Høgskulen på Vestlandet

BACHELOROPPGAVE

Mobilapplikasjon for interne ved Kronbar

Mobile application for volunteers at Kronbar

Øystein Helmersen Vikane, Kjetil Dyrland & Vilhelm Skagestad

Dataingeniør
Fakultet for ingeniør- og naturvitenskap (FIN)
Veileder: Bjarte Kileng
02.06.2020

Tittelside for hovedprosjekt

Rapportens tittel: Mobilapplikasjon for interne ved Kronbar	*Dato:02.06.2020*
Forfattere: Øystein Helmersen Vikane, Kjetil Dyrland & Vilhelm Skagestad	*Antall sider u/vedlegg: 30* *Antall sider vedlegg: 5*
Studieretning: Dataingeniør	*Antall disketter/CD-er:0*
Kontaktperson ved studieretning: Bjarte Kileng	*Gradering:* Ingen
Merknader: Ingen	

Oppdragsgiver: Kronbar	*Oppdragsgivers referanse:* Ingen
Oppdragsgivers kontaktperson: Magnus Marthinsen & Sindre Ranaweera	*Telefon:* *45188818 / 98056005*

Sammendrag:

Denne rapporten beskriver prosessen av å utvikle en mobilapplikasjon for frivillige ansatte hos Kronbar. Applikasjonen skal gjøre håndtering av vakter og info enklere og mer tilgjengelig enn nåværende løsning.

Stikkord:

Mobilapplikasjon	Kronbar	React-native

Forord

Denne rapporten dokumenterer bachelorprosjektet - Mobilapplikasjon for interne ved Kronbar.

Rapporten går gjennom prosessen ved å utvikle en mobilapplikasjon for Kronbar som fungerer som kunde for oss. Prosjektet er gjennomført av en gruppe studenter ved Dataingeniør linjen ved Høgskolen på Vestlandet; Øystein Helmersen Vikane, Kjetil Dyrland og Vilhelm Skagestad.

Vi vil takke Kronbar og Leder for Kronbar, Magnus Marthinsen for muligheten til å ha denne interessante oppgaven som vår bacheloroppgave.

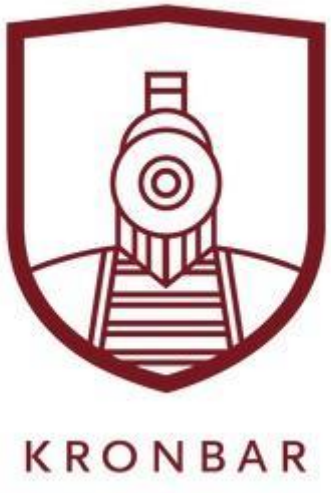
Vi vil også rette en stor takk til Bjarte Kileng som har vært vår veileder gjennom prosjektet for den gode hjelpen og veiledningen vi har fått til arbeidet vi har utført på denne oppgaven.

Innholdsfortegnelse

1 Innledning

1.1 Motivasjon

Å utvikle en intern mobilapplikasjon for Kronbar fremsto som en av de mer interessante bachelor oppgavene som gruppen kunne velge mellom og i tillegg samspiller oppgaven mest med kunnskapsgrunnlaget gruppen har tilegnet seg gjennom 2,5 år med studier. Alle i gruppen har hatt applikasjonsutvikling i emnet "Mobile og distribuerte applikasjoner" (DAT153) tidligere i semesteret og dermed så gruppen på det som en sjanse til å kunne bygge videre på kunnskapen dette emnet. Det å utvikle en applikasjon for en kunde og skape et etterspurt produkt er også en faktor som er gjør denne oppgaven enda mer spennende.



Figur 1 Kronbar sin logo

1.2 Oppdragsgiver

Kronbar er oppdragsgiver og kunde i prosjektet. Kronbar er en bar som holder til på campus ved HVL på Kronstad. Den er drevet utelukkende av frivillige fra HVL, og er en populær plass for studentene til sosiale arrangement og konserter.

1.3 Problemstilling

Denne applikasjonen skal utvikles for intern styring av frivillige på Kronbar, og skal fungere både på iOS og Android. Hovedbrukstilfellet innebærer å kunne logge inn på applikasjonen med brukernavn og passord. Deretter sette seg opp, endre eller sjekke vaktlister, samt en kalender med informasjon til de frivillige som jobber på Kronbar. En administrator skal også kunne logge inn og få en oversikt over alle vakter som er satt opp av de frivillige.

1.4 Mål

Målet er å utvikle en mobilapplikasjon til internt bruk med en rekke funksjoner som er lagt i en prioritetsliste fra kunden. Den første og viktigste funksjonen innebærer at en administrator skal kunne legge inn vakter med ulike bruksbehov som ansvarshavende, barpersonell, teknisk etc. De frivillige skal så kunne sette seg opp på disse vaktene. Hovedmålet er å kunne implementere alle de viktigste spesifikasjonene i oppgaven. Dette inkluderer et vaktsystem som de frivillige kan melde seg opp på vakter på. I tillegg en oppslagstavle som administratoren kan legge ut innlegg på, et internforum som frivillige kan legge ut innlegg og diskutere på samt andre mindre funksjoner som er nyttig for bruker.

1.5 Kontekst

Kronbar har hatt et ønske om å fornye sitt digitale system for de frivillige som jobber der ved å utvikle en mobilapplikasjon med samme funksjoner som eksisterende web løsning. Her vil de gjøre det enklere å følge med på oppslagstavler og å bruke vaktsystemet. Vaktsystemet er det som sees på som hoveddelen av mobilapplikasjonen, muligheten for å enkelt melde seg opp på vakter som er tilgjengelige, antall poeng en får for vakten og ha kontroll på hvor en ligger i forhold til vaktkravet om antall vakter og totalsum av poeng er den viktigste delen og få med i mobilapplikasjonen. Vaktkravet er et krav som Kronbar stiller til alle som jobber som frivillige ved Kronbar, dette går på antall vakter og ulike jobber som en utfører for Kronbar, ulike vakter og jobber har forskjellig vekt i poeng som en til slutt summerer opp i en totalsum av det arbeidet du har gjennomført. Vaktkravet er enn satt sum som alle som jobber frivillig skal oppnå.
For dette prosjektet så vil gruppen vår jobbe som et selvstendig team med Kronbar som kunde.

1.6 Avgrensninger

Det er naturlig at det vil være en del begrensninger på et slikt prosjekt med de tidsrammene som er satt for å fullføre oppgaven innen semesteret er over. Siden oppgaven er såpass omfattende og kunden har sagt at det ikke er forventet at vi fullfører hele kravlisten (ref. 10.3), ble det satt noen avgrensninger i forhold til oppgaven. Punktene som er med i den initielle planen er nærmere beskrevet i punkt 2.1.3. De resterende punktene vil bli sett på etter de initielle kravene er fullført.

1.6.1 Teknisk kunnskap

Da kunden ønsker å bruke et rammeverk som ikke er gjennomgått i løpet av studiet så må gruppen tilegne seg tilstrekkelig kunnskap på egenhånd gjennom bruk av ressurser på internett.
Kunnskap fra studiet har gitt grunnleggende forståelse for rammeverk og mulighet for at det er oppnåelig å tilegne seg kunnskapen slik at en kan gjennomføre prosjektet med ønsket teknologi.

1.7 Ressurser

For å gjennomføre dette prosjektet er vi avhengig av en del ressurser. Siden utviklingen av applikasjonen vil foregå på forskjellige maskiner er det viktig å enkelt kunne dele kode og ressurser. Koden må kunne enkelt deles mellom de interne i prosjektet slik at alle kan være oppdatert på hva de andre jobber med. På grunn av covid19-situasjonen som vi er i så vil det i tillegg være ekstra behov for kommunikasjonsplattformer da vi ikke kan møtes for å jobbe sammen eller ha fysiske møter med veileder og leder for Kronbar. Til dette trenger vi et videomøteprogram og et sted der planlegging og kommunikasjon kan foregå.
Mot slutten av prosjektet vil det også bli behov for personer med mobiltelefoner for testing av applikasjonen. Kronbar har selv sagt at de kan stille med dette. Videre vil veileder og leder for kronbar fungere som gode ressurser for oss i nesten alle ledd gjennom utviklingen ved å svare på ting vi lurer på.

1.8 Oppbygging av rapporten

Kap 1. Innledning, gir en oversikt over prosjektet med hva problemstillingen er, mål og motivasjon for prosjektet, hvem er kunden for prosjektet, hvorfor de vil ha utviklet en applikasjon og til slutt informasjon om begrensninger og ressurser brukt i prosjektet for å nå det sluttresultatet som kommer til å bli levert.

