Tampereen ammattikorkeakoulu

Movie-App Mobiilisovellus

Riku Mikkonen

OPINNÄYTETYÖ
Kesäkuu 2025

Tietojenkäsittely
Ohjelmistotuotanto

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittely
Ohjelmistotuotanto

MIKKONEN RIKU
Movie-App Mobiilisovellus

Opinnäytetyö 36 sivua, joista liitteitä 0 sivua
Kesäkuu 2025

Opinnäytetyössä suunniteltiin ja toteutettiin toimiva mobiilisovellus, jonka avulla elokuvia ja sarjoja pystytään selaamaan, ohjelmien tietoja tarkastelemaan sekä antamaan arvosteluja ja tallentamaan suosikkeihin. Sovelluksen kehityksessä hyödynnettiin nykyaikaisia mobiilisovellusten kehitystyökaluja, ja siihen integroitiin ulkopuolinen tietopalvelu elokuva- ja sarjatietojen hakemista varten. Työn aihe valittiin henkilökohtaisten kiinnostuksen perusteella, eikä toimeksiantajaa ollut.

Sovelluksen keskeiset toiminnot on rakennettu siten, että käyttäjät pystyivät kirjautumaan sisään ja käyttämään henkilökohtaisia ominaisuuksia, kuten suosikkilistojen hallintaa. Käyttöliittymä on suunniteltu käyttäjäystävälliseksi ja selkeäksi, jolloin eri näkymien välillä siirtyminen on sujuvaa. Projektin aikana sovelluskehityksen periaatteita ja käytännön taitoja syvennettiin, erityisesti kolmannen osapuolen rajapintojen hyödyntämisen osalta.

Projektin lopputuloksena valmistui toimiva sovellus, joka vastasi asetettuja tavoitteita ja tarjosi käyttäjälle monipuoliset mahdollisuudet elokuvien ja sarjojen selaamiseen ja arviointiin. Työ vahvisti ammattilista osaamista mobiilikehityksessä ja tarjosi konkreettisen näyttöprojektin.

Asiasanat: mobiilisovellus, elokuvat ja sarjat, käyttäjäkokemus

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programmer in Business Information Systems
Option of Software Development

MIKKONEN RIKU
Movie-App

Bachelor's thesis 36 pages, appendices 0 pages
June 2025

The purpose of this thesis was to design and implement a mobile application that allows users to browse movies and series, view details, save favorites and rate movies and series. The topic was chosen based on interest in mobile development. No external client was involved.

The application was created using widely used mobile development tools. Data was retrieved from a public movie database, and user authentication was implemented to support personalized features. Navigation between screens was arranged to provide user-friendly experience.

The application met the set objectives and demonstrated the intended core functionality. Practical skills in mobile application development and external API integration were strengthened.

Key words: mobile application, Movie and series browsing, User interface

SISÄLLYS

LYHENTEET JA TERMIT

API	Application Programming Interface. Rajapinta, jonka kautta sovellus voi hakea ja lähettää tietoa toiselle palvelulle.
Backend	Sovelluksen taustajärjestelmä, joka vastaa esimerkiksi datan käsittelystä ja tietokantojen hallinnasta. Ei näy käyttäjälle.
Branch (haara)	Versionhallinnan haarojen avulla voidaan kehittää uusia ominaisuuksia tai tehdä muutoksia vaikuttamatta suoraan pääkoodiin.
Debounce	Tekniikka, jolla viivästetään funktiokutsua (esim API-hakua). Vähentää turhia kutsuja ja parantaa suorituskykyä.
Frontend	Sovelluksen tai verkkosivun käyttöliittymä, jonka käyttäjä näkee ja jonka kanssa hän on vuorovaikutuksessa.
Komponentti	Komponentti on uudelleenkäytettävä ja itsenäinen käyttöliittymän osa React Native-sovellusta. Jokainen komponentti voi sisältää ulkoasua ja toiminnallisuutta. Komponentit mahdollistavat sovelluksen rakenteen jakamisen pienempiin osiin.
media_type	Kenttä, jolla merkitään hakutuloksen tyyppi (esim. movie, tv, person). Tätä käytetään, jotta sovellus osaa ohjata käyttäjän oikeaan näkymään.
Renderöinti	Sovelluksen käyttöliittymän (UI) päivittämistä näytölle. Kun komponentti tai näkymä ”renderöidään”, sen sisältöä tuotetaan ja näytetään käyttäjälle sovelluksessa.
Repository	Niin sanottu koodivarasto GitHubissa. Sisältää kaikki projektin tiedostot.
Stack Navigation	React Native-kirjastoon kuuluva navigaatioratkaisu, jossa näkymät avautuvat päällekkäin (Stack). Uusi näkymä avautuu vanhan päälle. Tähän kirjastoon kuuluu takaisinpainike.

1 JOHDANTO

Tässä opinnäytetyössä kerrotaan, miten suunnittelin ja kehitin Movie-App mobiilisovelluksen. Sovelluksen ideana on tarjota käyttäjälle mahdollisuuden selata, etsiä ja tallentaa suosikkielokuvia- ja sarjoja yhdellä ja samalla alustalla. Tavoitteena oli luoda käyttäjäystävällinen ja helppokäyttöinen sovellus, joka kokoaa keskeiset toiminnot yhteen paikkaan.

