Høgskulen på Vestlandet

BACHELOROPPGAVE

Kronbarappen - En mobilapplikasjon for besøkende ved Kronbar

The Kronbar App - A mobile application for customers at the student bar Kronbar

Arne Kvaleberg

Sivert Lunde

Jokubas Morsund

Informasjonsteknologi og dataingeniør

Fakultet for ingeniør- og naturvitskap

01.06.2020

Høgskulen på Vestlandet

Fakultet for ingeniør- og naturvitskap

Institutt for datateknologi, elektroteknologi og realfag

TITTELSIDE FOR HOVEDPROSJEKT

Rapportens tittel: **KronbarAppen - en mobilapplikasjon for besøkende ved Kronbar**	*Dato:* 02.06.2020
Forfatter(e): **Sivert Lunde, Arne Kvaleberg, Jokub Morsund**	*Antall sider u/vedlegg:* **36**
Studieretning: **Informasjonsteknologi, Dataingeniør**	*Antall sider vedlegg:* **5**
Kontaktperson ved studieretning: **Volker Stolz**	*Gradering:* **Ingen**
Merknader:	

Oppdragsgiver: **Studentsamfunnet Kronbar**	*Oppdragsgivers referanse:* **Magnus Marthinsen**
Oppdragsgiver kontaktperson: **Magnus Marthinsen**	*Kontakt:* Kronbar.bergen@hvl.no

Sammendrag:
Denne rapporten beskriver prosjektet vårt - KronbarAppen for studentbaren Kronbar. Prosjektet baserer seg på å utvikle en applikasjon for besøkende ved baren hvor de kan se kommende arrangementer, samt registrere bruker for et konsept Kronbar kaller 'Komfortbillett' som reduserer priser på utvalgte varer. Applikasjonen tar i bruk API-er fra flere tjenester for å sette sammen informasjon på en plass, og bruker tar i bruk skylagring for brukerinformasjon. Løsningen vi har utviklet og som i presenterer i denne oppgaven vil være en plattform Kronbar bygger videre på mot en eventuell lansering.

Stikkord:

React Native	Mobilapplikasjon	Kryssplattform

Høgskulen 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

"What you can achieve depends on what you can imagine."

- Taumu, npmjs.com

Disse vise ordene kom vi over når vi leste om *react-scroll-paged-view* på npmjs.com. Vi tenker dette er et fint sitat å ta med oss videre i prosjektet - for selv om vi ikke kan mye om rammeverkene vi nå skal ta i bruk, er det faktisk kun fantasien som setter grenser på hva vi til slutt kan få til med det.

I denne rapporten vil vi gå igjennom prosjektet vårt - Kronbarappen, for Kronbar. Prosjektet har basert seg på å utvikle en mobilapplikasjon som kan brukes av besøkende ved Kronbar.

Vi ønsker å takke Kronbar for å ha gitt oss muligheten til å jobbe med et slikt spennende prosjekt, spesielt Magnus Marthinsen som har bidratt med god hjelp underveis.

Vi ønsker også å takke vår interne veileder på HVL, Volker Stolz som har kommet med hjelp, veiledning og verdifulle tilbakemeldinger under prosjektperioden.

Vi vil også takke Sven-Olai for morsomme timer i DAT102 med sine gøyale kortstokker.

En takk gis også til Lars-Petter Helland og hans fantastiske smil som har fått oss gjennom mange tøffe morgensforelesninger.

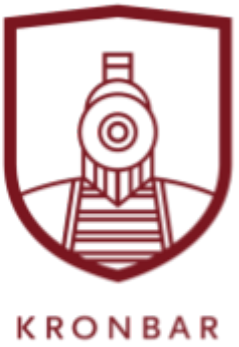
INNHOLDSFORTEGNELSE

1 INNLEDNING

I dagens samfunn er mobilen et av de vanligste verktøyene vi bruker. Den følger med oss overalt, hele dagen. Kan alle bedrifter ha nytte av å utvikle en applikasjon for sine kunder? Denne bachelorrapporten vil ta for seg utviklingsløpet av en applikasjon for studentbaren Kronbar.

1.1 Motivasjon og mål

Kronbar er en bar og konsertscene i Bergen, drevet utelukkende av frivillige studenter fra Høgskulen på Vestlandet. Lokalet er et flott industribygg som tidligere har vært et maskineri for damplokomotiv, et fredet kulturminne fra NSBs eldre dager. De driver et bredt utvalg av arrangementer for studenter, som quiz, jam, spillkveld og mer. I tillegg tilbyr de utleie av lokalet, slik at studenter har et rimelig sted å arrangere sine egne arrangementer.



Figur 1: Logo til Kronbar

1.1.1 Problemstilling

Kronbar har i lengre tid hatt Facebook som primær markedsføringskanal. I en tid hvor dette mediet har fallende popularitet blant nye studenter har de behov for en ny kanal, gjerne i form av en egen app. Kronbar har også en konsept kalt 'Komfortbillett' som de ønsker å få digitalt. Dette er i dag et oblat som festes på studentkortet og gir billigere varer, som for eksempel kaffe, i baren. De ønsker gjerne også å kunne vise frem arrangementer de har enkelt.

Problemstillingen vår er å kunne integrere komfortbillett og markedsføringsmuligheter i en mobilapplikasjon rettet mot de besøkende ved Kronbar.

1.1.2 Mål

Vårt mål er å bygge en applikasjon som etter hvert skal kunne tas i bruk av besøkende. Applikasjon skal være så selvstendig som mulig, og benytter seg av Google Firebase for innlogging og skylagring. Mindre delmål inkluderer mindre viktige funksjoner som å kunne sende egne notifikasjoner til brukere samt oppkobling mot mobilapplikasjonen for interne ved Kronbar.

1.2 Kontekst

1.2.1 Prosjekt

Kronbar har kommet frem til at komfortbilletten gjerne kan være litt for tungvint slik det er i dag. Ikke bare krever det innkjøp av oblater og koordinering fra Kronbar sin side, men det krever også at studentene er nødt til å ha med seg studentkortet hver gang de besøker Kronbar. Hvis studentkort mistes må billetten erstattes, eller bare generell slitasje ved å dra kortet inn og ut av lommebok. De ønsker derfor å se på muligheten til å digitalisere denne.

Videre bruker de veldig mye tid på å markedsføre arrangement gjennom stands på HVL. Kronbar ønsker derfor også at en kalender skal implementeres i appen, slik at studenter enkelt kan se kommende arrangement.

1.2.2 Gruppen som del av bedriften

Kronbar er ikke en IT-bedrift, og drives utelukkende av studenter. Vi er derfor et selvstendig team som arbeider med applikasjonen, og de er ikke avhengig av vårt produkt for deres drift.

1.3 Avgrensninger

1.3.1 Kunnskap om teknologi

Prosjektets suksess er avhengig av at gruppen klarer å tilegne seg kunnskap om de ulike teknologien vi skal bruke. Vi tar i bruk flere rammeverk vi ikke har vært innom i studieløpet, og vil derfor være veldig avhengig av dokumentasjonen tilknyttet til disse.

1.3.2 Produkttesting

Siden applikasjonen vår er rettet mot besøkende, er det viktig at vi får testet at applikasjonen vår fungerer ordentlig, og at brukergrensesnittet kan forstås av alle.

1.4 Ressurser

Oppgaven baserer seg på et Javascript rammeverk opprettet av Facebook, React Native. Dette er et rammeverk som kan brukes for å enkelt utvikle en kryssplattformapplikasjon, altså som kan brukes på tvers av plattformer. Fordelen med dette er at vi kan bruke samme kodebase for både Android og iOS. Ulempen er at det krever litt mer arbeid å få til operativsystemspesifikke funksjoner, som for eksempel kamera, men fordelene veier opp for det.

Vi tar i bruk Google sin tjeneste Firebase for skylagring og autentisering av brukere. Firebase er et produkt som er markedsført som en 'alt-i-ett'-tjeneste, og som gjør at vi ikke har behov for å utvikle en tjenerdel for applikasjonen vår.

Videre tar vi også i bruk Studentbergen sitt API for kalenderen vår. Kronbar har i lengre tid postet arrangementene sine her og brukt API-et på hjemmesiden for å vise kommende arrangement.

Når det kommer til utviklingsmiljø bruker vi Visual Studio Code for React Native-utviklingen. Vi bruker også Xcode og Android studio for simulering av iPhone og Android respektivt, samt til endringer som må gjøres individuelt for hver telefon. Dette inkluderer ting som appikon, innlastingsskjerm og mer. For versjonskontroll bruker vi git med gitlab som tjenerside.

1.5 Oppbygging av rapporten

Rapporten deles inn i 7 kapitler som forklarer gjennomgangen av prosjektet.

Kapittel 1 - Definerer mål, motivasjon og avgrensninger ved prosjektet.

Kapittel 2 - Beskriver bakgrunnen til prosjektet og innledende modellering og design.

Kapittel 3 - Går gjennom alternative løsninger, valg av verktøy og valg av utviklingsmetodikk.

Kapittel 4 - Utdyper hvordan applikasjonen ble utviklet, og forklarer hvordan vi har brukt tredjepartsverktøy.

Kappitel 5 - Presenterer evalueringen av prosjektet og sluttmøter med oppdragsgiver.

Kapittel 6 - Diskuterer resultatet av prosjektet, hva som kunne bli gjort annerledes og hvorfor vi har valgt det vi har valgt.

Kapittel 7 - Beskriver fremtidig arbeid vi gjerne skulle gjort hvis vi hadde tid, og hva som må gjøres før applikasjonen kan tas i bruk.

2 PROSJEKTBESKRIVELSE

Dette kapittelet vil ta for seg bakgrunnen og beskrivelse av prosjektet, blant annet kravspesifikasjonen og innledende idè til løsning.

2.1 Praktisk bakgrunn

2.1.1 Prosjekteier

Kronbar er en bar og konsertscene i Bergen, drevet utelukkende av frivillige studenter fra Høgskolen på Vestlandet. Lokalet er et flott industribygg som tidligere har vært et maskineri for damplokomotiv, et fredet kulturminne fra NSBs eldre dager. De driver et bredt utvalg av arrangementer for studenter, som quiz, jam, spillkveld og mer. I tillegg tilbyr de utleie av lokalet, slik at studenter har et rimelig sted å arrangere sine egne arrangementer.

Hovedårsaken til at Kronbar ønsker å utvikle en applikasjon er å nå ut til flere studenter med arrangementer, samt gi studentene en bedre oversikt over hva som skjer når. I tillegg opererer Kronbar med et konsept som heter 'Komfortbillett'. Det er et oblat som festes på studentkortet, og som gir innehaveren billigere priser på diverse kjøp, både i bar og billettpriser til arrangement. Oblatet skal også bli tatt med i applikasjonen og digitaliseres.

2.1.2 Tidligere arbeid

Kronbar har aldri prøvd å digitalisere eller legge til et verktøy slik som dette. Det nærmeste en kommer når det gjelder informasjon anngående forskjellige arrangement er hjemmenettsiden "Kronbar.no" og den føler ikke Kronbar selv er utfyllende nok. I tillegg så er en mobilapplikasjon enklere å sette seg inn i for nye kunder og avlaster frivillige som jobber på Kronbar og lar de dedikere mer tid til viktigere arbeid slik som å være bak disken og gjøre klar for forskjellige arrangement i stedet for å sitte med inngangen.

2.1.3 Innledende kravspesifikasjon

Kronbar presenterte en oppgave med en del frihet med tanke på hvilken måte applikasjonen kunne bli utviklet. Oppgaven sa at det foretrukne språket var React Native men om det vart ønske for det at hvilken som helst språk som gjorde jobben hadde blitt akseptert. Det fulgte også med en liste av krav som gjorde at fokus ikke vært gitt til unødvendig funksjonar og det kunne alltid bli slått opp hva var det neste som trengte fokus, etter en gitt funksjonalitet ble ferdig implementert. Kvar av punkten på listen fulgte med en beskrivelse og mål med kravet, samtidig som prioritet og hvor viktig det va å fokusere på hver og en gitt krav. Vi hadde også et større mål som vi hadde som

hovedfokus som var; applikasjonen som vert utvikla skulle brukes til å gi mest mulig informasjon på en lettest mulig måte.' Helt siden start av utviklingen har denne ideen vert prioritert fremst.

