

# **Sotware Engineering og Testing**

## **Rapport**

Gruppe 3

Ole Sander Skjørberg, Omid Mahrofzade, Sander Smedstad, Mykola Dukhnivskyi, Mohamed Abdiqadir Bashir

21. november 2024  
Halden



# Sammendrag

Hovedmålet for prosjektet var å utvikle en brukervennlig og sikker barnesikringsløsning for skap og skuff. Løsningen skulle basere seg på NFC- og Bluetooth-teknologi. Løsningen skulle bidra til å beskytte barn mot potensielt farlige gjenstander i hjemmet, og tilby foreldre et nyttig og fleksibelt verktøy for å sikre hjemmet. Prosjektet hadde også spesifikke delmål:

**Utvikle en funksjonell og stabil låsemekanisme:** Vi fullførte dette målet ved å designe en låsemekanisme basert på *compliant mechanisms*, som tilbyr en enkel montering og fleksibilitet for bruker. Låsen kan benyttes via en mobil applikasjon gjennom Bluetooth Low Energy (BLE), eller via NFC.

**Brukervennlig design og montering:** Dette målet ble også oppfylt ved at låsen har simple metoder for bruk, den kan monteres med dobbeltsideteip eller med skruer. Den har også gode metoder for bruk gjennom app, mobil og kortbasert tilgang, noe som gir brukern fleksibelt bruk og enkel installasjon.

**Bærekraftig materialvalg:** Prosjektet brukte 3D-printede resirkulerte materialer, men disse testene viste at noen materialer som PETG og PLA ikke kunne oppfylle kravene til varigheten. Nylon var slitesterkt, men viste seg å være for stift. Det kommet til å kreve mer arbeid for å finne det optimale materialet når vi ønsker å få god balanse mellom fleksibiliteten og holdbarheten.

**Sikkerhet og nødløsninger:** Sikkerheten til produktet ble forsterket med at låsen gir varslinger ved lavt batteri og har manuell nødløsning ved strømtap. Selv om en grunnleggende løsning ble implementert, ble det planlagt å forbedre krypteringssystemet noe ytterligere for å sikre dataoverføring.

## 0.1 Oppsummering og videre anbefaling

De viktigste målene for prosjektet ble stort sett oppfylt, spesielt de som fokuserer på funksjonalitet, brukervennlighet og sikkerhet. Samtidig gjenstår noen forbedringer, særlig med hensyn til materialvalg og videre utvikling av datasikkerheten i appen. Dette kan kreve ytterligere ressurser og testing.

# Innhold

<b>Sammendrag</b>	<b>i</b>
0.1 Oppsummering og videre anbefaling . . . . .	i
<b>Innhold</b>	<b>ii</b>
<b>Figurer</b>	<b>iv</b>
<b>1 Innledning</b>	<b>1</b>
1.1 Bakgrunn og motivasjon . . . . .	1
1.2 Rapportens oppbygging . . . . .	2
<b>2 Teori</b>	<b>4</b>
2.1 Liknende løsninger . . . . .	4
2.2 Teknologier og verktøy . . . . .	7
<b>3 Metode / Prosess</b>	<b>11</b>
3.1 Metodikk i gruppe . . . . .	11
3.2 Bruker og brukertesting . . . . .	13
3.3 Risikovurdering for produkt . . . . .	14
3.4 Referanser relevant for verdivurdering . . . . .	20
<b>4 Design / Frontend</b>	<b>21</b>
4.1 Låsemekanismen . . . . .	21

4.2	Materialvalg og oppbygging . . . . .	22
4.3	Bruk av batterier . . . . .	23
4.4	Fremgangsmåte . . . . .	23
4.5	Skisser / Prototyper . . . . .	23
<b>5</b>	<b>Implementasjon / Gjennomføring / Backend</b>	<b>30</b>
5.1	Kravspesifikasjon . . . . .	30
5.2	Avhengigheter og arkitektur . . . . .	32
5.3	Maskin-implementasjon . . . . .	34
5.4	App funksjonalitet og oppbygging . . . . .	50
<b>6</b>	<b>Testing / Evaluering / Resultat</b>	<b>53</b>
6.1	Testing av materiale . . . . .	53
6.2	Testing Raspberry PI . . . . .	53
6.3	Applikasjon testing . . . . .	56
<b>7</b>	<b>Drøfting / Diskusjon</b>	<b>58</b>
7.1	Teamvalg . . . . .	58
7.2	Sikkerhetsvalg . . . . .	58
7.3	Designvalg og brukeropplevelser . . . . .	59
7.4	Teknologivalg . . . . .	59
7.5	Valg av programmeringsspråk . . . . .	60
7.6	Funksjonalitetsvalg . . . . .	60
<b>8</b>	<b>Konklusjon</b>	<b>62</b>
	<b>Referanseliste</b>	<b>64</b>
8.1	Nettsider . . . . .	64
8.2	Kunstig intelligens . . . . .	66

# Figurer

3.1	Sprinter . . . . .	13
3.2	Riskoidentifisering . . . . .	16
3.3	Risikoanalyse . . . . .	18
3.4	Risikohåndtering . . . . .	19
4.1	Låsemekanisme . . . . .	25
4.2	Tidlig prototype . . . . .	25
4.3	PETG og PLA . . . . .	26
4.4	Flex . . . . .	27
4.5	Nylon . . . . .	28
4.6	Skisse1 . . . . .	29
4.7	Skisse2 . . . . .	29
5.1	Lock Databse . . . . .	37
5.2	Access Log, Fremmednøkler i user id og lock id som refererer til hverandre i dens tabell. . . . .	39
5.3	Users tabell . . . . .	40
5.4	Locks tabell, Kan eventuelt legge til serienummer eller modell. . . . .	40
5.5	Lock Ownership tabell, Sammensatt PK av user id og lock id og fremmednøkkel på begge som refererer til hverandre - skaper et mange til mange forhold mellom brukere og låser. . . . .	40
5.6	ER-diagram . . . . .	41

5.7 RealVNCViewer . . . . .	44
5.8 Prototype . . . . .	44
5.9 Databasefeil . . . . .	45
5.10 Database . . . . .	48
5.11 Koblingsskjema . . . . .	49
5.12 Appdesign . . . . .	52
6.1 Resultat av testing Ras.Pi . . . . .	55

# **Kapittel 1**

## **Innledning**

Små barn har mykere bein, mindre utviklet muskulatur og er ikke utviklet nok til å vurdere farer rundt seg og dermed gjør dem ekstremt skjøre. Derfor er det veldig viktig og kritisk at vi gjør alt vi kan for å beskytte dem fra farlige situasjoner.

Barne-sikringsløsninger er da ekstremt sentrale i hus med småbarn. En barnesikring kan være mye forskjellig, det kan være alt fra en plastplugg på stikkontakter slik at barn forhindres fra elektriske støt, til en grind som brukes foran dører og trapper for å forhindre barn i å sette seg selv i farlige situasjoner, skap og skuff sikringer er også viktige slik at barn holder seg unna eventuelle farlige gjenstander og masse mer! Vi skal derfor i denne oppgaven ta tak i en barne-sikringsløsning til skap og skuffer med bruk av NFC-og bluetooth teknologi.

### **1.1 Bakgrunn og motivasjon**

Vi synes NFC-teknologien er et spennende og innovativt konsept, og vi ønsker å ta det enda lenger. Vi har diskutert flere ideer for hvordan vi kan benytte denne teknologien for å skape noe nyttig til hjemmet, som vil være til fordel for folk i hverdagen. Et medlem av gruppen snakket med familien sin om denne NFC-teknologien, og da oppstod en interessant idé. Tanken var å utvikle en lås som enkelt kunne sikre skapene, med hensyn på småbarnsforeldre. Nevøen hans på 1 år har en tendens til å åpne skapene og utforske ting han ikke burde leke med – som dyre glass eller elektronikk under TV-en. En slik lås ville derfor være en nyttig løsning for å forhindre at små barn får tilgang til farlige eller skjøre gjenstander.

Motivasjonen vår er å gjøre NFC-teknologien nyttig i hjemmet, da den er meget brukervennlig. Den er også mer kostnadseffektiv enn mange andre løsninger på markedet og gjør da NFC til en ideell teknologi som kan møte mange behov til moderne hjem.

### **1.1.1 Problemstilling / Forskningsspørsmål**

Hvordan sikre farlige gjenstander fra småbarn med en unik barnesikringsløsning, med fleksible, innovative og moderne løsninger som gir gode brukeropplevelser?

## **1.2 Rapportens oppbygging**

Under denne seksjonen finner ligger det en oversikt over hva hvert kapittel inneholder, samt litt informasjon om disse.

### **1.2.1 Teori**

I denne seksjonen skal det gjøres rede for all teori som ligger bak prosjektet.

### **1.2.2 Metode**

Under metode blir det reflektert over hvilke metoder vi som gruppe har benyttet oss av, og om disse fungerte for oss eller ikke.

### **1.2.3 Design**

Beskrivelser av produkt og bedriftens design, logo og navn.

### **1.2.4 Implementasjon**

Hele den tekniske prosessen skal forklares i detalj og drøfte avgjørelser som er unike for dette prosjektet.

### **1.2.5 Testing/Evaluering/Resultat**

Testing og resultater som forekommer.

### **1.2.6 Drøfting**

Drøfte problemstillingen og går gjennom hele prosjektet.

### **1.2.7 Konklusjon**

Dette er en oppsummering av drofingen, her blir hovedpoeng fra oppgaven fremstilt.

# Kapittel 2

## Teori

I dette kapittelet skal det gjøres rede for all teori som ligger bak prosjektet, samt informasjon om teknologien vi har benyttet oss av, hvorfor vi valgte akkurat denne teknologien og hvilke andre løsninger som allerede er på markedet og sammenlikne den med vår løsning. Videre skal også tas tak i arkitekturen, rammeverket, teknologi og verktøy vi har brukt og hvorfor.

### 2.1 Liknende løsninger

I denne inndelingen vil vi sammenlikne vårt produkt med andre liknende løsninger.

#### 2.1.1 Eksisterende barnesikringslåser for skap

Det finnes allerede barnesikringsløsninger til skap som benytter seg av NFC og/eller bluetooth. I denne sammenlikningen skal det vurderes hva som skiller vår løsning ut fra 2 andre liknende produkter når det kommer til kvalitet, brukervennlighet og pris.

#### Kort om produktene

*Ikea Rothult* er en rimelig skaplås som benytter seg av NFC-teknologi og åpnes kun ved vedlagte kort (kan også legge til et kort til). Prisen på denne er 379kr per enhet, og hver enhet kommer med låsen, 2 NFC-nøkkelskjermer, skruer for montering og bruksanvisning.

*Yeelock* er også en skaplås til en god pris. Låsen benytter seg av Bluetooth for å åpne skapet. Prisen per enhet er på 695kr og følger med låsen, dobbeltsideteip og bruksanvisning.

*TapNest* som er vår skaplås, benytter seg av NFC-teknologi og Bluetooth, samt en fysisk løsning. Pris per enhet vil koste 1200kr, pakken inneholder selve låsen, et NFC-kort, en bruksanvisning, 2stk ladbare AA-batterier, dobbelt-sideteip for ryddig installasjon og bruksanvisning.

**Brukervennlighet**

Ikea sin Rothult monteres med skruer og må borres inn i skapet, dette kan være en dårlig løsning for mange da det krever presisjon og nøyaktighet og kan være at nytte verktøy ikke er tilgjengelig hos alle. Låsen kan kun åpnes med kort og har ingen andre måter å åpne skapet på noe som begrenser tilgjengelighet.

Sammenliknet med Rothult bruker Yeelock sin løsning dobbelt-sideteip for montering som gjør det til en enklere og mer ryddig montering. Låsen åpnes kun ved bruk av en app via Bluetooth på mobil/nettbrett, og kan kun åpnes slik, dette kan være en utfordring dersom du må åpne skapet uten strøm på enhetene.

TapNest(vårt produkt) som Rothult bruker dobbelt-sideteip for enkel og sømløs motering. Vår løsning benytter seg også av flere måter å åpne låsen, den kan brukes ved et tæpp "med NFC-kort, åpnes via app og åpnes via 'tæpp' med mobilen som gjør det enkelt for forskjellige personer med forskjellige kunnskapsnivåer å åpne skapet. Det skal også nevnes at fordi vi velger å ha en såpass teknisk løsning, kommer vårt produkt også til å være litt større i størrelsen enn de andre produktene til vi også får løst dette.

**Pris**

Rothult sin pris er på 379kr per enhet og er det rimeligste alternativet. Det kommer av enkel og begrenset og bla. at kunden er nødt til å kjøpe f.eks. batterier ved siden av.

Yeelock sin løsning koster 695kr per enhet og har også begrenset funksjonalitet med har en mer teknisk løsning enn Ikea sin Rothult med app. Yeelock sin løsning kommer heller ikke med batterier så disse må kjøpes ved siden av.

TapNest koster 1200kr og er da det dyreste alternativet i akkurat denne sammenlikningen. Denne løsningen inkluderer også mer fleksibilitet i åpningsmuligheter for låsen som kort, app og mobil-tæpp".

**Nødløsninger**

En slik elektrisk lås trenger nødløsninger for at den skal unngå å forbli lukket. Rothult sin løsning er at enheten gir fra seg lyd når batteritilstanden blir lav, dersom den går tom for strøm er det ingen andre fysiske løsninger for å åpne skapet og må da åpnes med makt.

Yeelock har også en liknende løsning, ved lav batteritilstand piper enheten hver gang den blir brukt, og ved kritisk tilstand åpnes låsen automatisk. Yeelock har likevel ingen fysiske åpningsmuligheter. TapNest har flere løsninger dersom batteriet blir lavt, den kommer til å gi ut et rødt lys dersom batteritilstanden er lav, men åpner seg ikke dersom den går tom for strøm. Sammenliknet med de andre låsene har vår lås en fysisk løsning dersom den går tom for strøm, og det gjør også at den alltid forholder seg låst til du selv velger å åpne skapet.

**Batterier**

Rothult benytter seg av tre enkelte AAA-batterier som betyr at de må byttes ut regelmessig. Yeelock benytter seg av to AAA-batterier som kjøpes separat der kunden kan velge mellom å bruke ladbare AAA-batterier eller enkelte.

TapNest benytter seg av 2 ladbare AA-batterier som følger med låsen. Ved bruk av ladbare batterier som er litt dyrere enn engangsbatterier, sparer vi miljøet ved unødig el-avfall. Ifølge Tek.no, hevder IKEA at deres oppladbare batterier vil være mer miljøvennlige etter ca. 50 sykluser, hvorav de fleste oppladbare batterier kan bruker opptil flere 100 ganger før de må byttes. I testen til Tek.no finner de også ut av at kostnaden av et ladbart batteri vil være tjent inn etter 4-6 sykluser.

## Oppsummering og konklusjon

I denne sammenlikningen har vi brukt 2 liknende låser til en mindre pris enn vår egen, dette var fordi vi ikke fant låser med samme funksjonalitet. De eneste låsene med denne teknologien var litt større låser til en enda dyrere pris. Fordelen med prisen vi har satt, gjør at ALT er inkludert i pakken. Kunden bestiller pakken og kan sette opp låsen med engang uten hodebry om å trenge å kjøpe annet ved siden av. Vi har også en fordel med at vi tilbyr dobbelt-sideteip som monteringsløsning, fordel for å borre hull i skapene for montering. Løsningen vår blir større enn konkurrentene da vi har mer teknologi og fleksibilitet vi ønsker å ha plass til, men ikke altfor store forskjeller mellom disse.

Konkurrentene våre har ingen nødløsning som vår lås, da låsen vår er den eneste med en fysisk løndløsning dersom teknologien skulle svikte. Batteriene våre vil være ladbare og bidrar til å minske el-avfall og er derfor mer miljøvennlige enn tradisjonelle AA-batterier.

Målet vårt er å tilby en rimelig lås i forhold til funksjonaliteten den har. Som nevnt tidligere finnes det ingen produkter som er identiske med vårt, noe som rettferdiggjør prisen. I tillegg har vi argumentert for dette i sammenligningen, der vi fremhever styrkene ved vårt produkt.

## 2.2 Teknologier og verktøy

I denne inndelingen gjør vi rede for hva Bluetooth og NFC-teknologi er og hvorfor vi har valgt teknologien, samt sikkerhet rundt disse.

### 2.2.1 NFC - teknologi

NFC står for «Near Field Communication» og gjør det slik at enheter klarer å kommunisere med hverandre over ekstremt korte avstander. Denne teknologien finnes i nyere smarttelefoner og brukes av nordmenn hver dag til bla. kontaktløs betaling med telefon. NFC benytter seg av radiosignaler der den er avhengig av at minst en av enhetene har strøm, f.eks. et kort med NFC-chip brukes for å komme seg inn en dør med en «terminal» du skanner kortet på, da kortet er strømløst og terminal koplet til strøm (typisk på kontorbygninger og skolebygg).

Vi valgte akkurat denne teknologien fordi den er moderne, lett tilgjengelig og enkel for et gjennomsnittlig person å bruke. Teknologien blir allerede brukt av flere eldre personer ved kjøp i butikker, der bankkort blir lagt inn i en applikasjon på mobilen og bruker da den integrerte NFC-brikken til å tæppe på en betalingsterminal. Grunnen til at vi viser til at det er mange eldre som bruker og mestrer denne teknologien, er fordi vi alle vet at eldre generelt har dårlige IT-kunnskaper enn de som er litt yngre. Vår argumentasjon vil da være at dersom vi har en teknologi som til og med eldre mennesker mestrer, vil dette ikke være et problem for vår målgruppe å også mestre denne teknologien. Vår målgruppe er da småbarns foreldre, disse er som regel mellom 25-35 år gamle og kommer generelt sett ikke til å ha noe problem ved bruken av denne teknologien.

## 2.2.2 Bluetooth - teknologi

I vår enhet benytter vi også Bluetooth-teknologi for å muliggjøre kommunikasjon mellom låsen og tilhørende app. Denne teknologien er avgjørende for at brukerne enkelt kan kontrollere og overvåke låsen direkte fra mobilen, noe som skaper en sømløs brukeropplevelse. Ettersom låsen er batteridrevet, har vi valgt å bruke en spesialversjon av Bluetooth kalt BLE (Bluetooth Low Energy). BLE-teknologi er optimalisert for strømbesparende applikasjoner, noe som bidrar til å forlenge batteriets levetid betraktelig.

BLE ble introdusert som en energieffektiv variant av Bluetooth, utviklet for applikasjoner som trenger å kommunisere trådløst uten høyt energiforbruk. Dette er spesielt viktig for batteridrevne enheter, som vår lås, hvor det er ønskelig å minimere hvor ofte batteriene må byttes eller lades. Dette gjør at låsen vår kan fungere i lengre perioder uten at brukeren trenger å bekymre seg for strømnivået. Ved å benytte BLE sikrer vi at produktet er både pålitelig og praktisk for sluttbrukeren.

Gjennom BLE-forbindelsen kan brukeren enkelt koble appen til låsen og få tilgang til ulike grunnleggende funksjoner. Disse funksjonene inkluderer blant annet oppsett av låsen, åpning og lukking, overvåking av batteristatus, samt muligheten til å legge til NFC-kort for å gi flere brukere tilgang. I tillegg gir BLE muligheten til å utføre andre innstillinger, som tilpasning av sikkerhetsnivå og spesifikke tilganger. Ved å bruke BLE kan vi også sende viktige oppdateringer og varsler til brukeren, for eksempel om batterinivået begynner å bli lavt, slik at brukeren alltid er informert og kan ta nødvendige forholdsregler.

Denne løsningen muliggjør også enkel og sikker paring mellom mobilen og låsen, der brukeren kan etablere en kobling gjennom appen ved hjelp av et intuitivt oppsett. Ved å bruke BLE drar vi nytte av teknologiens avanserte sikkerhetsmekanismer, som sikrer at kun autoriserte brukere kan koble seg til låsen. Dette gir brukeren en trygghet i at systemet er sikkert, samtidig som det gir fleksibilitet til å administrere tilganger på en brukervennlig måte.

