Høgskulen på Vestlandet

BACHELOROPPGAVE

Smart publiseringsløsning av nettsider til Styreportalen AS

Smart content management system to Styreportalen AS

Jørn Indrebø – 571733

Marcus Korsnes Morlandstø – 571540

Jostein Rådmannsøy Digernes - 571542

Dataingeniør/Informasjonsteknologi

Fakultet for ingeniør- og naturvitenskap

Institutt for datateknologi, elektroteknologi og realfag

Innleveringsdato: 01.06.2020

Jeg bekrefter at arbeidet er selvstendig utarbeidet, og at referanser/kildehenvisninger til alle kilder som er brukt i arbeidet er oppgitt, jf. Forskrift om studium og eksamen ved Høgskulen på Vestlandet, § 10.

TITTELSIDE FOR HOVEDPROSJEKT

Rapportens tittel: Smart publiseringsløsning av nettsider til Styreportalen AS Smart content management system to Styreportalen AS	*Dato: 01.06.2020*
Forfattere: Jørn Indrebø, Marcus Korsnes Morlandstø og Jostein Rådmannsøy Digernes	*Antall sider u/vedlegg: 46* *Antall sider vedlegg: 3*
Studieretning: Dataingeniør/Informasjonsteknologi	*Antall disketter/CD-er:* Ingen
Kontaktperson ved studieretning: Tosin Daniel Oyetoyan	*Gradering:* Ingen
Merknader: Ingen	

Oppdragsgiver: Styreportalen AS	*Oppdragsgivers referanse:* Ingen
Oppdragsgivers kontaktperson: Stian Sømoen	*Telefon:* +47 91319131

Sammendrag:

Prosjektet går ut på å lage en prototype for en publiseringsløsning, med fokus på backend-delen. Sammen med et brukergrensesnitt skal disse kunne generere nettsider, på en enkel og intuitiv måte. Prototypen vil senere bli brukt som utgangspunkt når Styreportalen AS skal utvikle en komplett publiseringsløsning.

Stikkord:

JavaScript	Firebase	Publiseringsløsning

Høgskolen på Vestlandet, Fakultet for ingeniør- og naturvitenskap
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 dokumenterer bachelorprosjektet «P10 - Publiseringsløsning A.» Rapporten beskriver prosessen med å utvikle en prototype for en publiseringsløsning som ble laget til oppdragsgiveren Styreportalen AS. Prosjektet er gjennomført av Jørn Indrebø, Marcus Korsnes Morlandstø og Jostein Rådmannsøy Digernes.

Vi ønsker å rette en stor takk til Stian Sømoen fra Styreportalen AS, som har vært vår eksterne veileder i dette prosjektet. Stian har vært lett tilgjengelig for oss gjennom hele prosjektperioden, og har kommet med gode innspill hele veien.

Vi ønsker også å takke vår interne veileder Tosin Daniel Oyetoyan for gode råd gjennom prosjektet og utmerket veiledning i forbindelse med rapporten.

INNHOLDSFORTEGNELSE

FIGURLISTE

TABELLISTE

1 INNLEDNING

1.1 Motivasjon og mål

Det er viktig å ha en tilstedeværelse på nett for organisasjoner, store og små, både for å kunne legge ut informasjon til medlemmer, publisere arrangement, dele bilder og mer. Dette kan være vanskelig for en del organisasjoner som mangler kunnskap om nettsider og hvordan man lager dem. En løsning for dette, er å ta i bruk en publiseringsløsning for nettsider.

En publiseringsløsning, eller mer kjent på engelsk som «content management system» er en programvare som hjelper brukere å opprette egne nettsider og administrere innholdet uten behov for spesialisert teknisk kunnskap. Publiseringsløsninger håndterer den grunnleggende infrastrukturen slik at brukerne kan fokusere på innholdet og utseende (Kinsta, 2020). Det finnes allerede publiseringsløsninger som gjør det lettere å lage nettsider uten behov for forkunnskaper om koding, men skal man ha en moderne og kompleks nettside kan en typisk publiseringsløsning fortsatt være mye å sette seg inn i.

Oppdragsgiveren Styreportalen AS tilbyr i dag sin egen publiseringsløsning for sine kunder, men ønsker en ny og forbedret løsning enn den de for øyeblikket tar i bruk. De har tenkt ut et konsept for en enklere og mer fleksibel publiseringsløsning som de ønsker å få realisert. Vår problemstilling er å lage en prototype av backend-delen for denne nye publiseringsløsningen. Målet er å lage en løsning som senere kan bli implementert i et brukergrensesnitt som tillater at brukeren kan opprette en profesjonell nettside med få, enkle handlinger.

1.2 Kontekst

Styreportalen AS utvikler og drifter sin egen portal for organisasjoner som ønsker å ha alt innhold relatert til sin organisasjon, samlet på ett sted. En av tjenestene som portalen tilbyr, er muligheten til å opprette en hjemmeside gjennom deres publiseringsløsning. Denne skal som sagt fornyes siden den løsningen de tilbyr nå, er kun en enkel og generell nettsidemal med få muligheter for personlige tilpasninger. Dette fører til at kundene som tar i bruk deres nåværende løsning, får mer eller mindre samme nettside. Selskapet ønsker å utvikle en bedre løsning som tilbyr mer fleksible endringer av temaer, innhold og utseende på nettsidene etter kundenes ønske. De har

også uttrykt seg om hvilke teknologier de ønsker å bygge den nye løsningen på. Hovedsakelig skal utviklingsplattformen Firebase bli brukt for databaselagring og håndtering av backend.

1.3 Begrensninger

Hvert gruppemedlem har begrensninger når det kommer til kunnskap om temaet. Ingen i gruppen har laget noe liknende tidligere, så det er mye å lære. Tid er også en begrensning med tanke på den begrensede tidsrammen prosjektet skal gjennomføres i. Kommunikasjon og samarbeid mellom gruppemedlemmer vil også være en utfordring siden Covid-19 pandemien hindrer fysiske møter. Dette fører til en uvant måte å jobbe på, og krever alternative samarbeidsløsninger fra gruppemedlemmene.

1.4 Ressurser

Styreportalen AS er tilgjengelig for veiledning gjennom hele prosjektperioden. Gruppens interne veileder fra Høgskolen på Vestlandet, Tosin Daniel Oyetoyan vil også kunne gi oss veiledning knyttet til gjennomføring av prosjektet.

For å utvikle prototypen så trenger gruppen forskjellige verktøy. Styreportalen har gitt oss tilgang til Firebase sin betalte versjon av tjenesten, «Blaze Plan» og vi vil bruke produktene som er tilgjengelig på denne utviklingsplattformen. Styreportalen AS har og kjøpt inn pakker med forskjellige nettsidemaler som skal brukes som grunnlag for nettsidene i publiseringsløsningen. Disse har de og gjort tilgjengelige for oss.

Versjonskontroll er også viktig, og for dette tar vi i bruk Github. For koding er Visual Studio Code valgt, da dette er gruppens foretrukne utviklingsmiljø når det kommer til HTML/CSS/JavaScript. Det er og mye nytt som gruppen i utgangspunktet har lite kjennskap til. Offisiell dokumentasjon og opplæringsvideoer til de forskjellige teknologiene vil derfor også være nyttige ressurser.

2 PROSJEKBESKRIVELSE

2.1 Praktisk bakgrunn

2.1.1 Prosjekteier

Styreportalen AS er et firma med lokaler på Nesttun i Bergen som utvikler, drifter og selger digitale produkter til diverse organisasjoner. Deres tre hovedprodukter er Styreportalen.no, Program.no og Billett.no. Program.no benyttes av kulturorganisasjoner til planlegging og gjennomføring av konkurranser og konserter. Billett.no er for salg av billetter til konserter og arrangementer. Styreportalen.no er et administrasjonssystem for organisasjonene der de kan ha oversikt over sin daglig drift, dokumentering av alt arbeid som blir gjort, samt medlemsregistrering og kommunikasjon mellom dem. Det er også her organisasjonene har mulighet til å opprette sin egen hjemmenettside gjennom deres publiseringsløsning. Stian Sømoen er leder for Styreportalen AS og er vår kontaktperson for prosjektet.

2.1.2 Tidligere arbeid

Publiseringsløsningen som Styreportalen AS bruker for øyeblikket består av én statisk nettsidemal og er basert på publiseringsverktøyet Joomla. Med denne løsningen krever hver nettside en egen database. Kunden har mulighet til å endre tekst-innhold enkelte steder, laste opp logo, lenke til eventuelle sosiale medier og legge ut enkle innlegg med tekst og bilder. Strukturen og utseende ellers vil være det samme, dermed vil nettsidene se relativt lik ut mellom de forskjellige organisasjonene. Styreportalen har et ønske om at nettsidene skal være mer dynamiske ved å tilby kundene en mer fleksibel publiseringsløsning enn det som er tilfellet i dag. Dette skal brukeren selv kunne gjøre på en enkel og brukervennlig måte der ingen koding kreves. Den nye løsningen eksisterer bare som et konsept og skal lages fra bunnen av. Helt i starten av prosjektet hadde Styreportalen AS klargjort enkelte ting for oss, slik at vi kunne komme relativt raskt i gang med utviklingen av prototypen. Vi skulle redigere pakken med nettsidemaler og integrere de sammen med backend. I tillegg hadde de opprettet et Firebase-prosjekt som de ga oss tilgang til. Dette skulle bli brukt for database-lagring og testing av de ulike funksjonene som ble utviklet. Videre var det opp til gruppen å lære teknologiene og finne gode løsninger underveis for å realisere dette konseptet..

2.1.3 Initielle krav

Styreportalen AS stilte spesifikke krav til teknologiene de ville at vi skulle bruke, men hadde derimot få konkrete krav når det kom til funksjonaliteter. Kravene kan oppsummeres i to punkter:

- Ta i bruk Firebase til lagring, hosting og utplassering av funksjoner.
- Ta utgangspunkt i nettsidemalene deres.

Det er ikke et krav i vår oppgave å lage et brukergrensesnitt til denne publiseringsløsningen, da Styreportalen AS hadde en annen bacheloroppgave som handlet om dette. Denne oppgaven ble derimot ikke valgt, så Styreportalen holdt det åpent for at vi kunne utvikle et brukergrensesnitt om det var tid.

Ved hjelp av backend-løsningen som gruppen skal lage, skal oppdragsgiver i fremtiden kunne utvikle et brukervennlig og intuitivt brukergrensesnitt. I backend-løsningen må vi derfor ta hensyn til at brukeren skal kunne ta utgangspunkt i nettsidemalene til Styreportalen AS. Ved hjelp av disse malene, skal brukeren kunne ha mulighet til å gjøre endringer på innhold og utseende, legge til og slette innhold gjennom brukergrensesnittet. Det var foretrukket at så mange funksjonaliteter som mulig, blir lagret som Cloud Functions i Firebase.

2.1.4 Initiell løsnings-idé

Styreportalen AS hadde allerede et forslag for hvordan innholdet kunne bli lagret, så det var opp til oss å se på dette og videre finne de mest intuitive løsningene. Den initielle løsnings-ideen går ut på å bruke de ferdige nettsidemalene som grunnlag for brukernes nettsider. Hver nettsidemal skal brytes ned i flere logiske deler, kalt blokker og hver blokk skal fungere som en av flere byggeklosser som brukeren kan velge å ha med på sin nettside. Hver blokk har et unikt innhold og utseende som brukeren skal kunne endre på gjennom et brukergrensesnitt.

Måten brukeren skal jobbe mot dette, er at det først skal velges en ønsket mal. Videre velger brukeren hvilke blokker som skal være med, fyller inn innhold og redigerer utseende via valgene som er tilgjengelige fra brukergrensesnittet. Alt innhold lagres i Firebase, og det skal være det eneste som trengs å hentes for å få fram brukerens versjon av nettsidemalen. Funksjonene som modifiserer innholdet fra den valgte malen lagres i Cloud Functions. Etter at brukeren har gjort alle valgene sine og lastet opp dataen, skal det genereres en ny ferdigstilt versjon av nettsidemalen basert på brukerens valg.

Etter at brukeren har gjort alle valgene sine og lastet opp dataen, skal det genereres en ny ferdigstilt versjon av nettsidemalen basert på brukerens valg.

2.2 Litteratur om problemstillingen

Litteraturen vi har brukt i prosjektet har vært mest for å opparbeide oss forståelse for hvordan teknologien vi skal bruke fungerer. Dette ble gjort ved å lese dokumentasjon fra utviklerne av teknologien, og opplæringsvideoer for å se hvordan det brukes i praksis.

Det finnes massevis av eksisterende publiseringsløsninger med ulike formål og som bygger på ulike teknologier. Mange publiseringsløsninger tilbyr store verktøykasser for design av nettsider, organisering av innhold og publisering på nett. Et kjent eksempel er WordPress som har som mål å lage en publiseringsløsning som kan brukes av alle, og som samtidig tilbyr kraftige funksjoner (Wordpress, 2020). WordPress har åpen kildekode, og er skrevet i PHP. Den bruker MySQL- eller MariaDB-database. Til sammenlikning skal vår løsning skrives i JavaScript og bruke Cloud Firestore databasen til lagring av data.

Enduro.js er en annen publiseringsløsning som bygger på liknende teknologier som oss. Den bruker node.js og bygger på en flat-file database. Den virker mer rettet mot utviklere eller personer som har erfaring med nettsideutvikling, i motsetning til vår løsning, som er mer rettet mot uerfarne brukere. Enduro.js har også åpen kildekode, som ligger offentlig tilgjengelig på Github (Enduro.js, 2020).

3 DESIGN AV PROSJEKTET

3.1 Forslag til løsning

Den første tenkte løsningen går ut på å lagre alt innhold som skal vises på den endelige nettsiden i Firebase. Dette innebærer at både HTML-kode til nettsidemaler, tilhørende skripter og CSS, samt brukerens innhold skal lagres her. Cloud Firestore databasen deles opp i to samlinger, en for nettsidemalene og en for brukerinnholdet på nettsidene. Nettsidemalene som oppdragsgiveren har gitt blir altså delt opp i blokker med HTML-kode. Disse blokkene blir lagret i egne dokumenter i Firestore. Alt som brukeren skal kunne endre på til sin egen nettside skal byttes ut med midlertidige plassholdere for innhold. Dette gjelder for eksempel overskrifter, tekstavsnitt, referanser til bilder, og annen type tekst. Brukerens valg om hvilken mal, hvilke blokker de skal ha med og hva som skal stå i de ulike blokkene blir lagret i én samling.

Løsningen vår for å knytte disse sammen er å bruke Firebase sine Cloud Functions. Her laster vi opp funksjoner skrevet med JavaScript i et Node.js miljø, som skal kjøres når brukeren setter opp nettsiden sin. De skal hente data fra begge samlingene i Firestore og knytte dem sammen til en nettside.

En slik løsning som bruker Cloud Firestore og Cloud Functions var først foreslått av oppdragsgiver i starten av prosjektperioden. Vi ble gitt et forslag til fremgangsmåte, som inkluderte hvilken teknologi vi skulle ta i bruk. Gruppen vurderte dette som en god og gjennomførbar løsning.

