



Høgskulen
på Vestlandet

BACHELOROPPGAVE

AllWrite - Åpner for kreativ skriving og lesing

AllWrite - Enabling creative writing and reading

Sebastian Berge

Anders Mæhlum Halvorsen

Simen Østensen

Dataingeniør

Institutt for data- og realfag

Fakultet for ingeniør- og naturvitenskap

Innleveringsdato: 01.06.2020

Vi 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.

<i>Rapportens tittel:</i> AllWrite - Åpner for kreativ lesing og skriving AllWrite - Enabling creative reading and writing	<i>Dato:</i> 01.06.2020
<i>Forfatter(e):</i> Sebastian Berge, Anders Mæhlum Halvorsen og Simen Østensen	<i>Antall sider u/vedlegg:</i> 41
	<i>Antall sider vedlegg:</i> 1
<i>Studieretning: Dataingeniør</i>	<i>Antall disketter/CD-er:</i> 0
<i>Kontaktperson ved studieretning:</i> Harald Soleim	<i>Gradering:</i> Ingen
<i>Merknader:</i> Ingen	

<i>Oppdragsgiver:</i> Norbyte AS	<i>Oppdragsgivers referanse:</i> Ingen
<i>Oppdragsgivers kontaktperson:</i> Ivar Broder	<i>Telefon:</i> +47 977 17 701

<i>Sammendrag:</i> I denne rapporten beskrives det hvordan gruppen har utviklet et skrivesystem for skoler, til bruk for elever og lærere. Dette systemet heter AllWrite.app og er laget for Norbyte AS. AllWrite.app er et verktøy som skal engasjere unge til å lese og skrive mer. Systemet er et sammensatt verktøy bestående av en mobilapplikasjon for iOS og Android, en web-applikasjon og en database. Systemet som er utviklet i denne oppgaven skal videreutvikles av Norbyte AS.

Stikkord:

React	Flutter	Firebase
JavaScript	Dart	Cloud Functions

Høgskulen på Vestlandet, Fakultet for ingeniør- og natuvitenskap

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 skrevet av Sebastian Berge, Anders Mæhlum Halvorsen og Simen Østensen, som hovedprosjekt i DAT190 ved Høgskulen på Vestlandet. Oppgaven går ut på å lage en online opplevelse som stimulerer til kreativ lesing og skriving. Dette gjennomføres for å gi skoler et verktøy, der lærere kan planlegge en oppgave og elever kan skrive fritt fra fantasien. Verktøyet skal være tilgjengelig som en web- og mobilapplikasjon både på iOS og Android.

Vi ønsker å takke oppdragsgiver, Norbyte AS, for en god og utfyllende oppgave. I tillegg ønsker vi å takke Harald Soleim for gode tilbakemeldinger under arbeidet med rapporten og systemet.

Innholdsfortegnelse

1. INNLEDNING	1
1.1 MOTIVASJON OG MÅL	1
1.2 KONTEKST	2
1.3 AVGRENSNINGER	2
1.4 RESSURSER	2
1.5 OPPBYGGING AV RAPPORTEN	3
2. PROSJEKTBEKRIVELSE	5
2.1 PRAKTISK BAKGRUNN	5
2.2 LITTERATUR OM PROBLEMSTILLING	8
3. DESIGN AV PROSJEKTET	10
3.1 FORSLAG TIL LØSNING	10
3.2 VALGT LØSNING	15
3.3 PROSJEKTMETODIKK	16
3.4 EVALUERINGSPLAN	20
4. DETALJERT DESIGN	21
4.1 BAKGRUNN	21
4.2 ARKITEKTUR	22
4.3 GRAFISK GRENSESNIITT	27
5. EVALUERING	31
5.1 EVALUERINGSMETODE	31
5.2 EVALUERINGSRESULTAT	32
6. RESULTATER	34
7. DISKUSJON	34
7.1 KONSEKVENSER AV VALGT LØSNING	34
7.2 GJENNOMFØRING	34
8. KONKLUSJON OG VIDERE ARBEID	37
9. REFERANSER	38
10. APPENDIX	42
10.1 GANTT DIAGRAM	42

Figurliste

<i>Figur 1: Systemarkitektur</i>	22
<i>Figur 2: Databasestruktur</i>	23
<i>Figur 3: Prosessgraf</i>	24
<i>Figur 4: Arkitektur mellom mobilapplikasjon og database</i>	26
<i>Figur 5: Arkitektur mellom web-applikasjon og database</i>	27
<i>Figur 6: Illustrasjon web- og mobilapplikasjon</i>	28
<i>Figur 7: QR-kode</i>	28
<i>Figur 8: Opprettelse av write</i>	29
<i>Figur 9: Illustrasjon over lærerens dashboard</i>	29
<i>Figur 10: Illustrasjon av flyten under en write på mobilapplikasjonen</i>	30

Tabelliste

<i>Tabell 1: Risikoanalyse</i>	18
<i>Tabell 2: Resultat fra hovedfunksjonstest</i>	33

Ordliste:

Begrep	Forklaring
Autentisering	Autentisering refererer til prosessen med å validere at man er den man sier man er.
Back-end	Back-end refererer til den delen av programvaren som ligger nærmest databasen.
Bug	En “bug” refererer til et problem i koden som forårsaker at koden ikke kjører som den skal, eller at den ikke kjører i det hele tatt.
Front-end	Front-end refererer til det grafiske grensesnittet.
Git	Git er et versjonskontrollsystem.
KPI	Key Performance Indicator er et måletall som viser hvor godt en bedrift klarer seg (Twin, 2019).
Minste Brukbare Produkt (MBP)	Minste Brukbare Produkt er en referanse til en liste av funksjonaliteter som utgjør at et produkt som kan brukes, uten annen funksjonalitet.
Multi User Dungeon (MUD)	Multi User Dungeon er en type tekstbasert sanntidsrollespill som blir spilt over nett.
Native	Native menes at noe er designet for å fungere spesifikt på et system, for eksempel kun for iOS eller kun for Android.

Package	Package er et sett med relaterte klasser og grensesnitt.
QR-Kode	En QR-kode er en lesbar maskinkode, som inneholder data; som oftest strenger, nettadresser eller nummer.
Rammeverk	Rammeverk er en konseptuell struktur som er ment for å støtte videre utbygging.
Software Development Kit (SDK)	Et Software Development Kit er en samling av utviklingsverktøy i en enkel installerbar pakke. En SDK inneholder veldig ofte en kompilator for å kunne bygge applikasjoner.
Spillifisering	Spillifisering refererer til handlingen om å anvende spilldesignelementer og spillprinsipper utenfor spill-sammenhenger.
Tilstand	Tilstand referer til at applikasjonen husker de forskjellige tilstandene en variabel kan ha. For eksempel en bil kan ha tre forskjellige tilstander: stå i ro, kjøre og rygge.
User Experience (UX)	User Experience refererer til totalopplevelsen en bruker har ved anvendelse av et produkt.
Write	En “Write” er en oppgave som en lærer kan opprette på sin side. Den inneholder visse kriterier som elevene skal følge, som for eksempel antall ord og sjanger.

1. INNLEDNING

1.1 Motivasjon og mål

En finsk undersøkelse utført i 2019 fulgte 2500 elever gjennom grunnskolen. Resultatet av denne undersøkelsen viste at antallet bøker barn velger å lese, er en viktig driver for lese- og skriveutvikling i senere år. Spesielt ble det vektlagt at dette gjaldt barn som velger å lese skjønnlitterære bøker (Torppa et al., 2019).

Det å bli en erfaren leser er avgjørende for hvordan man presterer i videre utdanning, hvordan man får det i arbeidslivet samt hvordan man klarer seg generelt ellers i livet. I dag vokser unge opp i et samfunn som krever gode leseferdigheter, men i en undersøkelse utført av PISA viser det seg at færre norske ungdommer leser daglig nå enn før. Det store skillet i leseforståelsen oppstår mellom de som aldri leser noe og de som leser litt. Derfor er det avgjørende at unge får muligheten til å bli kjent med litteratur som kan engasjere dem, noe som kan medføre at de sitter seg ned med en bok framfor dataen på fritiden (Utdanningsforskning, 2019).

Norbyte AS er et selskap som har mye erfaring innen utvikling av dataspill. De leverer skreddersydde løsninger for å imøtekomme kundenes behov innen spillifisering, e-læring og markedsføring. De ønsker å gjøre noe for å endre utviklingen i samfunnet som den finske undersøkelsen viser. Selskapets ide er å lage en gratis applikasjon for både web og mobil, som skal engasjere folk i å lese og skrive. Dette systemet skal være en online opplevelse hvor brukeren skriver deler av en tekst som formes inn i en fortelling. Formålet med prosjektet er å tilby, samtidig som engasjere, unge elever til å skrive og lese mer.

Målet med dette prosjektet er å utvikle en online opplevelse som stimulerer til kreativ lesing og skriving. Det skal gjøres ved å gi skoler et verktøy der lærere kan planlegge en oppgave og elever kan fritt skrive fra fantasien. Verktøyet skal være tilgjengelig som en web-applikasjon og mobilapplikasjon både på iOS og Android.

1.2 Kontekst

Det skal utvikles en løsning som stimulerer kreativ lesing og skriving for skoleelever. Dette gjennomføres ved at en lærer starter ved å opprette en “write”. Læreren bestemmer da reglene og kategorien til denne “writen”, og skriver en kommentar om hva elevene skal skrive om. Elevene som er medlem i klassen, kan da starte denne “writen” og skrive første delen av denne teksten. Etter at elevene er ferdig med å skrive, vil hver elev i klassen få en annen elev sin tekst, som de skal fortsette på. Slik vil det fortsette et gitt antall ganger. Når alle elevene har levert for siste gang, er neste steg å lese de andre sine historier. Hver elev får da et visst antall historier som de skal lese gjennom, og kan deretter stemme på sine favoritter.

1.3 Avgrensninger

Oppdragsgiver hadde en del egenskaper og funksjoner han ønsket å få inn i applikasjonen. Dermed måtte vi fra starten bli enig om et minste brukbare produkt (MBP), som tilsvarte de mest nødvendige funksjonene for å opprette et fungerende sluttprodukt.

Mobilapplikasjonen ble avgrenset til at den kun var tilgjengelig for elever. Tilsvarende avgrenset vi web-delen av prosjektet til kun for lærere. Disse avgrensningene ble gjennomført for å gjøre oppgaven mindre kompleks.

