som ble gjort i starten av hver sprint etter at vi konkretiserte oppgavene i mindre oppgaver. Etter fase 1 gikk vi fra å jobbe primært som gruppe til mer individuelle oppgaver. Dette førte til at vi måtte gjøre endringer i måten vi estimerte på. Tidligere ble hver oppgave estimert til tid brukt per person, noe som gjorde det vanskelig å få et nøyaktig estimat dersom flere jobbet på samme oppgave (figur x1). Vi bestemte derfor for å videre estimere det totale antall timer brukt uavhengig av antall deltakere, se figur 27 og 28. Etter statusmøtene vi hadde med Sikri fikk vi tilbakemelding om å

stramme opp strukturen i statusmøtene. Dette tok gruppen i betraktning og vi møtte opp mer forberedt til møtene deretter. En klar agenda for hvert møte ble laget, samt at flyten ble bedre.

Sprint 1 | 17.01 - 03.02						
Oppgaver	**Ansvarlig**	**Deltakere**	**Prioritet**	**Estimering**	**Brukt**	Totalt:
Brukerhistorier	Andreas	Alle	2	2,5	6	30
Risikovurdering	Sondre	Sondre	3	1,5	1	1
Flowchart	Simen & Elias	Simen og Elias	1	3	1	2

Figur 27) *Utsnitt av estimering fra Sprint 1*

Sprint 5 | 18.03 - 31.03						
Oppgaver	**Ansvarlig**	**Deltakere**	**Prioritet**	**Estimering**	**Brukt**	Totalt:
Sprint retrospekt og planning	Maren	Alle	1	10	10	10
i18next for localization	Andreas	Andreas, Maren	1	6	7	7
Oppdater personopplysninger	Simen	Simen	1	6	6	6
Oppdatering av databasen	Simen & Elias	Simen & Elias	1	5	10	10

Figur 28) *Utsnitt av estimering fra Sprint 5*

5.0 Refleksjon

I dette kapittelet vil vi presentere våre refleksjoner opparbeidet gjennom bachelorprosjektet. Dette inkluderer hvordan vi har sikret kvalitet, tanker rundt rollefordeling, prosjektstyring og læringsutbytte. Et gjentagende tema vil være utfordringer og hvordan vi har håndtert disse.

5.1 Kvalitet og kvalitetssikring

Vi erfarte at vi flere ganger gjorde endringer ut i prosjektet for å bedre følge Sikri sine standarder. Det ble for eksempel brukt tid på omstrukturering av mappestrukturen i frontend for å tilpasse den til et helhetlig system og Sikri sine standarder. Vi ble også introdusert for nye teknologier som i18next senere i prosjektet og måtte gjøre endringer i koden for å få dette implementert. Ved at vi ikke ble introdusert for all teknologi fra begynnelsen, fikk gruppen ett fokusområde om gangen, der vi slapp å bli overveldet av mengden ny teknologi. Likevel ble mye tid brukt på å sette seg inn i

disse nye teknologiene etterhvert som de ble et krav. Dette gjorde det enklere å anvende de nye teknologiene på en god måte. Hvis gruppen skulle ha blitt introdusert for alle teknologiene fra starten av, hadde det vært nødvendig med en grundig gjennomgang av hver teknologi og hvordan Sikri anvender disse.

Et tiltak vi er fornøyd med å ha innført er de ukentlige statusmøtene med Sikri. Disse bidro til å opprettholde kvalitet gjennom prosjektet og var svært nyttige for å holde Sikri oppdatert på utført arbeid. I disse møtene fikk vi presentert og diskutert ideer og løsninger til meddommertjenesten. Dette var viktig for både gruppen og Sikri, da det ga oss god veiledning samtidig som at Sikri ble forsikret om at arbeidet ble gjort og var av tilfredsstillende kvalitet.

5.2 Rollefordeling

Gruppens fordeling i et frontend team og et backend team ble gjort i starten av semesteret, men ikke satt i praksis før fase 1 var fullført. Denne inndelingen ble gjort for å bedre arbeidsflyten, ved at hver person fikk et mindre område å fokusere på. Dette gjorde at man kunne fordype seg mer i det spesifikke temaet, i stedet for å lære litt om mye. Rollene ble fordelt etter preferanser og forkunnskaper ble også tatt i betraktning. En ulempe med å dele inn i slike teams er at man ikke tilegner seg den breddekunnskapen man ellers ville gjort.

Tidlig i prosjektet besluttet gruppen å ikke ha noen utpreget leder med mer ansvar enn andre. Vi ble enige om at en Scrum master, som hadde ansvar for å opprettholde kontakt med våre veiledere og eventuelle andre aktuelle personer, ville være hensiktsmessig, men så ikke verdien med en «leder». Ettersom gruppen etter hvert delte seg inn i to teams, ett for backend og ett for frontend, ville det blitt vanskelig for en utpreget leder å holde oversikt over alles arbeidsoppgaver. Derfor la vi ned gode rutiner for arbeid og sørget for at dersom gruppen ble avsporet var vi kjappe med å si ifra. Dersom det oppsto uenigheter i gruppen ble dette ofte løst ved at flertallet gikk gjennom, i stedet for at en utpekt leder tar en beslutning. Vi så at gruppen sporet av mer enn ønskelig og at en utpreget leder som sier ifra og hjelper oss holde fokus, kunne vært til hjelp. Det var derimot ikke til stor hindring og effekten av en leder ville vært minimal.