3.1.1 Valgt løsning

Siden oppdragsgiver hadde et krav om hvordan den helhetlige løsningen burde utføres, så kom det ikke fram noen spesielt bedre forslag for hvordan vi kunne løst det basert på teknologiene. Dermed ble det valgt å følge løsningen som ble beskrevet over i delkapittel 3.1. Ingen alternative løsninger til Firebase ble vurdert, da dette er en del av oppgaven.

3.2 Valg av verktøy

I dette kapittelet vil vi nevne verktøyene som blir tatt i bruk i løsningen. Det forklares i detalj hva de ulike verktøyene er og hvordan de blir brukt i oppgaven.

3.2.1 Firebase

Google Firebase er en utviklingsplattform for bygging av mobil- og webapplikasjoner. Plattformen tilbyr skybaserte sanntidsdatabaser, forskjellige API-er, flere autentiseringstyper og muligheter for hosting av applikasjonene på nett (Tutorials Point, 2020d). Firebase spiller en sentral rolle for den nye publiseringsløsningen og gruppen vil ta i bruk en del av Firebase sine tilgjengelige teknologier i prosjektet.

3.2.1.1 Cloud Firestore

Cloud Firestore er en sky basert NoSQL database der data blir lagret som JSON og blir synkronisert og oppdatert i sanntid for alle klienter som er tilkoblet. Det er også mulighet for offline-støtte med lokal datalagring. Cloud Firestore bygger på Firebases originale database, Realtime Database med en ny og mer intuitiv datamodell. Sammenlignet med Realtime Database som kun lagrer all data i en stor JSON tre-struktur, så lagrer Cloud Firestore data i forskjellige dokumenter som blir organisert i samlinger. Samlingene fungerer som beholdere for disse dokumentene. Dokumenter kan igjen inneholde samlinger under seg. Dette gjør at Cloud Firestore krever mindre denormalisering og utflating av data, som gjør databasen mer fleksibel og skalerbar. På grunn av denne hierarkiske strukturen kan man bruke spørringer for henting av individuelle, spesifikke dokumenter eller for å hente alle dokumentene i en spesifikk samling (Firebase, 2020a). Vi vil bruke denne databasen i løsningen vår som spesifisert i oppdragsgiverens krav.

3.2.1.2 Firebase Storage

Storage tilbyr lagring av filer. Ved bruk av Firebase sin SDK kan man laste opp og laste ned filer direkte fra klienten, mens behandlingen blir utført fra server-siden. Det er også integrert en deklarativ sikkerhetsmodell for å tillate tilgang basert på filnavn, størrelse, innholdstype og andre metadata (Firebase, 2020b). Dette vil blant annet bli tatt i bruk for lagring av alt fil-innhold som brukeren vil ha med på nettsidene sine.

3.2.1.3 Firebase Hosting

Firebase Hosting tilbyr rask og sikker hosting av mobil- eller webapplikasjoner. Dette gjelder for både statisk og dynamisk innhold. Det er dette Styreportalen AS skal bruke for å drifte kundene sine nettsider. Det er også mulig å koble Firebase Hosting sammen med Cloud Functions og ta i bruk dens opprettede funksjoner i tillegg (Firebase, 2020c).

3.2.1.4 Cloud Functions

Firebase sine Cloud Functions gir muligheter for kjøring av forskjellig backend-kode som kan brukes som respons på hendelser som blir utløst av Firebase-funksjoner eller HTTP-forespørsler. Koden forblir lagret på Google sin sky og kjører i et administrert miljø (Firebase, 2020d).

3.2.2 Visual Studio Code

Visual Studio Code er et kraftig utviklingsmiljø der man kan skrive bl.a. web-, mobil- og skyapplikasjoner. Vi benytter dette verktøyet primært for all koden som skal skrives i prosjektet. Verktøyet kommer med innebygd støtte for JavaScript og Node.js, og har en del tilgjengelige utvidelser som kan komme til nytte (Microsoft, 2020).

3.2.3 Github

Vi bruker GitHub for å samle kode vi skriver i forbindelse med prosjektet. GitHub er en kode-vert plattform for samarbeid og versjonskontroll. Med GitHub kan man samarbeide parallelt i prosjekter (W3Schools, 2020).

3.2.4 HTML og CSS

HTML og CSS er to av kjerneteknologiene for oppbygging av nettsider. HTML er et markeringsspråk som beskriver semantikk og struktur på en nettside (Tutorials Point, 2020c), mens CSS er mer et designspråk som kan gi nettsider et mer visuelt utseende. Med CSS kan man bl.a. tilpasse oppsett, farger, fonter, bilder og lignende (Tutorials Point, 2020b). Nettsidemalene som skal kunne endres på av brukeren krever at vi kan litt om HTML og CSS. Å lage det offisielle brukergrensesnittet til publiseringsløsningen er ikke en del av oppgaven, men vi trenger uansett å lage et enkelt brukergrensesnitt som kan brukes for testing av ulike funksjoner.

3.2.5 JavaScript

JavaScript er et programmeringsspråk som støtter objektorientert og funksjonell programmering. Språket blir mye brukt i sammenheng med HTML og CSS for utvikling av webapplikasjoner. JavaScript gir mulighet for å oppdatere, manipulere og validere data. Språket kan også brukes for å endre innhold i HTML- og CSS-filer, som er essensielt for oppgaven som skal løses (Tutorials Point, 2020a).

3.2.6 Node.js

Node.js er en åpen kildekode plattform som gir et hendelsesdrevet, ikke-blokkerende I/O- og kryssplattform-kjøreomgivelse. Det blir brukt bl.a. for å bygge skalerbare applikasjoner på serversiden ved hjelp av JavaScript. Node.js brukes hovedsakelig for å bygge nettverksprogrammer som webservere (TutorialsTeacher, 2020). Node.js vil bli brukt når vi skal skrive Cloud Functions.

3.3 Prosjektmetodikk

3.3.1 Utviklingsmetodikk

I vårt første møte med oppdragsgiver ble vi enige om å utvikle en løsning for oppgaven med en smidig tilnærmingsmetodikk. Gruppen har valgt å benytte seg av en Scrum-tilnærming. Scrum-rammeverket baserer seg blant annet på utvikling og leveranser i korte iterasjoner med fast lengde, selvstyrte team, fokus på høy kvalitet og kontinuerlig læring (Schwaber & Sutherland, 2017).

Gruppen har valgt å løse prosjektet på en inkrementell måte. Løsningen ble utviklet i iterasjoner, der målet for hver iterasjon har vært å få til funksjonalitet som i stor grad kan benyttes i prototypen. Vi tar små kunnskapsmessige skritt for å bli bedre for hver iterasjon. Gruppen fokuserer på funksjonalitetene som er viktigst for oppdragsgiver. At de er på plass og fungerer som planlagt er viktig for oppdragsgiver med tanke på å bygge videre på prosjektet i fremtiden. Vi har ukentlige møter med oppdragsgiver der vi får vurdert funksjonaliteten som er gjort. Under møtene blir det klargjort om de ulike funksjonalitetene kan anses som ferdig, eller om det trengs en ny sprint. Møtene bidrar også til at vi hele tiden får informasjon om vi er på rett vei. Det er også opprettet en Trello-vegg som gruppen bruker i prosjektperioden.

Scrum baserer seg som sagt på selvstyrte team, og kommunikasjon innad i teamet står her sentralt. På grunn av Covid-19 har vi hatt et stort fokus på dette punktet. Siden gruppen ikke kan møtes fysisk, utføres det korte møter via kommunikasjonsplattformen Discord nesten hver eneste dag. Det blir og skrevet en logg hvor hvert gruppemedlem oppsummerer hva de har gjort med hensyn på prosjektet hver eneste dag. Dette er med på å bidra til at alle gruppemedlemmer er oppdaterte på hva som må gjøres og eventuelt om noen i gruppen har støtt på problemer som vedkommende trenger hjelp til å løse.

3.3.2 Prosjektplan

For at gruppen skal klare å fullføre prosjektet på en effektiv måte, så er det laget en plan for hva som må gjøres for å nå prosjektets mål. Dette ble gjort i form av et GANTT diagram (se vedlegg 9.3). Denne planen kan bidra til å gi gruppen en oversikt underveis i prosjektperioden, ved å vise om gruppen er i rute med gjøremålene som er satt for å fullføre prosjektet innen planlagt tid. Planen viser hvor mye tid vi planlegger å bruke på hver aktivitet.

GANTT diagrammet er delt opp i tre deler. I delen «Prosjektstart» gikk vi gjennom bacheloroppgavene vi kunne bli tildelt, og satt deretter opp en prioritert liste over hvilke oppgaver vi helst kunne tenke oss. Det var også i denne delen vi faktisk fikk tildelt oppgaven, og hadde våre første møter med oppdragsgiver og veileder. Vår kontaktperson, Stian Sømoen ga i korte trekk informasjon om hvordan han så for seg at oppgaven kunne løses. Han ga også informasjon om krav til løsningen som måtte være med, hvilke forventninger han hadde til oss og hva vi kunne forvente fra han.

I delen «Oppgaver fra Høgskolen på Vestlandet» finner man diverse obligatoriske gjøremål gitt av høgskolen i forbindelse med faget. Punktet «Diverse mindre oppgaver» er oppgaver som å skrive referat fra møter, registrere blogg, registrere prosjekttittel og lignende.

Den siste delen er «Oppgaver fra oppdragsgiver». Her får man oversikt over aktiviteter knyttet til oppgavene fra oppdragsgiveren. Gruppen var usikker på hvor mye tid hver aktivitet ville ta, men etter en diskusjon kom vi fram til forslaget i vedlegg 9.2, som senere ble revidert i vedlegg 9.3. Vi kommer som sagt til å jobbe iterativt med oppgaven fra oppdragsgiver. Vi har delt den opp i flere deloppgaver, som vi vil forsøke å løse en etter en. Vi får også tilbakemelding fra oppdragsgiver hver uke på løsningene vi har laget. Disse tilbakemeldingene, inkludert resultater fra testing, er avgjørende for når vi går videre til neste deloppgave. I kapittel 4 vil vi gå nærmere inn på punktene «Fungerende prototype for nettsidemal del 1» og «Fungerende prototype for

nettsidemal del 2» fra GANTT-diagrammet. Det er hovedsakelig disse oppgavene vi har forsøkt å løse i flere iterasjoner.

3.3.3 Risikovurdering

Vi regner med å støte på risikoer i løpet av prosjektet. Å vite om mulige risikoer, sannsynligheten for at de inntreffer og konsekvensene av dette, er viktig for å minimere effekten det kan ha på prosjektets fremgang. Ved å kjenne til mulige risikoer kan vi tidlig forebygge for at de ikke skal inntreffe eller ha handlingsplaner klare dersom en risiko skulle inntreffe.

Gruppen har valgt å bruke et risikoskjema for å avdekke hvilke risikoer vi eventuelt kan støte på (se vedlegg 9.1). Skjemaet er inndelt i risikokategoriene «Planlegging», «Kommunikasjon», «Kompetanse» og «Produkt». Skjemaet sier også noe om sannsynligheten (P) for at risikoen inntreffer, hvor kritisk risikoen er (C) og angir en risikofaktor (P*C). «P» og «C» har en skala fra 1 til 5. Hver risiko i risikoskjemaet inneholder også en tiltaksplan.

3.4 Evalueringsplan

Under evaluering av resultatet av oppgaven, vil det først og fremst testes om alt har blitt riktig implementert. Vi vil ta i bruk enhetstesting hvis vi får tid til det.

Vi ser ikke for oss at vi skal få ekte sluttbrukere til å teste applikasjonen. Dette er fordi brukergrensesnitt og brukeropplevelse ikke er en direkte del av oppgaven. Om vi skulle ha laget et brukergrensesnitt ment for sluttbrukere ville det naturligvis vært nyttig å få direkte tilbakemeldinger fra dem.

I prosjektet er det forskjellige moduler som skal snakke med hverandre, for eksempel Firestore-databasen og brukergrensesnittet. Derfor vil vi også gjennomføre integrasjonstesting.

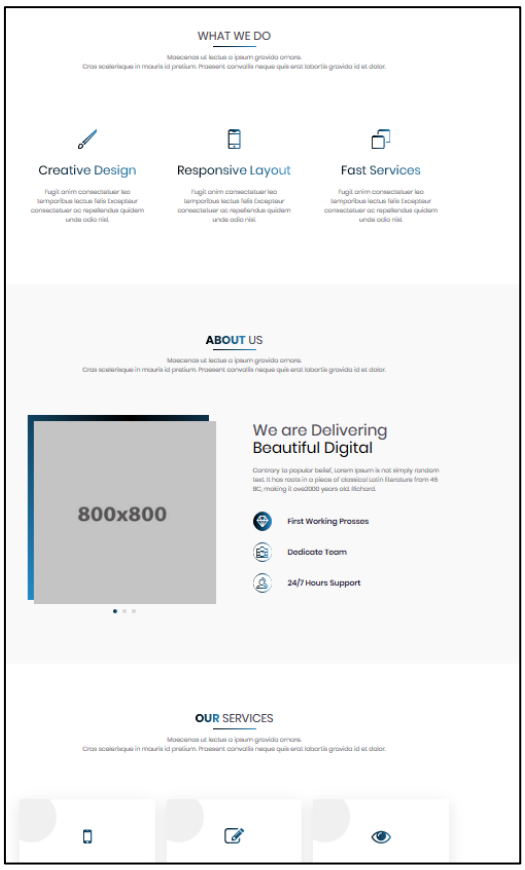
Det er viktig at løsningen vår tilfredsstiller kravene til oppdragsgiver. Her vil vi passe på å ha hyppige, ukentlige møter med ekstern veileder hvor vi presenterer arbeidet vårt og hva vi har tenkt. Slik får vi tilbakemeldinger på om dette stemmer med kravene. I slutten av prosjekttiden planlegger vi å gjennomføre en brukerakseptansetest med prosjekteier, hvor vi får vite om han er fornøyd med løsningen vår.

4 DETALJERT DESIGN

I dette kapittelet vil vi snakke om hvordan vi har gått frem for å løse oppgaven. Først vil vi gi en mer detaljert forklaring av konseptet som baserer seg på én utvalgt nettsidemal. Videre vil det bli gitt en beskrivelse av utviklingsprosessen der vi har benyttet oss av en iterativ tilnærming for å løse oppgaven. Videre vil det vises frem en konseptuell modell av det fullstendige prosjektet, før vi går mer i detalj på de ulike funksjonalitetene og komponentene.