1.4 Ressurser

Prosjektet hadde to egne UX grupper som tok seg av brukeropplevelsen for web og mobil. Disse gruppene sørget for detaljerte funksjonsbeskrivelser, som vi implementerte i prosjektets applikasjoner. Ressurser gruppen selv har brukt er Firebase med dens funksjoner og to rammeverk (Flutter for mobil og React for web).

1.5 Oppbygging av rapporten

Kapittel 1:

I dette kapitlet presenteres målet og motivasjonen for oppgaven. Videre tar vi for oss konteksten og hvilke ressurser som har blitt brukt.

Kapittel 2:

I dette kapitlet beskrives det hvorfor prosjektet er viktig for eieren, tidligere arbeid som er relevant, kravspesifikasjoner og innledende løsning.

Kapittel 3:

I dette kapitlet presenteres de mulige fremgangsmåtene for å utvikle systemet, samt hva vi har valgt og hvorfor. Videre tar vi for oss hvilke ressurser som er nødvendige, risikoer som er knyttet til prosjektet og en evalueringsmetode.

Kapittel 4:

I dette kapitlet tar vi for oss designet av systemet. Først vil vi beskrive arkitekturen av systemet i detalj og deretter vil vi presentere de viktigste funksjonene fra det grafiske grensesnittet.

Kapittel 5:

I dette kapitlet forklares hvilke evalueringsmetode som er brukt i prosjektet, hvordan de er gjennomført og resultatene vi fikk fra de forskjellige evalueringene.

Kapittel 6:

I dette kapitlet forklares det hvordan evalueringsresultatene bidrar til det måloppnåelse av det overordnede målet.

Kapittel 7:

I dette kapitlet følger det en diskusjon av konsekvensene av valgt løsning, hvilke utfordringer vi har møtt og hva vi kunne gjort annerledes.

Kapittel 8:

I dette kapitlet følger vår konklusjon og videre arbeid.

Kapittel 9:

Referanser.

Kapittel 10:

Appendiks med gantt-diagram.

2. PROSJEKTBEKRIVELSE

I dette kapitlet beskrives det hvorfor prosjektet er viktig for eieren, tidligere arbeid som er relevant, kravspesifikasjoner og innledende løsning.

2.1 Praktisk bakgrunn

Oppdragsgiver ønsket å lage et operativt system som skal fremme digital skrivelyst og glede, samt øke både kollektiv og individuell læringsvilje. Dette er årsakene for at prosjektet er blitt til. Prosjektet ble gjennomført sammen med prosjekteier Norbyte AS, Høgskulen på Vestlandet og Høgskolen Kristiania. Vi hadde ansvaret for front-end delen, altså utviklingen av applikasjonene, mens studentene ved Høgskolen Kristiania skulle ha ansvaret for brukeropplevelsen, det grafiske grensesnittet og databasen.

2.1.1 Prosjekteier

Prosjekteier Norbyte AS arbeider med ideutvikling og prosjektgjennomføring for nye digitale prosjekter. Selskapet har løpende inntekter fra salg av tidligere app-konsepter.

2.1.2 Tidligere arbeid

Daglig leder i Norbyte AS har vært salgsdirektør i den største norske spillprodusenten Funcom i åtte år, og har stått ansvarlig for brutto-salg på over 130 mill. USD. I tillegg har daglig leder mange års erfaring fra å produsere innhold for Multi-User Dungeons, og har designet på inneværende prosjekt i en årrekke.

Norbyte AS har tidligere levert design og utvikling til løsningen Geddit. Selskapet har løpende inntekter herfra som resultat av salg av konsept og design.

2.1.3 Initielle krav

Løsningen skal tilrettelegge for gjennomføring av en “AllWrite” - en digital samskrivingsprosess, for en skoleklasse. AllWriten settes opp av en lærer og gjennomføres med en klasse. Det skal videre brukes etablert teknologi for å sikre kvalitativ utvikling og drift av systemet.

2.1.4 Initiell løsningsidé

AllWrite er den eneste satsingen for prosjekteier Norbyte AS. Årsaken til dette er en underliggende tro på at digitale tekstbaserte opplevelser, hvor man selv bidrar med å skrive, vil være svært attraktivt i dagens samfunn.

Prosjektet er i bunn inspirert av den første formen for tekstbasert rollespill som var på internett midt på 90-tallet og fremover: Multi-User Dungeons (MUDs). MUDs er et tekstbasert spill som deles av mange brukere samtidig. Tekst (både kode/scripts og innholdsmessig tekst) går gjennom godkjennelsesfaser, og brukere samarbeider om å bygge innhold i en online setting for andre brukere som “spiller” i innholdet. MUDs er kategorisert som en tekstverden som har en puls som slår cirka hvert sekund. Med hver puls kan en gjøre endringer i den tekstuelle verden ved bruk av programmatisk scripts på innholdssiden, og “valg” fra spillere.

Det ligger et omfattende design bak prosjektet AllWrite. Oppdragsgiver har mål om å oppnå 5% økt skrive lyst i skoleklasser som tar i bruk systemet, og det er dermed første nøkkeltallsindikator (KPI) som skal nås i første fase. Løsningen vil være gratis for skoler og lærere, og produktet kan enkelt lokaliseres til fremmedspråk. Ved oppnåelse av en slik KPI, og løpende produktforbedringer, mener Norbyte AS at de vil klare å distribuere løsningen globalt, og hjelpe alle på grunnskolen gratis med et produkt som får dem til å skrive mer, og bedre.

Norbyte AS har brutt ned designet av AllWrite for å sikre at funksjonalitet skal være gjennomførbart med de gitte ressurser deltagende elever har. Det er følgelig for første fase satt en definisjon av en MBP som skal muliggjøre teknisk begrenset gjennomføring for klasser med reelle brukere. Denne MBPen vil la en lærer sette opp og gjennomføre en enkel AllWrite med klassen sin.

For en lærer vil prosessen være: sette opp AllWriten på en webside, invitere elever inn (QR kode eller lignende), godkjenne tekstbidrag fra elevene, og gjennomgå resultatene med klassen. Gjennomføringen vil følgelig kunne være tilpasset pensum da det er læreren selv som velger tema, og form, på det elevene skal skrive. Videre sikrer det faktum at læreren godkjenner bidragene og at tekstene som produseres har blitt kvalitetssikret av en lærer.

Når elevene installerer AllWrite på mobil, kan de bli medlem i en klasse ved hjelp av en adgangskode (QR-kode eller lignende). Elevene vil da kunne skrive avsnittet med tekst etter de kriterier lærer har definert (minimums antall ord, maksimalt antall ord, formkrav, innholdskrav). Elevene sender deretter avsnittet sitt via mobil-applikasjonen. Videre vil elevene få en annen elevs tekst (anonymt eller ikke), og skal avslutte denne innholdsmessig. Til slutt skal hver elev lese et antall andre bidrag fra klassen, og gi hjerter til de historiene vedkommende liker best.

MBP skal også ha følgende egenskaper:

- Støtte for norsk og engelsk språk via et system som kan utvides for å tilføre ytterligere lokalisering.
- Støtte for at flere klasser kan kjøre systemet samtidig.
- At lærer kan:
 - Sette antall runder elevene skal skrive i (standard er 2 runder).
 - Sette anonymitetsgrader ved oppsett av en AllWrite (kreditert med navn, anonymitet mellom elever (standard) og helt anonymt).
 - Sette hvor mange historier hver elev skal vurdere i siste fase.

- Ha støtte for en “credit list” (Hvem jobbet på prosjektet).
- Ha en end-user license agreement (EULA) ved oppstart.
- Kontaktinformasjon til lærer som elever kan se inne i tjenesten.
- La elever lese tidligere tekster som har vunnet, kreditert til klassens navn.

Etter MBPen er oppdragsgivers ambisjon å tilgjengeliggjøre følgende funksjonalitet for å øke kvaliteten på opplevelsen:

- Kryssklassefunksjonalitet (For eksempel at en klasse avslutter en tekst som kommer fra en annen klasse og motsatt).
- Kryssklassefunksjonalitet hvor man kan sende tegninger (Den historien klassen likte best, av alle historiene de leste fra en annen klasse, lager de en tegning av. De som skrev historien som vant, mottar derfor flere tegninger fra den andre klassen.)
- Lærere kan nå lærere fra andre klasser.
- Tilføre enkle avatarer for brukere, eventuelt med enkel animasjon.
- Oppnåelsessystem som låser opp forskjellige avatarer (Om en elev gjennomfører for eksempel tre “writes”, vil eleven få en ny avatar).
- Tilføre kategorier til tekster (Eventuelt ved hjelp av hashtags, som for eksempel #Eventyr, #Romantikk).
- Polish og produktforbedring basert på feedback fra MBP.
- Oppsett av AllWrite som ikke bare er for klasserom, men for skrivegrupper.
- Oppsett av AllWrite som går 1:1.
- Automatisering av oppsett i system (Planlegger).
- Automatisert deadline på automatiske oppsett.

2.2 Litteratur om problemstilling

AllWrite er en ide som bygger på forbedring av lesing og skriving. Elevene får først instruksjoner om hva teksten skal omhandle. Deretter skal eleven skrive første del av teksten. Når de er ferdig, får de utdelt andre elever sine tekster som de skal fortsette på. Slik fortsetter det helt til siste iterasjon, som er definert av læreren. Til slutt får alle

elever utdelt et visst antall historier som skal lese og vurdere. Selv om dette er en unik ide finnes det relaterte systemer som oppnår noe av det samme.

Book Creator er et system hvor elever kan lage sin egen bok ved hjelp av tekst, bilder, lyd og figurer. Elevene får utdelt et tema fra læreren om hva boken skal omhandle, og videre skriver og designer elevene ferdig boken og publiserer den. Boken vil så bli tilgjengelig for de andre elevene å lese.

Book Creator og AllWrite deler flere av de samme verdiene, men disse er utført på forskjellige måter. I motsetning til Book Creator setter AllWrite fokus på det å skrive videre på medelever sine tekster, mens Book Creator setter hovedfokuset sitt på å lage en egen bok uten noen form for samarbeid. I likhet vil tekstene bli tilgjengeliggjort for alle elevene i klassen.

3. DESIGN AV PROSJEKTET

I dette kapitlet tar vi for oss designet av prosjektet. Vi vil først ta for oss alternative løsninger på prosjektet og en diskusjon av alternativene. Deretter vil vi ta for oss hvilke løsning som har blitt valgt, hvilke prosjektmetodikk som har blitt brukt og hvordan vi planlegger å evaluere arbeidet.

