Høgskulen på Vestlandet

BACHELOROPPGAVE

Webportal for organisasjoner

Webportal for organizations

Petter Knudsen

Stefan Go Thuen

Ole Martinus Alstad Rambech

DAT190-1 20V Bacheloroppgave - Data/Informasjonsteknologi

Fakultet for ingeniør- og naturvitskap

Veileder Tosin Daniel Oyetoyan

Innleveringsdato 02.06.2020

Høgskulen på Vestlandet

Fakultet for ingeniør- og naturvitskap

Institutt for datateknologi, elektroteknologi og realfag

TITTELSIDE FOR HOVEDPROSJEKT

Rapportens tittel: Webportal for organisasjoner	*Dato:* *15. jan. 2020*
Forfatter(e): Petter Knudsen, Stefan Go Thuen, Ole Martinus Alstad Rambech	*Antall sider u/vedlegg: 55* *Antall sider vedlegg: 0*
Studieretning: *DATA og IT*	*Antall disketter/CD-er:* 0
Kontaktperson ved studieretning: Tosin Daniel Oyetoyan	*Gradering:* Ingen
Merknader: Ingen	

Oppdragsgiver: Styreportalen AS	*Oppdragsgivers referanse:* Ingen
Oppdragsgivers kontaktperson: Stian Sømoen	*Telefon:* Ingen

Sammendrag:

Rapporten beskriver prosessen av å utvikle en lukket webportal for Styreportalen AS sin tjeneste Medlemssiden.no. Medlemssiden.no er en lukket side for orginasjoner som vil kunne legge ut hendelser og prosjekter for deres medlemmer mens de gir ytterlige informasjon. Webportalen vil erstatte Styreportalen AS sin allerede eksisterende løsning, som til dagens standard ikke er god nok.

I denne rapporten vil det bli gitt en beskrivelse av prosessen, og evaluering av hvorvidt løsningen løser problemstillingen.

Stikkord:

Firebase	React	Redux

Høgskulen på Vestlandet, Fakultet for ingeniør- og natuvitskap
Postadresse: Postboks 7030, 5020 BERGEN Besøksadresse: Inndalsveien 28, Bergen
Tlf. 55 58 75 00 Fax 55 58 77 90 E-post: post@hvl.no Hjemmeside: http://www.hvl.no

FORORD

Denne rapporten er dokumentasjonen for bachelorprosjektet Medlemssiden.no - Styreportalen AS. Rapporten beskriver gruppens tankegang og prosess gjennom utviklingen av den nye medlemssiden for Styreportalen AS. Prosjektet er gjennomført av Petter Knudsen, Stefan Go Thuen og Ole Martinus Rambech.

Det gis stor takk til Styreportalen AS som har latt oss være en del av dette prosjektet, og for all hjelpen og veiledningen de har gitt. Spesielt ønsker vi å takke Stian Sømoen som har vært den eksterne veilederen for dette prosjektet og all den hjelpen han har gitt gjennom hele prosjektet.

Vi gir også stor takk til den interne veilederen Tosin Daniel Oyetoyan ved Høgskolen på Vestlandet som har gitt gode råd og veiledning gjennom hele prosjektperioden.

TABLE OF CONTENT

1 INNLEDNING

I dette kapitlet introduseres bakgrunnen for prosjektet. I de følgende avsnittene vil målsetting og motivasjon, kontekst, avgrensninger og ressurser avklares.

1.1 Motivasjon og mål

Gruppen i dette prosjektet består av Petter Knudsen, Stefan Go Thuen og Ole Martinus Rambech. Da gruppen skulle velge prosjekt var det ønskelig med web-teknologi, helst både front-end og back-end. Oppgaven Medlemssiden.no spesifiserte teknologiene React og Firebase, og gruppen var i enighet om at oppgaven var interessant, og valgte den.

Figur 1: Styreportalen AS logo

Styreportalen AS utvikler online verktøy for drifting av organisasjoner. De tilbyr Styreportalen.no, en hjemmeside, Notearkiv.no, Medlemssiden.no, Medlemsapp og Epost. Selskapet ble stiftet i 2016, og holder til på Lundhaugvegen 14 på Nesttun. Bedriften har i dag 6 ansatte.

1.1.1 Problemstilling

Frivillige organisasjoner tar i dag bruk digitale medlemssystemer og medlemsregistre for driften av organisasjon. Disse verktøyene lar styret og medlemmene enkelt få oversikt over hva organisasjonen driver med til enhver tid. Styreportalen AS har i dag en løsning på disse verktøyene som er implementert med gammel og utdatert teknologi. De ser derfor behov for å utvikle en ny og moderne løsning for å holde seg konkurransedyktige i dagens marked.

1.1.2 Mål

Prosjektets mål er inndelt i to deler, hvor del en innebærer utvikling av hovedkomponentene i applikasjonen, mens del to innebærer videreutvikling

av komponentene med nye funksjonaliteter, som er mindre viktig for sluttproduktet.

Hovedkomponentene består av et innloggingssystem, en startside som inneholder en informasjonsstrøm, samt lister over organisasjoner, aktiviteter og prosjekt. Hver organisasjon skal ha sin egen side med liste over medlemmer, styret og verv, dokument, info om organisasjonen, og info organisasjonen har over deg som medlem. Hvert prosjekt skal ha sin egen side med detaljer om prosjektet. Applikasjonen skal også inneholde en kalender med oversikt over alle aktiviteter.

Videreutviklingen består av å legge til støtte for egendefinerte menyer, visning av dokumenter, og utdeling av oppgaver, innad i hver organisasjon

1.2 Kontekst

1.2.1 Prosjektet

Styreportalen AS tilbyr en rekke verktøy for organisasjoner, og ønsker å forbedre flere av produktene sine. Deres eksisterende løsning for medlemssiden.no er ikke god nok for dagens web-standard, og dermed ønsker Styreportalen å utvikle en bedre løsning. Det er viktig for Styreportalen at produktet lever opp til kundens forventninger, slik at konkurrentene ikke tar over markedsandelen til Styreportalen. Medlemssiden.no er skrevet i PHP/JavaScript/MySQL basert på Joomla CMS, og dette ønsker Styreportalen å endre til en mer moderne løsning som tar i bruk React, Node.js og Firebase, for gjøre driften billigere og utvikling av nye funksjoner lettere i fremtiden.

1.2.2 Gruppen som del av prosjektet

Styreportalen AS holder i dag på med utvikling av en ny og bedre styreportal, en ny app. Gruppen skal stå for utviklingen av Medlemssiden.no. Siden prosjektet er en del av et større system, er det viktig at det er en rød tråd mellom de forskjellige komponentene av systemet. Gruppen er derfor i konstant kontakt med Styreportalen AS, for å vise frem arbeidet og få tilbakemeldinger og nye arbeidsoppgaver.

1.3 Avgrensninger

Det er naturlig at gruppen vil ha noen avgrensninger i forhold til prosjektet som følge av begrenset tidsramme og manglende kunnskap om teknologien som blir benyttet. Avgrensningene som er mest aktuell er manglende kunnskap om teknologi. I tillegg befinner gruppen seg i en uvanlig situasjon, da Covid-19 pandemien herjer for fullt, noe som skaper en ny avgrensning - mangel på møter i person og mangel på felles arbeidsplass.

1.3.1 Kunnskap om teknologien

Prosjektets tre medlemmer starter arbeidet på prosjektet med lav kunnskap om teknologien applikasjonen skal utvikles med, og må dermed lære samtidig som arbeidet pågår. Styreportalen AS er tilgjengelig for hjelp om gruppen skal trenge det.

1.3.2 Covid-19

Covid-19 har gjort at møter mellom mennesker helst skal unngås, noe som har ført til at gruppen har jobbet fra sitt eget hjem. Dette skaper en unik utfordring, da gruppen finner det vanskeligere å arbeide hjemmefra, enn det er å jobbe sammen på HVL. Blant annet fordi det er vanskeligere å hjelpe hverandre med kode som ikke fungerer og vanskeligere å illustrere hvor problem ligger/hvordan kode fungerer. Covid situasjonen har, med andre ord, forverret avgrensningen med mangel på kunnskap.

1.4 Ressurser

I prosjektet tas det i bruk både menneskelige og tekniske ressurser. Med menneskelige ressurser menes det mennesker som kan bidra med råd, veiledning og hjelp, og Med tekniske ressurser menes det verktøy som tas i bruk for å jobbe med prosjektet.

1.4.1 Menneskelige ressurser

Gruppen vil ha tilgang til en rekke menneskelige ressurser. Høyskolen stiller med intern veileder som er tilgjengelig for gruppen. Styreportalen AS stiller med en Prosjekteier, samt en React utvikler, som vil være tilgjengelige for gruppen. I tillegg vil medlemmene i gruppen være tilgjengelige for hverandre.

1.4.2 Tekniske ressurser

Prosjektet tar i bruk Git og GitHub som versjonskontroll av produktet. Prosjektets kommunikasjon gjennomføres over Microsoft Teams, Discord, Google Hangouts, Trello og Whereby. Utvikling av produktet utføres med Visual Studio Code som editor for både React og Node, og Firebase benyttes som vert for middleware og applikasjon.

1.5 Oppbygging av rapporten

Denne rapporten inneholder 10 kapitler som som omtaler prosessen ved å utvikle Medlemssiden.

Kapittel 1 gir et oversiktsbilde over prosjekt, problemstilling, hvem Styreportalen AS er, hva målet for prosjektet er, hvilke avgrensninger prosjektet står overfor, og hvilke ressurser gruppen tar i bruk og har tilgjengelig for utviklingsprosessen.

Kapittel 2 beskriver nødvendigheten for prosjektet og gir en dypere forklaring på løsningen sitt innhold. I tillegg omtaler kapittelet oppdragsgiver sine ønsker om prosjektet og dets initielle krav.

Kapittel 3 beskriver hvilke mulige løsninger prosjektet kan ta i bruk med begrunnelse for valgt løsning samt hvilke risikoer den løsningen tilbringer.

Kapittel 4 forklarer i detaljer hvordan løsningen fungerer og designet for løsningen.

Kapittel 5 gir en beskrivelse av hvilke metoder som har blitt benyttet for å evaluere produktet. Evalueringsmetodene som er blitt benyttet vil bli begrunnet og resultatet av disse.

Kapittel 6 forklarer hvordan valgt løsning og teknologi har påvirket resultatet. I tillegg blir det diskutert hvordan prossesen gjennom hele prosjektperioden.

Kapittel 7 beskriver om målene for prosjektet har blitt nådd og hvordan løsningen kan bli viderarbeidet.

Kapittel 8 er en liste over referanser brukt i rapporten.

Kapittel 9 inneholder en risiko liste, gantt skjema og en liste om vedlegg.

2 PROSJEKTBESKRIVELSE

I dette kapitlet beskrives prosjektets bakgrunn, følgende avsnitt vil beskrive produkteier, praktisk bakgrunn, tidligere arbeid, initielle krav, initiell løsningsidé og litteratur om problemstillingen.

2.1 Produkteier

Styreportalen AS ønsker alle rettigheter til den endelige produktløsningen. Løsningen vil integreres som en del av deres løsning på drifting av organisasjoner. Gruppen vil ikke holde noe eierskap til produktet, men har lov å omtale og vise frem applikasjonen i oppgaven. Styreportalen ønsker derimot at gruppen ikke viser frem strukturen i databasen. Styreportalen har selv opprettet kravspesifikasjonen for produktet, og dersom gruppen ønsker å vike fra kravspesifikasjonen må kontaktperson i Styreportalen konsulteres.

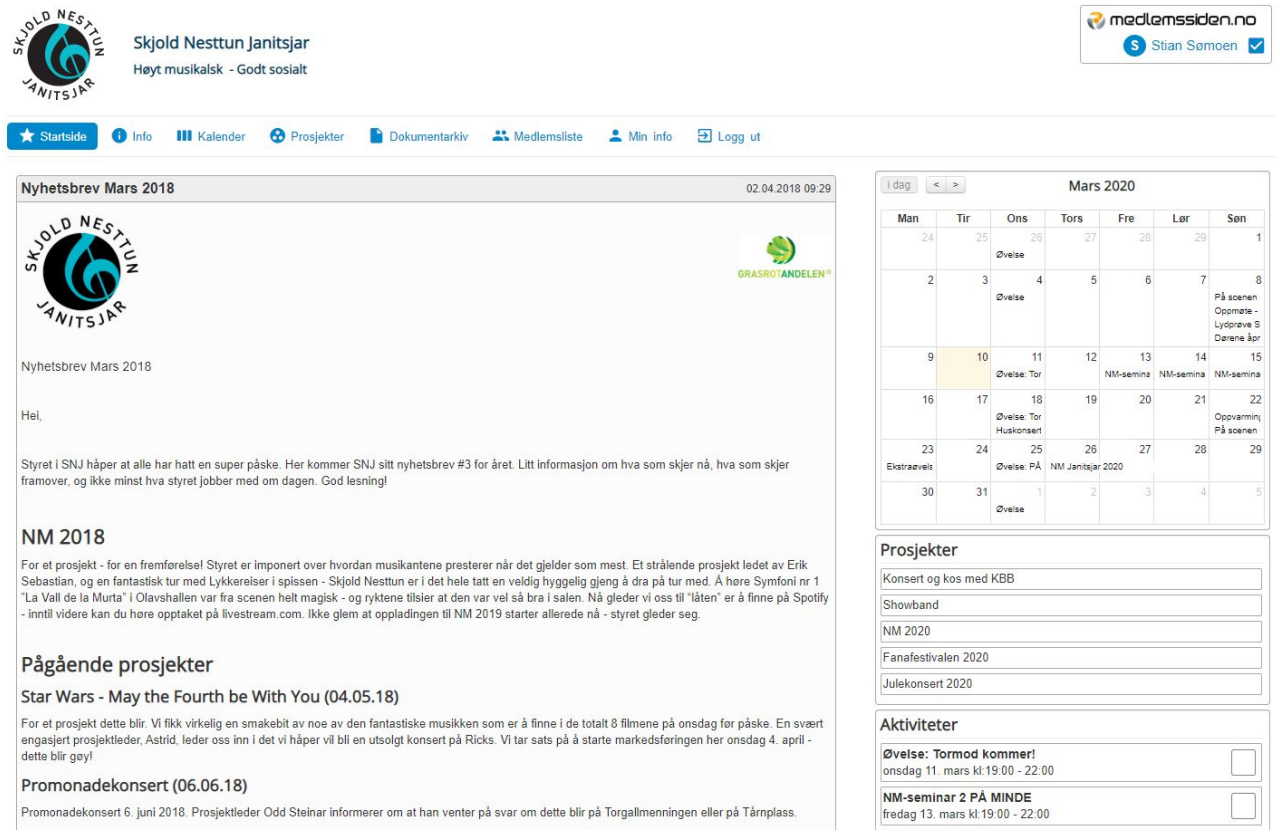
2.2 Praktisk bakgrunn

Prosjektet er viktig for Styreportalen på flere nivå. Først og fremst for å forbedre kildekoden med nytt språk. Implementering av nytt språk og ny teknologi som er bedre egnet til problemstillingen vil gjøre det lettere å vedlikeholde applikasjonen, som for eksempel ved å implementere ny funksjonalitet som gjør brukeropplevelsen i applikasjonen bedre. Prosjektet vil også føre til et bedre produkt for kunden enn det de benytter i dag. Prosjektet bygger på samme grunntanke som i eksisterende løsning, derimot vil resultatet av prosjektet være en applikasjon med mer og bedre funksjonalitet, samt et produkt som oppnår bedre brukeropplevelse. Et produkt som er lettere å holde ved like, samt gir kunden en bedre opplevelse, vil også gi Styreportalen muligheten til å være mer konkurransedyktig, som vil hjelpe selskapet å holde eller øke sin markedsandel.