Design og tekniske spesifikasjoner

Oppgaven som ble presentert hadde fire hovedpunkter som vi måtte ta hensyn til:

1. Applikasjonen må fungere på både Android og iOS.
2. Kronbar ønsker en kalender som viser kommende arrangement.
3. Annonsering. Kronbar ønsker å kunne promotere ting til brukerne. Dette kan være noe som ukens kaffe, eller for eksempel et arrangement som skal fremheves.
4. Man skal kunne kjøpe og vise komforbiletten sin via applikasjonen. Det skal også være mulig å kjøpe komfortbillett gjennom baren og deretter få denne overført til mobil. Billetten skal samtidig kunne aktiveres og deaktiveres av et administrasjonsverktøy.

Dette var de viktigste ønskene til Kronbar med alt annet prioritert under disse fire punktene. Før disse spesifikasjonene kunne bli oppfylt måtte det først og fremst bli lagt til en del grunnarbeid. Et stor ønske var at vi fulgte god programdesign så godt vi kunne, noe vi har fått innføring i gjennom flere fag i studieløpet.

I tillegg til dette hadde Kronbar følgende, mindre prioriterte ønsker:

5. Brukerne skulle kunne invitere hverandre på forskjellige arrangement og dele tilbud de finner i applikasjonen
6. Applikasjonen skal presentere åpningstidene. Disse må kunne ta høyde for ferier og helligdager.
7. Lagring av brukerstatistikk om hvordan studenter håndterer og bruker applikasjonen. Er det mange som ser på hvert arrangement? Får alle med seg alle arrangementene? Kronbar vil at markedsstatistikk skal kunne hentes ut for å forbedre tilbudet som blir gitt til studentane.
8. Integrasjon med intern-applikasjonen (se annet prosjekt gitt av Kronbar).

Disse punktene tar for seg mindre viktige funksjoner med applikasjonen som kunne legges til dersom det var ekstra tid til overs.

Utviklingsmetodikk

Kronbar ga en del frihet med tanke på hvilken utviklingsmetode som skulle bli tatt i bruk. I motsetning til Scrum der alt skal gjøres etter etablert prosedyre, begynner Kanban der det ble avsluttet sist og utviklingen fortsetter litt om gangen derfra. Valget å bruke Kanban utviklingsmetodikk kom fra flere ideer som gruppa hadde til felles. Vi ville ha en fast flyt, samt fortsette sammen der vi sist avsluttet. Det var brukt forskjellige verktøy som gjorde at arbeidet gjekk raskere og at minst mulig tid gikk før vi klarte å komme i gang med arbeidet. Kommentarer

på koden som ble skrevet vert også brukt slik at vi lett kunne finne fram til det som ga eventuelle problem eller som måtte bli endra på i ettertid.

2.1.4 Innledende løsnings-idé

Idéen til en potensiell løsning på problemstillingen ble gitt til oss av Kronbar. Den bygger på at Kronbar ønsker å lette arbeidsmengde fra frivillige som jobber der, samtidig som at de har løst å digitalisere og modernisere noen aspekt av Kronbar. En av disse er komfortbillett som nevnt tidligere er en fysisk oblat som kunder fester til studentkortet. Å feste oblatet til studentkortet er en problematisk løsning, for eksempel med tanke på at etter seks semester og seks forskjellige oblat; visst du ønsker å samle, blir studentkortet rotete og du ser så vidt hvem som eier kortet.

Funksjonalitet

Som nevnt i 2.1.1, er applikasjonen ute etter å gjenskape tjenester som Kronbar allerede tilbyr, bare lettere tilgjengelig for studenter via bruk av en mobiltelefon. Disse tjenestene som er nevnt i 2.1.3.1 er litt forskjellige men til slutt ender med å avlaste arbeidsmengden til frivillige på Kronbar. Framover skal vi gå gjennom hvordan applikasjonen ender opp med å gjøre nøyaktig det, samtidig som hva den tilbyr og hvorfor applikasjonen fungerer og eventuelt hvordan endte det opp slik visst det fantest en mer optimalisert eller moderne løsning.

Løsningen vår vil optimalisere et par enkelte prosesser som fleste kunder ved Kronbar er uviten og ikke tar nytte av, samtidig som at den gjør komfortbilletten digital. Dette er de forskjellige sidene som brukerne har tilgang til når de først åpner applikasjonen.

Forside

Forsiden gir brukeren en rask innføring til hvilken arrangement er den nærmeste, hvilken kaffe fra menyen er ukens utvalgte, samt prisen på den. Videre kan du se en beskrivelse om Kronbar og en liten beskrivelse om hver av de forskjellige arrangementene som Kronbar arrangerer.

Kalender

Viser brukeren hvilken arrangement skjer denne, neste og de framtidige ukene. Du kan samtidig klikke deg inn på de forskjellige datoene for å se mer informasjon om nøyaktig det som skjer den dagen, samtidig som at det står en start/slutt tidspunkt og en Facebook-link som fører til Facebook-siden for arrangementet.

Komfortbillett/Innlogging

Gjennom bruk av Firebase kan brukeren logge seg inn gjennom en av tre metoder. Det går ann å bruke Google eller Facebook for å logge inn, samtidig som at du kan lage en egen bruker for

Kronbar-applikasjonen. Når en ny bruker blir opprettet blir den automatisk henta inn av administrator-verktøyet. Når personen har betalt for komfortbillett er det så lett som å søke de opp og sette ein boolean lik true. Det vises klart om personen har komfortbillett eller ikke på komfortbillett-skjermen, slik at det er lett å se for alle under kontroll. På denne skjermen har du også et bilde av studenten som studenten selv kan velge, noe som er nødvendig om registreringsprosessen skal kunne bli fullført. Bildet er en stor del av identifisering når en person skal vise at han eller ho har aktiv komfortbillett og er derfor nødvendig for at komfortbilletten skal kunne bli aktivert i det heile. Uten at personen har lagt til et bilde vil ikke komfortbilletten aktivere seg.

Meny

Viser menyen til Kronbar som blir hentet fra Untappd. Dette er hovedsakelig på applikasjonen slik at det ikke sløses tid i køen å se på skjermen over baren for å bestemme seg hva noen skal ha, i stedet for kan man velge ut når man sitter at med bordet og være klar med bestilling til man er med baren. Samtidig er det mulig at dette utfordrer folk til å velge ut noe mer interessant å bestille da de kan se gjennom menyen og muligens finne noe annerledes som de liker. Vi går mer inn på temaet angående menyen i 4.2.8 og 4.4.6.

Det hjelper også bord å ta større bestillinger samtidig, da personene ved bordet enkelt kan se hvor mye hver enkel vare kostet, og dermed enkelt finne riktig sum å 'vippse'.

2.2 Litteratur om problemstillingen

Gruppen har ved tidligere anledninger drevet med utvikling av mobilapplikasjoner. To av medlemmene hadde DAT153, Mobile og distribuerte webapplikasjoner, som blant annet tar for seg utvikling av applikasjoner for Android. Det siste gruppemedlemmet var innom apputvikling i DAT156, Praksis i arbeidslivet, da sammen med utviklere i Fieldcom.

Gjennom disse fagene, spesielt DAT153, har vi hatt tilgang til diverse litteratur angående mobilapplikasjonsutvikling. Dette gjelder hovedsakelig forelesninger i faget og læreboken i dette faget. Selv om vi ikke bruker denne litteraturen som direkte kilder i rapporten, har det fortsatt gitt oss en god grunnforståelse av utviklingen av mobilapplikasjoner.

Videre har dokumentasjonene om de respektive verktøyene vært til stor hjelp for forståelse om utviklingsløpet.

3 DESIGN AV PROSJEKTET

Dette kapittelet vil ta for seg de forskjellige løsningene som ble lagt frem for prosjektet, og hvorfor vi valgte den vi gjorde. Vi vil også diskutere valg av verktøy og utviklingsmetodikk.

3.1 Mulige løsninger

3.1.1 Alternativ løsning 1 - Utvikle to applikasjoner i Swift og Java

Dette innebærer at vi ser bort i fra et kryssplattform rammeverk og utvikler to separate applikasjoner. Vi kjenner til Android-utvikling fra studiet, men har lite til ingen kunnskap til Swift. Hovedproblemet med denne løsningen er at arbeidsmengden blir doblet. Det ville krevd to forskjellige kodebaser, og det ville vært forskjeller på hva som ville vært støttet av de forskjellige operativsystemene. Det positive med denne løsningen er at ytelsesevnen nok ville vært sterkere, men vi gjør ingen særlig krevende operasjoner.

3.1.2 Alternativ løsning 2 - React Native med egen tjenerside

Denne løsningen var den vi startet prosjektet med. Men, da vi skulle sette opp tjenersiden kom vi frem til at vi kunne spare mye tid på å benytte oss av Firestore, som er Firebase sin lagringstjeneste. Siden vi allerede hadde etablert Firebase autentisering i appen var det lite arbeid å implementere Firestore.

En egen tjenerside ville bestått av en Java-applikasjon med rammeverkene Spring og Swagger for en enkel REST-API som appen kunne kommunisere med. Her ville API-et videre kommunisert med en database som lagret informasjon om brukeren. Fordelen her er at vi har kontroll på akkurat hvilke metoder vi vil ha på API-et. En negativ side er at vi selv må sørge for sikkerhet, og tidsbruken på å sette opp applikasjonen ville vært ganske høy.

3.1.3 Alternativ løsning 3 - Webapplikasjon med egen tjenerdel

Dette er alternativ som ble foreslått av vår veileder. Kronbar har allerede en hjemmeside, og vi utforsket muligheten for å kunne videreutvikle denne til å inneholde informasjonen vi trenger. Fordelen med dette er at vi ikke trenger noen lisenser for å publisere en app, og vi er godt kjent med utvikling av webapplikasjoner gjennom studiene. Ulempene vi fant er mangelen på mulighet for sending av notifikasjoner, samt mangel på vern mot klientsideendringer. Gjennom 'Inspiser Element' kan man endre på utseende av hvordan nettsteder ser ut, og vi ønsker ikke at brukere enkelt skal kunne endre komfortbillettsiden til å se ut som de har en billett når de ikke har. Vi ønsker også å bufre litt informasjon på telefonen, og brukere kan ha blokkeringer mot dette i

nettleseren. Sistnevnte er dog ikke noe stort hinder. Denne løsningen ville også benyttet seg av samme tjenerdel som nevnt i 3.1.2.

3.1.4 Diskusjon av alternativene

Dersom det skulle vise seg å være for tidkrevende å lære seg React Native ville vi nok realistisk sett gått for alternativ 1, da dette kan best følge kravspesifikasjonene. Dette ville ført til applikasjoner som har veldig dårlig brukergrensesnitt, da tiden ville gått til å få funksjonene på plass på hver plattform.

Hvis det skulle oppstått problemer med Firebase ville vi gått for løsning 2. Vi har et tjenerskall som kan kommunisere med en lokal database. Dette ville tatt tid og ville gått på bekostning av både antall funksjoner i applikasjonen og brukergrensesnittet.

Dersom det skulle vise seg å være store utfordringer med mobilapplikasjonsutvikling i sin helhet ville vi gått for alternativ 3. Dette ville gått på tross av flere av kravspesifikasjonene, men kunne delvis oppfylt noen.

3.2 Valgt løsning

Vi har valgt å benytte oss av React Native med Google Firebase som tjenerside. Dette er for å få en lik kryssplattformapplikasjon mellom iOS og Android. Firebase gjør at vi kan spare masse tid på å kunne la være å sette opp egen tjenerside. Dette gir oss god tid til å implementere de funksjonene vi ønsker, samt skape et fint brukergrensesnitt.

3.3 Valg av verktøy

Visual Studio Code

Hovedutviklingsmiljøet vårt er Visual Studio Code fra Microsoft. Den har god støtte for de fleste filtyper og god integrering av git og fungerer på både Windows og MacOS.