Kap 2. Prosjektbeskrivelse, går dypere inn på informasjon om prosjektet og løsningen som utvikles. Det inneholder en innføring i hva dagens løsning er og hva de ønsker at en ny løsning skal inneholde og til slutt metoder og verktøy som blir benyttet.

Kap 3. Design av prosjektet, er en beskrivelse på tilnærminger av prosjektdesign og utførelse av prosjektet med prosjektmetodikk som inneholder risikovurdering, utviklingsmetodikk, prosjektplan og evalueringsplan.

Kap 4. Detaljert design, går dypere inn på designet for løsningen vår.

Kap 5. Evaluering, gir oversikt over evalueringer benyttet gjennom prosjektet og resultater av disse.

Kap 6. Diskusjon, blir det diskutert rundt konsekvenser ved valg av løsninger, verktøy, utviklingsmetode og hvordan det påvirker resultatet.

Kap 7. Konklusjon og videre arbeid, beskriver konklusjoner for prosjektet og videre arbeid etter endt prosjekt.

Kap 8. Referanser, lister opp alle referanser og litteratur som er brukt for arbeidet med mobilapplikasjonen og rapporten.

Kap 9. Referanser, hvor det gjennomgås diagrammer for risiko og progresjon ved bruk av Gantt.

Kap 10. Vedlegg, diagrammer og lister som er referert til i rapporten.

2 Prosjektbeskrivelse

2.1 Praktisk bakgrunn

Kronbar arrangerer hyppig arrangementer hvor alle over 18 år er velkommen, student eller ikke. Eksempler på arrangementer er konserter, jam og quiz, samt markering av spesielle dager som blant annet Halloween og Valentines Day. Kronbar sin stab består av sirka 20 frivillige studenter fra Høgskolen på Vestlandet. Lokalet er plassert i den nordlige delen av skolen, og er et fredet kulturminne fra NSBs eldre dager da det tidligere har vært et maskineri for damplokomotiv. Per dags dato setter de frivillige seg selv opp på vakter som passer via en internettside. Dette ønsker Kronbar å erstatte med en mobilapplikasjon.

2.1.1 Prosjekteier

Kronbar vil få alle rettighetene til det ferdige produktet. Kronbar har selv utledet retningslinjene og kravspesifikasjonene for produktet. Dersom løsningen viker fra kravspesifikasjonene må dette avklares med oppdragsgiver og leder for Kronbar.

2.1.2 Dagens Løsning

Kronbar har i dag en løsning i form av en internettside som deler mange av funksjonene til applikasjonen som skal lages i dette prosjektet. Eksempler er innlogging, registrering av vakter, kalender og oppslagstavle. Likevel er det ønskelig at det nye produktet erstatter den gamle løsningen og ikke bare er en oppgradering. Slik databasen er satt opp nå er det en tidligere frivillig hos Kronbar som hoster serveren via sin private datamaskin. Dette ønsker Kronbar også å endre på ved å enten flytte server hosting til en av skolens egne servere eller ved hjelp av googles firebase hosting. Alt av lagret data vil da flyttes samtidig.

2.1.3 Initielle krav

For krav så har Kronbar lagt til en liste i kravlisten som ligger som vedlegg under 10.3. Hvert krav har ett mål og én beskrivelse som forklarer problemet. Det er også lagt ved prioritet til hvert punkt etter hva Kronbar synes er viktigst å fullføre. De punktene som først og fremst prosjektet har fokus på er:

1. Flere plattformer: Mobilapplikasjonen skal fungere på både iOS og Android enheter.
    a. Flere operativsystemer er et pluss.
2. Vaktsystem: Et fungerende internsystem der frivillige kan sette seg opp på vakter og se sine vakter.
    a. Administratorer skal kunne opprette forskjellige vakter med forskjellige vaktkrav.
3. Oppslagstavle: Applikasjonen skal ha en oppslagstavle der administrator kan legge ut info.
4. Internforum: Frivillige skal kunne legge ut ønsker om vaktbytter.
    a. Dersom en ansatt ikke har mulighet til å ta en vakt kan vedkommende legge denne ut som åpen for andre til å ta.

5. Gamification: Frivillige skal kunne se hvordan de ligger an i forhold til vaktkravene som er satt.

[For Kronbars beskrivelse av kravene, se punkt 10.3]

2.1.4 Initiell løsnings-idé

En løsning der frivillige kan benytte seg av mobilapplikasjonen til å enkelt og raskt utføre oppgaver på mobil er ideen. Mobilapplikasjonen skal ha et enkelt og intuitivt brukergrensesnitt. Løsningen vil begynne med en innlogging før man får opp alle funksjonene som applikasjonen har å tilby. Etter innlogging kan det enkelt navigeres mellom funksjoner som kalender, profil, oppslagstavle osv. Løsningen skal fungere på både iOS og Android. Det er også ønsket at koden er skrevet i React-Native.

3 Design av prosjektet

3.1 Forslag til løsning

Ved utvikling av mobilapplikasjoner finnes det en rekke mulige løsninger. Alt fra programmeringsspråk til utviklingsmiljø. Oppgaven til Kronbar går ut på å lage en applikasjon til iOS og Android. Dermed kan valg av løsning basere seg på hvordan dette skal gjøres.

3.1.1 Alternativ løsning 1 – Kryssplattform

Den første løsningen som ble vurdert er en kryssplattform applikasjon. Det betyr at det er mulig å lage én felles native applikasjon på både Android og iOS. Det gjør at man bare trenger å skrive i ett språk og dermed ha én felles app. Fordelen med dette er at man kan spare mye tid og mye unødvendig kode på å ha et rammeverk som fungerer på begge plattformer. Problemene med denne løsningen kan forekomme ved at ingen av medlemmene i prosjektet har erfaringen med kryssplattformspråk som enten React-Native eller Flutter. Det betyr at det kan gå mye tid til både læring og implementering av dette rammeverket.

3.1.2 Alternativ løsning 2 – Native IOS og Android

Det andre løsningsforslaget baserer seg på å bygge én native mobilapplikasjon for både iOS og Android. Ettersom første prioriteten til kunden var at appen skal fungere på både iOS og Android mobiler så var det uaktuelt å kun prioritere én av plattformene. Dette betyr at appen må bli utviklet i Swift på iOS og Java på Android. En slik løsning vil føre til at applikasjonen mer eller mindre blir utviklet to ganger, og arbeidet blir dermed doblet.

Fordelene med en slik løsning vil være at det enkelt kan gjøres plattform-spesifikke endringer. I tillegg vil også selve brukergrensesnittet føles mer naturlig da det er laget i et native språk, og applikasjonen blir utviklet med dette i bakhodet. Ettersom koden kjøres direkte på operativsystemet vil det også føre til en kortere overførelsestid fra kode til maskinkode. Som igjen vil gjøre applikasjonen raskere. På den andre siden så kan denne løsningen føre til uforutsette plattform-spesifikke problemer, og kan føre til at utviklingsprosessen på én av plattformene vil bli mer tidskrevende enn den andre.

3.1.3 Tilpasse den eksisterende webapplikasjonen til mobil

En annen løsning vil være å kunne tilpasse den eksisterende webløsningen til en mobil webapplikasjon. En av fordelene med dette vil være at det vil lett kunne synkroniseres med dagens webløsning. Det vil gjøre den lett tilgjengelig på både Android og iOS ved operativsystemets egen webtjener. Ulempene kommer av at ytelsen vil være relativt treg med tanke på at det er en web app som er i dag ment å kjøre på en laptop. I tillegg kan den føles gammeldags for de som allerede har brukt dette systemet i mange år på en datamaskin.

3.1.4 Lage en hybrid app til mobil

Det var også muligheter for å lage en helt ny webapplikasjon til mobil i form av en hybrid webapplikasjon. En hybrid webapplikasjon er en nettbasert applikasjon som kan kjøres på iOS og Android med tilgang til funksjonene på mobilen. En av fordelene med en slik løsning vil være at man unngår å bruke lang tid på å lære et nytt språk, og selve utviklingen vil være enklere og mer effektiv. Ulempen med denne løsningen er at den vil være relativt treg i forhold til en native-applikasjon. Den vil heller ikke ha like god tilgang til hardware og visse funksjoner som for eksempel push-varsel.

3.1.5 Diskusjon av alternativene

De to første alternative løsningene over har en rekke likheter. Begge løsningene med hovedfokus på kravspesifikasjonen om å utvikle en mobilapplikasjon på både iOS og Android. Disse vil helt klart gi den beste ytelsen og den beste opplevelsen for brukeren. Ulempen er at disse vil ta relativt lengre tid å utvikle, og dermed kan funksjoner være nødvendig å utelukke for å levere produktet i tide.