5.3 Prosjektgjennomføring

Når gruppen ser tilbake på gjennomførelsen av prosjektet så legger vi spesielt merke til hvordan vi har anvendt teori fra de ulike emnene i bachelorforløpet fra UiA og brukt det til vår fordel. Gjennom å starte smått med bla. sketching og brukerhistorier la vi et grunnlag som gjorde utviklingsperioden så smertefri som mulig. Vi ville derfor ikke vært foruten disse analysene og teknikkene, men dette tatt i betraktning kan gruppen vurdere i hvilken grad det ville hjulpet å ta i bruk mer fra teorien. De tiltakene gruppen valgte å anvende i løpet av prosjektgjennomføringen ble valgt for å tilfredsstille balansegangen mellom å støtte seg på mest mulig teori, samtidig som det relativt korte tidsrommet på oppgaven ble tatt i betraktning.

5.3.1 Risikovurdering

Som nevnt lagde gruppen en risikovurdering for å øke bevisstheten av risikoer som kunne oppstå underveis i prosjektet. Risikoen om lav kvalitet på kode fikk høy sannsynlighet og konsekvens av gruppen og fikk den høyeste risikoen av alle. Vi satte risikoresponsen til begrens, da vi tenkte det var uunngåelig at det ikke skulle forekomme med så mye ny teknologi, men at det kunne bli bedre ved å ta grep. Det ville være en læringsprosess som ville resultere i bedre kode utover i prosjektet. Det at vi lagde denne risikovurderingen gjorde oss mer bevisste når problemer oppstår, ikke bare de store, men også de litt mindre som mild sykdom og koronakarantene. Først og fremst klarte vi å begrense noen risikoer slik at de ikke skjedde, som kommunikasjonssvikt og konflikter, men når risikoer forekom hadde vi allerede diskutert og kommet frem til løsning som da bare var å følge. Basert på erfaringene gjennom prosjektet tror vi at tiden det tok å utarbeide risikovurderingen har helhetlig spart oss for tid. Vi ser på utarbeiding av risikovurdering i begynnelsen av prosjektet som en suksess og hadde derfor benyttet oss av dette igjen ved et fremtidig prosjekt.

5.3.2 Estimering

Gruppen hadde utfordringer med estimere oppgavene for hver sprint med korrekt mengde tid. Erfaringen var at vi ikke direkte underestimerte mengden tid en oppgave ville kreve, men at vi ikke visste hva oppgavene vi estimerte besto av. Dermed satt vi igjen med lærdom at ved å konkretisere oppgaver i betraktelig høyere grad, ville estimeringen bli mer nøyaktig. De første sprintene hadde Sprint Backlogs med lite konkrete oppgaver som førte til at store deler av arbeidet ikke kunne spores til en oppgave. Konkretiseringen av oppgaver hadde derfor gitt et bedre innblikk i arbeidet som ble gjennomført sprinten.

5.4 Konklusjon og læringsutbytte

Det produktet som ble overlevert til Sikri var på ingen måte et komplett produkt. Dette var både gruppen og Sikri inneforstått med at kom til å være tilfellet med tanke på omfanget på oppgaven som ble gitt i starten av prosjektet. Systemet i skrivende stund består av grunnleggende funksjonalitet, design og en struktur som gruppen anser godt nok egnet for videre utvikling av Sikri. Våre ambisjoner rundt å fullføre det nye scopet ble ikke oppfylt. Scopet ble utformet på grunnlag av informasjonen vi hadde på det tidspunktet, men ettersom systemet ble større og mer komplekst, innså vi at dette var ikke gjennomførbart for oss. Dette var ikke et nederlag for gruppen ettersom hvert enkelt medlem sitter igjen med en verdifull erfaring, samt ferdigheter innenfor utvikling og ikke minst samarbeid.
Denne erfaringen sammen med disse ferdighetene gjør gruppemedlemmene i stand til å prestere bedre om et nytt prosjekt skulle oppstå.

6.0 Egenvurdering

Under dette semesteret har vært medlem bidratt i like stor mengde. Dette på tross av at hvert gruppemedlem har hatt ulike utgangspunkt når det kommer til relevante ferdigheter og interesser. Samarbeidet som har tatt plass under dette prosjektet er et resultat av hvordan vi har bistått hverandre på tvers av styrker og svakheter. Vi er alle tilfreds, med hvert medlems bidrag innenfor analyse- og designfasen, utvikling og

rapportskriving. Under skal hvert medlem videre utdype hvordan de har bidratt i prosjektet.

Andreas Hovland Martinsen

Jeg har i dette bachelorprosjektet vært en del av frontend-teamet, der jeg har vært med på å utforme brukergrensesnittet til applikasjonen. I tillegg jeg har bidratt gjennom hele analyse- og designfasen på lik linje med de andre på gruppen. Jeg har fått muligheten til å jobbe med relevant og moderne teknologi som ReactJS, TypeScript og Azure DevOps, og har dermed opparbeidet meg nye ferdigheter, kunnskap og gode erfaringer når det kommer til systemutvikling og prosjektstyring.

Martin Elias Gauslaa

Under dette bachelorprosjektet har jeg opplevd utfordring og mestring, men viktigst av alt har tilegnet meg nye ferdigheter og forbedret mine eksisterende ferdigheter. Jeg har jobbet på backend-teamet sammen med Simen og har sammen med han analysert, utviklet og tatt beslutninger for å opprette en backend som står til Sikris krav og forventninger.