2.2.1 Tidligere arbeid

Prosjektet bygger på en tidligere løsning av medlemssiden.no,som er en egen lukket side der medlemmer og foresatte kan logge seg inn og se informasjon som styret legger ut. Det er også mulig å se kalender, samt å melde fravær. Man kan også redigere sin egen informasjon i medlemsregisteret. Det tidligere

arbeidet er nå utdatert, og det trengs en bedre løsning. På grunn av det blir tatt inn ny teknologi vil ikke prosjektet bygge på kildekoden til den tidligere løsningen, og bygger dermed ikke direkte på tidligere arbeid.



Figur 2: Eksisterende løsning på Medlemssiden.no

Funksjonaliteten til den nye applikasjonen som skal utvikles i løpet av prosjektet er dermed basert på den tidligere løsningen. Styreportalen har lært av hva som fungerte godt, og hva som ikke fungerte så bra, og baserer dermed kravene for prosjektet på tidligere arbeid.

Prosjektet tar derimot i bruk en React template, slik at fokus er mindre på å lage et godt brukergrensesnitt, og mer på å opprette god funksjonalitet. Gruppen kan dermed ta i bruk komponenter fra den eksisterende templaten, endre på innhold for å møte forventet funksjonalitet uten å bruke for mye tid på å fokusere på utseende.

2.2.2 Initielle krav

Valg av teknologi for utvikling av prosjektet var forhåndsbestemt av Styreportalen, og gruppen hadde ikke mulighet til å velge teknologien selv.

Styreportalen AS sin eksisterende løsning er skrevet i PHP/JavaScript/Html/MySQL og er basert på Joomla CMS. Dette betyr hver organisasjon har egen database som har ført til at styreportalen har over 200 databaser. Som konsekvens av teknologivalg er det dermed tungvint å holde kontroll på backend av systemet, samt å holde det vedlike. Styreportalen ønsker dermed å gå over til å ta i bruk Firebase sin Firestore som sin database. Å bruke Firestore som database vil mitigere problemet, da å ta i bruk Firestore åpner muligheten til å kun trenge å opprettholde en database, hvor hver organisasjon har sin egen node i databasen. Dette gjør det lettere å vedlikeholde og ha kontroll over systemet.

Styreportalen ønsker å endre frontend teknologien til React, JavaScript og css. React er et frontend rammeverk som er bygget for å kunne lage avanserte brukergrensesnitt raskt og effektivt. Det er i essensen veldig likt som å opprette brukergrensesnitt med html/javascript/css, men React håndterer dynamisk endring av brukergrensesnittet for utvikleren.

Hovedgrunnen til bytte teknologien er derimot at databasen blir til en real time database. En real time database vil gi applikasjonen muligheten til å kunne hente ut endringer i databasen mens de skjer, i real time. Dette gir muligheten til å utvikle ekstremt dynamiske applikasjoner med chat, notifikasjoner osv.

Den initielle oppgavebeskrivelsen er delt inn i to deler. Første del er å opprette en React applikasjon, hvor det er listet opp en rekke krav. Applikasjonen skal inneholde en forside med en informasjonsstrøm, en kalender med mulighet til å melde fravær, en medlemsliste, en styreoversikt, en kontaktside og en ‘min info’ side. Autorisering av bruker skal skje gjennom innlogging, og innloggingssystemet skal være Firebase Authentication.

Andre del lister opp tre nye funksjoner som siden skal inneholde. Det skal være støtte for egendefinerte menyer, siden skal inneholde et dokumentarkiv, samt en oversikt over oppgaver for brukeren som er innlogget.

#	Funksjonelt krav
1	Informasjonsstrøm(feed)

2	Kalender
3	Medlemsliste og styreliste
4	Kontaktside
5	Min side
6	Brukerautentisering med Firebase authentication

Tabell 1: Funksjonelle krav for del en av initielle krav

#	Funksjonelt krav
1	Egendefinerte menyer
2	Dokumentarkiv
3	Personlige oppgaver

Tabell 2: Funksjonelle krav for del to av initielle krav

I et møte mellom gruppen og oppdragsgiver ble de initielle kravene oppklart. Prosjekteier hadde med detaljerte bilder om hvordan han ønsket at siden skulle være satt opp, hvordan de forskjellige funksjonene i del en og to skulle fungere, og gikk gjennom den tidligere løsningen av medlemssiden.

Prosjekteier gikk raskt gjennom kravene for hjemmesiden til applikasjonen. Hjemmesiden til applikasjonen skal inneholde en informasjonsstrøm. Informasjonsstrømmen skal inneholde artikler alle organisasjonene en innlogget bruker er medlem i, men kun de artiklene brukeren har rettigheter til å se skal vises. Hjemmesiden skal også inneholde kommende prosjekter og aktiviteter for organisasjonene brukeren er medlem i.

Prosjekteier ble det oppklarte at dette ikke var krav, men forslag til hvordan oppgaven skulle løses. Prosjekteier forklarte at det kom til å være muligheter til å være i konstant kontakt dersom det skulle være spørsmål, og at hvis gruppen opprettet nye funksjoner så kunne veilederen gi tilbakemelding.

Prosjekteier kom derimot med noen ikke-funksjonelle krav. Applikasjonen skal være rask - responstiden må være lav. Applikasjonen skal være responsiv

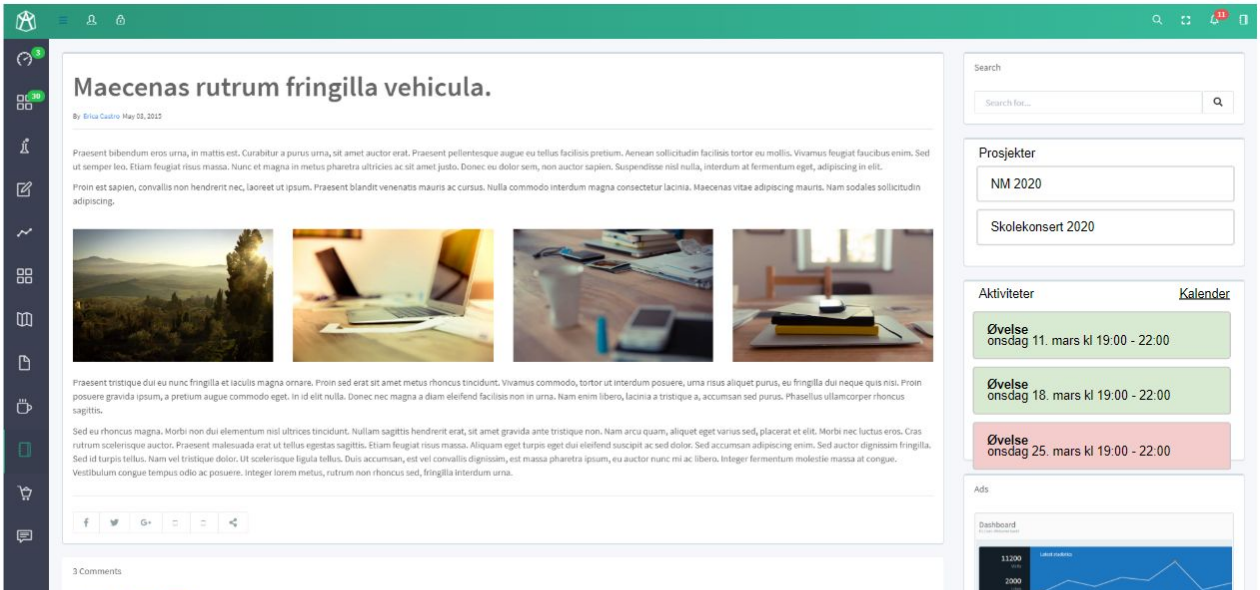
- den må fungere på alle enheter. Applikasjonen bør være effektiv - ikke bruke unødvendige ressurser i databasen.

#	Ikke-funksjonelt krav
1	Lav responstid
2	Høy responsivitet
3	Effektivitet

Tabell 3: Ikke-funksjonelle krav for prosjektet

Gruppen og Prosjekteier kom til enighet om å holde ukentlige møter, hvor gruppen skal gå igjennom ukens arbeid, få tilbakemeldinger, kunne stille spørsmål, gi innspill, samt få innspill, i neste ukes arbeid.

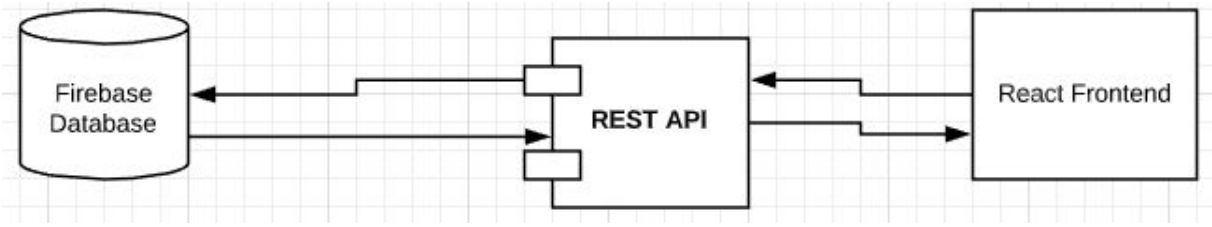
Gruppen fikk levert ut et React template som basis for utviklingen av applikasjonen. Prosjekteier gikk gjennom templaten og forklarte hvilke deler av templaten som var tenkt at gruppen skulle bruke. For første ukes arbeid fikk gruppen som oppgave å bli kjent med templaten, samt opprette et innloggingssystemet for applikasjonen, og hvis det var mer tid til overs, hente inn artikler og vise de frem i nyhetsstrømmen i hjemmesiden. Figur under illustrer hvordan Prosjekteier tenkte at hjemmesiden kunne se ut, basert på komponenten BlogPost i React templaten vi fikk utlevert.



Figur 3: Forlsag til løsning på forside

2.2.3 Initiell løsnings-idé

Etter det første møtet mellom Prosjekteier startet gruppen på planlegging av en løsning for problemstillingen. Fokuset var på å få oversikt over hvordan applikasjonen skulle kommunisere mellom frontend og backend. Tidligere erfaringer med lignende prosjekt førte til at svaret endte på å bruke et rest-api som mellomvare, et kommunikasjonsledd mellom databasen og frontend klienten. Firebase tilbyr en tjeneste kaldt 'functions', hvor gruppen kunne laste opp node.js kode som kjører i skyen ved forespørsler til en spesifikk url.



Figur 4: Kommunikasjon mellom database og frontend

Mellomleddet skulle skrives i node.js, og skulle ha forskjellige ruter som tilbyr forskjellige tjenester. Etter møtet med Prosjekteier var det klart for gruppen at mellomleddet trengte én rute for innlogging, én for registrering, én for å hente artikler for informasjonsstrømmen, én for å hente ut informasjon om brukeren, én for å hente ut informasjon om prosjekter, én for å hente ut informasjon om aktiviteter og én for å hente ut informasjon om organisasjoner.

Etter at en løsning på kommunikasjonen mellom front og backend var funnet, var det over på å planlegge strukturen for innlogging, samt autorisering av brukere. Firebase har to løsninger, en klient-basert løsning, og en token-basert løsning. I klientløsningen inneholder klienten informasjon om brukeren, mens i token-basert løsning sendes en id-token til klienten ved innlogging, og hver gang klienten skal autoriseres av rest-api så må klienten sende id-tokenet med i forespørselen. Gruppen endte opp med å ta i bruk en token-basert løsning, da denne løsningen dekket alle behov.

Løsningen for frontend var en større oppgave. Gruppen måtte sette seg inn i React templaten de fikk utlevert av oppdragsgiver, bli kjent med strukturen,

hvordan de forskjellige komponentene såg ut, og dermed hva som kunne brukes. En komponent i templaten, kaldt blog-post, kunne med enkle endringer brukes som informasjonsstrøm. Templaten inneholdt både komponent for innlogging og registrering som begge kunne brukes. Templaten inneholdt også en kalender, som kunne brukes til å vise kommende aktiviteter, samt en data-grid som kunne brukes til å fremstille en medlemsliste for organisasjoner.