De to andre løsningene innebærer å lage en mobil webapplikasjon. De største ulempene med dette er at lite eller ingen offline muligheter, begrenset funksjoner og dårligere ytelse enn native apps vil ha.

Ved valg av løsning så vil egen erfaring, kunden sine anbefalinger og videre utvikling bli vurdert.

3.2 Valgt løsning

Den valgte løsningen for prosjektet er bruk av kryssplattform-applikasjon. I og med at hovedmålet med applikasjonen er funksjonell bruk av appen gir en kryssplattformløsning alle byggeklosser som trengs for å implementere dette. Kryssplattform er også kunden sine anbefalinger og ønske. Dette er en viktig del av valget ettersom applikasjonen skal videreutvikles av kunden.

Det finnes også ulemper med denne løsningen som vi kommer tilbake til under punkt 6.1. Den største er problemet som oppstår er ved fleroppgavekjøring. Dette er ikke implementert på samme måte som native apps. Det finnes også mulige løsningen for dette.

3.3 Valg av verktøy

3.3.1 Kryssplattformspråk

Når det gjelder valg av språk sto det mellom å bruke Facebook sitt React-Native rammeverk eller bruk av Google sitt Flutter rammeverk. React-Native bruker en egen versjon av Javascript og

deretter oppretter en tilsvarende iOS løsning og Android løsning av dette. React bruker Javascript med en XLM-lignende syntaks som heter JSX. Javascript er et språk som har tilgang til en stor rekke med eksterne bibliotek som finnes også for native iOS og Android utvikling. Dette gjør det enkelt å hente tredjeparts bibliotek som kan legges til og brukes i appen.

Flutter er Google's rammeverk for å bygge kryssplattform native apps og webapplikasjon. I likhet med React-native så kan flutter bygge en native-applikasjon for både iOS og Android, men skiller seg ut med at den har flere innebygde verktøy som gjør det enklere å bygge et naturlig brukergrensesnitt. Flutter bruker Dart som sitt programmeringsspråk og kan også gjøre enkle backend løsninger med Dart. Både Dart og Flutter er et relativt nyere programvare. Allerede 6.Mai 2020 kom Flutter med en stor oppdatering som endret mye av hvordan en bruker dette rammeverket.

En de viktigste grunnene til valget vil være egne erfaringer og pålitelighet med rammeverket. Flutter er et nokså nytt rammeverk som fortsatt er under utvikling. React-Native har vært til stede i mange år og brukes av svært mange store bedrifter. Det bygges også på Javascript som er et mye brukt språk og er godt testet. Det er også et språk som medlemmene i gruppen har erfaringer med fra før. På den andre siden har ingen erfaring med Flutter og Dart.

En av kravspesifikasjonene til kunden er at den kan lett sammenslåes med en annen applikasjon som er under utvikling av en annen prosjektgruppe. Disse har valgt React-Native som sitt rammeverk. Fordelen ved at begge bruker React-Native som rammeverk vil være at denne sammenslåingen bli betraktelig lettere. Kundens anbefalinger er også å bruke React-Native til å utvikle appen.

Ut ifra disse grunnene så var valg av kryssplattformspråk et av de enklere valgene. Med tanke på tidligere erfaringer, pålitelighet og videre utvikling av appen så er valget React-Native.

3.3.2 Tjener

Ved valg av tjener sto det mellom å bruke en egen tjener som bruker rammeverk Spring, eller Google Cloud og Firebase som kommuniserer med klientsiden. Begge disse er mye brukt av bedrifter og organisasjoner. Grunnen til at det sto mellom at disse var at Firebase og en Java-basert tjener er det eneste gruppen har erfaring med fra før. Det var også viktig å ikke bruke for mye tid i begynnelsen av prosjektet på å sette seg inn i et nytt rammeverk som trolig vil bli byttet ut av Kronbar senere.

Spring er et rammeverk som erstatter Enterprise JavaBean modellen med enklere og raskere alternativer til javakode. Spring er verdens mest brukte Java rammeverk. En av grunnene til å velge Spring når det gjelder en Java-basert løsning er at det er raskere og enklere å bruke enn en rekke andre rammeverk. Spring er også et godt brukt rammeverk innenfor tjenerløsninger. For prosjektet betyr dette at selve back-enden av appen kan skrives i Java, som er et språk som er godt kjent i gruppen. Det er derfor det enkleste og beste alternativet når det gjelder back-end. Spring er også anbefalt fra kunden og skal brukes i videreutvikling av appen. Spring gjør oppkoblingen og kommunikasjonen med server og database mer simpelt i forhold til tradisjonell Java.

Firebase er en tjener fra Google som krever lite programmering for å sette opp[9]. Lagring av database og kommunikasjon gjøres nesten helt automatisk. Firebase er dermed utrolig

tidseffektiv i forhold til en mer tradisjonell løsning. Dette kan også være en ulempe med tanke på at en ikke vet så mye om akkurat hvordan tjenersiden er satt opp. Det er kun applikasjonen som kommuniserer med Google sin sky. Dette kan bli et problem hvis det skulle oppstå eventuelle feil med kommunikasjonen mellom databasen og appen.

I begynnelsen var planen å bruke Spring som tjener til applikasjon. Dette ble revurdert ettersom ingen av medlemmene har tidligere erfaring med dette rammeverket og ville ikke bruke for mye tid på å sette seg inn i dette. I tillegg ble det diskutert i møter med kunden der kunden lot oss bestemme valget selv ettersom dette er noe som er lett å bytte ut i ettertid for dem. Applikasjon bruker derfor Firebase som database og til autentisering av bruker.

3.3.3 Kommunikasjon

Slack brukes for alt fra kommunikasjon, deling av filer og holde videomøter. Slack er et kommunikasjonsverktøy som gjør det enkelt å samarbeide. Med flere funksjoner som ulike kanaler for forskjellige temaer og inkludert innebygd kodedeling ved GitLab og videomøteprogrammet Zoom. Zoom er et verktøy som brukes til å holde videomøter med kunden, veileder og prosjektmedlemmene internt.

3.3.4 Utviklingsmiljø

Ved valg av IDE har vi valgt Visual Studio Code ved bruk av React-Native. VS code er et raskt og enkelt utviklingsmiljø med innebygd terminal som gjør det lett å kjøre både iOS- og Android-emulatorene side om side. I tillegg brukes Xcode og Android studio til å gjøre plattform-spesifikke endringer og til å kjøre emulatorene.

For kodedeling har vi valgt GitLab som er et verktøy som bruker git. På samme måte som GitHub så kan koden lastes opp på en egen repository. GitLab har også mange ulike funksjoner og fungerer innebygd i Slack som brukes ved kommunikasjon.

3.4 Prosjektmetodikk

I den innledende fasen av prosjektet ble vi introdusert for intern veileder Bjarte Kileng og oppdragsgiver hos Kronbar, Magnus Marthinsen. Sammen med et utvalg andre ansatte hos Kronbar har Marthinsen produsert en liste med kravspesifikasjoner og ønsket funksjonalitet for den nye mobilapplikasjonen. Gjennom et oppstartsmøte med både veileder og oppdragsgiver ble det gjennomgått en rekke oppklaringer til prosjektet. Deriblant arbeidsplass, hvilket utstyr som var til disposisjon, eierforhold til resultatet og hvor ofte det skulle holdes møter.

Nåværende løsning og funksjonalitet ble introdusert. Marthinsen demonstrerte systemet, gav et dypere innblikk i hvordan dette var bygget opp og forklarte hvorfor den nåværende løsningen ikke er optimal for ansatte hos Kronbar.

3.4.1 Utviklingsmetodikk

I utviklingsperioden brukes den smidige utviklingsmetodikken Scrum i samarbeid med Extreme Programming (XP)[5]. Scrum baserer seg på sprints, korte tidsintervaller, som varer i 2-3 uker[4]. På grunn av den korte tiden hver sprint har er kommunikasjon og samarbeid med kunden utrolig viktig for å bruke tiden effektivt. En smidig metode betyr også at den er fleksibel med tanke på endringer, tidlige leveranser og rask utvikling. Dermed passer denne metoden godt for dette prosjektet.

3.4.2 Prosjektplan

Utviklingsmetodikken baserer seg på en sprint eller en iterasjon med mål og kravspesifikasjoner for hver iterasjon. Etter første iterasjon så er målet å ha en tidlig versjon av appen som kan vises til kunden og få eventuelle tilbakemeldinger. Det er dermed viktig å ha minimum én kravspesifikasjon å kunne vise frem etter hver iterasjon.