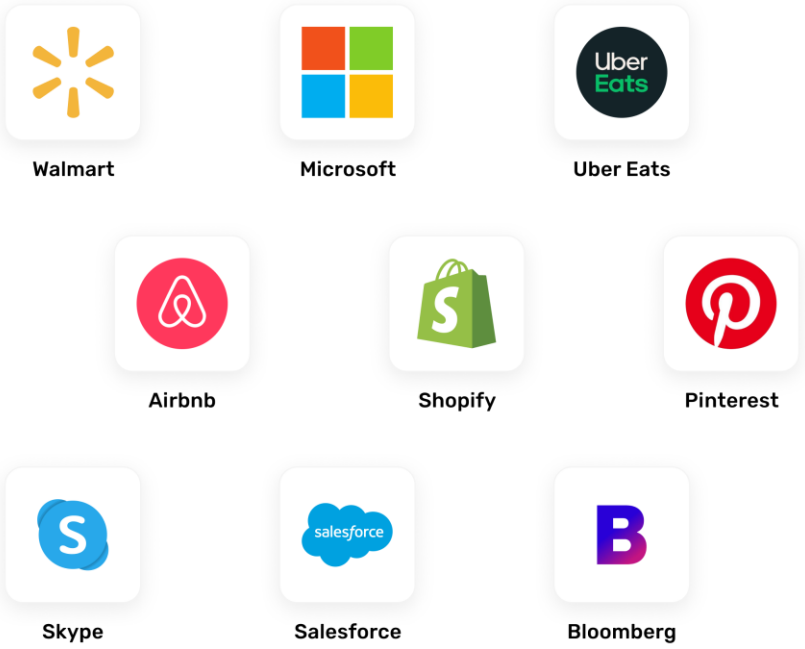
Olen katsonut koko elämäni aikana paljon elokuvia ja sarjoja. Tämän myötä olin pitkään pohtinut, kuinka voisin luoda oman sovelluksen, jossa voisi selata elokuvia, löytää uutta katsottavaa ja arvioida itse elokuvia. Valitsin sovelluskehityksessä käytettäväksi React Nativen-kirjaston, jonka avulla voi kehittää mobiilisovelluksia JavaScript ohjelmointikielellä Android ja iOS-laitteille (React Native 2025a). Näin ollen oman mielenkiinnon ja oppimisen kannalta päätin yhdistää mobiilikehittämisen ja opinnäytetyön prosessin.

Opinnäytetyön kohderyhmänä ovat paljon elokuvia ja sarjoja katsovat kuluttajat, jotka arvostavat helppokäyttöistä sovellusta, josta voi etsiä ja arvostella sekä lisätä suosikkeihin mieluisia ohjelmia myöhempää selaamista varten. Työssä keskitytään sovelluksen ohjelmointiin sekä käyttöliittymän kehittämiseen.

2 TEKNOLOGIAT JA TYÖKALUT

2.1 React Native

React Native on Facebookin (nykyinen Meta) kehittämä avoimen lähdekoodin ohjelmistokirjasto. Kyseisellä kirjastolla voi rakentaa natiivikomponentteja hyödyntäviä mobiilisovelluksia, jota käytetään useissa kansainvälisesti tunnetuissa sovelluksissa, kuten Instagram ja Tesla mobiilisovellus. Kirjaston avulla voidaan saavuttaa hyvä suorituskyvyn ja näin mahdollistaa nopeamman kehityksen, koska suurin osa koodista on jaettavissa eri alustojen kesken. (React Native 2025b). Tässä vielä kuva tunnetuista sovelluksista, jossa on käytetty React Native-kirjastoa: (Kuva 1)



KUVA 1. Tunnettuja React Nativea hyödyntäviä sovelluksia (Kuva: Patel Rushi 2023).