## 2.2.3 Sikkerhet i teknologien

I dagens digitale landskap er sikkerhet en avgjørende komponent for å beskytte sensitiv informasjon og sikre trygg kommunikasjon. Kryptering er sentral for å gjøre data utilgjengelig for uvedkommende, noe som styrker et produkt til å være både sikkert og pålitelig. Ved å kryptere data sikrer man at kun autoriserte parter har tilgang til informasjonen, noe som vesentlig reduserer risikoen for uautorisert tilgang og misbruk. Uten tilstrekkelig kryptering kan data lett komme i feil hender, noe som kan få alvorlige konsekvenser for både individer og organisasjoner. I vårt produkt benyttes en app som kommuniserer med en server og database, og videre med låsen via Bluetooth. For å beskytte sensitiv informasjon vurderes det å sikre flere ledd i denne kommunikasjonen med kryptering i fremtiden. Dette vil hindre uvedkommende i å få tilgang til våre data.

For øyeblikket opplever vi utfordringer med å opprette en stabil Bluetooth-forbindelse mellom appen og låsen, noe som hindrer en praktisk demonstrasjon av krypteringsløsningen. Serveren er heller ikke avgjørende for å demonstrere produktets grunnleggende funksjonalitet. Vi har derfor valgt å fokusere på de tiltakene som er nødvendige for å implementere kryptering i fremtiden.

## Implementering av kryptering i Java Bluetooth

Ved å benytte sikre Bluetooth-sokler og Java's kryptografibiblioteker, kan det effektivt kryptere Bluetooth-forbindelsen mellom Android-appen og låsen på Raspberry Pi. Følgende er en overordnet beskrivelse av fremgangsmåten:

### Opprett sikre Bluetooth-sokler:

- På Android-appen: Velg en metode som benytter sikre Bluetooth-sokler ved opprettelse av forbindelsen, slik at data mellom appen og låsen automatisk krypteres av Bluetooth-protokollen.
- På Raspberry Pi-låsen: Konfigurer Bluetooth-programvaren til å akseptere sikre forbindelser, slik at låsen kun kommuniserer med enheter som har riktig autentisering.

### Applikasjonslagskryptering for ekstra sikkerhet:

Selv om Bluetooth-soklene i seg selv er sikre, kan vi legge til et ekstra lag ved å kryptere dataene i applikasjonen ved hjelp av Java's kryptografibiblioteker.

*Kryptering på avsendersiden:* Data krypteres før overføring ved hjelp av en krypteringsalgoritme som AES. (Advanced Encryption Standard) Dekryptering på mottakersiden: Mottatte data dekrypteres på låsen med samme algoritme og nøkkel.

### Sikker nøkkeldistribusjon:

*Deling av nøkler:* Appen og låsen må ha tilgang til samme krypteringsnøkkel, som kan utveksles ved paring eller lagres sikkert på begge enhetene.

### Sikker lagring:

- På Android: Bruk Android Keystore System for sikker lagring. - På Raspberry Pi: Lagre nøkler i et beskyttet område på filsystemet med begrenset tilgang.

### Paring og autentisering:

- Etabler en sikker forbindelse: Appen og låsen må pares for å sikre gjensidig tillit og trygg nøkkeldistribusjon. Paringsprosessen bruker vanligvis en PIN-kode eller tilsvarende bekrefteelse.
- Enhetsstillatser: Forsikre at begge enheter har riktige stillatser og innstillinger for sikre Bluetooth-forbindelser.
- Oppdatert programvare: Sørg for oppdaterte Bluetooth-drivere og programvare for å støtte de nyeste sikkerhets-protokollene.

### Testing og validering:

- Test krypteringen: Verifiser at dataene forblir krypterte under overføring og kan dekrypteres på mottakersiden.

- Feilhåndtering: Implementere mekanismer for å håndtere feil i kryptering og tilkobling for å sikre systemets robusthet. Denne implementeringen av kryptering i Bluetooth-forbindelsen bidrar til å styrke sikkerheten og påliteligheten til produktet, og sikrer at dataene forblir beskyttet under kommunikasjonen.

# **Kapittel 3**

## **Metode / Prosess**

I dette kapitelet skal bedriften gjøre rede for arbeidsmetodikken vi har brukt for dette prosjektet, der vi forklarer hvordan prosessen har vært, og div opp-og nedturer. Her vil vi også vise til hvem våre brukere er, hva behovene til brukerne våre er, som vårt produkt skal løse, og hvordan vi har planlagt å brukerteste produktet vårt. Videre har vi lagt ved en risikoanalyse av produktet vårt, der vi reflekterer over hva vi ser på som kritiske og ikke så kritiske potensielle problemer, samt håndtering av risker.

### **3.1 Metodikk i gruppe**

Vi har som gruppe brukt de første 3–4 ukene på å ha planlagte, ukentlige møter der vi diskuterte hvordan vi skulle gjennomføre prosjektet. Gruppen oppdaget imidlertid at vi ikke fikk lagt ned nok arbeid i prosjektet, og vi satt litt fast i planleggingsfasen. Derfor satte vi opp en veilednings-time med foreleser for å 'tvinge' oss til å komme i gang. Dette hjalp oss mye, og vi følte endelig at vi fikk kontroll over prosjektet.

Selv om de første ukene ikke var så effektive som vi ønsket, klarte vi å fordele hovedansvarsområder for å sikre at prosjektet nådde mål. Vi ble enige om rollefordelingen, der enkelte fikk ansvar som utvikler/koder, frontend-design, database og generelle oppgaver, app-design, og dokumentasjon/økonomi utover personlige egenskaper og tidligere erfaringer. Vi oppdaget da raskt hvem som passet til de ulike oppgavene, så denne fordelingen var heldigvis ukomplisert.

Gruppen ble enige om å bruke Agile-metodikken, der vi delte oppgavene inn i individuelle sprinter og hadde ukentlige mandagsmøter (sprint reviews) som varte fra 2–5 timer. På møtene gikk vi gjennom fullført arbeid og planla videre oppgaver for resten av uken. Denne metoden hjalp oss med å gjøre små justeringer underveis dersom noe ikke gikk som forventet, og vi kunne enkelt tilpasse oss dersom behovene endret seg. Til planlegging av sprintene brukte vi GitHub til å dele inn arbeidet i issues, og vi opprettet nye for hver sprint. Vi brukte strukturen, for eksempel 'Design Frontend S5', som gir oversikt over arbeidsområdet, der 'S5' står for sprint nummer 5. Under disse issue-ene delte vi opp i konkrete oppgaver med et tidsestimat for hver oppgave. Timer brukt på

oppgavene dokumenteres i et Excel-dokument med antall timer brukt på hver oppgave og en kommentar om hva timene har gått til. Timelister vedlegges i rapporten.

En ulempe med Agile - metoden var at alle i gruppen måtte være disiplinerte, bruke tiden som planlagt, og komme med resultater til møtene. Vi opplevde noen opp- og nedturer, der enkelte gruppemedlemmer ikke alltid klarte å levere det planlagte antallet timer. Dette førte til at vi tidvis lå litt bak skjema og måtte legge inn ekstra timer i neste sprint. Heldigvis var denne arbeidsformen som regel fordelaktig, ettersom vi ikke alltid var avhengige av hverandres arbeid for å fullføre våre egne oppgaver.



Figur 3.1: Sprinter

I tillegg til de ukentlige mandagsmøtene, startet vi med møter hver onsdag og torsdag i begynnelsen. Etter hvert som mandagsmøtene ble mer sentrale, fokuserte vi hovedsakelig på disse, i tillegg til torsdagsmøtene. På torsdagsmøtene samlet vi oss for å oppsummere hva vi hadde oppnådd i løpet av sprinten, samt avklare eventuelle spørsmål eller samarbeidsmuligheter før sprintreview på mandag. Disse møtene varte fra 15 minutter til 1,5 time, avhengig av hva som måtte diskuteres. Møtene viste seg å være til stor hjelp for å sikre at vi tidvis ble ferdige med sprinten.

I figur 'Sprinter' viser vi en oversikt over hvor mange sprinter som ikke er gjennomført av hvert gruppemedlem, samt hvor mange som er fullført. Figuren viser også et utkast der prosjektet ikke er fullført, for å illustrere de gjenværende sprintene.

## 3.2 Bruker og brukertesting

Våre brukere er primært foreldre med små barn som ønsker en enkel og pålitelig måte å sikre at skuffer og skap ikke er lett tilgjengelige for nysgjerrige smårollinger. I tillegg kan produktet vårt være nyttig for alle som vil beskytte eiendeler og holde dem utilgjengelige uten bruk av avanserte låser.

### 3.2.1 Personas

**Sara Kristiansen:** er 31 år gammel og har en datter på 11 måneder som heter Emma. Emma har nettopp lært å kravle, og hun elsker å utforske alle kroker i huset. Sara og samboeren Jonas er bekymret for at Emma kan klare å åpne et skap og komme i kontakt med farlige produkter, som vaskemidler under vasken.

**Marius Hauge:** er 28 år gammel og bor alene. Han har ingen barn, men liker å holde hjemmet sitt organisert og ryddig. Marius ønsker å beskytte enkelte eiendeler uten å måtte bruke avanserte låsesystemer. For ham er produktet en praktisk måte å gjemme vakkert ting fra uønsket tilgang.

**Linda og Petter Olsen:** er et ektepar i slutten av 30-årene med to barn, en fireåring og en seksåring. De har nylig flyttet inn i et nytt hjem og er opptatt av å sikre barnas sikkerhet på best mulig måte, spesielt når det gjelder farlige husholdningsartikler og skarpe gjenstander som oppbevares i skuffer.

### 3.2.2 Brukerbehov mot produkt

Brukerne trenger en pålitelig og brukervennlig låseløsning for skuffer og skap som beskytter små barn fra farlige produkter, som vaskemidler og skarpe gjenstander. For foreldre som Sara og Jonas er trygghet i hverdagen avgjørende – de ønsker å vite at barna deres ikke har tilgang til farlige ting i hjemmet. Marius ønsker å holde enkelte eiendeler beskyttet, selv uten barn i hjemmet, og Linda og Petter ønsker å ha en effektiv sikring for sine to barn. Produktet må derfor være både enkelt å bruke og sikre nok til at små barn ikke kan manipulere det. Løsningen bør også være fleksibel og diskret, slik at den passer inn i ulike hjemmemiljøer og kan brukes på forskjellige typer skap og skuffer uten å være til sjenanse.

### 3.2.3 Brukertesting

Vi planlegger å gjennomføre brukertesting med et representativt utvalg av våre målgrupper – det vil si foreldre med små barn, samt personer som ønsker å beskytte eiendeler. Testingen vil inkludere scenarier der brukerne installerer og anvender produktet i sitt eget hjem, for å sikre at det er enkelt å montere og bruke i hverdagen. Vi vil innhente tilbakemeldinger på brukervennlighet, sikkerhet og hvor godt produktet oppfyller deres behov. I tillegg vil vi gjennomføre observasjons-tester hvor vi kan se hvordan barn eventuelt forsøker å åpne sikrede skuffer og skap. Dette vil gi oss en god indikasjon på produktets effektivitet i virkelige situasjoner. Brukernes tilbakemeldinger vil bli brukt til å forbedre produktets design og funksjonalitet før endelig lansering.

## 3.3 Risikovurdering for produkt

I denne seksjonen vil bedriften gjøre rede for potensielle problemer med både produktet, og andre faktorer som kan påvirke bedriften og bruker.

### 3.3.1 Verdivurdering (Beskrivelse av hver verdi som er viktig å bevare)

**Sikkerhet for barn og uvedkommende:** Dette er kritisk for å forhindre ulykker med uautorisert tilgang. For eksempel, hvis et mindre barn åpner opp et skap med farlige gjenstander i seg, kan det føre til alvorlige skader. Derfor er dette høy prioritet.

**Sikring av tilgangen til lås:** Vi anser det veldig viktig å beskytte eiendom og verdisaker. Hvis ivedkommende får tilgang til låser som skal bevare dette, kan det føre til tyveri eller misbruk av sensitiv data. Dette er også høy prioritet.

**Funksjonalitet og strømtilkobling:** For å sikre at produktet er konstant operativt, må det være pålitelig og ha nok batteri-levetid og statusoppdateringer. For eksempel hvis produktet går tom for strøm uten at eieren har fått tatt tak i dette, kan det være ubruklig i kritiske øyeblikk. Dette er medium prioritert - av grunner som blir beskrevet videre.

**Brukervennlighet:** Produktet må være enkelt å bruke for å sikre at brukerne kan styre det problemfritt. Dersom produktet er for komplisert, kan det føre til feilbruk eller at man ikke vil bruke det. Dette kan også føre til dårlig rykte og mindre salg. Dette er medium prioritert.

**Personvern:** Beskyttelse av brukerdata og andre sensitiv informasjon er essensielt for å opprettholde tillit og lovgivning. Dersom persondata blir kompromittert, kan det føre til konsekvenser for brukerne og bedriften. Dette er høy prioritet.

**Konkurransefortrinn:** Bedriften anser det viktig å være oppdatert med våre konkurrenter i markedet og hva de tilbyr. Derfor ønsker vi å implementere teknologi som gjør produktet til en frys å ha, men samtidig som den ivaretar tilliten. Som for eksempel ved bruk av NFC-teknologien som vil gjøre åpne- og låse funksjonen effektiv, men samtidig sikker i form av at den kun er registrert på det brukeren ønsker å ha som nøkkel. Dette er medium prioritert.

**Kundens tillit:** Det er viktig å bevare det kunden ser på som viktig. Den teknologien vi bruker og hvordan låsen er bygd gjør at det er vanskelig for barn eller ivedkommende å komme seg inn til steder som brukeren ønsker låst. Dette viser til at vi tar barnas sikkerhet og brukerens behov på alvor, noe som øker foreldres eller andre brukeres tillit til produktet. Medium prioritert.

### 3.3.2 Risikoidentifisering (Hva kan gå galt som vi ønsker å forandre?)

- Produktet knekker (blir ødelagt/skadet)
- Produktet går tom for strøm
- Applikasjonen fungerer ikke
  - Tredjeparts-kontrakter avskrives (f.eks. komponenter eller materiell som benyttes slutter å produseres)
  - Eksterne enheter støtter ikke vår teknologi Sikkerhetsløsninger er ikke bra nok

Se tabell/figur Risikoidentifisering for videre informasjon. 3.2

Hendelse	Navn	Beskrivelse	Indikator	System/gruppe/lokasjon	Årsak
H1	Slitasje	Produktet knekker	Låsen fungerer optimalt	Hardware	- Misbruk - Produksjonsfeil
H2	Strøm	Produkt går tom for strøm	Elektroniske låsen fungerer ikke	Hardware	Strømløse batterier/glemt å lade batteri
H3	Appfeil	Software feil	- Ikke mulig å logge seg inn i appen - Låsen fungerer ikke via appen	Software	- Feil i kode - Servere nede - Feil ved oppdateringer - Bugs
H4	Produksjon	Slutter å produsere materiell	- Forvarsel - Mister tilgang på digitale verktøy - Miste lisenser	Software	- Økonomiglipp - Bedrifter er slått konkurs - Tjeneste endringer
H5	Enhet	Eksterne enheter støtter ikke vår teknologi	- Manglende tilgang	Software	- Manglende støttede enheter - For lite kunnskap om produkt

Figur 3.2: Risikoidentifisering

## Koble risiko med verdi

**Slitasje (H1):** Sikkerhet og sikring av tilgang. Dersom produktet blir ødelagt, kan det føre til at uvedkommende får tilgang.

**Strøm (H2):** Knyttet til funksjonalitet. Dersom produktet skulle gått tom for strøm, kan det føre til at produktet er ubruklig.

**Appfeil (H3):** Knyttet til funksjonalitet og personvern. Feil i appen kan hindre brukeren å ha tilgang, at data ikke blir lagret eller i verste fall kompromittert.

**Produksjon (H4):** Knyttet til sikring av tilgang. Dersom produksjonen stopper opp, kan det påvirke tilgjengeligheten av produktet og bedriftens økonomi.

**Enhet (H5):** Knyttet til brukervennlighet. Dersom eksterne enheter ikke støtter teknologien vi bruker, kan brukerne mislike produktet eller ikke ønske å bruke det som tiltenkt.

## Identifisering av risiko

For å identifisere potensielle risker knyttet til vårt produkt, har vi benyttet oss av en kombinasjon av brainstorming og analyse av data fra lignende produkter. Denne tilnærmingen har gitt oss en god bit med forståelse av mulige utfordringer og sårbarhet ved vårt produkt.

**Brainstorming:** Vi gjennomførte som team noen enkle brainstorming-økter i fellesskapet når denne oppgaven ble lagt frem. Vi er et team som har mye forskjellig kompetanse og forskjellig tankegang når det kommer til å identifisere mulige utfordringer ved vårt produkt og dens respektive oppgaver. Dette fremmet en kreativ tankegang i brainstormingen og grunnlaget for en vellykket risikovurdering. I disse øktene øktene fokuserte vi på følgende områder:

*Sikkerhet:* hvordan kan produktet misbrukes eller feile på en måte som kompromitterer sikkerheten til barn og uvedkommende?

*Funksjonalitet:* hvilke tekniske feil kan oppstå som hindrer produktet i å fungere som den forventes?

*Brukervennlighet:* hvilke brukerfeil kan oppstå, og hvordan kan det påvirke produktets effektivitet og bruksområde?

*Personvern:* hvordan kan brukerdata bli kompromittert, hvilke aspekter må beskyttes og hvilke tiltak kan vi iverksette for å forhindre dette?

**Analyse av data fra lignende produkter:** Vi satt oss ned og analyserte data og mye tilbakemeldinger fra andre lignende produkter og deres brukere. Dette inkluderte:

*Historiske feil:* Vi undersøkte hva slags vanlige feil og problemer det har blitt rapportert av brukerne til andre smarte låser. Det som ble notert ned av punkter her var batterilevetid, tilkoblingsproblemer og manglende tiltak dersom man har mistet tilgang.

*Brukeranmeldelser:* Vi analyserte en rekke anmeldelser og tilbakemeldinger fra brukere for å identifisere bekymringer eller andre risker.

*Bransjestandarder:* Vi har vurdert relevante bransjestandarder og beste praksis for å sikre oss at vårt produkt oppfyller disse kravene som stilles.

Ved å kombinere brainstorming-øktene og data fra lignende produkter, har vi vært i stand til å identifisere og prioritere de mest kritiske risker for vår smarte lås. Metoden vi brukte sikrer at vi både tar hensyn til teoretiske og praktiske aspekter ved risikoidentifisering.

### 3.3.3 Risikoanalyse og vurdering

For å analysere og vurdere risikoene, har vi benyttet en kombinasjon av kvalitative og kvantitative metoder. Vi har kategorisert risikoene basert på sannsynlighet og konsekvens, og deretter beregnet den samlede risikoen for hver hendelse. Dette er kategoriene for konsekvens og sannsynlighet vi har basert det på.

#### Konsekvens

- *Lav:* Minimal påvirkning, Lettere å rette opp enn andre ting
- *Medium:* Moderat / synlig påvirkning. Krever noen ressurser for å rette opp.
- *Høy:* Alvorlig påvirkning. Krever betydelige ressurser og tid for å rette opp.