Et problem som ofte oppstår i større applikasjoner skrevet med React er at deling av tilstandsdata gjennom applikasjonen. Denne dataen kan kun gå fra toppnivå og nedover, og det blir rotete hvis dataen trengs i mange forskjellige komponenter i forskjellige ledd. Gruppen fant en løsning på dette ved å implementere redux i applikasjonen. Redux er en form for informasjonsbank som stilles ved siden av applikasjonen, slik at hver komponent i applikasjonen kan hente ut akkurat den informasjonen de trenger uten å måtte arve det fra en forelder komponent.

Større deler av tilstandsinformasjonen i applikasjonen hentes inn fra databasen. For å slippe unødvendige forespørsler til databasen er det viktig å plassere innhentingen av informasjon i strategiske posisjoner. Et eksempel på unødvendig innhenting av informasjon kan være å hente inn data før en bruker har logget inn. Et annet eksempel er å hente inn samme data i forskjellige komponenter. Gruppen fant en mulig løsning på dette ved å se på informasjonen som blir brukt i

flere komponenter, og dermed hente inn informasjonen fra toppnivået i appen.

2.3 Litteratur om problemstilling

I dagens samfunn er medlemssystem hovedsakelig digitalisert (Medtek Norge, uå). Utenom Styreportalen AS sin tidligere løsning finnes det flere andre løsninger på å gjøre foreningers drift lettere. En konkurrent av medlemssiden.no er styreweb.com. De ble lansert i 2006 og har mye av samme funksjonalitet som gruppens løsning(Styreweb, 29.05.2020). De bruker Jquery som frontend og mysql som backend. Styreweb.com inneholder flere funksjoner du ikke ser i medlemssiden.no. Deriblant mulighet til å holde oversikt over utlånt utstyr, støtte for møter, med oppmøteliste, saksliste og møtereferat. Arrangement med støtte for billettsalg og deling av arrangement link som kan benyttes av brukere som ikke tilhører styreweb.

3 DESIGN AV PROSJEKTET

I løpet av dette kapitlet vil designet av prosjektet beskrives. Ulike løsninger er vurdert for å finne det beste designet for applikasjonen. Først beskrives den valgte løsningen, deretter en alternativ løsning. Løsningene diskuteres dermed opp mot hverandre.

3.1 Valgt løsning

Siden gruppen selv ikke har muligheten til å velge rammeverk eller database unnlater vi å diskutere ulike løsninger på dette.

3.1.1 Uviklingsmetode - Scrum

Scrum er et agilt rammeverk for utvikling av informasjonssystemer, men er ofte brukt i annet kunnskapsarbeid. Scrum er designet for grupper på ti eller færre medlemmer, som bryter arbeidet ned i små mål som kan bli fullført innen en bestemt tidsramme. Dette blir kalt sprints, og varer vanligvis i 2 uker. Dette tillater gruppen å utvikle en prototype av en funksjon, presentere den til produkteier, få tilbakemelding, og videreutvikle den etter nye retningslinjer i neste sprint. Scrum bryter opp en gruppe i flere spesifikke roller, produkteier, utviklingslag, og scrum mester.

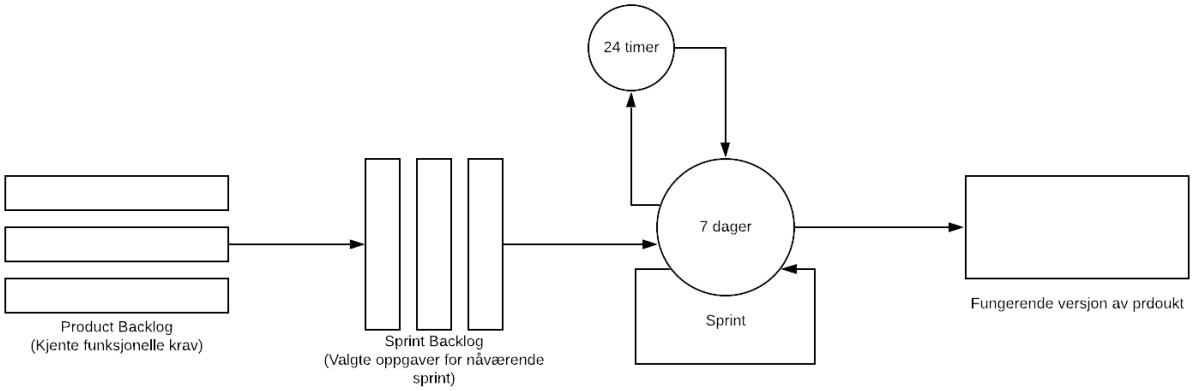
Produkteier representerer produktets eiere, og er stemmer til kunden. Produkteier er ansvarlig for at produktet oppnår det forventede resultatet. Produkteier er dermed ansvarlig for produkt backloggen[1] og for å maksimere verdien til det endelige produktet. En gruppe som følger scrum bør ikke ha mer enn en produkteier, men en produkteier kan være medlem i flere grupper. I dette prosjektet vil Prosjekteier ta på seg rollen som produkteier.

Utviklingslaget tar for seg alle oppgavene som er krevet for å utvikle verdifulle løsninger hver sprint. Utviklingslaget består al mellom 3 til 9 medlemmer, og kan inkludere utviklere, arkitekter, designere osv. Laget er selvorganisert, og skal kun motta arbeid fra produkteier. I dette prosjektet vil gruppen bestående av Petter, Stefan og Ole ta på seg rollen som utviklingslag.

Scrum mester er personen som er ansvarlig for å sikre at utviklingslaget bruker agile verdier og prinsipper, og følger prosessen og praksiser som

[1] Produkt 'backloggen' er en oversikt over arbeid som må gjøres, og består som oftest av brukerhistorier. Et eksempel på en typisk brukerhistorie er 'Som en bruker, forventer jeg at x skjer når jeg gjør y'.

teamet har blitt enige om å bruke. Scrum masteren skal beskytte utviklingslaget fra for mange distraksjoner. I dette prosjektet tok Petter på seg rollen som Scrum mester.



Figur 5: Illustrasjon av produktets Scrum prosess

3.1.2 Kommunikasjon med database - direkte fra klient

En løsning for kommunikasjon mellom databasen, som er informasjonskilden brukt til å bygge opp applikasjonen, og klienten, som er applikasjonen som kjører lokalt hos sluttbrukeren, er å utvikle klienten med muligheten til å selv kommunisere direkte med databasen.

3.1.3 Designmønster - MVC

Model-View-Controller (MVC) er et designmønster for utvikling av programvare som bygger på prinsipper om separation of concerns[2]. Mønsteret deler opp programmet inn i tre deler, model, view og controller.

Modellen ('model') håndterer data, logikk og regler for applikasjonen. Brukergrensesnitt ('view') definerer hvordan data blir vist til brukeren.

[2] Seperation of concerns er prinsippet om å dele programvare inn i spesifikke seksjoner som hver håndterer sin egen oppgave

Kontrollør ('controller') håndterer kommunikasjon fra brukeren og oversetter det til kommandoer for modellen og brukergrensesnittet.

3.1.4 Applikasjonstilstand - Redux

Redux er en tilstands-oppbevarer som lagrer applikasjonstilstanden som objekt tre. Redux er forutsigbar, og oppfører seg konsekvent, og dermed vil forventet resultat alltid være det samme. Tilstanden i Redux lagres som et objekt-tre, og oppbevares i en 'store' som er tilgjengelig i hele applikasjonen. For å endre tilstanden til treet må man utføre en 'action' som har en spesifikk type, og om det trengs, payload, som inneholder ny informasjon.

"Redux helps you write applications that behave consistently, run in different environments (client, server, and native), and are easy to test." (Redux, u.å).

3.2 Alternativ løsning 1

3.2.1 Utviklingsmetode - AUP

Agile Unified Process (AUP) er en forenklet versjon av Rational Unified Process (RUP). AUP forklarer en lett måte å utvikle en software applikasjon med å bruke agile teknikker og konsepter.

AUP bruker syv disipliner for utvikling:

1. **Modell**. Forstå organisasjonens virksomhet, problemet som vil bli løst av prosjektet og en mulig løsning av det problemet.
2. **Implementasjon**. Transformer modell til kjørbar kode og gjør grunnleggende tester.
3. **Testing**. Gjør en objektiv evaluering for å garantere kvalitet. Dette inkluderer å finne defekter, bekrefte at systemet fungerer som designet og validere at kravene er møtt.
4. **Deployment**. Planlegg utgivelsen av systemet og jobb mot målet for å gjøre systemet tilgjengelig for brukere.
5. **Konfigurasjon håndtering**. Håndter adgang til prosjektets artefakter. Dette inkluderer ikke bare å ha en oversikt over forskjellige artefakter versjoner over tid men også kontrollere og håndtere forandringer til dem.
6. **Prosjektledelse.** Direkter de forskjellige aktivitetene som tar hånd under prosjektet. Dette inkluderer håndtering av risiko, direkter prosjekt medlemmene og koordinere med personene og systemene

på utsiden av prosjektet for å forsikre at produktet blir levert i tide og innenfor budsjettet.

7. **Miljø**. Støtt resten av anstrengelsen med å garantere at de ordentlig prosessene, veiledning og verktøy er tilgjengelig for prosjekt medlemmene under behov.

AUP er også basert på disse filosofiene:

1. **Ditt personale vet hva de gjør**. Personer kommer ikke til å lese detaljert prosess dokumentasjon, men det de kommer til å ville få er veiledning på et høyt nivå og/eller opplæring av og til. AUP gir deg tilgang til detaljer hvis du er interessert men tvinger ikke de på deg.
2. **Enkelhet**. Alt er beskrevet konsist med bruk av en håndfull av sider, ikke tusenvis av dem.
3. **Smidighet**. AUP følger verdiene og prinsippene til "agile software development" og Agile Alliance.
4. **Fokus på viktige aktiviteter.** Å ha fokuset på de aktiviteten som faktisk er nødvendige og ikke alle mulige ting som kan skje til deg på et prosjekt.
5. **Verktøy selvstendighet.** Du kan bruke hvilket som helst verktøy du vil med AUP. Anbefalingen er at du bruker de verktøyene som passer best til jobben, som ofte er de enkleste verktøyene.
6. **Du vil skreddersy AUP til å møte dine behov.**

(Wikipedia, 2 April 2020)

3.2.2 Kommunikasjon med database- middleware

En løsning for kommunikasjon mellom databasen og klienten er ved implementasjon av et mellomledd. Dette mellomleddet vil oppføre seg som et REST-api (Representational state transfer), som betyr at klienten sender en http-forespørsel til en spesifikk url, og får et definert svar tilbake. En url er koblet sammen med en metode som vil kjøres ved en forespørsel, og denne metoden håndterer kommunikasjonen med databasen, og returnerer et svar til klienten.

3.2.3 Desigmønster - MVP

Model View Presenter (MVP) er et designmønster for utvikling av programvare. Mønsteret deler programmet inn i tre deler, model, view og presenter.

Modellen ('model') er et grensesnitt som definerer dataen soms kal vises eller håndteres i brukergrensesnittet. Brukergrensesnitt ('view') er et passivt grensesnitt som viser dataen, og sender kommandoer videre til presentatøren. Presentatøren ('presenter') utfører handlinger på modellen og brukergrensesnittet. Den henter data fra modellen og formaterer det for visning i brukergrensesnittet.

3.2.4 Applikasjonstilstand - Flux

Flux er en applikasjonsarkitektur som blant annet blir tatt i bruk av Facebook. Flux er ikke et rammeverk, men et mønster, som følger en enveis flyt av data. Flux består av tre hoveddeler, som er "dispatcher", "stores" og "views" (React komponenter).

3.3 Diskusjon av alternativene

3.3.1 Utviklingsmetode

Både Scrum og AUP er agile utviklingsmetoder, og vil begge hjelpe gruppen i å nå et tilfredsstillende sluttresultat. Derimot har gruppen erfaring med bruk av Scrum, men har ikke erfaring med bruk av AUP. I tillegg til at gruppen og Prosjekteier har ukentlige møter, passer det best å ta i bruk Scrum med sprints på en uke.

3.3.2 Kommunikasjon med database

Å koble klienten og databasen sammen med bruk av et mellomledd har fordelen av interoperabilitet. Mellomleddet kan endres uten å måtte publisere en ny versjon av klienten, og det separerer databasehåndtering fra klient koden, som kan føre til en ryddigere kodebase. Styreportalen AS arbeider med å lage en mobilapplikasjon for medlemssiden, og et mellomledd kunne da bli brukt for både web og mobilapplikasjonene.

Å koble klient og database direkte sammen uten et mellomledd har fordelen at det er raskere. Et mellomledd kan øke responstiden av kommunikasjonen mellom klient og databasen, da all kommunikasjon først må gjennom

mellomleddet. I en applikasjon hvor store deler av brukergrensesnittet bygges opp av informasjon levert av databasen, vil en lavere responstid være svært positivt for brukeropplevelsen.

Prosjektet verdsetter brukeropplevelse høyt, og det gir dermed mest mening å gå fremover med direkte sammenkobling av database og klient.

3.3.3 Designmønster

Den største forskjellen mellom MVC og MVP er at MVC fjerner logikk fra brukergrensesnittet, og at presentatøren håndterer brukerinteraksjonen. Dette fører til en ren separasjon mellom brukergrensesnitt og modell, som gjør det lettere å teste, men fører til mer arbeid. Gruppen har valgt å gå videre med MVC, da gruppen har mer erfaring med dette designmønsteret enn MVP.