3.1 Forslag til løsning

3.1.1 Alternative løsninger for back-end

Problemstillingen gitt av oppdragsgiver gikk ut på at vi skulle lagre og håndtere data fra både elever og lærere. Denne dataen må håndteres på en riktig måte for at hele systemet skal kunne fungere som beskrevet. Det var i all hovedsak tre forskjellige metoder som var aktuelle for oss.

Fra begynnelsen av prosjektet (før vi valgte denne oppgaven), var prosjektet originalt planlagt med en relasjonsdatabase i form av MySQL eller PostgreSQL. Begge systemene er veldig enkle relasjonsdatabasesystemer, som bare håndterer enkel data, og ikke mye mer. Ettersom en av de andre gruppene i Oslo skulle ha ansvar for databasen, var det en av de nevnte systemene de kunne tilby, da de har lært dette på UX-studiet.

ASP.NET Core er et rammeverk utviklet av Microsoft. Rammeverket ble utviklet for å kunne jobbe med moderne skybaserte og internettbaserte applikasjoner. Dette vil si at rammeverket har god støtte for å kunne jobbe med både mobil og web. ASP.NET gjør det mulig å bruke SQL som lagringsmotor for data, akkurat som MySQL og PostgreSQL. Ulikt fra systemene sist nevnt, har ASP.NET integrert server og data håndteringsmekanismer, som gjør det mulig effektivt å håndtere dataen og lage funksjoner for forskjellige situasjoner (Microsoft, u.å).

Firebase er en mobil- og nettapplikasjonsutviklingsplattform som kan bli brukt for håndtering og lagring av data, samt autentisering av brukere (Firebase, u.å.). Firebase ble utviklet av Firebase, men i senere tid kjøpt opp av Google. På bakgrunn av dette er Firebase godt egnet for Google sine utviklingsverktøy, men støtter også andre rammeverk svært godt (Firebase, u.å.). I senere tid har også funksjonalitet som meldingssystem, maskinlæring, applikasjonsdistribuering og integrert testing blitt en del av plattformen. Dette er nyttig funksjonalitet som kan bli brukt i senere tid, dersom man velger Firebase.

3.1.2 Alternative løsninger for mobil

Oppdragsgiver ønsket en applikasjon for både iOS og Android. Vi så for oss tre forskjellige alternativer for hvordan vi kunne gjennomført dette. Vi kunne utviklet en native applikasjon, en hybrid applikasjon eller en web-applikasjon.

En native applikasjon er et program som er utviklet for å bli brukt på en spesifikk plattform, som iOS eller Android. Fordelen med en native applikasjon er at man har muligheten til å utnytte enhetens programvare og maskinvare fullt ut, men ulempen er at man må utvikle en applikasjon for iOS og en for Android (Guilty, 2019). I nyere tid har det kommet noen alternativer som gjør at man kan skrive native kode, som fungerer på flere plattformer. Disse alternativene er blant annet React Native og Flutter.

React har mulighet til å bli brukt som native mobilkode for iOS og Android. Dette verktøyet gir oss derfor muligheten til å skrive i samme språk som i web-applikasjonen, samtidig som man får tilgang til native plattformfunksjoner. Dette gjennomføres ved at JavaScript-koden i React blir oversatt til Java for Android og C++ for iOS (Nevercode, u.å.).

Flutter er et annet kryss-plattform-alternativ for iOS og Android. Her blir koden

kompilert til native ARM-maskinkode og dermed gir Flutter full tilgang til native plattformfunksjoner i iOS og Android (Flutter, 2020).

Det andre alternativet var å lage en web-applikasjon, slik at vi bare har en kildekode som er responsivt for både mobil og web. Fordelen med dette alternativet er at utviklingen går raskere, men ulempen er at man ikke kan utnytte funksjonene på mobil fullt ut.

Det tredje alternativet var å lage en hybrid-applikasjon, som lages ved hjelp av CSS, Javascript og HTML. I motsetning til en web-applikasjon, kjører en hybrid-applikasjon i en egen container, og utnytter enhetens nettlesermotor (men ikke nettleseren) for å gjengi HTML og behandle JavaScript lokalt (Griffith, u.å.).

3.1.3 Alternative løsninger for web

Oppdragsgiver ønsket at løsningen skulle være tilgjengelig i web-applikasjon, slik at man ikke er avhengig av å bruke en app, som kun er tilgjengelig for smarttelefoner. For at prosjektet skulle være mulig å gjennomføre, måtte løsningen være dynamisk. Det er mange alternative løsninger for hvordan vi kunne laget web-applikasjonen, men vi så for oss tre forskjellige måter vi kunne gjennomføre dette.

Det første alternativet var å lage web-applikasjonen i Flutter for Web. Dette alternativet baserer seg på Flutter, på samme måte som Flutter er for mobil, hvor man lager web-applikasjonen slik man lager applikasjoner for mobil (Flutter, 2020). Ulempen med dette alternativet er at Flutter for Web er fortsatt i betatesting og kan derfor være noe ustabil.

Det andre alternativet var å lage web-applikasjon i React. React er et JavaScript-bibliotek som tillater en å lage komponentbaserte web-applikasjoner og brukergrensesnitt (React Docs, u.å.). JavaScript er brukt av flere høy-profils

applikasjoner og er et dynamisk språk. Språket er klient-basert og kjører i nettleseren (Mozilla, u.å.).

Det tredje alternativet var å lage web-applikasjonen med PHP. PHP er et server-basert skriptspråk og språket inneholder HTML-kode som kan utføre dynamiske endringer. Forskjellen mellom PHP og klient-baserte skriptspråk som for eksempel JavaScript, er at PHP-kode kjører på serveren, som igjen genererer HTML-kode som sendes til klienten (The PHP Group, u.å.).

3.1.4 Diskusjon av alternativene

Når det gjelder alternativene for mobil, var kravet fra oppdragsgiver at applikasjonen skulle fungere både på iOS og Android. For å ha en kodebase og slippe å måtte kode to forskjellige applikasjoner, sto valget mellom en native applikasjon (Flutter eller React Native), web-applikasjon eller en hybrid applikasjon. Med tanke på funksjonsbegrensningene som eksisterer tilknyttet en web-applikasjon og hybrid applikasjon, valgte vi å gå for en native applikasjon. React Native og Flutter har mange like egenskaper. Den største forskjellen er at React Native skrives i JavaScript, mens Flutter skrives i Dart. Hvilke Software Development Kit (SDK) som ble valgt, er tatt med hensyn på egne erfaringer og kjennskap til de forskjellige SDKene.

For webapplikasjonen var alle alternativene ganske like, hvor de største forskjellene er rammeverk og programmeringsspråk. Det er fordeler og ulemper med alle alternativene. Ettersom vi vurderte Flutter for mobildelen av prosjektet, hadde dette vært veldig enkelt å bruke for webapplikasjonen også (kodebasene kan se nokså like ut). Problemet er at Flutter for Web er fortsatt i betatesting (Google har ikke lansert offisiell versjon enda) og grunnet dette, er det en del problemer og “bugs” med dette alternativet.

JavaScript-biblioteket React, er et godt alternativ for gruppen. Dette er fordi JavaScript kjører på klient-siden i motsetning til PHP som kjører på server-siden. Dette gjør at vi enkelt kan bygge en dynamisk web-applikasjon som har støtte til offline-modus om man bruker teknologier som eksempelvis Firebase.

Fordelen med å bygge en web-applikasjon ved hjelp av PHP, er at det er godt støttet og relativt enkelt og raskt å koble opp mot kjente relasjonsdatabaser. Ulempen med PHP er at det er server-basert, og av den grunn måtte vi ha satt opp ett eget miljø.

Alternativene for back-end var svært forskjellige. Hvert enkelt alternativ har fordeler og ulemper som ville påvirket prosjektet i stor grad. Ved å bruke en enkel relasjonsdatabase i form av MySQL eller PostgreSQL, ville prosjektet ha blitt forenklet i stor grad med tanke på implementering. Dette ville dog kommet på bekostning av brukeropplevelsen, med tanke på flyten i systemet (oppdatering av informasjon ville ikke vært så dynamisk).

ASP.NET Core har også sine fordeler og ulemper. Etersom ASP.NET Core er utviklet av Microsoft, er det primært fokusert på Microsoft sitt teknologiske miljø. Dette vil utelukke en viss funksjonalitet på iOS sin side. ASP.NET Core har på den andre siden et godt miljø, med et sterkt utviklet rammeverk, som er enkelt og intuitivt å bruke opp mot systemet vårt.

Sist nevnt har vi Firebase. Firebase er en konkurrent mot ASP.NET. Firebase tilbyr mange av de samme funksjonalitetene som ASP.NET tilbyr, men har en viss forenkling av mange av dem. Dette gjør Firebase til en mer intuitiv og simpel plattform å bruke. På den andre siden er Firebase en plattform som stadig oppdateres, og nye funksjonaliteter blir lagt til, og gamle blir endret. Dette kan føre til at prosjektet jevnlig må oppdateres for å holde systemet gående.

3.2 Valgt løsning

De valgte teknologiene ble utvalgt på grunnlag av fremtidig produksjon av systemet, samt sammenkoblingene mellom disse teknologiene. Dette vil si moderne teknologier, med støttede samfunn bak seg, som har stor sannsynlighet for å bli brukt mange år fremover.

For den nettbaserte delen av prosjektet ble React valgt som foretrukket rammeverk. React er et utbredt JavaScript-bibliotek og er komponentbasert, som vil si at man bygger enkapsulerte komponenter som håndterer sin egen tilstand (React Docs, u.å.). Dette gjør at det er relativt enkelt å bygge web-applikasjonen slik den er basert fra UX-designet, hvor enkelte deler av web-applikasjonen er statiske, mens andre er dynamiske. React er laget av Facebook og med dette følger det med mange open-source biblioteker, slik at tilleggsfunksjoner som skal være med (nå og i framtiden) er enkelt å passe inn. Firebase og React har også god støtte med hverandre, noe som er kritisk for det ferdige produktet. Ved agile-utvikling er det også raskt og intuitivt å introdusere nye egenskaper til web-applikasjonen, da dette er veldig bra lagt opp når man skriver React.

For mobilapplikasjonen ble Flutter valgt som vår foretrukne teknologi. Flutter kan tilby native opptreden sammenkoblet med intuitivt og fremtidig programvareutviklingssett. Flutter har også cross-plattform muligheter, som gjør det mulig å programmere for både iOS og Android samtidig. Ettersom mobil- og nettapplikasjonsutviklingsplattformen Firebase også er laget av Google, vil dette si at de to teknologiene er veldig kompatible. Dette vil føre til en kontinuerlig og trygg overføring av data mellom Flutter og Firebase.