Etter den første iterasjonen kommer den største fasen av prosjektet. Her vil de fleste funksjonene og kravspesifikasjonene bli implementert og grundig testet. De viktigste delmålene vil bli fokusert på først og samtidig testet i sann tid. I avslutningsfasen vil arbeidet gå ut på siste testing, designendringer og eventuelle feilrettinger. Dette er nærmere beskrevet i det initielle Gantt diagrammet i 10.2 som en del av prosjektplanen.
(Vedlegg 10.2).

3.4.3 Rollefordeling

Det har blitt praktisert en løs rollefordeling i prosjektet. Siden gruppen kun består av 3 personer har det ikke blitt sett på som nødvendig med en gruppeleder. Helt fra starten av prosjektet har hvert gruppemedlem rapportert fremdrift og status til de andre etter hvert som nytt arbeid blitt gjennomført. Så mottatt tilbakemeldinger og eventuelle innspill på dette. Det har blitt praktisert å jobbe både sammen og på egenhånd. En slik ordning krever god kommunikasjon mellom gruppemedlemmene.

3.4.4 Risikovurdering

I løpet av dette prosjektet vil det bli brukt en risikoliste (Vedlegg 10.1) som vurderer ulike risiko og hvilke konsekvenser som ligger til grunn. Dermed blir det vurdert en skala på 1-5 for å vurdere hvor sannsynlig en risiko kan forekomme og for å beskrive konsekvensene for risikoen dersom den skulle forekomme. Disse to tallene multipliseres og danner en risikofaktor som viser hvor problematisk hver enkel risiko er.

En risikofaktor på under 10 anses som uproblematisk, mens en skala på 10-20 er middels problematisk og over 20 er problematisk. Til slutt blir det beskrevet hvilke tiltak som kan gjøres for å unngå at risikoen inntreffer, når risikoen er aktuell og interessentene for hver risiko.

3.5 Evalueringsplan

Planen for evaluering av mobilapplikasjonen og arbeidet som er gjort er delt opp i flere stadier med testing på ulike nivåer av applikasjonen og funksjonene i den. De ulike formene for testing er enhetstesting, iterasjon testing, brukertesting og akseptansetesting.

Enhetstesting er testing av isolerte deler av koden for å kontrollere ved gitt informasjon eller bruk gir ønsket resultat. Planen er å bruke Jest, som er Facebooks test bibliotek og er inkludert i React-Native, til enhetstestingen. Jest er et utrolig raskt og enkelt testbibliotek med mange ulike verktøy[10]. Planen er at enhetstesting vil bli gjennomført under hele prosjektperioden for å kunne sjekke om de metodene og komponentene som blir implementert fungerer som de skal.

Integrasjonstesting tester hvordan flere funksjoner og komponenter fungerer sammen. En av de beste funksjonene i Jest er bruk av såkalte mocks. Disse tester kall på ulike funksjoner uten å implementere selve funksjonen. Dette er en stor fordel for å kunne teste om funksjonen blir kalt på og om den endrer state uten å kalle på den faktiske funksjonen. Planen er å bruke dette til å teste de ulike kravspesifikasjonene i appen hver for seg. Snapshot testing, som tester om de ulike komponentene henger sammen og ikke endrer seg uventet, skal bli brukt til å teste at brukergrensesnittet ser ut som det skal.

Brukertesting vil ifølge planen bli gjennomført mot slutten av prosjektet for å få konstruktiv kritikk og tilbakemeldinger på funksjoner og brukervennligheten til applikasjonen. Dette er for å se hva som kan endres eller som er godkjent før prosjektet skal avsluttes. Planen er å kunne sende en versjon som kan testes til kunden noen uker før prosjektperioden er ferdig. Dermed vil det være tid til eventuelle endringer og feilrettinger som kunden har å komme med. I tillegg vil en rekke spørsmål bli presentert for de ulike kravspesifikasjonene. Svarene og tilbakemeldingene vil deretter bli gjennomgått og evaluert i et møte med kunden. Så bli det vurdert om det trengs å gjøre endringer og hvor viktige disse endringene er.

Akseptansetesting brukes til å teste om kravspesifikasjonene og brukstilfellet fra kunden er møtt. De viktigste faktorene for å kunne teste dette vil være å kunne se om hvilke kravspesifikasjoner fra kunden er gjennomført og hvilken prioritet disse har. Planen er også å ha jevnlige møter med kunden for å vise fremgangen og avklare eventuelle spørsmål og endringer med prosjektet. En slik test består av tilbakemeldinger fra kunden, om kravene i kravlisten er bestått og om produktet fungerer som det skal.

4 Detaljert Design

4.1 Innlogging

Siden applikasjonen trenger informasjon om brukeren er det nødvendig med en innlogging for identifisering. Dette gjør at det kan lagres brukerdata i skyen og gi personlige funksjoner basert på om bruker er administrator, ansatt eller tilfeldig bruker. Dette uavhengig av plattform.

Valgt løsning når det kommer til innlogging er "Firebase Authentication". Firebase Authentication tilbyr backend tjenester, SDK-er som er lett å bruke og klargjorte UI biblioteker til å autentisere brukere i applikasjonen[9]. Det støtter autentisering via epost og passord, telefonnummer, populære autentiseringstilbydere som Google, Facebook Twitter, og flere.

Figur 2 Skjermbilde fra youtube film

I første omgang blir det kun brukt autentisering med epost og passord. Firebase Authentication SDK tilbyr metoder for å lage og endre brukere som bruker epost og passord til å logge inn. Firebase Authentication håndterer også utsending av e-poster for glemt passord.

Som vist på figur 3 blir signInWithEmailAndPassword-metoden under auth() brukt. Dette er JavaScript kode som kjøres på klienten for å logge inn en bruker. Denne tar to parametere, som er email og passord, og logger inn brukeren. Det å lage bruker er gjort på samme måte med en annen metode som heter createUserWithEmailAndPassword (figur 4). Her blir det også lagt til en attributt på brukeren som er displayName. Denne attributten vil bli bundet til brukeren og kan brukes ved en senere anledning.

```
const login = ({ navigation }) => {

    const[email,  setEmail] = useState('');
    const[password,  setPassword] = useState('');
    const[errorMessage,  setError] = useState('');

     const handleLogin = () => {
        firebase
            .auth()
            .signInWithEmailAndPassword(email.email, password.password)
            .catch(error => setError({ errorMessage: error.message }));
    }
```

Figur 3 Metode for innlogging i Kronbar applikasjonen

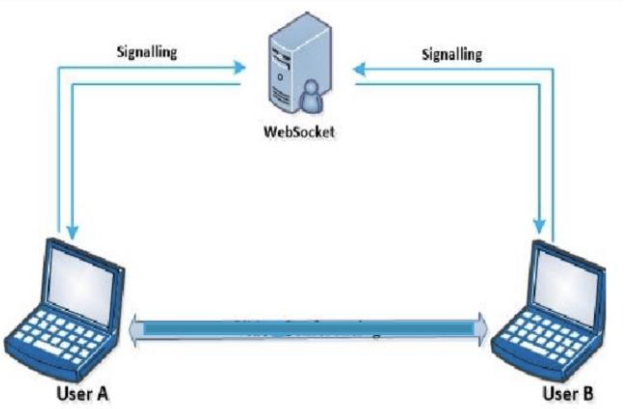
```
handleSignUp = () => {
    firebase
    .auth()
    .createUserWithEmailAndPassword(this.state.email,  this.state.password)
    .then(userCredidentials => {
        return  userCredidentials.user.updateProfile({
            displayName: this.state.name
        })
    })
    .catch(error => this.setState({errorMessage: error.message}));
}
```

Figur 4 Opprette bruker i Kronbar applikasjonen

4.2 Databaser

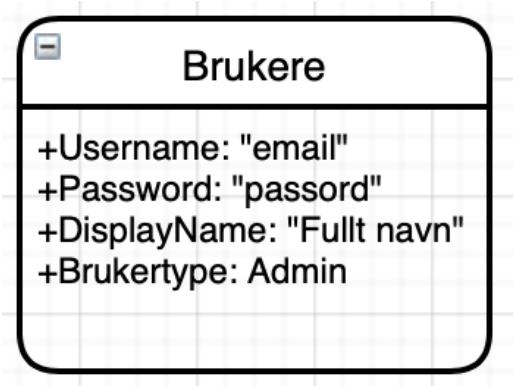
Siden Firebase Authentication blir brukt for innlogging er det ryddig at Firebase sin databaseløsning også blir brukt. Firebase Realtime Database er en NoSQL Cloud Database som er utgitt under Google's Firebase mobile plattformen som et av deres sentrale produkter[13]. Det er bygget på Google's infrastruktur. Så utviklere uten mye backend kunnskap kan bygge applikasjoner raskt uten å tenke for mye på skaleringsmuligheter. Dermed kan det legges mer fokus på å bygge best mulig applikasjon.