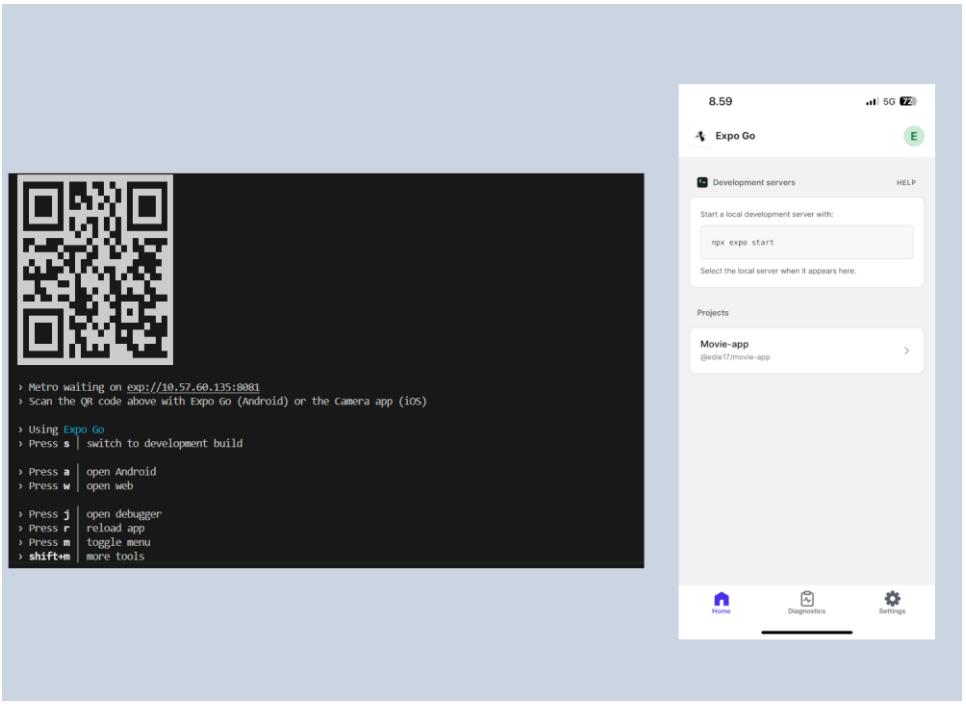
React Nativen on suunniteltu helpottamaan web-kehittäjien siirtymistä mobiilikehitykseen ilman, että heidän tarvitsee opetella erillisiä natiiviohjelmointikieliä, kuten Kotlinia tai Swiftia. Se mahdollistaa koodin jakamisen alustojen välillä, mutta

osa käyttöliittymästä rakennetaan web-teknologiolla, mikä voi vaikuttaa käyttökokemukseen. (Sakniuk & Boduch 2024, 2 %).

2.2 Expo

Expo on avoimen lähdekoodin ohjelmistokehys, jonka tarkoituksena on helpottaa Android ja iOS-sovellusten kehittämistä (Expo 2024a). Expon kehityksen aloitti Charlie Cheever vuonna 2015 kehittääkseen kaikille avoimen alustan, jolla jokainen voivat kehittää omia mobiilisovelluksia. (Kokalitcheva, K. 2016).

Expo tarjoaa React Native-kehittäjälle hyvän testaus- ja kehitysympäristön. Yksi keskeisin työkalu on Expo-Go-sovellus, jonka avulla kehittäjä voi testata sovellusta reaaliaikaisesti ilman, että tarvitsee asentaa sovellusta puhelimeen. Tämä mahdollistaa nopean testauksen sovelluksen kehityksen aikana. (Expo 2025). Alapuolella kuva Expo-Go-sovelluksen käytöstä: (KUVA 2)

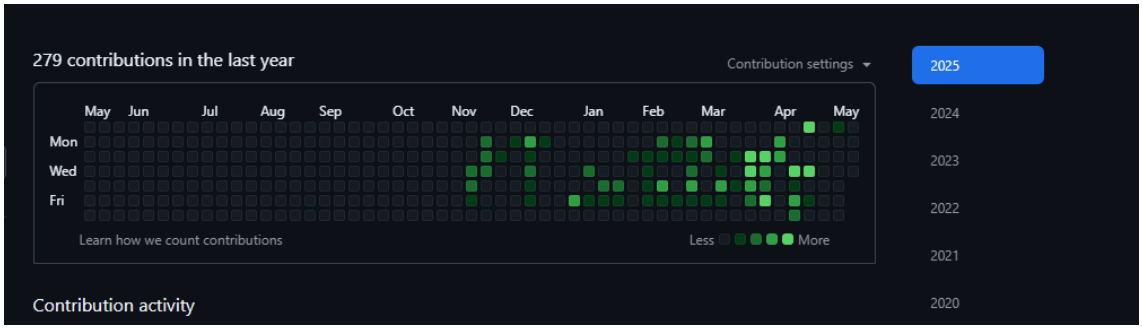


KUVA 2. Esimerkki Expo-Go-sovelluksen näkymästä tietokoneen konsolissa ja puhelimessa (Kuva: Mikkonen Riku, 2025).

Toinen merkittävä Expon tarjoama työkalu on EAS Build, joka on pilvipohjainen työkalu sovellusten rakentamiseen. Sen avulla kehittäjä voi luoda valmiit asennuspaketit Android ja iOS-laitteille ilman tarvetta omistaa esimerkiksi Applen konetta iOS-version rakentamista varten. (Expo 2024b).

2.3 GitHub

GitHub on pilvipohjainen alusta, johon voit varastoida ja jakaa koodia, sekä työskennellä yhdessä muiden kehittäjien kanssa saman projektin parissa. GitHubissa koodi tallennetaan niin sanottuihin repositoriohin eli tietovarastoihin. Alusta perustuu Git-versionhallintajärjestelmään, jonka avulla kehittäjät voivat seurata ja hallita koodin muutoksia (GitHub Docs n.d. a).