For back-end delen av prosjektet var det essensielt at prosjektet var skalerbart, samt at det var lett å endre strukturen og den lagrede dataen på databasen på en effektiv og grei måte. Dette er noe Firebase kan tilby. Plattformen er godt støttet opp imot både Flutter og React med sine autentiserings- og lagringsfunksjoner.

3.3 Prosjektmetodikk

Prosjektet følger en iterativ prosjektmetodikk som går ut på at arbeidsomfanget deles opp i flere mindre leveranser, som utføres innen korte tidsrammer. Disse tidsrammene kalles iterasjoner. Gjennom disse iterasjonene bygges løsningen stegvis etter konsultasjon med oppdragsgiver. Fordelen med denne prosjektmetodikken er at man avdekker designproblemer tidlig og kan tilpasse seg etter oppdragsgivers tilbakemeldinger (Næss, 2020).

3.3.1 Utviklingsmetodikk

For å opprettholde en god arbeidsflyt i prosjektet, anvender vi Trello for administrasjon av prosjektet. Trello er et samarbeidsverktøy som brukes for å få oversikt over prosjektets oppgaver, der man kan fordele oppgavene til forskjellige utviklere. Dette er et nettbasert program for bruk av utviklingsmetodikken Kanban.

Kanban er et arbeidshåndteringssystem som er ment for å visualisere prosjektets oppgaver. Den består av seks prinsipper som hjelper gruppen med å administrere oppgaver og optimalisere det løpende arbeidet (Högstrand, 2019).

Gruppen har fulgt flere av Kanban-systemets prinsipper. For det første har vi en visualisering av arbeidsprosessen (trinnene som arbeidet flyter gjennom) på en Kanban-tavle på Trello. Vi har delt tavlen inn i fire kolonner:

- En kolonne for oppgaver som skal gjennomføres.
- En kolonne som representerer oppgaver som er under arbeid.
- En kolonne for oppgaver som skal testes.
- En kolonne for oppgaver som er godkjent og ferdig.

For å unngå at ikke oppgaver gjøres samtidig, har vi begrensning av pågående arbeid. Det vil si at vi har angitt hvor mange aktiviteter man kan foreta samtidig i hver kolonne.

Ettersom gruppen har hatt en felles Kanban-tavle, har vi satt tydelige regler på hvordan vi skal håndtere, legge til og endre en oppgave. For å opprettholde orden har vi hatt hyppige møter hvor vi har diskutert ulike oppgaver og gitt tilbakemeldinger på tidlige oppgaver. Dette har også gjerne resultert i at gruppen gjerne har prøvd andre løsninger på et problem eller at flere har måttet samarbeide med en oppgave for å fullføre den.

Gruppen anvender også Git som versjonskontrollsystem. Dette er et verktøy som brukes for å dele kildekoden innad i gruppen og holde orden på riktig versjon av koden. Når en utvikler har gjennomført en oppgave “pushes” koden opp på Git. Da kan de andre på gruppen gå gjennom koden, for å se hva som har blitt gjennomført. På denne måten kan vi unngå små feil, som lett kan oppstå. Når koden er gjennomgått og godkjent blir den sammenflettet med hovedkodebasen, som blir den nye versjonen av koden.

3.3.2 Prosjektplan

Prosjektplan ligger som vedlegg.

3.3.3 Risikoanalyse

Risiko er en funksjon av en konsekvens (K) og en sannsynlighet (S) for at noe skal skje, og man finner den totale risikofaktoren (RF) ved å gange disse to sammen. I risikoanalysen benyttes det en skala fra 1 til 5, hvor 1 betyr at det er ubetydelige konsekvenser dersom risikoen inntreffer og ytterst usannsynlig at risikoen inntreffer. Øverst i skalaen er tallet 5 som betyr at det vil få katastrofale følger dersom risikoen inntreffer, og at det nesten er sikkert at risikoen vil inntreffe (Skjølsvik & Voldsund, 2017). I tabell 1 har vi gjennomført en risikoanalyse for prosjektet.

Tabell 1: Risikoanalyse

Suksessfaktorer	S	K	RF	Tiltak	Interessent	Fase
For liten tid for å ferdigstille applikasjonene	2	4	8	Planlegge arbeidsoppgavene, slik at vi har kontinuerlig framgang	Gruppen og oppdragsgiver	Planlegging
Data i databasen blir håndtert på feil måte	2	4	8	Gjennomfører grundige tester for å passe på at håndteringen av data skjer på riktig måte	Gruppen	Planlegging/ gjennomføring
Teknisk svikt under funksjonstestene	3	2	6	Før vi eventuelt har funksjonstest, må vi være sikre på at vi har testet applikasjonen nok	Oppdragsgiver	Gjennomføring
Applikasjonen er ikke brukervennlig	2	2	4	Gruppen skal ha funksjonstester opp mot noen skoler i Oslo, slik at man får tilbakemeldinger fra målgruppen	Gruppen og oppdragsgiver	Gjennomføring
Får ikke gjennomført funksjonstest på grunn av covid-19	4	3	12	Hvis vi ikke får gjennomført funksjonstest som planlagt, vil det gjennomføres en test mot personer i gruppen	Gruppen og oppdragsgiver	Gjennomføring

3.3.4 Risikohåndtering

For hver av risikoene er det viktig for prosjektet å ha risikoreduserende tiltak. Det finnes fire hovedtyper risikostrategier for å bestemme risikoreduserende tiltak. Den første strategien er å avdempende. Det vil si at man reduserer risikoen for at risikoen inntreffer, eller at man reduserer konsekvensen hvis den oppstår. Den andre strategien er å overføre, som vil si at man flytter den identifiserte risikoen til en annen gruppe, som er bedre rustet til å håndtere dem. Den tredje strategien er å unngå, som vil si at man velger en alternativ løsning som eliminerer risikoen. Den siste strategien er å akseptere. Det vil si at man aksepterer at risikoen er til stede, og man er innforstått med at det ikke er mulig å dempe, overføre eller unngå den (Skjølsvik & Voldsund, 2017).

Før liten tid til å ferdigstille applikasjonene - Avdempende strategi

Måten vi reduserer at denne risikoen inntreffer, er å planlegge arbeidsoppgavene grundig. I samarbeid med oppdragsgiver har vi blitt enig om en gjennomførbar MBP.

Data i databasen blir håndtert på feil måte - Avdempende strategi

For å redusere denne risikoen gjennomføres grundige tester, slik at vi kan se om dataene håndteres på riktig måte.

Teknisk svikt under funksjonstest - Aksepterende strategi

Før en eventuell funksjonstest må applikasjonen testes grundig, slik at sannsynligheten for at feilen oppstår er minimal. Vi vet ikke hvordan systemet vil opptre når det er mange brukere, og derfor er det her muligheter for å få testet systemet og håndtere feilene i ettertid.

Applikasjonen er ikke brukervennlig - Overførende strategi

Gruppen har et tett samarbeid med to UX grupper i Oslo. Disse gruppene tar hånd om flyten i applikasjonen og har erfaring innen brukergrensesnitt. I tillegg skal de gjennomføre en funksjonstest på designet opp mot skoler i Oslo, for å se hvor brukervennlig det er.

Får ikke gjennomført funksjonstest på grunn av covid-19 - Aksepterende strategi

Får vi ikke gjennomført funksjonstest opp mot en skole i Oslo grunnet covid-19, gjennomfører vi først en test mot prosjektets medlemmer. Deretter hvis det lar seg gjøre, vil oppdragsgiver gjennomføre en test mot en liten gruppe personer, som er i den riktige målgruppen. Grunnen for at denne risikoen har høy risikofaktor er at vi ikke får observert hvordan elevene reagerer på applikasjonene.

3.4 Evalueringsplan

Måten prosjektet skal evaluere resultatet av oppgaven, er ved hjelp av forskjellige testmetoder. På både mobil- og web-applikasjonen skal det gjennomføres enhetstesting. I tillegg vil det bli gjennomført integrasjonstester på mobilapplikasjonen. En enhetstest er testing av de minste komponentene i programmet, som vil si at man tester metodene i en klasse for å se at de har den ønskede oppførselen (Universitetet i Oslo, 2010). En integrasjonstest er testing av flere komponenter i programmet, som brukes for å se om en transaksjon eller en større operasjon utføres (Universitetet i Oslo, 2010). I React gjennomføres en enhetstest ved at man tester en og en komponent, ved hjelp av et verktøy som kalles Riteway. I Flutter gjennomføres enhetstest på spesifikke metoder, ved hjelp av det innebygde test-verktøyet, mens integrasjonstester gjennomføres ved bruk av “flutter_driver”-package.

Prosjektet skal også gjennomføre to funksjonstester, også kalt brukertester. Ved hjelp av disse testene får vi svar på om brukeren kan bruke applikasjonene på en tilfredsstillende måte, eller om vi må gjennomføre endringer. I tillegg får vi testet at systemet ikke inneholder feil som må rettes opp. Denne testmetoden vil også bli anvendt innad i gruppen for å sjekke at applikasjonen har den ønskede oppførselen før vi gjennomfører funksjonstester mot eksterne brukere.

Grunnet prosjektets tidsramme har vi ikke tilstrekkelig med ressurser for å undersøke hvorvidt applikasjonene stimulerer til kreativ lesing og skrivning. Derfor baseres resultatet/måloppnåelsen til prosjektet på observasjoner som er gjort under de ulike funksjonstestene.

4. DETALJERT DESIGN

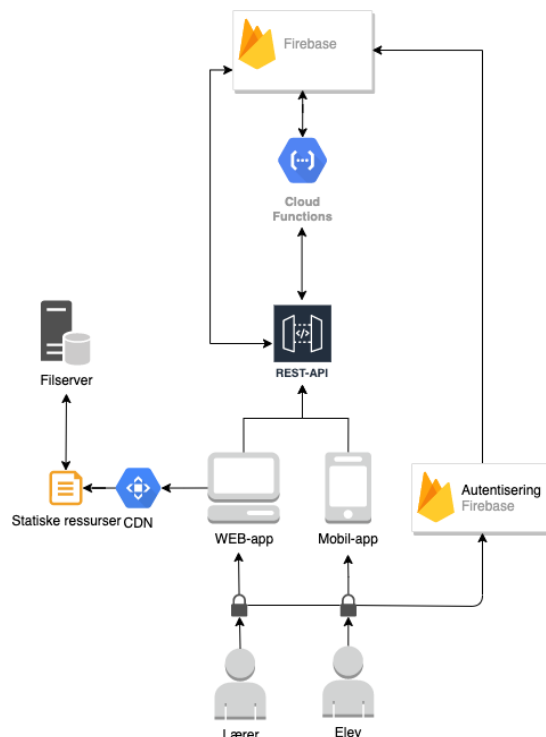
I dette kapitlet tar vi for oss designet av produktet. Først vil vi beskrive arkitekturen av produktet i detalj, og deretter vil vi presentere de viktigste funksjonene fra det grafiske grensesnittet.