Firebase Realtime Database (RTDB) lagrer data som JSON tre. Det tilbyr mange SDKer fra iOS, Android og Web. Realtime Database bruker WebSocket teknologi for synkronisering, så alle klienter som er koblet til kan motta oppdatert informasjon når dataen i serveren forandres og motsatt. Det vil si at når klienten oppdaterer data så vil serveren oppdateres i løpet av millisekunder (toveis kommunikasjon). I tillegg tilbyr Firebase Realtime Database støtte for offline funksjonalitet, slik at klienten fortsatt kan se data og endre data uten internettilkobling, for så å synkronisere automatisk når tilkoblingen er online igjen.



Figur 5 Synkronisering via WebSocket

Brukere (figur 6) i prosjektets database blir lagret med brukernavn og passord pluss et par andre egenskaper som er “DisplayName” og “Brukertype”. DisplayName blir brukt kun for det visuelle slik at vi kan bruke fullt navn når vi refererer til en person. Brukertype blir brukt for å gi spesielle funksjoner til forskjellige brukere. For eksempel å gjøre det mulig for administrator å legge ut vakter og poste på oppslagstavlen uten at vanlige brukere kan gjøre dette.

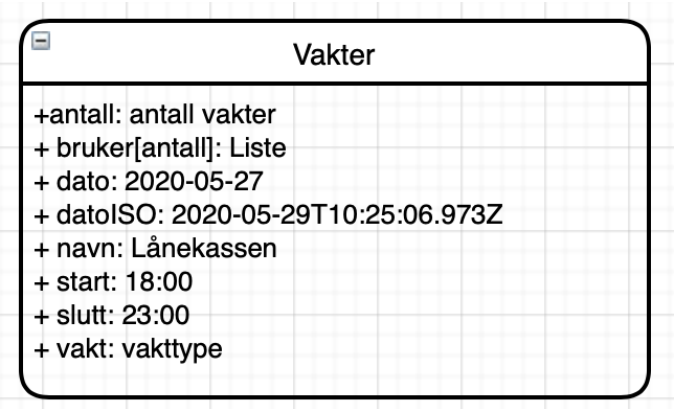


Figur 6 Brukere i databasen

Figur 7 Oppslag i databasen

Oppslag (Posts) er lagret med den teksten som oppslaget skal inneholde og tiden den ble lagt til (figur 7). Siden kravet for oppgaven var at kun administrator skal kunne legge ut oppslag er det ikke nødvendig å linke post til bruker som har skrevet oppslaget, den kommer i stedet fra “Kronbar” og med tiden den ble postet.

Vaktene i databasen er lagret med en del info. Blant annet start, slutt navn og bruker. Det som er verdt å legge merke til her er at ‘bruker’ er en liste på størrelse med antall vakter. Denne inneholder brukere(eposter) til alle som er meldt opp på den aktuelle vakten og ‘ledig’ der det ikke er meldt opp noen bruker. Dato er dato for vakten og datoISO er dato og UTC-klokkeslett på ISO-format, altså ÅÅÅÅ-MM-DD ‘T’ TT:MM:SS (figur 8 - datoISO).

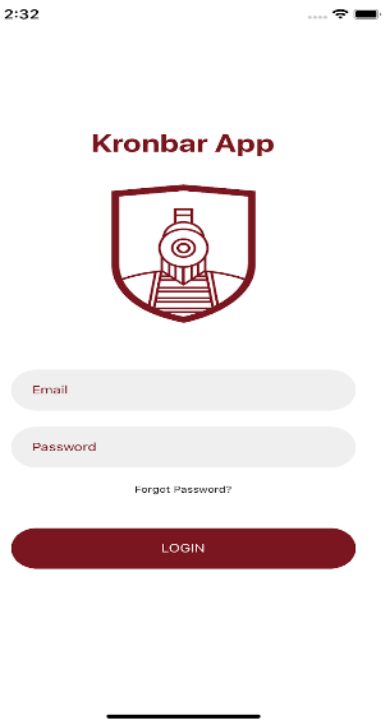


Figur 8 Vakter i databasen

4.3 Applikasjon

4.3.1 Innlogging

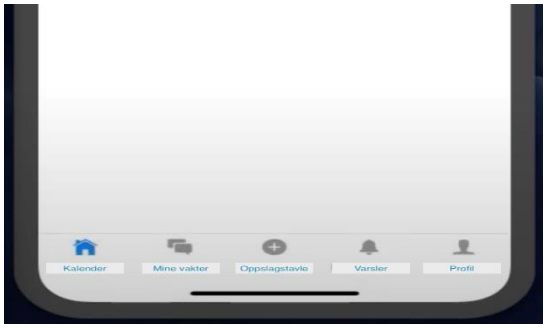
Innloggingssiden (figur 9) er en enkel side som kun inneholder mulighet for å logge inn og lage bruker. Det skal ikke være mulig å utføre noen av applikasjonens funksjoner uten å logge inn, derfor trengs det heller ikke noe mer på denne siden.



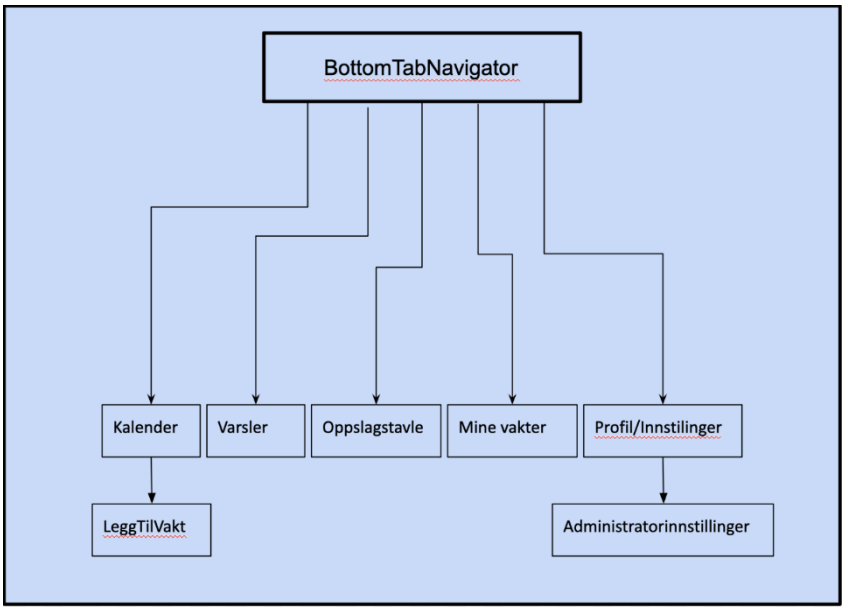
Figur 9 Innloggingsside i applikasjon

4.3.2 Hjem

Etter innlogging er applikasjonen designet som følger. En hjemmeside har en Tab Navigator på bunnen som brukes til å enkelt navigere mellom alle sidene som applikasjonen har. Denne inneholder "Mine vakter", "Oppslagstavle", "Kalender", "Varsler" og "Profil/Innstillinger". Navigasjonsveiene er illustrert i figur 11.



Figur 10 Fanenavigator i applikasjon

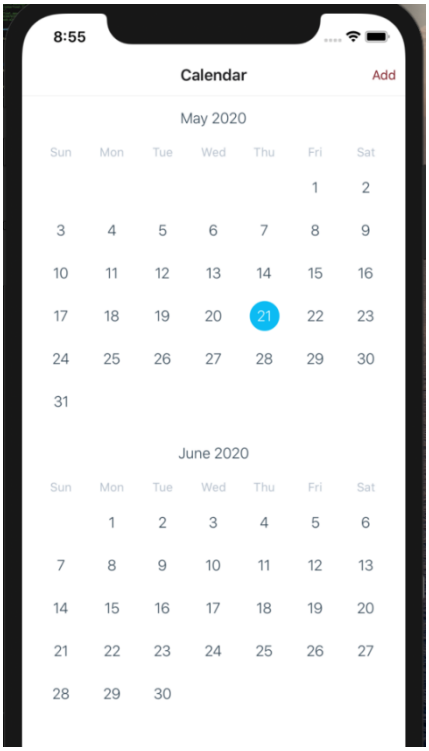


Figur 11 Illustrasjon av navigasjon

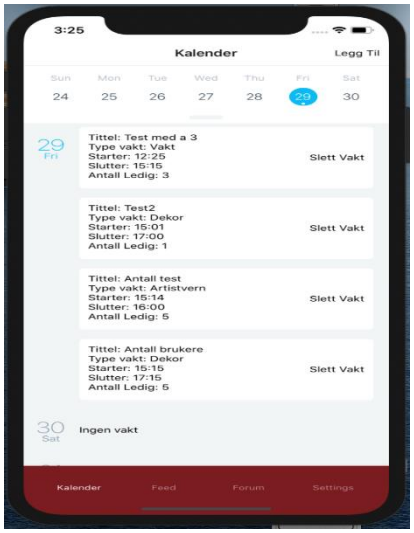
4.3.3 Kalender

Kalendersiden (figur 12) inneholder en kalender der dagene vedkommende er satt opp på vakter er markert. Her er det mulighet for å sette seg opp på nye vakter og se alle vakter som er tilgjengelige på valgt dato.