Figur 2: Logo til Visual Studio Code

Figur 3: Logo til Google Firebase

Google Firebase

Firebase er en tjenersideapplikasjon som kan brukes for autentisering og skylagring. Fordelen med dette er at all autentisering skjer på Google sine servere og krever lite behandling fra vår del. Firebase har innebygd støtte for innlogging via flere tjenster, blant annet Facebook, Google, Github, E-mail/Passord, Apple med mer.

Firebase har også *Firestore* og *Firebase Storage* som er to produkter vi også benytter oss av. Dette

lar oss lagre brukerobjekter i Firebase, samt lagre profilbilder i en database slik at brukere kan logge inn på flere telefoner og samtidig ha samme bilde.

React Native

Figur 4: Logo til React Native

React Native er et Javascript-rammeverk opprettet av Facebook, og er i dag drevet av både Facebook og et aktivt brukersamfunn. Native innebærer at det kan kjøres 'natively' på iOS og Android, altså at det ikke krever en separat kodebase. React Native kjører på Node, og fordelen er at det er enkelt å installere moduler. Moduler er skrevet av brukersamfunnet og ligger åpent tilgjengelig for alle å bruke på nett.

Git

Git er et verktøy for versjonskontroll av kode, og brukes for å enkelt dele kode mellom utviklere. Det skaper en mulighet for å jobbe opp mot samme kodebase uten problemer. Du jobber uavhengig fra andre prosjektarbeidere og kan dele og motta kode ved bruk av git.

Figur 5: Logo til git

Figur 6: Logo til Xcode

Xcode

Xcode er utviklingsmiljøet fra Apple, og brukes for applikasjoner som skal brukes på Apple sine produkter, som iPhone, iPad og Apple Watch. Xcode inkluderer også en iOS-simulator som kan brukes for å teste applikasjonen på maskinen uten å måtte koble til en fysisk telefon.

Android Studio

Android Studio er Android sitt svar på Xcode - det brukes for utvikling av Androidapplikasjoner. Også her blir det inkludert en simulator for Android til å teste applikasjoner.

Figur 7: Logo til Android Studio

Figur 8: Logo til Node.js

Node.js

Node.js er et Javascriptmiljø som lar datamaskinen kjøre Javascriptkode uten en nettleser.

3.4 Prosjektmetodikk

3.4.1 Utviklingsmetodikk

Utviklingsmetoden Kanban som gruppen bruker har mange forskjellige formål. Gruppen ble kjent med denne metoden i DAT109-kurset på Høgskulen på Vestlandet og har erfart dette som en effektiv form for utvikling i tidligere prosjekter. Den har først og fremst som mål å legge til rette for kontinuerlige forbedringer gjennom etablerte prinsipper. Dette vil si at prinsippene som er på plass fra før kan alltid bli utarbeidet og forbedret. Det er stort fokus på en kontinuerende arbeidsflyt og nøyaktig på grunn av dette følte for at Kanban var riktig for dette spesifikke prosjektet.

3.4.2 Prosjektplan

Vi har et Gantt-diagram for prosjektløpet (Se vedlegg 9.1).

Gantt-skjemaet er særlig viktig i et prosjekt som dette, hvor det er flere tidsfristen å opprettholde. Med innlevering rundt annenhver uke er det derfor nødvendig å kunne følge et satt tidsskjema, slik at vi vet når vi må begynne på diverse innleveringer. Vi har forholdt oss på skjema for det meste, dog innledende oppsett tok lenger tid enn forventet. Dette ble veid opp av at noen skjermer i applikasjonen ble fortere ferdigstilt enn forventet.

3.4.3 Risikovurdering

Vi har laget en risikoanalyse for prosjektet (Se vedlegg 9.2) for å få et overblikk over hvilke risikoer vi kan møte på under utviklingsløpet, og hvordan vi eventuelt skal håndtere det.

3.5 Evalueringsplan

Vi har prøvd å gjøre kontinuerlige evalueringer av arbeidet underveis. Vi har hatt regelmessige møter med oppdragsgiver for å få tilbakemelding på produktet vårt, samt forklare hva vi planlegger å gjøre videre.

3.5.1 Evaluering fra Kanban

Vi benytter oss av Kanban som utviklingsmetode. Dette lar oss kunne raskt ta tak i tilbakemeldinger vi får underveis, og at vi snakker med oppdragsgiver før vi gjør drastiske endringer i enten funksjonalitet eller brukergrensesnitt. Vi kan bruke tilbakemeldingene vi får til å evaluere arbeidet vårt.

3.5.2 Kommunikasjon fra veileder og oppdragsgiver

Gjennom utviklingsløpet ønsker vi å holde god kontakt med både oppdragsgiver og veileder, slik at vi kan få kontinuerlige tilbakemeldinger på produktet vårt. På den måten kan vi justere løpet vårt ut i fra tilbakemeldingene.

3.5.3 Brukertesting

Vi ønsker underveis å la frivillige på Kronbar teste applikasjonen under utvikling, slik at vi kan få en større respons på hvordan den er å bruke.

3.5.4 Sluttmøte med oppdragsgiver

Når prosjektet nærmer seg slutten vil vi holde et møte med oppdragsgiver hvor vi ønsker å få en konkret tilbakemelding på hvilken grad vi løste problemstillingen vi ble gitt.

4 DETALJERT DESIGN

I dette kapittelet vil vi gå gjennom i dypere detaljer hvordan vi har oppnådd resultatet vårt. Vi vil se spesifikt på komponentene i de forskjellige skjermene, samt gå nærmere inn på hvilke eksterne tjenester og bibliotek vi har tatt i bruk. Ved å lese dette kapittelet skal du få en bedre forståelse for selve implementasjonen av applikasjonen og hvordan den er sydd sammen.

4.1 Prosjektskall

Prosjekskallet beskriver grunnleggende hva applikasjonen er bygd på og hva som er felles for alle skjermene.

4.1.1 React native

Fordelen med React Native, som diskutert i kapittel 3, er at man får en delt kodebase for plattformene man utvikler på. Det betyr at man bare trenger å kode ting èn gang, så vil rammeverket ta seg av resten når det kommer til bygging til maskinvarekode som skal kjøre på enhetene.

I tillegg bygger applikasjonen på Node.js som er en kjøreomgivelse for Javascript uten behov for nettleser. Det betyr at vi enkelt kan teste koden vår uten en nettleser som skal kjøre koden, men at det skjer lokalt på enheten.

React Native har et samfunn som bygger bibliotek man kan laste ned og bruke i utviklingen sin. Dette kan være alt fra bildekaruseller til måling av hvor sterkt et passord er under registrering. Installasjonen av disse er svært enkelt, og gjør at en sparer mye tid.

Props

En React Native-komponent kan ses på som en funksjon som skal generere et element for visning på skjermen. Disse funksjonene kan ta parametere, som av standard heter *props*.

```
<NextEvent event={this.state.response} navigation={this.props.navigation} />
```

Figur 9: Utdrag av kode for NextEvent

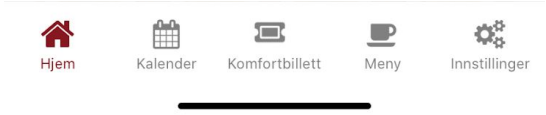
I figuren ovenfor ser man inkluderingen av *NextEvent* i hjemskjermen. Her sender man med to props; et *event* (arrangement) og *navigation* (navigatøren). Man kan da bruke disse to objektene i NextEvent når dette elementet skal genereres, som vist i figuren under hvor vi henter ut tittelen på arrangementet.

```
<Text style={(props.event.name).length > 8 ? styles.titleBig : styles.titleSmall}>{props.event.name}</Text>
```

Figur 10: Utdrag av kode for props i NextEvent

4.1.2 React navigation

En viktig del av brukeropplevelsen er god navigasjonsflyt i applikasjonen. Her benytter vi oss av react-navigation [18]. Dette er et bibliotek som tilbyr mange forskjellige navigasjonsløsninger avhengig av design. Vi valgte en løsning som er kjent fra populære plattformer som blant annet Facebook, Instagram og YouTube; bottomTabNavigator. Dette designet gir en navigatør som alltid er synlig i bunnen av skjermen.



Figur 11: Skjermdump av Navigasjonsmenyen

BottomTabNavigator	StackNavigator	Skjermer
Hjem	HomeStackNavigator	Hjem - Arrangement
Kalender	CalendarStackNavigator	Kalender - Arrangement
Komfortbillett	ComfortStackNavigator	Komfortbillett - Innlogging - Registrering
Meny	BarMenuStackNavigator	Meny
Innstillinger	SettingsStackNavigator	Innstillinger

Figur 12: Innledende modellering av nødvendige skjermer

Figuren ovenfor viser de forskjellige skjermene som kan navigeres til med navigatøren. *StackNavigator* brukes for å kunne gå tilbake mellom skjermer; eksempelvis, når man trykker på et arrangement i kalender forventer man at tilbakeknappen tar deg tilbake dit du nettopp var. *StackNavigator* legger til navigeringen i en stabel, og kan dermed enkelt hoppe tilbake til forrige side. Den fungerer helt likt som både iOS og Android gjør, så brukere vil kjenne igjen navigeringen fra operativsystemet. iOS støtter også gester (Engelsk: *gestures, eller swiping)* for å gå tilbake.

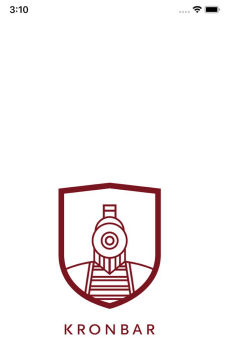
4.2 Skjermer

4.2.1 Innlastningsskjerm

Den første skjermen en bruker møter på ved oppstart av applikasjonen er innlastningsskjermen. Dette er en skjerm som vises i løpet av de første sekundene av oppstartsprosessen slik at brukere blir møtt med en ferdig lastet side heller enn en side som går gjennom en mengde endringer.

Innlastningsskjermen er implementert ved bruk av react-native-splash-screen (heretter omtalt som "RNSS").

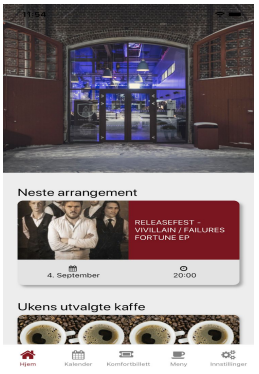
Et react native prosjekt har fra starten av en standard innlastningsskjerm. Denne skjermen inneholder React Native sin logo. Dette må selvsagt endres da det vil forvirre brukeren. Der React Native ellers bruker samme kode for iOS og Android, krever RNSS mye egen kode for hver av platformene. Dette ble løst ved å følge en guide på nett (Appstud, 2019).



Figur 13: Innlasningsskjerm

4.2.2 Hjem

Straks applikasjonen er lastet inn, vil brukere bli tatt til hjemskjermen. Dette er en side som fungerer som en overordnet informasjonskilde. Her skal brukere få relevant informasjon om arrangementer, tilbud, åpningstider og generelle beskrivelser. Hjemsiden er satt sammen av mange forskjellige komponenter som vi går dypere inn på i delkapitlene under.



Figur 14: Hjemskjerm

Topptekst

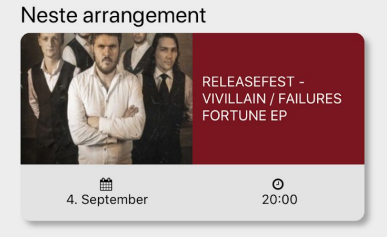
Øverst på forsiden til applikasjonen har vi lagt til en bildetopptekst. Dette er et bilde fra inngangen til kronbar og er ment å gi applikasjonen et moderne preg samt å gi brukeren av applikasjonen en familiær følelse.

Toppteksten er implementert ved bruk av react-native-animated-header (heretter omtalt som "RNAH") (Aaron_ta, 2019). RNAH tar et bilde som prop, og gir en topptekst som slår seg sammen når brukeren blar nedover på siden. Dette lar oss bruke en stor del av forsiden til å vise et bilde uten at det begrenser bruksområde vi har tilgjengelig på skjermen.

