Elokuvaharrastelijan sovellus made by NWA

Esittely:

Tämä on Movie Database by Nerds With Attitude-web-sovellus, joka mahdollistaa elokuvien ja näyttelijöiden arvostelun. Lisäksi sovelluksessa on edistynyt käyttäjien hallinta, voi luoda, liittyä ja poistaa ryhmiä, tarkastella muiden käyttäjien profiileja, jakaa niitä sekä nähdä heidän antamansa arvostelut ja lisäämät linkit.

Ominaisuudet

1. **Elokuvien ja näyttelijöiden arvostelu:**
    - Käyttäjät voivat lisätä arvosteluja elokuvista ja näyttelijöistä.
    - Arvosteluissa voi käyttää numeerista arviointia 1-5.
2. **Ryhmien hallinta:**
    - Käyttäjät voivat luoda omia ryhmiä.
    - Ryhmässä näkyy käyttäjien arvostelut.
3. **Käyttäjäprofiilit:**
    - Jokaisella käyttäjällä on oma profiilisivu.
    - Profiilissa näkyy käyttäjän antamat arvostelut ja lisäämät linkit.
    - Käyttäjä voi jakaa oman profiilisivun linkin.
4. **Linkkien jakaminen:**
    - Käyttäjät voivat lisätä linkkejä, kuten YouTube-videoita tai artikkeleita, sivustolle jakamista varten.
5. **Käyttäjän hallinta:**
    - Käyttäjät voivat rekisteröityä sivulle luomalla oman tunnuksen.
    - Käyttäjä voi hallita omaa salasanaa.
    - Käyttäjä voi poistaa oman tunnuksen.

Tekninen toteutus

Frontend (React)

- Käytämme Reactia käyttöliittymän rakentamiseen.
- Käytämme React Routeria navigoinnin hallintaan.
- Käytämme tilanhallintaa, esimerkiksi useState-koukkuja, elokuvien ja käyttäjäryhmien tilan hallintaan.
- Käytämme tyylien hallintaan esimerkiksi CSS-moduuleita.

Backend (Node.js) ja tietokanta (PostgreSQL)

- Käytämme Node.js:ää palvelimen rakentamiseen.
- PostgreSQL toimii tietokantana.
- Tietokanta on Render-palvelimella.
- Käytämme JWT (JSON Web Token) -autentikointia käyttäjien tunnistamiseen

Toni:

Projektissa toimin projektipäällikkönä ja olen suorittanut seuraavat tehtävät:

Projektin Suunnittelu:

- Laadin projektisuunnitelman, johon sisältyi visuaalinen mock-up Figmalla, tietokannan ER-kaavio Draw Io:lla sekä REST-rajapinnan suunnitelma README-kansiossa.
- Olin vastuussa GitHub-kanban-taulukon perustamisesta ja ylläpidosta sekä projektin alustamisesta Node.js:llä ja Reactilla.
- Rakensin ensimmäisenä mallin CRUD-toiminnoille tietokantaan asti, jota hyödynnettiin muissa toiminnoissa, ja loin .env-tiedoston arkaluonteisten kirjautumistietojen tallentamiseen.

Settings-Sivu:

- Kehitin käyttäjien hallintasivun, joka sisältää lähes kaikki CRUD-toiminnot frontendistä tietokantaan, mukaan lukien kirjautumisen, rekisteröitymisen, käyttäjän poistamisen ja CSS-tyylittelyn/responsiivisuuden.
- Muokkasin tietokannan rakennetta tämän toiminnallisuuden tukemiseksi.

Käyttäjän rekisteröityminen käyttää tokenia tunnistautumiseen, ja jokainen käyttäjän nimimerkki on uniikki. Rekisteröitymisen yhteydessä tarkistetaan myös salasanan vahvuus tai mahdollinen tyhjä sisäänkirjautuminen. Kirjautumisessa on huomioitu brute-forcing-hyökkäykset viiveellä kirjautumisessa ja SQL-injektiot on estetty sanitoinnilla backend-koodissa. Käyttäjän poistamisen yhteydessä varoitetaan käyttäjää pysyvistä seuraamuksista, ja poisto toimii siten, että ensin poistetaan kaikki tietokantataulut ennen käyttäjän poistamista.