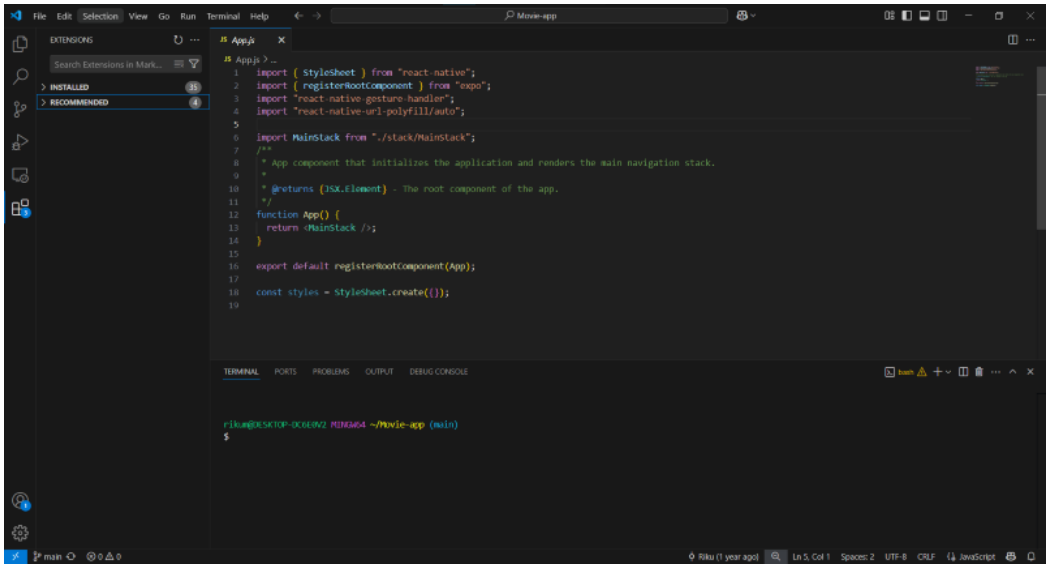
KUVA 3. Kuva GitHub-profiilin aktiivisuudesta (Kuva: Mikkonen Riku, 2025).

GitHubia käytetään laajasti ohjelmointiyrityksissä, koska se mahdollistaa usean kehittäjän työskennellä samanaikaisesti samassa projektissa ilman, että koodi sotkeutuu tai tulee päällekkäisyyksiä (GitHub, 2022). Samanaikainen kehitys onnistuu luomalla jokaiselle käyttäjälle oman haaran (Branch), jossa voi tehdä muutoksia itsenäisesti ennen niiden yhdistämistä pääprojektiin (GitHub Docs, n.d. b).

2.4 Visual Studio Code

Visual Studio Code (VS Code) on Microsoftin kehittämä maksuton avoimen lähdekoodin monialustainen ohjelmointiin tarkoitettu tekstieditori, joka on saatavilla Windows, MacOS ja Linux käyttöjärjestelmille (Microsoft. n.d.). Ohjelma tukee useita ohjelmointikieliä, kuten JavaScript, TypeScript ja Java. Ohjelmassa on

mahdollista myös asentaa monia muita laajennuksia (Extensions), jotka helpottavat ja tehostavat ohjelmointia. Ohessa kuva VS Coden editorista: (KUVA 4)



KUVA 4. Esimerkki VS Code-editorista tietokoneella (Kuva: Riku Mikkonen, 2025).

Vs Code sisältää myös sisäänrakennetun älykkään koodin täydennystyökalun, (IntelliSense), joka tarjoaa automaattisia ehdotuksia ja virheiden tunnistusta. Lisäksi ohjelma sisältää sisäänrakennetun Git-versiohallintatyökalun, jonka avulla kehittäjät voivat hallita muutoksia suoraan ohjelmasta käsin. (Visual Studio Code 2025).

2.5 The Movie Database API (TMDB API)

The Movie Database API (TMDB API) kehittäjille suunnattu, yksityiskäytössä ilmainen ohjelmointirajapinta, jonka avulla voidaan hakea tietoa elokuvista ja tv-sarjoista, kuten kansikuvia, näyttelijätietoja ja arvosteluja. API- rajapinnan kautta on mahdollista lähettää arvosteluja, jos käyttäjä on kirjautunut. (TMDB 2024).

Tietojen hakeminen API-rajapinnan kautta onnistuu käyttäen HTTP-kutsuja ja vastaukset saadaan JSON-muodossa. JSON (JavaScript Object Notation) on kevyt ja helposti käsiteltävä tietorakenne. Se on erityisen yhteensopiva JavaScript-pohjaisten sovellusten, kuten Reactin ja Node.js:n, kanssa, koska JSON on alun perin suunniteltu JavaScriptiä varten. (JSON. n.d).

3 MOVIE-APP MOBIILISOVELLUS

Tässä luvussa esitellään Movie-App-sovelluksen toiminnallisuuksia ja ominaisuuksia Frontend-puolelta. Sovelluksesta otettuja näyttökuvia hyödynnetään opinnäytetyöprojektin havainnollistamiseen ja sisällön ymmärrettävyyteen.

3.1 Sovelluksen ominaisuudet

Movie-App-sovellus on elokuvien ja sarjojen näyttämiseen tarkoitettu sovellus, joka mahdollistaa ohjelmien hakemisen, selaamisen ja yksityiskohtien tarkastelun. Käyttäjä voi kirjautuneena lisätä sarjoja tai elokuvia suosikkeihin, sekä arvostella ohjelmia. Sovellus on kehitetty englanninkieliseksi.