Maren Ryder Lislevand

I løpet av bacheloroppgaven har jeg tilegnet meg masse ny kunnskap og kjent på både opp- og nedturer. Jeg har blitt mer bevisst på mine styrker og svakheter, og kan jobbe videre med disse etter prosjektet er ferdig. Sammen med alle de andre gruppemedlemmene har jeg deltatt på alle Scrum-eventene og statusmøter med Sikri for å sikre fremgang og kvalitet. Jeg var en del av frontend-teamet og har her jobbet med TypeScript React for å utvikle deler av brukergrensesnittet til systemet.

Simen Sundbø

Dette bachelorprosjektet har vært utrolig lærerikt og i løpet av de 5 månedene har jeg opparbeidet meg nye egenskaper jeg tar med meg videre. Gjennom dette prosjektet

har jeg hatt rollen Scrum Master, hvor jeg har hatt ansvar i å følge opp de ulike scrum elementene. Viderer har jeg for det meste jobbet med backend, hvor mye tid har gått til å sette seg inn i de respektive teknologiene.

Sondre Monge

Gjennom dette bachelorprosjektet har jeg både fått tatt i bruk den kunnskapen og erfaringen jeg har opparbeidet gjennom studieløpet, samt at jeg har tilegnet meg verdifull kunnskap og erfaring som kan tas med videre i livet. Jeg har operert som en del av frontend-teamet, hvor vi har utviklet og utformet prosessen for rekruttering.

7.0 Uttalelse fra oppdragsgiver

13.05.22

Statement from Sikri

Sikri is a software company, specializing in delivering essential digital solutions to the Public and Private sector. We were established in 2020 but our history dates back more than 20 years. We have a strong track record in delivering the market leading system for case management and archiving and are extending our reach into a wider range of software and solutions.

Sikri has had a close cooperation with The University of Agder (UiA) for several years. For Sikri this includes having students or student groups stay with us during their bachelor year. Through their stay, they work on their project while getting guidance and resources from Sikri. The projects are generally related to a business areas Sikri is involved in. For Sikri this has been a fruitful arrangement. It enables us to contribute to the important work UiA does in offering good education possibilities for people who wants to work in IT and at the same time giving us new insight and fresh perspectives from engaged students.

Spring 2022 the group consisted of Martin Elias Gauslaa, Andreas Hovland Martinsen, Maren Ryder Lislevand, Sondre Monge, Simen Sundbø. Sikri was looking to renew and preferably create a new version of its application "Lekdommer", something which the students showed great interest in developing. Except some technical requirements for Sikri to be able to continue develop and maintain the application, their project had few or no guidelines as to how the project should be conducted.

The group started off by exploring the term Lekdommer/Meddommer gaining great knowledge of the domain and what laws apply to this. They also performed research related to the existing application in order to gain knowledge of what core functionality was needed by their application and new features desired by users of the current system. After all information had been gathered, the students started working on developing and prioritizing user stories. Following this, they set up frameworks of both backend and frontend of the application and performed database modelling. The students have been well-organized, collaborated good as a team and worked thoroughly through the whole project. They have gained good knowledge in React and ASP.NET Core respectively.
The students were able to develop many of the necessary functionalities both in backend and frontend in the relative short timeframe of the project. They have created an extensive list of user stories meaning Sikri are well equipped to continue the work where they left off.
We found the students to be committed, positive and to have a good skillset.

Best regards

Fredrik B. Werpen
Product owner
Sikri AS

Marius Holen
Technical Advisor

Sikri AS · Vollsveien 4, 1366 Lysaker
87 65 43 21 · post@sikri.no · www.sikri.no
Org.nr.: 922 308 055

8.0 Litteraturliste

Atlassian (u. å.). *What is version control?* Hentet 3. mai 2022 fra https://www.atlassian.com/git/tutorials/what-is-version-control

AutoMapper. (u.å.). *What is AutoMapper?* Hentet 12. mai 2022 fra https://docs.automapper.org/en/latest/Getting-started.html

Aven, T. (2020, 5. november). *Risikoanalyse.* Store norske leksikon. https://snl.no/risikoanalyse

Barbettini, N. (2016, 9. august) *Using Entity Framework Core as an In-Memory Database for ASP.NET Core.* Stormpath. https://stormpath.com/blog/tutorial-entity-framework-core-in-memory-database-asp-net-core

Benyon, D. (2019). *Designing User experience: A guide to HCI, UX and interaction design* (4. utg.) Pearson

CodeSandbox. (u.å.). *Where teams build faster, together.* Hentet 3. mai 2022 fra https://codesandbox.io/

Create React App. (2022, 12. januar). *Adding TypeScript*. https://create-react-app.dev/docs/adding-typescript/

ESlint (u.å.). *Find and fix problems in your JavaScript code*. Hentet 4. mai 2022 fra https://eslint.org/

Facebook. (2022, 3. april) *React*. Github https://github.com/facebook/react/blob/main/README.md

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

Gill, N.S. (2021, 11. oktober) *Understanding ReactJs Project Structure and Folder Setups.* Xenonstack. https://www.xenonstack.com/insights/reactjs-project-structure

Git (u.å.). *Git.* Hentet 15. mai 2022 fra https://git-scm.com/

Google. (u.å.). *Enkel og sikker tilgang til alt innholdet deres.* Hentet 6. mai 2022 fra https://www.google.com/drive/

Henry, S. L. (2022, 18 mars). *WCAG 2 Overview.* W3. https://www.w3.org/WAI/standards-guidelines/wcag/

Hoffer J., Ramesh V, & Topi K (2019) *Modern Database Management* (12. utg.) Pearson

Indeed Editorial Team. (2021, 2. november) *A Guide to Scrum Teams*. https://www.indeed.com/career-advice/career-development/scrum-team

Jones, M. (2021, 8. juni). *The Repository-Service Pattern with DI and ASP.NET 5.0.* Exception Not Found. https://exceptionnotfound.net/the-repository-service-pattern-with-dependency-injection-and-asp-net-core/

Karlsen, T. (2005). *Kommunikasjon: målstyrt samarbeid og informasjon* (3. utg.). Gyldendal undervisning.