3.3.4 Applikasjonstilstand

Redux er et JavaScript bibliotek, mens Flux er et mønster. Facebook har utviklet verktøy for å hjelpe implementasjonen av Flux. Både Flux og Redux har handlinger som kan endre tilstanden. I Flux er en handling et JavaScript objekt, og dette er standard tilfellet i Redux også, men ved å implementere mellomledd i Redux, så kan handlinger også være funksjoner. Dette er hjelpsomt for å abstrahere vekk funksjoner for endring av tilstanden, som for eksempel kommunikasjon med database.

Templaten prosjektet bygger på tar allerede i bruk Redux. I tillegg har prosjektmedlemmene mer erfaring med Redux. Dette gjør at Redux er den mest attraktive løsningen, og prosjektet tar dermed i bruk Redux.

3.4 Valg av verktøy

3.4.1 Visual studio code

visual studio code er en lett kodeeditor fra microsoft. Den kommer med allerede innebygd støtte for javascript, react og node. Det finnes også et marked der man kan laste ned utvidelser for støtte til andre språk som java, c#, python o.l. (Visual studio code, uå)

3.4.2 Git og github

Git er et versjonskontrollsystem som utviklere kan bruke til å dele kodebase. Det gjør det også mulig å parallellisere arbeid ved å gå ut i "brancher". En branch kan symbolisere en bestemt funksjon som implementeres. Når en

funksjon, eller branch, er ferdig implementert kan man gjøre en "pull request". En "pull request" lar resten av medlemmene i prosjektet se gjennom koden, og enten godkjenne at branchen skal inn i kodebasen, eller komme med kommentarer om ting som bør endres.

Github er verten gruppen har brukt for å lagre kodebasen gjennom git.

3.4.3 Trello

Trello er en plattform som lar deg organisere prosjekter ved hjelp av tavler, lister og kort. Dette gjør det lettere å holde oversikt over hvilke oppgaver som må gjøres, hvem som skal gjøre dem, og status på dem.

3.5 Prosjektmetodikk

I dette delkapitlet blir det beskrevet hvilke utviklingsmetodikk som er benyttet under prosjektet. Det gis en oversikt over bruk av Scrum, opprettet prosjektplan, og risikovurdering.

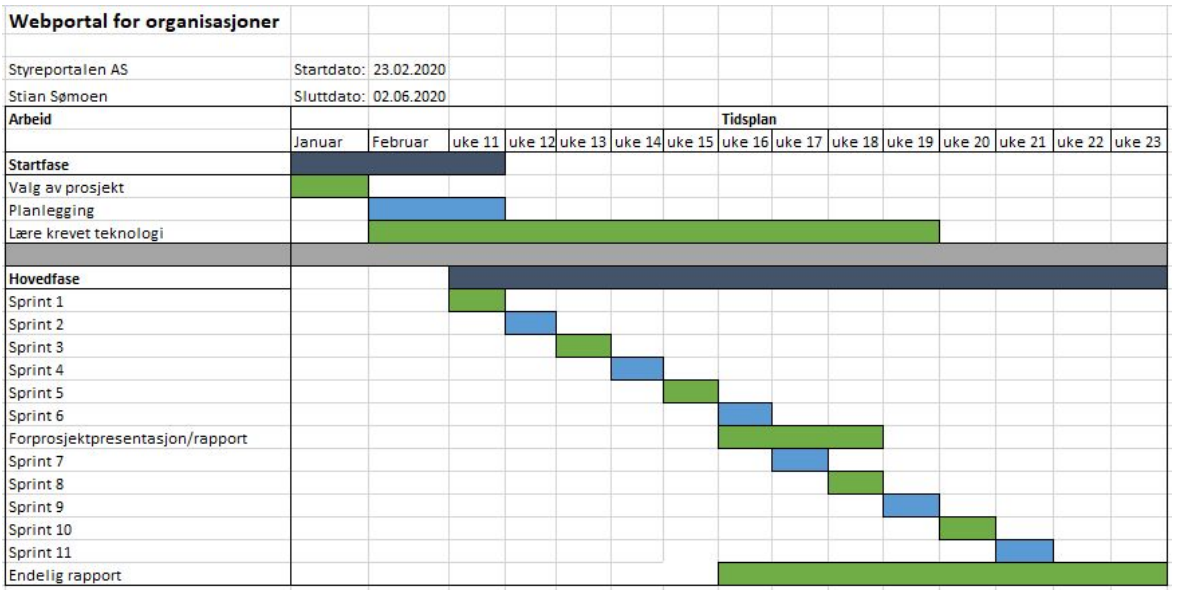
3.5.1 Scrum

Gruppen har valgt å ta i bruk Scrum som utviklingsmetode, som ble beskrevet i punkt 3.1.1. Under dette prosjektet vil det benyttes Scrum med sprinter på en uke, med møter hver tirsdag. Prosjekteier vil komme med tilbakemeldinger på arbeidet som har vært utført i sprinten, og vil komme med nye brukerhistorier som del av backloggen, som beskriver nye funksjonaliteter og oppgaver som kan startes på i neste sprint. Det vil være opp til Scrummesteren å sørge for at arbeidsmengden ikke blir for stor, og for å evaluere sammen med utviklingslaget hva som er realistisk å få utført i løpet av sprinten.

3.5.2 Prosjektplan

Da gruppen kommer til å ha ukentlige møter med Prosjekteier i tillegg til at gruppemedlemmene vil ta kontakt med Prosjekteier angående ulike spørsmål om prosjektet, vil det være høyt ønskelig for både gruppen og oppdragsgiver at funksjonaliteten og brukergrensesnittet til applikasjonen styres av Prosjekteier, med restriksjoner pålagt av gruppen ettersom hva det blir tid til å utvikle. Gruppen tar da for seg implementering av funksjonalitet, får Prosjekteier til å kontrollere at funksjonalitet fungerer forventet, fikser mulige feil, og går videre til neste funksjonalitet. Dette innebærer også at Prosjekteier kontrollerer at brukergrensesnittet opprettholder god brukervennlighet.

Gruppen opprettet et GANTT diagram for å fremvise planlagt prioritering av prosjektet. Her vises det at prosjektet deles opp i 11 sprinter på en uke hver.



Webportal for organisasjoner															
Styreportalen AS	Startdato:	23.02.2020													
Stian Sømoen	Sluttdato:	02.06.2020													
Arbeid								**Tidsplan**							
	Januar	Februar	uke 11	uke 12	uke 13	uke 14	uke 15	uke 16	uke 17	uke 18	uke 19	uke 20	uke 21	uke 22	uke 23
Startfase															
Valg av prosjekt															
Planlegging															
Lære krevet teknologi															
Hovedfase															
Sprint 1															
Sprint 2															
Sprint 3															
Sprint 4															
Sprint 5															
Sprint 6															
Forprosjektpresentasjon/rapport															
Sprint 7															
Sprint 8															
Sprint 9															
Sprint 10															
Sprint 11															
Endelig rapport															

Figur 6: GANTT diagram

Originalt var planen for gruppen å utføre 11 sprinter på 1 uke hver. Grunnet effektivt arbeid fra gruppen, samt mangel på struktur i databasen som har ført til at gruppen ikke har kunnet arbeide med noen funksjoner, ble gruppen ferdig 1 uke før planen. Revidert gantt skjema blir utforsket i 4.8.1.

Nedenfor beskrives deler av arbeidet utført i løpet av de 10 sprintene.

Sprint	Arbeid
1	I første sprint arbeidet gruppen med å opprette en prototype som innehold et innloggings - og registreringssystem for brukere, samt en informasjonsstrøm.
2	I andre sprint arbeidet gruppen med å utføre endringer i informasjonsstrømmen, basert på tilbakemeldinger fra ekstern veileder. Innlegg over en spesifikk lengde blir 'kuttet ned', slik at brukeren selv må trykke på 'se mer' for å vise hele innlegget. Bilder og filer som er en del av innlegget blir også vist, samt dato og navn på forfatter av inlegget. Gruppen startet også arbeid på en prosjektside, og viser eksisterende prosjekt ved siden av informasjonsstrømmen. Dette er starten på opprettelsen av en startside for applikasjonen. Til slutt opprettet gruppen en sidemeny for fremvisning av brukerinstillinger. **Gruppen når sin første milepæl: Prototype med fungerende innlogging**
3	I tedje sprint Videreutviklet gruppen prosjektsiden og sidemenyen for fremvisning av brukerinstillinger. Brukerinstillingsmenyen kan nå brukes til å endre passord på brukeren. I informasjonsstrømmen har nye tilbakemeldinger fra ekstern veileder ført til nye endringer. Gruppen endret fonten på tittel og innhold i innleggene. Informasjonsstrømmen har blitt tilpasset andre enheter, som mobiler. I tillegg til å ha en informasjonsstrøm og en liste med prosjekter, la gruppen til en liste med organisasjoner brukeren er medlem i, samt en liste over de neste 5 aktivitetene i organisasjonene bruker er medlem i
4	I fjerde sprint opprettet gruppen en kalender med fremvisning av aktiviteter. Kalenderen fremviser aktivitene, har støtte for visning av måneder, uker, dager og agenda, hvor agenda viser alle aktivitetene og hvilke tidsrom de befinner seg i.
5	I femte sprint tok gruppen delvis påskeferie. Som en utfordring fra ekstern veileder implementerte likevel gruppen et kommentarfelt under innlegg i informasjonsstrømmen, og endret formatet på pdf-filer. Gruppen opprettet en side for organisasjoner, hvor medlemmene i organisasjoner blir fremvist. etter tilbakemelding fra ekstern veileder. I tillegg la gruppen til muligheten for å melde fravær/deltagelse i en aktivitet i både startsiden og i kalender. **Gruppen når sin andre og tredje milepæl: Fullført informasjonsstrøm, og fullført kalender**
6	I sjette sprint gruppen arbeidet på å videreutvikle organisasjonssiden. La til støtte for fremvisning av filer, og endret på utseende av medlenslisten etter tilbakemelding fra ekstern veileder. I prosjektsiden la gruppen til menyer for fremvisning av innlegg og filer som tilhørte prosjektet.
7	I sjuende sprint endret gruppen på tilkobling mellom klient og database. I starten av prosjektet tok gruppen i bruk et mellomledd, men ekstern veileder kommenterte på at responstiden fra databasen var større enn ønskelig. Gruppen endret dermed fra å ta i bruk et mellomledd, til å koble databasen og klienten sammen direkte. I tillegg endret gruppen prosjektsiden til å støtte fremvisning av musikk-noter tilkoblet prosjektet. **Gruppen når sin fjerde og femte milepæl: Fullført kobling av database og klient, samt fullført prosjektside**
8	I åttende sprint arbeidet gruppen med utseende av en rekke komponenter av applikasjonen basert på tilbakemeldinger fra ekstern veileder, i tillegg til å endre oppførsel til et kommentarfelt under ett innlegg i informasjonsstrømmen. Tidligere ville en ny kommentar føre til at alle kommentarer lukket seg, men etter endringen oppførte kommentarfeltet seg som forventet.
9	I åttende sprint kom ekstern veileder med en forespørsel om en side for fremvisning av faktura. Dette var fullført i niende sprint. **Gruppen oppnår sin sjette milepæl: fullført fakturaside**
10	I tiende sprint legger gruppen til små endringer i utseende til applikasjonen. Ekstern veileder omtaler applikasjonen som '95%' fullført.

Tabell 4: Forklaring av sprinter

3.5.3 Risikovurdering

For å vurdere risiko i prosjektet har gruppen vurdert hvilke situasjoner som kan ha en negativ effekt over prosjektets progresjon og utfall. Gruppen har funnet fem situasjoner som kan ha en negativ effekt hvis de ikke mitigeres gjennom nødvendige tiltak.

Første risiko er at firebase, databasen til prosjektet, går ned. I perioden prosjektet blir utviklet er verden under effekt av korona-pandemien, og internett har som konsekvens fått større belastning. Det er ikke utenkelig at firebase dermed kan bli overbelastet i perioder, som kan føre til at databasen blir treg, eller i verste fall blir utilgjengelig. Det er ingen tiltak gruppen kan gjøre for å motvirke korona-pandemien, og kan derfor ikke minske belastningen på internett eller firebase. Prosjektet er derimot ikke avhengig av at databasen er tilgjengelig til alle tider, da informasjonen kan lagres lokalt. Dette er et spesialtilfelle, og har dermed kunstig høy sannsynlighet for tiden prosjektet utvikles i.

Andre risiko er mangel på kunnskap om teknologien prosjektet utvikles i. Dette kan ha en stor effekt på prosjektet, da mangel på kunnskap fører til lavere kvalitet på produktet, og kan føre til at mindre arbeid blir utført. For å mitigere dette har gruppen tatt initiativ til å lære seg teknologien før prosjektstart, samt lærer ved å arbeide på prosjektet.

Tredje risiko er at internettet går ned som gjør at kommunikasjon mellom prosjektmedlemmene, oppdragsgiveren og github som er hosting servicen prosjektet bruker blir utilgjengelig. Vanligvis er ikke internett nede så lenge men i tilfelle det skjer kan kontinuerlig nedlastning av koden når den blir oppdatert på github.

Fjerde risiko er mangel på oppgaveforståelse som kan gjøre at funksjoner som ikke skulle bli implementert blir implementert. Det vil føre til at tid og ressurser blir kastet bort på funksjoner som ikke skulle være i prosjektet i steden for funksjoner som kunne være nyttig for prosjektet. Dette blir håndtert med kontinuerlig møter og tilbakemeldinger fra oppgavegiver for å få en bedre forståelse av oppgaven og av hva som har blir fullført.