4.1 Bakgrunn

Før prosjektstart var det avtalt med oppdragsgiver at vi skulle ha ansvar for å utvikle front-end delen av prosjektet. Som forklart i punkt 2.1 var avtalen at UX-gruppene i Oslo skulle stå for brukeropplevelsen, det grafiske grensesnittet og håndtering av databasen. Dette skulle da være klart til 9. mars, slik at vi kunne ha fokus på utviklingsdelen og raskt kunne ha et system oppe å kjøre. Når datoen for oppstart kom, var verken det grafiske grensesnittet eller databasen klar. For å komme i gang med arbeidet måtte vi derfor lage vårt eget grafiske grensesnitt basert på UX-kriteriene. I tillegg ønsket oppdragsgiver at vi skulle ta over oppsettet med databasen, slik at vi raskest mulig kunne komme i gang med arbeidet.

4.2 Arkitektur

Den overordnede arkitekturen av systemet er bygd opp i samarbeid med oppdragsgiver og er illustrert i figur 1. Både web- og mobil-klienten må gjennom en autentisering i firebase for å få tilgang til applikasjonene. Statistiske ressurser som bilder og lokaliserings-strenger er lagret hos klienten, slik at applikasjonene er responsiv. For å få dynamiske endringer på klientene brukes også Cloud Functions. Cloud Functions er et server-løst rammeverk som lar oss kjøre back-end-kode som svar på hendelser utløst av funksjoner og HTTPS-forespørsler (Google, 2020). På denne måten kan man lytte etter oppdateringer i databasen og gjennomføre endringer deretter.



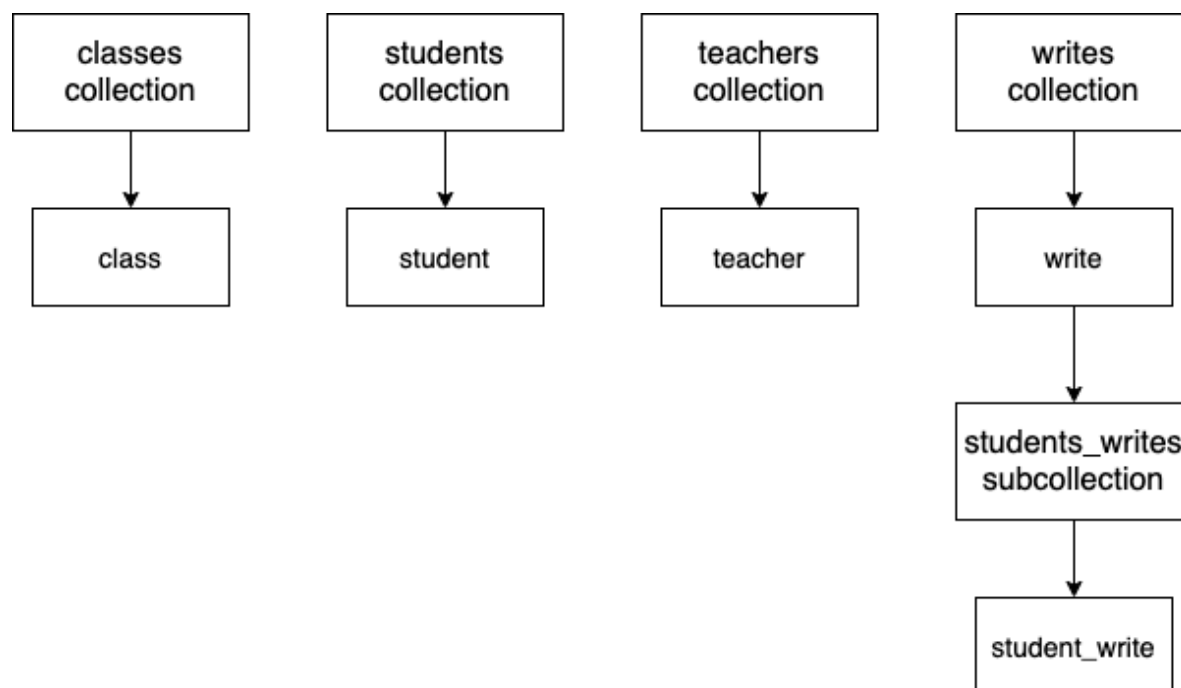
Figur 1: Systemarkitektur

4.2.1 Arkitektur av databasen

Som forklart i punkt 3.2 er databasen opprettet i Cloud Firestore (fra Firebase). Cloud Firestore er en fleksibel og skalerbar database for mobil og web. Det er en sky-basert NoSQL database som mobil- og webapplikasjoner kan få tilgang til direkte gjennom SDKer (Google, 2020).

Vi anvender også som tidligere forklart i punkt 4.2 Cloud Functions. Dette anvendes for å få dynamiske endringer på klientene basert på hendelser i databasen. I prosjektet har vi to slike funksjoner. Den ene håndterer ID-generering av forskjellige brukere, mens den andre brukes for å håndtere logikken under selve “written”. Det vil si at den håndterer utdeling av andre elevers historier, slik at en person ikke skal skrive videre på sin egen historie, men fortsette på det en annen elev har skrevet.

For nye brukere som registreres i systemet, bruker vi Firebase Authentication. Her lagres informasjon om brukeren som opprettes, eksempelvis passord og e-post. Dette er en adskilt del fra Firestore, noe som gjør at ved opprettelse av nye brukere opprettes brukeren også som et dokument i Firestore (bortsett fra passord). Dette dokumentet blir lagret under “student collection” eller “teachers collection” basert på om man registrer seg som lærer eller student, som illustreres i figur 2.

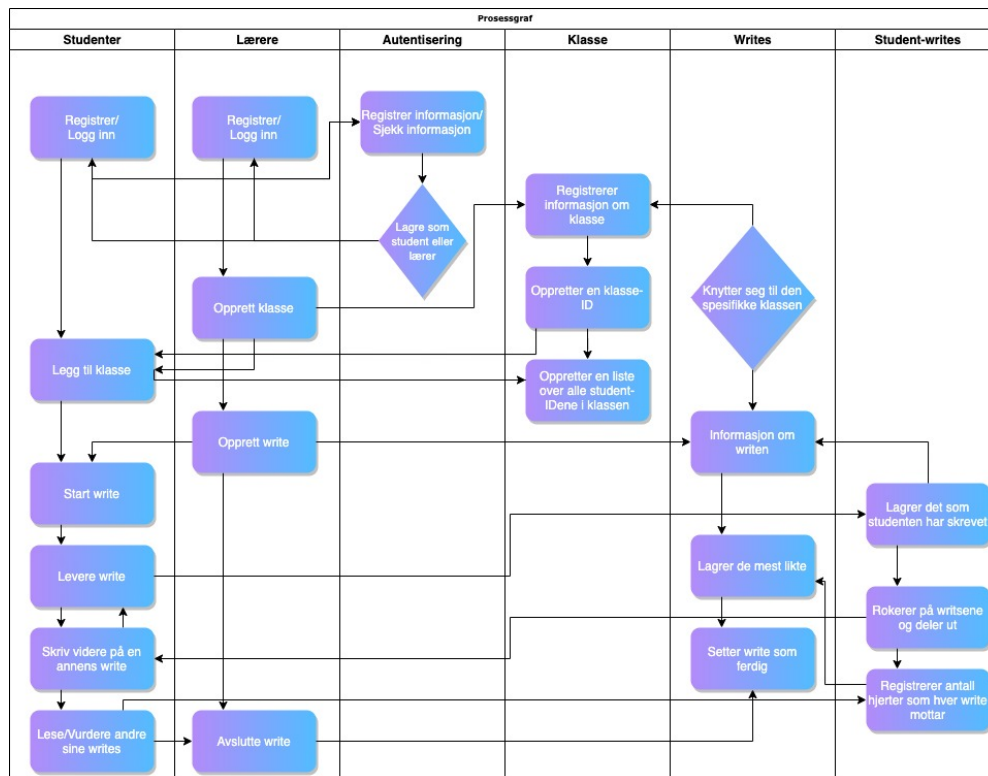


Figur 2: Databasestruktur

Videre følger det en “Classes collection” som er en samling av forskjellige klasser. Disse inneholder forskjellige IDer, som studenter kan bruke for å legge seg til en klasse. Når en elev har lagt seg til klassen, opprettes en liste med de forskjellige student-IDene i denne klassen.

Til slutt følger det en “writes collection” som er en samling av forskjellige “writes”. En “write” opprettes i en klasse og hører da til i den klassen den er opprettet i. I “writes collection” er det en subcollection som heter “student_writes collection” hvor elevenes “student_writes” ligger. Hver “student_write” har en student-id, slik at eleven som har skrevet den, eier den. Ved flere iterasjoner, omrokeres eierskaps-IDene, slik at “student_writen” får en ny eier, og da kan en ny elev fortsette på teksten.

For å få en bedre oversikt over databasen, har gruppen laget en prosessgraf illustrert i figur 3. Her er stegene fra man registrer/logger seg inn, enten som student eller lærer, til man er ferdig med en “write”.



Figur 3: Prosessgraf

4.2.2 Arkitektur av mobilapplikasjon

Mobilapplikasjonen er som nevnt tidligere utviklet ved hjelp av Flutter SDK. Flutter er bygd opp på tilstandsprinsippet. Dette vil si at applikasjonen holder en tilstand, og for hver gang tilstanden blir oppdatert vil også brukergrensesnittet bli tegnet opp på nytt. På denne måten kan appen lytte til endringer i databasen og automatisk oppdatere brukergrensesnittet med den nye dataen (Flutter, 2020).

I sammenheng med tilstandshåndtering valgte vi å ta i bruk “provider”- package som gjør det mulig for widgets å lytte til oppdateringer fra sine foreldre-widget i widget-treet. Dette er noe som gjør det mulig for læreren å gjøre endringer i “written”, selv om elevene allerede har startet (Flutter, 2020).