I applikasjonen blir en vakt seende ut som på figur 13. Her vil man kun se 'Tittel', 'Type Vakt', 'Starter', 'Slutter' og 'Antall ledig'. Dette ble oppgitt i kravspesifikasjonene (ref. 2.1.3).



Figur 12 Kalender i applikasjon



Figur 13 Vakter i kalender

4.3.4 Mine vakter

Her vil det bli en listevisning med alle vakter man er satt opp på. I tillegg vil det være mulighet for å se en del info om vaktprogresjon og liknende. Eksempler på info på denne siden er:

- Antall vakter totalt.
- Vaktkrav og progresjon mot det.
- Arbeidstimer.
- Antall semester som frivillig.

4.3.5 Oppslagstavle

Denne siden vil inneholde en oppslagstavle der administratorer kan legge ut nyttig informasjon som resten av brukere kan se. Det vil også sendes ut varsel hver gang det kommer en ny post for at alle skal få med seg nyttig informasjon.

Figur 14 Oppslagstavle i applikasjon

4.3.6 Varsler

Her vil alle varslene som har med applikasjonen bli loggført i en listevisning. Eksempler på dette kan være "X har bedt om at du tar vakt Mandag kveld" eller "Nytt oppslag fra X på oppslagstavlen".

4.3.7 Profil Innstillinger

På profilsiden er det mulighet for å gjøre alt som har med profilen å gjøre. For eksempel oppdatere profilbilde, endre epost, bytte passord. Det er også herfra man logger ut.

5 EVALUERING

5.1 Evalueringsmetode

Gjennom hele prosjektet blir det brukt evalueringsmetoder for å forsikre at kvaliteten og resultatene på arbeidet er innenfor kravene som er stilt i oppgaven. I løpet av prosjektperioden brukes fire forskjellige metoder for evaluering.

5.1.1 Enhetstesting

Som nevnt tidligere i 3.5 brukes Jest til å teste de ulike komponentene og mindre funksjonene i appen ved hjelp av enhetstesting. Slike tester er utrolig raske og validerer kun enkle funksjoner. Det holder derfor ikke å bare bruke enhetstesting til å evaluere koden da dette ikke tester de ulike komponentene og funksjonene i sammen. Dermed har Jest blitt brukt som et verktøy, men har ikke hatt den høyeste prioritet når det kommer til evaluering. Det er også lagt vekt på at kunnskapen om enhetstesting for React-Native ikke er noe gruppen har fra før prosjektet og at kunden heller ikke har stilt noen krav om dette.

5.1.2 Integrasjonstesting

Mocks funksjoner med Jest brukes til å evaluere hvordan de ulike komponentene og funksjonene fungerer sammen. Jest har innebygde mock funksjoner som kan brukes i testing. Dette brukes til å sjekke om funksjonene blir kalt på som forventet. Et eksempel der dette er brukt er funksjonen getVakt, som henter alle vakter fra firebase databasen og legger til i en liste, som blir kalt på når selve kalender blir lastet inn. Denne testen er utrolig viktig for å hindre at getVakt, som er en dyr prosess, blir kalt på mer enn nødvendig. I tillegg er slike tester viktige for å sjekke at funksjonene får de riktige parameterne når de blir kalt på.

Snapshot testing er en annen form for integrasjonstesting der testen tar et bilde av hvordan skjermen ser ut og sammenligner det mot forventet resultat. På grunn av dårlig tid mot slutten og ettersom dette prosjektet skal videreutvikles og designet skal endres på så har ikke denne testingen blitt brukt som planlagt i evalueringen.

5.1.3 Brukertesting

Brukertesting var noe som både kunden og prosjektgruppen ville få til i begynnelsen av prosjektet. Planen var at dette skulle bli gjennomført i midten av prosjektperioden og dermed få god tid til eventuelle endringer og feilrettinger. Ettersom dette ikke var mulig å møtes fysisk og det oppsto uventede problemer underveis, som nevnt i 7.3, så var det dessverre ikke nok ønsket tid til å gjennomføre dette i løpet av prosjektet. Det sagt så er applikasjonen fortsatt under utvikling og det vil mest sannsynlig bli gjennomført en brukertesting av appen i fremtiden.

5.1.4 Akseptansetesting

Mot slutten av prosjektet ble akseptansetesting brukt for de ulike kravspesifikasjonene. Dette ble gjort ved å demonstrere de ulike funksjonene for kunden og vurdere tilbakemeldingene fra dem. Det var også en fordel at kunden, Kronbar, er en del av brukergruppen for den endelige appen. Dermed kunne tilbakemeldingene kobles direkte opp mot den antatte brukergruppen. De

funksjonene som ble ferdig implementert i appen ble godkjent av kunden og holdt kravene til oppgaveteksten.

5.2 Evalueringsresultat

Målet for bachelorprosjektet var å utvikle en kryssplattform-applikasjon som kunne forbedre måten Kronbar håndterer vaktplanlegging blant frivillige. Gjennom manuell testing, tekniske tester og akseptansetester har vi fått svar på om vi har nådd målene som er satt.

Tilbakemeldingene vi har fått gjennom møter med Kronbar er at de er fornøyde med det som er produsert til tross for vanskelige arbeidsforhold og problemer som nevnt under 7.3. Applikasjonen har nå den mest kritiske funksjonen som er selve kalenderen og mulighet for å legge til og slette vakter. Måten det ble fremstilt i applikasjonen var kunden spesielt fornøyd med. Koden til applikasjonen er ryddig og lett å videreutvikle, dette satt kunden stor pris på.

Funksjonene som hadde lavere prioritet ble ikke implementert, noe som var fullt forståelig fra kunden sin side. Det ble også sagt at det ikke var forventet at det skulle bli tid til dem. Vi skulle gjerne ha oppfylt alle kravspesifikasjonene dersom det var mer tid til disposisjon, men dette var dessverre ikke tilfellet.

Resultatet av evalueringen har gitt oss følgende konklusjoner:

- Produktet har løsninger som enkelt kan videreutvikles uten mye re-implementering.
- Produktet har lave koblinger og god struktur slik at endringer og videreutvikling kan gjennomføres.
- Produktet kan tas i bruk umiddelbart.
- Produktet har løsninger som har verdi for kunden.

Alt i alt så er Kronbar svært fornøyde med resultatet og gleder seg til å videreutvikle applikasjonen, og deretter slå den sammen med den andre applikasjonen for eksterne brukere av Kronbar.

6 Resultater

Resultater er tidligere beskrevet i 5.2. Der kom det frem informasjon om hvordan produktet har nytteverdi for kunden.

Produktet har lave koblinger og god struktur slik at endringer og videreutvikling kan gjennomføres.

Ved utvikling av applikasjonen har det blitt lagt fokus helt fra starten på at de forskjellige delene av koden skal påvirke hverandre minst mulig. Dette på grunn av at applikasjonen skal videreutvikles. I tillegg er planen deres å slå den sammen med et annet prosjekt. Det er derfor viktig at koden kan lett endres på uten å påvirke andre deler av applikasjonen. Det er også lagt vekt på at koden er oversiktlig og lett å forstå for en utvikler.

En av React Natives store fordeler kommer også godt med i dette prosjektet. Alle komponentene er JavaScript-klasser som er mer eller mindre fullstendig gjenbrukbare. I tillegg er det sørget for at komponentene som er implementert er generalisert til en grad som gjør at utskifting enkelt kan utføres ved behov.

Produktet har løsninger som har verdi for kunden

For at prosjektet kan anses som nyttig er det viktig at produktet har verdi for kunden. Som nevnt tidligere så er produktet verdifullt for kunden. En av grunnene til dette er at produktet er lagt opp slik at det kan lett videreutvikles og inneholder funksjoner som har verdi for den endelige brukergruppen.

