Raportti Lainaus Projekti:

***-Loppuraportti,*** *jossa arvioitte ratkaisujen toimivuuden yhdessä tiimin kanssa sanallisesti*  
*-Pysyittekö alkuperäisessä suunnitelmassa? Jos ei niin mikä muuttui?*  
*-Jokainen* ***arvioi tiimin toimintaa ja****myös* ***omaa toimintaa*** *tiimin jäsenenä*  
*-Esittele sovelluksen eri toiminnallisuudet (myös ne mitä ei ole toteutettu)*  
***-Mistä projektityö löytyy*** *(mihin se tallennettu)*  
*-Loppuraportin tulisi olla vähintään neljä A4-sivua*  
*-Loppuraporttiin kuuluu* ***testiraportti****(mitä ja miten on sovellus on testattu ja kuka on testannut, testiympäristön esittely)*  
***-Arvioi toteutus*** *asteikolla 1-5 (anna* ***kokonaisarvosana projektityöstä****) ja lyhyt perustelu arvioinnille*

Alkuperäisessä suunnitelmassa me käytettiin figmaa, sivun suunnitteluun ja sivu on ottanut jotain inspiraatio niistä alkuperäisistä sivuista.

Projekti löytyy: <https://github.com/miskamero/Group-project>

Arvion projektin 4.5/5 siitä mitä user tiimille sai aikaseksi, Admin tiimille antaisin anteliaan 1.5/5

User tiimissä tehtyyn hyvin hommia ja aktiivisesti. Ei ollut päivääkään kun user tiimi ei tehnyt mitään, vaan aina oli jotain gitissä. Admi

Ratkaisujen toimivuuden yhdessä tiimin kanssa, oli todella heikkoa. Suurin osa ajasta kaikki oli hiljaa. Harvoin kukaan pyyti apua tiimi kavereilta, jos tarvi, mutta user tiimi sai osion tehtyä hyvin, sopivalla haasteella. Samaa ei voi sanoa admin tiimistä.

Kuva, joka sisältää kohteen teksti, kuvakaappaus, ohjelmisto, Multimediaohjelmisto

Kuvaus luotu automaattisesti

Kuva, joka sisältää kohteen kuvakaappaus, teksti, Multimediaohjelmisto, Grafiikkaohjelmisto

Kuvaus luotu automaattisesti

Tiimi Arvio:

Tiimi työ en sujunut hirveän hyvin, jotkut tekivät paljon ja hyvin, mutta jotkut eivät tehneet mitään.

Yksinäiset Arviot:

Jooa: UserTeam teki hyvin hommia, AdminTeam ei, piti tehdä admintiimin hommat

Miska: Mielestäni tiimissämme oli vain kaksi henkilöä jotka oikeasti teki hommia. Muut olisivat voineet tehdä tehtäviä. Muiden syyt siihen, että miksi he eivät tehneet mitään kuulosti vain tekosyille.

Verneri:

Sakke:

Julia

Testi Raportti:

Tässä projektissa testasimme kaikilla eri tavoilla, eli mentiin joka sivulle ja testattiin kaikkia mahdollisia tapoja mitä käyttäjä voi tehdä ja korjataan virheet.

Testaus tapa toimi, sillä tuli vastaan virheitä, jotka me korjasimme.

projekti testattii

Dokumentaatio

Projekti käyttää TypeScriptiä, Sass:ia ja Vitejs ja perustuu Reactiin

Projekti on rakennettu Nodejs versiolla <18

Käyttö:

npm i

npm run start

Komponentit:

**Lainaukset.tsx:**

Lainaukset komponentissa on lainaus sivun pääsivu. Jossa käyttäjä voi lainata, ja palauttaa kirjoja

**QRlainaus.tsx:**

QRlainaus komponentissa lainataan kirja URL:in mukaan “localhost:5173/ ‘Kirjan ID’ ”. Tätä komponenttia käytettään QR-koodin lukemisessa. QR-koodiin tulee nettisivun nimi + / kirjan id . Jonka avulla se kirja lainautuu

**Kirjautuminen.tsx ja Rekisteröinti.tsx:**

Log in ja Sign up sivusto “localhost:5173/login” tai “localhost:5173/signup”

Lisäys.tsx:

Lisäys komponentissa on admin sivu, jonka avulla voi nähdä toisten tilijen tiedot “localhost:5173/admin”

services tiedostot:

Services tiedostoja käytetään tietokannan manipuloimiseen. services.d.ts on “declare” tiedosto services.ts:lle. Eli siellä declerataan services.ts tiedoston muuttajat ja funktiot.

**Services.ts sfunktiot:**

**getUsers**

Etsii kaikki ja palauttaa kaikki tiedot lainaukset.json tiedostosta

**getBooks**

Etsii kaikki ja palauttaa kaikki tiedot kirjatdb.json tiedostosta

**updateBook**

**updateUser**

**borrowBook** (userName: string, bookID: string)

Ottaa tili id:n ja kirjan id ja lisää sen, kyseisen idn “tuoteet” listaan. Antaa kyseiset virheet jos ei voi lainata. Esim jos kirja on jo varattu

**returnBook** (userName: string, bookID: string)

Ottaa tili id:n ja kirjan id ja poistaa kirjan id:n, kyseisen id:n “tuoteet” listasta, jos voi

**addUser (**id: string, password: string)

Tätä funktiota käytetään Rekisteröinti sivustolla, joka luo tilin tietokantaan

**hash** (password: string)

Kryptaa salasanan

Tätä käytetään tilin luonnissa

**compare** (password: string, hash: string)

Palauttaa true jos “password” ja "hash” on sama

Tätä käytetään kirjautumisess

Käyttö esimerkki:

import \* as Action from '../../services/services';

const ReturnBook = async (userName: string, bookID: string) => {

const result = await Action.returnBook(userName, bookID);

// if the result is an error, display the error message

if (result.success === false) {

displayError(result.message);

console.log("error");

}

UpdateData();

setReturnBooks('');

};