Vi bruker også “provider”- package til lokalisering av appen. Appens hoved-widget lytter til telefonens systemspråk, og endrer språket på alle widgetene basert på dette. Vi har i all hovedsak bare lagt til to språk, som er norsk og engelsk, men systemet er tilrettelagt for å legge til flere språk.

Autentiseringen for appen fungerer på samme måte for web-delen som for mobildelen. Vi utnytter Firebase sine egne autentiseringstjenester. Firebase har utviklet sine egne packages for Flutter som gjør prosessen for innlogging, utlogging og sletting av kontoer veldig intuitivt.

Lesing og skriving mot databasen håndteres ved å ta i bruk Firestore sine egne packages for Flutter. I sammenheng med Firestore sin package utnytter vi også “provider”- package for å håndtere databaselytting over widgets, som nevnt ovenfor. Som vist på figur 4, blir skriving mot databasen håndtert ved hjelp av Cloud Functions. I den nåværende utgaven bruker Cloud Functions kun for “write-collection”. Måten dette fungerer på er at når et dokument i “write-collectionen” blir oppdatert vil en funksjon bli trigget på det oppdaterte dokumentet.

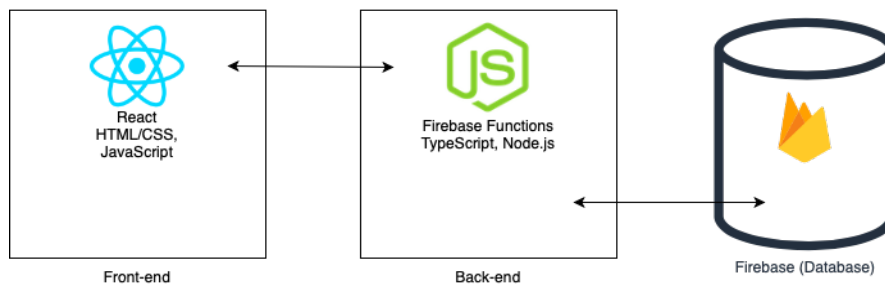


Figur 4: Arkitektur mellom mobilapplikasjon og database

4.2.3 Arkitektur av webapplikasjon

Webapplikasjonen er bygd som en “Single Page Application” (SPA), som vil si at applikasjonen er sammensatt av individuelle komponenter som kan erstattes eller oppdateres uavhengig, uten å måtte oppdatere hele siden. Dermed trenger ikke siden å lastes for hver brukerhandling, noe som sparer båndbredde og ikke legger inn eksterne filer hver gang siden lastes inn. Hensikten med dette er å laste inn den påfølgende siden veldig raskt, sammenlignet med tradisjonelt forespørsel-svar syklus (Jadhav, Sawat, Deshmukh, 2015). I en SPA er det bare tre skript-filer og disse kan lastes opp til enhver statisk hosting server, i vårt tilfelle Firebase Hosting. Etter første gang index.html er lastet inn, blir ikke mer HTML sendt over nettverket, men data blir forespurt fra serveren (Angular University, 2020).

Ut fra figur 5, er selve webapplikasjonen front-end delen av figuren hvor React er rammeverket. Fra klienten sendes forespørsler om data enten direkte til databasen, eller via back-enden ved større operasjoner. Dette kommer også litt an på hvilken type operasjon som utføres.



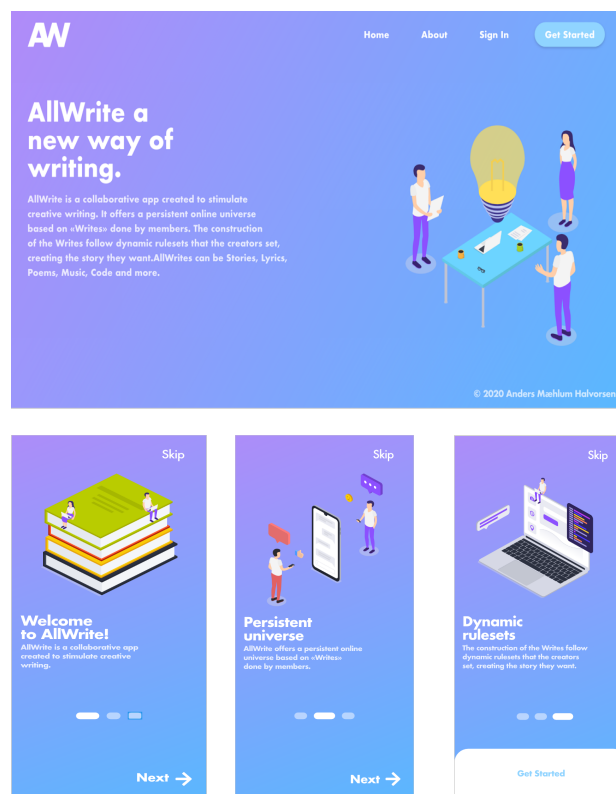
Figur 5: Arkitektur mellom web-applikasjon og database

Webapplikasjonen følger komponent-basert arkitektur slik React er bygd opp, og hver komponent passer på sin egen tilstand (React Docs, u.å.). Hvis man ser på web-applikasjonen i det større bildet, kan det tenkes at det er en arkitektur lignende MVC (Model View Controller). I dette tilfellet har gruppen brukt Firestore Hooks (data fra Firestore) som modellen, React (JavaScript) som selve viewet og React Hooks som kontrolleren (Hooks håndterer tilstand i React) - men ettersom dette foregår på klientsiden, er MVC en arkitektur som er noe vanskelig å implementere fullt ut.

4.3 Grafisk grensesnitt

Det grafiske grensesnittet ble planlagt slik at flyten i mobilapplikasjonen og webapplikasjonen skulle føles mest mulig lik. Derfor er det brukt samme fargekombinasjon og skrifttype i begge applikasjonene, som illustrert i figur 6. I tillegg ønsket vi at utseende skulle appellere til unge brukere. En koreansk undersøkelse utført i 2016 viser at grafiske farger i et spill har stor innflytelse på følelser og psykologi hos barn. Derfor må den som planlegger grafikken til spillet velge farger som tar hensyn til dette (Lee, Cho, Sim & Lee, 2016). Fargekombinasjonen vi har valgt er blå og lilla. Blå representerer indre trygghet og selvtillit, mens lilla er relatert til fantasien og

spiritualiteten (Lee et. al., 2016). Disse fargene bygger opp under prosjektets mål, som fokuserer på at systemet skal stimulere til kreativ lesing og skriving.



Figur 6: Illustrasjon web- og mobilapplikasjon

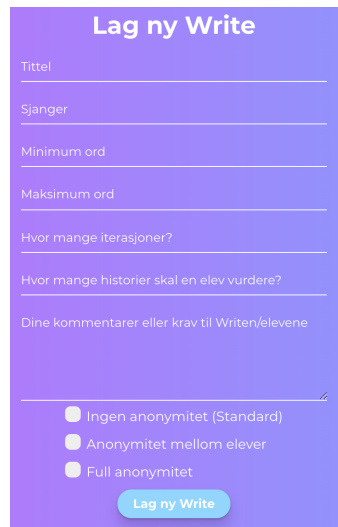
Vi har valgt ut noen av de viktigste funksjonene fra det grafiske grensesnittet, og vil forklare de nærmere. Den første funksjonen er illustrert i figur 7 og er en unik QR-kode som elevene kan skanne for å bli med i en klasse. Denne genereres i web-applikasjonen ved hjelp av en åpen kildekode i React, slik at vi får IDen som en QR-kode. Som en alternativ løsning kan man også skrive inn IDen til klassen manuelt, hvis for eksempel kameraet på telefonen er dårlig eller ikke fungerer.

Skann QR-kode for å delta i klassen: Testklasse AllWrite



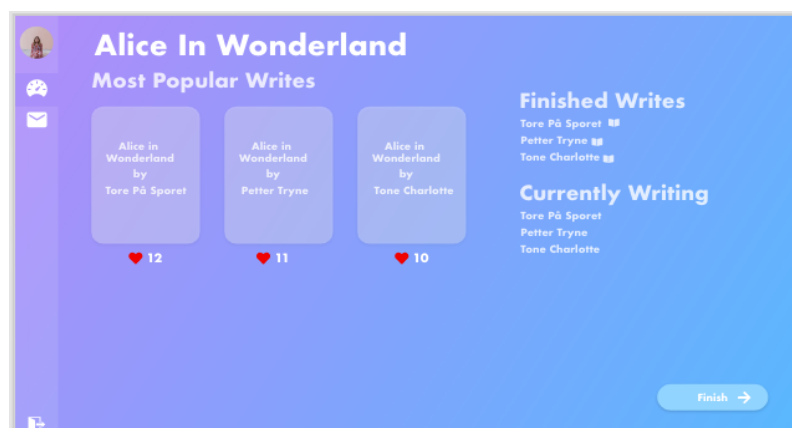
Figur 7: QR-kode

Den andre funksjonen er illustrert i figur 8, som viser hvordan en lærer oppretter en “write”. Her kan læreren lage overordnet tittel, velge hvilke sjanger det skal skrives innen, minimum og maksimum antall ord, hvor mange iterasjoner som skal gjennomføres, hvor mange historier hver elev skal vurdere, retningslinjer til “writen”, og hvilken anonymitetsgrad “writen” skal ha. Alle disse inputene legger føringer for hvordan “writen” skal fungere.



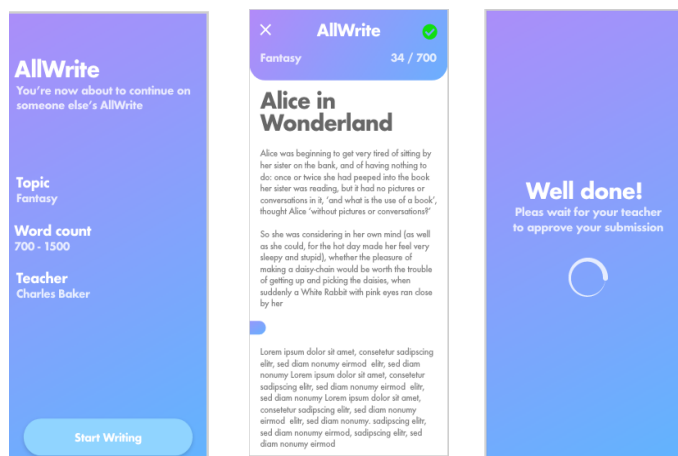
Figur 8: Opprettelse av write

Den tredje funksjonen er at læreren har et live dashboard med full oversikt over hvem som fortsatt holder på å skrive, og hvem som er ferdig å skrive. I tillegg kan læreren lese det som de forskjellige elevene har skrevet. Til slutt har også læreren en live oversikt over hvilke “writen” som har fått flest hjerter fra elevene. Dette er illustrert i figur 9.