Kelhini, F. (2022, 17. januar). *Axios vs. fetch(): Which is best for making HTTP requests?*. LogRocket. https://blog.logrocket.com/axios-vs-fetch-best-http-requests/

Kumar VM (2018, 29. oktober) *Getting Started With Unit Testing Using C# And xUnit.* C# corner. https://www.c-sharpcorner.com/article/getting-started-with-unit-testing-using-c-sharp-and-xunit/

Less. (u.å.). *It's CSS, with just a little more.* Hentet 14. mai 2022 fra https://lesscss.org/

Liang, M (2021, 11. mars) *Understanding Object-Relational Mapping: Pros, Cons, and Types.* Altexsoft. https://www.altexsoft.com/blog/object-relational-mapping/

Microsoft (2022, 11. mars b) *What is .NET? Introduction and overview.* https://docs.microsoft.com/nb-no/dotnet/core/introduction

Microsoft (2022, 4. februar a) What is Azure Pipelines? https://docs.microsoft.com/en-us/azure/devops/pipelines/get-started/what-is-azure-pipelines?view=azure-devops

Microsoft (u.å. a). *Hva er DevOps?* Hentet 6. mai 2022 fra https://azure.microsoft.com/nb-no/overview/what-is-devops/#devops-overview

Microsoft (u.å. b). *Azure DevOps.* Hentet 6. mai 2022 fra https://azure.microsoft.com/en-us/services/devops/#overview

Microsoft (2022, 18. mars c) *Introduction* https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/language-specification/introduction

Microsoft (2022, 2. april d) *Manage your Scrum process template artifacts* https://docs.microsoft.com/en-us/azure/devops/boards/work-items/guidance/scrum-process?view=azure-devops

Microsoft DevLabs. (2022, 15. mars) *Estimate.* Visual Studio Marketplace https://marketplace.visualstudio.com/items?itemName=ms-devlabs.estimate

MUI. (u.å.). *Installation*. Material UI. Hentet 2. mai 2022 fra https://mui.com/material-ui/getting-started/installation/

Norges Domstoler (u.å. a). *Hva er en meddommer?* Hentet 20. april 2022 fra https://www.domstol.no/no/meddommer/hva-er-en-meddommer/

Norges Domstoler (u.å. b). *Valg av meddommer for kommuner og fylkeskommuner.* Hentet 6. mai 2022 fra https://www.domstol.no/no/meddommer/valg-av-meddommere-for-kommuner-og-fylkeskommuner/

ParTech. (2020, 2. september) *What are the advantages of the entity framework?* https://www.partech.nl/nl/publicaties/2020/11/introduction-to-entity-framework#

Prettier. (u.å.). *What is Prettier?* Hentet 4. mai 2022 fra https://prettier.io/

Postman. (u.å.). *What is Postman?* Hentet 11. mai 2022 fra https://www.postman.com/

React-i18next. (u.å.). *Introduction.* Hentet 6. mai 2022 fra https://react.i18next.com/#what-is-react-i18next

ReactJS (u.å.). *Components and Props.* Hentet 14. mai 2022 fra https://reactjs.org/docs/components-and-props.html

Rehkopf, M. (u.å.). *User stories with examples and a template* Hentet 02. mai 2022 fra https://www.atlassian.com/agile/project-management/user-stories

Sass. (u.å.). *Sass Basics.* Hentet 2. mai 2022 fra https://sass-lang.com/guide

Scrum. (u.å. e). *What is a Daily Scrum?* Hentet 28. april 2022 fra https://www.scrum.org/resources/what-is-a-daily-scrum/

Scrum. (u.å. b). *What is a Product Backlog?* Hentet 28. april 2022 fra https://www.scrum.org/resources/what-is-a-product-backlog/

Scrum. (u.å. c). *What is a Sprint Backlog?* Hentet 28. april 2022 fra https://www.scrum.org/resources/what-is-a-sprint-backlog/

Scrum. (u.å. d). *What is Sprint Planning?* Hentet 28. april 2022 fra https://www.scrum.org/resources/what-is-sprint-planning/

Scrum. (u.å. g). *What is a Sprint Retrospective?* Hentet 28. april 2022 fra https://www.scrum.org/resources/what-is-a-sprint-retrospective/

Scrum. (u.å. f). *What is a Sprint Review?* Hentet 28. april 2022 fra https://www.scrum.org/resources/what-is-a-sprint-review/

Scrum. (u.å. a). *What is Scrum?* Hentet 14. mai 2022 fra https://www.scrum.org/resources/what-is-scrum

Sikri (u.å. b). *Elements Lekdommer: En landsdekkende webbasert løsning for kommuner.* Hentet 9. mai 2022 fra https://www.sikri.no/lekdommer

Sikri. (u.å. a). *Om Sikri AS: Sikrer fremtiden og bevarer fortiden.* Hentet 20. februar 2022 fra https://www.sikri.no/om-oss

Simek, P. (2021, 6. juli). *Pair Programming.* Developer Experience. https://developerexperience.io/practices/pair-programming

TypeScript. (u.å. b). *TypeScript for JavaScript Programmers.* Hentet 02. mai 2022 fra https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html

TypeScript (u.å. a). *TypeScript*. Hentet 14. mai 2022 fra https://www.typescriptlang.org/