Figur 15: Bildetopptekst

Neste arrangement

Den første dynamiske komponenten på forsiden er neste-arrangement-komponenten. Denne tar i mot arrangement-data for det førstkommende arrangementet på kronbar, hentet fra studentbergen.no (les mer under kapittel 4.4.5). Med denne dataen generer den en visning som inneholder bilde og tittel på arrangementet, samt dato og klokkeslett for startpunkt. Sted er ikke relevant å inkludere da alle arrangementer finner sted på kronbar.



Figur 16: Visning av neste arrangement

Neste arrangement er manuelt implementert hvor eneste bruk av eksternt bibliotek er ikoner hentet fra fontawesome (FontAwesome, u.å.). Når en bruker klikker på komponenten vil applikasjonen navigere videre til en mer detaljert visning av arrangementet (se kapittel 4.2.3).

Ukens kaffe

Ukens kaffe er en promotering av Kronbar hvor en spesifikk kaffe er satt ned i pris. Dette endrer seg fra uke til uke og komponenten må derfor være dynamisk. For å gjøre dette enklest mulig for de som administrerer dette for Kronbar, henter den navn og pris på kaffe fra Firestore (se kapittel 4.4.2). Dette kan igjen oppdateres fra et mer brukervennlig administratorverktøy, som vi har laget et gjennomførbarhetsbevis for (se kapittel 4.5).

Figur 17: Promotering av ukens kaffe

Komponenten består av enkel styling og visning av innhentet informasjon og benytter ingen eksterne bibliotek.

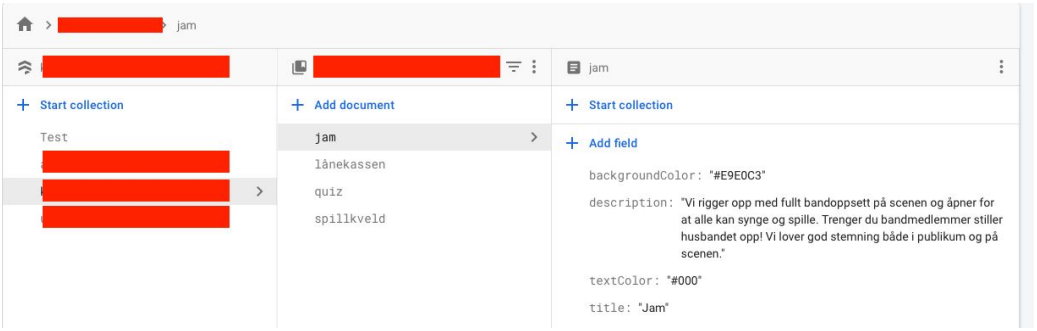
Arrangementinformasjonkarusell

Denne komponenten skal gi brukeren generell informasjon om arrangementene til Kronbar.

Informasjonen hentes fra en egen samling i Firestore, og hver rute genereres dynamisk ut i fra elementene applikasjonen henter. Det betyr at man enkelt kan endre både informasjon, titler og antall arrangementer uten å gjøre noe på klientsiden.

Vi har tatt i bruk React-Native-Snap-Carousel (heretter omtalt som "RNSC") for å få en horisontal karusell som også viser litt av de to nærliggende rutene.

Figur 18: Beskrivelser av arrangementer på Kronbar



Figur 19: Firestore-strukturen som er brukt for karusellen

Sosiale medier

Helt nederst på hjemskjermen finner brukeren lenker til Kronbar sine sosiale medier; Facebook og Instagram. Ikonene peker til sine respektive nettsteder, og vil igjen åpnes i sine respektive applikasjoner hvis brukeren har installert disse på telefonen - hvis ikke åpnes det i nettleser.

4.2.3 Arrangement

Arrangement er en side i applikasjonen som viser detaljert informasjon om et arrangement. Du kan nå denne enten ved å komme fra Neste Arrangement på hjemskjermen eller fra å klikke på et arrangement i kalenderen.

Informasjonen her hentes ut gjennom props som blir satt i Hjem. Dette er igjen hentet fra studentbergen, og hvert arrangement blir lagret på telefonen i JSON-struktur. Det betyr at vi enkelt kan hente ut hver undertittel fra arrangementet ved å eksempelvis ta event.title for overskrift. Her viser vi varighet av arrangement, tittel, ingress/introtekst og hovedtekst. Det er vil også vise en knapp som tar deg til Facebookarrangementet og en knapp dersom det er billettsalg.

Figur 20: Utdypende visning av et arrangement. Åpnes fra hjem- eller kalenderskjermen

Problemet med API-et til studentbergen er at det av en eller annen grunn følger med HTML-markører i responsen man får. Det betyr at tekstene er fylt med markører som (innholdsbokser), <p> (tekstelementer) og
 (linjeskift) for å nevne noen. Disse tolkes ikke av vår applikasjon, og printes dermed som ren tekst.

```
const filterHtml = () => {
  return event.article.replace(/\r\n\r\n/g, '\n').replace(/<[^>]*>?/gm, '');
}
```

Figur 21 : Kodesnutten som formaterer alle tags vekk fra en gitt HTML string.

Disse må vi derfor filtrere ut. Artikkelteksten blir kjørt gjennom en erstattingsfunksjon som bruker et "Regular Expression", også kalt regex. Dette definerer en struktur som strengen må følge, og vil erstatte de delene som ikke følger det. Det finnes dog situasjoner hvor regexer ikke kan filtrere HTML ordentlig. Som vår veileder poengterte er det vanskelig, om ikke umulig, å filtrere HTML gjennom regex på en god og alltid fungerende måte. Vi har valgt å gå for aggressiv filtrering. Det betyr i hovedsak at alt som ligger mellom to spissparanteser vil bli fjernet. Det finnes fortsatt tilfeller hvor dette ikke vil fungere, men det er en passende midlertidig løsning for prosjektets art og tidsramme.

```
const formatTimes = () => {
  if (event.startTime.length > 10) {
    startTime = formatTime(startTime);
    endTime = formatTime(endTime);
  }
}

const formatTime = (date) => {
  let d = new Date(date),
    time = d.getHours(),
    minutes = d.getUTCMinutes();
  return time + ":" + (d.getMinutes() < 10 ? '0' : '') + minutes;
}
```

Figur 22: Skjermdump av kode for tidsformatering.

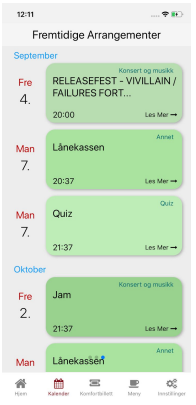
For å passe på at ingenting skal kræsje dersom et arrangement av en eller annen grunn skulle mangle en del, for eksempel ingress, så gjør vi sjekker på at disse eksisterer før vi prøver å printe teksten.

4.2.4 Kalender

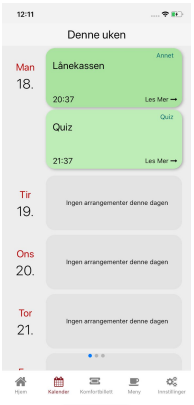
Kalenderen har som oppgave å vise kommende arrangement på Kronbar. I en modulbasert visning lister den opp arrangementene dag for dag nedover for hver uke.

Vi har valgt å vise denne uken og neste uke individuelt, for så å samle alle arrangement etter dette i en siste visning kalt 'Fremtidige Arrangementer'.

Skjermen henter arrangementer fra lokalt på telefonen - dette blir lagret først på hjemskjermen når applikasjonen først starter. Vi tar i bruk et bibliotek kalt AsyncStorage (heretter omtalt som 'lokallagring').



Figur 23: Skjermdump av fremtidige arrangement (Kommende måneder)



Figur 24: Skjermdump av kommende arrangement (Innen denne og neste uke)

Vi sorterer så arrangementene basert på uke, og legger de til i de respektive listene.

Kalender består av flere komponenter. Hver skjerm i karusellen er en CalendarWeek, som igjen består av flere CalendarDays som inneholder et CalendarEvent. På den måten kan vi gjenbruke kode og stå mye friere til å generere ting automatisk. Innledende tenkte tenkte vi at CalendarDay var nok, men vi innså at det kunne komme mer enn ett arrangement på en dag, for eksempel først på dagtid så kveldstid.

```
const filterJson = (data) => {
  let length = data.length;
  let thisWeekNr = new Date().getWeek();
  var nextWeekDate = new Date();
  nextWeekDate.setDate(nextWeekDate.getDate() + 7);
  let nextWeekNr = nextWeekDate.getWeek();
  let thisWeekEvents = [];
  let nextWeekEvents = [];
  let remainingEvents = [];
  for (let i = 0; i < length; i++) {
    let eventWeekNr = new Date(data[i].endTime).getWeek();
    if (eventWeekNr === thisWeekNr) {
      thisWeekEvents.push(data[i]);
    } else if (eventWeekNr === nextWeekNr) {
      nextWeekEvents.push(data[i]);
    } else if (eventWeekNr > nextWeekNr) {
      remainingEvents.push(data[i]);
    }
  }
```

Figur 25 : Utdrag av hvordan koden for arrangement-skjermen er strukturert

Vi valgte å gå for 2 uker frem i tid fordi de fleste av Kronbar sine arrangement skjer på en rotasjon på 2-4 uker. Derfor vil en tredje uke ofte repetere inneværende uke, og det kan derfor være enklere å bare gruppere alle senere arrangement sammen. Arrangement pleier heller ikke publiseres så langt frem i tid, med unntak av sjeldnere arrangement som konserter.

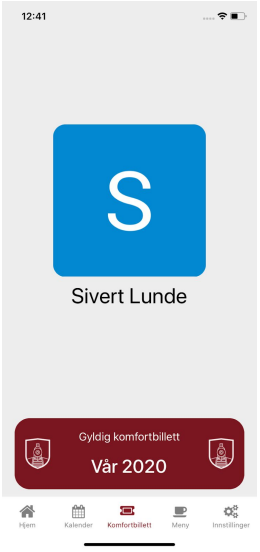
Det vil dog ikke kreve mye arbeid å endre dette i fremtiden (se figur 25) dersom situasjonen skulle endre seg.

4.2.5 Komfortbillett

Et av oppdragsgivers ønsker (som nevnt i kapittel 2.1.3), var å digitalisere komfortbilletten deres. Per i dag er dette et klistremerke som festes på studentbeviset til studenter. Det viktigste punktet i implementeringen av komfortbilletten var å sørge for at billetten ikke blir misbrukt. Den er personlig og skal ikke deles med andre.

Figur 27 : Komfortbillett-skjermen før mann er logget inn.

Vi valgte derfor å ta inspirasjon fra studentbevisappen [31] og dens design. Vårt design inneholder dermed et bilde av studenten, fullt navn og et dynamisk felt i bunnen av siden som kan modifiseres etter behov for å forhindre etterligninger og forfalskninger. Visningen av komfortbilletten avhenger av at en bruker har registrert seg i applikasjonen (se kapittel 4.2.7) og at personale fra Kronbar har registrert at brukeren har kjøpt billett (se kapittel 4.5).



Figur 26: Komfortbillett-skjermen etter at mann er logget inn og har gyldig komfortbillett.

Dersom kravene for å vise komfortbilletten ikke er oppfylt, vil komfortbillettsiden vise informasjon om komfortbilletten, sammen med et dynamisk felt i bunnen som avhenger av brukerdata.

Hvis bruker ikke er logget inn, vil den vise en innloggingsknapp som navigerer til innloggingssiden (se kapittel 4.2.6).

Hvis bruker ikke har fullført registreringen av profilen sin, vil den vise en registreringsknapp som navigerer til registreringssiden (se kapittel 4.2.7).

Hvis bruker ikke har registrert på sin bruker at komfortbillett er betalt, vil den vise en informasjonstekst om å henvende seg til baren dersom dette er galt.