#### Sannsynlighet

- *Lav:* lite sannsynlig at det skjer
- *Medium:* Moderat sannsynlighet for at det skjer
- *Høy:* Ganske sannsynlig for at det skjer

Hendelse	Kategori (KIT)	Konsekvens	Sannsynlighet	Risiko	Begrunnelse
H1	T	Medium	Lav	M	Produktet er laget av slitesterkt materiell, men kan fortsatt bli ødelagt/slitt i stykker ved misbruk eller produksjonsfeil
H2	T	Lav	Medium	L	Batteriene kan gå tomme, men indikatorer skal være på plass og manuell åpning av låsen er mulig
H3	K og T	Høy	Lav	H	Feil i appen kan hindre tilgang og kompromittere sikkerhet
H4	T og I	Lav	Lav	L	<del>Backup</del> -løsninger er på plass, og det er lite sannsynlig at en avtale går ut uten forvarsel
H5	T	Medium	Medium	M	Ikke alle enheter støtter teknologien enda, men de fleste gjør det

Figur 3.3: Risikoanalyse

### Risikoanalyse

Vi har analysert hver risiko ved å vurdere sannsynligheten og konsekvenser hvis det inntreffer. Dette gir oss en risikovurdering som hjelper oss med å prioritere hvilke risikør som krever tiltak. Tabell "Risikoanalyse" viser en oversikt av risikoanalysen vi har produsert, videre skal vi også begrunne vurderingene som er gjort. 3.3

**Slitasje (H1):** Selv om sannsynligheten er lav på grunn av slitesterkt materiale benyttet, kan konsekvensene være moderate hvis produktet blir ødelagt. Da kan det føre til uautorisert tilgang og kunder kan klage som kan føre til mindre tillit og andre konsekvenser.

**Strøm (H2):** Sannsynligheten er medium fordi batteriene kan gå tomme, men konsekvensene er lave siden manuell åpning er mulig. Vi vurderer av erfaring at oppsyn av batteriets levetid ikke følges opp nok og derfor settes sannsynligheten opp fra lav.

**Appfeil (H3):** Sannsynligheten er lav på grunn av grundig testing av utviklere, men konsekvensene er høye fordi feil kan hindre tilgang og kompromittere sikkerheten.

**Produksjon (H4):** Sannsynligheten er lav fordi vi har backup-løsninger klare, og det er kun avtaler med profesjonelle leverandører som vil gi ut forvarsel dersom det skulle oppstå noe utforventet. Konsekvensene er lave fordi vi har løsninger som gjør at produksjonen ikke bør stoppe dersom en leverandør ikke klarer å levere.

**Enhet (H5):** Sannsynligheten er medium fordi ikke alle enheter støtter teknologier som NFC-lesing eller feil plattform. Konsekvensene er da moderate fordi det kan påvirke brukervennligheten.

### 3.3.4 Risikohåndtering

For hver risiko har vi klart å identifisere tiltak for å redusere sannsynligheten og konsekvensene. Tabellen Risikohåndtering gir en beskrivelse av de ulike riskene. Videre vil vi også reflektere over hvilke risikør vi velger å gjøre noe med, samt hvilke risikør vi ikke ønsker å behandle. 3.4

Hendelse	Tiltak
Slitasje (H1)	Bruke slitesterkt materiale og tilby kundeservice for erstatning ved for eksempel produksjonsfeil
Strøm (H2)	Implementere varsler i appen for lavt batterinivå samt en manuell åpningsteknikk
Appfeil (H3)	Utføre grundige tester før utgivelse og ha en robust supportløsning med grundige utviklere
Produksjon (H4)	Ha backup-løsninger og overvåkning av leverandørens mulighet for å levere
Enhet (H5)	Utvide støtte til flere plattformer og enheter og tilby alternative metoder for å bruke låsen

Figur 3.4: Risikohåndtering

### Hvilke risker ønsker vi å behandle?

**Strøm (H2):** Løsningen vår er basert på at låsen ikke skal få direkte strømforsyning, da dette kan skape problemer ved monteringen av låsen. Vårt produkt skal få strøm av ladbare AA-batterier, noe som har en del fordeler, men som også har sine problemstillinger.

Dersom batteriet går tom for strøm, skal vi ha en mulig fysisk løsning for å manuelt lukke opp låsen. Løsningen går ut på at vi fysisk lirker låsen opp ved hjelp av et kort eller liknende. Låsen er konstruert på den måten at man kan enkelt åpne den uten strøm, men at det fortsatt er vanskelig å få den opp for et barn eller for noen som ikke vet hvordan den fungerer. Dette blir illustrert i produktinformasjonen med en forklaring på hvordan man skal åpne den når man har gått tom for strøm.

**Appfeil (3):** Følgende må være i orden for at appen ikke skal slutte å fungere: gode tester som feilsøker før den blir utgitt og gode support løsninger. Det kan oppstå flere feil i appen. Siden appen bruker Bluetooth til å kommunisere, så er det alltid en fare for at appen ikke klarer å koble til en spesifikk låsene, det eneste man får gjort her er å skrive en god kode og logge hvor i prosessen det gikk galt, så man enkelt kan finne en løsning på problemet gjennom en support-rolle. Andre feil som kan oppstå er generelle syntaks eller kode feil, disse burde ikke oppstå men det kan skje. På grunn av sånne feil, blir det satt opp tester for funksjonaliteten i appen for å sikre at feil ikke oppstår, oppstår en feil så må vi ta det via support hvor de kan sjekke hvor appen feilet.

**Produksjon (4):** Dersom 3. parts tjenester/materiell blir utilgjengelige er vi nødt til å sørge for at vi har en backup løsning. Tjenester / materiell leveres også kun av profesjonelle, som må holde sin del av avtalen. Dette inkluderer også forvarsler om noe ikke vil bli levert lenger.

### Hvilke risker ønsker vi ikke å behandle?

**Slitasje (1):** Vi velger ikke å gjøre noe med slitasje, fordi vi allerede har valgt ut slitesterkt materiale som vi gjør nødvendige tester på. Dersom vi hadde produsert med enda sterkere materiell, ville kostnaden for oss og kunden vært høyere. (Beskrevet nøy under kapittelet Design")

**Enhet (5):** Vi har ikke ressurser til å støtte alle enheter, men vi strekker oss til å dekke de aller fleste. Dette kommer an på at kunden har enheter som støtter vår teknologi. Vi ønsker ikke å gjøre noe med dette da det finnes andre metoder å benytte låsen på. Et grovt eksempel er at det fortsatt vil være mange som ikke vil bytte ut sin eldre iPhone som ikke støtter den nye NFC-teknologien.

### 3.4 Referanser relevant for verdivurdering

For at vi kan sikre at vår risikovurdering og håndtering er i tråd med beste praksis og relevante standarder satt i samfunnet, har vi benyttet oss av følgende referanser:

**ISO 31000:** Standard for risikostyring som gir retningslinjer og prinsipper for risikohåndtering.

**EN 71:** Standard for sikkerhet av leker, relevant for produkter som brukes av barn.

**GDPR:** Personvernsnorordning som regulerer behandlingen av personopplysninger innenfor EU/EØS.

## **Kapittel 4**

# **Design / Frontend**

I dette kapittelet skal bedriften gjøre rede for beslutninger og arbeid som en bruker kan ta og se på. Vi vil beskrive låsemekanismen vi benytter oss av, materialvalg og oppbygging, forklaring av logoen, strømforsyning, fremgangsmåte og tesing av forskjellige materialer. Det kommer også til å refereres til figur/bilde av logo, skisser og andre relevante objekter.

### **4.1 Låsemekanismen**

Gruppen har satt som mål å utvikle en lås som kombinerer estetikk, brukervennlighet og bærekraft. En viktig del av dette målet er utviklingen av vår egen logo, TapNest", som symboliserer produktets verdier – enkelhet, trygghet og brukervennlighet. Låsen er designet med tanke på enkel montering og plassering, slik at den ikke er synlig når skapet er låst. Produktet vil også inkludere NFC-funksjonalitet som vil være plassert i bunnen av produktet, slik at brukeren enkelt kan tappe skapet med et NFC-kort eller brikke for å åpne skapet. Låsemekanismen er inspirert av forskning ved BYU CMR (Brigham Young University Compliant Mechanisms Research), som spesialiserer seg på bruk av fleksible materialer.

Materialvalget bygger på ideen om å bruke 3D-printing med resirkulerbare materialer. Låsen skal kunne monteres på innsiden av skap og skuffer ved bruk av skruer eller dobbeltsidig tape. Dette gir brukeren fleksibilitet til å flytte produktet mellom ulike møbler, noe som ikke bare gjør produktet brukervennlig, men også mer bærekraftig ved å tillate gjenbruk. Boksen har dimensjoner på 10 x 14 x 2 cm, som tilsvarer en liten notatbok. Denne størrelsen er valgt for å gi nok plass til mekanismen, samtidig som det er rom for å gjøre produktet mer kompakt i fremtiden. Se figur Låsemekanisme" i neste seksjon.

Selve låsemekanismen er basert på compliant mechanisms, eller fleksible mekanismer, som oppnår bevegelse gjennom bøyelige materialer i stedet for tradisjonelle bevegelige deler som hengsler og ledd. Dette gir flere fordeler, inkludert enklere produksjon og redusert risiko for slitasje på bevegelige deler. Hele mekanismen kan 3D-printes som en sammenhengende enhet, noe som forenkler produksjonen og gjør den mer kostnadseffektiv. Logoen TapNest" understreker også denne enkelheten i design og produksjon.

Mekanismen fungerer som en bryter med to posisjoner: åpen og lukket. En magnet er festet til den ene enden av bryteren, mens selve låsedelen er plassert på den andre enden. Bevegelsen av bryteren styres av en magnetcoil som består av en kobberspole. Når det tilføres strøm til coilen, skapes et magnetfelt som enten tiltrekker eller frastøter magneten, avhengig av strømretningen, og aktiverer dermed bryteren mellom de to posisjonene. Se figur "Tidlig prototype" i neste seksjon.

## 4.2 Materialvalg og oppbygging

Produktet er planlagt fremstilt ved bruk av 3D-printede resirkulerte materialer for å redusere miljøpåvirkningen og sikre en effektiv produksjonsprosess. Flere plastmaterialer ble testet under utviklingen, inkludert PETG, PLA, FLEX, og nylon, men ingen av de tilgjengelige materialene oppfylte alle kravene til funksjonalitet og holdbarhet.

**PETG og PLA:** Disse materialene fungerte tilfredsstillende for å demonstrere bryterens grunnleggende funksjonalitet, men hadde ikke tilstrekkelig holdbarhet for langvarig bruk.  
4.3

**FLEX:** Dette materialet var for seigt og mistet stabilitet, noe som gjorde at mekanismen ikke fungerte effektivt. 4.4

**Nylon:** Nylon ble valgt på grunn av sin styrke og slitestyrke, men viste seg å være for stift for denne spesifikke mekanismen. 4.5

Videre arbeid vil fokusere på å finne et mer passende materiale eller en kombinasjon av materialer som kan balansere fleksibilitet og holdbarhet bedre. Logoen vår, "TapNest", er et symbol på denne balansegangen – enkelhet, pålitelighet og bærekraft.

### 4.2.1 Fordeler med compliant mechanisms

Designet av låsemekanismen med compliant mechanisms gir flere fordeler:

**Få bevegelige deler:** Bevegelse oppnås gjennom bøyning av materialet, noe som reduserer antallet kontaktflater som kan slites ut og øker levetiden til produktet.

**Motstand mot miljømessige faktorer:** Mekanismer uten tradisjonelle ledd er mindre utsatt for støv, smuss og korrosjon, noe som øker påliteligheten i ulike miljøer.

Dette bidrar til å gjøre produktet enklere å produsere, mer kostnadseffektivt, og bedre tilpasset bærekraftige løsninger. Logoen 'TapNest' gjenspeiler også vår forpliktelse til enkle og effektive løsninger som er både pålitelige og bærekraftige.

## 4.3 Bruk av batterier

Produktet vil benytte AA-batterier for strømforsyning. Dette gir en praktisk og lett tilgjengelig løsning for brukerne, som kan bytte batterier ved behov uten spesielle verktøy eller tilpasninger. AA-batteriene er valgt for å understreke produktets brukervennlighet og tilgjengelighet, i tråd med verdiene som er representert i logoen 'TapNest'.

## 4.4 Fremgangsmåte

I utviklingsprosessen har gruppen gjennomgått flere iterasjoner av designet for å optimalisere låsemekanismen. Den første fasen besto av å definere kriteriene som låsen skulle oppfylle. Det ble lagt vekt på at produktet skulle kunne produseres ved bruk av resirkulerte materialer, være brukervennlig, og kunne forblie enten åpen eller lukket posisjon i henhold til brukerens behov.

I de første skissene besto låsemekanismen av flere deler, inkludert en ball som holdt skapet låst. Etter en del tester ble det besluttet å fjerne ballen for å gjøre mekanismen enklere og redusere antallet deler. Denne endringen gjorde at produktet ble enklere å produsere, samtidig som det fortsatt oppfylte alle funksjonelle krav. Logoen TapNest ble integrert i denne prosessen for å sikre at alle designelementer, inkludert brukervennlighet og sikkerhet, ble ivaretatt.

For modellering ble Fusion 360 brukt, et verktøy som gjorde det enkelt å lage designene som senere kunne 3D-printes. Ulike prototyper ble laget, og tester ble utført med forskjellige materialer for å vurdere egnetheten til mekanismen.

## 4.5 Skisser / Prototyper

I denne seksjonen legger vi ved bilder/figurer som er relevante for dette kapittelet, og inneholder skisser og bilder av prototypen, og andre relevante bilder. Hvert bilde blir forklart i begynnelsen av seksjonen og bildene kommer deretter.

**Låsemekanisme:** Disse to bildene viser hvordan låsemekanismen skal fungere. I tilstand: åpen og lukket. 4.1

**Tidlig prototype:** Disse to bildene representerer hvordan produktet kan se ut og er en tidlig prototype. 4.2

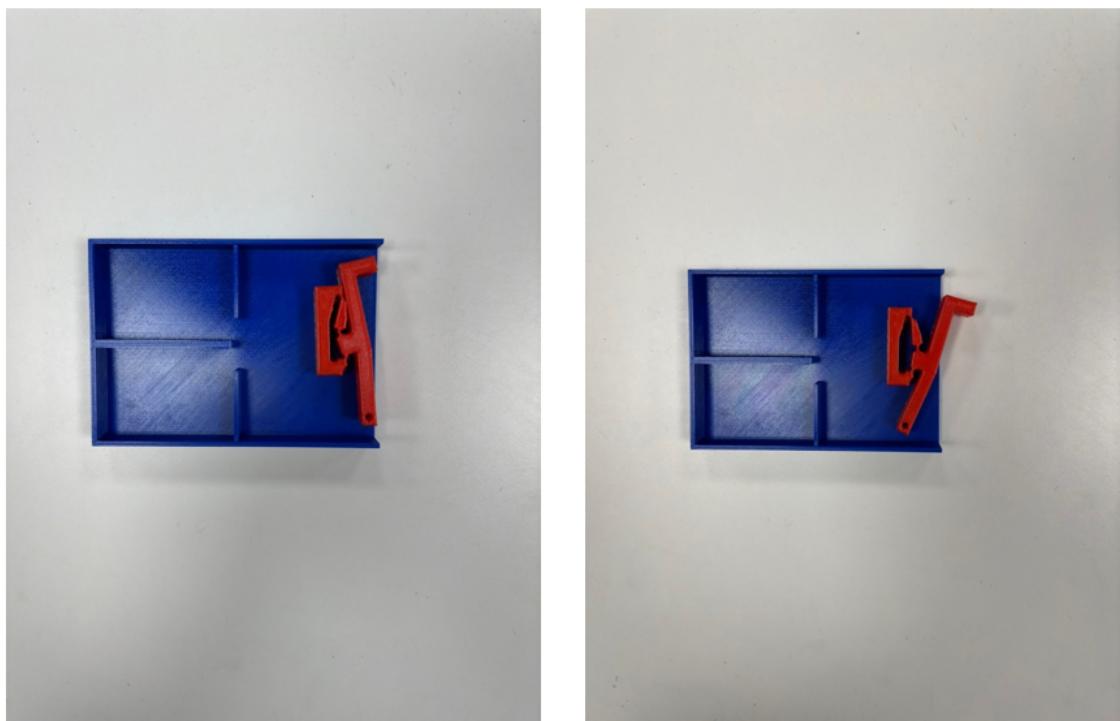
**PETG og PLA:** Første printen av mekanismen er printet i både petg og pla (fargen er lik på de to ulike filamenta). Modellen knakk etter et par ganger med testing og bruk. 4.3

**FLEX:** Den neste modellen ble printet i var flex filament. Et filament som er veldig flexibelt. Dette viste seg til å være flekisbelt og mekanismen funka ikke som planlagt. 4.4

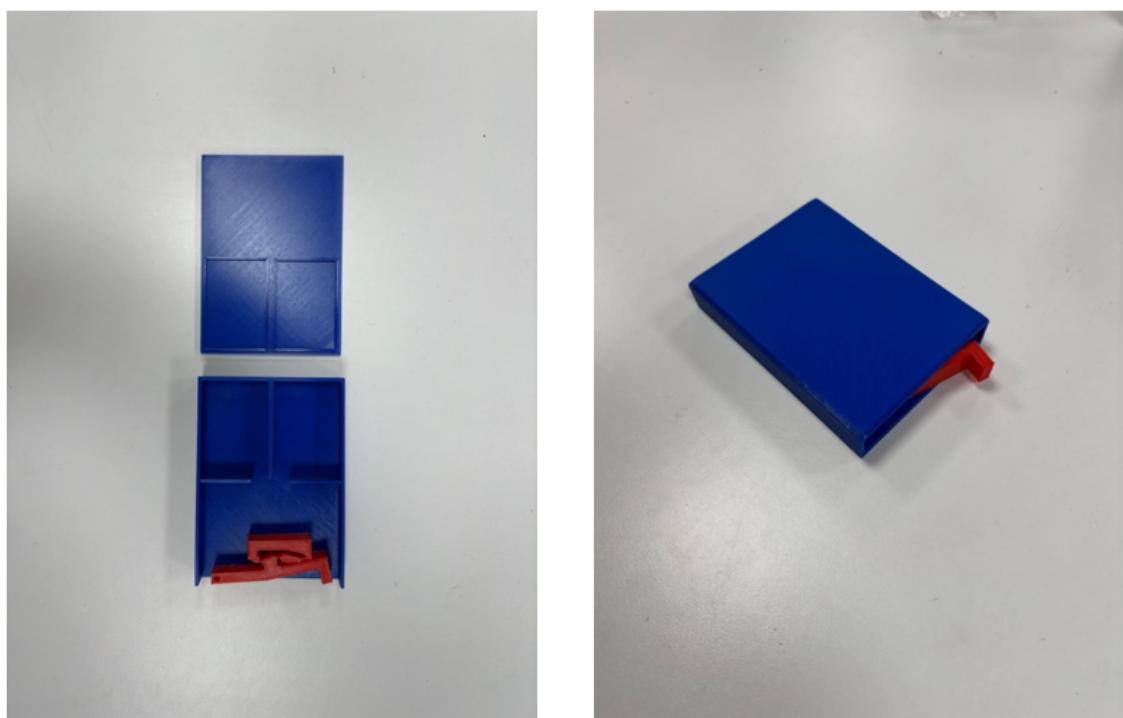
**Nylon:** Dette var det siste filamentet vi prøvde ut. Det gjekk heller ikke som forventet, filamentet er svært slitesterkt, men det var for sterkt til at mekanismen ikke bøyde seg. 4.5