Femte risiko er sykdom. Som nevnt over er prosjektet gjennomført i en verden under effekt av korona-pandemien, dette gjør at medlemmene av prosjektet kan lettere å bli syk og bli utilgjengelig for en periode under utviklingen av løsningen for prosjektet. For å unngå at dette skal skje følger prosjektmedlemmene retningslinjene som har blitt satt av regjeringen samtidig

som de har god hygiene vett. I tillegg skal alt arbeid etter en arbeidsøkt bli lastet opp på våres nettbaserte lagrings plass Github der de andre medlemmene kan fortsette på arbeidet i tilfelle en blir syk. Dette vil minske skaden for prosjektet om en på gruppen blir utilgjengelig på grunn av sykdom.

3.6 Evalueringsplan

I de neste avsnittetene diskuteres raskt hvordan gruppen planlegger at hver modul er korrekt, at modulene fungerer sammen, at løsningen er brukervennlig og at løsningen tilfredsstiller brukerkravene.

3.6.1 Unit testing

For å teste at hver modul er korrekt tar gruppen i bruk React Testing Library for å opprette unit tester. En unit test er kode som tester at spesifikke komponenter leverer et forventet resultat, og er nyttig for å validere at forventet resultat ikke endrer seg fra forskjellige enheter, forskjellig applikasjonstilstand, og ved nye endringer.

3.6.2 Integrasjonstesting

For å teste at modulene fungerer sammen tar gruppen i bruk integrasjonstesting med metoden Top Down Testing. Integrasjonstesting fungerer ved å sette sammen de ulike delene av applikasjonen, for å sjekke at modulene fungerer som forventet i samkjør. Top Down Testing består av å teste toppnivå modulene når de er ferdig, og ta i bruk 'stubs' [3]for moduler i et lavere nivå hvis de ikke er ferdig.

3.6.3 Bruker & Akseptansetesting

I dette prosjektet har Styreportalen AS tatt ansvar for både bruker og akseptansetestingen. Gruppen laster opp applikasjonen til firebase hosting når nye funksjonaliteter er implementert, slik at Styreportalen kan teste, og validere, at funksjonalitetene oppnår forventet resultat.

[3] En stub er kode lagt inn som leverer tilbake forventet svar, uten at modulen som håndterer den delen av applikasjonen er ferdig.

4 DETALJERT DESIGN

I dette kapitlet beskrives design av produktet. I punktene under blir løsning og fremgangsmåte beskrevet.

4.1 Innledning

I innledningen av prosjektet ble gruppen introdusert for Prosjekteier Stian Sømoen. Sammen med Stian gikk gruppen gjennom kravspesifikasjoner og ønsket funksjonalitet for den nye medlemssiden. For å opprette kravspesifikasjonene har Prosjekteier tatt i bruk den eksisterende medlemssiden, og tilbakemeldinger fra kunder.

Prosjekteier gikk gjennom eksisterende løsning, hvorfor et nytt produkt var ønsket, og hvilke ressurser gruppen fikk tilgang til for å fullføre prosjektet. Gruppen fikk også et innblikk i hvilke kunnskaper som var nødvendig for å kunne utføre prosjektet.

Under oppstart satt gruppen opp utviklingsmiljø, fikk tilgang til kanban-board på trello og en testdatabase

4.2 Planlegging

For at prosjektets progresjon skulle være på et ønskelig nivå satt noterte gruppen ned hvilke teknologier som burde utforskes før prosjektstart, ettersom at gruppens kunnskap om teknologien var kritisk for utvikling av produktet.

Naturen til prosjektet var at produktet ikke var ferdig planlagt, så gruppen kunne dermed ikke få en detaljert beskrivelse av funksjonalitet. Gruppen opprettet dermed ikke en detaljert plan over de ulike scrum-sprintene for prosjektet, da Prosjekteier ville komme med brukerhistorier og tilbakemeldinger ukentlig.

For å holde styr på oppgavene i prosjektet er det benyttet trello, en Kanban-tavle aktig applikasjon hvor egendefinerte oppgaver legges til egendefinert lister. Gruppen og Prosjekteier kan benytte tavlen for å legge ut oppgaver, spørsmål og forslag, samt kommentere på en oppgave, markere hvem som arbeider på en oppgave, og hvilke stadie en oppgave er i, f.eks under arbeid, under testing eller ferdig.

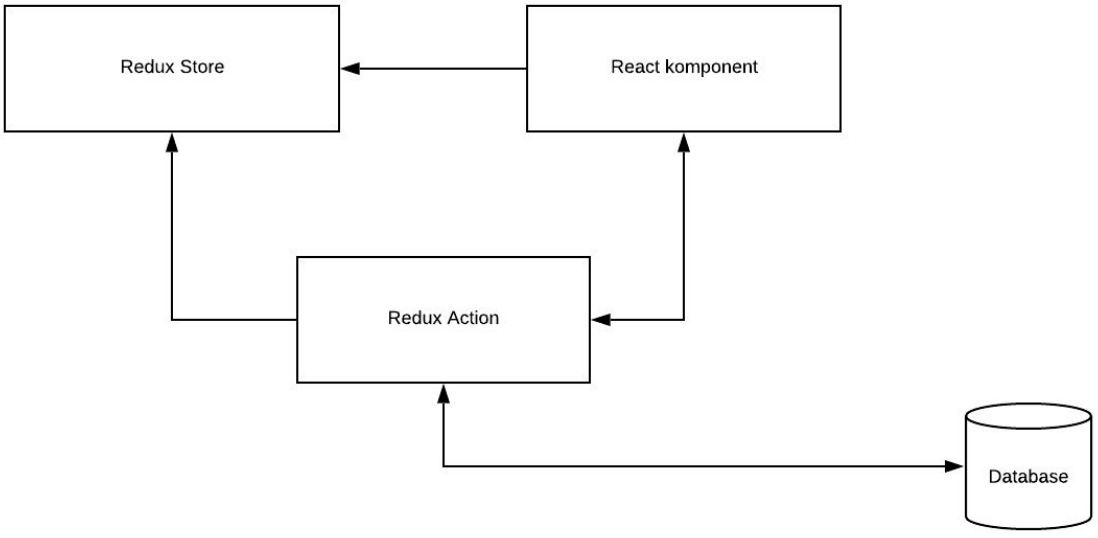
4.3 Medlemssiden

4.3.1 Integrasjon i systemet

Styreportalen AS har et system som består av flere applikasjoner koblet sammen av en database, slik at systemet kan snakke sammen over en rekke forskjellige applikasjoner. En av applikasjonene i systemet er medlemssiden. Medlemssiden kommer til å ha en egen web-url. Som del av dette prosjektet vil ikke medlemssiden kobles sammen med en felles database med resten av systemet, som kan føre til at databasestrukturen som tas i bruk i prosjektet ikke er den endelige databasestrukturen, og kan føre til at endringer i hvordan applikasjonen håndterer informasjon fra databasen må endres.

4.3.2 Arkitektur

Gruppen har fått fullstendig frihet for valg av arkitekturen i medlemssiden utenom valg av database og hoved-teknologi, Firebase og React. Gruppen har dermed valgt å ta i bruk Redux sammen med react, som fører til designmønsteret MVC. Satt sammen vil Redux store, som holder på dataen tatt i bruk av applikasjonen, tilsvare modellen, React komponenter vil tilsvare brukergrensesnittet, og kontrolløren vil tilsvare 'actions' i redux.



Figur 7: React-Redux som MVC

Applikasjonen er strukturert med en mappestruktur som følger strukturen som var eksisterende i React templaten applikasjonen er bygget på. Denne mappestrukturen inneholder deler inn i 3 hovedgrupper: 'Store', som inneholder alle Redux-filene, 'Components', som inneholder alle React-komponentene, og 'Styles', som inneholder styling av React-komponentene i form av scss filer.

4.3.3 Funksjonalitet

Medlemssiden består av åtte hovedkomponenter, og hver hovedkomponent inneholder andre komponenter som har sin egen spesifikke funksjonalitet. Hovedkomponentene i Medlemssiden deles inn i:

- Innloggingsside
- Navbar
- Sidebar
- Startside
- Organisasjonsside
- Prosjektside
- Kalenderside
- Fakturaside

Innloggingssiden gir brukeren muligheten til å logge inn i systemet, gitt at de har en gyldig bruker. Siden gir også brukeren mulighet til å tilbakestille passordet sitt hvis det er behov for det.

Navbaren består av fem knapper. Knapp en kollapser/utvider sidebaren, som i en kollapset tilstand kun viser ikonet til meny-elementet, mens i en utvidet tilstand også viser navnet på elementet. Knapp to logger brukeren ut. Knapp fører nettleseren inn/ut av fullskjermsmodus. Knapp fire viser notifikasjoner, og knapp fem utvider offsidebar, som inneholder info om brukerens profil.

Sidebaren består av en meny som inneholder fem forskjellige elementer som kan brukes til å navigere rundt i applikasjonen. Element en, startside, bringer brukeren til startsiden. Element to, organisasjoner, utvider seg til å vise en

sub-meny som inneholder alle organisasjonene brukeren er medlem i, som bringer brukeren til organisasjonssiden hvis trykket på. Element tre, kalender, bringer brukeren til kalendersiden hvis trykket på. Element fire, prosjekter, bringer brukeren til prosjektsiden hvis trykket på. Element fem, faktura, bringer brukeren til fakturasiden hvis trykket på.

Startsiden inneholder en informasjonsstrøm som viser utlegg fra alle organisasjonene brukeren er medlem i. Hvert utlegg kan inneholde tekst, bilder og filer. Informasjonsstrømmen er designet til å laste inn fem utelgg fra hver organisasjon brukeren er medlem i, og hvis brukeren når bunnen av informasjonsstrømmen vil fem nye lastes inn, helt til det ikke er flere utlegg igjen.

Startsiden inneholder en søkefunksjonalitet, som gjør det mulig for brukeren å søke etter spesifikke ord i et utlegg. Startsiden inneholder også en liste over organisasjonene brukeren er medlem i, med en lenke til organisasjonssiden samt mulighet til å filtrere vekk utlegg fra denne organisasjonen, en liste over prosjekter, med lenke til prosjektsiden, og en liste med aktiviteter, med mulighet til å melde om man kommer til aktiviteten eller ikke.

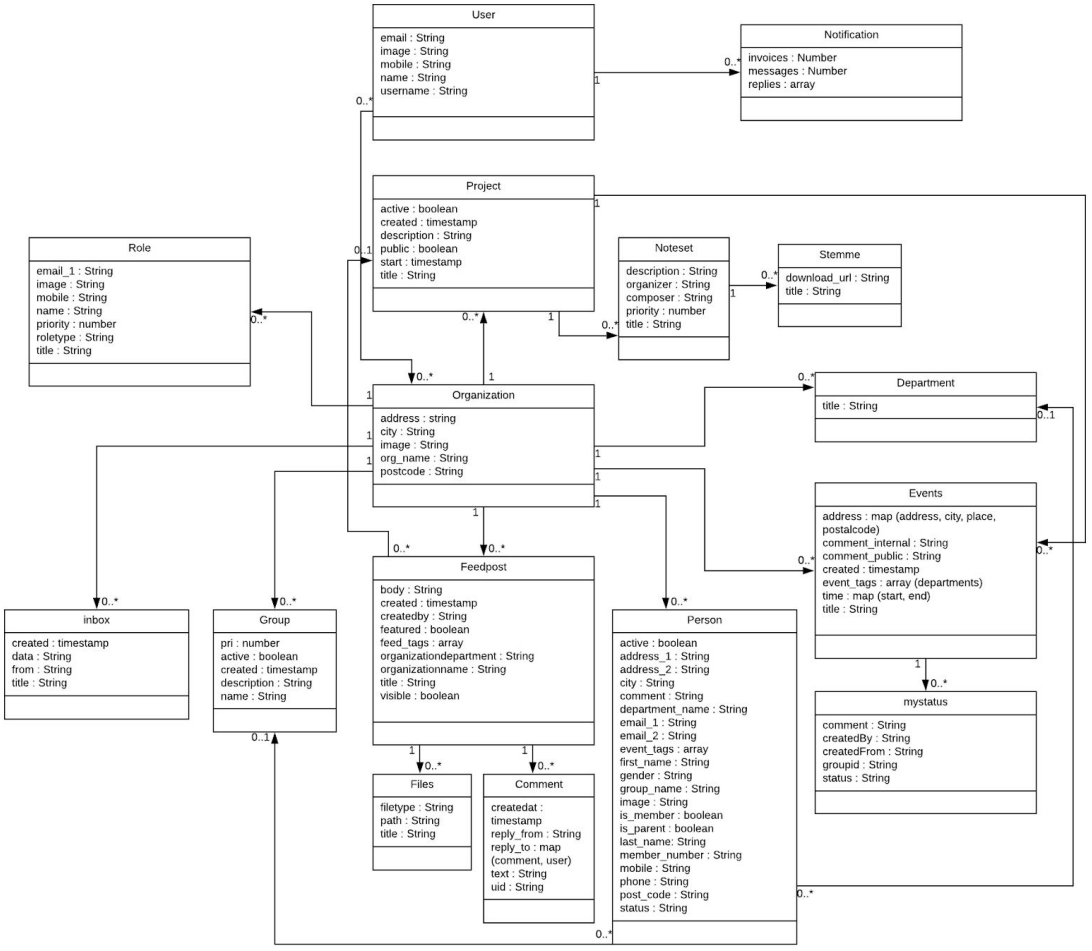
Organisasjonssiden inneholder en liste over alle medlemmene i organisasjonen, med mulighet til å søke etter et medlem, en liste over styret i organisasjonen, et filarkiv, og en 'min info' side, hvor brukeren kan redigere og sende inn forslag til endring av informasjonen organisasjonen har på brukeren.

Prosjektsiden inneholder en liste over alle prosjekter. Her kan man trykke seg inn på et prosjekt, og se detaljer om prosjektet, blandt annet hvilke utlegg, hvilke filer, hvilke aktiviteter, og hvilke noter som er tilknyttet prosjektet.