De involverte i prosjektet er godt fornøyd med gjennomføringen av prosjektet

Selv om målene og kravene til oppgaven ikke ble gjennomført som ønsket så er det endelige resultatet noe som har stor verdi for kunden. Dermed er også de involverte tilfreds med den endelige evalueringen av resultatet.

7 DISKUSJON

7.1 Konsekvenser ved valgt løsning

Diskusjonene ved valg av løsning og om bruk av React-Native til å lage en kryssplattform-applikasjon har gått frem og tilbake. Dette inkluderer om utviklingen vil være mer tidseffektiv eller ikke. Ettersom React-Native skrives i ett språk skal det i teorien være mer effektivt å kode og gjøre endringer i koden. Dette er ikke alltid tilfelle da det kan oppstå uventede programvarefeil eller problemer som det ellers ikke ville gjort ved utvikling av native applikasjoner. Men uten slike problemer kan vi trygt si at en kryssplattform applikasjon har en mer effektiv utviklingstid, spesielt i mindre prosjekter.

En annen fordel med React-Native er at en kan utvikle funksjonaliteter som fungerer på begge plattformer uten å ha noe særlig kjennskap til disse. Et eksempel på dette er "DatePicker" i appen som bruker en native-funksjon til å velge tid og dato. Denne fungerer like bra på begge plattformer uten å trenge å implementere disse hver for seg på begge plattformene. Dermed er det mulig å kunne starte utviklingen på begge plattformer uten å ha noe kjennskap til hvert enkelt kodespråk og utviklingsmiljøet til hver av disse. Det er fortsatt viktig å ha noe kjennskap til utvikling på de ulike plattformene da både testing og installering kan variere fra hver enkel plattform.

Den største ulempen ved å bruke React-Native som rammeverk er ytelsen. Dette er siden React-Native kun kjører på tre tråder. Disse inneholder en UI-tråd, én Javascript tråd og én tråd som kan kommunisere mellom disse. Den siste tråden fungerer som en bro mellom de andre trådene ved å oversette React-Native til native kode. Problemer kan oppstå hvis store filer må bli sendt mellom trådene som igjen kan føre til ytelsesproblemer. UI tråden håndterer rendering av de forskjellige views-ene i brukergrensesnittet. Dette betyr at brukergrensesnittet er enkeltrådet og hvert enkelt view på skjermen må lastes inn individuelt. Treg navigering mellom skjermene kan derfor også forekomme i React-Native.

Disse konsekvensene anses å være relativt små med tanke på at appen per dags dato ikke bruker noen form for store funksjoner som må kjøres side om side, men kan bli et problem i fremtiden hvis applikasjon blir videreutviklet og oppdatert med store nye funksjoner. Et eksempel på en slik funksjon som kan skape dette problemer kan være en form for livequiz eller livekonsert der en videostrøm og en livechat går samtidig.

7.2 Andre konsekvenser

Konsekvensene av endringer i løpet av prosjektperioden er relativt små. En av grunnene til dette er at ved bruk av en smidig metode som scrum så er endringer enklere å forholde seg til. En av de største endringene ut ifra planen måtte gjøres under den tredje sprinten når vi innså at implementeringen av vaktsystem funksjonen tok lengre tid enn antatt. Da ble denne sprinten forlenget med én uke og fortsettelsen av denne implementasjonen ble lagt inn i neste sprint. Dette gjorde at de funksjonene som ble planlagt i siste sprint utgikk. Dette var mer en konsekvens av lite erfaring med React-Native og mobilutvikling fra før, og ikke en direkte konsekvens av valgt utviklingsmetode.

7.3 Refleksjon

Under planleggingen av prosjektet var det klart for alle parter hvor stort omfanget av prosjektet var. På grunn av lite erfaring med såpass store prosjekter og med en uvant arbeidsmetode som denne så var det ingen overraskelse når planen måtte endres på. Ettersom det er brukt en smidig metode og holdt god kommunikasjon under prosjektperioden så har skadene blitt redusert. Det som kunne blitt gjort annerledes kunne vært å starte enda tidligere med planleggingen av prosjektet. Dette kunne resultert i et bedre produkt og dermed vært muligheter for å gjennomføre brukertesting av produktet. Det kunne også blitt gjort mer læring og en grundigere sjekk på de verktøyene som ble brukt før utviklingen startet. Dette kunne hindret at problemer og bugs stoppet opp utviklingen underveis.

8 Konklusjon og videre arbeid

Basert på resultater og evalueringer så er det utviklet et større grunnlag for det endelige målet til Kronbar, med en felles mobilapplikasjon for frivillige interne og eksterne kunder. Produktet er en egen applikasjon de interne der ønsket rammeverk er brukt. Det er implementert funksjoner som er deler av funksjonskravet til kunden, og som enkelt videreføres i senere arbeid om en felles mobilapplikasjon.

Krav som er implementert i løsningen som er levert til kunde er:

- Funksjonalitet på flere plattformer.
- Personlig innlogging og registrering.
- Kalender med vaktsystem.
- Oppslagstavle for posting av oppslag fra administratorer.
- Oversiktlig kode med lav kobling som gjør det enkelt å implementere sammen med ekstern-applikasjon for videre utvikling.

Når en ser på målene satt for prosjektet ved start og kravene stilt fra kunde i forhold til det som blir levert så har ikke alt blitt oppnådd innenfor tidsfristen. Nåværende løsning på oppgaven er en mobilapplikasjon med fungerende løsninger som er listet over. Grunnlaget i nåværende løsning er enkel å videreutvikle. I tillegg etter samtaler med Kronbar angående løsningen, kravene og det vi har oppnådd så er de og veldig fornøyd med det arbeidet som er blitt gjort ut ifra oppgavene og kravene som var stilt ved start av prosjektet. Planen til kunden er å videreutvikle løsningen vår og integrere dette med et annet prosjekt. Prosjektet kan derfor sees på som en suksess selv om ikke alle målene ble nådd.

Hvordan dette skal gjennomføres er opp til Kronbar. De kan velge å ferdigstille mobilapplikasjonene som to selvstendige mobilapplikasjoner først. Deretter sette det sammen, eller ta funksjonene som er utviklet og sette sammen i én mobilapplikasjon. Begge applikasjonene er basert på samme rammeverk. Dette legger grunnlag til at det skal være enkelt å implementere funksjonene fra den ene applikasjonen til den andre.

Resultatet fra prosjektet kan være nyttige for andre grupper eller prosjekter. Spesielt når det kommer til arbeid som inkluderer bruk av teknologi som vedkommende ikke har brukt eller kjennskap til fra før. Det å starte et stort tidsbasert prosjekt med mangel på kunnskap er noe som må vurderes nøye med tanke på risikoer og mål for prosjektet. Det er viktig å ta hensyn til tiden som brukes på å tilegne seg denne nye kunnskapen på når en vurderer målene og resultatet.

9 REFERANSER

9.1 Kildeliste

1. Facebook (2020) Getting Started. Tilgjengelig fra: https://reactnative.dev/docs/getting-started (Hentet: 10.03.20).
2. React Navigation (2020) Getting Started. Tilgjengelig fra: https://reactnavigation.org/docs/getting-started (Hentet: 12.03.20).
3. Wix (2020) react-native-calendars. Tilgjengelig fra: https://github.com/wix/react-native-calendars (Hentet: 25.03.20).
4. Scrum (software development) (2020). Tilgjengelig fra: https://en.wikipedia.org/wiki/Scrum_(software_development) (Hentet: 03.04.20).
5. Extreme Programming (2020). Tilgjengelig fra: https://en.wikipedia.org/wiki/Extreme_programming (Hentet: 03.04.20).
6. Raphael Ugwu (2020) Overcoming single-threaded limitations in React Native. Tilgjengelig fra: https://blog.logrocket.com/overcoming-single-threaded-limitations-in-react-native/ (Hentet: 25.04.20).
7. Alfian Losari (2018) Firebase Realtime Database. Tilgjengelig fra: https://medium.com/@alfianlosari/firebase-realtime-database-many-to-many-relationship-schema-4155d9647f0f (Hentet 19.05.20).
8. React Native: What it is and how it works. Tilgjengelig fra:

   https://medium.com/we-talk-it/react-native-what-it-is-and-how-it-works-e2182d008f5e (Hentet 24.05.20).
9. Firebase Authentication. Tilgjengelig fra: https://firebase.google.com/docs/auth (Hentet 19.05.20).
10. Jest (2020) Getting Started. Tilgjengelig fra: https://jestjs.io/docs/en/getting-started.html (Hentet 20.04.20).
11. Guilty (2019) Hybrid- vs native-app. Tilgjengelig fra: https://guilty.no/blogg/hybrid-vs-native-app (Hentet: 28.05.20).
12. Spring (2020) Why Spring?. Tilgjengelig fra: https://spring.io/why-spring (Hentet: 04.05.20).
13. Firebase (2020) Firebase Realtime Database. Tilgjengelig fra: https://firebase.google.com/docs/database/?gclid=CjwKCAjwztL2BRATEiwAvnALcoCkcQDL7KdeCqUiL1F7bG9eu9ED5GJGmXFW5-GU4YeBvrpgCAQTfxoCDHUQAvD_BwE (Hentet: 22.05.20)
14. YouTube-film (Figur 2) - Introducing Firebase Authentication (2016). Tilgjengelig fra: https://www.youtube.com/watch?v=8sGY55yxicA&feature=emb_title (Hentet: 15.05.2020)