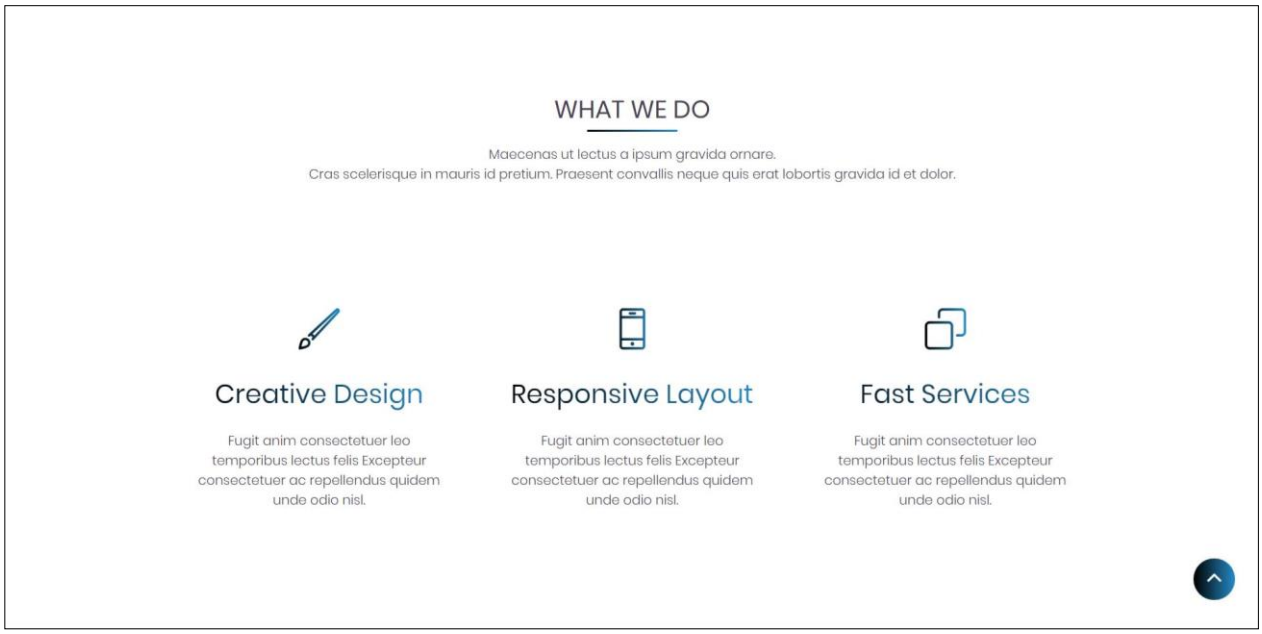
4.1 Innledning

Som tidligere nevnt, har Styreportalen AS kjøpt inn pakker med ferdige nettsidemaler. Deres ønske er å lage en prototype der disse malene benyttes i løsningen som et grunnlag. Malen vi tar utgangspunkt i er en ensideapplikasjon (single-page application) og består av en del filer med HTML, CSS og skripter.



Figur 4-1 Utsnitt av nettsidemal.

Nettsiden er delt opp i blokker og inneholder ulike komponenter som er naturlig å ha på en hjemmeside. På figur 4-1 kan man se ett utsnitt av nettsidemalen vi har tatt utgangspunkt i. Her kan man se eksempler på hvordan disse blokkene ser ut og hvordan de ligger etter hverandre. På figur 4-2 lengre nede ser man ett eksempel på det som vi definerer som én blokk.



Figur 4-2 Eksempel på hvordan innholdet i én av blokkene ser ut

Denne blokken med overskrift «What we do» består av et tekstavsnitt, tre bokser med ett ikon, én overskrift og ett tekstavsnitt hver. I publiseringsløsningen skal en bruker kunne velge hva som skal stå av tekst, ikoner og kanskje velge å legge til flere «bokser» med ikon, overskrift og tekst.

Totalt består vår utvalgte prototype-mal av 15 forskjellige kodeblokker. Oppgaven vår er å lage en så dynamisk publiseringsløsning som mulig innenfor grensene som denne malen tilbyr. Brukeren skal kunne velge fritt blant de 15 kodeblokkene, fylle inn egen input og få generert en nettside uten at brukeren kan ødelegge malen.

4.2 Oppbygging og utviklingsprosess for løsningen

4.2.1 Utviklingsprosess

For å finne riktige løsninger i prosjektet valgte gruppen en iterativ utviklingsprosess. Vi delte opp prosjektet i iterasjoner, hvor målet for hver iterasjon var å levere en fungerende del av systemet, et inkrement. Vi har hatt regelmessige leveranser (inkrementer) til oppdragsgiver, og utviklingen har gått stegvis der nye kravspesifikasjoner fra oppdragsgiver har kommet frem etter hvert. Vi hadde ikke en fullstendig liste med kravspesifikasjoner ved oppstart av prosjektet, men nye krav har blitt definert underveis, ofte etter vurderinger fra oppdragsgiver av det som ble levert tidligere.

Diagrammene under viser iterasjoner vi har vært igjennom i forbindelse med punktene «Fungerende prototype for nettsidemal del 1» og «Fungerende prototype for nettsidemal del 2» i GANTT-diagrammet. Selv om begge disse punktene tar for seg den samme nettsidemalen, følte vi at det var nødvendig å dele den opp i to oppgaver. I den førstnevnte oppgaven var det forventet at det ville være mye fokus på å lære seg selve teknologien som skulle benyttes, og det ble forventet at det ville bli mye prøving og feiling. Før den andre oppgaven var det forventet at vi hadde lært mye fra førstnevnte oppgave, og at fokuset i mye større grad nå kunne rettes mot å ferdigstille en fullstendig prototype. Hver iterasjon varte i en uke. «Oppgavebeskrivelse» i diagrammene tar for seg krav som har vokst frem underveis i prosjektperioden og som i stor grad har vært nødvendige for å kunne løse det initielle kravet om å få til en velfungerende publiseringsløsning.

Hoved målet for det førstnevnte diagrammet, «Fungerende prototype for nettsidemal del 1», var å klare å få generert en nettside ved hjelp av HTML-strenger lagret i Firestore. Dette involverte mye prøving og feiling, i tillegg til massiv læring av Firebase som var teknologi vi ikke var kjent med fra før. Vi valgte å konsentrere oss om en kodeblokk vi kalte "Img_full" for å få dette til. Denne kodeblokken ble hentet fra nettsidemalen som vi etter hvert skulle lage en fullstendig prototype for. I denne kodeblokken burde bruker kunne laste opp et bilde, man burde kunne linke til en video, og man måtte fylle ut noen tekstfelter. I tillegg til hovedmålet fokuserte vi på å få satt i gang noen førsteutkast til en del andre oppgaver. Brukergrensesnittet som ble laget for denne deloppgaven skulle inneholde lite, men tilstrekkelig med funksjonalitet. Å få i stand et brukergrensesnitt og en løsning for å generere en nettside ved hjelp av HTML-lagret i Firestore ble for oss minimumskravene i denne oppgaven før vi kunne bevege oss videre til «Fungerende prototype for nettsidemal del 2». Vi gikk videre til del 2 når følgende funksjonalitet og testing var tilfredsstilt:

1. Bruker går inn på en nettside og får opp et brukergrensesnitt.
2. Fra en meny skal bruker kunne velge kodeblokken «Img_full» fra en nedtrekksmeny.
3. Bruker skal få opp input-felter for denne blokken.
4. Bruker sin input skal kunne lastes opp til Firestore.
5. HTML-kode og input fra bruker som begge er lagret i Firestore skal kunne benyttes av Cloud Function til å generere en nettside når bruker går inn på en spesifikk URL.

Oppgavebeskrivelser	Iterasjon 1	Iterasjon 2	Iterasjon 3	Iterasjon 4
Dele HTML-mal opp i passende kodeblokker, strukturere HTML-koden i blokkene og lagre til Firestore. (FR 3)	Påbegynt	Ferdig	Ferdig	Gjorde noen små endringer i strukturering av HTML-kode i Firebase. Ferdig
Cloud Function for generering av nettside. (FR 4)	Påbegynt	Ferdig	Ferdig	Ferdig
Brukergrensesnitt for testing av kodeblokker. (FR 5, FR 6)		Påbegynt	Sannsynligvis snart ferdig. Kan laste oppe tekststrenger. Kan ikke laste opp bilder	Ferdig. Bruker kan nå laste opp bilder
Funksjon for opplasting av bilder til Firebase Storage (FR 1.1)			Påbegynt og sannsynligvis snart ferdig	Ferdig
Struktur for lagring av input fra brukergrensesnitt til Firestore fra én forhåndsbestemt kodeblokk.			Påbegynt og sannsynligvis snart ferdig	Ferdig
Testing ved bruk av brukergrensesnitt og funksjon for generering av nettside. (FR 1, FR 2)		Generering av nettside med Cloud Function på eksempeldata fra Firestore. Brukergrensesnitt er ikke tatt i bruk	Generering av nettside med Cloud Functions på data som blir lagret i Firestore etter input fra bruker	Generering av nettside med Cloud Functions på data som blir lagret i Firestore etter input fra bruker
Status etter testing		Cloud Function for generering av nettside ser ut til å fungere	Brukergrensesnitt klarer å hente nødvendig informasjon fra data-struktur som er hentet fra Firestore. Input-data fra brukergrensesnitt blir også lagret som forventet i Firestore. Generering av nettside ser ut til å fungere	Brukergrensesnitt klarer å hente nødvendig informasjon fra data-struktur som er hentet fra Firestore. Input-data fra brukergrensesnitt blir også lagret som forventet i Firestore. Dette gjelder også for bilder som bruker har lastet opp. Generering av nettside fungerer som ønsket
Status og diskusjon rundt arbeidet som ble gjort under hver iterasjon	Ingen merknader	Arbeidet med brukergrensesnittet går dårlig. Forkaster arbeidet som er gjort med dette, og forsøker med en annen tilnærming	Arbeidet går bra. I neste uke er det først og fremst brukergrensesnittet og en bilde-funksjon vi må arbeide med for at vi skal nå målet for denne seksjonen	Målet er nådd. Vi klarer nå å få generert en nettside ved hjelp av brukerinput og mal-fil. Vi vil bygge videre på dette senere i prosjektet. Videre ser vi for oss at vi må strukturere HTML-koden som ligger i kodeblokkene i Firebase annerledes, slik at bruker kan lage en enda mer dynamisk nettside

Tabell 4-1 Iterasjoner i «Fungerende prototype for nettsidemal 1».

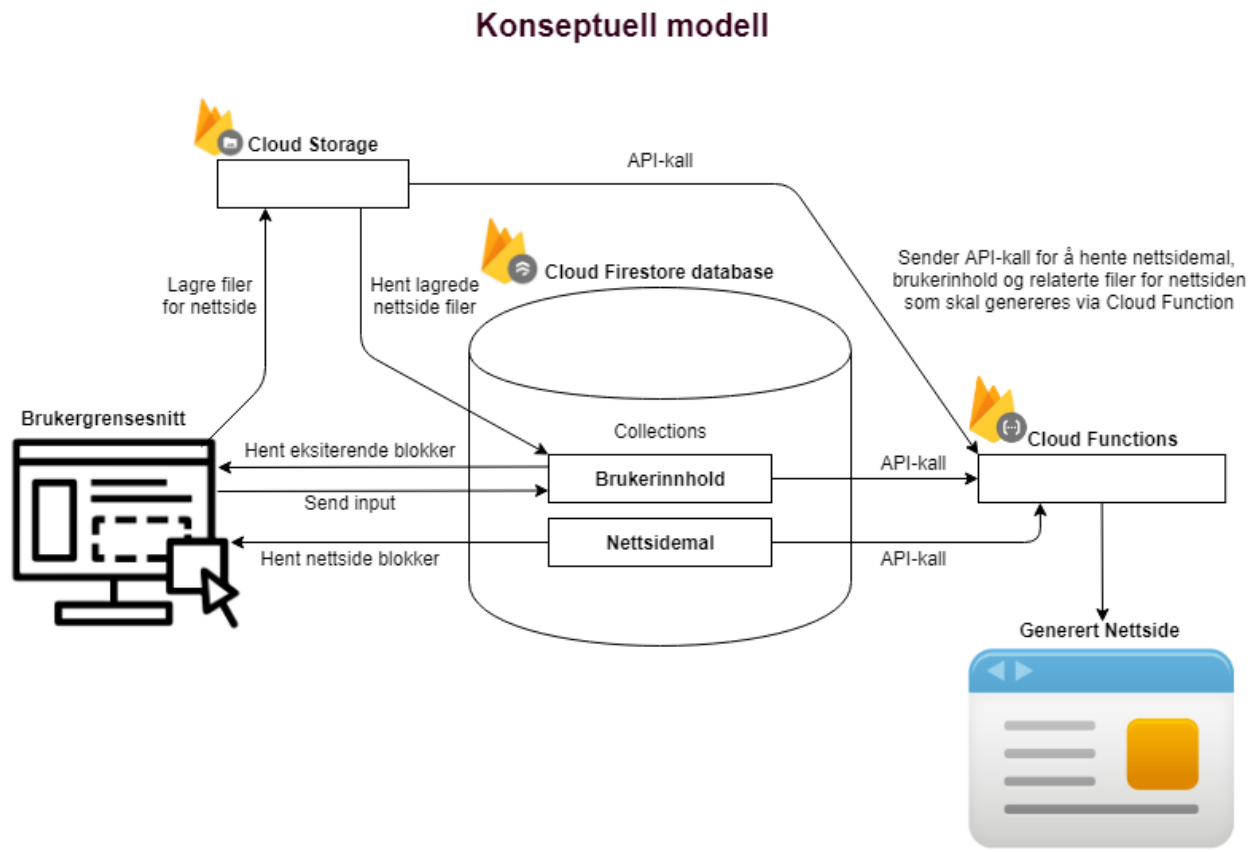
Diagrammet for «Fungerende prototype for nettsidemal del 2» beskriver iterasjonene i denne deloppgaven. Her bygget vi videre på det vi gjorde og lærte i deloppgaven «Fungerende prototype for nettsidemal del 1». Når målene for del 1 var nådd, hadde vi tilegnet oss tilstrekkelig kunnskap for å implementere løsningen i del 1 til alle kodeblokker i den fullstendige nettsidemalen. Målet for denne oppgaven var å få til en fullstendig prototype for nettsidemalen vi hadde tatt utgangspunkt i. Vi arbeidet også med å utvide brukergrensesnittet med mer funksjonalitet, og testet hyppig hvordan input fra brukergrensesnittet ble lagret i firebase, og hvordan resultatet av de genererte nettsidene ble.

Oppgavebeskrivelser	Iterasjon 1	Iterasjon 2	Iterasjon 3	Iterasjon 4	Iterasjon 5
Endre strukturen i kodeblokkene i Firestore for at bruker skal stå mer fritt til å velge deler av innholdet i en gitt kodeblokk	Gjøre det mulig for bruker å legge inn input på alle steder det bør være mulig å legge inn input	Det mangler fortsatt muligheter for input noen steder. Fortsetter med å forbedre dette	Input er ferdig. Nå er det fokus på blokkene der man kan velge ønsket antall med bilder, innlegg og diverse	Videre arbeid på oppgavene i 3. I tillegg til mer informativ informasjon til brukeren for hver blokk	Ferdig
Tilpasse Cloud Functions for generering av nettside	Påbegynt	Tilpasser funksjonen til ny lagringsstruktur i Firebase	Videre arbeid på oppgavene i iterasjon 2	Videre arbeid på oppgavene i iterasjon 2 og 3	Ferdig
Videreutvikle brukergrensesnitt for mer funksjonalitet.	Påbegynt	Påbegynt	Få til forhåndsvisning av blokkene i en nedtrekksmeny	Ferdiggjøre valg av ikoner og endingsmuligheter i navigasjonsbar	Ferdig. Gjorde og brukergrensesnittet mer brukervennlig og informativt
Mulighet for bruker til å kunne velge ikoner.			Påbegynt	Funnet løsning for valg av diverse typer ikoner	Ferdig
Endringer for lagring av brukerinput fra til databasen fra én kodeblokk.		Påbegynt	Påbegynt	Påbegynt	Ferdig
Testing ved bruk av brukergrensesnitt og generering av nettside.	Påbegynt	Pågående testing	Pågående testing	Påbegynt	Ferdig
Status og diskusjon rundt arbeidet som ble gjort under hver iterasjon.	Etter hvert som vi videreutvikler brukergrensesnittet ser vi at det er mange endringer vi kan, og bør gjøre, med tanke på struktur og innhold til de forskjellige kodeblokkene som er lagret i Firestore	Løsningen vi har valgt for at bruker skal ha mulighet for å velge bort deler fra èn kodeblokk ser ut til å fungere	Noen kodeblokker er vanskelige å endre på slik at bruker kan velge bort deler av innhold. Implementering av andre type tester har vist seg vanskelig å få til i denne iterasjonen. Oppdragsgiver ønsker også at brukergrensesnittet skal bli enda mer brukervennlig og informativt	Av 15 kodeblokker har vi 5 stykker igjen som vi må arbeide en mer med. Vi regner med å kunne gjenbruke løsninger som vi har implementert i andre kodeblokker en del steder. Det bør derfor gå forholdsvis fort. Vi har også kommet godt i gang med testing. Noen enhetstester er nå på plass	I denne iterasjonen fikk vi fullført alle oppgavene vi jobbet med, og fikk godkjent akseptansetester av oppdragsgiver

Tabell 4-2 Iterasjoner i «Fungerende prototype for nettsidemal 2»

4.2.2 Oppbygging

I forhold til oppbyggingen av løsningen har vi laget en konseptuell modell for å gi en enklere forståelse av hvordan komponentene i løsningen henger sammen, som man kan se i figur 4-3.