Kalendersiden inneholder en kalender med oversikt over kommende aktiviteter. Her kan man trykke seg innpå en aktivitet for å melde om man kommer eller ikke. Kalenderen kan vise månedlig, ukentlig, eller daglig, samt kan sorteres etter agenda, hvor kun kommende aktiviteter vises.

Fakturasiden viser ubetalte faktura brukeren har.

Hver komponent har ansvar for å hente inn dataen den trenger for å fungere, og gjør dette ved å aktivere en metode som henter ut informasjon fra databasen.



Figur 8: Databasestruktur

4.4 Brukergrensesnitt

4.4.1 React-komponenter

Brukergrensesnittet i medlemssiden er bygget opp av React-komponenter. React-komponenter gjør at brukergrensesnittet kan deles opp i forskjellige deler, og kan lett brukes på nytt. React-komponenter er generelt uavhengige av hverandre, og følger prinsipper "Solid-responsibility" ved at de gjør sin oppgave i isolasjon.

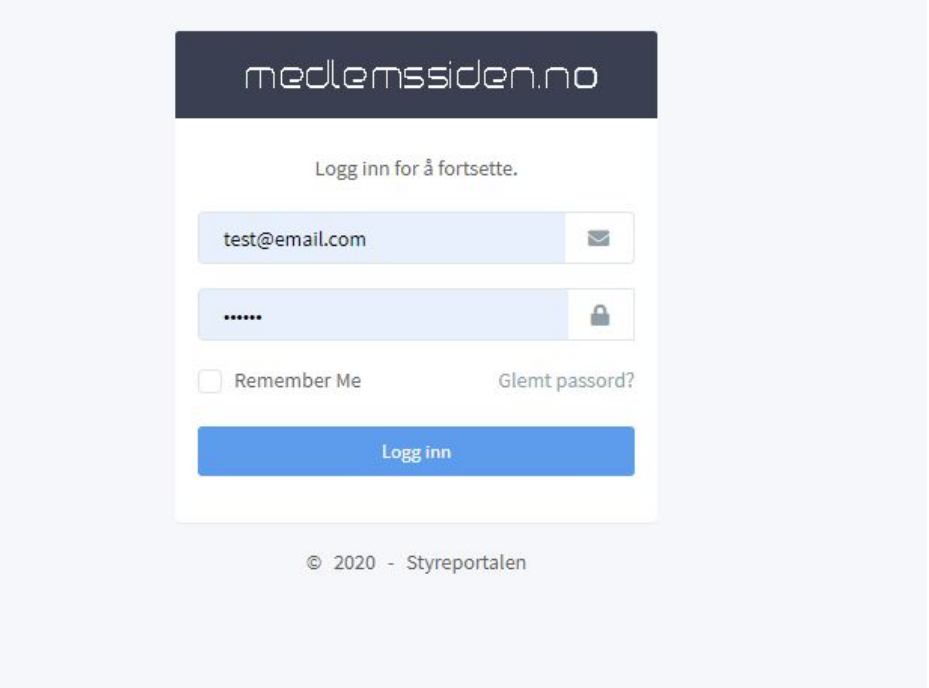
React komponenter finnes i flere varianter. I prosjekter har gruppen tatt i bruk 'Stateless functional components', men template prosjektet bygger på tar i bruk 'Class components', så prosjektet er bygget opp på både

klassekomponenter og funksjonelle komponenter. Forskjellen på klassekomponenter og funksjonelle komponenter er at klassekomponenter er bygget opp som JavaScript klasser, mens funksjonelle komponenter er bygget opp som JavaScript funksjoner, og at klassekomponentene kan inneholde en tilstand, mens funksjonelle komponenter er tilstandsløse. Dette er et problem, men for å føre inn tilstand i funksjonelle komponenter har React ført inn en useState hook, som gjør oppbevaring av tilstand mulig.

For å sikre at komponentene i prosjektet er gjenbrukbare, er det gjort et forsøk på å generalisere komponentene slik at de ikke er tilknyttet en spesifikk del av applikasjonen. Et eksempel på en gjenbrukbar er Feed-komponenten, som tas i bruk i både startsiden og i en prosjektside.

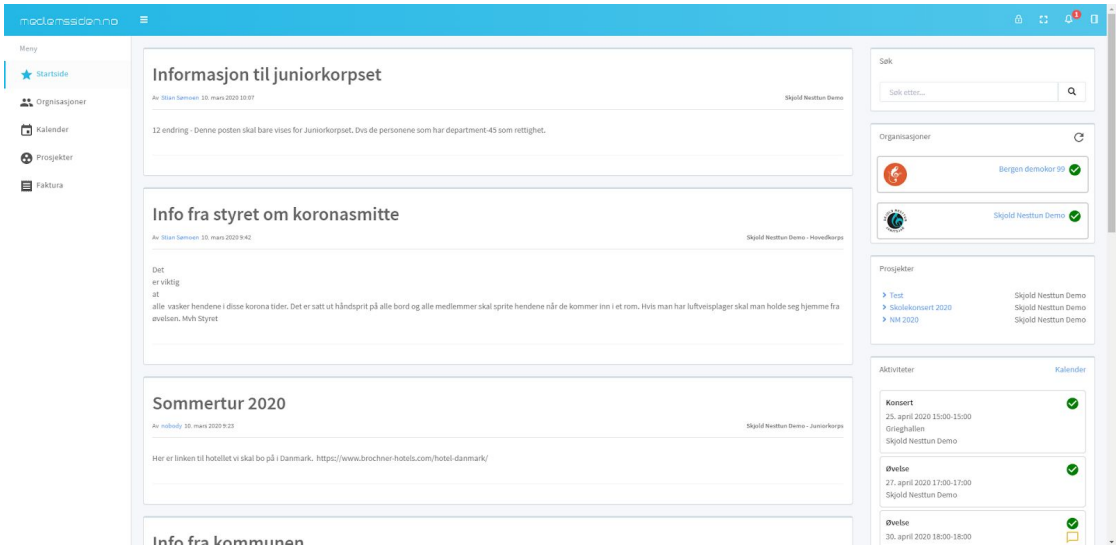
4.4.2 Oppbygging

Applikasjonen er bygget opp mot to typer fremvisninger. Side-fremvisning, som viser en hel side, og base-fremvisning, som har en navbar og en sidebar. Applikasjonen er bygget som en 'single-page application' som betyr at alt skjer på en side, men ved bruk av React-Router-Dom, så vil ulike url-er knyttes til spesifikke komponenter av applikasjonen.



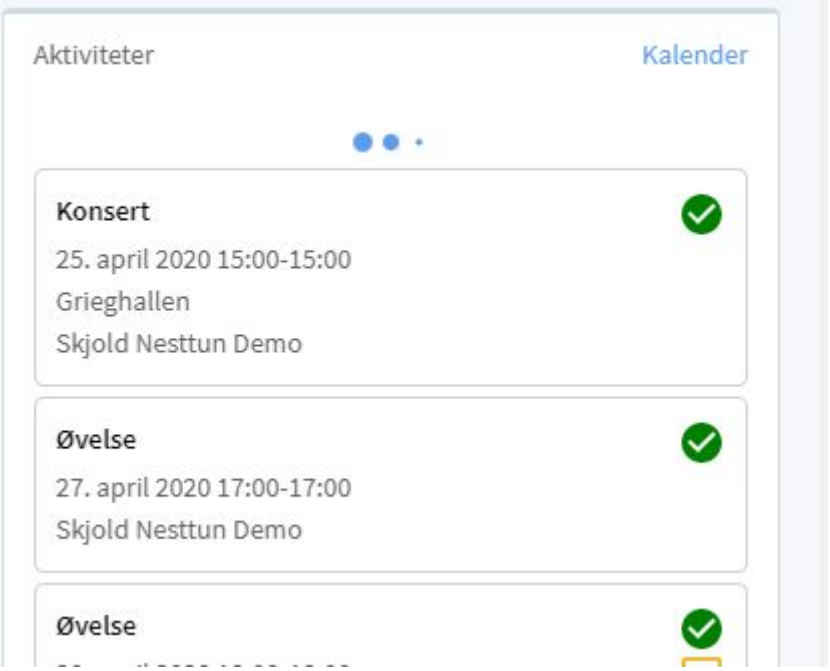
Figur 9: side-fremvisning

Hvis en url er definert i en liste av sider, vil url-en føre til en del av applikasjonen som ikke inneholde basen av applikasjonen, nemlig sidebar og navbar, men hvis url-en ikke er definert i listen vil url-en, vil url-en føre til en del av applikasjonen som inneholder basen.



Figur 10: base-fremvisning

Store deler av applikasjonen bygger opp brukergrensesnittet med informasjons hentet inn fra databasen. For å indikere til brukeren at data lastes inn, vises en 'ballsync' indikator i brukergrensesnittet.



Figur 11: Indikator for innhenting av data fra database

Klikkbare elementer indikeres ved at musen blir til en 'pointer', altså et ikon av en hånd med pekefingeren frem, i tillegg til at noen elementer, nærmere sagt lenker, også endrer farge.

4.4.3 Universell utforming

I prosjektet har ikke gruppen fokusert på universell utforming, da dette ikke var et krav fra oppdragsgiver. Gruppen har likevel fulgt noen retningslinjer for WCAG.

4.5 Redux

Medlemssiden har en Redux-store for lagring av applikasjonstilstand og data. Hver modul har sin egen 'reducer' som holder styr på modul-tilstanden, mens Redux-store holder styr på alle reducer-tilstandene. React-komponentene henter ut dataene de trenger fra Redux-store. Dette gjør de gjennom React-redux hook, useSelector. I tillegg kan komponenter gjøre endringer i tilstanden gjennom React-redux hook, 'useDispatch', hvor de kan 'dispatche' 'actions'. En action kan være enten en metode, eller et objekt. Objektene håndteres av en Redux-reducer, mens en action defineres via et mellomledd i Redux. Hver komponent har selv ansvar for at dataen den trenger er hentet inn

fra databasen, og om den ikke er det, må den 'dispatche' en action som henter inn dataen fra databasen, og lagrer den i en Redux-reducer.

4.6 Responsivitet

Det er et krav fra oppdragsgiver at applikasjonen skal fungere på alle enheter. Dermed har gruppen designet applikasjonen med responsivitet i tankene, det vil si at den tilpasser seg selv for de forskjellige enhetene. For å oppnå responsivitet har gruppen testet ut applikasjonen på forskjellige enheter, og definert 'media-queries' gjennom css, som endrer på css-definisjonen basert på størrelsen på enhetens skjerm.

```
.org-image {
  object-fit: cover;
  margin: 10px;
  max-height: 100px;
  max-width: 100px;
  @media (max-width: 600px) {
    max-height: 80px;
    max-width: 80px;
  }
}
```

Figur 12: Eksempel på media-query

I tillegg til at gruppen har definert media queries, så har gruppen også tatt i bruk eksterne biblioteker som f.eks 'Reactstrap' og 'Material UI', som har designet responsive komponenter.

Figur 13: Organisasjonsmedlem for en skjerm på 521 piksler

Figur 14: Organisasjonsmedlem for en skjerm på 519 piksler

4.7 Arbeidsfordeling

Prosjektgruppen har bestått av tre personer, men oppgavefordelingen har vært relativt avslappet. Det var viktig for gruppemedlemmene at alle jobbet litt med forskjellige ting, slik at alle fikk erfaring fra de forskjellige aspektene av applikasjonen, så gruppen hadde dialog om hvordan oppgavene skulle fordeles. Konklusjonen var at medlemmene selv bestemmer hvilke oppgaver de ville jobbe med, sett at oppgaven ikke var tatt av et annet medlem.

4.8 Endelig prosjektplan

Viser til punkt 3.5.2 hvor prosjektplanen beskrives fra starten av prosjektet.

Siden Scrum ble valgt som gruppen sitt agile rammeverk blir prosjektplanen å lage forskjellige sprints i forhold til scrum sine prinsipper. Grunnet at prosjektet sine interesser og planer var definert fra starten sto det bare igjen å velge en scrum master. Etter scrum master ble valgt ble det satt en tidsramme for sprintene i forhold til leveringsfristen. Det ble bestemt at sprintene skulle vare en uke lang periode og på slutten av hver sprint periode et møte med prosjekt eieren der implementasjonenene som har blitt gjort blir diskutert.

Detaljert forklaring sprintene blir fortalt i tabell 4.

4.8.1 Revidert GANTT-skjema

Figur 15: Revidert GANTT-Skjema

GANTT-Skjemaet i figuren ovenfor viser endelig gjennomføring av prosjektet. Stjerner symboliserer nådde milepæler.

	Milepæler	Kategori	Tilordnet til	Fremdrift	Start	Slutt
Startfase						
	Valg av prosjekt	Fullført	Gruppe 16	100 %	10.02.2020	23.02.2020
	Planlegging	Fullført	Gruppe 16	100 %	23.02.2020	06.03.2020
	Lære krevet teknologi	Går etter plan	Gruppe 16	75 %	23.02.2020	02.06.2020
Hovedfase						
	Prototype med innlogging	Fullført	Petter Knudsen	100 %	10.03.2020	16.03.2020
	Koble klient med database	Fullført	Gruppe 16	100 %	10.03.2020	22.04.2020
	Informasjonsstrøm	fullført	Petter Knudsen	100 %	13.03.2020	07.04.2020
	Prosjektside	fullført	Ole Martinus Rambech	100 %	16.03.2020	21.04.2020
	Organisasjonsside	Fullført	Stefan Go Thuen	100 %	07.04.2020	11.05.2020
	Kalender	Fullført	Petter Knudsen	100 %	30.03.2020	08.04.2020
	Forprosjektspresentasjon/rapport	Fullført	Gruppe 16	100 %	07.04.2020	23.04.2020
	Fakturaside	Fullført	Petter Knudsen	100 %	27.04.2020	08.05.2020
	Endelig rapport	Fullført	Gruppe 16	100 %	06.04.2020	02.06.2020

Figur 16: Forklaring av milepæler

5 EVALUERING

5.1 Evalueringsmetode

For å evaluere at resultatet av applikasjonen oppnår forventet resultat er det nødvendig å utføre tester. I avsnittene under er de ulike metodene vi har brukt til evaluere produktet for å få den beste løsning til problemstillingen.