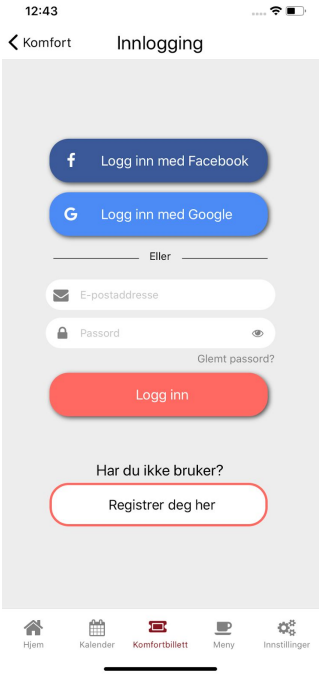
4.2.6 Innlogging

Dersom du prøver å navigere til 'Komfortbillett' uten å være logget inn vil du bli tatt til innloggingsskjerm. Her kan brukeren logge inn med Facebook, Google, eller e-post og passord. Når han er logget inn vil han bli sendt tilbake til komfortbillett.

Her er også muligheten for å registrere seg dersom man ikke har bruker fra før. Denne skjermen utdypes i kapittel 4.2.7.

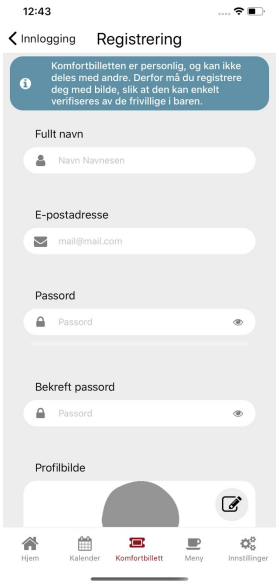
Innlogging skjer gjennom Firebase Authentication. Både logg inn med Facebook og Google vil opprette nye brukerobjekter dersom de ikke eksisterer fra før av. Brukere vil ikke trenge å huske noe brukernavn eller passord ved bruk av disse valgene. Dersom e-posten allerede er registrert, vil brukerne bli slått sammen. Andre scenarioer som hva som skjer dersom man mister tilgang til Facebook- eller Googlekontoen sin utdypes i kapittel 4.4.1.

Man vil uansett bli tatt videre til registreringsskjermen for å fullføre registreringen, blant annet på grunn av et krav om at ansikt må være godt synlig på profilbildet.



Figur 28 : Komfortbillett-skjermen når noen vil logge inn.

4.2.7 Registrering



Figur 29: *Skjermdump av registreringsskjerm*

Registreringsskjermens rolle er å la brukeren fylle ut nødvendig informasjon om seg selv. Her vil skjermen se litt forskjellig ut avhengig om man registrerer seg gjennom Facebook/Google eller ved e-post og passord.

Ved Facebook/Google vil man kunne endre navn og bilde. Bildet hentes direkte fra brukerobjektet i koden, og vil da være profilbilde brukeren har på Facebook eller Google. Det er ikke alle som har profilbilde, eller har et som ikke viser ansikt tydelig, så brukeren vil ha mulighet til å endre dette.

Ved registrering med e-post og passord vil også disse feltene være synlig (se figur 28).

Her bruker vi et bibliotek for å vise styrken på passord [24], og et for bildevalg [23].

Brukeren må også godkjenne vilkår før de får registrere seg. Deretter lagres bildet i lokallagringen, samt i Firebase Storage før brukeren blir sendt til komfortbillett-skjermen.

Hvordan brukerobjektet lagres i Firestore utdypes i kapittel 4.4.2. Når firebase godkjenner skjemaet bli det laget et dokument i *users*-samlingen i firestore som inneholder informasjonen til brukeren. Dokumentet kan nås ved *Firebase User Id*-en til brukeren.

4.2.8 Meny

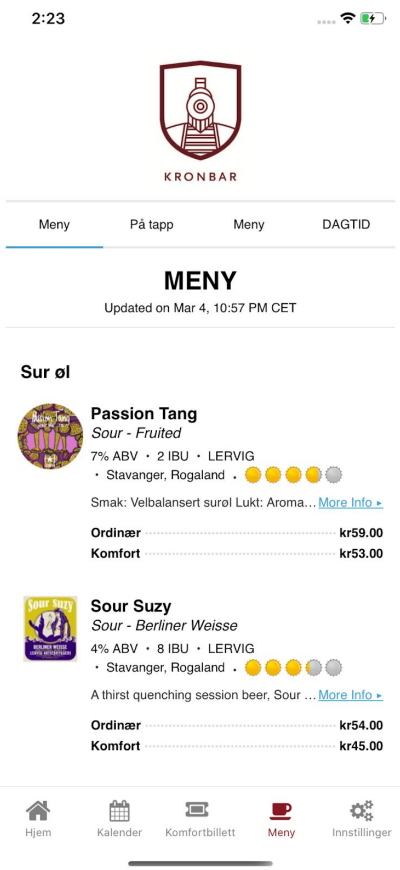
Kronbar bruker i dag et nettsted kalt Untappd for å vise menyen sin på TV-skjermer i lokalet. Tjenersiden av dette vises i kapittel 4.4.6, her vil vi ta for oss klientsiden.

Oppgaven til skjermen er å presentere produktene Kronbar selger. Det er en veldig enkel komponent som ikke består av mer enn en Webview og et stilark.

Her får vi all informasjon hentet fra Untappd, og med det priser både med og uten komfort.

Siden den hentes fra nettet, er mye av designet utenfor vår kontroll. Det er lagt inn hinder for at brukere skal kunne navigere i selve webviewen, og dersom man klikker på linker vil man bli tatt til nettleseren i stedet. På den måten unngår vi at brukeren blir sittende fast på tilfeldige sider.

Heldigvis er designet ganske i stil med resten av applikasjonen, med små overskrifter til venstre og et relativt modulært design. Den største fordelen er at endringer gjort i tjenersiden bli reflektert i applikasjonen så fort menyen lastes inn på ny.



Figur 30 : Skjermdump av bar-meny.

4.2.9 Innstillinger

Skjermen for innstillinger er ikke ferdigstilt. Denne inneholder per nå kun en knapp for å logge ut, men er tenkt å inneholde mer. Dette diskuteres videre under fremtidig arbeid i kapittel 7.2.8.

4.3 Lagring av data

Applikasjonen trenger at en del data skal lagres. Noe kan lagres på telefonen, men for å verifisere integriteten til dataen er også noen deler nødt til å lagres på en ekstern tjener. For å unngå at brukeren bruker unødvendig mye data, lagrer vi blant annet profilbildet lokalt.

4.4 Tjenester

Applikasjonen tar i bruk flere tjenester. Vi vil gå gjennom disse og forklare hva det er, oppsett av dem og hvorfor vi bruker de.

4.4.1 Firebase autentisering

Firebase er en tjeneste fra Google som samler diverse tjenester innenfor kategoriene utvikling, kvalitet, analyse og vekst. Disse tjenestene kan brukes uavhengig av hverandre, men kan også kombineres sømløst. Firebase er også svært gunstig for bruk i mindre applikasjoner grunnet google sine priser. Blant utviklingsverktøyene deres har vi valgt å benytte oss av brukerautentisering, firestore og skylagring. Samtlige av disse er gratis i bruk frem til en viss kvote er fylt. Disse kvotene er såpass store at vi i samarbeid med Kronbar konkluderte med at dette vil være en enkel og økonomisk fornuftig tjeneste å ta i bruk.

Firebase autentisering samler de mest brukte metodene for innlogging til en felles tjeneste. Her kan man enkelt skru av og på hvilke innloggingsmetoder man ønsker å støtte. En stor fordel med å bruke firebase til å samle disse er at alle brukere, uavhengig av innloggingsmetode, blir registrert som en firebasebruker. Dette åpner for at man enkelt kan implementere logikk basert på én type bruker, heller enn å skrive kode for flere forskjellige innloggingsscenarioer. Vi prioriterte de tre metodene vi opplever som mest brukt i dag; innlogging med Facebook, Google og e-post/passord.

Facebook

Innlogging med Facebook er etter gruppens erfaringer den generelt mest brukte metoden for innlogging i applikasjoner. I tillegg til å aktivere facebook innlogging i firebase, måtte vi konfigurere koblingen mellom Kronbar-appen, Firebase og en Facebookapplikasjon. Sistnevnte er gratis å sette opp og tar bare noen minutter.

Når en bruker logger inn med Facebook, blir de tatt til Facebook-applikasjonen dersom de har lastet ned denne, og nettleser hvis ikke. Her må de logge inn på facebook dersom de ikke allerede er logget inn. Videre blir de tatt til en skjerm i facebook som inneholder Kronbar sin logo og applikasjonen sitt navn hvor de blir bedt om å gi tillatelse til at Kronbar-appen kan hente og bruke noe av dataen deres fra Facebook. Straks dette er godkjent, opprettes en firebasebruker med dataene fra facebook dersom den ikke allerede eksisterer og brukeren blir sendt tilbake til Kronbar-appen med et firebase-autentiseringstoken. Først i Kronbar-appen skilles det på om brukeren er en ny bruker eller ikke. Oppførselen i forkant er lik uansett.

Google

Innlogging meg google er i likhet med Facebook svært hyppig brukt, og da spesielt på mobiltelefoner som kjører Android. Oppsettet til Google-innlogging i firebase var svært enkelt grunnet at Firebase er laget av Google. Her måtte vi bare sette opp koblingen mellom Kronbar-appen og Firebase.

Brukeropplevelsen er tilnærmet identisk når man logger inn med Google som med Facebook. Brukeren blir tatt til en innloggingsside for Google, blir bedt om å gi tillatelse til at applikasjonen kan hente og bruke noe dataen deres fra Google og blir så sendt tilbake til Kronbar-appen med et firebase-autentiseringstoken. Her er det også lik oppførsel for nye og eksisterende brukere.

E-post/passord

I tillegg til innlogging via sosiale medier er det viktig å gi brukere et alternativ. Her er e-post og passord et forventet alternativ. Dersom en bruker har registrert seg tidligere, trenger han bare å oppgi sine innloggingsdetaljer og trykke på logg inn. Hvis en ny bruker skal logge inn med e-post, må han først registrere seg i Kronbar-appen. Straks en bruker er registrert, opprettes en firebasebruker tilsvarende brukere fra Google og Facebook.

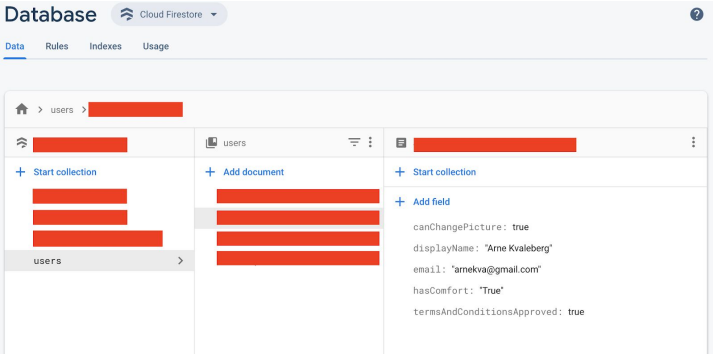
Firebasebrukere

En Firebasebruker kan holde på flere forskjellige innlogginger samtidig. Dersom en bruker registrerer seg med Google, for så å logge inn med Facebook i senere tid, vil disse to innloggingene kobles sammen (avhengig av brukers tillatelse) hvis de deler e-postadresse. Dette er en innebygget funksjon i Firebaseautentisering.

Videre, dersom noen skulle miste tilgang til Facebook- eller Googlekontoen sin, kan de få tilsendt en link for å nullstille passord til e-postinnlogging. Dette fungerer uavhengig av om brukeren tidligere har registrert seg, eller logget inn med e-post. På denne måten er brukeren sikret tilgang til kontoen gjennom e-postadressen sin.

4.4.2 Firestore

Firestore er Firebase sitt dokumentlagringssystem. Her lagrer vi brukerobjektene våre, og kan hente disse ut når nødvendig på klientsiden.



Figur 31 : Skjermdump av Firestore brukertabell.

Man kan også sette opp regler for skrive- og leserettigheter. I testmiljøet vårt er skriving og lesing tillatt av alle, som sett i figuren under.