Figur 9: Illustrasjon over lærerens dashboard

Parallelt med lærerens dashboard på web-applikasjonen, får alle elevene opp en aktiv ”write” på mobilapplikasjonen. Før man starter kan eleven lese føringene til ”writen”, og deretter starte å skrive, som illustrert i figur 10. Når de har skrevet nok ord i forhold til minimums- og maksimums-grensen, og er fornøyd med teksten, leverer de og venter på at de andre elevene i klassen skal fullføre. Basert på hvor mange iterasjoner læreren har valgt når ”writen” ble opprettet, skal eleven fortsette på andres historier tilsvarende ganger. For eksempel hvis en lærer har valgt 3 iterasjoner, er første iterasjon starten på fortellingen, andre iterasjon hoveddel, og tredje iterasjon avslutning.



Figur 10: Illustrasjon av flyten under en write på mobilapplikasjonen

Etter at elevene er ferdig med alle iterasjonene, får de utdelt et visst antall tekster som de skal vurdere. Her kan de altså dele ut hjerter basert på hvor godt de likte historien. Når de er ferdig å dele ut hjerter fullfører de og ”writen” ferdig.

5. EVALUERING

I dette kapitlet forklares hvilke evalueringsmetode som er brukt i prosjektet, hvordan de er gjennomført og resultatene som vi fikk fra de forskjellige evalueringene.

5.1 Evalueringsmetode

For å evaluere resultatet har vi som planlagt i evalueringsplanen gjennomført enhetstesting, integrasjonstest og funksjonstester. Enhetstestene ble gjennomført på forskjellige metoder i applikasjonene for å passe på at de hadde den ønskede oppførselen. Disse testene ble gjennomført underveis når metodene ble opprettet, slik at vi fikk testet de før vi anvendte de i systemet. På mobilapplikasjonen ble det gjennomført en enhetstest på metoden som skulle håndtere studentenes “write”-data. Dette gjorde vi for å sjekke at riktig data skulle bli sendt til databasen, samt at håndteringene kunne bli utført på en tilstrekkelig måte.

Det ble også gjennomført enhetstest på web-applikasjonen på metoden som sorterer elever på dashbordet til læreren. Denne metoden skal for hver iterasjon sortere ut hvilke elever som er ferdig og hvilke som fortsatt skriver. Dette er en viktig metode for at læreren skal kunne holde orden på hvilke elever som er ferdig, og hvem som ikke er det.

I tillegg har vi gjennomført integrasjonstester på mobilapplikasjonen der vi har testet flere komponenter sammen for å se at en transaksjon eller at en større operasjon utføres. Dette har vi gjennomført på innloggingsprosessen og innlastingen av dashbordet. Etter å ha gjennomført testen oppdaget vi en feil ved innlastingen av dashbordelementer. Dette ble dog fikset etter endringer i innlastning-strukturen.

For å få et produkt som er brukervennlig, har vi gjennomført funksjonstester for å teste funksjonaliteten i koden som helhet. Det er også viktig å påpeke at denne testen ikke

gjennomføres for å teste hvor gode testdeltakerne er å bruke løsningen, men skal avgjøre hvor enkelt og intuitivt produktet er å bruke (Foss-Pedersen, 2017). Funksjonstestene som har blitt gjennomført i prosjektet er interne tester, tester med oppdragsgiver og andre gruppemedlemmer og en test med alle medlemmene på prosjektet. Det var også planlagt en ekstern test mot en skole i Oslo, men grunnet komplikasjoner med covid-19 hadde de ikke anledning til å delta i denne testen før etter sommeren.

Vi har anvendt en kvalitativ metode for å observere samhandlingen mellom testdeltakerne og applikasjonene. Dette har vi gjennomført ved å anvende en deltakende observasjon, som går ut på at man tar del i samhandlingen i den gruppen man skal undersøke (Vårdal, 2020). I tillegg valgte vi at observasjonen skulle være åpen, slik at testdeltakerne visste at de ble observert. Vi anvendte åpen observasjon fordi vi mente at det ikke ville påvirke deres adferd, på bakgrunn av at testdeltakerne ikke kunne vite på forhånd hva som var riktig og galt å gjøre underveis.

5.2 Evalueringresultat

Gruppen har gjennomført kontinuerlige funksjonstester innad i gruppen for å se at applikasjonen har den ønskede oppførselen. Disse testene var nyttig fordi vi fikk innsikt i hvordan Cloud Functions fungerte. På denne måten avdekket vi en stor feil i måten vi hentet og oppdaterte data som gjorde at hovedfunksjonaliteten i programmet krasjet. Disse testene avdekket også små feil med systemet som vi fikk fikset opp i.

Det ble deretter gjennomført en funksjonstest med oppdragsgiver og hans samarbeidspartner. Denne testen avdekket små forbedringer samt en misforståelse mellom gruppens løsning og løsningen oppdragsgiver originalt ville ha.

Mot prosjektets slutt ble det også gjennomført en funksjonstest med alle medlemmene på prosjektet. Siden vi ikke kunne gjennomføre en brukertest opp mot en skole i Oslo,

anvendte vi dette som vår hovedfunksjonstest. Selv om mange av medlemmene hadde jobbet med UXet for applikasjonen, hadde de ikke anvendt vårt grafiske grensesnitt og vi kunne derfor teste hvor intuitivt og enkelt produktet var å bruke. Uten å forklare hva de skulle gjøre, skulle vi se hvor enkelt dette var å forstå. Resultatene vi fikk fra denne testen vises i tabell 2.

Tabell 2: Resultat fra hovedfunksjonstest

Mangler	Detaljert forklaring	Rettelse
Applikasjonen måtte forklare hva brukeren skulle gjøre tydeligere	<ul style="list-style-type: none"> - Det var ikke intuitivt at en bruker skulle trykke på et “+” tegnet for å delta i en klasse - Det var ikke intuitivt at en elev skulle trykke på boksen under aktive “writes” for å starte 	For å rette opp i dette er det lagt inn forklarende tekst i tillegg til “+” tegnet. Det samme er gjort for å starte “written”.
Lærer hadde ikke oversikt over hvem som hadde fullført hver iterasjon	<ul style="list-style-type: none"> - For hver iterasjon, oppdaterte ikke listen over hvem som var ferdig og hvem som ikke var ferdig 	Denne er fikset slik at det for hver iterasjon kan sees hvilke elever som er ferdig og hvilke som ikke er.
En bruker av mobilapplikasjonen kunne ikke bla for å lese tekstene til de andre brukerne	<ul style="list-style-type: none"> - Når en elev skulle vurdere de andre elevene sine tekster var det ikke mulig å bla for å lese hele teksten 	Denne siden på mobilapplikasjonen er endret på slik at det er mulig å bla
En lærer kunne trykke to ganger for å opprette en identisk “write”.	<ul style="list-style-type: none"> - Når en lærer skulle opprette en “write”, var det en feil i systemet som gjorde at hun kunne opprette flere. 	Her har vi endret slik at skjemaet forsvinner etter opprettelse av ny “write”

6. RESULTATER

Selv om funksjonstesten vi hadde med medlemmene i prosjektet avdekket noen mangler som vi måtte rette opp, observerte vi at deltakerne hadde det gøy mens de skrev. Historiene som vi satt igjen med til slutt fikk folk til å le. Under funksjonstesten kjørte systemet som det skulle, uten noen form for store feil. Manglene som vi fikk fra funksjonstesten ble i tillegg rettet opp i, så systemet skal nå være solid for videre testing og utvikling. Basert på observasjonene og systemet i helhet, kan vi si at målet med å utvikle en online opplevelse som stimulerer til kreativ lesing og skriving er oppnådd. Det må selvfølgelig gjennomføres grundigere observasjoner mot målgruppen, men ut ifra forutsetningene lover dette godt for framtidig utvikling av systemet.

7. DISKUSJON

I dette kapittelet følger det en diskusjon av konsekvensene av valgt løsning, hvilke utfordringer vi har møtt og hva vi kunne gjort annerledes.

7.1 Konsekvenser av valgt løsning

De forskjellige verktøyene som vi valgte var relativt nye løsninger for oss, som vi ikke hadde brukt i skolesammenheng. Dermed gikk det mye tid til utforskning og opplæring. I etterkant ser vi at på web-applikasjonen kunne vi gått for Redux for React, som er en global tilstandshåndtering på tilsvarende måte som det vi har gjort i Flutter. Da hadde web-applikasjonen hatt en klarere MVC-arkitektur.

7.2 Gjennomføring

Gjennomføringen av prosjektet har bydd på noen utfordringer som igjen har satt sitt preg på utviklingen av produktet. Vi skal se litt nærmere på utfordringer og løsninger, samt hva vi kunne gjort annerledes for å få til et best mulig sluttprodukt.

7.2.1 utfordringer og løsninger

Allerede i en tidlig fase av prosjektet kom gruppen over en av de større utfordringene gjennom prosjektet - med tanke på tid og arbeid. Den ene gruppen på Østlandet som skulle ta på seg arbeidet med å designe og konstruere databasen med tilhørende funksjoner som oppdragsgiver hadde beskrevet, fullførte ikke dette til angitt tidsfrist. På grunn av det, var vi nødt til å ta over dette arbeidet, slik at utviklingsarbeidet kunne starte opp så fort som mulig. Oppdragsgiver ville på et tidspunkt leie inn eksterne konsulenter for database-arbeidet, men ettersom vi allerede var seint ute med utviklingsarbeidet, ble det besluttet at resultatet ville bli best om vi gjorde dette selv.

Gjennom prosjektet har oppdragsgiver i flere tilfeller “pushet” ekstra på tilleggsfunksjoner som ikke har vært en del av MBPen. I tillegg har gruppen måtte overholde noen stramme tidsfrister til tider, som har resultert i noe mindre fokus på ren optimalisert kode. For at prosjektet skulle resultere med best mulig produkt, har vi hatt samtaler med oppdragsgiver hvor vi har forklart at det er best å fokusere på MBPen først, slik at sluttproduktet blir best. Når det kommer til tidsfristene, har vi gjentatte ganger tatt dette opp - men vi har igjen fått nye tidsfrister etter mindre “stilleperioder”.