5.1.1 Scrum som en evalueringsmetode

Scrum er som tidligere nevnt utviklingsmetoden for dette prosjektet. Siden scrum muliggjør tilbakemeldinger gjennom hele utviklingsfasen kan disse tilbakemeldingene brukes til å måle resultatet opp mot problemstillingen. Prosjekteier har bidratt med å gi tilbakemeldinger etter hver sprint slik at gruppen kan gjøre justeringer slik at produktet kan oppfylle kravene om brukbarhet overfor brukergruppen. Tilbakemeldinger fra Prosjekteier går inn i backloggen for neste sprint.

5.1.2 Unit testing

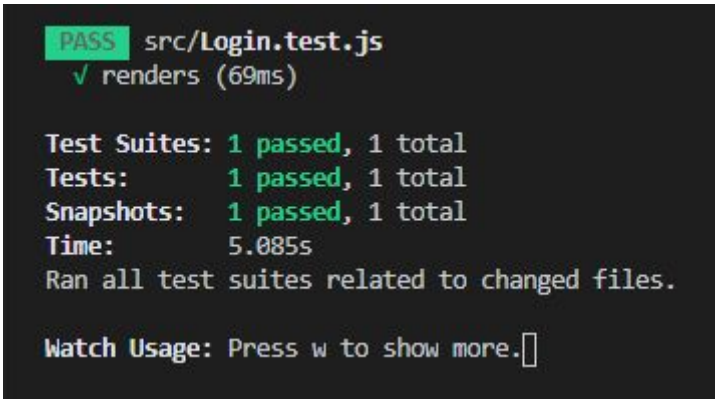
For å unngå at all testing må utføres manuelt, kan man opprette såkalte 'unit-tester'. Unit tester er en form for kode som utfører en spesifikk handling, og sjekker at resultatet er som forventet, og kan automatiseres. For å utføre unit-testene i prosjektet tok gruppen i bruk 'React-Testing-Library'.

React-Testing-Library (RTL) gjør det mulig å utvikle tester som kan teste at React-komponenter vises riktig i forhold til forventet resultat. RTL kan utføre handlinger på komponentene, som f.eks å klikke på en knapp for å endre tilstanden, og dermed sjekke at forventet resultat har endret seg. Dette gjør at gruppen kan skrive tester for komponenter, og ved endringer i applikasjonen kan testene kjøres for å teste at applikasjonens oppførsel fremdeles når forventet resultat.

Etter å ha satt seg inn i hvordan man kunne teste applikasjonen via automatiserte tester møtte gruppen et problem: databasen. For å kunne teste store deler av applikasjonen trengs en verifisert bruker i databasen. I tillegg er store deler av applikasjonen avhengig av data fra databasen. Gruppen og veileder kom frem til at det ville ta for lang tid å sette seg inn i hvordan

automatiserte tester med database fungerte, så gruppen har valgt å ikke gå videre med automatiserte tester for hele applikasjonen.

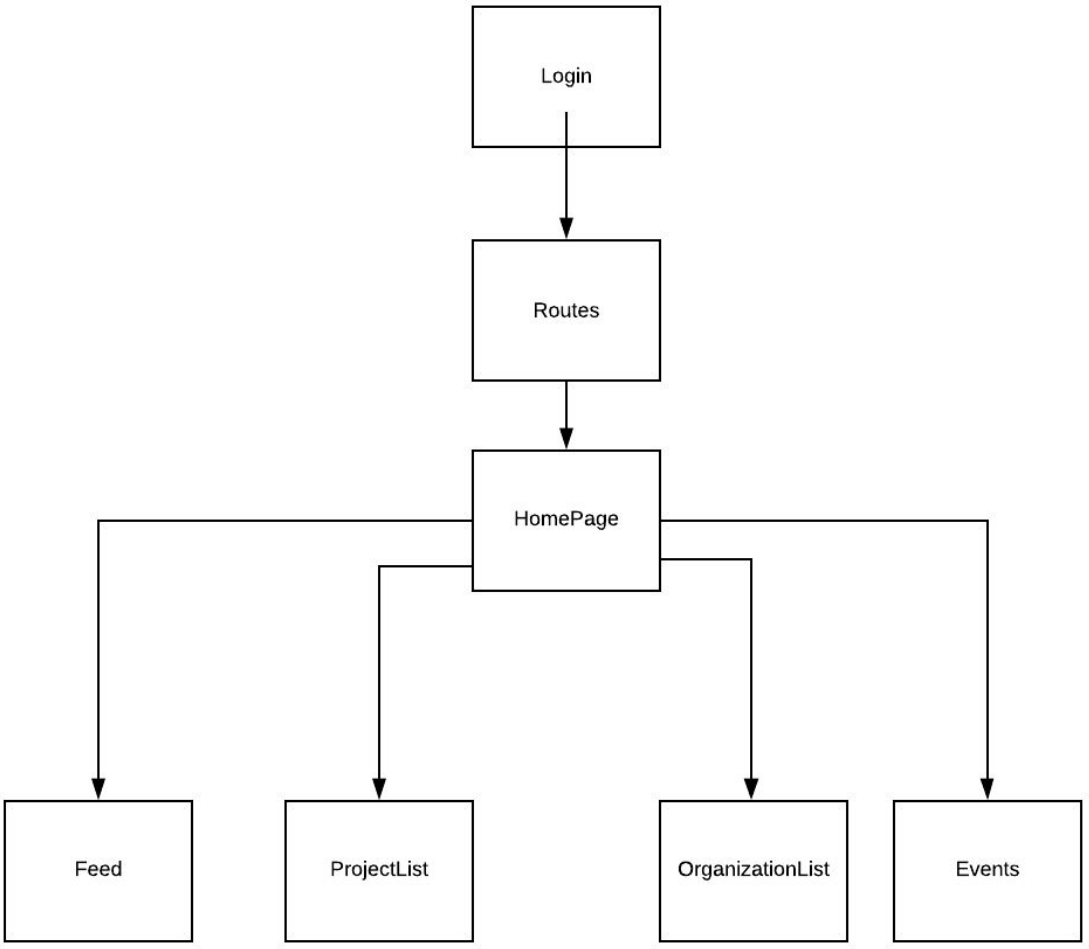
Deler av applikasjonen kan derimot testes. Et eksempel på en test gruppen har opprettet er Login.test. Testen, som er en render test, sjekker at komponenten vises - i tillegg til å lagre et snapshot av komponenten. Dette betyr at så lenge testen er vellykket, har ikke komponenten endret seg fra forrige gang testen ble kjørt.



Figur 17: Resultat av Login.test

5.1.3 Integrasjonstesting

Mens unit-testing tester enkelte moduler av systemet, så tester integrasjonstesting at de forskjellige modulene fungerer sammen. I prosjektet tok gruppen i bruk top-down testing. Top-down testing er en testmetode som tester moduler på toppnivå, og simulerer moduler på lavere nivå hvis modulene ikke er ferdige.



Figur 18: ufullstendig Flow-chart

I figur 18 ser man et flow-chart av en liten del av applikasjonen. Login er i dette tilfellet topp-modulen. Deretter kommer Routes, og deretter Homepage. I Top-Down testing vil, så snart Login-modulen er ferdig, systemet testes. I prosjektet fullførte gruppen Login-modulen før Homepage-modulen. I dette tilfellet måtte gruppen simulere Homepage-modulen for å teste at systemet virket. Forventet oppførsel ved innlogging er at brukeren blir ført fra Login til Homepage gjennom Routes-modulen. Homepage-modulen kunne dermed være en blank side med tittelen 'Homepage'. På denne måten kunne gruppen bekrefte at både Login og Routes modulene fungerte som forventet, med at innlogging endret tilstanden i applikasjonen fra uautorisert til autorisert, som gjorde at brukeren ble redirigert fra ruten '/login', som er Login-modulen, til

ruten '/', som er Homepage-modulen, via Routes-modulen, som er modulen som håndterer hvilke rute brukeren blir sendt til.

For hver modul som ikke er avhengig av en modul i høyere nivå som ikke ble ferdigstilt, testet gruppen at systemet fungerte. Hvis modulen var avhengig av en modul i lavere nivå som ikke var ferdigstilt, simulerte gruppen at modulen var ferdig, slik at forventet oppførsel kunne bekreftes.

5.1.4 Bruker & Akseptansetesting

For bruker og akseptansetesting var Prosjekteier ansvarlig. Når gruppen fullførte en modul og hadde fullført egen testing, ble den nye modulen lastet opp slik at Prosjekteier kunne teste modulen selv. Gruppen fikk dermed tilbakemelding fra Prosjekteier.

Figur 19: Tilbakemelding fra Prosjekteier via trello

Prosjekteier fikk også brukere av den eksisterende medlemssiden.no til å teste applikasjonen. Tilbakemeldingen vi fikk var at brukerne var fornøyde, og at det eneste spørsmålet de hadde var 'Når kan vi ta det i bruk'.

5.2 Evalueringsresultat

Evalueringsmetodene som har blitt benyttet av prosjektmedlemmene har gitt viktig informasjon som lager grunnlaget til problemstillingens besvarelse.

5.2.1 Intervju med oppdragsgiveren

Er du fornøyd med resultatet?
Vi har vist applikasjon til eksisterende brukere og har bare fått gode tilbakemeldinger, synes selv at det ser veldig bra ut, og er fornøyd med resultatet. Vi skal teste mer for å se om det er noe vi vil endre på. Dere har nådd 95% av ønsket funksjonalitet i forhold til hvor mye av strukturen vi har kartlagt, og det er vi veldig fornøyde med.

Hvis vi hadde mer tid ville du gjort noe annerledes?

Er fornøyd teknisk sett, ser bra ut hadde vi hatt mer tid ville jeg fortsette på oppgaven, oppgaver med å forbedre eksisterende funksjoner og legge til mer støtte til foresatte.

Noe du så for deg som ble gjort på en annen måte og om det var bedre eller dårligere?

Synes det har blitt veldig bra, det har blitt lagt til mer funksjonalitet enn som var originalt planlagt. Jeg ble overrasket av funksjoner som har blitt gjort som jeg ikke trodde var mulig, som dynamisk submeny for organisasjoner i menyen.

Etter testing fra oppdragsgiver har arbeidet som har blitt gjort og levert evaluert som programmet de ønsket. Etter å ha hatt tett oppfølging og fulgt utviklingsmetoden som har blitt valgt, ble det sett veldig tidlig i prosjektet at gruppemedlemmene har forstått kravene og ønskene til oppdragsgiveren. I tillegg har innovasjon og kreativitet som har blitt brukt til å lage mulige design og løsninger til tatt vel imot og brukt i den endelige løsningen.

5.2.2 Levert Produkt

Store deler av funksjonalitet og design som var hovedfokuset i oppgaven har blitt gjennomført som ønsket. Funksjoner som Feed, Kalender og Prosjektside har blitt gjennomført som ønsket, men har forbedringspotensiale. Utenom det har oppdragsgiveren sett på designet hver gang det har skjedd en stor endring, dermed har resultatet blitt skreddersydd til oppdragsgiveren sine ønsker.

Siden Firebase er tatt i bruk i prosjektet er det blitt satt opp en Firebase deploy som kjører prosjektet på nettet gjennom Firebase sine servere. Dermed kan prosjektmedlemmene laste opp nye versjoner av prosjektet som er tilgjengelig for ønskede testere. Dette gjør sånn brukere kan teste de nye oppdateringene og gi tilbakemeldinger som lager en brukervennlig nettside.

Grunnet tett oppfølging av oppdragsgiveren og tilbakemeldinger fra brukere er resultatet produktet oppdragsgiveren ønsket og en nettside som er både brukervennlig og lett å vedlikeholde. Ellers er figur 9 et bilde på hvordan det leverte produktet ser ut.

6 DISKUSJON

6.1 Prosjektstart

Gruppen valgte prosjekt i slutten av januar, to måneder før arbeidet på applikasjonen startet i mars. Oppgavebeskrivelsen inneholdt en kravspesifikasjon, som gav en pekepinne på hva prosjektet ville gå ut på. Det ble holdt et møte med Prosjekteier samt intern for å diskutere oppgaven, forventet resultat, og opprette en plan frem mot prosjektstarten. Arbeidsplass var bestemt å være på høyskolen, da oppdragsgivers lokaler ikke var store nok, og det var ønskelig at gruppen lærte så mye om de relevante teknologiene, React, node.js og firebase, som mulig.

Gruppen mottok også noen skisser over hvordan applikasjonen muligens kom til å se ut, og fikk tilgang til både trello og firebase koblet opp mot prosjektet, og oppretter en egen github selv, hvor både ekstern og intern veileder fikk tilgang. ved prosjektstart gikk Prosjekteier gjennom oppgaven i større detalj, og gav gruppen tilgang til en React-template.

6.2 Prosjektarbeid

Prosjektet har ikke vært gjennomført uten utfordringer. I de neste avsnittene diskuteres disse utfordringene.

6.2.1 Mangel på kunnskap

Utover de første ukene av prosjektet var det tydelig at selv om gruppen hadde satt av betydelig tid for å lære seg React i løpet av forprosjektet, var det ikke nok til å kunne hoppe rett inn i utviklingen av prosjektet uten problemer.

Et stort problem var at komponentene i React-templaten tok i bruk klasse-komponenter, mens gruppen selv kun hadde erfaring med funksjonelle-komponenter. Et annet problem var at ingen av medlemmene i gruppen hadde god kunnskap om CSS, og hvordan man endrer utseende på React-komponentene slik som det var ønskelig. Det største problemet kan likevel sies å være mangel på kunnskap om Redux, da dette førte til at prosjektet krasjet i flere situasjoner, og mye tid ble brukt på å finne ut hovrfor.