Figur 4-3 En konseptuell modell av hvordan de ulike komponentene i løsningen henger sammen

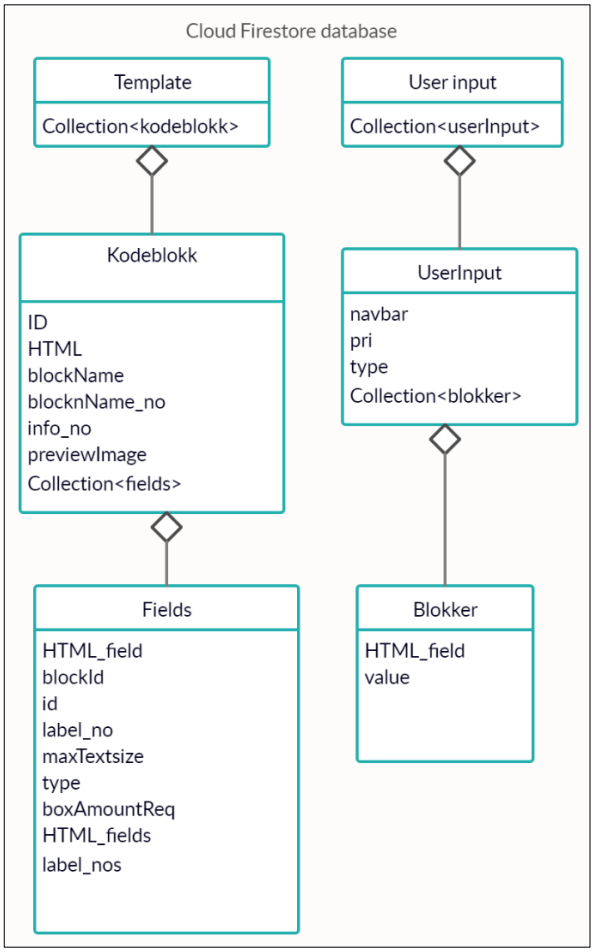
1. For at brukergrensesnittet skal vite hvilke kodeblokker som eksisterer, og hvilken informasjon som må fylles ut, utføres det en spørring mot den valgte «Nettsidemal»-samlingen, som ligger lagret i Cloud Firestore databasen. Denne datasamlingen inneholder all HTML-koden for brukerens utvalgte nettsidemal og er delt opp i flere unike kodeblokker.
2. I brukergrensesnittet velger brukeren kodeblokk(er) fra «Nettsidemal», fyller inn nødvendig informasjon og laster opp brukerinputen til Firestore som «Brukerinnhold». Dersom det er filer involvert som for eksempel bilder, blir disse lagret i Firebase Storage mens referansene til dem lagres i «Brukerinnhold»-samlingen i Firestore. Hvis brukeren allerede har opprettet en nettside, så kan de eksisterende blokkene hentes fra Firestore og endres på eller bli slettet.
3. En funksjon i Cloud Functions benyttes for å generere nettsiden. Den sender API-kall til Firestore og henter data både fra «Nettsidemal» og fra «Brukerinnhold» for å generere nettsiden. Den henter også alle filer fra Storage som er relatert til brukerens nettside.
4. Når all data er hentet, så blir brukerens nettside blir generert gjennom den nevnte funksjonen.

4.3 Strukturering av samlinger i Firestore

Det ble vist i modellen i kapittel 4.2 at vi hadde to samlinger med data i Firestore, en for valgt nettsidemal og en for brukerinnhold. Det første viktige steget i prosjektet var å dele opp nettsidemalen og fylt opp denne samlingen i Firestore. Dette var en tidkrevende prosess, da vi manuelt delte opp nettsiden, samtidig som vi satt inn plassholdere hvor det skulle komme brukerinnhold.

Vi har en samling som inneholder én nettsidemal, hvor vi har delt opp koden i blokker og erstattet innholdet til en del felter i koden med plassholdere. Plassholderne skal så senere erstattes med input fra den andre samlingen, som inneholder data som brukeren har fylt inn i brukergrensesnittet og som har blitt lastet opp til Firestore.

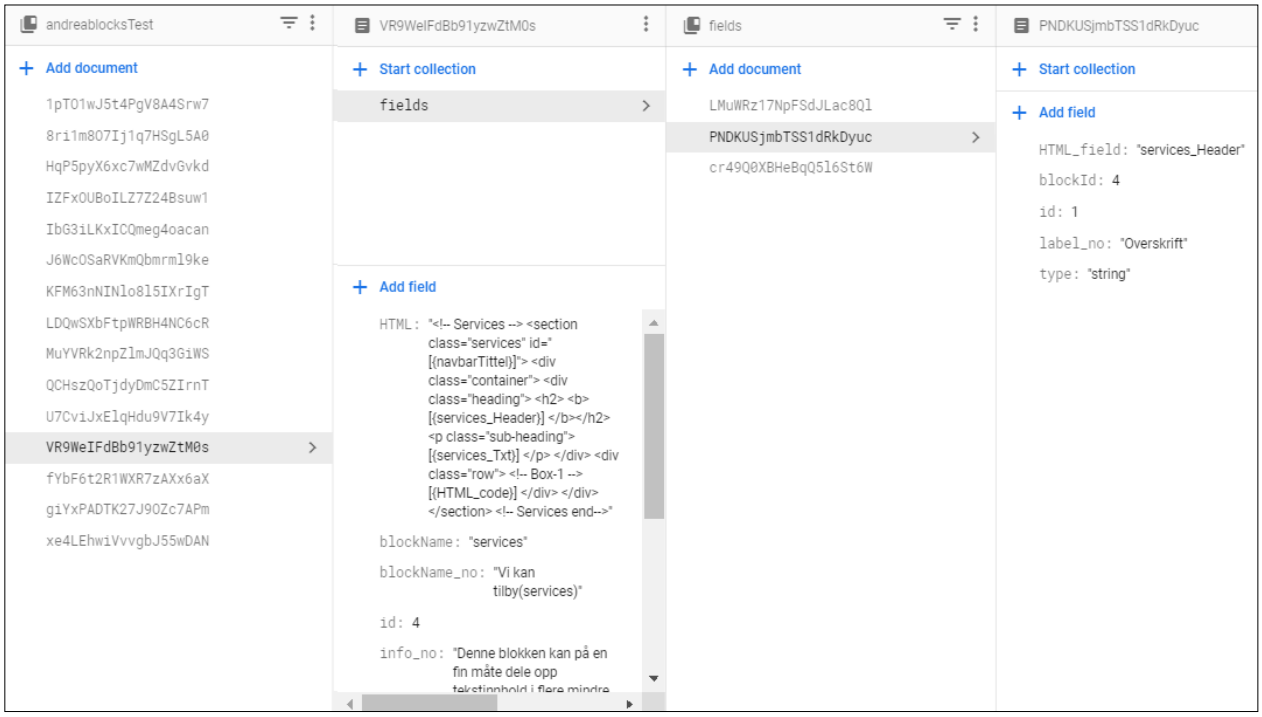
Samlingen som inneholder den modifiserte nettsidemalen, utgjør kjernen i prosjektet. Alle muligheter brukeren får i brukergrensesnittet baserer seg på funksjoner skrevet for brukergrensesnittet, som igjen har vært mulig å skrive på grunn av strukturen til denne samlingen. Vi har underveis i prosjektet måtte skrive om denne strukturen flere ganger, for å kunne inkludere flere nødvendige funksjoner og krav.



Figur 4-4 Et mer detaljert UML-diagram av strukturen

I figur 4-4 kan man se et UML-diagram som viser struktur og innhold for de to samlingene. Hvilke attributter i dokumentet «Fields» som blir brukt, varierer mellom de ulike type plassholderne.

På figur 4-5 ser man hvordan strukturen er bygd opp, med fokus på kodeblokken som vi har valgt å kalle «services». Til venstre ser man en samling med navn «andreablocksTest». Dette er samlingen til nettsidemalen vi har benyttet i dette tilfellet. Her finner man alle de 15 ulike kodeblokkene som brukeren kan velge mellom i publiseringsløsningen, lagret som dokumenter. Videre til høyre ser man en undersamling kalt «fields». Innholdet her avgjør inputen som brukeren blir bedt om å fylle inn, dersom vedkommende velger den gitte kodeblokken. Feltene i «fields» tilsvarer plassholderne i nettsidemalen.



Figur 4-5 Skjermbilde av hvordan nettsidemalen er lagret i Firestore.

Det skilles mellom ulike typer i «fields». Disse typene forklares i Tabell 4-1 nedenfor. De enkleste typene er tekst og tekstavsnitt. Dette er felter på nettsiden som skal fylles med ren tekst. En annen type som er litt mer kompleks er «HTML-kjede». Felter av denne typen består av definerte moduler med egne felter. Det velges hvor mange moduler som skal komme etter hverandre. Felter som har typen «HTML-URL» er også valgfrie. Det vil si at hvis brukeren lar input-feltet for denne typen stå tomt, vil de ikke bli inkludert når nettsiden genereres. Felter med andre typer enn de overnevnte vil være obligatoriske, med unntak av det valgfrie input-feltet hvor brukeren kan inkludere en kodeblokk i startmenyen sin.

Type plassholder	Beskrivelse
Tekst, tekstavsnitt	Ren tekst. F.eks. Overskrifter, avsnitt. Skiller mellom korte tekstinnhold og lengre tekster.
Bilde, video	Bilde- eller video-referanse.
Ikon	Referanse til ikon. Ikonreferansene er definert i et CSS stilark som er knyttet til den genererte nettsiden.
HTML-URL	En enkel HMTL-streng. F.eks. En knapp til sosiale medier, med egendefinert lenke.
HTML-kjede	HTML-streng, som inneholder moduler som kan «kjedes». Altså bruker kan legge til flere slike moduler etter hverandre, og ha ulikt innhold i hver av dem. F.eks. «Bokser med ikon, overskrift og tekst» fra figur 4-2.

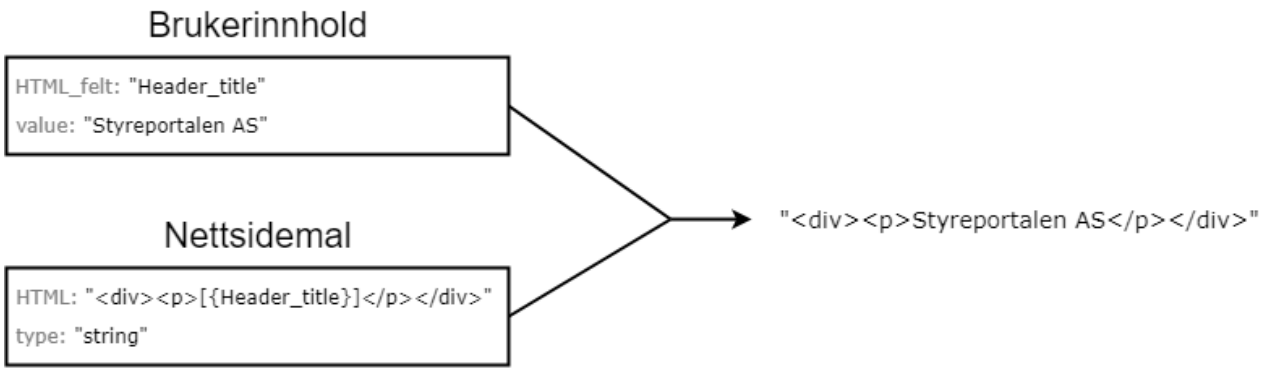
Tabell 4-3 En oversikt over de ulike typene plassholderne/feltene kan ha.

Samlingen for den modifiserte nettsidemalen lastes opp til Firestore som én batch via et skript. Dette har gjort det enklere å gjøre endringer på strukturen underveis i prosjektet. Samlingen for input-data er laget manuelt i Firestore, men fylles opp med data når bruker laster opp input fra brukergrensesnittet.

4.4 Generering av nettside

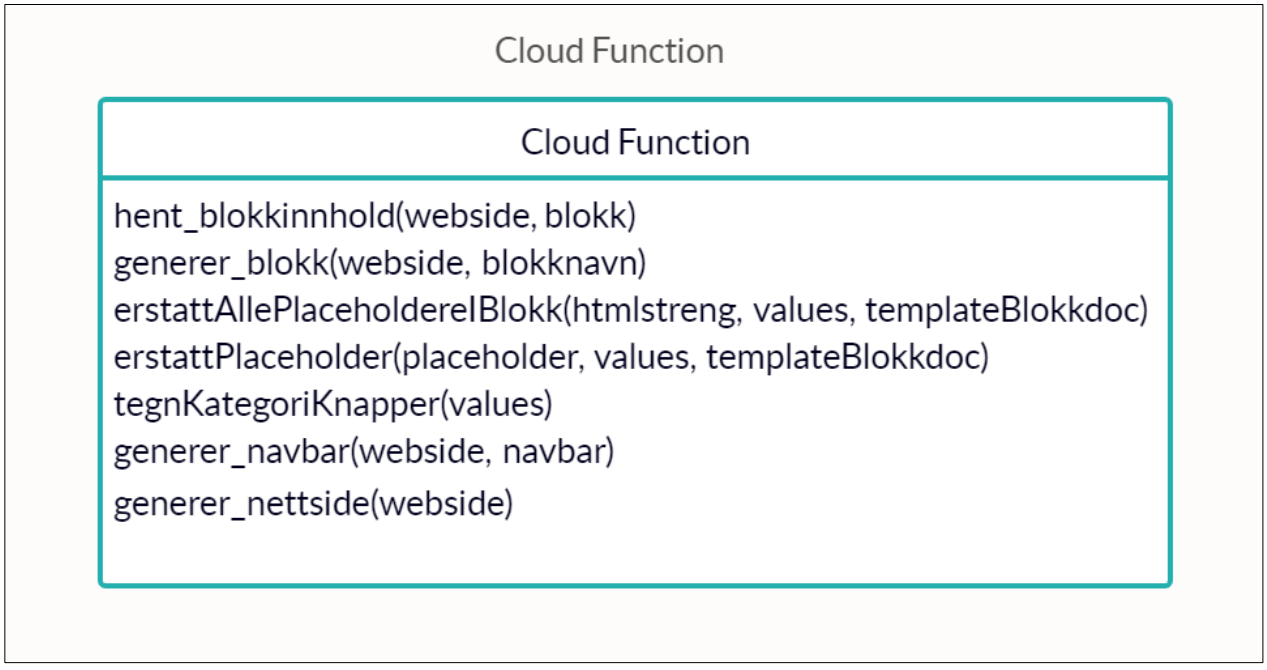
Etter å ha fått til en logisk strukturering av dataen i Firestore, så var det neste steget å lage en funksjon som klarte å generere en nettside ved hjelp av de to samlingene. Den ene samlingen inneholdt altså HTML-koden for hver blokk til en utvalgt mal, hvor bl.a. tekstinnhold var erstattet med plassholdere. Den andre samlingen inneholdt input-data fra brukeren, som skulle legges inn i plassholderne. Begge samlingene blir lastet opp til Firestore manuelt.