Sovelluksen käynnistyessä käyttäjälle avautuu kirjautumisikkuna, josta hän voi valita, haluaako kirjautua vai käyttää sovellusta vierailijatilassa. Vierailijana käyttö on osittain rajoitettua, mutta perustoiminnot ovat käytössä, kuten elokuvien ja sarjojen selaaminen.

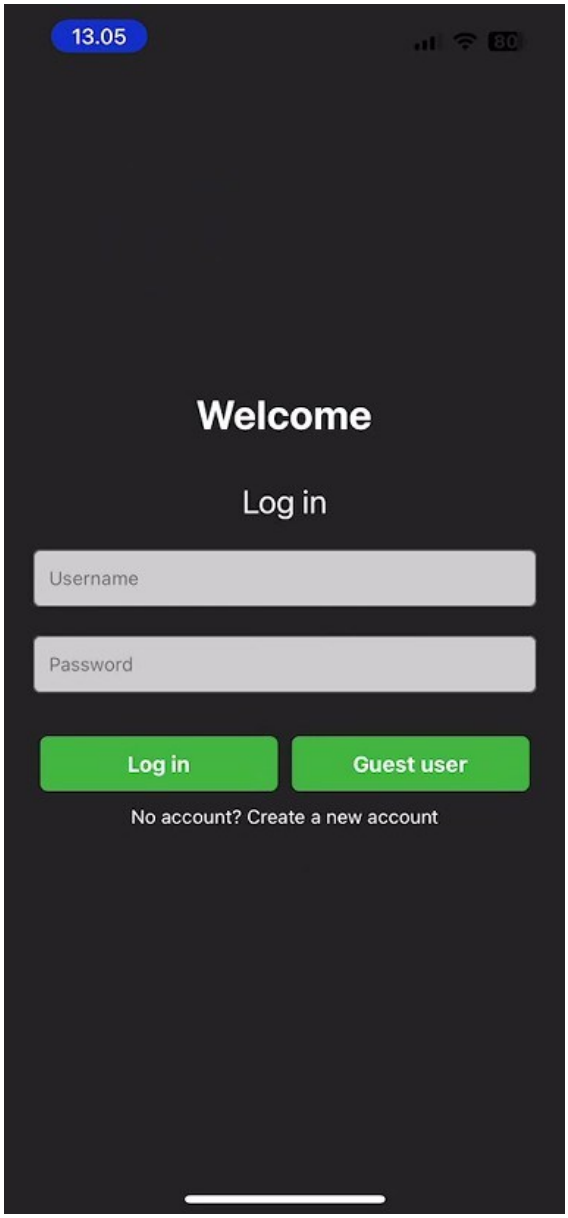
Sovelluksessa on useita eri näkymiä, kuten esimerkiksi elokuvat, sarjat, suosikit ja arvostellut sivut. Näiden välillä liikutaan käyttäen muun muassa yläreunassa sijaitsevia Segmented Buttons-näppäimiä sekä alareunassa sijaitsevaa navigointipalkkia (BottomTabs).

Sovellus hyödyntää TMDB (The Movie Database) – rajapintaa (API) elokuvien ja sarjojen tietojen hakemiseen. TMDB kautta toteutetaan myös kirjautuminen, joka mahdollistaa käyttäjäkohtaiset ominaisuudet, kuten suosikkien ja arvostelujen lisäämisen.

3.2 Kirjautuminen ja vierailijaikkuna

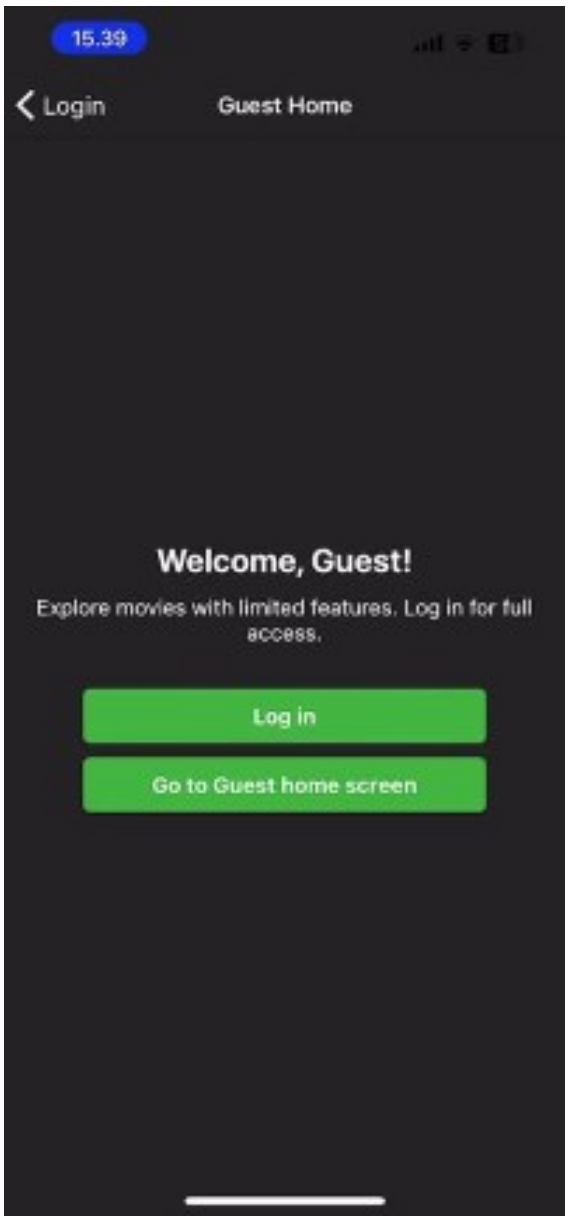
Sovelluksen käynnistyessä avautuu kirjautumisnäkymä, josta voi valita kirjautumis- tai vierailijatoiminnon välillä. Näkymässä kaksi tekstikenttää käyttäjätunnukselle ja salasanalle sekä kaksi painiketta, jolla voi valita kirjautumisen tai vieraili-