Henkilökohtainen Käyttäjäsivu:

- Suunnittelin henkilökohtaisen käyttäjäsivun, jossa on kaikki CRUD-toiminnot, mukaan lukien linkkien tallennus, -haku, -poistaminen, salasanan vaihto ja käyttäjän poistaminen sekä CSS-tyylittely/responsiivisuus.
- Piilotin sen myös, jos ei ole kirjautunut palveluun.
- Rakensin tietokantaan uuden taulun tätä sivua varten.

Tallennetuissa linkeissä käyttäjä voi kuvailla tallentamaansa linkkiä, ja tallennuksen aikaleima näkyy. Lisäksi linkki toimii hyperlinkkinä kyseiselle sivulle.

Testikoodit:

- Laadin kattavan testikoodin käyttäjien lisäyksen, rekisteröitymisen ja poiston varmistamiseksi, käyttäen Mocha/Chai-kirjastoja.
- Kirjoitin 16 testitapausta, jotka kattavat koko prosessin käyttäjän hakemisesta lähtien, tarkoituksena varmistaa toiminnallisuuden eheys ja tietoturva.
- Koodi noudattaa ohjelmistotestauksen periaatteita, eli ketju on täysin yhtenäinen ja kattaa kaikki ongelmat mitä ohjelmoijan täytyy huomioida käyttäjän turvallisuuden varmistamiseksi.

Käyttäjäsivun Jakaminen:

- Työskentelin ryhmässä käyttäjäsivun jakamistoiminnallisuuden parissa.
- Olin vastuussa linkkien jakamisen mahdollistamisesta React Router Dom -toiminnon avulla, joka mahdollistaa osoitteiden ohjaamisen omille URL-osoitteilleen.
- Lisäsin myös toiminnon, jolla käyttäjä voi kopioida oman profiilisivunsa URL-osoitteen suoraan. Tähän käytin hyväksi tokenin dekoodaamista.

Haasteet:

- Testikoodin kirjoittaminen oli uusi alue minulle, ja kohtasin haasteita testiympäristön pystyttämisessä. Aloin ratkaisemaan ongelmaa aloittamalla yksinkertaisesta testistä opettajan esimerkin avulla. Kun yhteys saatiin toimimaan, laajensin testikoodia kattamaan monimutkaisempia toimintoja.

Osallistuin aktiivisesti myös projektin arkkitehtuurin suunnitteluun ja tietokantarakenteen määrittelyyn. Olin lisäksi aktiivisesti mukana tiimin ohjaamisessa ja mentoroinnissa, varmistaen sujuvan työskentelyn ja tehokkaan yhteistyön projektin eri vaiheissa.

Olen ylpeä siitä, mitä olemme saavuttaneet projektin aikana ryhmänä, ja toivon, että lopputulos heijastaa sitä tinkimätöntä laatua ja huolellisuutta, jota olemme panneet projektiin.

Jyri:

Vastasin sovelluksen Profiilisivun luonnista ja database pohjan luonnista renderiin. Lisäksi tein myös osioita mm. Review- ja Groupsivuille. Koodi on jaettu kahteen tiedostoon, *UserSearch.js* ja *UserProfile.js*, jotka muodostavat kaksi erillistä React-komponenttia. Koodissa on käytetty mm. State-koukkuja, asykronista hakua, tapahtumakäsittelijöitä, router/link-komponenttia, apufunktioita aikaleimojen muotiluun ja *loading ja fetching*-tiloja kertomaan, onko tietoja haettu tai ladataan. Lisäksi sivun muotoiluun käytetään mm. *ProfileCard.js* - korttia. Käyttöliittymä on jaettu useisiin komponentteihin selkeyden vuoksi.

Database on luotu postgreSQL-ympäristöön. Database on ladattu Render palvelimelle. Databasesta löytyy *Review, Actor, Customer, Groups, Groupusers, Joinrequests* ja *Personalpage*-taulukot.

```
POST_REQUEST:
  "INSERT INTO join_requests (groupid, username, admin, status) VALUES ($1, $2, $3, 'pending')",
```

Kuva 1Kuvassa postgreSQL-komento jota käytetään ryhmään liittymispyynnössä.

Henkka:

Omalta osaltani olin vastuussa sovelluksen perustan luomisesta, joka toimii nyt "Home"-sivuna. Samalla loin sivustolle yhtenäisen tyylipohjan, jota noudatetaan koko sivustolla. Navigointia varten loin elementin nimeltä "Navbar". Olin myös pääasiassa mukana luomassa Reviews-sivua ja elokuva-arvostelujen osalta ja siihen liittyviä ominaisuuksia, lukuun ottamatta järjestyksen muuttamista. Olin myös vastuussa "ReviewForm"-komponentin luomisesta ja käyttäjän arvostelun tallentamisesta tietokantaan. Näillä pääkomponenteilla on myös alikomponentteja, joita olen luonut sivuston toiminnan parantamiseksi, kuten esimerkiksi ”MovieCard”-komponentti. Osallistuin myös Group- ja Profiles-sivujen kehittämiseen.