Nettsiden genereres ved hjelp av en funksjon som henter data fra de to samlingene. De er lagret i Firestore med lik struktur som gjør sammenkoblingen enklere, som beskrevet i forrige delkapittel. Plassholderne i samlingen med nettsidemalen tilsvarer feltene i samlingen med brukerinnholdet. Nettsidemalen blir først hentet fra databasen, så blir plassholderne erstattet med brukerinnholdet og til slutt blir den nye HTML-strengen returnert. Figur 4-6 viser en visuell illustrasjon av dette.



Figur 4-6 Illustrerer erstatningen av en plassholder når de to Firestore-samlingene kobles sammen til én HTML-streng.

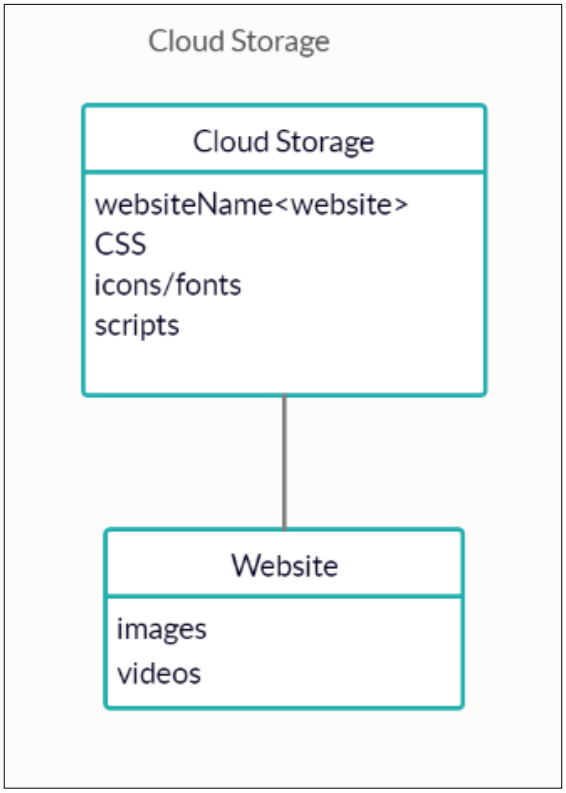
Dette blir altså gjort for alle plassholderne i hver kodeblokk som blir endret på. Kodeblokkene blir så slått sammen til en hel nettside. Siden nettsidemalen vi bruker i prototypen er en ensideapplikasjon, så følger blokkene naturlig etter hverandre. Selve genereringen skjer når man kjører funksjonen som ligger i Cloud Functions. Denne blir trigget når det blir gjort en HTTP-forespørsel med nettsidens navn som parameter. Den ferdiggenererte nettsiden blir returnert i HTTP-responsen som en HTML-streng og så fremvist i nettleseren. Med denne løsningen betyr det at nettsiden ikke blir hostet noe sted, men blir generert hver gang HTTP-forespørselen blir gjort. Løsningen fungerer fint for prototypen, men dette bør gjøres på en annen måte i den endelige publiseringsløsningen. Dette blir diskutert nærmere i konklusjonen i kapittel 7.



Figur 4-7 Oversikt over funksjoner som blir tatt i bruk i Cloud Functions

4.5 Lagring av filer i Firebase Storage

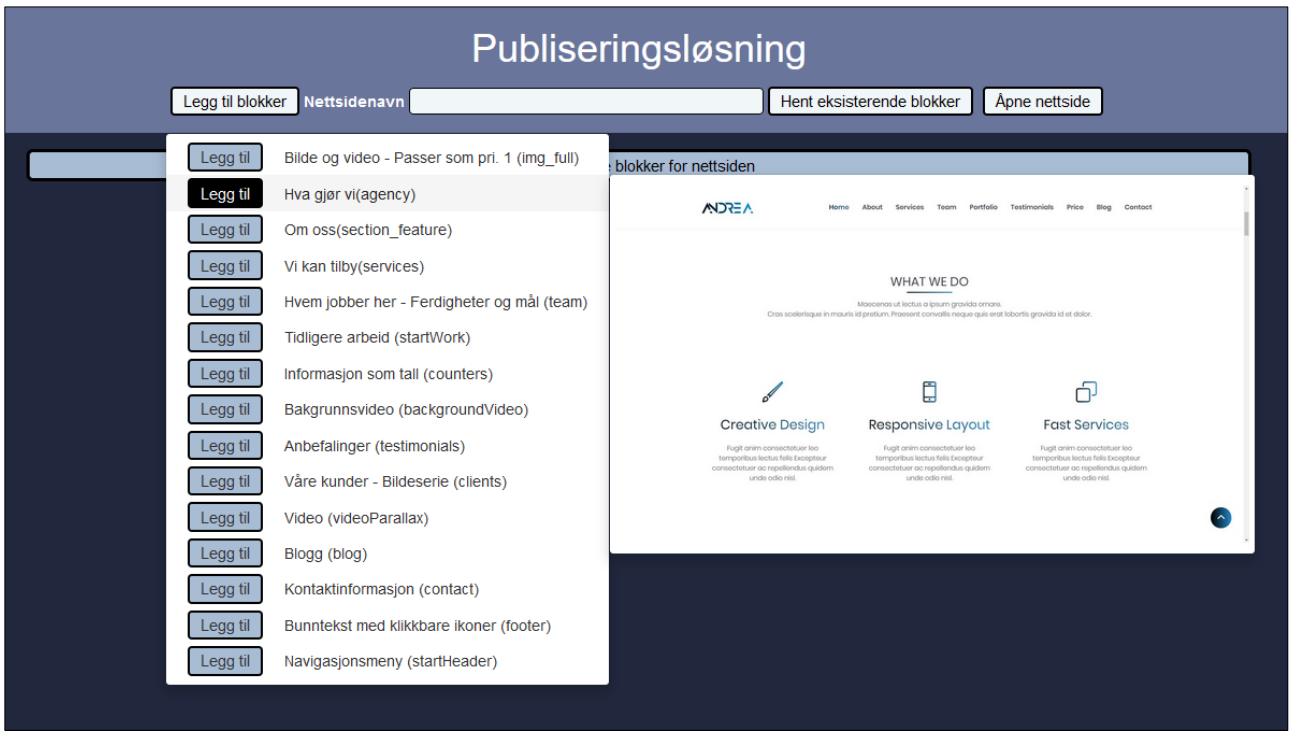
Firebase Storage ble først tatt i bruk når vi skulle gjøre det mulig for en bruker å laste opp ønskede bilder og videoer til nettsiden sin. Det var flere steder i nettsidemalen hvor det var naturlig at brukeren burde ha mulighet til det. Løsningen ble at bilder og videoer ble lagret i Storage, mens lenken til dem ble lagret som input i brukerinnhold-samlingen. Mer spesifikt lages det en mappe for nettsiden hvor den spesifikke nettsidens bilder og videoer lagres. Etter hvert lastet vi også opp filer som var knyttet til den utvalgte nettsidemalen på Firebase Storage. Dette inkluderte bl.a. CSS-filer, JavaScript-filer og forskjellige fonter/ikoner. Disse filene blir referert til i den ferdiggenererte nettsiden.



Figur 4-8 Fil-typer som befinner seg i Storage

4.6 Brukergrensesnitt

Ganske tidlig i utviklingsperioden merket gruppen behovet for å lage et brukergrensesnitt. Å lage et brukergrensesnitt var ikke et krav til selve oppgaven, men ble likevel sett på som nødvendig for å enklere kunne utføre testing og vise frem resultater. For å få testet f.eks. funksjoner for opplasting av bilder og videoer ble det helt nødvendig å få dette på plass. Dette brukergrensesnittet var hovedsakelig laget for testing og er ikke ment for sluttbrukere å ta i bruk. Brukeropplevelse og lignende har dermed ikke vært et fokus.



Figur 4-9 Bilde av brukergrensesnittet. Her ser man nedtrekksmenyen med alle blokkene som kan velges for utvalgt mal

Brukergrensesnittet har bidratt til å gi gruppen verdifull informasjon om hvordan HTML-koden fra nettsidemalene burde struktureres, oppdeles og endres i Firestore-databasen. Uten et brukergrensesnitt så måtte også all brukerinput blitt ført inn manuelt i Firestore, for å så bli testet.

Brukergrensesnittet gjør CRUD operasjoner direkte mot Firestore, uten sanitering eller lignende sikkerhetstiltak. Figur 4-9 viser hvordan brukergrensesnittet ser ut. Som man ser, kan man velge alle blokker som er tilgjengelig i den utvalgte nettsidemalen i nedtrekksmenyen. Når man har musepekeren over ett av punktene, vil man få opp ett forhåndsvisningsbilde av den blokken. Bildet er ment som både inspirasjon og informasjon til brukeren for hvordan de kan benytte seg av blokken. Både menypunktene for hver blokk og de relaterte bildene er hentet fra Firestore. Menypunktene hentes med en «read»-operasjon til en samling som er lagret i Firestore, mens bildene hentes med en «read»-operasjon til en mappe i Firebase Storage der disse bildene er lagret. Knappen «Hent eksisterende blokker» kan benyttes hvis brukeren allerede har en eksisterende nettside opplastet, og vil gjøre endringer på noen av blokkene. Da lastes både kodeblokker og lagret input inn i brukergrensesnittet. Dette gjøres også ved hjelp av «read»-operasjoner mot Firestore. Figur 4-10 under viser et eksempel av brukergrensesnittet der brukeren har valgt en kodeblokk.

Figur 4-10 Bilde av brukergrensesnittet etter å ha valgt en blokk. Man får opp input-feltene som er spesifikt koblet til den blokken. Velger man andre blokker, blir de lagt til under med sine egne unike input-felter.

Her utførers det en «read»-operasjon for å få tak i nødvendige inputfelter som brukeren må fylle ut. I denne blokken er det implementert en løsning som gjør at brukeren selv kan velge å ha med «Avsnitt med ikon» eller ikke. Hvis det er ønsket å ha dette med så kan brukeren også velge ønsket antall av disse. Når brukeren er fornøyd med valg av kodeblokker og har fylt ut nødvendig informasjon, vil det det bli utført en «create»-operasjon når brukeren laster opp innholdet. Tekstinnhold blir lastet direkte opp til Firestore, mens videoer og bilder blir lastet opp til Storage og URL-lenken til dem lagres i Firestore. Dersom nettsiden allerede eksisterer, vil det i tillegg til «create»-operasjoner også bli utført «delete»-operasjoner på det eksisterende innholdet. De slettes istedenfor å oppdateres siden Firestore gir unike id-er til dokumenter, som er vanskelig å hente opp igjen. «Delete»-operasjonene utføres dersom en bruker velger å slette kodeblokker i brukergrensesnittet for en eksisterende nettside, òg i det bruker laster opp innhold. Da slettes alt innhold som fantes fra før, også blir alt lastet opp på nytt.

5 EVALUERING

5.1 Evalueringsmetode

5.1.1 Metoder brukt

Hovedmetodene brukt til testing i prosjektet vårt er integrasjonstesting og akseptansetesting. Akseptansetestingen ble gjennomført med prosjekteier i slutten av prosjektperioden. I tillegg til den endelige akseptansetesten, har vi som nevnt i tidligere kapitler gjennomført flere møter med prosjekteier gjennom prosjektets iterasjoner. Brukergrensesnittet vi har utviklet har blitt brukt for å gjennomføre de fleste av testtilfellene. Vi har ikke tatt i bruk enhetstesting. Vi har heller ikke gjennomført en brukertest, da dette ikke ble sett på som relevant for oppgaven.

Enhetstesting er en viktig form for testing i programvareutvikling. Det går ut på å teste de minste enhetene i programmet. Det er mange fordeler ved å bruke enhetstesting, bl.a. kan man lett finne feil som er knyttet til én spesifikk enhet, som fører til at det blir enkelt å fikse. Enheter testes isolert fra de andre enhetene, slik at testresultatet er uavhengig og påvirkes ikke av andre funksjonaliteter (Naik & Tripathy, 2008, p. 51). Enhetstesting gir også en trygghet når man gjør endringer i koden. Skriver man dekkende nok enhetstester kan man hele tiden under utvikling kunne svare på om produktet fungerer eller ikke.

Planen vår var å skrive enhetstester hvis vi fikk tid. På grunn av kort tidsramme i prosjektet ble dette ikke prioritert, og vi har ikke tatt i bruk enhetstester. Istedenfor har testingen av funksjonaliteter skjedd samtidig som skrivingen av koden. Vi har skrevet kode, kjørt den og undersøkt om den fungerer som ønsket. Om noen feil oppsto ble de rettet opp. Denne løsningen er ingen fullkommen erstatning for enhetstesting, men til gjengjeld gir det en god flyt i arbeidet og vil fortsatt gi et fungerende resultat uten avgjørende feil.

5.1.2 Integrasjonstesting

Integrasjonstesting går ut på å teste de forskjellige modulene for å validere at de fungerer ordentlig sammen i det komplette systemet (IEEE, 2010, p. 181). Vårt prosjekt består av en rekke ulike komponenter. Brukergrensesnittet brukes som et verktøy for å gjennomføre integrasjonstesten. Vi bruker dette for å se om vi får kommunisert med Firestore fra web-klienten.

Vi vil også teste om funksjonen i Cloud Functions som skal generere nettsiden får kontakt med databasen og genererer nettsiden med databaseinnholdet.