janäkymän välillä. Mikäli käyttäjällä ei ole vielä tunnusta, näkymästä on mahdollista päästä rekisteröitymään TMDB-sivustolle. Ohessa kuva kirjautumisnäkymästä: (KUVA 5)



KUVA 5. Näkymä kirjautumisikkunasta (Kuva: Mikkonen Riku, 2025).

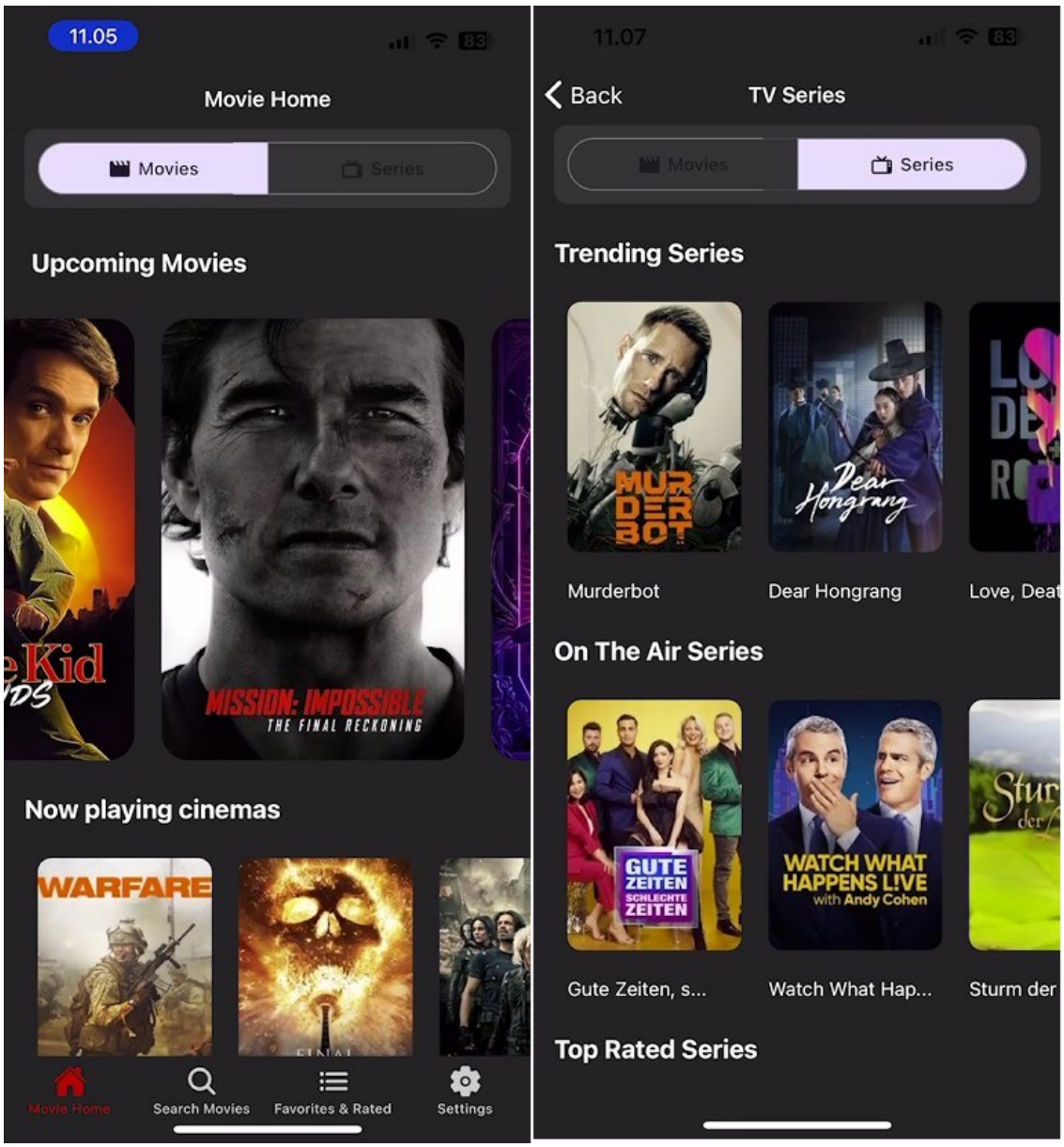
Kun käyttäjä valitsee vierailijanäkymän, on hänellä rajoitetumpi mahdollisuus käyttää sovellusta, kuin kirjautuneella. Vierailijatilassa käyttäjä pystyy selaamaan elokuvia ja sarjoja, etsimään ohjelmia ja lukemaan tietoja elokuvista ja sarjoista. Osa toiminnoista on estetty, esimerkiksi ohjelmien arvostelu ja suosikkeihin lisääminen. Kirjautuneella käyttäjällä ei ole erillistä aloitusnäkymää vaan sovellus siirtyy etusivulle, jossa näkyvät ajankohtaiset ja suosituimmat elokuvat. Ohessa kuva Vierailijaikkunasta: (KUVA 6)