React-sovelluksen "Home"-komponentti hallinnoi elokuvatietoja käyttämällä OMDB API:ta elokuvatietojen hakemiseen. Komponentissa on toimintoja, kuten elokuvahaku, genrehaku ja mahdollisuus näyttää yksityiskohtaisia elokuvatietoja. Se käsittelee myös käyttäjän vuorovaikutusta ja käynnistää "ReviewForm"-lomakkeen näyttämisen valituille elokuville. Puhtaalla ja modulaarisella koodilla se tarjoaa tehokkaan tilanhallinnan ja API-integraation saumattoman elokuvien etsintäkokemuksen takaamiseksi.

”Navbar”-komponentti hallinnoi navigointipalkkia ja sisältää toimintoja, kuten teeman vaihtaminen, uloskirjautuminen ja navigointivälilehtien hallinta. Se käyttää Reactin useRef-koukkua navigointipalkin tilan hallintaan ja tuo kuvakkeita (FaBars ja FaTimes) FontAwesome-kirjastosta, mistä saadaan hampurilaiskesi menevä navigaatio-nappi pienemmille näytöille. Komponentti käyttää myös React Routeria reititykseen ja näyttää eri välilehdet Link-elementtien avulla. "Navbar" tarjoaa selkeän ja käyttäjäystävällisen navigointikokemuksen sovelluksen eri osien välillä.

”Reviews”-komponentti vastaa elokuvien arvostelujen hallinnasta ja näyttämisestä käyttäjälle. Se käyttää Reactin useState- ja useEffect-koukkuja tietojen tilan seuraamiseen ja hakemiseen. Arvostelut haetaan palvelimelta, ja käyttäjällä on mahdollisuus muuttaa arvostelujen järjestystä eri kriteerien avulla. Komponentti tarjoaa käyttäjäystävällisen käyttöliittymän elokuva-arvostelujen selaamiseen ja järjestämiseen.

Niko:

Tein pääsääntöisesti group-sivuun liittyvää koodia (PostgreSQL database tableja ja columneja, backend-koodia routeihin, sekä react frontend-koodia)

Group-sivun ominaisuudet:

Kaikkien ryhmien katselu:

Group sivun avautuessa ensimmäisenä näkyy kaikki olemassa olevat ryhmät.

Yksittäisten ryhmien katselu:

Yksittäisten ryhmien tutkiminen tapahtuu painamalla “View Group” painiketta ryhmän nimen alapuolelta

Ryhmiä voidaan etsiä nimen perusteella:

Group sivulla on searchbar josta pystyy hakemaan ryhmiä.

Ryhmien luominen (vaatii sisäänkirjautumisen):

Ryhmiä voidaan luoda etusivun “Create Group” painikkeesta

Käyttäjäkohtaisten ryhmien tarkistelu (vaatii sisäänkirjautumisen):

Ryhmät, joissa kirjautunut käyttäjä on joko jäsen tai omistaja löytyvät “My Groups” näppäimen takaa.

Ryhmien poistaminen (täytyy olla ryhmän omistaja):

Kirjautunut käyttäjä pystyy poistamaan ryhmiä painamalla “Delete” näppäintä mikäli hän on ryhmän omistaja.

Ryhmään liittyminen (lähettää pyynnön omistajalle ennen kuin liittyminen tapahtuu):

Käyttäjä pystyy lähettämään liittymispyynnön haluamaansa ryhmään painamalla “join” näppäintä. Ryhmän omistajalle näkyy kaikki tiettyihin ryhmiin tulleet liittymispyynnöt, omistaja pystyy joko hyväksymään käyttäjän ryhmään tai poistamaan pyynnön, jolloin käyttäjää ei liitetä ryhmän jäseneksi.

Ryhmästä poistuminen (vaatii sisäänkirjautumisen):

Käyttäjä pystyy poistumaan ryhmästä “Leave Group” nappia painamalla

Ryhmän jäsenten elokuva-arvostelujen tarkistelu (vaatii sisäänkirjautumisen, sekä käyttäjän täytyy olla ryhmän jäsen):

Kaikki ryhmän jäsenet näkevät toistensa arvostelut “Group Reviews” näppäintä painamalla.