**Skisse1:** Skisse av produktet med mekanismen inni og en ide og forklaring hvor de ulike komponentene skal ligge. 4.6

**Skisse2:** Skisse av selve produktet uten mekanismen inni. 4.7



Figur 4.1: Låsemekanisme



Figur 4.2: Tidlig prototype



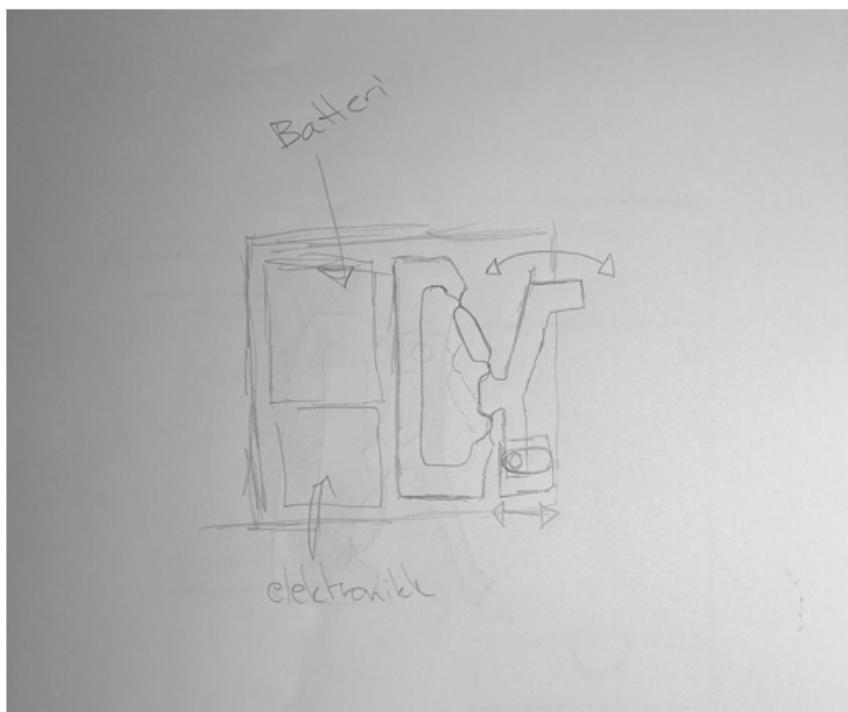
Figur 4.3: PETG og PLA



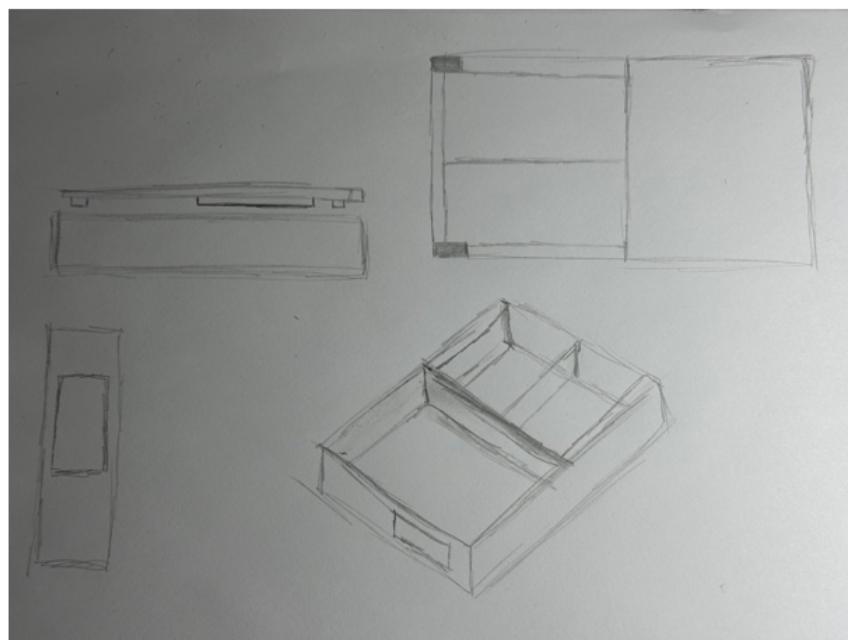
Figur 4.4: Flex



Figur 4.5: Nylon



Figur 4.6: Skisse1



Figur 4.7: Skisse2

# Kapittel 5

## Implementasjon / Gjennomføring / Backend

Dette kapittelet vil omhandle alt av implementasjon av kode og alt annet tekniske sammenhenger, vi vil inkludere arkitektur, avgjørelser, kravspesifikasjon og valg tatt etterhvert.

### 5.1 Kravspesifikasjon

**Formål:** Målet med prosjektet er å utvikle en smartlås som kan brukes på skap (som medisinskap eller vitrineskap) for å beskytte barn mot tilgang. I tillegg skal en mobilapplikasjon gjøre det mulig å fjernstyre låsen.

**Målgruppe:** Primært familier med barn men også inkludert små bedrifter og privatpersoner som ønsker å sikre objekter i skap/skuffer i hjemmet sitt.

#### 5.1.1 Funksjonelle krav

##### Lås:

- Låsen skal kunne monteres på ulike typer skap (f.eks. medisinskap, vitrineskap).
- Skal kunne låses og låses opp ved skanning av en autorisert NFC-tag.
- Låsen skal varsle appen ved lavt batterinivå.
- Skal ha et mekanisk nødopplåsingsalternativ (som nøkkel) for å håndtere teknisk svikt.
- Låsen skal motta signal fra appen for å aktivere lås-/opplåsningsprosessen når NFC har blitt skannet.

**Applikasjon:**

- Appen skal vise en liste over alle låsene som er registrert til brukeren.
- Appen skal vise batteristatus for hver lås.
- Skal kunne hente og vise informasjon fra låsen via Bluetooth, inkludert tilkoblingsstatus.
- Brukeren skal kunne legge til, endre og fjerne NFC-tagger som er tilknyttet låsene.
- Skal gi varsler ved forsøk på uautorisert tilgang.
- Skal tilby en enkel aktivitetslogg over låse- og opplåsningshendelser.
- Brukeren skal kunne konfigurere enkle innstillinger for låsene via appen.

**Database:**

- Systemet skal benytte en database for å lagre:

- Informasjon om brukere eller andre roller
- Data om låser, inkludert unik ID, tilstand (åpen/lukket), batteristatus, og tilhørende bruker.
- Informasjon om NFC-tagger, inkludert tag-ID og dens tilhørende lås
- Aktivitetslogger for alle lås og opplåsningshendelser.
- Databasen skal være strukturert for å:
  - Sikre rask og effektiv spørring.
  - Beskytte brukerdata i henhold til relevante personvernregler (GDPR).
  - Redusere redundans ved bruk av normalisering (opptil BCNF).

- Skal kunne støtte flere låser og nøkler

**5.1.2 Ikke-funksjonelle krav:****Sikkerhet:**

- Kommunikasjonen mellom låsen og appen skal være kryptert for å sikre data mot uautorisert tilgang.

**Brukervennlighet:**

- Appen skal ha et enkelt og intuitivt brukergrensesnitt for rask navigering.
- Installasjonen av låsen skal være enkel, med en kort og tydelig veiledning.

**Skalerbarhet:**

- Systemet skal støtte administrasjon av opptil 5 låser og 10 NFC-tagger per bruker i MVP-fasen, med mulighet for å utvide senere.

**Ytelse:**

- Låsen skal reagere på NFC-skanning innen 1 sekund.
- Appen skal kunne oppdatere informasjon fra låsen i nær sanntid.

**Begrensninger:**

- Låsen vil i denne fasen være optimalisert for innendørs bruk under normale romtemperaturer og er ikke designet for ekstreme værforhold.
- Appen vil være tilgjengelig for Android-enheter i første omgang, med planlagt utvidelse til iOS i fremtiden.

**Refleksjon:**

Denne kravspesifikasjonen viser kjernefunksjonalitet som en del av utviklingen av en Minimum Viable Product (MVP). Fokus har vært på å utvikle grunnleggende funksjoner for lås og app, inkludert opplåsingmekanismer, NFC-integrasjon, kontrollsenter (database) og grunnleggende administrasjonsmuligheter i appen.

## 5.2 Avhengigheter og arkitektur

Her vil det gjøres rede for avhengigheter brukt i prosjektet, og informasjon om ulike verktøy for utvikling av applikasjon, samt forklaring av arkitektur for applikasjon.

### 5.2.1 Prosjektets avhengigheter

Kravspesifikasjonene for prosjektet inkluderer behovet for Bluetooth-sammenkobling mellom appen og låsen, samt NFC-lesing. Dette medfører en rekke avhengigheter og biblioteker som er nødvendige for at appkoden skal fungere som tiltenkt. Applikasjonen benytter flere av Androids egne biblioteker for å aktivere, koble til og overføre data via Bluetooth. I tillegg importeres enkelte biblioteker for å tilføre funksjonaliteter som ikke nødvendigvis inngår i standard Android Studio. For

å utnytte Bluetooth-funksjonalitetene, er det også nødvendig å ha avhengigheter som gir tillatelse til å bruke de innebygde rammeverkene i Android-enheten når appen er i bruk.

Testingen av koden krever også noen avhengigheter, for å teste koden så kommer det frem at det er nødvendig med avhengighet mot Junit og Mockito, samt er det nødvendig med en avhengighet for Robolectric for å kunne kjøre testene på JVM uten og faktisk ha en telefon eller en emulator for å kjøre appen.

## **Android tools for app utvikling**

Her ligger det en oversikt over de ulike verktøyene og komponentene som brukes i utviklingen av Android-appen, med fokus på Bluetooth-funksjonalitet.

**Bluetooth Tools:** Verktøy som gjør det mulig for Bluetooth-funksjoner i appen, for eksempel tilkobling til enheter og aktivering av Bluetooth.

*BluetoothDevice*: Representerer en Bluetooth-enhet og brukes til å opprette forbindelser.

*BluetoothSocket*: Brukes for dataoverføring mellom Bluetooth-enheten og en annen enhet, som i vårt tilfelle er en Raspberry Pi som sekundærenhet.

*BroadcastReceiver*: Denne avhengigheten brukes i BluetoothPairing-klassen for å sjekke tilkoblingsstatus mellom Bluetooth-enheten og vår sekundærenhet (Raspberry Pi). BroadcastReceiver holder oversikt over statusendringer i Bluetooth-tilkoblingen, som når enheten ikke klarer å koble seg på eller når tilkoblingen er opprettet.

*Intent* og *IntentFilter*: Brukes til å håndtere hendelser knyttet til statusendringer i Bluetooth-tilkoblingen.

## **Permissions og ActivityCompat:**

*ActivityCompat* og *Manifest.permission.BLUETOOTH\_CONNECT*: I koden brukes ActivityCompat før hver Bluetooth-implementasjon for å sikre at appen har tillatelse til å bruke Bluetooth. ActivityCompat og *Manifest.permission.BLUETOOTH\_CONNECT* brukes til å håndtere og verifisere Bluetooth-tillatelser.

## **Testing med JUnit, Mockito og Robolectric:**

*JUnit*: Brukes for enhets- og integrasjonstesting av klasser og metoder, slik som *BluetoothService* og *BluetoothPairing*.

*Mockito*: Brukes for å opprette falske objekter som kan testes uten behov for faktiske Bluetooth-enheter.

*Robolectric*: Brukes til å kjøre Android-testene uten en fysisk enhet, og muliggjør testing av Android-komponenter som *Context* og *BroadcastReceiver*.

**Dataoverføring med `java.io.OutputStream` og `java.io.InputStream`:** Disse klassene muliggjør sending og mottak av data over Bluetooth-tilkoblingen.

*BluetoothService-klassen:* bruker InputStream og OutputStream til å lese og skrive data.

- InputStream leser data fra en enhet tilkoblet Bluetooth, for eksempel statusen til en lås.
- OutputStream brukes til å sende data til en enhet ved hjelp av metoden write i BluetoothService.

Disse funksjonene er kritiske for Bluetooth-tilkoblingen, ettersom de gir grunnleggende funksjonalitet.

**Bruk av Java Collections (f.eks. Map og HashMap):** Java.Util gir map-funksjonalitet, som brukes for å håndtere låseobjektene i appen.

*Map og HashMap:* Bruker lockId som nøkkel for å holde oversikt over låsobjekter. HashMap brukes til å oppdatere eller legge til nye låser i locksMap basert på tilgjengelig data.

### 5.2.2 Arkitektur

Applikasjonen følger ikke en spesifikk struktur helt og holdent, men er inspirert av heksagonalarkitektur og implementerer mange prinsipper for løst koblet kode, men passer best inn som en modulær arkitektur hvor en klasse har et bestemt ansvars område. Den er designet for å minimere avhengighetene mellom klassene. For eksempel vil klassene BluetoothService"og BluetoothPairing" håndtere all Bluetooth-funksjonalitet, slik at ikke alle klassene må være avhengige av Bluetooth. Valget av denne arkitekturen er bevisst; den tillater at ulike funksjoner ikke er helt avhengige av hverandre. Målet er at selv om NFC-lesingen skulle slutte å fungere, skal brukeren fortsatt kunne låse opp og igjen låsen kun ved hjelp av Bluetooth-sammenkoblingen.

## 5.3 Maskin-implementasjon

I denne seksjonen av kapittelet vil vi vise til hvilke tekniske plattformer vi har benyttet oss av i prosjektet, dette inkluderer valg av database, programmeringsspråk, presentasjon av app og fysiske løsninger.

### 5.3.1 Database

I vårt prosjekt har vi valgt å utvikle en database som en essensiell del av låssystemet vårt. Hovedformålet med databasen er å håndtere, organisere og beskytte informasjon om brukerne, tilgangshendelser og eierskapet av låser på en sikker og effektiv måte. Databasen gjør det mulig å lagre kritisk informasjon i et system som gir rask tilgang, samtidig som at man kan sikre personvern og datasikkerhet.

## Systemfunksjonalitet

Låssystemet er bygd opp med en strukturert database som samhandler med en mobilapp og en elektronisk lås. Appen skal fungere som et kontrollsenter for både brukere og administratorer, mens databasen lagrer, logger og håndterer viktig informasjon. Dette er informasjon som er knyttet til brukere, låser, eierskap og tilgangslogger. Denne strukturen sikrer enkel styring med hensyn på datasikkerhet og brukervennlighet.

**Kommunikasjon mellom app og database:** Når en bruker prøver å åpne en lås gjennom appen, så sender appen en forespørsel til databasen for å verifisere at brukeren har tilgang. Appen sender brukerens ID og låsens ID til databasen som sjekker om det finnes rettigheter registrert mellom de to. Når dette bekreftes eller avkreftes så sendes tillatelsen tilbake til appen.

**Tilgangskontroll og eierskap:** Eierskap og brukerens roller er strukturert gjennom databasen. Dette slik at hver bruker kan tildeles rollen som enten 'eier' eller 'bruker' av låser. På denne måten kan man enkelt håndtere tilganger på en sikker måte. For eksempel en administrator har tilgang til flere system funksjonaliteter enn vanlige brukere. Dette kontrollen og opprettholder sikkerheten ved å begrense tilgang til sensitive funksjoner.

**Loggføring av hendelser:** Hver gang noen prøver å låse opp eller låse en enhet, registreres denne handlingen i en tilgangslogg i databasen. Dette inkluderer også tidspunkter, brukerens ID, låsens ID og handlings typen (låse eller låse opp). Dette er med for å bidra til at man alltid har en oversikt over hvem som har hatt tilgang til hvilken lås og når det skjer. Noe som kan gi bedre sikkerhet i forhold til å identifisere feil eller uautoriserte hendelser.

**Datahåndtering og sikkerhet:** Siden personvern og sikkerhet er viktig, følger databasen også strenge retningslinjer. Databasen minimerer data ved å kun lagre nødvendige opplysninger om brukere og låser, og brukerinformasjonen deles ikke utenom definerte eierskap. Tilgangen styres også strengt gjennom fremmednøkler, så kun autoriserte brukere kan endre spesifikke felt.

## Planlegging og fokusområder

- **Funksjonalitet og brukerdata**

- **Brukere:** Data over de som administrerer og bruker låsen (eiere, administratorer)
- **Låser:** Hvilke låser er det som er tilgjengelige, tilhørende steder plassert, serienummer og status (åpen eller lukket)
- **Adgangs-logg:** Historikk for å spore hvem som låser/åpner og når det skjer
- **Brukerroller:** Tilgangsnivåer. Vanlig bruker, administrator eller kanskje vedlikeholdspersonell
- **Sessions:** Tidspunkter for inn- og utlogginger. Slik at systemet kan kreve autoriseringer før bruk

- **Personvern og sikkerhet**

- **Minimere data:** Lagre kun nødvendig informasjon, f.eks. initialer eller pseudonymer istedenfor fulle navn der mulig.
- **Kryptering av sensitive data:** Krypter passord, personlige identifikatorer, og annen sensitiv informasjon.
- **Datatilgang:** Kun autoriserte personer skal ha tilgang til sensitive deler av systemet.
- **Logging av datatilgang:** Registrer hvem som har tilgang til dataene og når.

For å være i samsvar med GDPR, må vi tenke på:

- **Brukersamtykke:** Før vi lagrer sensitive data om brukeren.
- **Data-minimering:** Kun lagre det som er strengt nødvendig.
- **Tilgangskontroll:** Begrense hvem som kan se hvilke data.
- **Sletting av data:** Implementer en mekanisme for å slette brukerdata på forespørsel.

**Lock:**

User_ID	User_Name	User_Email	Lock_ID	Lock_Location	Access_Time	Access_Action	Role
1	Ola	ola@gmail.com	1001	Living room	2024-10-10 10:00	Unlock	User

Figur 5.1: Lock Database

**Brainstorming / Basere databasen på informasjon****• Bruker:**

- Hvem er det som har tilgang til systemet og hva slags rolle brukerne har.
- **Bruker-ID**
- **Navn**
- **E-post**
- **Rolle** (Administrator, Bruker, etc.)

**• Lås:**

- Hver fysiske lås som skal brukes.
- **Lås-ID**
- **Plassering** (hvor låsen befinner seg)
- **Status** (låst eller ulåst)

**• Adgangslogg (Access Log):**

- Registrer over låser som brukes og hvem som brukte låsen.
- **Bruker-ID** (hvem som brukte låsen)
- **Lås-ID** (hvilken lås som ble brukt)
- **Tidspunkt** (når det skjedde)
- **Handling** (låse opp eller låse)

**Samler funnet informasjon i tabeller**

Setter opp med en rad dummy data for å enklere se dens tilknytning til andre tabeller. (Se tabell for Lock Database, 5.1)

- I denne tabellen ser man at det kan oppstå redundans i flere kolonner, slik som **User\_Name**, **User\_Email** og **Lock\_Location**.
- Det vil være vanskelig å oppdatere en brukers e-post dersom brukeren får en ny.
- Hvis vi skal slette loggen, vil det fjerne informasjon som vi ikke nødvendigvis ønsker å tape.

## Normaliseringsprosess

**1. Normalform:** Sørger for at alle kolonner inneholder atomære verdier. Altså ingen lister eller flere verdier i en celle. Må også ha en unik primærnøkkel for hver rad. Kan se på tabellen over at den er allerede på første normalform. Dette er fordi vi har ingen atomære verdier, ingen sammensatte verdier og vi har en unik rad hver gang basert på tilgangshendelse. Altså "Access time".

**2. Normalform:** Går deretter videre til andre normalform. For å oppnå andre normalform så må vi fjerne partielle avhengigheter. Altså at alle ikke-nøkkelattributter som for eksempel 'User Name', kan ikke avhenge av bare deler av primærnøkkelen. Her er det nødvendig å bryte opp den store tabellen til mindre tabeller. (Se tabeller for Users, Locks og Lock Ownership, 5.3, 5.4, 5.5)

**3. Normalform:** For å oppnå tredje normalform må transitive avhengigheter fjernes. Oppretter tabellen "Access Log" for å skille logg-dataen. Siden tid og handlinger ikke er direkte avhengige av brukeren eller selve låsen.

**Access\_Log:**

User_ID	Lock_ID	Access_Time - PK	Access_Action
1	1001	2024-10-10 10:00	Unlock

Figur 5.2: Access Log, Fremmednøkler i user id og lock id som refererer til hverandre i dens tabell.