KUVA 6. Vierailijatilan aloitusnäkymä (Kuva: Mikkonen Riku, 2025).

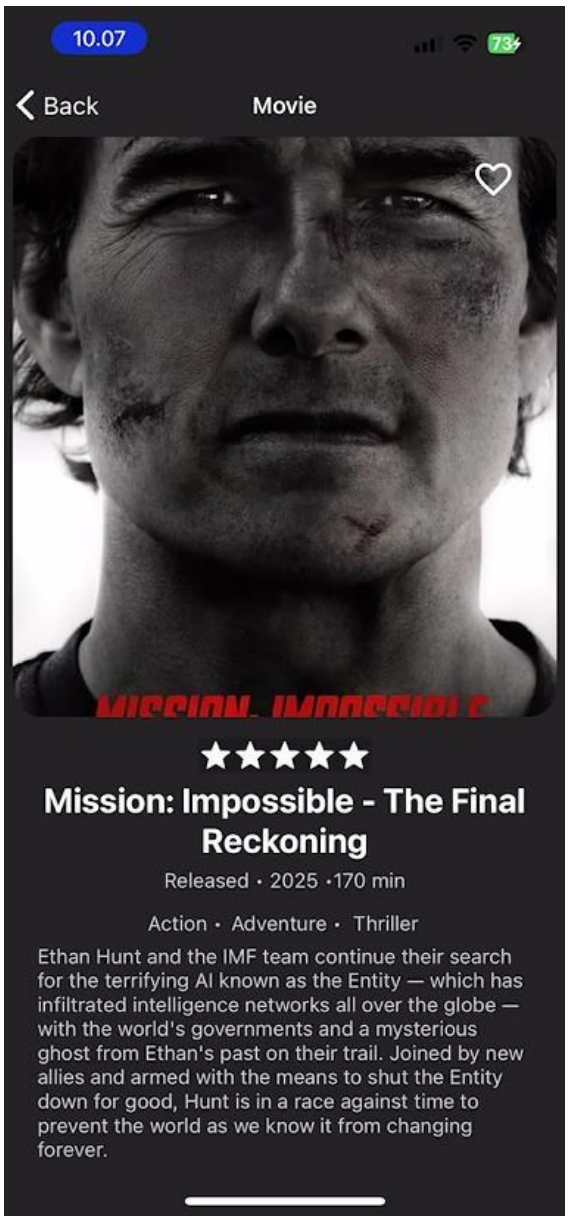
3.3 Sisältöjen tarkastelunäkymä

Kirjautumisen tai vierailijatilan valinnan jälkeen käyttäjä siirtyy sovelluksen etusivulle. Etusivulla näytetään tulevia elokuvia, elokuvateattereissa parhaillaan pyörivät elokuvat, suosituimmat sekä parhaiten arvostelluimmat elokuvat. Etusivun yläreunassa on kaksi painiketta, joiden avulla käyttäjä voi vaihtaa sarjat-sivulle. Sarjat-sivulla esitetään muun muassa suositut sarjat sekä käynnissä olevia sarjat. Ohessa kuva elokuvat ja sarjat-sivusta: (KUVA 7)



KUVA 7. Elokuva (etusivu) ja sarjat-sivut vierekkäin (Kuva: Mikkonen Riku 2025).