9.2 Figurliste

10 Vedlegg

10.1 Risikoliste

Nr	Suksessfaktorer	Sannsynlighet (1-5)	Konsekvens (1-5)	Risikofaktor (S*K)	Tiltak
1	Applikasjonen kan ikke brukes av oppdragsgiver	2	5	10	Opprettholde god kommunikasjon med oppdragsgiver.
2	Misforstått krav og spesifikasjoner	2	4	8	Skrive gode møtereferater og spør hvis noe er uklart.
3	Skjev arbeidsfordeling innad i gruppen	3	2	6	Opprettholde god kommunikasjon i gruppen og jobbe på et felles dokument så ikke flere jobber med det samme.
4	Fravær/sykdom	2	4	8	Regjeringen har satt opp tiltak for oss. Arbeider hjemmefra og unngår store folkemengder.
5	Dårlig samarbeid	2	3	6	Gi hverandre ærlige tilbakemeldinger på hvordan arbeidet går.
6	Mangel på kompetanse	2	4	8	Ha fokus på å tilegne seg nødvendig informasjon.
7	Dårlig disponering av tid	3	4	12	Ikke utsette ting til siste liten og sette av nok timer til arbeid.
Skrevet 05.05.2020					

1.

Dette er en risiko som vi får svar på når applikasjonen skal testes av oppdragsgiver. Sannsynligheten for dette er relativt liten, men konsekvensen er derimot veldig stor. Dersom ikke applikasjonen kan brukes får ikke oppdragsgiver det de har spurt om. For å unngå dette er det viktig at vi opprettholder god kommunikasjon med oppdragsgiver gjennom hele utviklingsperioden.

2.

I begynnelsen av prosjektet brukte vi mye tid på å kartlegge alle kravene og spesifikasjonene gjennom hyppige møter med oppdragsgiver. Det ble skrevet møtereferater fra alle møtene slik at vi kunne gå tilbake å se om det var noe som var glemt. Dette gjorde vi for å minske sannsynligheten for denne risikoen og som vi kan se senere i prosjektet, har dette fungert.

3.

Skjev arbeidsfordeling innad i gruppen vil alltid være en risiko. Ofte blir ikke dette oppdaget heller. For å unngå dette så forsøkte vi å opprettholde god kommunikasjon innad i gruppen samt at vi skrev rapport på ett felles dokument. Dermed kunne alle gruppemedlemmene se hva som ble gjort av hvem. Samtidig så gjorde dette at flere jobbet med samme punkt.

4.

Fravær/Sykdom er et punkt som er veldig aktuelt i disse koronatider. Fra midten av mars ble det innført retningslinjer fra regjeringen som sa at det ikke var lov til å oppholde seg på skolen. I tillegg skulle alle som møtte hverandre opprettholde 1 meter avstand. Dette førte til at hele bacheloren måtte skrives fra eget hjem og fravær ble et faktum. På en annen side så gjorde disse retningslinjene at sannsynligheten for å bli syk ble mindre enn den var før.

5. Dårlig samarbeid løser vi med god kommunikasjon og ærlige tilbakemeldinger på arbeid. Sannsynligheten for at vi samarbeider dårlig er relativt liten, da vi har samarbeidet på oppgaver i mange år og kjenner hverandre godt.

6 og 7.

Mangel på kompetanse og dårlig disponering av tid vil vi få mer og mer svar på ettersom prosjektet nærmer seg slutten. Mye av kompetansen som oppgaven krever har gruppen enda ikke støtt på. Dette kommer av at vi jobber med enkeltdeler av applikasjonen. Når dette skal settes sammen kan man støte på problemer. Frem til nå så har det enda ikke oppstått problemer som ikke har latt seg løse. Når det gjelder disponering av tid så er dette også noe vi vil få svar på når det nærmer seg sluttfasen av prosjektet. På deloppgavene har gruppen stort sett fulgt tidsplanen med kun få dager avvik.

10.2 GANTT diagram

10.3 Kravspesifikasjoner fra Kronbar

Punkt	Tittel	Beskrivelse	Mål	Prioritet 1-3
1	Flere plattformer	Applikasjonen må fungere på både Android og iOS. Eventuelle andre operativ-systemer er positivt, men ikke viktig.	Applikasjonen fungerer på både Android og iOS.	1
2	Vaktsystem	I dag bruker Kronbar en egen nettside for å melde seg opp på vakter. Vi ønsker å gjøre dette i en applikasjon, slik at frivillige får de samme funksjonene på mobil. Dette innebærer at en admini-strator kan legge inn vakter, hvor hver vakt kan ha flere frivillige av forskjellige typer (ansvarshavende, barperso-nell, teknisk osv). De frivillige skal da kunne sette seg opp på disse vaktene. En admini-strator skal kunne ta folk av eller legge til frivillige på en vakt. Administratorer skal kunne se en fullstendig over-sikt over alle frivillige. Administratorer skal kunne opprette forskjellige frivillig-typer med forskjellige vakt-krav. Vaktkravet skal kunne endres.	Et fungerende internsystem i applikasjonen. Frivillige skal kunne sette seg opp på vakter og se sine vakter.	1
3	Oppslagstavle	En administrator skal kunne legge ut oppslag på en oppslagstavle. De frivillige skal mota push-varsel om at det er kommet nytt innlegg.	Erstatte ukentlige oppdateringer på facebook med en fungerende oppslagstavle.	2

4	Internforum	De frivillige skal kunne legge ut at de ønsker å bytte en vakt, og andre frivillige skal kunne godta byttingen. Det skal også være mulig å legge ut innlegg av den mer sosiale sorten.	Automatisering av vaktskifte samt et hyggelig forum for de interne.	2
5	Gamification	De frivillige skal se hvordan de ligger an i forhold til vakt-kravet for inneværende semester. De frivillige skal se hvor mange «poeng» de har, hvor en vakt over vaktkravet er ett poeng. Stand er 0,5 poeng.	De frivillige skal kunne se egen innsats.	2
6	Lovlige personallister	Skatteetaten har noen krav til personallister som gjør at dagens løsning ikke holder, og vi må føre personalliste på papir også. Det er ønskelig å slippe dette.	Internsiden skal fungere som personalliste	3
7	Integrasjon med dagens løsning	Systemet skal være integrert med dagens system slik at de interne kan velge om de skal bruke mobilapplikasjonen eller en internettside.	Det skal fortsatt være mulig å bruke datamaskin.	3
8	Kalender-integrasjon	Når frivillige setter seg opp på vakt skal de få tilbud om å automatisk legge inn vakten i kalenderen på telefonen.	Gjøre det lett for de frivillige å holde kontroll på tiden sin.	3
9	Push-varsler	De frivillige skal motta varsler når en vakt nærmer seg og det er ledige plasser på vakten.	Lettere å fylle vakter.	3
10	Personlige valg	Alle push-varsel-typer skal kunne skrus av og på fra applikasjonen.	De interne blir ikke spammet hvis de skrur av varselet.	3
11	Beregning av arbeidstimer	I forbindelse med søknader spørres det av og til om hvor mange timer det arbeides med organisasjonen. En oversikt over hvor mange timer det er lagt ned per år er ønskelig. Hvis en vakt er 8 timer og det er 4 stk på vakt teller det som 32 arbeids-timer, 5 stk 40 timer osv.	Det skal være lett å få tak i administrative tall som kreves i søknader.	3
12	Integrasjon med ekstern-applikasjon	Flytte hele applikasjonen inn i en applikasjon for Kronbar sine gjester, gjemt bak en innlogging.	Kun en applikasjon for alle Kronbar sine behov.	3