En sentral utfordring har vært forståelsen av oppgavebeskrivelse fra oppdragsgiver. Selv med ukentlige møter (to per uke), har det i enkelte tilfeller vært misforståelser som har gjort at gjennomføringen til gruppen ikke alltid har vært det oppdragsgiver så for seg. Mot slutten av prosjektet fikk vi tilsendt den ene gruppen i Oslo sitt UX og grafiske grensesnitt. Dette var noe oppdragsgiver hadde jobbet tett opp mot og har hele tiden trodd at vi har hatt dette i vår besittelse. Vi så derfor fort hvorfor det har oppstått misforståelser i hvordan han så for seg hvordan applikasjonene skulle se ut. Dette har det imidlertid som oftest vært enkle løsninger på, da dette er noe vi enkelt kunne endre på.

7.2.2 Hva kunne vi gjort annerledes?

Dersom UX-gruppene hadde hatt klart grafisk grensesnitt til vår oppstart kunne mange av misforståelsene vært unngått. Det som skjedde var at vi opprettet et grafisk grensesnitt basert på UXet deres, mens de fortsatte å utvikle sitt. Oppdragsgiver arbeidet tett opp mot deres grafiske grensesnitt, noe som han trodde vi hadde fått tilsendt. På bakgrunn av dette oppstod det misforståelser mellom vår løsning og oppdragsgivers påtenkte løsning. For å løse dette burde vi hatt et tettere samarbeid med UX-gruppene i Oslo, slik at vi alltid var oppdatert på deres nyeste løsninger. Det som derimot skjedde, var at vi ikke fikk sett hele deres grafiske grensesnitt før mot slutten av utviklingsperioden.

I web-applikasjonen har vi valgt å bruke React sin egen tilstandshåndtering. Hver komponent har da en egen tilstand som bare kontrolleres i den komponenten. Data vi har trengt å dele med flere komponenter, har blitt utført ved å anvende React Context. Det vi isteden kunne gjort var å implementere Redux. Redux er en forutsigbar tilstandscontainer for JavaScript-applikasjoner. Ved bruk av Redux har man mulighet til å gjøre en tilstand tilgjengelig globalt, slik at man slipper å videresende den mellom komponenter. Ved implementering av Redux ville web-applikasjonen hatt en klarere MVC-arkitektur.

UX-gruppen som skulle lage databasen, hadde grunnleggende kunnskaper innen relasjonsdatabaser. Dersom vi hadde gått for denne løsningen hadde de kanskje hatt klar en database til vår oppstart av prosjektet, men valgte å ikke benytte denne løsningen ettersom det var en stor usikkerhet på om de kunne opprettholde den fastsatte fristen. Hadde det blitt satt opp en relasjonsdatabase måtte vi i tillegg satt opp en egen server og gjennomført de dynamiske oppdateringene på en annen måte. Vi hadde heller ikke hatt databasen og autentiseringen samlet på samme plattform.

8. KONKLUSJON OG VIDERE ARBEID

Målet med dette prosjektet var å utvikle en online opplevelse som stimulerer til kreativ lesing og skriving. Det skulle gjennomføres ved å gi skoler et verktøy der lærere kan planlegge en oppgave og elever fritt kan skrive fra fantasien. Verktøyet skulle være tilgjengelig som en web-applikasjon og mobilapplikasjon både på iOS og Android.

Basert på resultater fra evaluering i punkt 5 kan vi si at vi har oppnådd målet. Lærere kan planlegge en oppgave der elever fritt kan skrive fra fantasien. Verktøyet er tilgjengelig som en web-applikasjon og mobilapplikasjon både på iOS og Android. Hvorvidt systemet stimulerer til kreativ lesing og skriving har vi ikke gjennomført nok observasjoner mot målgruppen til å kunne påstå. Derimot fikk vi positive reaksjoner fra vår hovedfunksjonstest, som lover godt for framtidig utvikling av systemet.

I det nåværende systemet er det kun et punkt fra MBPen som gjenstår og det er at en lærer kan sette hvor mange historier hver elev skal vurdere i siste fase. Dette bør implementeres ved bruk av Cloud Functions, da det er her vi gjennomfører dynamiske endringer i “writsene”. Dette ble ikke gjennomført grunnet liten tid i slutten av utviklingsperioden.

Prosjektet skal videreutvikles fra det som har blitt gjort under bachelorperioden og gruppen har fått tilbud om å bli med i det videre utviklingsarbeidet. For videre utvikling anbefaler vi å gjennomføre en grundigere funksjonstest mot målgruppen, slik at man får reelle tilbakemeldinger. I tillegg bør det implementeres et bedre grafisk grensesnitt som er enklere og mer intuitivt for unge brukere.

9. REFERANSER

Angular University. (2020, 24. April). Angular Single Page Applications (SPA): What are the Benefits? Hentet fra: <https://blog.angular-university.io/why-a-single-page-application-what-are-the-benefits-what-is-a-spa/>

Avdeling for offentlige anskaffelser. (2018, 11. Mai). Test og godkjenning av leveranse. Hentet fra: <https://www.anskaffelser.no/hva-skal-du-kjope/it/systemanskaffelser/folge-opp-leveransene/test-og-godkjenning-av-leveransene>

Bucanek, J. (2009). *Learn Objective-C for Java Developers* (1. utg.). New York: Apress.

Firebase. (u.å.). Firebase helps mobile and web app teams succeed. Hentet fra: https://firebase.google.com/?gclid=EAIaIQobChMIu-KMw8rg6QIVhKwYCh1-IgfuEAAYASAAEgJHvfD_BwE

Firebase. (u.å.). Mix and match Firebase products to solve common app development Challenges. Hentet fra: <https://firebase.google.com/use-cases>

Flutter. (2020, 13. Mars). State Management. Hentet fra: <https://flutter.dev/docs/development/data-and-backend/state-mgmt/intro>

Flutter. (2020, 09. Mai). FAQ. Hentet fra: <https://flutter.dev/docs/resources/faq>

Flutter. (2020, 14. Mai). Provider 4.1.2. Hentet fra: <https://pub.dev/packages/provider>

Fosse-Pedersen, R.J. (2017, 24. August). Fem tips til deg som skal gjennomføre

brukertester. Hentet fra: <https://www.usit.uio.no/om/organisasjon/bnt/web/ux/blogg/2017/brukertest.html>

Google. (2020, 13. April). Cloud Functions for Firebase. Hentet fra:
<https://firebase.google.com/docs/functions>

Google. (2020, 20. april). Cloud Firestore. Hentet fra:
<https://firebase.google.com/docs/firestore>

Guilty.(2019, 04. Februar). Hybrid- vs native-app. Hentet fra:
<https://guilty.no/blogg/hybrid-vs-native-app>

Griffith, C.(u.å). What is Hybrid App Development?. Hentet fra:
<https://ionicframework.com/resources/articles/what-is-hybrid-app-development>

Högstrand, J. (2019, 6. juni). Hva er kanban?
Hentet fra: <https://www.prosjektbloggen.no/hva-er-kanban>

Jadhav, M. A., Sawant, B. R., & Deshmukh, A. (2015). Single page application using angularjs. *International Journal of Computer Science and Information Technologies*, 6(3), 2876-2879.

Lee, C. Cho, O. Sim, H. Lee, W. (2016). Color psychological therapeutic methods in child game graphics. *Advanced Science and Technology Letters*.125, 35-40.
<http://dx.doi.org/10.14257/astl.2016.125.07>

Microsoft (u.å). What is ASP.NET Core?. Hentet fra:

https://dotnet.microsoft.com/learn/aspnet/what-is-aspnet-core?fbclid=IwAR3n04EHpBI5WPuTD9OE_v27u6KfmRTqH2NCcuCKeRhICA3Y59_i4fDiKKg

Mozilla. (u.å.). A re-introduction to JavaScript (JS tutorial). Hentet fra:
https://developer.mozilla.org/en-US/docs/Web/JavaScript/A_re-introduction_to_JavaScript

Nevercode (u.å.). Flutter vs React Native: A developer's perspective. Hentet fra:
<https://nevercode.io/blog/flutter-vs-react-native-a-developers-perspective/>

Næss, A.K. (2020, 12. mars). Slik jobber du iterativt i henhold til agileshift. Hentet fra: https://www.prosjektbloggen.no/slik-jobber-du-iterativt-i-henhold-til-agileshift?__hstc=31225248.c4a577029c49e44b73bd3be6fa38565.1584144000241.1584144000242.1584144000243.1&__hssc=31225248.1.1584144000244&__hsfp=3071927421

React Docs. (u.å.). Thinking in React. Hentet fra: <https://reactjs.org/docs/thinking-in-react.html>

React Docs. (u.å.). Getting Started. Hentet fra:
<https://reactjs.org/docs/getting-started.html>

Redux. (u.å.). Getting Started with Redux. Hentet fra:
<https://redux.js.org/introduction/getting-started>

Skjølsvik, T. & Voldsund, K. H (2017). Forretningsforståelse. Oslo: Cappelen Damm.

The PHP Group (u.å.). What is PHP? Hentet fra:
<https://www.php.net/manual/en/intro-what-is.php>

Torppa, M., Niemi, P., Vasalampi, K., Lerkkanen, M.K., Tolvanen, A. & Poikkeus, A.M. (2019) “Leisure Reading (But Not Any Kind) and Reading Comprehension Support Each Other—A Longitudinal Study Across Grades 1 and 9”, Society for Research in Child Development, utgitt online 30. mars 2019

Twin A (2019, 18. September) Key Performance Indicators(KPIs). Hentet fra: <https://www.investopedia.com/terms/k/kpi.asp>

Universitetet i Oslo. (2019, 30. november) TestDrevet utvikling (TDD), Testdreven utvikling er en måte å utvikle applikasjoner på. Hentet fra: <https://www.uio.no/for-ansatte/enhetssider/los/usit/arrangementer/forum/2010/20101124.html>

Utdanningsforbundet. (2019, 25. juni). Å lese bøker på fritida er viktigere enn mange har trodd. Hentet fra: <https://utdanningsforskning.no/artikler/a lese boker pa fritida er viktigere enn mange har trodd/>

Vårdal, L (2019). Kvalitative og kvantitative metoder. Hentet fra: <https://ndla.no/nb/subjects/subject:43/topic:1:190302/topic:1:197973/resource:1:190746>

10. APPENDIX

10.1 GANTT diagram