UiA. (2009, 14. september) *Hvor mye skal studentene jobbe?* Universitetet i Agder https://www.uia.no/en/news/001-nyhetsarkiv-2003-2014/hvor-mye-skal-studentene-jobbe

Visual Studio Code (u.å. b). *Collaborate with Live Share.* Henter 4. mai 2022 fra https://code.visualstudio.com/learn/collaboration/live-share

Visual Studio Code (u.å. a). *Why did we build Visual Studio Code?* Hentet 15. Mai 2022 fra https://code.visualstudio.com/docs/editor/whyvscode

9.0 Vedlegg

Vedlegg 1 - Gruppekontrakt

Gruppekontrakt

Navn	Telefon	E-post
Gauslaa, Martin Elias	992 51 247	martineg@uia.no
Lislevand, Maren Ryder	974 73 518	marenrl@uia.no
Martinsen, Andreas H.	910 08 947	andrhm16@uia.no
Monge, Sondre T.	417 62 051	sondrm18@uia.no
Sundbø, Simen	994 20 123	simens@uia.no

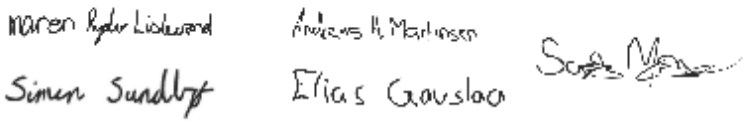
Generelt:

- Fravær og oppgaver som ikke er utført loggføres sprintvis. Dette gjøres for å sikre oss at alle får en karakter de fortjener.
- Vi skal logge oppgaver vi har utført.
- Scrummaster: Simen, først og fremst til mars. Vara scrummaster: Sondre.
- Grupperomansvarlig: Maren. Vara grupperomansvarlig: Andreas.
- Alle studenter møter forberedt til ny arbeidsdag.
- Samlingene avholdes primært mellom 8 og 17, mandag til fredag.
- Vi gjennomfører daily standup hver dag.
- Vi skal bruke Canvas, Azure DevOps og Discord/teams aktivt og det skal sjekkes daglig.
- Vi har et bevisst forhold til at jobb og andre aktiviteter ikke kommer foran studiet.
- Ambisjonsnivå: A

---

1. Møte presis til avtalt tidspunkt. Respekt for andres tid.
2. Forfall eller forsinkelser meldes på telefon eller melding til gruppen.
3. Vi lytter til hverandre, det finnes ingen dumme spørsmål.
4. Vi bruker ikke gruppetiden på annet sosialt snakk.
5. Vi deler eventuelle kostnader.
6. Pause i 10 minutter hver time (Maren og Elias)
7. Kjør lunsjpause på minst 20min.
8. Plan og tidspunkt for neste samling gjennomgås på slutten av arbeidsdagen.
9. Taushetsløfte innad i gruppa, dette gjelder både arbeidsmessig og privat informasjon som måtte komme frem under samarbeidet i gruppa.
10. Vi respekterer hverandre og tar vare på hverandre.
11. Alle må skrive under gruppekontakt.

Sign:



Vedlegg 2 – Utdrag fra Loggbok

Ukedag, dato. måned, klokkeslett
Hovedfokus:
Personer:
Spesielle forekomster:
Nye temaer:

Uke 2

Fredag, 7. Januar, 9:15-15:00
Hovedfokus: pre-sprint
Gruppekontrakt, Risikovurdering, Forelesning, Timeplan, Loggbok

Uke 3

Mandag, 10. Januar, 09:15-15:15
Hovedfokus: pre-sprint
Kontrakt med sikri, Milepælsplan, Teknisk kravspesifikasjon, Brukerhistorie, tidsestimering.
Spesielle forekomster:
Vi tok et valg om å bruke format 1 når det kommer til brukerhistorier.
Fra kundene?
hadde dette fungert?
hva il dere ha?

Tirsdag, 11. Januar, 09:15-15:00
Hovedfokus: pre-sprint
- Gjorde ferdig brukerhistorier og systemkrav
- Tidsplan
- Sketch
- Estimere design milepæler for prosjektet (Sprint 1)

Spesielle forekomster:
Vi tok valg om å jobbe med is-305 på onsdager etter forelesningene i faget.

Onsdag, 12. Januar, 13-15
Hovedfokus: pre-sprint
Personer: Asbjørn og Marius.
- Møte med sikri
    - Risikovurdering
    - Brukerhistorie
    - Systemkrav
    - Sketch
    - Milepælsplan
    - DevOps

Spesielle forekomster:
Planla faste møter med Sikri tirsdager klokka 13

Vedlegg 3 – Risikoanalyse

Konsekvens	Katastrofalt	5	5	10	15	20	25
	Mer	4	4	8	12	16	20
	Moderat	3	3	6	9	12	15
	Mindre	2	2	4	6	8	10
	Ubetydelig	1	1	2	3	4	5
			1	2	3	4	5
			Sjeldent	Usannsynlig	Mulig	Sannsynlig	Høyst sannsynlig
	Sannsynlighet						