**BNCF (BoyceCodd Normal Form):** For at tabellen skal være i **3NF** (Tredje Normalform) og at alle determinanter må være supernøkler, må vi sørge for at det ikke finnes noen avhengigheter der en ikke-supernøkkel bestemmer verdien til andre attributter.

Tabellene kan struktureres som følger:

- **User** - holder informasjonen om brukeren med en primærnøkkel **User\_ID**
- **Locks** - holder informasjon om låser, med primærnøkkel **Lock\_ID**
- **Lock\_ownership** - er til for å knytte brukere med deres låser, med sammensatt primærnøkkel **User\_ID** og **Lock\_ID**
- **Access\_Log** - Logger hendelsene med primærnøkkel **Access\_Timestamp** og refererer til **User\_ID** og **Lock\_ID**

Med denne tabell-strukturen sørger vi for at alle normaliseringsformene er oppnådd. Oppsettet sørger også for en sikker struktur, fri for redundans og med effektivt bruk av minimalt med informasjon.

#### **Tabellstruktur:**

- **Users** (User ID, User Name, User Email, Role)
- **Locks** (Lock ID, Lock Location)
- **Lock Ownership** (User ID\*, Lock ID\*, Ownership)
- **Access Log** (Access Timestamp, User ID\*, Lock ID\*, Access Action)

**Users:**

User_ID - PK	User_Name	User_Email - Unik	Role
1	Ola	ola@gmail.com	User

Figur 5.3: Users tabell

**Locks:**

Lock_ID - PK	Lock_Location
1001	Living room

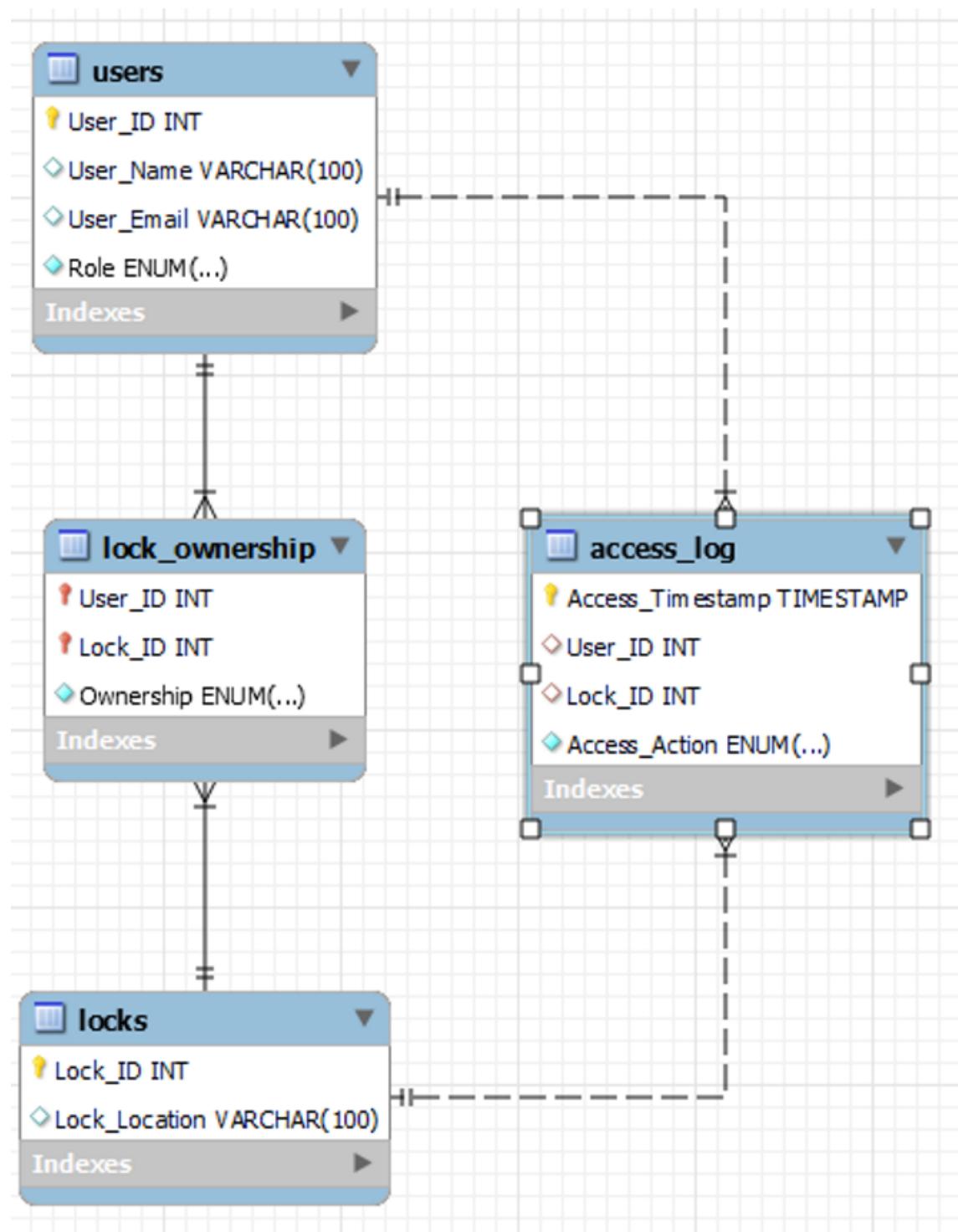
Figur 5.4: Locks tabell, Kan eventuelt legge til serienummer eller modell.

**ER-diagram:** ER-diagrammet vårt viser hvordan dataene våres er organisert i databasen, hvordan de er koblet sammen og deres attributter. 5.6

**Lock\_Ownership:**

User_ID	Lock_ID	Ownership
1	1001	Owner

Figur 5.5: Lock Ownership tabell, Sammensatt PK av user id og lock id og fremmednøkkel på begge som refererer til hverandre - skaper et mange til mange forhold mellom brukere og låser.



Figur 5.6: ER-diagram

### 5.3.2 Raspberry PI

For å administrere alle perspektiver av prototypen er det aktuelt å bruke en Raspberry Pi. Dette er et ekstremt bra verktøy til å vise det helhetlige bildet av de mest grunnleggende og de viktigste funksjonalitetene vi ønsker å ha i sluttpunktet.

#### Fordeler med Raspberry PI

**Kostnadseffektiv:** Raspberry Pi er relativt billig sammenlignet med mange andre utviklingsplattformer. En Raspberry Pi 3, kan kjøpes for noen hundrelapper, noe som gjør det til en økonomisk løsning for prototypen. Dette passer godt med våre budsjetttrammer for prosjektet.

**Allsidighet:** Raspberry Pi kan brukes til et bredt spekter av oppgaver. Den støtter mange forskjellige operativsystemer og kan kjøre flere programmeringsspråk, inkludert Python, Java, C++, osv. Dette gjør den ideell for et prosjekt som involverer kommunikasjon mellom flere enheter som RFID, Bluetooth, og en app eller nettside.

**Støtte for IoT og eksterne enheter:** Raspberry Pi har gode muligheter for å koble til sensorer og eksterne enheter via GPIO-pinnene (General Purpose Input/Output). Dette gjør det enkelt å integrere RFID-lesere, Bluetooth-moduler og andre sensorer i systemet. Den har innebygd støtte for Bluetooth og Wi-Fi, noe som er perfekt for prosjektet, som trenger trådløs kommunikasjon med låser og apper.

**Stor utviklerstøtte og dokumentasjon:** Det finnes en massiv mengde dokumentasjon, veiledninger, og fellesskap rundt Raspberry Pi. Hvis vi møter på problemer, er det svært sannsynlig at noen andre har opplevd det samme og funnet en løsning. Dette kan spare tid i utviklingen.

**Enkel fjernstyring** Raspberry Pi kan enkelt administreres over nettverket ved bruk av SSH eller VNC, slik at teamet kan jobbe på den fra forskjellige steder uten å være fysisk tilstede.

**Energi- og plassbesparende** Raspberry Pi trekker lite strøm og er kompakt, noe som kan være viktig i prototypingfasen og i senere utviklingsfaser hvor lavt strømforbruk og liten størrelse kan være kritiske faktorer.

#### 5.3.3 Ulemper med Raspberry Pi

**Begrenset ytelse:** Selv om Raspberry Pi er allsidig, er den fortsatt en lav-ytelses enhet sammenlignet med dedikerte utviklings-kort eller mikrokontrollere for spesifikke oppgaver. Hvis systemet vårt hadde krevd tung databehandling eller svært lav latentstid (f.eks. i kritiske sikkerhetssystemer), kan Raspberry Pi vise seg å være for treg. Når flere tjenester som Bluetooth, Wi-Fi, og en database kjører samtidig, kan det påvirke ytelsen negativt, spesielt på eldre Raspberry Pi-modeller.

**Begrenset holdbarhet i produksjonsmiljøer:** Raspberry Pi er primært designet for prototyping og utdanning, ikke nødvendigvis for robusthet og langsiktig bruk i industrielle eller sikkerhetskritiske applikasjoner.

**Kompleksitet med GPIO og maskinvare:** Selv om Raspberry Pi har GPIO-pinner som gjør det mulig å koble til eksterne enheter, kan det være utfordrende å oppnå stabil og nøyaktig kommunikasjon, spesielt når dere kombinerer flere enheter som RFID-lesere, Bluetooth og kanskje også sensorer. Integrasjon med ulike moduler kan kreve ekstra tid til debugging og testing.

**Sikkerhetsproblemer** Raspberry Pi kjører et vanlig operativsystem (som Linux), noe som gjør det mer utsatt for sikkerhetstrusler sammenlignet med dedikerte mikrokontrollere som har færre funksjoner og derfor mindre angreps-overflate.

**Strømavbrudd og lagring** Raspberry Pi er avhengig av SD-kort for lagring, som kan være utsatt for korrasjon ved strømavbrudd. Dette kan være et problem i en adgangskontroll-sammenheng der systemet alltid må være stabilt og operativt.

#### 5.3.4 Oppsett av Raspberry Pi

1. Først var vi nødt til å laste ned Raspberry Pi OS på Raspberry Pi sine sider (<https://www.raspberrypi.com/software/>).
2. Deretter kunne det lagres direkte på vår Raspberry Pi.
3. For Remote controll så setter vi opp en VNC på R.Pi.
4. Laster ned programvaren som gjør at vi kan styre denne eksternt (<https://www.realvnc.com/en/connect/download/viewer/>).

**VNC adresse:** 192.168.0.77

**Signatur:** 03-6d-38-42-a6-d7-05-aa

**'Raspberrypi (VNC Server)'**

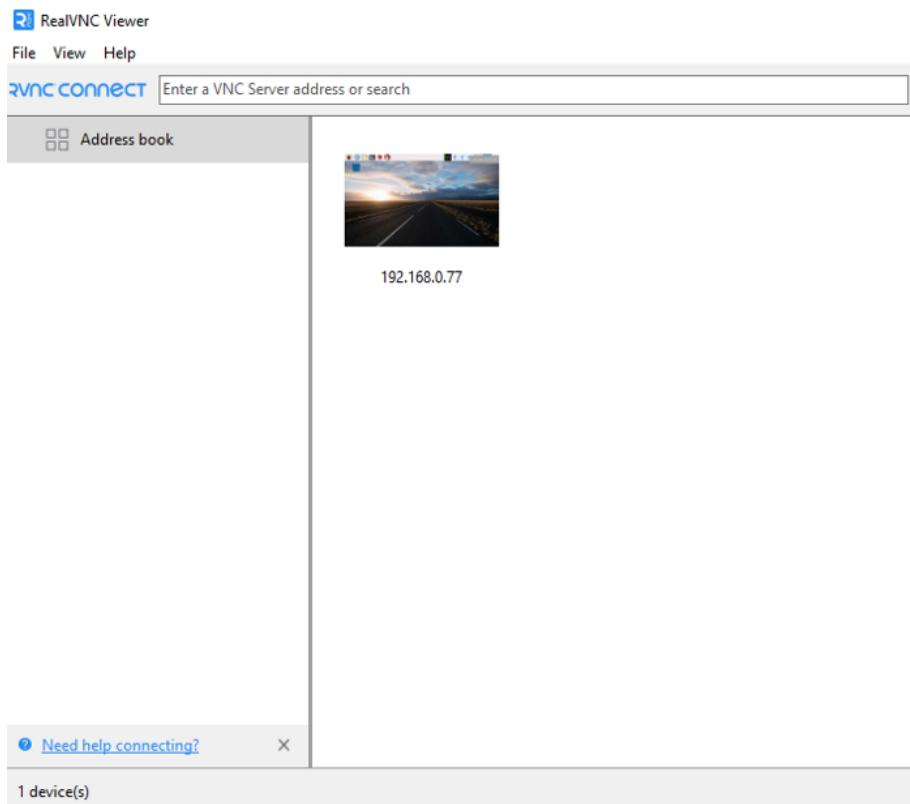
**Bukernavn:** pi

**Passord:** raspberry

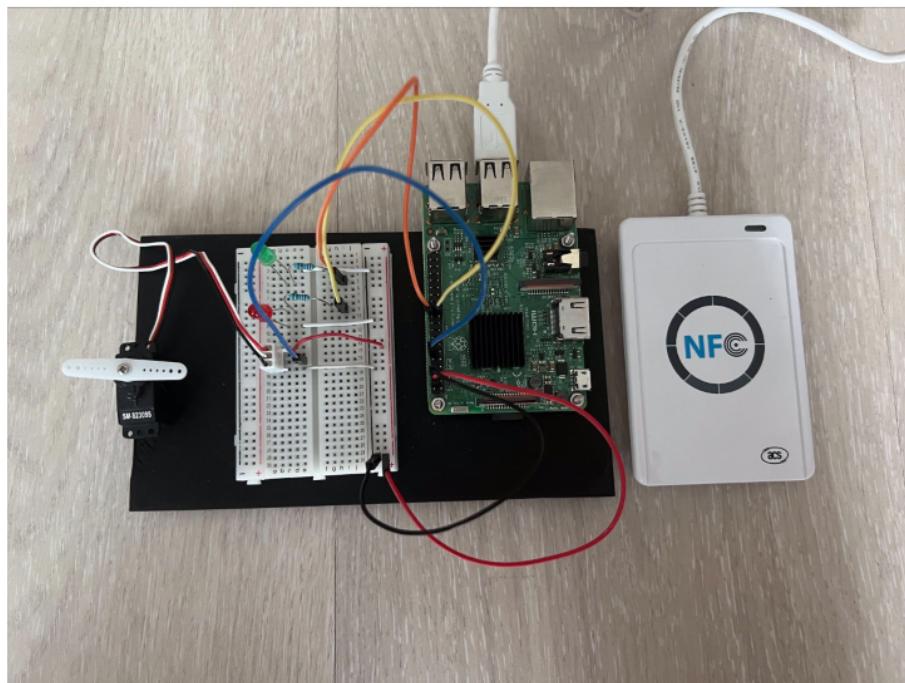
Det ligger vedlagtbilde av ferdig oppsatt RaspberryPi i RealVNC Viewer. 5.7

#### 5.3.5 Problematikk i database oppsett i Raspberry Pi

Vi opplevde mye problematikk i forhold til installasjon av databasen, se bilde for 'Databasefeil' for referanse. 5.9



Figur 5.7: RealVNCViewer



Figur 5.8: Prototype

```
pi@raspberrypi:~ $ sudo apt-get install mysql-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
Package mysql-server is not available, but is referred to by another package.
This may mean that the package is missing, has been obsoleted, or
is only available from another source

E: Package 'mysql-server' has no installation candidate
pi@raspberrypi:~ $ sudo apt-get install mariadb-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
Package mariadb-server is not available, but is referred to by another package.
This may mean that the package is missing, has been obsoleted, or
is only available from another source

E: Package 'mariadb-server' has no installation candidate
```

Figur 5.9: Databasefeil

## Feilsøking

Det ble utført en del feilsøking når det kom til å forsøke oppsettet av en database på en Raspberry Pi, som har kjørt på et litt eldre OS.

- **Oppsettet av Raspberry Pi**

- **Installering av OS**

- \* Vi oppdaterte Raspberry Pi fra Raspbian Jessie til Bullseye ved å følge diverse guider.
    - \* For å sjekke OS-versjonen brukte vi kommandoen:

```
cat /etc/os-release
```

- **Oppdatering av pakker**

- \* Etter oppgradering forsøkte vi å oppdatere alle pakkene ved å bruke:

```
sudo apt update
sudo apt full-upgrade
```

- \* Vi fikk feilmeldinger om "unmet dependencies", som følgende:

```
E: Unmet dependencies. Try using -f.
```

### - Feilsøking av pakke-avhengigheter

- \* For å forsøke å fikse avhengighets problemer, kjørte vi denne kommandoen:

```
sudo apt-get -f install
```

- \* Feilmeldinger viste at flere pakker hadde avhengighets-problemer, for eksempel:

```
dpkg: error processing package libnih1 (--remove):
dependency problems - not removing
```

## • Lasting av databaseprogramvare

### - Forsøk på installering av MariaDB/MySQL

- \* Vi forsøkte å installere MariaDB-serveren med:

```
sudo apt install mariadb-server
```

- \* Møtte igjen avhengighets-problemer med følgende feilmelding:

```
Package mariadb-server is not available, but is referred to by an
```

### - Fallback til ny Installering

- \* Etter flere mislykkede forsøk på oppdatering og installering, vurderte vi å re-installere operativsystemet fra bunnen av for å sikre at jeg hadde en ren start.

## • Nytt OS og databaseinstallasjon

- Lastet ned Raspbian Bullseye i .img-format og pakket ut filen.
- Brukte dd-kommandoen for å skrive bildet til SD-kortet:

```
sudo dd if=/path/to/raspbian.img of=/dev/mmcblk0 bs=4M
```

- Møtte feil ved at filbanen ikke var korrekt. Feilmeldinger indikerte at filen ikke ble funnet.

## • Konklusjon

- Oppsettet av Raspberry Pi har vært utfordrende, særlig på grunn av avhengighets-problemer med programvare.
- Fremover planlegger vi å fullføre installasjonen av databaseløsningen.
- Mangelen på en mikro SD-kortleser har gjort det problematisk å re-installere OS og utføre nødvendige oppdateringer. Men når vi har skaffet tilgang til enten en ny Raspberry Pi eller en SD-kortleser så skal problemene være fikset kort tid.

## Videre arbeid og automatisk kjøring av programvare

Vi fulgte konklusjonen og skaffet en SD-kortleser slik at det var mulig for oss å re-installere hele operativsystemet på maskinen. Deretter begynte vi oppsettet på nytt.

- Vi oppdaterte systemet på nytt med de nødvendige funksjonene og applikasjonene som vi trengte.
- Deretter opprettet vi en MariaDB-server for å håndtere databasen.

