Sprint rapport 2

Innholdsfortegnelse

2.1 Retrospekt

Uke	Gjøremål	Forklaring
38	Fiksing, Github	Fiksing av innlevering 1 ved hjelp av tilbakemelding og opprettelse av Github
39	Python	Starte med programmering av en prototype av et brukergrensesnitt i Python

Vi gjorde wireframing av et "point zero" produkt, hvor vi kan visualisere eventuelle features vi brainstormer sammen. På denne måten fikk vi god oversikt over hvilke valg vi burde ta, slik som valg av library. Vi fikk med det også tilgang til den tidligst mulige bruker testingen av vårt eget produkt. Ved gitt scenario slik som "Hvordan legger vi til en enhet?" (Definisjon på enhet i dette tilfellet er et Bluetooth-produkt), kunne vi se hvordan en mulig bruker ville bevege seg gjennom programmet. For å bedre representere et slikt scenario, samtidig som å teste library, bestemte vi oss for å faktisk programmere opp en demo som vi kunne bruke til dette eksperimentet. Denne demoen ligger under "Tktest" i GitHub repositoriet.

Ved oppretting av github repository bestemte vi oss for å bruke gitkraken grensesnittet for i vår mening, å best kunne konstruere en sikker og trygg struktur på prosjektet våres. Vi valgte gitkraken fordi det tilbyr en visuell representasjon av git-historikken, noe som gir oss bedre oversikt over forgreninger og merges. Dette forenkler samarbeidet og reduserer risikoen for konflikter når vi jobber parallelt. I tillegg integreres det enkelt med GitHub, noe som gjør det enkelt å håndtere pull requests og se status direkte fra grensesnittet. Dette gjør arbeidsflyten mer effektiv og sikrer at vi kan levere bedre kode på en raskere måte med mindre feil. Vi føler også at gitkraken er en brukervennlig løsning, noe som kan gjøre utvikling lettere for oss som fortsatt er noe i startsfasen av å lære oss utvikling, da spesielt i team som i dette faget. Med gitkraken blir hele arbeidsflyten vår enklere og mer oversiktlig.

Når det gjelder hvilket programmerings språk som vi endte opp med, ble det Python siden vi syntes det var en blanding av enkelt å bruke og at det var mange kraftige biblioteker som kan brukes for å lage et interaktivt brukergrensesnitt. Vi var og er fortsatt klar over at det kan oppstå problemer med at forelesningene og emnet generelt er rettet mot bruk av Java, til nå har vi enda ikke møtt på noen problemer som ikke kan løses og vi er sikre på at fremtidige problemstillinger kan løses i Python.

3.1 Avhengigheter til HomeHub

Tkinter:

Valg av GUI library ble gjort under konstruksjon demoen “TkTest”. Library vi har valgt å bruke kalles “TkInter” og er et veletablert og fleksibelt library som er fint til konstruksjon av grensesnitt.

PyTest:

Vi bruker PyTest for selve testingen av koden, da det er et effektivt verktøy for å automatisere enhetstester, sikre at funksjonaliteten fungerer som forventet, og identifisere eventuelle feil tidlig. Med PyTest kan vi enkelt skrive og kjøre tester, og det støtter ulike t yper av testscenarier, noe som gjør det enkelt for oss å validere koden i flere miljøer om nødvendig. En av de store fordelene med PyTest er at det er enkelt å skalere, noe som betyr at vi kan begynne med enkle tester og gradvis bygge ut mer og mer på systemet som prosjektet vårt vokser. Dette gir oss muligheten til å sikre kvalitet på prosjektet vårt, fra mindre moduler til større systemkomponenter. PyTest har også et stort samfunn og mange tilgjengelige plugins, noe som kan komme godt med siden vi også kan integrere det med GitHub.

Uuid:

Vi bruker Uuid for å kunne danne tilfeldige ider når vi produserer et objekt. Grunnlaget er blant annet at vi selvsagt ikke ser for oss et scenarie, hvor bruker har kompetanse til å sette id selv. Og i det tilfellet vi kunne sett for oss at programmet ville inkludere en form for innlogging (Noe vi forså vidt allerede til dels har gjort ved å inkludere 2faktor i prosjekt beskrivelsen) ville vi behøvd en id for hvert objekt brukeren har i hjemmet sitt, samt muligens bruke Uuid til kryptering av eventuelle passord (Litt usikkerhet rundt den biten enda).

OS:

Vi kommer til å bruke OS biblioteket for å få korrekt pathing til eventuelle bilder osv.

4.1 Sprint 3

Uke	Gjøremål	Forklaring
40	Diskusjon rundt avhengigheter og arkitektur	Redegjøring for hvilke libraries / rammeverker som kan være nødvendig for at vi skal kunne produsere produktet vi har lagt frem.
40 - 41	Kjernefunksjonalitet	Påbegynnelse av selve utviklingen av produktet. Dvs: read / write funksjon for å skrive over data til database, Generelle ekstra funksjoner til eventuelle objekter osv.

Vi bruker t-shirt sizing for å estimere tiden det tar for å fullføre sprint 3 oppgavene.