Nr.	Risiko	Sann-synlighet 1-5	Konse-kvens 1-5	Risiko-nivå	Risikorespons	Kommentar
1	Mild sykdom	5	1	5	Akseptere	Hjemmekontor
2	Langvarig sykefravær	2	4	8	Akseptere	Fordele personens oppgaver på de resterende gruppemedlemmene
3	Korona-karantene	4	1	4	Begrense	Følge smittevernregler. Hjemmekontor ved karantene
4	Kommunikasjons svikt med sikri	2	4	8	Unngå	Små kommunikasjonsbrister vil mest sannsynlig oppstå, men i disse tilfellene vil ikke konsekvensene være like store.
5	Kommunikasjons svikt internt i gruppen	1	2	2	Unngå	Skal møtes hver dag
6	Feilprioritering og unødig tidsbruk	2	4	8	Begrense	MoSCoW og tett oppfølging. Sprint planning.
7	Feil-scoping	2	3	6	Begrense	En dynamisk prosess
8	Konflikter innad i gruppen som hindrer fremdrift i arbeidet	1	4	4	Unngå	Åpen og ærlig samtale. Være profesjonelle og løse konflikter tidlig. Ta i bruk sprint retrospective for å se på hva som har gått bra/dårlig. Daily standup for å forhindre konflikter før de oppstår.
9	Unnasluntring/dårlig motivasjon	1	2	2	Unngå	Holde hverandre motiverte. Slå ned på unnasluntring

10	Møte opp for sent	4	1	4	Akseptere	Dette kommer til å skje enkelte ganger, men hvis det blir gjentatt for ofte kan det vise til mangel på disiplin
11	Manglende kunnskap og ferdigheter	2	4	8	Begrense	Bruke veilederne aktivt for å få hjelp der vi står fast.
12	Ulike kodestandarder	2	2	4	Unngå	Bli enige om kodestandard før vi setter i gang
13	Lav kvalitet på kode	4	4	16	Begrense	Lav kvalitet på kode vil oppstå mest sannsynlig tidlig i prosjektet, men vil løses og være en del av læringsprosessen.
14	Lav kvalitet på testing	4	3	12	Begrense	Gruppen innehar begrenset erfaring med testing fra før av men har ambisjoner om å mestre dette.
15	Utstyr som svikter	2	3	6	Akseptere	Utenfor vår kontroll. Dette inkluderer internett og strøm.
16	Feil bruk av programvare	2	2	4	Unngå	Opparbeid kunnskap før bruk
17	Dårlig struktur i arbeid	1	4	4	Unngå	Følge scrum oppsett. Grundig planlegging av sprinter og oppfølging via daily standups. Samt nøye oppfølging av Sikri
18	Tap av data	2	4	8	Unngå	God struktur og gjennomføre jevnlige backups. Bruk av git og skylagring kan være med på å forhindre at problemer oppstår. Konsekvensnivå kan variere
19	Lav sikkerhets-standard	2	4	8	Unngå	Verne sensitive opplysninger i prosjektet
20	Brudd på taushetsplikt	1	5	5	Unngå	Ikke dele sensitiv informasjon med utenforstående
21	Overkomplisert brukergrensesnitt	2	4	8	Unngå	Konsekvens vil variere avhengig av hvor komplisert brukergrensesnittet er. I verste fall leverer vi et produkt som kunden ikke vil bruke.
22	Svak dokumentasjon	2	3	6	Begrense	Svak dokumentasjon vil skade gruppens mulighet til å lære fra prosessen.
23	Manglende kunnskap om domenet	3	3	9	Begrense	Det er viktig for funksjonaliteten at aspektene ved lekdommer ikke misforstås.

24	Sitter fast på en oppgave	5	2	10	Akseptere	Kommer til å forekomme, men lag rutine for lav terskel om å spørre om hjelp
25	Mislykket integrasjon opp mot folkeregisteret	2	4	8	Akseptere	Denne funksjonaliteten er en "must have" innenfor brukerhistorier.

Vedlegg 4 – Kodestandard

Generelt

- Språk skal være engelsk (unntak der engelsk oversettelse kan være forvirrende)
- Navn på mapper, filer, klasser, metoder og funksjoner, variabler osv. skal være beskrivende slik at koden er enkel å forstå for alle
- Kommenter kode!

C#

Navngiving

PascalCase: NewObject;
camelCase: newObject;

Klasse navn(PascalCase) = ExampleClass

Lokal variabel(camelCase) = localVariable

Privat eller intern for klassen(_camelCase) = _privateVariable

Kommentarer på egen linje

Stor forbokstav, punktum til slutt

Ett mellomrom mellom // og start av tekst

En statement per linje

En deklarasjon per linje

Minst en linje mellomrom mellom method definitions og property definitions.

TypeScript

Navngiving

- Mapper = camelCase
- Filer = PascalCase

- Klasser = PascalCase
- Variabler = camelCase
- Funksjoner og metoder = camelCase
- Props = PascalCase

- CSS-filer skal ha samme navn som tilhørende tsx-fil. Avluttes med Style

Ryddig kode

- Bruk ESLint og Prettier aktivt for å "rydde" opp i koden
- Unngå inline styling. Overlat dette til CSS-filer

- Én komponent skal ha ansvar for én oppgave

Vedlegg 5 - Git-guide

Git-guide

Innhold

1. Klone repository til din lokale maskin

I terminalen:

1. Naviger deg til stedet der du vil lagre repoet på din maskin
2. Kopier URLen til repoet (denne finner du i repoet på Azure Repos) og utfør følgende kommando:

```
git clone [URL]
```

3. Du har nå klonet repoet!

2. Opprette ny branch og samtidig bytte til den nye branchen

Når du oppretter en ny branch se til at du har siste versjon av main-branchen. Har du ikke siste versjon, utfør følgende kommando i din lokale main-branch:

```
git pull
```

Du har nå siste versjon og kan opprette en ny branch. Ved navngiving av ny branch brukes følgende format:

"f_[initialer]_[PBInr.]_[PBInavn]"

eksempel:

"f_AB_123456_add_save_button_in_application_form"