- For å laste ned serveren brukte vi kommandoen:

```
sudo apt install mariadb-server
```

- Deretter må vi sikre denne på en måte ved å bruke:

```
sudo mysql_secure_installation
```

- Gikk gjennom sikkerhets-stegene og satt et nytt root-passord. Denne satte vi til å være 'pi'.
  - For å få tilgang til databasen brukes kommandoen under, og deretter skriver inn passordet:

```
sudo mysql -u root -p
```

- Bare for å se om alt er riktig i etterkant, oppretter vi en database som heter 'Lockdb'.
  - For å få tilgang til localhosten som opprettes, må vi lage en bruker med alle rettigheter til å endre databasen. Denne opprettes ved kommandoene:

```
CREATE USER 'ole'@'localhost' IDENTIFIED BY 'changeme';
GRANT ALL PRIVILEGES ON lockdb.* TO 'ole'@'localhost';
```

- Ønsker GUI-verktøy for å kunne se databasen og enklere implementere dens verdier og egenskaper. Derfor laster vi ned PHPMyAdmin og PHP på maskinen med disse kommandoene:

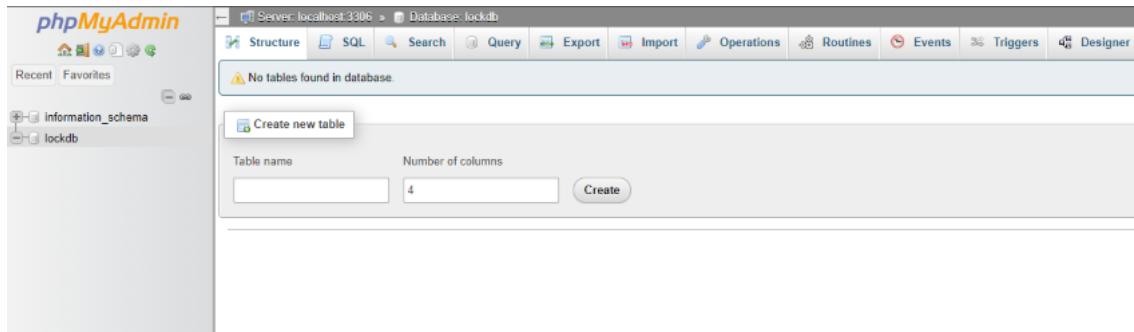
```
sudo apt install phpmyadmin
sudo apt install php
```

- For PHP-serveren velges apache2 som må konfigureres med en enkel kode for at det skal fungere som tenkt:

```
sudo nano /etc/apache2/apache2.conf
Include /etc/phpmyadmin/apache.conf
```

- For å få tilgang til apache og PHP, må vi koble dette opp mot nettet og HTML:

```
sudo ln -s /usr/share/phpmyadmin /var/www/html
```



Figur 5.10: Database

Når disse stegene er fullført, er databasen opprettet og man kan få tilgang til den ved å logge inn på brukeren vi har opprettet. Denne serveren er oppe så lenge maskinen går og den ser slik ut på web-basert form: Se bilde for Database. 5.10

Vi ønsker at opplevelsen når man tester prototypen skal være så behagelig som mulig. For å oppnå dette må vi legge til noen funksjonaliteter som fjerner unødvendige steg for å kunne benytte prototypen. For å automatisk kunne kjøre en Python-fil når man kobler Raspberry Pi-en opp mot strøm, må vi legge til et oppstartsscript. Dette gjøres enkelt i terminalen i **rc.local**-filen. For å redigere **rc.local**-filen bruker vi kommandoen:

```
sudo nano /etc/rc.local
```

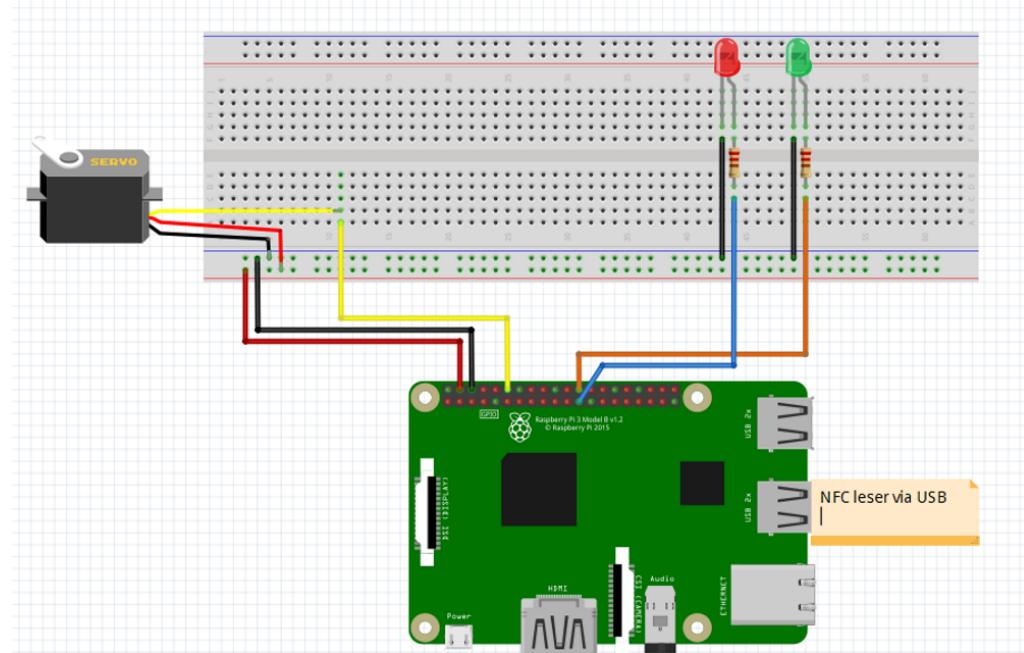
Når vi er inne på filen, vet vi allerede banen i mappestrukturen til filen som skal benyttes. Nederst på **rc.local**, før **exit 0**:, legger vi til denne banen:

```
python3 /home/raspberry/nfc_lock.py &
```

Når dette er lagt til, kan man lagre filen og restarte Raspberry Pi-en for å sjekke om riktig bane er lagt til, og at det fungerer som det skal.

En enkel metode for å finne riktig filbane i mappestrukturen er å kalle på navnet til filen med **realpath**. Sånn som i dette eksempelet:

```
realpath nfc_lock.py
```



Figur 5.11: Koblingsskjema

### Raspberry GPIO

Pins i bruk (se bilde Raspberry, 5.11):

**4** - 5V

**6** - GND

**12** - Servo

**23** - Rød lysdiode

**24** - Grønn lysdiode

**Pin 4 - 5V:** Dette er strømforsyningsspinne som leverer 5V. Gir direkte 5V fra Raspberryen, som brukes til å drive de komponentene som krever denne spenningen.

**Pin 6 - GND:** Jordingspinnen eller GND-pinnen gir en felles referanse for alle komponentene som er koblet til Raspberryen. Denne er nødvendig for at kresten skal fungere riktig, da alle komponentene trenger et jordingspunkt for å kunne kommunisere og dele strømmen jevnt.

**Pin 12 - Servo:** Denne pinnen støtter PWM (pulse width modulation), noe som er avgjørende for å kontrollere servoen. PWM brukes til å sende signaler av varierende pulsbreddesignaler til servoen, som bestemmer posisjonen dens. Det gjør det mulig for å justere signalet til servoen nøyaktig, slik at den kan rotere til den ønskelige vinkelen.

**Pin 23 - Rød lysdiode:** Denne GPIO-pinnen brukes til å kontrollere en rød LED. Fungerer som en utgangspinne som gir et digitalt signal som enten kan gi høy eller lav verdi, altså at LED'en er på eller av.

**Pin 24 - Grønn lysdiode:** Denne GPIO-pinnen brukes til å styre den grønne lysdioden på samme måte som ved pin-23 for den røde. Denne fungerer også som en enkel utgangspinne som gir digitale signaler som kontrollerer lysdioden. Grunnen til at det er to forskjellige pinner for hver lysdiode er for å kunne styre dem individuelt slik at dem fungerer som en statusindikator for låsen.

**NFC-leser koblet direkte til USB:** Det er koblet en NFC-leser direkte til USB-porten på Raspberryen. Dette er fordi NFC-leseren krever en mer stabil, rask datakommunikasjon som USB-porten gir. USB-porten er bedre egnet for å overføre data mellom Raspberryen og NFC-leseren, som gir raskere overføringshastigheter, bedre leseeegenskaper og protokoller som GPIO-pinnene ikke støtter.

## 5.4 App funksjonalitet og oppbygging

Android applikasjonens oppgaver er å koble seg enkelt til låser via Bluetooth for så å sende og motta informasjon fra låsene. Appen skal fortelle låsen om den skal være åpen eller lukket avhengig av hva bruker ønsker. Strømnivået til låsen er noe som appen skal motta via Bluetooth koblingen og kunne vise det på skjermen til bruker, slik at de vet når batteriet må byttes/lades. Når telefonen tapper på en av våre NFC taggs så skal appen sende informasjon på samme måte som om man skulle trykket på knappen i appen (se bilde for app) , for å låse eller låse opp for enklere tilgang til låsen uten å måtte trykke på noe inne i appen. 5.12

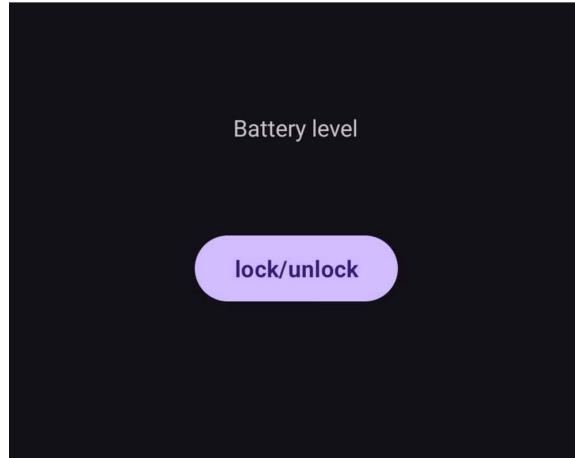
### 5.4.1 Appens oppbygging og implementasjon

Bluetooth kommunikasjonen tilkoblingen til appen blir utført av en klasse som heter BluetoothPairing. Formålet til denne klassen er å etablere ett bånd med en annen enhet, sette opp en socket for tilkobling og lage en BroadcastReceiver for å lytte etter endringer/tilkoblings statusen. BluetoothPairing er avhengig av flere tillatelsener for å kunne bruke Bluetooth og medfølgende funksjonaliteter. Disse tillatelsene blir definert i androidManifest.xml og deretter blir det kjørt kontroller i kodden for å verifisere at appen har fått tillatelsene før den utfører funksjonaliteten. BroadcastReceiveren er satt opp for å kunne gi en enkelt måte å feilsøke hvor i kodden det har gått galt og for å gi en bedre tilbakemelding på hva som skjer i kodden når alt fungerer. Det er også satt opp en maps funksjon som heter locksMap, dette skal være en liste over alle låsene i systemet, dette blir brukt i en klasse som heter BluetoothService.

BluetoothService er en klasse som bygger på hva BluetoothPairing har gjort. klassens formål blir å hente og sende ut data igjennom Bluetooth tilkoblingen, deretter skal denne informasjonen håndteres. Måten denne klassen fungerer er at det blir satt opp en input og output stream, dette er det som blir brukt for å hente og sende informasjonen. Denne klassen bruker da en read og write metode for å bruke input og output streamen. Locks er en klasse som skal holde på objekter med informasjon om hver lås. Når BluetoothService har lest informasjonen den har fått fra en lås så blir

informasjonen kontrollert ved hjelp av locksMap, i denne metoden blir det gjort en sjekk om låsen appen får informasjon om allerede eksisterer. Hvis informasjonen som blir mottatt ikke stemmer med et allerede eksisterende objekt så blir det opprettet et nytt objekt i Locks klassen, derimot hvis det allerede eksisterer en lås som stemmer overens med lista så blir bare informasjonen oppdatert. Måten dette blir sjekket på er at listen locksMap lagrer lås id-en til alle låsene som blir koblet til, sånn at når programmet henter informasjon fra den tilkoblede enheten så henter den også lås id-en til låsen og ser om den id-en eksisterer i lista. Kommunikasjonen mellom låsen og appen via Bluetooth sender informasjonen ved bruk av binær informasjon, hvor de første fire bittene sier noe om lås ID-en, de neste 4 bittene sier noe om låsens batteri nivå og de neste 4 bittene sier noe om hva slags status låsen har.

Håndtering av NFC/RFID lesing skal bli håndtert av klassen NfcService. denne klassen er ikke implementert enda, men skal være en del av applikasjonene ved det endelige produktet. I NfcService klassen kommer telefonen til å lese av RFID fra NFC taggen og sjekke dette opp mot brukerens tagger. Når taggen er verifisert sender appen informasjon til låsen ved hjelp av BluetoothService klassen.



Figur 5.12: Appdesign

Den viktigste klassen i koden er MainActivity, jobben til denne klassen er å hente metoder fra de andre klassene og implementere funksjonalitet fra eksempelvis BluetoothService og BluetoothPairing. MainActivity bruker de andre klassen for å kunne oppnå funksjonalitet i samsvar med hva bruker ser på skjermen sin. MainActivity jobber i samsvar med Activity Main, klassen som kun blir brukt til å sette opp et bruker grense snitt, for å gi knapper og views funksjonalitet. Når en feil oppstår i applikasjonen, eksempelvis sammenkoblingsfeil med Bluetooth, så fanger BroadcastReceiveren opp feilene og logger dette internt i appen, dette kan også bli brukt til å gi brukeren en feilmelding i brukergrensesnittet så alle parter har en forståelse på hva som har gått feil.

Resterende klasser i applikasjonene er Locks, NfcTaggs, Permissons og User. Disse klassen blir brukt for å lage forskjellige objekter som er relevant for appen. Locks klassen er en klasse som skal brukes for å opprette objekter rundt alle låsene i systemet. Alle NFC taggene blir også lagret som objekter av klassen NfcTaggs. Permissons og User er ment for å holde på informasjonen om alle brukerne og hvem som har tillatelse til å åpne låsene.

Koden er bygget opp ved bruk av modulær og objekt orientert arkitektur. Dette sørger for at koden blir enklere og oppretholdende da hver klasse har en bestemt oppgave. Eventuelle endringer i koden og det å legge til nye funksjoner i koden påvirker allerede eksisterende kode til en liten grad.

# Kapittel 6

## Testing / Evaluering / Resultat

I dette kapittelet skal det redegjøres for alle tester vi har gjort gjennom hele prosjektet, dette inkluderer testing av kode og materialer til prototype. Vi vil også inkludere funn og resultater av tester.

### 6.1 Testing av materiale

Produktet ble 3D-printet med fire forskjellige materialer for å vurdere deres egnethet til bryteren: PETG, PLA, FLEX og nylon. Resultatene viste at ingen av disse materialene fullt ut oppfylte kravene til både funksjonalitet og holdbarhet:

**PETG og PLA:** Fungerte tilfredsstillende for å demonstrere grunnleggende funksjonalitet, men hadde ikke tilstrekkelig holdbarhet for langvarig bruk.

**FLEX:** For seigt til å fungere effektivt i mekanismen, da høy fleksibilitet førte til tap av stabilitet.

**Nylon:** Viste seg å være for stift, noe som hindret bevegelsen i bryteren.

Videre arbeid vil fokusere på å finne et mer passende materiale eller en kombinasjon av materialer som kan balansere fleksibilitet og holdbarhet bedre. Logoen TapNester ment å symbolisere denne søken etter den optimale balansen mellom funksjonalitet, estetikk og brukervennlighet.

### 6.2 Testing Raspberry PI

I denne delen skal det fokuseres på de ulike testene utviklet for å sikre funksjonalitet i Raspberry Pi sin side av prototypen. Raspberry Pi'en har i oppgave å kontrollere låsen med de ulike funksjonalitetene tilknyttet, ta imot data fra applikasjonen, prosessere dette og sende informasjon tilbake. Testene som er brukt er utformet for å verifisere at koden som kontrollerer lås og kommunikasjon fungerer korrekt. Fokusområdet er å dekke både grunnleggende funksjoner

og spesielle situasjoner der ulike komponenter iteragerer. For prototypen er testene kjørt på komponentnivå med fokus på å sikre styringen av GPIO-pins samt kommunikasjonen over Bluetooth. Strategien for testene baserer seg på enhets testing (unit testing), hvor målet er å validere enkeltfunksjoner og moduler isolert fra hverandre.

### 6.2.1 Beskrivelse av testing

Testene er laget primært for å sikre pålitelig og korrekt funksjonalitet i koden, spesielt med tanke på de kritiske låse- og opplåsningsfunksjonene som styres. Ettersom at produktet integrerer flere systemer, er det viktig å sikre at både funksjonaliteten og integrasjonen fungerer stabilt.

#### Test av lock() og unlock() funksjoner i gpio\_control.py

**Test\_lock():** Denne tester at når lock() funksjonen kalles, så settes LED-pinnene riktig (låst LED tennes, ulåst LED skrus av), og at servoen justeres for å fysisk vise at den låser. Ved å benytte mocking av GPIO-modulen kan vi simulere høy og lav utgangspanning på disse, noe som gjør at det er mulig å teste oppførselen uten fysisk tilkobling. Dette tilsvarer en state-basedtesting, hvor vi får fokuset på den interne tilstanden i systemet gjennom GPIO-pinnene som leser av høye eller lave verdier.

**Test\_unlock():** Sjekker at den gjør det motsatte av lock(). Altså at låst LED skrus av mens ulåst LED tennes, samt at servoen settes i ulåst posisjon. Dette gjøres med samme metoder som i test\_lock()

#### Test av bluetooth\_server() funksjoner i bluetooth\_server.py"

Her implementeres det tester for å simulere dataflyten mellom klienten til serveren, som representerer kommunikasjonen som er mellom appen og raspberry pi låssystemet. Dette er også en enhets-test, men med fokus på grensesnitt-testing, hvor vi bekrefter at serveren mottar og tolker kommandoer riktig sendt over Bluetooth.

En del av denne testen simulerer også hvordan data sendes fra appen til serveren i form av en streng på 12 tegn. Dette for å bekrefte at systemet tolker kommandoene riktig sånn som tiltenkt. Denne strengen viser til lås\_id, batterinivået og om det låses eller åpnes. Data-driven testing brukes for å sikre at det tolkes på riktig måte.

**Test\_bluetooth\_server\_lock():** Tester at serveren reagerer på riktig kommando og kaller på lock() når den mottar signalet for å låse.

**Test\_bluetooth\_server\_unlock():** Gjør det samme, bare at den kaller på unlock() når dens signal for opplåsing sendes.

Ved bruk av disse testene og mocking for Bluetooth-kommunikasjonen, så gjør vi det mulig å teste logikken uten å benytte en faktisk Bluetooth-enhet som sender disse kommandoene til serveren.

```

raspberry@raspberrypi:~/projects/prosjekt $ python -m unittest discover -s tests
Bluetooth server listening on RFCOMM channel <MagicMock name='BluetoothSocket().__getitem__() id='547756842448
>
Accepted connection from ('00:00:00:00:00:00', 1)
Attempting to receive data...
Data received: b'000100650003'
lockID: 1, batteryLevel: 65, lockStatus: 3
Unlock
Attempting to receive data...
Data received: b''
Disconnected
All done
.Locked
.Unlocked
.
-----
Ran 3 tests in 2.146s
OK

```

Figur 6.1: Resultat av testing Ras.Pi

### 6.2.2 Resultater av testing

Totalt tre tester ble kjørt, alle med status "OK", altså uten feil eller unntak. Dette viser til at koden oppfyller de forventede funksjonelle kravene i låsesystemet.

Testen startet ved å sette opp en mock-simulering for Bluetooth-serveren, som reflekteres i første linje (Bluetooth server listening on RFCOMM channel). Mock-objektet bekrefter en tilkobling med en simulert adresse ('00:00:00:00:00', 1). Dette viser til at Bluetooth protokollen for kommunikasjon er korrekt i mock-miljøet. Serveren mottok data i form av en streng b'000100650003''. Denne ble korrekt analysert til å gi statuser for lockID, batteryLevel og lockstatus. Denne strengen utløser også prosessen for "unlock" av låsen. Paring og tolkning av signalet fungerer som forventet. Etter mottak av denne dataen mottar serveren en tom byte, noe som førte til at den frakobles. Dette er som forventet da systemet skal kjenne igjen enden på en tilkobling og håndtere frakoblingen. Deretter testes GPIO kontrollen av servoen. Noe som også gikk veldig bra gjennom. (se eksempel av utførelse i figuren Resultat av testing Ras.Pi, 6.1)

Alle testene ble kjørt gjennom terminalen (6.1). Terminalen gir en rask tilbakemelding og viser til enklere logganalyse av hva som er testet og gjennomført med eventuelle feilmeldinger som forekommer. Dette tillater observering av feilmeldinger, noe som er avgjørende for debugging og rettelser av koden.