T-shirt størrelser

- **XS (Ekstra liten)**: Veldig enkel oppgave, tar svært kort tid. 1-2 timer.
- **S (Liten)**: En mindre oppgave som kan gjennomføres raskt. 1 dag.
- **M (Medium)**: En oppgave med moderat kompleksitet. 2-3 dager.
- **L (Stor)**: En mer kompleks oppgave som krever mer tid. 1 uke.
- **XL (Ekstra stor)**: Svært kompleks oppgave som kan strekke seg over flere uker.
- **XXL (Ekstra, ekstra stor)**: Uoversiktlig oppgave som er for stor til å anslå. Krever at oppgaven deles opp i mindre deler.

Tilordning av størrelser:

Element	Størrelse	Kommentar
Diskusjon rundt avhengigheter og arkitektur	S-L	Veldig usikker hvor lang tid det tar
Kjernefunksjonalitet	XL-XXL	Også mye usikkerhet rundt dette punktet

5.1 Backlog

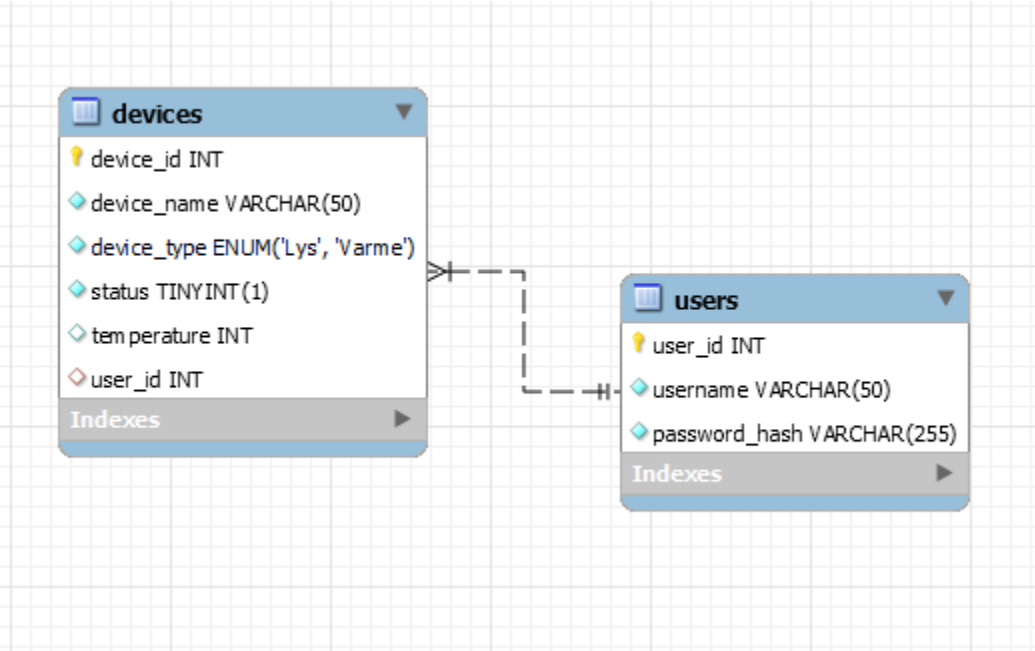
Element	Forklaring
Fiksing av Oblig 1	Fiksing av oblig 1 ble gjort etter tilbakemelding og er avsluttet.
Påbegynnelse av kode	Koden er påbegynt
Implementasjon av TKinter	TKinter elementer er påbegynt, dvs rot vindu og generelt design.
Implementasjon av kjernefunksjonalitet	Legge til flere klasser for kjernefunksjonalitet
Valg av avhengigheter	Valg av avhengigheter er dokumentert.
Valg av Arkitektur	Client-Server arkitektur er valgt og dokumentert.

5.2 Valg av arkitektur

Vi velger å strukturere vårt system med klient-server arkitektur. Det vil si at vi har en klient som håndterer brukerinteraksjon, men vi har også en server som kjører i bakgrunnen som inneholder ting som informasjon om brukere, IoT gjenstander osv.

I "HomeHub" sitt tilfelle vil dette si at vi f.eks. har informasjon relatert til boligens oppbygging lagret i en database hvor innholdet i radene vil være en form for sammenkobling med andre tjenester som gir ut den slags informasjon (Denne teknologien tar vi ikke mer stilling til en som så og vil herved være kjent som "Magi"). Det vil også være generell info knyttet til personen lagret i databasen, som f.eks. hvor mange IoT enheter de har integrert i programmet, samt et brukernavn og passord som brukes for å koble brukeren opp mot de forskjellige attributtene. Altså da nøkler som er koblet opp imot en primary key ID i systemet. Vi er kjent med hvilke datasikkerhetsmessige trusler et slikt system ville medføre, men det er også derfor vi har integrert et 2-faktor passord system blant annet. Passordet ville vært kryptert før det havnet i backend slik at passordet kun er kjent for brukeren, altså vi ville fulgt alle regler tilrettelagt for oss i henhold til GDPR. (Hvor vidt den endelige koden faktisk følger GDPR er irrelevant. Men tenk på det som om at det var planen i alle fall.

6.1 Database



Databasen slik tidligere nevnt vil lage en ny ID til bruker når de har registrert seg. Når bruker er tildelt en ID blir brukerens enhets attributter tildelt ut ifra hvilke endringer bruker gjør i klienten. Attributtene lagres og vil bli hentet fra server til klient, per server-klient arkitekturen tidligere nevnt.