Det var flere områder hvor gruppen manglet kunnskap, og dette kan ha ført til at utviklingen i starten gikk tregere enn forventet, men gruppen lærte raskt hvordan teknologien fungerte, og hvordan man skulle arbeide med det, og fikk nådd et akseptabelt resultat.

6.2.2 Mellomledd

I utgangspunktet var planen å bruke et mellomledd for kommunikasjon mellom databasen og klienten. I starten av prosjektet ble derfor dette skrevet i node.js som et eget github-prosjekt. Det fungerte som forventet, men da gruppen implementerte indikasjon på nedlastning av data fra databasen ble det åpenbart at det tok for lang tid å hente inn data fra databasen. Dette fikset gruppen med å legge kommunikasjonen inn i klienten, noe som gjennomsnittlig kortet ned innhentingstiden med flere sekunder.

6.2.3 Usikkerhet

Da prosjektet startet var ikke all funksjonalitet og struktur ferdig planlagt. Dette førte til at det var situasjoner hvor gruppen måtte vente på at strukturen ble ferdigstilt før arbeid kunne gå videre. Det skapte også en situasjon hvor det var vanskelig for gruppen å definere hvilke mål som var mulig å oppnå i løpet av tiden. Dette hadde Prosjekteier full forståelse for, men informerte gruppen om at det var bare å arbeide så langt vi kom, så skulle Styreportalen ordne resten.

6.3 Konsekvenser

Konsekvensene av endringer i plan, og utfordringer gruppen har møtt i løpet av arbeidet på prosjektet, anses som å være små. Da gruppen har arbeidet med en agil arbeidsmetode, er gruppen mer forberedt på endringer, og har kunnet tilpasse seg raskt.

6.3.1 Mangel på kunnskap

I intervju med Prosjekteier, se punkt 5.2.1, sier Prosjekteier at gruppen har oppnådd 95% ønsket funksjonalitet. Det viser seg dermed at selv med mangel på kunnskap ved oppstart av prosjektet, så har gruppen oppnådd et godt resultat. Resultatet har vært begrenset av at oppdragsgiver ikke har fått oppklart hvordan databasestrukturen for noen av funksjonalitetene skal være. Det er derimot mulig, at med mer kunnskap, og dermed mulighet for et bedre og raskere resultat i starten, at dette hadde fått oppdragsgiver til å fokusere mer på databasestrukturen som skal tas i bruk i prosjektet, og at gruppen dermed kunne utviklet mer.

6.3.2 Mellomledd

At gruppen måtte bytte mellom å ha et mellomledd og å koble klienten direkte mot databasen var en strek i regningen. Å integrere firebase direkte inn i klienten var nytt for gruppen. Arbeidet tok derimot ikke mer enn noen timer, og førte til en drastisk reduksjon i responstiden fra databasen, og førte dermed til økt brukeropplevelse.

6.3.3 Usikkerhet

Usikkerhet i prosjektet er den største utfordringen prosjektet har stått overfor. I løpet av prosjektet har det gjentatte ganger oppstått at gruppen enten ikke kan jobbe med en spesifikk oppgave, eller har måtte gjøre større endringer ved oppgaven på grunn av mangel på eksisterende struktur for funksjonaliteten. Dette har oppstått fordi ikke alle aspektene med produktet er ferdig planlagt. Som nevnt i 2.2.2, initielle krav, så var oppgaven delt inn i to deler. Gruppen kunne ikke starte på del to av oppgaven, da databasestrukturen mangler.

6.4 Resultat

Evalueringen av prosjektet er tidligere beskrevet i kapittel 5. Av informasjonen gruppen fikk ut av intervju med Prosjekteier, se 5.2.1, så er gruppens oppfatning at produktet har potensiale til å skape verdi for brukergruppen.

6.4.1 Brukertesting

I løpet av prosjektet har Prosjekteier stått for mye av brukertestingen, og gitt tilbakemeldinger til gruppen. Mot slutten av prosjektet har derimot brukere av den eksisterende løsningen fått teste produktet, og tilbakemeldingene har vært positive. Ved å ha konstant brukertesting anser gruppen at produktet er verdifullt for brukere, og at prosjektet er et godt resultat på brukerfronten.

6.4.2 Videreutvikling

Da prosjektet startet var det klart for gruppen at produktet ikke kom til å bli ferdigstilt til prosjektslutt. Det var derfor viktig at produktet ble utviklet med videreutvikling i tankene. Det har vært fokus på å holde lav kobling mellom moduler i applikasjonen, dokumentere koden, samt det var planlagt å skrive tester for viktige deler av systemet. Som nevnt i avsnitt 4.7, endte gruppen opp med å nedprioritere automatiserte tester, da det var mye å sette seg inn i, og liten tid. Gruppen håper derimot at prosjektet er i god stand til å videreutvikles ved prosjektslutt.

6.4.3 Prosjektansvarlig

Prosjekteier, som var prosjektansvarlig fra oppdragsgivers side, virker tilfreds med arbeidet gruppen har utført. I følge Prosjekteier, se intervju i avsnitt 5.2.1, har gruppen klart å opprette mer funksjonalitet enn originalt planlagt. Gruppen har oppnådd rundt 95% av kartlagt funksjonalitet. Etter positive tilbakemeldinger fra Prosjekteier, konkluderer gruppen med at gruppen har oppnådd et godt resultatet.

7 KONKLUSJON OG VIDERE ARBEID

Det har blitt laget en webportal der et medlem i en organisasjon logger inn på brukeren sin hos medlemssiden.no. Her kan medlemmet gå inn på organisasjonen å få opp et dokumentarkiv, medlemsliste, styreliste og informasjon organisasjonen har lagret om han/hun. Medlemmet kan få opp en informasjonsstrøm der organisasjonsstyret kan legge ut nyheter om organisasjoner. Medlemmet kan se en oversikt over aktiviteter(treninger, øvinger o.l) Og kan melde om han eller hun kommer til å møte opp eller ikke. Medlemmet kan se prosjekter, med egen informasjonsstrøm, liste over filer og noter.

Funksjonaliteten som er beskrevet over er ferdig implementert. Det eneste som manglet fra del én av de initielle kravene er en kontaktside. Dette er, på tidspunktet dette blir skrevet, fortsatt usikkert hvordan oppdragsgiver vil ha det, og har derfor ikke blitt implementert enda.

Del to av de initielle kravene besto av støtte for egendefinerte menyer for organisasjonene, dokumentarkiv og oppgaver for et medlem for en organisasjon. Av disse kravene ble kun dokumentarkiv implementert.

Målet med dette prosjektet var ikke at alle av de initielle kravene skulle bli implementert, men at det heller skulle bli gjort så mye som gruppen rakk. Styreportalen AS tar over utviklingen etter prosjektslutt og vil videreutvikle applikasjonen. Gruppen vil si seg fornøyd med arbeidet som ble gjort, siden de aller fleste kravene ble oppfylt.

Gruppens anbefalinger for videre arbeid er å utvikle egendefinerte menyer. I nåværende iterasjon får man opp noter for et prosjekt. Dette gir ikke mening for fotballag o.l. Oppdragsgiver vil utvikle støtte for foreldre, også foreldre med flere barn i flere organisasjoner(se 5.2).

Problemet dette produktet løser er ganske spisset inn for organisasjoner. Mange andre grupper kan likevel ha noe nytte av produktet. Store vennegrupper som skal på sydentur kan bruke informasjonsstrømmen til å spre informasjon og kalenderen til å holde oversikt over viktige datoer. Det finnes dog bedre og gratise løsninger for sånt, f.eks Facebook. En arbeidsplass kan bruke informasjonsstrømmen til å formidle nyheter.

8 REFERANSER

- Altexsoft(2018) The Good and the Bad of ReactJS and React Native [Internett] Tilgjengelig fra: https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-reactjs-and-react-native/ [Lest 22. april 2020]
- Styreportalen(2020) Enkel drift av lag og foreninger [Internett] Tilgjengelig fra: https://www.styreportalen.no/nb/ [Lest 15. mars 2020]
- Styreweb(2017) ENKLERE FORENINGSLIV MED STYREWEB [Internett] TIlgjengelig fra: https://www.styreweb.com/ [Lest 2. april 2020]
- Styreweb(2017) Funksjoner [Internett] TIlgjengelig fra: https://www.styreweb.com/funksjoner/ [Lest 2. april 2020]
- Redux (2020) Redux [Internett]. Tilgjengelig fra: https://redux.js.org/ [Lest 1. mai 2020].
- Wikipedia(2020) Agile Unified Process [Internett] Tilgjengelig fra: https://en.wikipedia.org/wiki/Agile_Unified_Process [Lest 28. april 2020]
- Visual studio code (2020) Getting Started [Internett] Tilgjengelig fra: https://code.visualstudio.com/docs [Lest 08. mai 2020]
- Medtek Norge(2020) Bransjeorganisasjonen for helse- og velferdsteknologi [Internett] Tilgjengelig fra: https://medteknorge.no/mot-en-ny-digital-hverdag/ [Lest 20 mai. 2020]

9 APPENDIX

9.1 Risikoliste

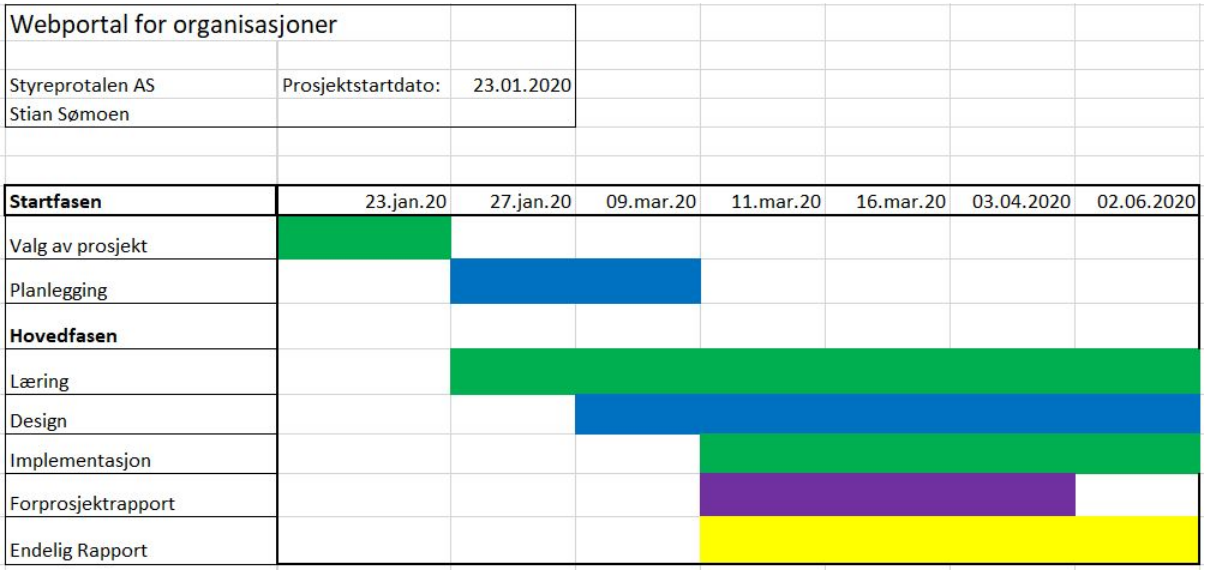
fra 1 til 5

Risiko #	Beskrivelse	Sannsynlighet	Konsekvens	Alvorlighetsgrad	Tiltak
1	Databasevert går ned	1	2	2	Vi kan ikke kontrollere om firebase går ned. Skulle det skje er det ikke veldig vanskelig å bytte database
2	Mangel på teknologikunnskap	3	4	12	Lær kontinuerlig. Spør og lær bort til hverandre.
3	Internett går ned midlertidig	1	1	1	Hent ut kodebasen hyppig. Jobb offline.
4	Mangel på oppgaveforståelse	2	5	10	holde møter med oppdragsgiver ofte.

5	Sykdom	2	3	6	Skulle en på gruppen bli syk i en kort periode bør det ikke være et problem for de to andre å gjøre en ekstra innsats slik at vi holder tidsplanen. I tillegg skal alle gruppemedlemmene laste opp koden sin til github etter hver arbeidsøkt sånn at de andre medlemmene kan fortsette på arbeide til en annen hvis de blir syk.

9.2 GANTT diagram

Initial Gantt diagram



Gantt diagram

	Milepæler	Kategori	Tilordnet til	Fremdrift	Start	Slutt
Startfase						
	Valg av prosjekt	Fullført	Gruppe 16	100 %	10.02.2020	23.02.2020
	Planlegging	Fullført	Gruppe 16	100 %	23.02.2020	06.03.2020
	Lære krevet teknologi	Går etter plan	Gruppe 16	75 %	23.02.2020	02.06.2020
Hovedfase						
	Prototype med innlogging	Fullført	Petter Knudsen	100 %	10.03.2020	16.03.2020
	Koble klient med database	Fullført	Gruppe 16	100 %	10.03.2020	22.04.2020
	Informasjonsstrøm	fullført	Petter Knudsen	100 %	13.03.2020	07.04.2020
	Prosjektside	fullført	Ole Martinus Rambech	100 %	16.03.2020	21.04.2020
	Organisasjonsside	Fullført	Stefan Go Thuen	100 %	07.04.2020	11.05.2020
	Kalender	Fullført	Petter Knudsen	100 %	30.03.2020	08.04.2020
	Forprosjektspresentasjon/rapport	Fullført	Gruppe 16	100 %	07.04.2020	23.04.2020
	Fakturaside	Fullført	Petter Knudsen	100 %	27.04.2020	08.05.2020
	Endelig rapport	Fullført	Gruppe 16	100 %	06.04.2020	02.06.2020