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {

    match /{document=**} {
      allow read, write: if request.time < timestamp.date(2020, 7, 31);
    }
  }
}
```

Figur 32 : Skjermdump for setting av regler i Firestore.

Disse reglene må endres når applikasjonen skal ut i et produksjonsmiljø. Disse reglene er kun satt til helt åpent for enklere endring av verdier gjennom administratorverktøyet vårt (forklares i kapittel 4.5).

4.4.3 Firebase lagring

Firebase lagring tilbyr en enkel løsning for lagring av filer. I Kronbar-appen sitt tilfelle ønsket vi å lagre brukerne sine bilder som et tiltak mot misbruk av komfortbilletten ved at brukerbildet kan verifiseres av Kronbar. Dette gjøres ved at bildet som er knyttet til en brukerkonto, og dermed en komfortbillett hvis dette er kjøpt, lagres i Firebase med brukeren sin Firebase-brukerID som navn på bildefilen.

4.4.4 Studentbergen.no

```
getEventsFromStudBergen() {
  const url = 'https://studentbergen.netflex.dev'
  const bearerToken = [redacted]
  const kronbar_id = [redacted]
  fetch(url + "/api/student_org/" + kronbar_id + "/events?published=1", {
    "method": "get",
    "headers": {
      "Authorization": "Bearer " + bearerToken,
      "Content-Type": "application/json"
    },
  })
    .then((httpResponse) => {
      if (httpResponse.ok) {
        this.filterJson(httpResponse)
        return httpResponse;
      }
      this.setState({ response: 'it didnt work' })
      return Promise.reject("Fetch did not succeed! " + httpResponse.status);

    })
    .catch(err => console.log(err));
}
```

Studentbergen er en nettside som prøver å samle alle arrangementer i bergen som er rettet mot studenter. Kronbar har i lengre tid lagt inn sine offentlige arrangement her. Studentbergen har et API hvor man kan hente ut og opprette egne arrangement.

Figur 33 : Skjermdump av GET-forespørsel mot Studentbergen

Kodearkitektur

Arrangementene hentes ut ved en GET-forespørsel mot studentbergen. Denne forespørselen må inneholde en 'token' for å kunne autentiseres. Denne skal egentlig holdes privat da den

også er gyldig for POST-forespørsler, men vi fikk opprettet en som kun er gyldig for GET av studentbergen sine utviklere. Dette gjør at vi kan gjøre forespørslene direkte fra telefonen uten en mellomtjener.

Vi får et JSON-objekt av alle publiserte arrangement de siste 6 måneder i respons, så vi må igjen filtrere bort disse. Her sorterer vi de også i rekkefølge.

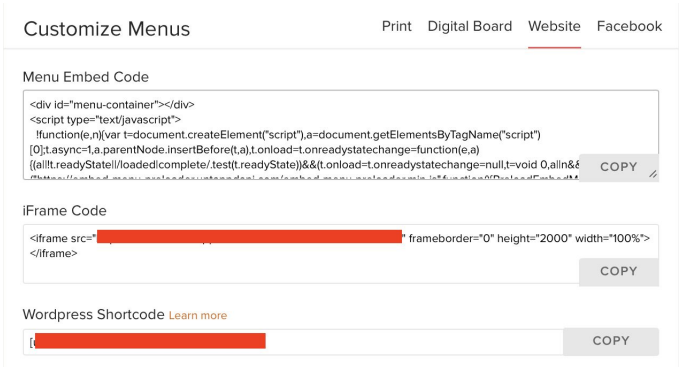
JSON-objekter er lette å håndtere, og vi kan enkelt hente ut tittel, bilde, beskrivelse, billettlenker med mer fra objektet.