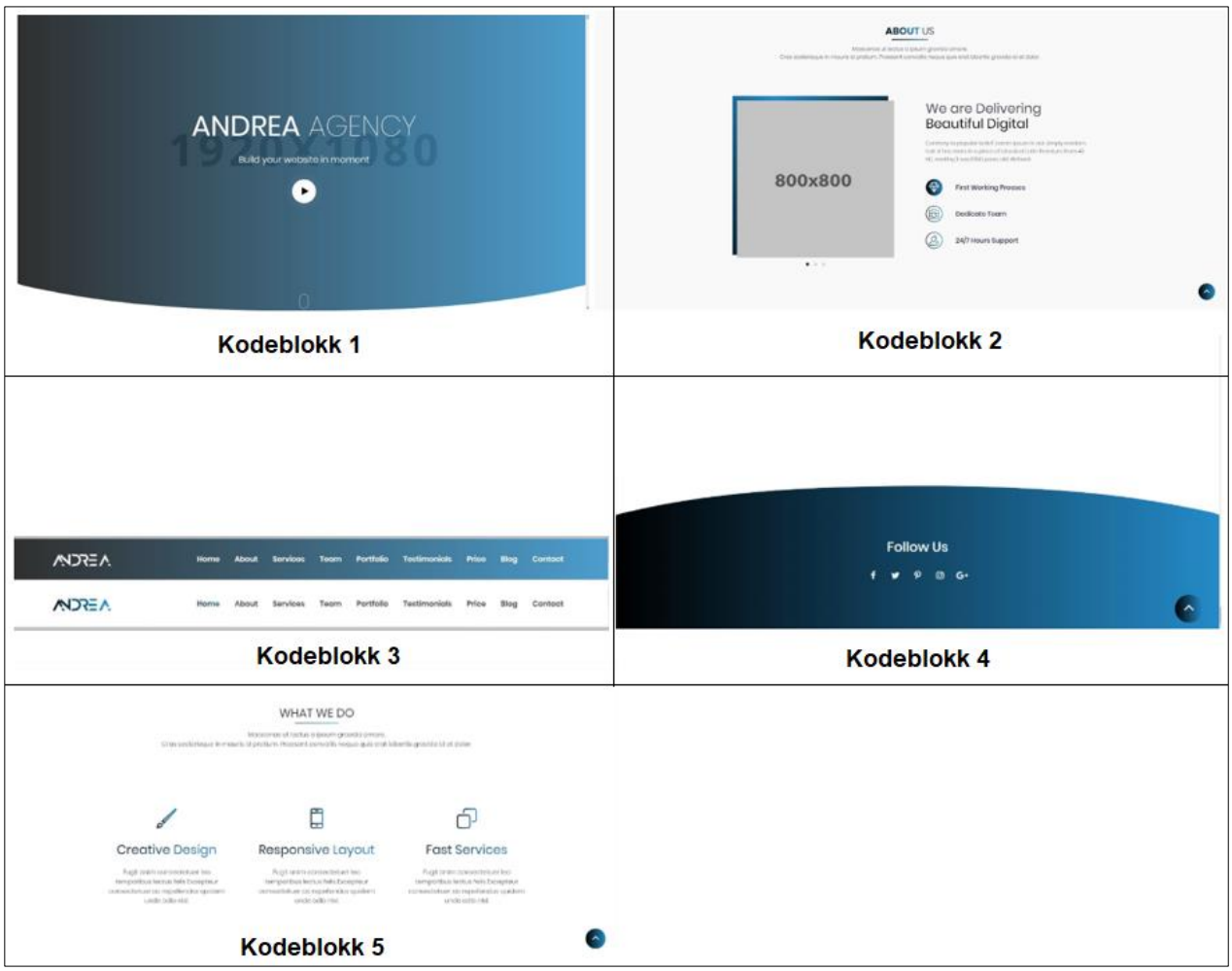
#	FR #	Testtilfelle	Beskrivelse	Forventet resultat
1	FR 1	Verifiser koblingen mellom brukergrensesnittet og databasen.	1. Last opp til nettsiden «Test» 2. Velg blokk kalt «img_full» 3. Fyll inn felter med overskrift, undertittel og bilde. 4. Last opp blokk.	Opprettes et nytt dokument med navn «Test». Den skal ha en subcollection med dokument av type «img_full» som igjen har dokumenter for feltene med overskrift, undertittel og bilde.
1.1	FR 1.1	Verifiser at bilder blir lastet opp til Storage.		Valgt bilde blir lastet opp til Firebase Storage under mappen til nettsiden «Test».
2	FR 2	Verifiser koblingen mellom funksjonen i Cloud Functions og databasen.	Kjør funksjonen i Cloud Functions ved å utføre en HTTP-forespørsel.	Nettside genereres med innhold hentet fra databasen.

Tabell 5-1 Testtilfeller, integrasjonstesting. «FR» er funksjonelle krav, og refererer til Tabell 4-1.

5.1.3 Akseptansetesting

Akseptansetesting er en form for formell testing med hensyn på bl.a. brukerbehov og brukerkrav som utføres for å avklare om et system oppfyller kravene eller ikke (Norwegian Testing Board, 2016, p. 7). I vårt prosjekt ble det utført en brukerakseptansetest sammen med prosjekteier for å avklare om vår løsning tilfredsstilte deres ønsker. Vi definerte testtilfeller som vi i gruppen gikk gjennom og utførte sammen med prosjekteier på et videomøte. Prosjekteier skulle så godkjenne eller ikke godkjenne testresultatet og komme med eventuelle kommentarer. Dette ble utført på slutten av prosjekttiden. Det er også verdt å nevne at gjennom ukentlige statusmøter med prosjekteier fikk vi demonstrert funksjonaliteter underveis og fikk tilbakemeldinger på dem.

Vi har satt opp en oversikt over fem testtilfeller som vi har gjennomgått med prosjekteier. Disse testtilfellene er basert på hvilke krav som er blitt til under utviklingen og på prosjekteierens initielle ønsker og krav. Akseptansetestene er utført med utgangspunkt i fem av 15 kodeblokker. Under kan man se et bilde av de utvalgte kodeblokkene og en tabell med testtilfeller. Testtilfellene blir ytterligere beskrevet under tabellen.



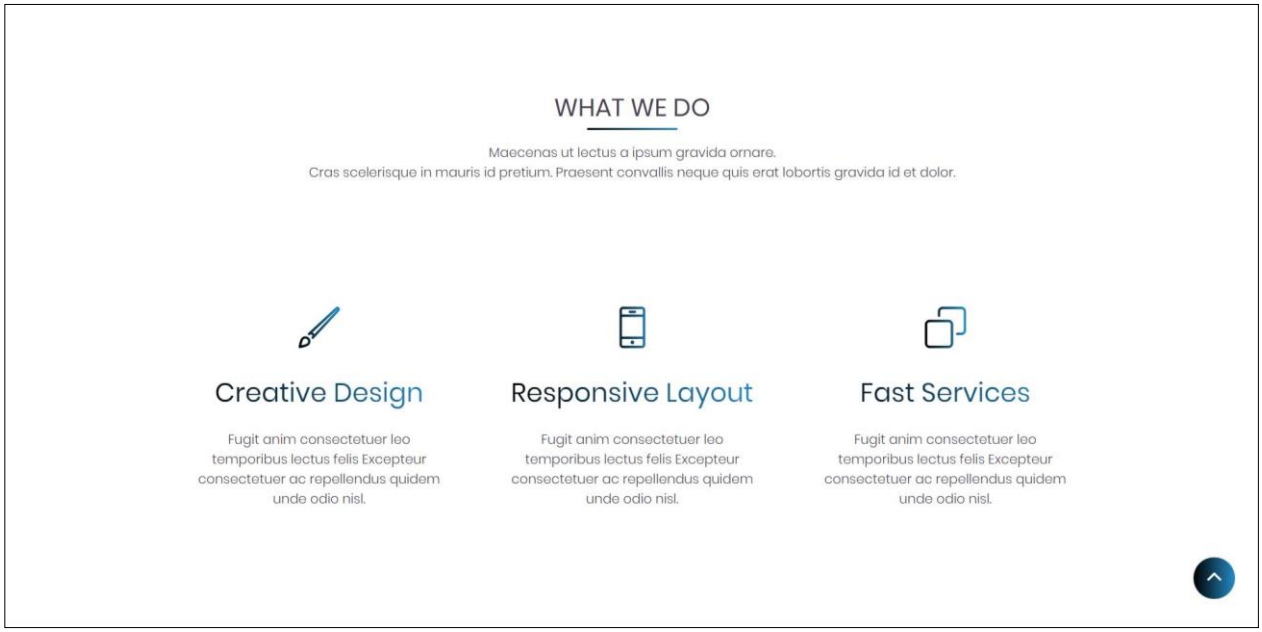
Figur 5-1 De fem kodeblokkene som blir brukt i brukerakseptansetesten.

#	FR #	Testtilfelle
1	FR 3	Nettsidemalen er modifisert slik at den kan tilpasses tilstrekkelig.
2	FR 2	Det er mulig å laste opp innhold til databasen, og så generere en nettside med dette innholdet.
3	FR 4	Når man oppretter en ny nettside med utvalgte blokker og egendefinert innhold, vil den resulterende nettsiden gjengi utseende til den valgte malen.
4	FR 5	Det er mulig å slette blokker til en eksisterende nettside.
5	FR 6	Det er mulig å redigere blokker til en eksisterende nettside.

Tabell 5-2 Testtilfeller brukerakseptansetesting. «FR» er funksjonelle krav, og refererer til Tabell 4-1.

5.1.3.1 Testtilfelle 1

Dette er en test som til en viss grad baserer seg på skjønn. Forskjellige personer vil i stor grad kunne konkludere forskjellig i denne testen. Det er likevel viktig å ha den med som en akseptansetest, ettersom at testen i stor grad er med på å vurdere brukerens inntrykk og opplevelse av publiseringsløsningen. For å kunne gi leseren en bedre forståelse av hva denne testen gikk ut på, kommer det nå en forklaring til formuleringen «kan tilpasses tilstrekkelig». I eksempelet under vises en av kodeblokkene som ble brukt i akseptansetestingen.



Figur 5-2 Kodeblokk.

Denne kodeblokken inneholder en overskrift og ett tekstavsnitt. Videre inneholder den tre felter som hver inneholder ett ikon, en overskrift, og ett tekstfelt. Dette er hvordan nettsiden vil se ut med den opprinnelige HTML-koden i nettsidemalen vi har benyttet. For å svare på om brukeren kan tilpasse den tilstrekkelig, kan følgende vurderinger gjøres:

- Bør alle felter være obligatoriske for brukeren i denne kodeblokken?
- Bør bruker kunne endre på mer enn tekstinnhold, i dette tilfellet også selv kunne velge ikoner?
- Bør bruker selv kunne velge hvor mange felter med ikoner, overskrifter og tekstfelter siden skal inneholde?

I kodeblokken som er avbildet over endte vi opp med å gjøre overskriften og det første tekstavsnittet obligatorisk. Brukeren kunne selv bestemme om vedkommende ønsket å benytte

seg av feltene som inneholdt ikon, overskrift og tekstinnhold, men med en begrensning. Brukeren måtte i så tilfelle ha 3, 6, 9 osv. forekomster av denne typen. Om dette tilfredsstilte at kodeblokken kunne tilpasses tilstrekkelig, var noe vi fikk vurdert med akseptansetesting. Testene ble som nevnt utført via et videomøte med oppdragsgiver. I denne testen var det oppdragsgiver som navigerte gjennom brukergrensesnittet og valgte blant kodeblokkene som ble beskrevet i starten av dette kapittelet. Dersom han fra et brukerperspektiv mente at en kodeblokk burde ha mulighet for flere eller færre tilpasninger, fikk vi tilbakemelding om det.

5.1.3.2 Testtilfelle 2

Her skulle det testes om det var mulig å laste opp innhold til databasen, for å så generere en nettside med dette innholdet. Oppdragsgiver visste allerede at dette var mulig og at det fungerte. Gjennom de ukentlige møtene vi har hatt med oppdragiver så har dette blitt vist frem flere ganger. Derfor ble dette formelt godkjent uten grundig testing på selve videomøtet.

5.1.3.3 Testtilfelle 3

I denne testen var det oppdragsgiver som navigerte gjennom brukergrensesnittet og skulle velge alle de fem kodeblokkene som ble vist i starten av dette kapittelet. Videre skulle alle obligatoriske og ønskede felter fylles inn og lastes opp. Det neste steget var å trykke på knappen for å generere nettsiden. Til slutt skulle det vurderes om nettsiden hadde klart å plassere all informasjon fra brukerinput som forventet.

5.1.3.4 Testtilfelle 4

Oppdragsgiver skulle her hente nettsiden han hadde laget i testtilfelle 3. Han skulle så velge én vilkårlig kodeblokk blant de fem tilgjengelige i nettsiden, og slette denne. Videre skulle han som i testtilfelle 3 laste opp, og generere nettsiden på nytt. Kodeblokken burde nå være borte fra nettsiden som forventet.

5.1.3.5 Testtilfelle 5

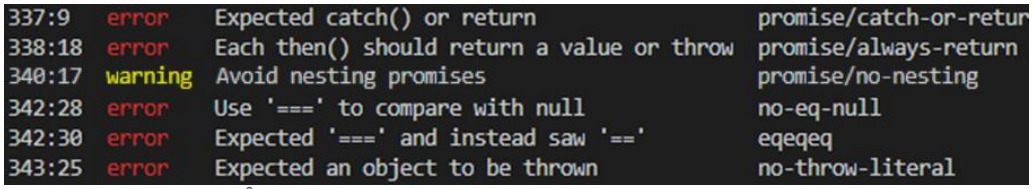
I det siste testtilfellet skulle oppdragsgiveren hente nettsiden han hadde laget i testtilfelle 4. Han skulle så gjøre endringer i input som han tidligere hadde fylt ut. Til slutt skulle han laste opp og generere nettsiden på nytt. Innholdet i de feltene han gjorde endringer burde nå ha endret seg som forventet.

5.1.4 Kvalitetstesting

Evaluering for å forsikre oss om at koden er av høy kvalitet er ikke satt i sterkt fokus. Dette er blitt diskutert med prosjekteier som sa at kvaliteten på koden ikke har høy prioritet, da dette er en prototype eller et «proof of concept». Det viktigste har vært å få til kjernefunksjonalitetene og finne gode løsninger.

Selv om kodekvalitet ikke var høy prioritet, har vi tatt i bruk «lint»-verktøyet ESLint. En «lint» er et statisk kodeanalyseringsverktøy som går igjennom kode og gir advarsler om syntaktiske feil, problematiske skrivemåter, formatteringskonvensjoner og liknende (Phacility, u.d.). ESLint tillater at brukeren kan spesifisere regler selv (ESLint, u.d.). Firebase har en liste med standardregler som vi har brukt mens vi har skrevet kode. Dette kodeanalyseringsverktøyet har vært til god nytte for gruppen, da vi ikke er så kjent med JavaScript fra før av. Det har hjulpet oss til å følge bestepraksiser, fange opp enkelte feil som vi ikke har sett, og gitt generelt høyere kvalitet på koden.

På figur 5-3 er noen eksempler på feilmeldinger fra ESLint vi møtte på i løpet av utviklingen. Etter å ha ordnet opp i feil som disse, oppnår vi bedre lesbarhet, ytelse, mer konsistens og lettere vedlikehold.