Vedlegg 6 - Sprint Planning

Sprint planning 03. Februar

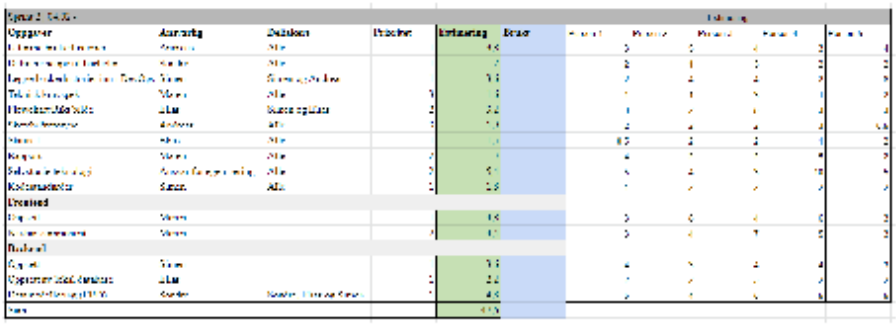
Til stede: Simen Sundbø, Elias Gauslaa, Sondre Tråholt Monge, Maren Ryder Lislevand, Andreas Hovland Martinsen.

Referent: Sondre Tråholt Monge

I. Saker

A. Sprint planning

Som en del av SCRUM-prosessen planlegges andre sprint med utgangspunkt i backlog.



Oppgaver	Ansvarlig	Deltakere	Prioritet	Estimering	Status
Done					
Backlog					

Antall timer planlagt per person: 47,6 pr pers, pr uke

B. Sprint Goal

Målet for Sprint 2 er å bli ferdig med planleggings- og designfasen, samt starte med oppsett og fordeling av oppgaver i starten av systemutviklingen.

II. Møte hevet

Sondre Tråholt Monge hever møtet kl 14:20

Vedlegg 7 - Sprint Retrospect og Review

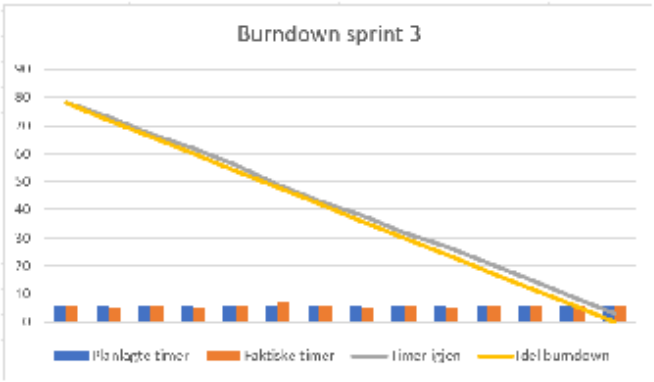
Sprint 3 retrospekt

Torsdag 3. Mars

Hva har vi gjort?

Sprint 3 har i all hovedsak gått til koding. Dette innebærer også å lære seg og sette seg inn i React og .Net. Board i Azure DevOps har blitt brukt til å holde oversikt over oppgavene(PBIer). Det har blitt lagt til underoppgaver underveis som PBIene blir dratt inn under In progress. På frontend har det blitt arbeidet med Navigasjonsbar, legge til personer, list ut og filtreringskomponenter. Backend har jobbet med å kunne legge inn personer. Få en full liste over personer og hente enkeltpersoner etter id. Oppgaver som også ble arbeidet med er kvotehåndtering og meddommer håndtering.

Sprint 3 | 18.02 - 03.03					
Oppgaver	**Ansvarlig**	**Deltakere**	**Prioritet**	**Estimering**	**Brukt**
Filtrere brukerhistorier	Maren	Maren	2	1	1
Systemkrav	Elias	Maren	2	1	1
Styringsgruppemøte	Simen	Alle	1	4	4
Burndown chart	Simen	Simen	2	2	1
Status 2	Maren	Alle	2	2	0
Frontend					
Installasjoner	Sondre, Andreas, Maren	Sondre, Andreas, Maren	1	2	6
Lade klar til vandelsjekk	Sondre	Sondre	1	12	22
Last ut beslått vandel	Sondre	Sondre	1	0	1
Last ut ikke aktuelle	Sondre	Sondre	1	0	1
Lade ut faktisk utvalg	Sondre	Sondre	1	0	1
Lade ut fordac	Sondre	Sondre	1	0	1
Filtreringskomponent	Andreas	Sondre	3	13	6
Legg inn kvote	Andreas	Andreas	2	7	3
Presentere kvote	Maren	Maren	2	8	
Presentere kommune og tellopplysninger	Maren	Maren	3	12	
Navbaruteseende	Maren	Maren	1	16	16
Testuck selvstudie React	Maren	Maren	1	16	10
Legg til personer, skjema	Andreas	Andreas	1	12	16
PBIer	Andreas	Andreas, Maren, Sondre			7
Sum Front End				120	90
Backend					
Legge til en person	Simen	Simen & Elias	1	8	16
Kvotejekk	Simen	Simen & Elias	3	8	
Legge til en person som meddommer	Simen	Simen	1	6	7
Vedlikehold av personopplysninger	Simen & Elias	Simen & Elias	2	6	2
Vandelsjekk-import-eksport	Simen & Elias	Simen & Elias	3	8	
Legg inn kvote	Elias	Elias	1	7	7
Selvstudie backend	Simen & Elias	Simen & Elias	1	8	10
Sum Back End				51	42
Sum total				272	209



Hva har gått bra og hva vil vi fortsette med?