```
{
 "id": 12115,
 "name": "Jam",
 "arrangmentsinfo-collection": null,
 "norwegian-version": null,
 "english-version": null,
 "image": {
     "file": 11570,
     "path": "1578495537\/534f1e7472fcf0a2544b37a2288ca16131bdefe702def9b2b04c84f69d8864b6.jpg",
     "name": null,
     "description": null
 },
 "startTime": "2020-03-10T17:00:00.000000Z",
 "available_no": true,
 "available_en": true,
 "endTime": "2020-03-10T22:59:00.000000Z",
 "form": null,
 "imageCaption": null,
 "imageCaption_en": null,
 "thematicProfile": null,
 "intro": null,
 "name_en": "Jam",
```


Figur 34 : *Skjermdump av svar på forespørsel mot studentbergen*

I figuren over ser man 20 av 100 linjer som returneres fra API-et. Her får vi dataen vi trenger til å vise på skjermen.

4.4.5 Untappd



Figur 35 : Skjermdump av Untappd iframe generator.

Kronbar bruker i dag Untappd til å vise menyene sine på skjermer i lokalet. Untappd har støtte for å også lage 'iframes' av menyen. Dette betyr at det er støtte for å lage javascriptkomponenter av innholdet som vi enkelt kan legge i appen vår.

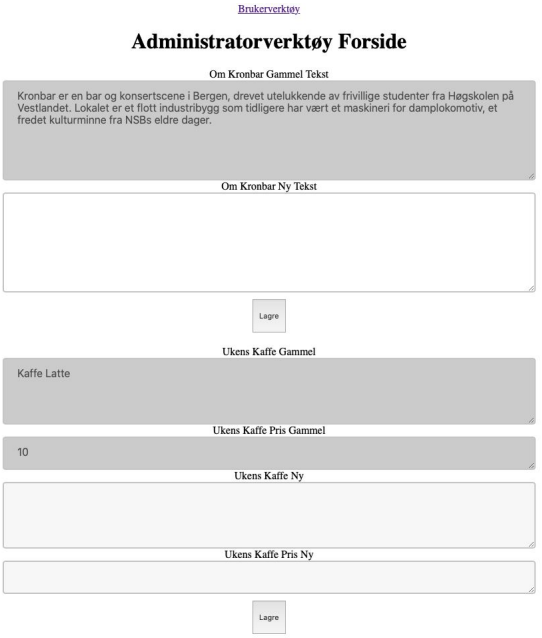
React Native har ikke innebygget støtte for 'iframes', men det finnes et bibliotek kalt webview vi kan bruke for å oppnå samme resultat.

4.5 Administratorverktøy

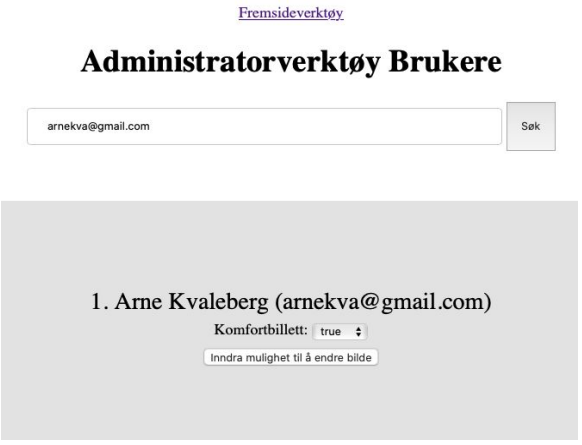
Som et gjennomførbarhetsbevis (*Engelsk: "Proof of concept"*) utviklet vi et veldig enkelt administratorverktøy for applikasjonen. Denne funksjonaliteten skal egentlig ligge i internapplikasjonen (jf. prosjektgruppe med oppgave om applikasjon for interne), men grunnet tidspress ble ikke dette mulig å samkjøre.

Vi satt derfor opp vår egen veldig enkle, lokale nettside for å gjøre endringer i databasen. Denne nettsiden krever ingen innlogging og vil ikke fungere når databasereglene blir satt opp.

Den er ment kun som et bevis på mulig gjennomførbarhet, samt for rask testing av verdiendringer.



Figur 36 : Verktøyet for endring av informasjonen på hjemskjermen.



Figur 37 : Skjermdump av administrasjonsverktøy for brukere.

5 EVALUERING

Evaluering er en viktig del av utviklingen. Det er viktig å få tilbakemeldinger fra prosjekteier, både når det gjelder produktet og selve utviklingsløpet. Siden vi også ønsker å teste bruken av applikasjonen ønsker vi også tilbakemeldinger fra andre en prosjekteier.

5.1 Evalueringsmetode

Vi beskrev tidligere i kapittel 3 at vår hovedkanal for evaluering er gjennom kommunikasjon med oppdragsgiver. Vi har prøvd å hatt møter rundt annenhver uke for å presentere resultatet vi har. Det har vært litt ekstra krevende på grunn av situasjonen med Covid-19, så vi har måttet gjøre det over Skype. Det gjør det litt vanskeligere å kunne vise produktet ordentlig.

5.1.1 Tilbakemelding under arbeidsperioden

Under arbeidsperioden har vi kontinuerlig vært i kontakt med bedriften, særlig i starten for å få en god overordnet idè av hva som var tenkt og hvordan vi skulle håndtere problemstillingen. Når vi begynte utviklingen var ikke behovet for hjelp særlig stor, og kontaktperioden ble redusert til rundt annenhver uke.

Vi hadde også planlagt å ha regelmessige brukertester sammen med de frivillige på Kronbar, men dette viste seg å være svært vanskelig når HVL valgte å stenge ned Covid-19. Dette inkluderte at Kronbar også ble stengt ned, da det er en del av HVL-bygget. Siden applikasjonen krever et veldig stort utviklingsmiljø på datamaskinen for å kjøre, er det svært vanskelig å kunne få andre enn oppdragsgiver og nærmeste bekjente til å teste.

5.1.2 Sluttmøte med oppdragsgiver

Gruppen gjennomførte et sluttmøte med oppdragsgiver 27. mai. Her ble sluttproduktet presentert og gjennomgått, samt prosessen bak utviklingen av applikasjonen.

På dette møtet ønsket vi å få en evaluering på hvor godt gjennomført arbeidet var, og hvordan løsningen stilte seg i forhold til hvordan oppdragsgiver så for seg at løsningen skulle bli. Denne evalueringsmetoden er den mest verdifulle tilbakemeldingen, da det best kan analysere resultatet ved å sammenligne nådd resultat med forventet.

Hele intervjuet kan leses i vedlegg 9.5.

Etter intervjuet kan vi konkludere med følgende:

- Prosjektet har nådd lenger enn oppdragsgiver forventet. Gruppen har klart å oppfylle kravene Kronbar så på som viktigst, og har klart å utvikle en applikasjon som etter hvert sannsynligvis kan tas i bruk.

- Oppdragsgiver forventet ikke at komfortbilletten skulle fungere så godt som den gjorde i applikasjonen, og hadde egentlig tenkt at denne kunne passet som et egen bacheloroppgave. Dette gjør videreutviklingen mye enklere for oppdragsgiver.
- Gruppen som utvikler internapplikasjonen har også gått for Firebase som tjenerdel, noe som gjør videre integrasjon mellom appen lettere når de bruker delt tjener.
- Designet på applikasjonen var ikke et hovedfokus, men det ble veldig verdsatt at det var lagt litt tid i design og utseende.
- Det var ingenting å utsette på forstanden av appen.
- Oppdragsgiver ser for seg at appen skal videreutvikles på, og lanseres etter sammenslåing av ekstern og intern applikasjon

5.2 Evalueringsresultat

Etter at evalueringsprosessen var gjennomført, dannet det seg et godt grunnlag for vurdering av hvilken grad problemstillingen ble besvart. Resultatet av evalueringen viser både oppfylte krav, og hvilke områder det kunne bli gjort mer arbeid.

Oppdragsgiver mener at vi har gjort mer enn nok, og at kravspesifikasjonene er oppfylt i større grad enn forventet. Det var dog noen krav som ikke ble påbegynt, men disse er av så lav prioritet at det ikke spiller en stor rolle for oppdragsgiver. Vi kan da konkludere med:

- Produktet har fått kjernefunksjonene kalender og komfortbillett implementert.
- Selv om applikasjonen ikke er i stand til å bli lansert til produksjon, er grunnlaget dannet og vil bli brukt til videre utvikling.
- Funksjonaliteten og designet av appen er implementert på en god og intuitiv måte.
- Prosjektpartene er i enighet om at prosjektet har oppnådd forventet resultat, og vært i fordel av alle involverte.

6 DISKUSJON

I dette kapittelet vil vi diskutere hva vi har gjort, og konsekvensene av dette. Vi vil se på hvordan valgene våre har ført til de resultatene vi har fått, og hva vi ville gjort annerledes hvis vi skulle gjort det på ny.

6.1 Prosjektstart

Vi kjente til oppgavebeskrivelsen i god tid før vi startet med arbeidet. Dette ga oss tid til å smått begynne å tenke ut løsninger og design på applikasjonen. Problemstillingen vår er relativt rett frem på hva som skal lages, men design og måten funksjoner blir implementert krevde mye modellering. Vi satt av en uke til modellering i starten, og sammen med oppdragsgiver ble vi enige om en god retning å starte i.

Vi ble enige om å møtes på lab på skolen 3 dager i uken for å jobbe med prosjektet, da to av medlemmene har deltidsjobb ved siden av studiet. Da skolen ble stengt ned grunnet Covid-19 ble vi nødt til å møtes hos en av gruppemedlemmene i stedet.

6.2 Prosjektgjennomføring

6.2.1 Oppstartsfasen

Oppstartsfasen besto av å bestemme hvilke verktøy vi skulle bruke for utviklingen, og hva vi kom til å trenge av fastvare og mykvare. Vi visste vi skulle bruke React Native, så vi fant VS Code som et passende utviklingsmiljø for dette. Videre fant vi ut at vi også trengte Xcode og Android Studio for simulering av enheter.

Den største utfordringen var å få den tomme applikasjonen til å kjøre på simulatorene. Det skulle vise seg vanskeligere enn antatt. En annen utfordring var læringskurven til React Native. Vi var kjent med Javascript før prosjektstart, og vi hadde så vidt vært innom Angular og Node gjennom diverse bedriftspresentasjoner, men ikke nok til å vite hvordan vi skrev det selv. Vi ble avhengig av eksterne ressurser for å lære oss det, og det gjorde oppstartsfasen lite effektiv.

I utgangspunktet skulle tjenersiden vår være en egenlaget Java-applikasjon for å behandle innkommende forespørsler. Dette ble ved et senere tidspunkt endret til Google Firebase, men tidsskjemaet vårt ble satt opp til at det skulle brukes et par uker på oppsett av denne tjeneren.

6.2.2 Utviklingsfasen

Når vi hadde blitt kjent med rammeverkene og utviklingsverktøyene kunne vi begynne å lage komponenter til applikasjonen. Vi bestemte oss for å prioritere funksjonalitet over utseende, og vi

begynte med kalenderen. Kalenderen er spesifisert som det viktigste kravet i oppgavebeskrivelsen. Vi fikk raskt API-et til å fungere, men det krevde en del filtrering før det oppførte seg som forventet. Vi har tidligere brukt API-er i Javascript-sammenheng før, så vi visste hvordan det fungerte.

Når kalenderen var ferdig begynte vi å implementere Firebase for innlogging. Dette er også mest bare API-kall, og det gikk relativt fint å få til.

Under hele fasen har vi vært plaget med diverse feil som man ofte får i IT-relaterte prosjekter; simulatorer nekter å starte, konflikter i git, feil i Node serveren med mer. Det har derfor gått mange timer på feilsøking i utviklingsfasen, noe vi gjerne skulle unngått. Disse timene går på bekostning av videreutvikling av brukergrensesnitt og funksjonalitet.

Etter innlogging virket gikk vi videre til å implementere komfortbilletten. Denne lagres i Google Firestore, og vi måtte knytte Firebase-brukere opp mot Firestore-dokumentene. Dette gjøres enkelt ved å gi Firestore-dokumentet et navn som tilsvarer bruker-iden som er en unik string for hver bruker.

Mot slutten av prosjektet har vi klart å lage en applikasjon vi selv er fornøyd med. De største utfordringene under denne fasen var sære feilmeldinger og generell feilsøking. Konsekvensene var at vi mistet mye tid som kunne gått til utvikling.

6.3 Utviklingsmetoder

Vi besluttet tidlig i prosjektfasen å benytte oss av Kanban som utviklingsmetode. Dette valget ble gjort basert på positive erfaringer fra tidligere prosjekter i skoleløpet, i tillegg til en vurdering av Kanban opp mot andre utviklingsmetoder som Scrum og Waterfall.

Vi så flere fordeler ved Kanban som appellerte til oss. Blant disse var den svært høye graden av fleksibilitet. Dette vurderte vi som essensielt i et prosjekt hvor vi ville lære svært mye nytt i form av verktøy vi brukte, samt språk for utviklingen, i løpet av kort tid. Uten tidligere erfaringer innenfor store deler av utviklingen av prosjektet var vi avhengig av å kunne tilpasse oss på kort tid. Scrum var også en kandidat til å fylle dette behovet, men vi vurderte sprinter på 1-2 uker til å være bindende i en for stor grad i forhold til forventede antall endringer underveis.

Kanban har vært et godt valg for vår gruppe i den typen prosjekt vi har jobbet med. Vi har ved flere anledninger benyttet oss av muligheten til å ta store avgjørelser underveis i prosjektet etterhvert som vi har gjort oss nye erfaringer. Et eksempel på dette er beslutningen om å gå vekk fra planen om å utvikle en egen tjenerside. Vi så større fordeler av å gå over til Google-tjenesten Firebase (les mer om Firebase i kapittel 4.4.1), både m.t.p. besparelser i tid, og muligheten for å implementere flere ønskede funksjoner. Uten å være knyttet til en sprint i Scrum, eller en lineær utviklingsmodell i Waterfall, bestemte vi oss på kort tid om å droppe fremgangen vi hadde gjort på vår egen tjenerside, for å heller bruke Firebase. Dette ble et avgjørende valg for fremgangen og det endelige resultatet av prosjektet vårt.

6.4 Brukergrensesnitt

Brukergrensesnittet i applikasjonen har endret seg drastisk over utviklingsløpet. Designet og stilen har alltid vært nedprioritert, og det innledende arbeidet gikk hovedsakelig til funksjonalitet. Når kjernefunksjonene etter hvert var kommet på plass, kunne designet prioriteres. Våre tanker er at selv om vi bygger en prototype er brukergrensesnittet likevel en viktig del, og viktig å ha på plass når applikasjonen skal brukertestes. Komponentene er også utviklet med et spesifikt design i tankene, og det gir oppdragsgiver en bedre forståelse av hvordan applikasjonen er tenkt.

6.5 Valg av funksjonalitet

Mot slutten av prosjektet er det åpenbart at vi ikke får oppfylt alle kravspesifikasjonene. Vi hadde valgt ut det vi mente var viktigst ut i fra kravspesifikasjonene, men det hadde vært ønskelig å kunne implementere betaling med eksempelvis Vipps. Vi har dog fått utviklet kjernefunksjonalitetene av applikasjonen; kalender, innlogging og en databasestruktur for brukere og komfortbilletter.

Firebase er også en tjeneste som kanskje vil kunne koste penger. Det er dog svært mye som skal til for at denne applikasjonen overskrider begrensningene til gratisversjonen. Blant annet kan det sendes ut ti tusen SMS-er per måned, det kan være opp mot tjue tusen lesinger og skrivinger av databasen hver dag og vi kan lagre opp til 5 gigabyte med data i *Storage*. Hvis vi regner med at Kronbar maksimalt får fem hundre skrivinger og tusen lesninger til dagen, som er et veldig generøst estimat, er det fortsatt bare 5% av dagsgrensen. Den eneste plassen det kan tenkes at man kan få inn litt kostnader er lagring, hvor det bare er 5 GB inkludert. Prisen for å øke dette er dog ikke høy; 24 øre per ekstra gigabyte og 50 øre per ti tusende opplasting. Prisen for dette vil derfor sannsynligvis ikke ha noen stor innvirkning på driften av applikasjonen.

7 KONKLUSJON OG VIDERE ARBEID

Målet med prosjektet var å utvikle en prototype av en applikasjon for Kronbar. Som vi har diskutert gjennom rapporten, har målet aldri vært å ferdigutvikle applikasjonen klar for lansering, men heller lage et bevis på gjennomførbarhet.

7.1 Måloppnåelse

Prosjektet har ikke oppnådd alle kravene som ble stilt i kravspesifikasjonen, men de viktigste er blitt ferdigstilt. Det er dog fortsatt store forbedringspotensialer, både når det gjelder funksjonalitet og design. De største forbedringer og punkter for videreutvikling vil vi ta opp under.

7.2 Fremtidig arbeid

Under følger flere punktet vi gjerne skulle ønsket å kunne implementere selv, men ikke har mulighet til på grunn av tid.

7.2.1 Administratorverktøy

Vi har i dag et veldig enkelt verktøy for å gjøre endringer i Firestore (se kapittel 4.5). Dette er et primitivt verktøyet som ikke har autentisering eller feilhåndteringer. Det kan derfor ikke tas i bruk av personalet. For å unngå å logge inn i Firebasekonsollen for å gjøre endringer burde dette verktøyet ferdigutvikles.

7.2.2 Integrering med internapplikasjon

En annen prosjektgruppe har også en oppgave fra Kronbar. Denne gruppen lager en tilsvarende applikasjon som oss, bare rettet mot de interne i stedet for besøkende. Et mål å nå her vil være at disse enten integreres til en eller samkjøres.

7.2.3 Regelsetting i database

I dag er det ingen regler i Firebasedatabasen. Dette betyr at brukere kan hente ut informasjon om andre brukere dersom de finner nøklene. Dette kan enkelt endres ved å sette regler i Firebasekonsollen om at brukere kun kan hente ut egen info.

Koden for dette kan se slik ut:

```
{ "rules": {
        "users": {
                "$uid": {
                        ".read": "$uid === auth.uid" }
                }
        }
}
```

En annen regel som burde settes er automatisk ugyldiggjøring av komfortbilletter ved starten av hvert semester.

Databasen er åpen da det gjør testing gjennom administratorverktøyet vårt en del enklere. Reglene kan settes på følgende måte:

7.2.4 Integrasjon av betalingsløsninger i applikasjonen

Applikasjonen vår har ikke mulighet for betaling for komfortbillett, dette må fortsatt gjøres i kassen. Ved å implementere for eksempel Vipps-betalingen kunne man sluppet leddet med å gå i baren for å betale. Da kan kontoen automatisk bli tilegnet komfortbillett ved godkjent kjøp.

7.2.5 Notifikasjoner og varslinger

En bruker skulle gjerne hatt mulighet til å trykke på et varslingsikon når han er inne på et arrangement og kunne sette en nedtelling frem i tid for å få en notifikasjon når arrangementet skjer. Det gjelder også at man gjennom administrasjonsverktøyet skal kunne sende varslinger til brukere om generelle promoteringsartikler som ukens kaffe.

7.2.6 Deling av arrangement og kalenderintegrering

Det er i dag ikke mulig å eksportere kalender til telefonens egne kalender. Dette kunne gitt brukere mulighet til å konstant se kommende arrangement i sin egen kalender kombinert med andre ting.

7.2.7 Referanser til eksterne bibliotek

Applikasjonen tar i bruk flere moduler som må refereres til ved kommersielt bruk, jf. lisensene de brukes under. Dette kan implementeres som en underkategori i ‘Innstillinger’-skjermen slik det blir gjort i andre mobilapplikasjoner som Spotify.

7.2.8 Ferdigstilling av Instillinger

Skjermen for innstillinger er ikke ferdig, og holder per nå kun på en “Logg Ut”-knapp. Vi ville implementert følgende hvis det var tid:

- Brukerverktøy
    - Deaktivere/Slette bruker og all info lagret om deg
        - Metode i firebase for dette
- Logg ut
- Lisenser/Referanser til eksterne bibliotek
- Kontaktinformasjon til Kronbar
- Diverse brytere for å skru av/på ting (Mørkt tema, varslinger)

8 REFERANSER

1. Facebook (2015). *React Native Documentation*. Tilgjengelig fra: https://github.com/facebook/react-native (Hentet 25. mars 2020)
2. Google (2014). *Firebase Documentation.* Tilgjengelig fra: https://firebase.google.com/docs (Hentet 14. april 2020)
3. Node.js Foundation (2010) *Node.js Documentation.* Tilgjengelig fra: https://nodejs.org/en/docs/ (Hentet 25. mars 2020)
4. Trello. Tilgjengelig fra: https://trello.com (Hentet 24. mars 2020)
5. TeamGantt (u.å.). *What is a Gantt Chart?* Tilgjengelig fra: https://www.teamgantt.com/what-is-a-gantt-chart (Hentet 18. mars 2020)
6. Appicon Team (u.å.). *Making iOS and Android app icons.* Tilgjengelig fra: https://appicon.co (Hentet 10. april 2020)
7. StudentBergen (u.å.). *Dokumentasjon for arrangement API.* Linken er ikke offentlig tilgjengelig (Hentet 2. april 2020)
8. Appstud (2019) *Add a splash screen to a React Native app*. Tilgjengelig fra: https://medium.com/@appstud/add-a-splash-screen-to-a-react-native-app-810492e773f9 (Hentet 14. mai 2020)
9. Adam Wathan og Steve Schoger (2018). *Building your color palette*. Tilgjengelig fra: https://refactoringui.com/previews/building-your-color-palette/ (Hentet 12. mai 2020)
10. Facebook (2014). *React Documentation.* Tilgjengelig fra: https://github.com/facebook/react (Hentet 25. mars 2020).
11. FontAwesome (u.å.). *Using in a package manager.* Tilgjengelig fra: https://fontawesome.com/how-to-use/on-the-web/setup/using-package-managers (Hentet 3. april 2020)
12. React Native Elements (u.å.). *React Native Elements Documentation.* Tilgjengelig fra: https://react-native-elements.github.io/react-native-elements/docs/getting_started.html (Hentet 28. april 2020)
13. React Native Community (u.å.). *React Native Modal Documentation*. Tilgjengelig fra: https://react-native-elements.github.io/react-native-elements/docs/getting_started.html (Hentet 12. mai 2020)
14. Aaron_ta (2019) *React Native Animated Header Documentation*. Tilgjengelig fra: https://www.npmjs.com/package/react-native-animated-header (Hentet 14. mai 2020)
15. Untapped (u.å.) *Untappd documentation*. Linken er ikke offentlig tilgjengelig (Hentet 2. mai 2020)
16. React Native Community (u.å.) *React Native Webview Documentation*. Tilgjengelig fra: https://github.com/react-native-community/react-native-webview (Hentet 27. april 2020)
17. Leecade (u.å.) *React Native Swiper*. Tilgjengelig fra: https://github.com/leecade/react-native-swiper (Hentet 13. mai 2020)

18. Expo (u.å.) *React Navigation Documentation*. Tilgjengelig fra: https://reactnavigation.org/docs/getting-started (Hentet 1. april 2020)
19. React Native Community (u.å.) *AsyncStorage Documentation*. Tilgjengelig fra: https://github.com/react-native-community/react-native-async-storage (Hentet 22. april 2020)
20. Archriss (2017) *React Native Snap Carousel Documentation*. Tilgjengelig fra: https://github.com/archriss/react-native-snap-carousel (Hentet 14. mai 2020)
21. React native Community (2018) *React Native Google SignIn Documentation*. Tilgjengelig fra: https://github.com/react-native-community/google-signin (Hentet 2. mai 2020)
22. Facebook (2015) *React Native fbsdk Documentation.* Tilgjengelig fra: https://github.com/facebook/react-native-fbsdk (Hentet 2. mai 2020)
23. React Native Community (2016) *React Native Image Picker*. Tilgjengelig fra: https://github.com/react-native-community/react-native-image-picker (Hentet 2. mai 2020)
24. Pritish Vaidya (2019) *React Native Password Strength Meter.* Tilgjengelig fra: https://www.npmjs.com/package/react-native-password-strength-meter (Hentet 3. mai)
25. Handlebar Labs (2018) *Fade in an Image with React Native with the Animated API.* Tilgjengelig fra: https://www.youtube.com/watch?v=vzPmI0GCDPM (Hentet 12. mai 2020)
26. Benjamin Todts (2019) *React Native's Animated.loop: Invoking A Callback Whenever An Iteration Finishes*. Tilgjengelig fra: https://medium.com/@benjamintodts/react-natives-animated-loop-invoking-a-callback-whenever-an-iteration-finishes-1c3581d38d54 (Hentet 13. mai 2020)
27. Microsoft (u.å.) *Visual Studio Code Documentation.* Tilgjengelig fra: https://code.visualstudio.com/docs (Hentet 16. april 2020)
28. Google (u.å.) *Android Developer Documentations*. Tilgjengelig fra: https://developer.android.com/docs (Hentet 16. april 2020)
29. Apple (u.å.) *Apple Developer Documentation.* Tilgjengelig fra: https://developer.apple.com/documentation/ (Hentet 16. april 2020)
30. React Native (u.å.) *Running On Device*. Tilgjengelig fra: https://reactnative.dev/docs/running-on-device (Hentet 15. april 2020)
31. Direktoratet for IKT og fellestjenester i høyere utdanning og forskning (u.å.) *Studentbevis*. Tilgjengelig fra: https://www.fellesstudentsystem.no/studentbevis/ (Hentet 2. mai 2020)

9 VEDLEGG

9.1 Gantt diagram

9.2 Risikoanalyse

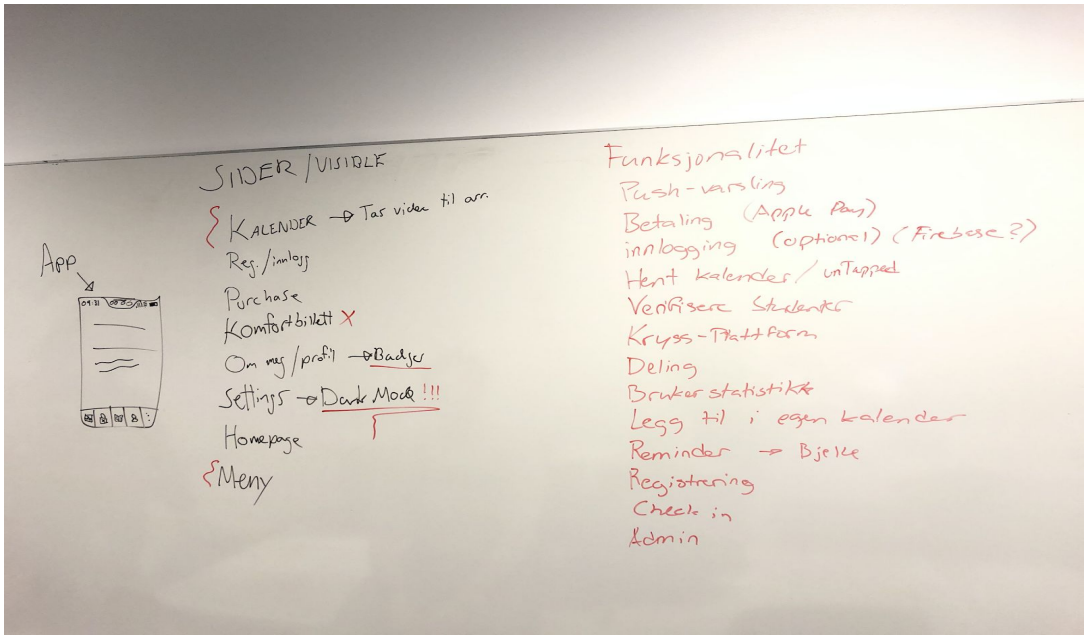
	S	K	RF	Tiltak
Sykdom	8	5	40	God hygiene, passe på å ikke overarbeide.
Dårlig samarbeid innad i gruppe	3	8	24	Forhåndsdefinere regler og arbeidsmetoder, og sørge for god kommunikasjon mellom gruppemedlemmene.
Dårlig kommunikasjon med oppdragsgiver	3	4	12	Holde jevne og gode møter, og jobbe smidig sammen med oppdragsgiver
Skjevfordeling av arbeid	2	4	8	Fordele oppgaver jevnlig, og sørge for at alle gruppemedlemmer jobber med oppgaver de klare å løse
Manglende kompetanse	6	5	30	Lære av hverandre, og sette av god tid til å lære det.
Misforståelse av kravspesifikasjon	5	5	25	Stille spørsmål dersom noe er uklart, ha jevnlige oppdateringsmøter for å vise progresjonen.
Utstyr og/eller mykvare som ikke fungerer	9	5	45	Bruke samme versjoner av mykvarer, ikke oppdatere til nye versjoner med mindre vi vet at rammeverkene støtter det.
Dårlig tidsplanlegging	7	5	35	Sørge for at vi holder oss så godt som mulig til tidsplanen, og justerer kursen underveis dersom vi ser at det er nødvendig.

S = Sannsynlighet på skala mellom 1-10

K = Konsekvens på skala mellom 1-10

RF = Risikofaktor (S * K)

9.3 Innledende modellering



9.4 Innledende skjermmodellering

Skjerm	Innhold
Hjem	Skal vise neste arrangement, åpningstider, generelle arrangementer og diverse promoteringer
Kalender	Enkel ukesvisning av kommende arrangementer
Arrangement	Utfyllende informasjon om et arrangement
Komfortbillett	Bevis på betalt komfortbillett - knyttet til bruker
Innlogging / Registrering	Kombinert skjerm for registrering og innlogging
Innstillinger	Administrering av bruker

9.5 Sluttmøte med oppdragsgiver

Magnus Marthinsen (MM), Arne Kvaleberg (AK)

AK: Går det greit om vi kan bruke sitater av svarene dine i teksten?

MM: Ja, det går helt fint.

AK: Hvordan er resultatet sammenlignet med hva du/dere så for dere?

MM: Fått til mer enn hva vi trodde, tenkte komfortbillett kunne være en helt egen oppgave. Skulle ønske jeg hadde noe å kritisere på, men finner ikke noe. De mindre prioriterte kravene var ikke viktige i det hele tatt, mer bare en ettertanke mens vi laget oppgaveteksten. Det viktigste for oss var å få en app som inneholder grunnfunksjonalitetene, og som beviser at prosjektet er mulig å gjennomføre. At man ikke kan sette varslinger på et arrangement er absolutt ikke noe problem for oss. Vi vil heller kunne publisere enn app med kalender og komfortbillett, også heller implementere varslinger i en senere versjon av appen.

AK: Hva synes du om applikasjonens design?

MM: Jeg har ett ord: Fantastisk. Det er et veldig fint design. Det er åpenbart at man ikke klarer å lage et 100% fantastisk design med gode color palettes og hele pakka, men på den korte tiden dere har hatt er det veldig bra.

AK: Hva synes du om applikasjonens funksjonaliteter?

MM: Veldig bra, og intuitivt grensesnitt. Veldig veldig bra. Jeg tror alle kunne tatt i bruk denne appen slik den ser ut nå. Klart, vi skulle gjerne testet appen blant besøkende ved baren, men det er jo som dere har nevnt ikke mulig under Corona-reglene fra Bergen Kommune og HVL. Vi vil jo bygge videre på det dere har utviklet, og vi vil absolutt teste den med både frivillige og besøkende fra alle studieretninger så fort vi har mulighet. Jeg har sett på Microsoft App Center, og vil se på mulighetene for å kunne laste opp en test-versjon der så folk kan prøve.

AK: Er det noe du er misfornøyd med eller skulle ønske ble gjort annerledes?

MM: Nei. Vi forventet som sagt aldri en 100% ferdig app, og så lenge grunnfunksjonaliteten er på plass er alt fint. At litt tekst overflower på små skjermer når dere har glemt dynamisk justeringer går helt fint, det er ikke noe stress.

AK: Hva synes du om kommunikasjonen og arbeidsflyten

MM: Det var vel ikke noe særlig behov for mye kontakt. Jeg føler dere forstod oppgaven godt, og det viser jo med at dere har truffet midt i blinken. Dere har jo ringt og sendt meldinger når dere har hatt problemer eller spørsmål, og inkludert oss i de større avgjørelsene som å gå for

Firebase.

AK: Andre kommentarer?
MM: Dere har vært flinke. Jeg har ikke noe særlig mer å si enn det.