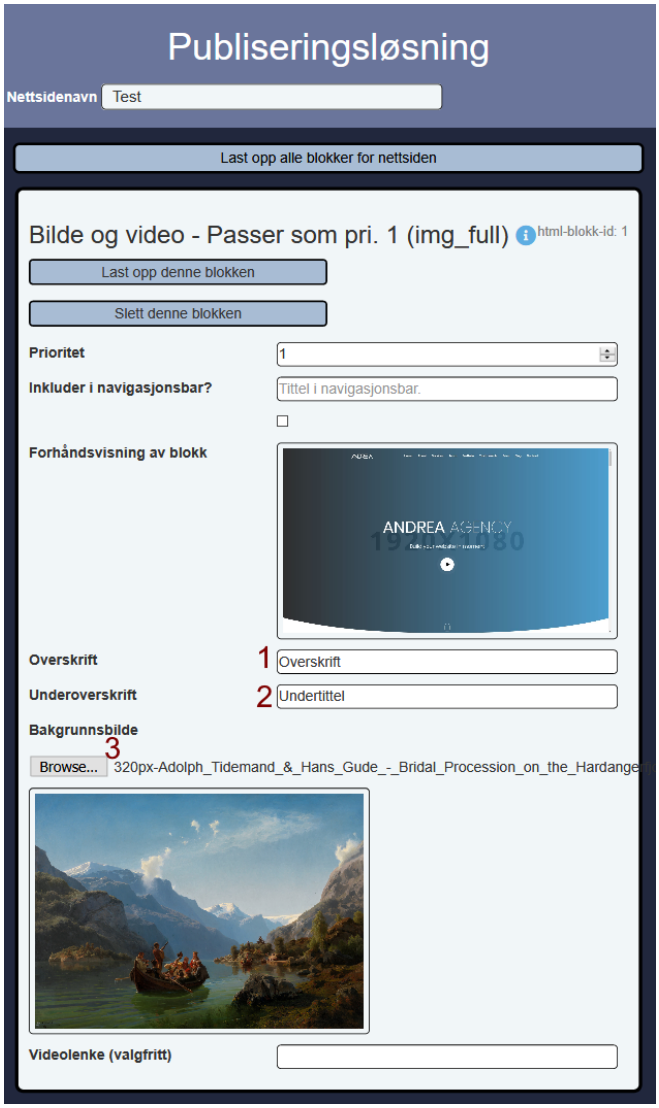


Figur 5-3 Eksempler på feilmeldinger fra ESLint.

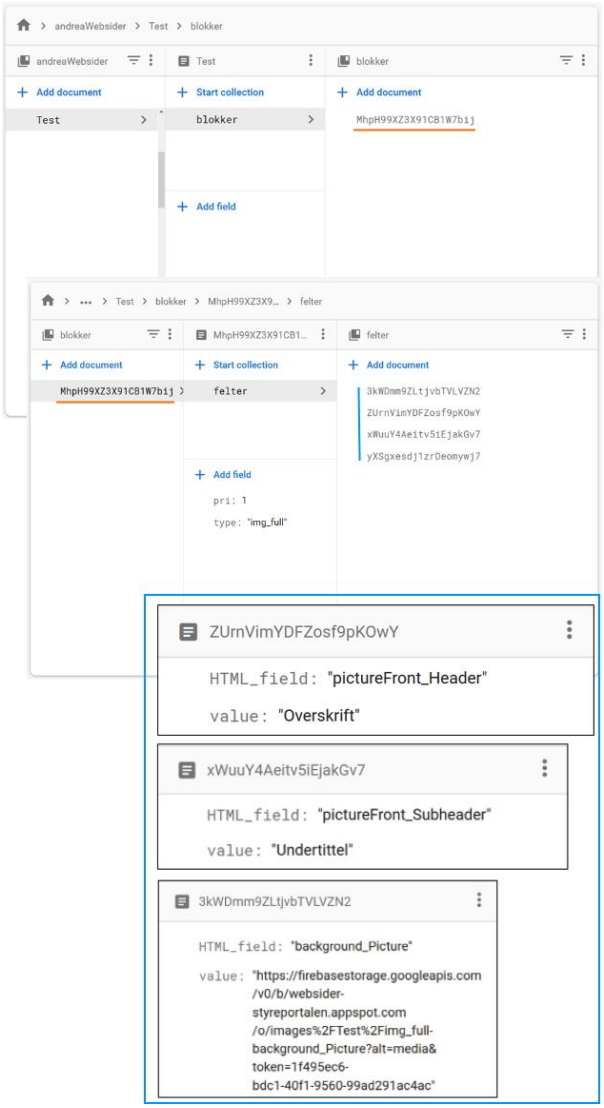
5.2 Evalueringsresultat

5.2.1 Resultat integrasjonstesting

Figurene under viser hvordan testtilfelle 1 i integrasjonstesten ble gjennomført. Figur 5-4 viser et utsnitt av brukergrensesnittet. Legg merke til verdien i «Nettsidenavn» øverst og de nummererte input-feltene lengre nede. Etter at blokken er lastet opp, sjekkes det i Firestore databasen om dataen ligger der. På figur 5-5 kan vi se at et dokument med navn «Test» er opprettet. I den er det ett blokkdokument (markert oransje). Dette dokumentet har flere feltdokumenter (markert blått), med «value» som tilsvarer input-verdiene. (Her «Overskrift» og «Undertittel»).

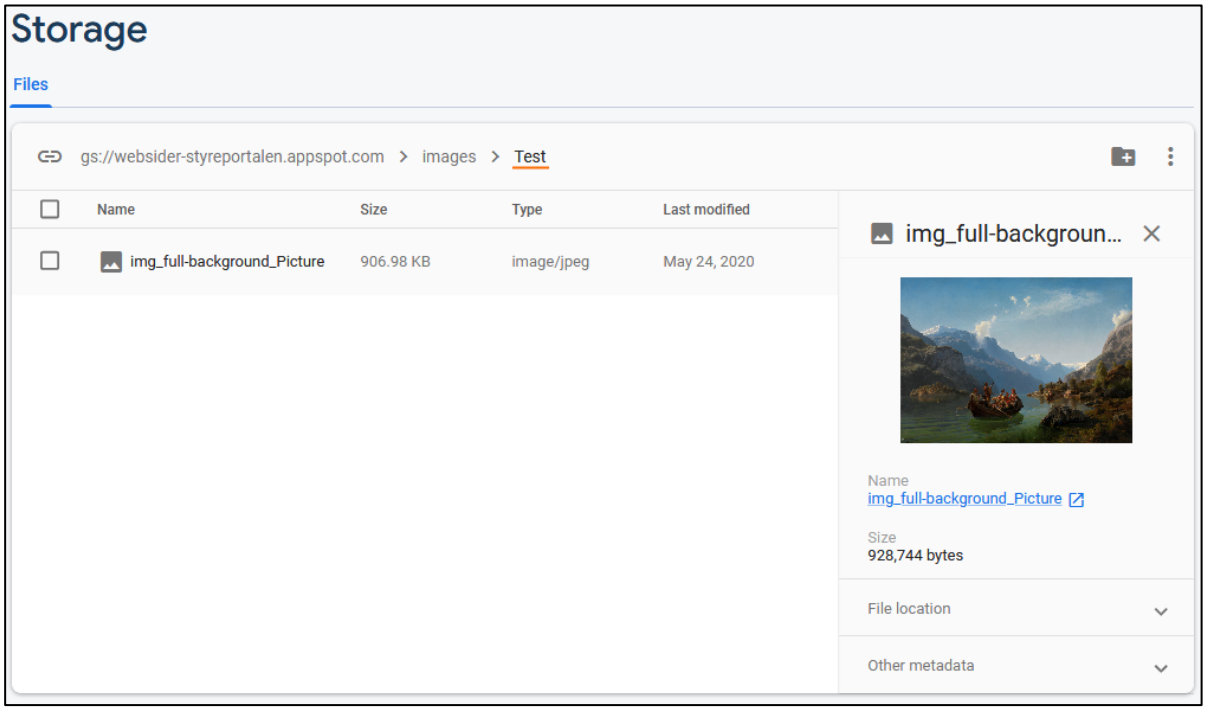


Figur 5-4 Opplastning av blokk fra brukergrensesnittet.



Figur 5-5 Ferdig opplastet data i Firestore-databasen.

Figur 5-6 under viser et skjermbilde fra Firebase Storage. Her ser man at bildet er blitt lastet opp, som forventet.



Figur 5-6 Skjermbilde fra Firebase Storage. Bildet sendt fra brukergrensesnittet ligger nå her.

Neste testtilfelle var om den genererte nettsiden klarer å hente data fra Firestore. På figur 5-7 vises et skjermbilde av den genererte siden. Teksten og bildet som vises, er lik som input-verdiene fra testtilfelle 1. Dette betyr at funksjonen som genererte nettsiden fikk kontakt med databasen og hentingen av data var vellykket.

Figur 5-7 Generert nettside med én blokk, (forside).

Tabellen under oppsummerer resultatene fra integrasjonstestingen.

Testtilfellenummer #	Virkelig resultat	Godkjent/Ikke godkjent
1	Som forventet	Godkjent
1.1	Som forventet	Godkjent
2	Som forventet	Godkjent

Tabell 5-3 Resultater integrasjonstesting

Integrasjonstestene har blitt gjennomført kontinuerlig under utvikling. Det har gitt oss mulighet til å vite om de ulike komponentene i prosjektet har hengt sammen og klart å kommunisere.

5.2.2 Resultat akseptansetesting

Testtilfelle nummer #	Godkjent/Ikke godkjent	Kommentar
1	Godkjent	Den er grei.
2	Godkjent	Også grei.
3	Godkjent	Funksjonaliteten er på plass. Fortsatt enkelte små-detaljer som kunne vært ordnet på.
4	Godkjent	
5	Godkjent	

Tabell 5-4 Resultater brukerakseptansetesting

Med bakgrunn i resultatene fra akseptansetestene og oppdragsgivers tilbakemeldinger kan vi trekke følgende slutninger:

- Vår løsning har vist at det er mulig bruke en ferdig nettsidemal, dele den opp for så å benytte den i en publiseringsløsning hvor bruker har mulighet til å lage en nettside.
- Det er i vår løsning mulig å gjøre endringer som å slette, endre og legge til innhold for en nettside som er laget ved hjelp av publiseringsløsningen.
- All informasjon blir som ønsket lagret i Firebase og dens tilhørende komponenter (Firebase Storage, Cloud Functions, Firestore).

Vår endelige prototype er per i dag en fungerende publiseringsløsning som ivaretar oppdragsgivers initielle kravspesifikasjoner. Oppdragsgiver er fornøyd med resultatet og sier at de vil ta utgangspunkt i vår prototype når de skal lage den endelige publiseringsløsningen.

6 DISKUSJON

I første del av dette kapittelet vil vi ta for oss konsekvenser av tilnærminger vi har gjort og hvordan de har påvirket det endelige resultatet. Videre kommer det en diskusjon om hvordan resultatet kunne blitt forbedret hvis prosjektet skulle blitt utført på nytt.

6.1 Konsekvenser og resultat av valgte tilnærminger

I denne delen vil vi gå gjennom hvordan vi har tilnærmet oss prosjektet og hvordan dette har bidratt til å påvirke resultatet. Våre hyppige møter med oppdragsgiver, brukergrensesnittet, hvordan gruppen har valgt å samarbeide og vår iterative arbeidsmetodikk vil bli diskutert her.

6.1.1 Ukentlige møter med oppdragsgiver

Det at gruppen tidlig i prosjektet avtalte å ha ukentlige møter med oppdragsgiver har i ettertid vist seg å være viktigere enn først antatt. Oppgaven hadde noen få generelle krav til løsningen, men ikke nok spesifikke krav til at vi hele tiden med sikkerhet visste om vi var på riktig vei. Å ha ukentlige møter hvor vi kunne ha en dialog med oppdragsgiver og vise frem det vi hadde gjort, hvilken problemer vi eventuelt hadde støtt på og diskutere veien videre i prosjektet, har vært essensielt for løsningen vi har kommet frem til.

Vi hadde vårt første møte med oppdragsgiver tidlig i februar, og det neste møtet var tidlig i mars. Grunnen til at det var så få møter i denne perioden skyldtes at gruppemedlemmene prioriterte ett annet skolefag i denne perioden. Gruppen ble enige om at vi skulle begynne å lære litt hver for oss, om de ulike teknologiene vi ville komme til å benytte i prosjektet. Tidlig i februar forstod vi det slik at Firebase og React var de to viktigste teknologiene vi kom til å ta i bruk og vi begynte derfor med å lære om disse. Det viste seg senere at React ikke var aktuelt likevel. En konsekvens av at vi valgte å ha lav møtehyppighet i denne perioden førte derfor til at dette ikke ble fanget opp, og at vi brukte tid på noe som ikke var relevant for oppgaven.

6.1.2 Lage ett brukergrensesnitt

I utgangspunktet hadde vi ikke planer om å lage et brukergrensesnitt, ettersom at vi i prosjektet skulle fokusere på å utvikle backend-delen. Det ble tidlig utfordrende å utføre testing mot backend og vi fant ganske raskt ut at det var nødvendig å få et hjelpeverktøy på plass. Konsekvensen av å ha fått på plass et brukergrensesnitt har vist seg å bli ett viktig hjelpemiddel for mer effektiv testing, og ikke minst for å enklere kunne vise frem hva vi har fått til, på de ukentlige møtene med Styreportalen AS. Brukergrensesnitt har også gitt oss muligheten til å se prosjektet fra en brukers synspunkt. Dette har vært verdifullt og har gitt både ekstern veileder og oss ideer til funksjonaliteter som burde være på plass, som vi først kanskje ikke hadde tenkt på.

6.1.3 Plan for hvordan gruppen skulle jobbe sammen

På grunn av covid-19 pandemien, har vi måtte arbeide som en gruppe på en helt annen måte enn vi har vært vant med tidligere. Å finne en best mulig måte å samarbeide på som en gruppe har vært viktig for prosjektets fremgang. Vi har i stor grad valgt å arbeide i samme tidsrom, og har benyttet blant annet kommunikasjonsplattformen Discord for å kommunisere. Discord har også en funksjon som tillater deling av skjerm, noe vi har benyttet oss av ofte. Dersom vi i perioder har arbeidet uten kommunikasjon, har hvert gruppemedlem logget i en intern logg hva vedkommende har gjort på den aktuelle dagen, og eventuelle løsninger og utfordringer vedkommende har støtt på. Måten vi har valgt å samarbeide på har fungert bra, og som en konsekvens av det har arbeidet med oppgaven gått knirkefritt gitt forholdene. Gruppemedlemmene har alltid hatt informasjon tilgjengelig om hva de andre gruppemedlemmene jobber med på et gitt tidspunkt i oppgaven, om ideer til funksjonalitet har blitt forkastet eller er ferdig osv.

6.1.4 Løse prosjektoppgaven med en iterativ tilnærming

Vi har gjennom hele prosjektet arbeidet iterativt. Målsettingen for hver iterasjon har vært å kunne levere en fungerende, stabil, integrert del av det totale systemet. Å ha mange små iterasjoner med leveranse og evaluering har fungert bra. Vi har klart å fange opp feil tidlig og rette disse, samtidig som det har vært enklere å finne ut hvor feilen ligger jo mindre iterasjonene har vært. Det har også vært en stor fordel å ha en iterativ tilnærming med tanke på de ukentlige møtene med oppdragsgiver og det at vi har hatt en forholdsvis åpen kravliste. Vi mener og tror at denne tilnærmingen har bidratt til bedre kodekvalitet, færre feil i kodingen og at den har bidratt til at vi har holdt oss på riktig spor, og ikke havnet på avveier.