Vi har opprettholdt de gode rutinene for arbeid som vi satt i tidligere sprinter. Dette har ført til at vi har jobbet godt og effektivt. Vi har satt i gang med en ny fase av prosjektet og har kommet godt i gang med dette og et godt utgangspunkt for videre utvikling har blitt lagt.
Vi gjennomførte også et gruppestyringsmøte med veiledninger og representant fra Sikri.
Vi nådde også målene som vi satt oss i sprint planningen. Her satt vi som mål å begynne med kode slik at vi ha noe å vise frem.

Hva som kunne ha blitt gjort bedre?
I denne sprinten har det blitt litt krøll med timeestimeringen. Vi har nå gått fra å jobbe sammen med så og si alle oppgaver, til å jobbe mye mer individuelt. Vi må derfor begynne å skrive individuelle timer for oppgavene slik det blir riktig format for alle.

I forrige sprint retrospekt sa vi at vi ble enige om at en oppgave ikke skulle vare mer enn 8 timer. Dette klarte vi ikke å overholde og vi estimerte oppgaver som var godt over 8 timer. Mye av grunnen til dette var at vi så for oss at det ville gå mye tid til å tilegne oss kunnskap om det vi skulle utvikle. I ettertid ser vi at vi burde vært flinkere til å skille mellom selvstudie og koding, selv om dette kan være vanskelig å skille noen ganger.

Vi ga oss selv litt for mange oppgaver og undervurderte nok hvor mye tid som går til å sette seg inn i ting, samt prøve og feile. Dette er tross alt nytt for oss alle, med ny teknologi og nye måter å jobbe på.

Tanken rundt prioritering i sprint planning var i hvilken rekkefølge oppgavene skulle bli arbeidet med. Det vi har funnet ut etter denne sprinten er at alle PBIene er Must haves og av høyeste prioritet. Det vi heller vil bruke prioriteringene til å vise oppgaveflyten.

Noe vi må ta med i betraktning når vi estimerer timer per sprint er at sprint retrospektiv og planning også tar tid og må inn i estimatet.

Vedlegg 8 – Systemkrav

Link til tilhørende brukerhistorier:
https://docs.google.com/spreadsheets/d/1PIYGCjxqlbvsdf95NOE8u3fsNOQt--FiNbWa0I4u020/edit#gid=0

Nr.	Systemkrav	Tilhørende brukerhistorie
1	Meddommerportal	1, 2, 3, 4
2	Fritaksbehandling	5, 6
3	Sikker innlogging	7,8
4	Oversiktlig grensesnitt	9, 10, 11
5	Innlegging og fremvisning av kvoter	12
6	Legge inn spesifikke personer i systemet	13, 15, 16
7	Vandelsvurdering	17, 18
8	Kravsjekk	19, 26
9	Valg av meddommere	20, 21, 22, 23, 25
10	Vedlikeholde personopplysninger.	24
11	Behandling av utvalg	27, 28
12	Sende meldinger	14, 29, 43, 44
13	Oversikt over meddommere	30, 31, 32
14	Statistikk	33
15	Arkiv for ulik dokumentasjon	34, 35, 36
16	Varslingssystem	37
17	Kommune og rettsopplysninger	38, 39, 40
18	Kunne legge til nye kommuner	41
19	Legge til nye brukere	42

Oversikt
Lekdommertjenesten skal fornyes ved å forenkle og effektivisere systemets rekrutteringsprosess av lekdommere. Systemet skal inneholde alle funksjonene til det eksisterende systemet,

Bruker
Brukere av systemet er ansatte i kommunen med ansvar for lekdommertjenesten.

Teknologier
Det vil bli tatt i bruk Azure DevOps for struktur og kvalitetskontroll. React med TypeScript skal bli brukt for utvikling av Frontend. APS.Net og C# skal bli brukt for utvikling av Backend. Git skal bli brukt for versjonskontroll. Docker skal bli brukt for å legge systemet inn i containere.

Applikasjonsspesifikasjon for systemet
Systemet skal optimalisere og effektivisere aspekter ved den nåværende tjenesten. Dette gjennomføres blant annet gjennom systemets vask av utvalg til potensielle meddommere. Ved siden av dette, skal systemet legge til rette for intern kommunikasjon, strukturert håndtering av fritakssøknader, samt oversikt over utvalgsstatistikk.

Mål
Målet med lekdommertjenesten er å lage et brukervennlig, moderne og effektiv produkt til kommunen. Dette systemet skal også være et nyttig verktøy for kommuner under rekrutteringsprosessen av meddommere. Ved bruk av lekdommertjenesten vil kommuner spare tid og ressurser.

Vedlegg 9 – Brukersamtale

Brukersamtale - Lekdommertjeneste
Si noe om plan for samtalen

Gjennomgang
Har du mulighet/lyst til å vise oss programmet sånn det er i dag?

Kan du vise hovedprosessen rundt rekruttering av lekdommere?

Kan du vise prosessen rundt det å legge meddommere inn i utvalg i en periode?

Kan du vise hvordan dere går frem for å vise medlemmer(kvote) i et utvalg?

Etter gjennomgang
Hva gjør du mest i systemet?

Hva føler du tar unødvendig mye tid?

Er det noe som irriterer deg ved systemet?

Er det noe du savner med systemet?

Er det noe du gjør utenfor lekdommertjenesten som er direkte tilknyttet rekrutteringsprosessen?

Oppfølging ved ja

Gjennomgang av prototype
Hva ser du her? Hva tror funksjonen på denne siden er?
Rekruttering

Utvalg

Fritak

Arkiv

Admin

Vedlegg 10 - Entity Relationship Diagram