Ryhmästä poistaminen (vain omistajan ominaisuus):

Omistaja pystyy poistamaan minkä vain käyttäjän omasta ryhmästään painamalla “Kick” näppäintä.

Backend koodissa käytetty router-ominaisuuksia joilla saadaan tarvittavat tiedot databasesta, sekä pystytään lisäämään ja poistamaan tietoja verkkosivun painikkeiden ja ominaisuuksien kautta.

```
router.delete("/kickUser/:groupid_usergroups/:usernameToKick", authenticateToken, async (req, res) => {
```

Erilaiset database komennot on yhdistetty yllä mainittuihin routereihin riippuen siitä mitä milläkin komennolla halutaan suorittaa.

```
KICK_USER: "DELETE FROM groupusers WHERE username = $1 AND groupid_usergroups = $2",
```

Frontend koodissa luodaan funktioita

```
const handleKickUser = async (usernameToKick) => {
```

jotka ajetaan halutussa kohdassa html koodia

```
<button onClick={() => handleKickUser(user.username)}>
```

Eelis:

Minun vastuullani projektissa oli “Actor”- ja “Recommendation” -välilehtien sekä niiden ominaisuuksien kehittäminen. Lisäksi vastasin “Reviews” -välilehden “Actor Reviews” -osion toteutuksesta ja loin pohjan “Toggle Theme” -ominaisuudelle.

“Actor” -välilehden ominaisuuksiin kuuluu elokuvien ja näyttelijöiden haku API-avaimen avulla. Lisäksi “Actor” -välilehdellä käyttäjä voi antaa näyttelijäarvioita täyttämällä näyttelijän arviointilomakkeen, joka tallennetaan tietokantaan. Avautuessaan sivu pyytää käyttäjää antamaan elokuvan nimen, minkä jälkeen se hakee API-kutsulla elokuvia ja näyttää elokuvien posterit näytöllä. Näitä postereita voi napsauttaa, jolloin sivu näyttää kyseisen elokuvan päänäyttelijät. Käyttäjä valitsee näyttelijän ja täyttää arviointilomakkeen, joka tallennetaan tietokantaan. Näihin arvioihin pääsee käsiksi toiselta välilehdeltä.

“Reviews” -välilehdellä vastasin “Actor reviews” -osion toteutuksesta. Käyttäjä voi hakea näyttelijöiden ja elokuvien arvioita eri kriteerien perusteella, ja nämä arviot näytetään sivulla. Käyttäjä voi myös hakea näyttelijöiden arvioita syöttämällä haluamansa näyttelijän nimen hakukenttään. Lisäksi sivulla on parhaiden näyttelijöiden hakuominaisuus, joka näyttää korkeimmin arvostellut näyttelijät käyttäjien antamien äänien keskiarvon perusteella. “Reviews” -sivulla on myös “reset” -toiminto, joka palauttaa näytön lähtötilanteeseen eli tulostaa kaikki arvioinnit.

“Recommendation” -välilehti on täysin “front-endissä” toimiva sivu, joka tekee API-kutsuja ja suosittelee käyttäjälle elokuvia katseltavaksi. Käyttäjä valitsee ensin elokuvan lajityypin ja julkaisuvuoden, minkä jälkeen “Get Recommendation” -nappia painamalla näytölle tulostuu käyttäjän kriteereihin sopiva elokuvasuositus. Elokuvaesittelyt näkyvät julisteina, ja käyttäjä voi katsoa elokuvan painamalla “Watch this movie” -nappia.

Lisäksi loin pohjan “Toggle Theme” -ominaisuudelle, joka on navigointilaatikossa toimiva nappi sovelluksen taustavärin vaihtamiseksi. Toiminnallisuus on toteutettu “Navbar.js” -kansiossa, minne loin pohjan ja Henri muokkasi ominaisuuden sopivaksi navigointilaatikossa käytettäväksi.

Linkki sovellukseen:

Tiivistelmä ja Kokemuksia:

Projekti oli mielestämme mukavan haastava ja ryhmän yhteistyö sujui hyvin kuten aina. Kaikille riitti tasaisesti haastetta, kun jokainen joutui vuorollaan kirjoittamaan niin Node- kuin Postgre-koodia. Alussa Reactin kirjoittaminen tuntui kaikille haastavalta, mutta mitä pidemmälle projekti eteni sitä enemmän ryhmä oppi Reactin käyttöä.