6.2 Utføre prosjekt på nytt

Gruppen har gjort opp noen meninger om hvordan resultatet kunne blitt forbedret, dersom vi hadde utført prosjektet på nytt. Vi har tatt utgangspunktet i én nettsidemal der både gruppen og oppdragsgiver i stor grad er fornøyd med resultatet. Med forbedret resultat i dette tilfellet, tenker vi hovedsakelig på det å implementere løsningen vår til flere av nettsidemalene vi fikk tildelt. Hvorvidt vår løsning er enkel å implementere i andre nettsidemaler, fikk vi aldri tid til å finne ut. To sentrale stikkord blir derfor om vi kunne brukt tiden bedre, eller effektivisert arbeidsmetodikken vår mer. Svaret på førstnevnte er nok ja. Vi fikk utdelt prosjektoppgaven sent i januar, og hadde første møte med oppdragsgiver tidlig i februar. Likevel kom vi ikke skikkelig i gang med prosjektoppgaven før tidlig i mars. Selv om dette var en travel periode, burde vi ha klart å sette av en dag i uken, der vi jobbet med prosjektet. Angående arbeidsmetodikk mener vi at arbeidsmetodikken vår har fungert bra, og at det er lite å utsette på her.

Oppsummert så mener vi først og fremst at det er tid som har hindret oss i å komme lengre enn det vi gjorde. Om vi hadde brukt mindre tid på den ene nettsidemalen vi har arbeidet med, fordi vi ønsket å forsøke å implementere løsningen i flere andre nettsidemaler, så ville vi mest sannsynlig endt opp med et dårligere resultat.

7 KONKLUSJON OG VIDERE ARBEID

7.1 Konklusjon

Det har vært motiverende å jobbe med et prosjekt der resultatene vi ender opp med, faktisk har en nytteverdi og kan brukes i videre arbeid. Å få mulighet til å jobbe med en reell oppdragsgiver har vært interessant på flere måter. Ukentlige møter med oppdragsgiver der vårt arbeid vurderes, og hvor kravspesifikasjoner endres underveis har gitt oss nyttig erfaring i veien videre mot arbeidslivet. Å arbeide sammen i en gruppe over en så lang periode, der vi selv legger planer for hvordan vi skal løse oppgaver for å nå det endelige målet, er noe vi har ansett som svært lærerikt og nyttig. Til slutt har det vært spennende å lære om Firebase, som er en teknologi som ingen i gruppen tidligere hadde kjennskap til.

Målet for oppgaven var å lage en løsning som senere kunne bli implementert i et brukergrensesnitt som tillater brukeren å opprette en profesjonell nettside med få, enkle handlinger.

Vår vurdering om målet er nådd er basert på evalueringsresultatene som er beskrevet i kapittel 5.2. Vi har gjennom hele prosjektperioden konsentrert oss om én nettsidemal. Ut fra resultatene kan vi derfor konkludere med at målet er nådd for én spesifikk nettsidemal. Testene og evalueringene vi har hatt med oppdragsgiver om prototypen tilsier at vi har laget en prototype som kan brukes som et grunnlag for videre utvikling. Prototypen viser at det er mulig å benytte seg av en ferdig nettsidemal, bryte den ned, slik at den kan brukes i en publiseringsløsning.

Vi fikk derimot aldri tid til å ta i bruk flere nettsidemaler. Vi kan derfor ikke si med sikkerhet at løsningen vil fungere for andre nettsidemaler, da dette aldri ble testet. Med tanke på hvordan løsningen vår er bygd opp, bør den også fungere for andre nettsidemaler der man har bearbeidet HTML-koden og lagret denne i Firestore. Da forutsettes det at malene er ensideapplikasjoner.

7.2 Videre arbeid

Dersom prototypen vår skal videreutvikles, må funksjonen i Cloud Functions som genererer nettsidene skrives om, slik at nettsiden hostes via for eksempel Firebase Hosting. Per nå returnerer denne funksjonen en streng, og nettsiden vil bli generert hver gang man gjør en HTTP-forespørsel til funksjonen. Vi snakket med oppdragsgiver om denne løsningen tidlig i prosjekttiden. Det ville være en god midlertidig løsning i forbindelse med utvikling av konseptet og for testing, men ikke som en endelig løsning. Oppdragsgiver anså at hosting av nettsiden ikke var noe vi skulle prioritere, da dette var noe de enkelt kunne implementere senere selv. Å utføre en endring hvor nettsider hostes vil føre til en del endringer i hvordan vi lagrer CSS og JavaScript-filer som hører til nettsiden. Per nå er alle disse filene lagret i Firebase Storage. Endringen vil også tillate bruk av en relativ lagringssti til disse filene istedenfor en URL-referanse til Storage.

Vi har også diskutert med prosjekteier hva annet som kan bli jobbet videre med i vårt prosjekt. Punktene under er eksempler på fremtidige funksjonaliteter.

- Knytte løsningen opp mot et brukervennlig og sikkert brukergrensesnitt.
- Gjøre publiseringsløsning mer rettet mot kontinuerlig publisering av innhold. F.eks. legge til funksjonalitet for publisering av innlegg i en «blogg»-blokk.
- Etter hvert legge til funksjonaliteter for site mapping og Google Analytics. For markedsføring og statistikk for brukeren.

Det er mange endringer og funksjonaliteter som kan bli lagt til oppgaven, og det blir spennende å se hvordan den endelige løsningen blir.

8 REFERANSER

Enduro.js, 2020. *Home: Enduro.js.* [Online]
Available at: https://www.endurojs.com/
[Accessed 3. April 2020].
ESLint, n.d. *About.* [Online]
Available at: https://eslint.org/docs/about/
[Accessed 19. Mai 2020].
Firebase, 2020a. *Documentation - Cloud Firestore.* [Online]
Available at: https://firebase.google.com/docs/firestore
[Accessed 2. April 2020].
Firebase, 2020b. *Documentation: Cloud Storage.* [Online]
Available at: https://firebase.google.com/docs/storage
[Accessed 2. April 2020].
Firebase, 2020c. *Introduction - Firebase Hosting.* [Online]
Available at: https://firebase.google.com/docs/hosting
[Accessed 2. April 2020].
Firebase, 2020d. *Introduction - Cloud Functions for Firebase.* [Online]
Available at: https://firebase.google.com/docs/functions
[Accessed 2. April 2020].
IEEE, 2010. *ISO/IEC/IEEE International Standard - Systems and software engineering -- Vocabulary.* 1. ed. s.l.:IEEE.
Kinsta, 2020. *Kinsta.* [Online]
Available at: https://kinsta.com/knowledgebase/content-management-system/
[Accessed 31. Mai 2020].
Microsoft, 2020. *Visual Studio Code - Docs.* [Online]
Available at: https://code.visualstudio.com/docs
[Accessed 26 Mai 2020].
Naik, K. & Tripathy, P., 2008. *Software Testing and Quality Assurance: Theory and Practice.* 1. ed. Hoboken, New Jersey: John Wiley & Sons.
Norwegian Testing Board, 2016. *Terminologi for test av programvare.* [Online]
Available at: http://www.istqb-norge.no/wp/wp-content/uploads/2016/02/2016-01-31-ISTQB-Terminologi-testing-v-2-4no.pdf
[Accessed 19. Mai 2020].

Phacility, n.d. *Arcanist User Guide: Lint.* [Online]
Available at: https://secure.phabricator.com/book/phabricator/article/arcanist_lint/
[Accessed 19. Mai 2020].
Schwaber, K. & Sutherland, J., 2017. *The Scrum Guide™.* [Online]
Available at: https://www.scrumguides.org/scrum-guide.html
[Accessed 3. April 2020].
Tutorials Point, 2020a. *Javascript Tutorial.* [Online]
Available at: https://www.tutorialspoint.com/javascript/index.htm
[Accessed 2. April 2020].
Tutorials Point, 2020b. *What is CSS?.* [Online]
Available at: https://www.tutorialspoint.com/css/what_is_css.htm
[Accessed 2. April 2020].
Tutorials Point, 2020c. *What is HTML ?.* [Online]
Available at: https://www.tutorialspoint.com/html/what_is_html.htm
[Accessed 2. April 2020].
Tutorials Point, 2020d. *Firebase - Home.* [Online]
Available at: https://www.tutorialspoint.com/firebase/index.htm
[Accessed 1. April 2020].
TutorialsTeacher, 2020. *What is Node.js?.* [Online]
Available at: https://www.tutorialsteacher.com/nodejs/what-is-nodejs
[Accessed 2. April 2020].
W3Schools, 2020. *What is Github?.* [Online]
Available at: https://www.w3schools.com/whatis/whatis_github.asp
[Accessed 3. April 2020].
Wordpress, 2020. *About: Democratize Publishing.* [Online]
Available at: https://wordpress.org/about
[Accessed 3. April 2020].

9 APPENDIX

9.1 Risikoliste

Type	Risiko	P	C	RF	Tiltak
Planlegging	Mangelfull tidsplan for forskjellige deler av prosjektet	2	3	6	Sørge for at vi på et tidlig stadie får oversikt over alt som skal gjøres og legge en plan for det. Ukjente deloppgaver bør ikke dukke opp på et senere stadie
	Koronavirus kan føre til endring av planer	4	2	8	Ha i tankene at utenkelige scenarioer kan intreffe, og ikke bli fullstendig overrasket om vi underveis må revurdere våre opprinnelige planer.
	Usikkerhet vedrørende hvor lang tid hver del av prosjektet vil ta	4	3	12	Sørge for at vi på et tidlig stadie får oversikt over alt som skal gjøres og begynner på det, sånn at vi får en oppfatning av hvor mye tid hver del krever.
	Koronavirus kan føre til at gruppemedlemmer får mindre tid til å bidra i selve prosjektet	3	4	12	Finne løsninger som gjør til at tiden man kan arbeide med prosjektet ikke reduseres. F.eks. få familie til å passe barn på dagtid, i stedet for at gruppemedlemmet må gjøre det selv.
Kompetanse	Lav kvalitetet på arbeidet som gjøres	3	4	12	Lære om teknologiene som skal brukes, og gode praksiser for bruk av teknologien, ved hjelp av relevant litteratur og personer som sitter med kompetanse innenfor teknologi og problem vi skal løse.
	Manglende kompetanse kan føre til at vi ikke når mål vi har satt til planlagt tid	2	5	10	Lære om teknologiene som skal brukes ved hjelp av relevant litteratur og personer som sitter med kompetanse innenfor teknologien
Kommunikasjon	Dårlig og/eller manglende kommunikasjon mot oppdragsgiver og internt i gruppen	1	4	4	Sørge for at kommunikasjon er på et tilfredsstillende nivå ved at gruppen har kontakt daglig og forteller hva de har gjort, i tillegg til hyppige møter med oppdragsgiver for å få tilbakemelding på at vi hele tiden beveger oss i riktig retning og ikke inn på et sidespor
	Manglende informasjon om hva som er viktigst å få få fullført, om man ikke klarer å få fullført alt	3	2	6	Ta kontakt med oppdragsgiver for prioritetsliste av deloppgaver
Produkt	For stor arbeidsmengde med hensyn på kompetanse og gitt tidsfrist	2	4	8	Sørge for å bruke tiden vi har disponibelt effektivt ved å pushe hverandre til å gjøre sitt beste, hjelpe hverandre når man står fast, passe på at vi hele tiden har kontroll og gjør det som oppdragsgiver faktisk ønsker
	Uklare krav og mål for oppgaven	2	5	10	Sørge for at vi ikke angriper oppgaven på feil måte
	Kan ikke brukes, eller bygges videre på av oppdragsgiver	1	5	5	Sørge for at vi forstår oppgaven, og hvordan vi skal løse den på en måte som tilfredsstiller oppdragsgivers krav og forventninger

9.2 GANTT diagram

Beskrivelse	Kategori	Fremdrift	Periode	Uke 4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Prosjektstart																							
Vurdering av tilgjenglige oppgaver	Mål	100 %	20.01.2020 - 23.01.2020																				
Tildeling av oppgave	Mål	100 %	24.01.2020																				
Første møte med oppdragsgiver	Mål	100 %	06.02.2020																				
Første møte med veileder	Mål	100 %	04.02.2020																				
Oppgaver fra Høgskolen på Vestlandet																							
Diverse mindre oppgaver	Milepæl	100 %	Periodevis																				
Forposjektrapport	Mål	100 %	26.03.2020 - 03.04.2020																				
Planlagte møter med veileder	Går etter planen	50 %	Annenhver uke																				
Bachelorrapport	Går etter planen	18 %	26.03.2020 - 02.06.2020																				
Blogg	Går etter planen	47 %	Skrive på bloggen ukentlig																				
Oppgaver fra oppdragsgiver																							
Planlagte statusmøter	Går etter planen	42 %	Ukentlig																				
Fungerende prototype for nettsidemal del 1	Milepæl	25 %	16.03.2020 - 05.04.2020																				
Fungerende prototype for nettsidemal del 2	Milepæl	10 %	06.04.2020 - 26.04.2020																				
Testing og eventuell retting av utført arbeid	Milepæl	40 %	Testing underveis av utført arbeid																				
Implementering av løsningen fra nettsidemal til andre nettsidemaler	Milepæl	0 %	27.04.2020 - 24.05.2020																				
Kompetanseheving	Går etter planen	65 %	Når nødvendig																				
Kommentarer																							

Fargekode **rød** betyr at vi har brukt mer tid på en oppgaven(e) enn det som var først planlagt.
Fargekode **blå** betyr at vi har utført oppgaven(e) under den bestemte tidsrammen.
Fargekode **gul** betyr at utføring av oppgaven(e) er avbrutt eller fjernet.

9.3 Revidert GANTT diagram

Beskrivelse	Kategori	Fremdrift	Periode	Uke 4 - 10	11	12	13	14	15	16	17	18	19	20	21	22	23
Prosjektstart																	
Vurdering av tilgjenglige oppgaver	Mål	100 %	20.01.2020 - 23.01.2020														
Tildeling av oppgave	Mål	100 %	24.01.2020														
Første møte med oppdragsgiver	Mål	100 %	06.02.2020														
Første møte med veileder	Mål	100 %	04.02.2020														
Oppgaver fra Høgskolen på Vestlandet																	
Diverse mindre oppgaver	Milepæl	100 %	Periodevis														
Forposjektrapport	Mål	100 %	26.03.2020 - 03.04.2020														
Planlagte møter med veileder	Går etter planen	100 %	Annenhver uke														
Bachelorrapport	Går etter planen	100 %	26.03.2020 - 02.06.2020														
Blogg	Går etter planen	100 %	Skrive på bloggen ukentlig														
Oppgaver fra oppdragsgiver																	
Planlagte statusmøter	Går etter planen	100 %	Ukentlig														
Fungerende prototype fra kodeblokk	Milepæl	100 %	16.03.2020 - 07.04.2020														
Fungerende prototype for Andrea-nettsidemal	Milepæl	100 %	08.04.2020 - 10.05.2020														
Testing og eventuell retting av utført arbeid	Milepæl	100 %	Testing underveis av utført arbeid														
Implementering av løsningen fra Andrea-nettsidemal til andre nettsidemaler	Milepæl	0 %	27.04.2020 - 24.05.2020														
Kompetanseheving	Går etter planen	100 %	Når nødvendig														
Kommentarer																	

Fargekode **rød** betyr at vi har brukt mer tid på en oppgaven(e) enn det som var først planlagt.
Fargekode **blå** betyr at vi har utført oppgaven(e) under den bestemte tidsrammen.
Fargekode **gul** betyr at utføring av oppgaven(e) er avbrutt eller fjernet.