### Nødvendige endringer som ble utført

I *bluetooth\_server.py* består testen av dataoverføringen via en simulert tilkobling. Her ble det oppdaget avvik i dataformatet som førte til feil i lockstatus-tolkningen.

**Forbedret parsing i Bluetooth koden:** Etter å ha lagt til en sjekk av dataformatet før videre prosesseringen, så ble Bluetooth-kommunikasjonen mer stabil. Dette resulterte i mer pålitelige testresultater samt at det sørget for at låse og åpne-kommandoer ble utført i flere kjøringer.

I *test\_gpio\_control.py* var forventningene at en viss PMW-verdi ville sette servoen til en spesifikk posisjon (eks 90 grader). Mockingen oppførte seg ikke nøyaktig som den faktiske maskinvaren, spesielt når det kom til presis PMW-simulering. Dette førte til avvik i testen.

**Intervall-basert testing av servo:** Ved å justere servoens posisjonstest til et akseptabelt intervall kunne vi oppnå bedre testresultater, til tross for begrensninger ved PMW-simulering.

Resultatene fra testingen med mocking indikerer at låsesystemet vil fungere som tiltenkt. Men noen funksjoner, spesielt de som involverer presis PMW-kontroll og Bluetooth-datahåndtering, vil kreve mer arbeid med annerledes maskinvare for å gi full funksjonalitet. Gjennomføringen gir oss ikke fullstendig data av Bluetooth-håndteringen med en faktisk tilkobling fra appen.

Testene er satt opp og utført i starten av november (sprint 7). Grunnen til at det er en sen utførelse av testene er opparbeiding av kompetanse og uferdig arbeid av bluetooth-serveren samt kommunikasjonen mellom appen og Raspberryen. Visse avgjørelser har påvirket at testene baserer seg primært på grunnleggende funksjonalitet fremfor å implementere alt.

## 6.3 Applikasjon testing

Fokuset i denne delen blir å se på de forskjellige testene som blir kjørt i Android applikasjonen, for å kontrollere funksjonalitet og kvalitet. Applikasjonens oppgaver er å kunne koble til en lås med Bluetooth og kunne sende og motta data over denne sammenkoblingen. Denne dataen skal håndteres og vises til bruker. Det er viktig og få kontrollert om alle funksjoner i applikasjonene fungerer som de skal.

### 6.3.1 Formålet med applikasjons tester

Testenes formål er å verifisere at funksjonaliteten i applikasjonen fungerer som tiltenkt. Ved å teste funksjonaliteten så tester vi også at applikasjonen opprettholder god kvalitet. Tester i koden kan hjelpe til med å luke ut feil i eventuelle oppdateringer som skal lanseres til applikasjonen før de blir publisert. Feilsøking er noe som blir lettere ved bruk av gode tester. Hvis det oppstår en feil er det mye lettere og finne feilen og finne ut hvor og hva som gikk galt hvis man har tester for hvert element i koden.

### 6.3.2 Test implementasjon

Applikasjonens tester er enhetstester. BluetoothService klassen og BluetoothPairing klassen har hver sin testing klasse som inneholder enhetstestene til den gitte klassen.

Testene i BluetoothPairingTest bruker Junit og robolectric for å kjøre og lage testene. Det er også opprettet shadow klasser for Bluetooth adapter og for Bluetooth device som et forsøk på å simulere en adapter og en device appen skal koble til. Det er gjort et forsøk på å kjøre tester for å sjekke at tilkobling og pairing metodene i BluetoothPairing klassen fungerer som de skal, men uten at det faktisk er en enhet og koble seg til. Problematikk og komplikasjoner rundt det å opprette en fiktiv tilkobling har gjort at disse testene fortsatt ikke fungerer som de skal.

BluetoothServiceTest klassen inneholder tester for informasjonshåndtering. Testene her passer på at informasjonen blir håndtert på riktig måte. Det blir testet om ønsket dummy data er dataen man får ut av metoden. Disse testene baserer seg på funksjonaliteten uten den faktiske bluetooth tilkoblingen, for å verifisere funksjonaliteten uten og være avhengig av noe variabelt sånn som bluetooth. I prototypen så blir det også opprettet en mocking klasse av BluetoothService, som skal hjelpe prototypen med å teste funksjonalitet og vise dette uten en faktisk enhet og koble til. Prototypen er hva vi har brukt for å teste bluetooth de forskjellige stadiene av tilkoblingen når bluetoothPairingTest ikke har fungert til å teste på ønsket måte. Det er laget mange logg instanser underveis i de forskjellige tilkoblings stadiene. Disse loggene er hva som har blitt brukt til å gi en tilbakemelding om hva som har gått feil og hvor langt inn i tilkoblingsprosessen applikasjonen har kommet.

Det er også opprettet en klasse for MainActivityTest, men denne klassen har bare blitt brukt for å undersøke mulighetene for å skrive flere tester som ikke var avhengig av en bluetooth tilkobling, i tillegg til å bruke appens prototype til å teste at knappen fungerer og at textViewet viser informasjon. Denne klassen har ikke blitt ferdigstilt.

Testene som er implementert til nå er implementert med tanke på en tilkoblet enhet. Etter hvert som produktet vokser og applikasjonen trenger å være koblet til flere enheter kan det skrives flere og nye tester som kan stressteste appen for å sikre at den fortsatt vil fungere mens den må håndtere mye informasjon samtidig.

### 6.3.3 Resultater

Underveis dukket det opp flere problemer med testene som førte til at det måtte gjøres endringer. Eksempelvis måtte det legges til en ny konstruktør i BluetoothService klassen for å kunne mocke riktig og få en fungerende prototype. Det ble skrevet en og en test for å sjekke at en test ikke påvirket den forrige. Testing av enhetene i applikasjonene har også ført til mange flere log meldinger enn tidligere for å letttere kunne feilsøke hvor noe i prosessen går feil eller for å se hva som skjer når appen kjøres. Det har blitt testet med å koble til Bluetooth med en faktisk enhet for prototypen, men det oppsto feil som ikke ble løst, dette førte til løsningen med å lage mocking klasse av BluetoothService for å vise funksjonalitet selv om det oppstår en feil. Testing av prototypen har også vist at knappen og textView oppfører seg som ønsket, samt at vi fikk testet at loggingen under kjøring fungerte som vi ville og viste relevant informasjon. Resultatene av bluetoothPairingTest testene har ført til mer fokus på å se på hva som blir loggført når prototypen kjøres, med tanke på problematikk rundt fiksjonell tilkobling.

Under utviklingen av applikasjonen så har testene og prototypen hjulpet med å avdekke svakheter med koden, som ikke var så lett å oppdage uten. Eksempelvis ble det veldig tydelig at det er viktig å spørre bruker om tillatelse til å bruke innebygd funksjonalitet i telefonen for at koden skulle fungere. Hvis ikke programmet spør om tillatelse til å bruke posisjon så kan applikasjonen ha problemer med nyere versjoner av Android.

# **Kapittel 7**

## **Drøfting / Diskusjon**

I dette kapittelet reflekteres det over valgene som er tatt gjennom prosjektet, og hvordan disse har påvirket utviklingen av produktet. Her fokuseres det overordnet på tekniske, praktiske og økonomiske avgjørelser, og vurderer deres fordeler og eventuelle ulemper. Målet er å gi en helhetlig forståelse av hvordan valgene har bidratt til produktets funksjonalitet, sikkerhet og brukeropplevelse, samt også hvilke utfordringer de har medført. I kapittelet vil det også vises noe til eventuelle forbedringsmuligheter for fremtidig utvikling.

### **7.1 Teamvalg**

Gruppen fordelte oppgaver basert på individuell kompetanse, noe som ga stor effekt i arbeidsflyten. Dette resulterte i fortgang og bedre gjennomføring. Ulemper ved denne metoden er at det var mer utfordrende å forstå helheten av prosjektet, noe som krevde mer arbeid til slutt med å sette seg inn i andre sitt arbeid.

Agilde-metodikken med ukentlige sprinter samt mandagsmøter, la til rette for god kommunikasjon og muligheter for justeringer undervis. Dette fungerte godt for problemløsning og ga fortgang i fremdriften. Her var utfordringen at det gikk en del tid i disse møtene, noe som var med å redusere den endelige tiden til relevant arbeid.

### **7.2 Sikkerhetsvalg**

For å beskytte produktet og brukernes data, undersøkte vi mulighetene for å implementere kryptering for all kommunikasjon, inkludert NFC- og Bluetooth-tilkoblinger. Vi identifiserte hva som må endres for å kunne sikre at personlig informasjon forblir beskyttet og at dataoverføringer ikke kan manipuleres. Selv om kryptering gir nødvendig sikkerhet, førte kompleksiteten til økt utviklingstid, som til tider utfordret fremdriften. I tillegg var det utfordrende å få Bluetooth-kommunikasjonen til å fungere stabilt mellom appen og låsen.

### 7.3 Designvalg og brukeropplevelser

Vi valgte å bruke compliant mechanisms i låsemekanismen, hvor vi benyttet bøyelige materialer i stedet for tradisjonelle bevegelige deler. Dette resulterte i en robust og slitesterk løsning med færre komponenter som kan slites ut, men krevde omfattende testing for å optimalisere materialenes funksjon. Det innovative designet reduserer slitasje, men kan også virke unødvendig komplekst. Designmessig har vi prioritert enkelhet for å sikre at produktet er intuitivt og tilgjengelig for målgruppen. Låsen kan monteres på de fleste skuffer og skap, og den kan åpnes manuelt selv om batteriet er tomt. Dette gjør produktet lett å installere, bruke og gir god fleksibilitet for brukeren. Materialvalget var også sentralt i utviklingen. Vi benyttet resirkulerbare materialer for å redusere kostnader og miljøpåvirkning. Til tross for at testene avdekket utfordringer med fleksibilitet og holdbarhet, lærte vi mye som kan bidra til ytterligere forbedringer av produktet. Siden vi brukte mye tid på selve låsemekanismen, fikk vi ikke tid til å utvikle magneten som faktisk får mekanismen til å fungere.

### 7.4 Teknologivalg

Valget av Raspberry Pi 3B+ som hovedenheten kom av behovet for en effektiv og fleksibel plattform. På den ene siden tilbyr Raspberry Pi et rikt utviklingsmiljø og støtte for et bredt spekter av verktøy, noe som gjorde det enklere å teste og integrere funksjoner som NFC, nettverk og Bluetooth-kommunikasjon. På den andre siden har enheten begrenset ytelse, noe som kan bli en utfordring dersom det utvikles noe som krever mer ressurser. Til forskjell fra en mikrokontroller, som er mer optimal for det ferdige produktet, så ga Raspberryen oss muligheten til en god utvikling, fleksibel håndtering og masse læring nødvendig for å oppnå et ferdig produkt.

Når det gjelder valg av MySQL som database, sikret dette en robust og sikker løsning. Her har vi kompetanse fra tidligere og kan utvikle noe som riktig håndterer lagring av data. På grunn av databasens evne til å støtte normalisering og relasjonsstruktur, er det enklere å fokusere på å skape en løsning som både er effektiv og som kan følge de nødvendige personvernsregler. Et annet relevant punkt er at MySQL åpner opp mulighetene for å skalere systemet i fremtiden, dersom det er ekstra funksjonalitet som skal legges til. Samtidig så kan dette virke negativt i en prototypefase hvor oppsettet blir for kompleks og vedlikehold er nødvendig. Men det som anses som det viktigste er at en robust database kan spare tid og ressurser i fremtididen.

Android Studio ble valgt som utviklingsmiljø for appen, noe som har flere fordeler. For det første gir det et godt tilpasset miljø for nybegynnere når det kommer til front-end utvikling. Android studio tilbyr enkelt oppsett av koden og gir mulighet til å teste applikasjonen før den går ut til brukere ved å benytte en innebygget emulator for å kjøre appen uten å ha en fysisk enhet å kjøre på. På den andre siden har Android Studio enkelte begrensninger, spesielt når det gjelder støtte for testing med verktøy som Mockito, noe som kreved ekstra innsats for å tilpasse testprosessen. Likevel, på grunn av den brede støtten fra utviklermiljøet, var dette valget et naturlig skritt for å prøve å utvikle en funksjonell og fin frontend som kan bygges videre på.

## 7.5 Valg av programmeringsspråk

Python ble valgt som programmeringsspråk for Raspberry Pi, hovedsakelig på grunn av sin enkelhet og tilgjengelighet når det kommer til styring av GPIO kontrollen og Bluetooth-oppsett. På den andre siden gjorde dette valget det mulig å legge til systemlogikk og integrasjon av funksjonalitet uten å bruke unødvendig tid på å forstå komplekt syntaks i for eksempel C++. Dette er noe som er spesielt verdifult i prototypefasen, der rask utvikling og testing er veldig avgjørende for videre utvikling. Python har posensiert lavere ytelse enn C++ eller Rust, noe som kan være en begrensning hvis systemet krever enda raskere respons eller mer optimalisert ressursbruk i fremtiden.

For utvikling av appen ble Java valgt, et språk kjent for sin stabilitet og plattformuavhengighet. Dette valget var veldig viktig ro å sikre at appen kunne kjøre på ulike Android-enheter, uten store tilpasninger eller endringer. Java har også integrasjoner med Android Studio, noe som gjorde det enklere å utnytte funksjonene som er i utviklingsverktøyene. På grunn av dette var Java et naturlig valg. På den andre siden kan Java være mer komplekst å lære og bruke enn for eksempel språk som Python. Et annet relevant punkt er at Java tilbyr veldig mange omfattende biblioteker som gjorde det mulig å bygge en robust applikasjon.

Derfor har valgene av Python og Java gitt oss en balanse mellom enkelhet og funksjonalitet i begge delene"av prosjektet. Dette kan også betraktes fra et annet perspektiv: mens Python muliggjorde rask prototyping, har Java lagt grunnlaget for en skalerbar og pålitelig applikasjon.

## 7.6 Funksjonalitetsvalg

Valget om å integrerer både NFC og Bluetooth som kjernefunksjoner i systemet baserer seg på ønsket om å gi brukerne fleksible og moderne alternativer for å benytte låsen. NFC gir en rask, enkel og kontaktløs metode for å låse opp, noe som er veldig brukervennlig. Den kan benyttes på forskjellige måter med forskjellige nøkler. Bluetooth, på den andre siden, åpner mer for avanserte funksjoner som fjernstyring. Selv om dette tilfører mer verdi til produktet, øker det også kompleksiteten både utviklings- og testprosessene. Samkjøring mellom disse to funksjonalitetene krever nøye planlegging og mye testing for å sikre at de fungerer sømløst sammen.

På grunn av utfordringene med å få Bluetooth-kommunikasjonen mellom appen og Raspberryen til å fungere, måtte vi ta et valg om å bruke mocking for å simulere tilkoblingen og mocke informasjonen som ble overført. Dette gjorde det mulig å demonstrere funksjonaliteten og vise at systemet fungerer som tiltenkt, selv uten en tilkobling. Kompetansen er tilstede, men tidspresset gjorde at mocking var derfor en nødvendighet i prosjektet. Dette innebar at testene ikke fult ut representerte de realistiske fremtidige bruksforholdene. Dette understrekker også viktigheten av å løse de tekniske utfordringene med Bluetooth i fremtidige versjoner.

Under testingen ble det gjort flere valg rundt hvordan teste produktet på best måte og hvordan verifisere funksjonaliteten til prototypen. Det å teste BluetoothPairing viste oss flere problemer rundt det å teste, mocke og simulerer bluetooth tilkoblinger i andorid studio. vi har forsøkt og mocke og lage våre egne shadow klasser for å simulere en bluetooth tilkobling, men dette har bare resultert i at vi ikke får ønsket utfall av hvilket verdier vi forventer og hva som er tillat i en andorid

applikasjon. Testing gjennom prototypen har åpnet muligheter for å se på loggene underveis i de forskjellige metode kallene og viser til hvor man kanskje kan legge til flere logg meldinger.

Frontend-utviklingen i Android studio var et bevisst valg for å sikre og få frem et profesjonelt og brukervennlig grensesnitt. Plattformen er som det ligger i ordet, godt tilpasset Android-utvikling. Men det å balansere designkravene mot de tekniske begrensningene vil gi utfordringer i prototypen. Derfor er det viktig at hver eneste bit av dette arbeidet er gjort riktig.

Samlet sett bidro disse funksjonalitetsvalgene med å gi prosjektet en solid retning å bevege seg i, samtidig som at de avdekket områder for forbedring og videreutvikling. Dette vises i fungerende grunnleggende funksjonalitet, sammen med at man kan se forbedringspotensiale.

## **Kapittel 8**

### **Konklusjon**

Vår lås benytter NFC OG Bluetooth-teknologi som gir mye fleksibilitet i bruken, disse teknologiene er ekstremt sentrale i hverdagene til vår målgruppe noe som gjør det enkelt for dem å benytte seg av da disse ofte er i 20-30 års alderen. Selv om vår målgruppe er småbarnsforeldre, kan den også brukes av andre. Låsen var har en nødløsning ved tom batteristrøm som gjør det enkelt å åpne låsen fysisk, dersom man ikke vet hvordan man kan åpne denne låsen vil man ikke ha tilgang til den, og dermed sikrer for andre enn småbarn. Dette er ikke en endelig løsning fordi sikkerheten blant voksne vil bli ekstremt ustabil, gruppen kommer derfor til å jobbe videre med en alternativ nødløsning sånn at produktet også kan benyttes av andre.

Selv om mye av valgene rundt oppgavefordelingen, arbeidsmetodikk og budsjettering var gode, er det forbedringspotensialer. Vi kan holde mer presise helhetlige møter for å øke produktiviteten, og får å øke den generelle forståelsen av prosjektet.

Det var en god beslutning å fokusere på kryptering for å beskytte brukernes data og sikre trygg kommunikasjon. Dette gir nødvendige sikkerhetsfordeler som er viktige for våre produkter. Imidlertid må vi i fremtiden prioritere å faktisk implementere kryptering i produktene våre, slik at vi kan oppnå de ønskede sikkerhetsmålene og sikre pålitelig kommunikasjon mellom komponentene.

Til tross for at låsen ble oppfattet som kompleks og at vi ikke fant det perfekte filamentet, mener vi fortsatt at dette er en god idé og en løsning vi ønsker å utvikle videre. Designet skiller seg fra andre produkter på markedet og har potensial til å være et bedre og mer kostnadseffektivt alternativ. Vi ser frem til å fortsette arbeidet med å optimalisere produktet og gjøre det enda mer funksjonelt og brukervennlig. Vi ønsker heller å fortsette å utvikle låsen og faktisk lage et fullstendig funksjonelt produkt.

Ut ifra problemstillingen har vi utviklet en funksjonell applikasjon som møter kravene stilt. Java og Android studio har åpnet muligheter for en fleksibel løsning som fortsatt er stabil. Bluetooth integrasjonene har avslørt utfordringer som krevde midlertidige tilpasninger, sånn som mocking av funksjonalitet. Løsningen nå er ikke optimal selv om den fungerer og er et godt start grunnlag for videre utvikling. En optimal løsning krever mer fokus på å utbedre Bluetooth integrasjon. Videre

krever en bedre løsningen mer omfattende testing med fysiske enheter. Prototypen fungerer nå for å demonstrere funksjonalitet og et utgangspunkt å bygge videre på.

Bruken av Raspberry Pi var veldig nyttig for utviklingen av produktet. Det tilbyr en bred fleksibel plattform med støtte for mange programmeringsspråk, egne GUI verktøy og mange spennende funksjonaliteter som gjorde utviklingen av prototypen enklere. Maskinvaren til Raspberryen gjorde det mulig å kjøre en rekke koder samtidig som at man benytter en database og bluetooth-server. I tillegg til innebygd nettverksfunksjoner blir Raspberry et perfekt miljø å utvikle en smartlås.

For det ferdige produktet vil en mindre mikrokontroller være et bedre valg. Mikrokontrollere er billigere, har lavere strømforbruk og er designet for å håndtere spesifikke oppgaver sånn som styring av låsen og kommunikasjonen gjennom NFC/Bluetooth. Mikrokontrollere har også færre komponenter som kan gå galt, noe som øker påliteligheten i det ferdige produktet.

Beslutningen om å bruke Python som programmeringsspråk for prototypen har vist seg å være et optimalt valg. Python er enkelt å bruke, det har omfattende støtte for maskinvare som er benyttet og det tillater oss å fokusere på utviklingen av systemlogikken uten veldig kompleks syntaks. Det åpner også opp for store muligheter for videreutvikling i samme språk.

Vi valgte å utføre simulert data i form av mocking fremfor en ekte Bluetooth-tilkobling direkte via appen. Dette var et nødvendighetsbasert valg vi måtte ta etter mange timer med feilsøking uten å få til en stabil kommunikasjon mellom appen og Raspberryen. Vi søkte hjelp med dette og kom ikke frem til en god løsning og måtte derfor ty til en nødløsning hvor vi også får vist frem funksjonaliteten og koden på en god måte. Derfor ble mocking en optimal løsning for denne problemstillingen og det fungerte bra med positive tester samtidig som at maskinvare oppførte seg som tiltenkt. Selv om dette ikke representerer en fullverdig løsning, var d

Databasen er satt opp på en strukturert og optimal måte for å håndtere produktets informasjon og data. Den er også designet for å ivareta informasjon som er relevant for bruken av produktet, samtidig som at datasikkerhet og personvern er i bildet. Til tross for ferdig oppsett av databasen er den foreløpig ikke direkte knyttet til produktet.

For videre arbeid av databasen må vi fokusere på å etablere kommunikasjonen mellom databasen og produktet. Noe som innebærer å implementere funksjoner som kan hente og sende informasjon mellom databasen og appen eller mikrokontrolleren.

Et sentralt område som trenger videre utvikling er frontend. Her har vi ikke klart å ferdigstille en optimal frontend i prosjektet våre. Det er en oppgave som er avgjørende for å levere et produkt som både er brukervennlig og intuitivt. For videre arbeid må det utvikles en frontend som følger prinsipper sånn som generell utforming med enkel navigasjon og tilgjengelighet, samtidig som at det inkluderer de funksjonene knyttet til smartlåsen.

# Referanseliste

Som referanser har vi brukt en del kilder fra Internett der vi har benyttet APA7 kildestruktur, vi har også brukt en nettside til å forkorte de lengste URL-ene slik at det blir en mer oversiktlig struktur i teksten (<https://n9.cl/no>). Vi har også benyttet oss av kunstig intelligens. Vi har benyttet KI til noen strukturelle tips, og fått den til å rette opp grammatikken i noen deler av dokumentasjonen, vi har lagt ved samtaler med KI som kilder under.

## 8.1 Nettsider

- Tabian, M. (n.d.). Bluetooth – Discover Devices. GitHub.

<https://github.com/mitchtabian/Bluetooth---Discover-Devices>

- Android Developers. (n.d.). Bluetooth. Android Developers.

<https://developer.android.com/develop/connectivity/bluetooth>

- Pi4J. (n.d.). Raspberry Pi 3B+ model pins (rev 1). Pi4J.

<https://www.pi4j.com/1.2/pins/model-3b-plus-rev1>.

- Raspberry Pi Foundation. (n.d.). Raspberry Pi software. Raspberry Pi.

<https://www.raspberrypi.com/software/>

- Thingiverse. (n.d.). Files for thing: 2988576. Thingiverse.

<https://www.thingiverse.com/thing:2988576/files>

- BYU Compliant Mechanisms. (n.d.). Maker resources. BYU Compliant Mechanisms.

<https://compliantmechanisms.byu.edu/maker-resources>

- Gaitskell, S. (2017, March 15). Why machines that bend are better. BBC News.

<https://www.bbc.com/news/technology-39227614>

- Statistisk sentralbyrå. (2021, November 4). Internett og mobil. Statistisk sentralbyrå.

<https://www.ssb.no/teknologi-og-innovasjon/faktaside/internett-og-mobil>

- Tripletex. (2022, November 1). Likviditetsbudsjett. Tripletex.

<https://www.tripletex.no/fagblogg/oppstartshjelpen/likviditetsbudsjett/>

- NRK. (2009, September 8). 267 mennesker fødes hvert minutt. NRK.

<https://www.nrk.no/urix/267-mennesker-fodes-hvert-minutt-1.7229498>

- Grandado. (n.d.). Yeelock smart skuffeskap lås, nøkkelløs, Bluetooth-app lås opp, tyverisikring, barnesikkerhetsfil, sikkerhet skuffebryter fra Xiaomi. Grandado.

<https://n9.cl/u7edv>

- Fruugo Norge. (n.d.). Yeelock smart drawer cabinet lock, nøkkelfri, Bluetooth-app lås opp, anti-theft switch, white. Fruugo Norge.

<https://n9.cl/zjk97>

- IKEA. (n.d.). Rothult smartlås, hvit. IKEA.

<https://www.ikea.com/no/no/p/rothult-smartlas-hvit-00359739/>

- Tek.no. (2021, March 9). Ikea Rothult. Tek.no.

<https://www.tek.no/korttest/i/y4MwPg/ikea-rothult>

- Tek.no. (2020, November 24). Det beste oppladbare batteriet. Tek.no.

<https://www.tek.no/samletest/i/Bj2Xme/det-bestе-oppladbare-batteriet>

- BN Bank. (n.d.). Enkeltpersonforetak eller AS?. BN Bank.

<https://www.bnbank.no/bedrift/enkeltpersonforetak-eller-as/>

- Brønnøysundregistrene. (n.d.). Bedrifts- og foretaksstatistikk. Brønnøysundregistrene.

<https://www.brreg.no/bruke-data-fra-bronnoysundregistrene/statistikk/bedrifts-og-foretaksstatistikk/>

- Altinn. (n.d.). Valg av organisasjonsform: Aksjeselskap. Altinn. <https://info.altinn.no/starte-og-drive/starte/valg-av-organisasjonsform/aksjeselskap/>

- Wikipedia contributors. (n.d.). Bluetooth Low Energy. I Wikipedia, The Free Encyclopedia.

[https://en.wikipedia.org/wiki/Bluetooth\\_Low\\_Energy](https://en.wikipedia.org/wiki/Bluetooth_Low_Energy)

## 8.2 Kunstig intelligens

- Svar fra Copilot:

You Sent by you:

Finnes det en global eller nasjonal standard for sikkerhet av leker for barn? Altså er det noe jeg bør tenke på dersom jeg skal produsere et produkt som har fokus i å sikre barn fra å komme i farlige situasjoner

Copilot Sent by Copilot:

Ja, det finnes både globale og nasjonale standarder for sikkerhet av leker for barn. Her er noen viktige standarder du bør være oppmerksom på:

1. ASTM F963: Dette er en omfattende standard for leketøysikkerhet utviklet av ASTM International. Den dekker en rekke sikkerhetskrav for leker og er kritisk for produsenter, importører og forhandlere i USA1. Denne standarden inkluderer krav til materialkvalitet, overflatebelegg, og mekaniske og fysiske egenskaper.
2. ISO 8124: Dette er en internasjonal standard for leketøysikkerhet som dekker lignende aspekter som ASTM F963, men er anvendelig globalt2. Den inkluderer krav til mekaniske og fysiske egenskaper, brennbarhet, kjemiske egenskaper, og mer.
3. EN 71: Dette er en europeisk standard for leketøysikkerhet som dekker flere deler, inkludert mekaniske og fysiske egenskaper, brennbarhet, kjemiske egenskaper, og elektriske egenskaper. Denne standarden er obligatorisk for leker som selges i EU.

Når du skal produsere et produkt med fokus på barns sikkerhet, bør du vurdere følgende:

- Materialvalg: Sørg for at materialene som brukes er trygge og ikke inneholder skadelige kjemikalier.
- Design: Unngå små deler som kan utgjøre en kvelningsfare, og sørg for at leken er robust og ikke lett går i stykker.
- Testing: Utfør grundige tester i henhold til relevante standarder for å sikre at produktet oppfyller alle sikkerhetskrav.

Har du noen spesifikke spørsmål om disse standardene eller om hvordan du kan implementere dem i ditt produkt?"

- Skriv punktvis om hva et akseselskap er

**ChatGPT sa:**

ChatGPT

Klart, her er en punktvis forklaring på hva et aksjeselskap (AS) er:

- **Definisjon:** Et aksjeselskap (AS) er en juridisk selskapsform der eierne har begrenset ansvar, kun begrenset til deres aksjeinnskudd.
- **Eierskap:** Et aksjeselskap eies av én eller flere aksjonærer.
- **Ansvarsbegrensning:** Aksjonærerne har ikke personlig ansvar for selskapets gjeld eller forpliktelser utover det kapitalbeløpet de har investert i aksjer.
- **Krav til egenkapital:** Minimumskravet til aksjekapital i et norsk aksjeselskap er 30 000 NOK.

- **Selskapsform:** Et AS er en selvstendig juridisk enhet, atskilt fra sine eiere.
- **Ledelse:** Det er krav om at et aksjeselskap skal ha et styre og en daglig leder, med visse unntak for små selskaper.
- **Regnskap og revisjon:** Et AS er pålagt å føre regnskap og sende inn årsregnskap til Brønnøysundregistrene. Revisjonsplikt gjelder for større aksjeselskaper.
- **Utbytte:** Aksjonærene kan motta utbytte fra selskapet basert på overskuddet.
- **Aksjetransaksjoner:** Aksjer i et aksjeselskap kan omsettes, men det kan være restriksjoner i vedtekten som regulerer salg til nye aksjonærer.
- **Skatt:** Et aksjeselskap betaler selskapsskatt på overskuddet.

Håper dette gir en tydelig oversikt over hva et aksjeselskap er.

- Chat gpt:

kan du finskrive dette for meg? Prosjektets avhengigheter Med tanke på at kravspesifikasjonene har stilt krav til at vi skal ha bluetooth sammekobling mellom app og lås samt nfc lesing så er det en del avhengigheter/biblioteker som kreves for å kjøre app koden som tiltenkt. Applikasjonen bruker flere av android sine egne biblioteker for å aktivere, koble til og overføre data ved bruk av bluetooth. Noen av bibliotekene som er importert blir også brukt for å gi koden funksjonaliteter som ikke nødvendigvis er en del av standard android studio. For å kunne utnytte bluetooth funksjonalitetene så er det også nødvendig med avhengigheter for tillatelse til å bruke de innebygde rammeverkene til android telefonen når appen brukes. Arkitektur Applikasjonen følger ikke en bestemt struktur 100%, men er inspirert av heksagonalarkitektur og bruker mange av prinsippene når det kommer til løst koblet kode. Appen blir kodet med løst koblet kode etter beste evne. Eksempelvis så skal BluetoothService og BluetoothPairing klassene ta seg av alt av bluetooth håndtering slik at ikke alle klassene skal være avhengig av å bruke bluetooth. Grunnen til at denne arkitekturen/måten og kode på er valgt er fordi dette gjør det slik at ikke all funksjonalitet skal være avhengig av hverandre. Hensikten er at selv om eksempelvis NFC lesing slutter å fungere skal du fortsatt kunne låse opp og igjen låsen ved bruk av bare bluetooth sammenkoblingen.

**ChatGPT sa:** ChatGPT

Selvfølgelig! Her er en finskrevet versjon:

### Prosjektets avhengigheter

Kravspesifikasjonene for prosjektet inkluderer behovet for Bluetooth-sammenkobling mellom appen og låsen, samt NFC-lesing. Dette medfører en rekke avhengigheter og biblioteker som er nødvendige for at appkoden skal fungere som tiltenkt. Applikasjonen benytter flere av Androids egne biblioteker for å aktivere, koble til og overføre data via Bluetooth. I tillegg importeres enkelte biblioteker for å tilføre funksjonaliteter som ikke nødvendigvis inngår i standard Android Studio. For å utnytte Bluetooth-funksjonalitetene, er det også nødvendig å ha avhengigheter som gir tillatelse til å bruke de innebygde rammeverkene i Android-enheten når appen er i bruk.

### Arkitektur

Applikasjonen følger ikke en spesifikk struktur helt og holdent, men er inspirert av heksagonalarkitektur og implementerer mange prinsipper for løst koblet kode. Den er

designet for å minimere avhengighetene mellom klassene. For eksempel vil klassene BluetoothService og BluetoothPairing håndtere all Bluetooth-funksjonalitet, slik at ikke alle klassene må være avhengige av Bluetooth. Valget av denne arkitekturen er bevisst; den tillater at ulike funksjoner ikke er helt avhengige av hverandre. Målet er at selv om NFC-lesingen skulle slutte å fungere, skal brukeren fortsatt kunne låse opp og igjen låsen kun ved hjelp av Bluetooth-sammenkoblingen.

Håper dette er til hjelp!

- **Svar fra Copilot:**

You **Sent by you:**

Finnes det en global eller nasjonal standard for sikkerhet av leker for barn? Altså er det noe jeg bør tenke på dersom jeg skal produsere et produkt som har fokus i å sikre barn fra å komme i farlige situasjoner

Copilot **Sent by Copilot:**

Ja, det finnes både globale og nasjonale standarder for sikkerhet av leker for barn. Her er noen viktige standarder du bør være oppmerksom på:

**ASTM F963: Dette er en omfattende standard for leketøysikkerhet utviklet av ASTM International. Den dekker en rekke sikkerhetskrav for leker og er kritisk for produsenter, importører og forhandlere i USA<sup>1</sup>. Denne standarden inkluderer krav til materialkvalitet, overflatebelegg, og mekaniske og fysiske egenskaper. ISO 8124: Dette er en internasjonal standard for leketøysikkerhet som dekker lignende aspekter som ASTM F963, men er anvendelig globalt<sup>2</sup>. Den inkluderer krav til mekaniske og fysiske egenskaper, brennbarhet, kjemiske egenskaper, og mer. EN 71: Dette er en europeisk standard for leketøysikkerhet som dekker flere deler, inkludert mekaniske og fysiske egenskaper, brennbarhet, kjemiske egenskaper, og elektriske egenskaper. Denne standarden er obligatorisk for leker som selges i EU.**

Når du skal produsere et produkt med fokus på barns sikkerhet, bør du vurdere følgende:

2. **Materialvalg:** Sørg for at materialene som brukes er trygge og ikke inneholder skadelige kjemikalier.
  - **Design:** Unngå små deler som kan utgjøre en kvelningsfare, og sørg for at leken er robust og ikke lett går i stykker.
  - **Testing:** Utfør grundige tester i henhold til relevante standarder for å sikre at produktet oppfyller alle sikkerhetskrav.

Har du noen spesifikke spørsmål om disse standardene eller om hvordan du kan implementere dem i ditt produkt?

- **Meg:** Jeg skal lage et sammendrag på rapporten vår til en akademisk tekst, rapporten er lagt ved. Sammendraget har disse kraven og skal være på mindre enn en side:

Nevn tydelig m alene for dette prosjektet, og rapporter hvorvidt hvert av m alene ble oppfylt eller ikke. Ut ifra dette skal det være mulig for de som finansierer prosjektet a avgjøre om prosjektet kan avsluttes, eller om ekstra arbeid skal utføres før prosjektet eventuelt kan avsluttes

Kom kun med Stikkord jeg kan ha med i sammendraget for å utgjøre en fin struktur

**ChatGPT:**

Her er stikkordene som kan utgjøre en fin struktur for sammendraget:

**Målene for prosjektet:**

- Hovedmål: Utvikle en sikker, brukervennlig barnesikringsløsning for skap og skuffer.
- Delmål:
  - \* Integrasjon av NFC- og Bluetooth-teknologi.
  - \* Utvikling av en løsning med fysisk nødløsning.
  - \* Økt brukervennlighet og bærekraft (bruk av resirkulerbare materialer).

**• Oppnåelse av mål:**

- NFC og Bluetooth: Implementert med god funksjonalitet.
- Brukervennlighet: Bekreftet gjennom brukertesting (enkel montering og fleksibilitet).
- Sikkerhet: Funksjonelle sikkerhetstiltak implementert (kryptering planlagt for fremtidige versjoner).
- Fysisk nødløsning: Løsning for manuell åpning ved strømbrudd inkludert.
- Bærekraft: Valg av materialer testet, videre utvikling nødvendig for optimal kombinasjon.

**• Anbefaling for videre arbeid:**

- Stabilitet og tilkoblingsutfordringer: Ytterligere optimalisering av Bluetooth-forbindelser.
- Produktionsforbedringer: Vurdering av materialvalg for bedre slitestyrke.
- Sikkerhetsforbedringer: Implementere mer avansert kryptering.

**• Avgjørelse for finansiering:**

- Prosjektets status: Klar for videre evaluering, men krever enkelte forbedringer før kommersiell lansering.
- Anbefaling: Ekstra arbeid anbefales for å sikre optimal drift og økt robusthet.

Disse punktene kan gi struktur til sammendraget og sikre at alle krav er dekket.

- Jeg skal skrive et diskusjon kapittel som skal være på rundt 1.5 side, diskusjon kapittelet er beskrevet sånn: Dere skal nå diskutere rundt valgene dere har tatt, og hvilken fordel/ulempe disse valgene har hatt for deres produkt.

ta utgangspunktet i kun disse punktene og skriv kun en liten mal om hva jeg bør ha med som tips : Teamvalg • Fordeling av oppgaver Oppgavene ble delt basert på individuelle kompetanser, noe som effektiviserte arbeidsflyten. Ulempen er at spesialisering kan ha begrenset forståelsen av helheten for enkelte teammedlemmer.

- Arbeidsmetoder og møter Regelmessige møter sikret god kommunikasjon, men noen ganger kunne tidsbruken på planlegging ha vært bedre brukt på utvikling.

Kostnader og budsjett • Pris for produktet Vi satte en målpris på 1200 NOK per enhet, noe som balanserer materialkostnader og en realistisk markedspris. Denne beslutningen kan gi utfordringer i fremtiden hvis komponentpriser øker eller flere funksjoner legges til.

Regelmessige møter sikret god kommunikasjon, men noen ganger kunne tidsbruken på planlegging ha vært bedre brukt på utvikling

ChatGPT

For å skrive et godt diskusjonskapittel basert på dine valg og deres fordeler og ulemper, kan du strukturere det slik:

Teamvalg

Fordeling av oppgaver

Beskriv hvordan dere fordelte oppgavene basert på medlemmers kompetanse. Diskuter hvordan dette bidro til effektiv arbeidsflyt, men påpek også ulempene, som spesialisering som kan ha begrenset enkeltes medlemmers helhetlige forståelse.

Arbeidsmetoder og møter

Reflekter over den metodikken dere brukte, som Agile, og hvordan ukentlige møter bidro til god kommunikasjon. Diskuter om tidsbruk i planleggingsmøtene noen ganger kunne vært bedre brukt til utvikling, slik at balansen mellom planlegging og produktivitet blir belyst.

Pris for produktet

Diskuter hvordan målprisen på 1200 NOK per enhet ble satt for å balansere materialkostnader og markedsrealisme. Påpek at selv om prisen dekker alt som trengs for installasjon, kan den bli en utfordring hvis kostnadene på komponenter øker eller hvis ytterligere funksjoner legges til.

Tips til refleksjon

Vis hvordan fordelene støttet prosjektets mål, og om ulempene førte til tilpasninger underveis. Ta med spesifikke eksempler fra gruppens opplevelser, for eksempel situasjoner der spesialisering ga en fordel, eller hvor møter ble avgjørende. Reflekter kort over hvilke lærdommer som kan tas med til fremtidige prosjekter. Dette gir et balansert bilde av valgene og deres konsekvenser, som tydelig viser både fremdrift og forbedringsmuligheter.