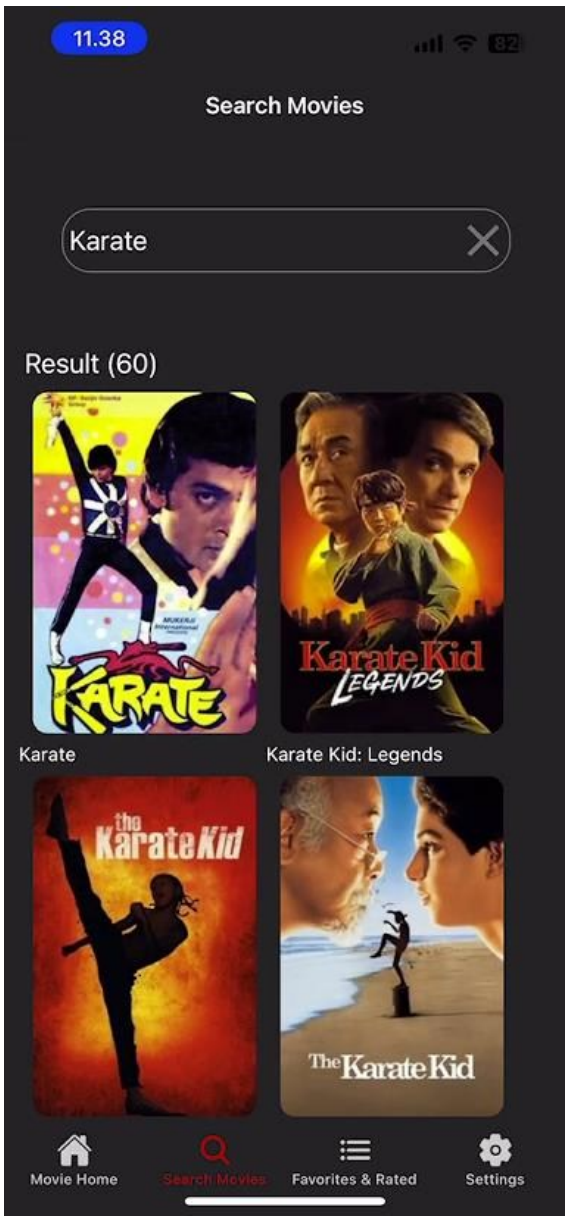
Sisältöä klikkaamalla käyttäjä siirtyy tarkempaan näkymään, jossa on muun muassa elokuvan julkaisuajankohta, elokuvan kuvaus ja näyttelijätiedot. Sama toimintaperiaate koskee myös sarjoja. Tarkemmassa näkymässä on myös ohjelman arvostelu ja suosikkeihin lisäystoiminnot. Ohessa kuva ohjelmien tarkemmasta näkymästä: (KUVA 8)



KUVA 8. Tarkempi näkymä ohjelmista (Kuva: Mikkonen Riku, 2025).

3.4 Haku ja näyttelijätiedot

Movie-App-sovelluksessa on hakutoiminto, jolla voi hakea elokuvia, sarjoja ja näyttelijöitä. Hakutoiminto löytyy elokuvat näkymän alapalkista (Bottom Tabs), jota painamalla käyttäjä siirtyy hakukenttään, jossa on kirjoituskenttä. Hakusanaa kirjoitettaessa sovellus listaa automaattisesti hakutulokset alapuolelle. Hakutulokset esitetään visuaalisena näkymänä, jossa on kuva sisällöstä, jota klikkaamalla siirtyy sisällön tarkempaan näkymään. Ohessa kuva hakunäkymästä: (KUVA 9)



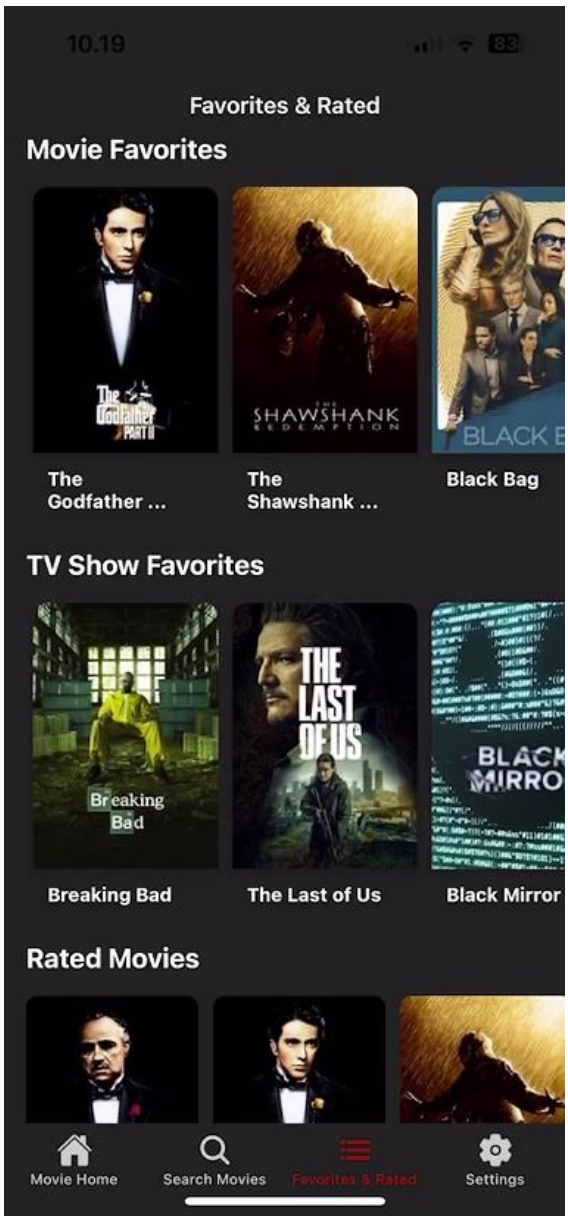
KUVA 9. Näkymä hakunäkymästä sovelluksessa (Kuva: Mikkonen Riku, 2025).

Näyttelijänäkymään voi siirtyä joko hakutoiminnon avulla tai ohjelmien tarkemmasta tietonäkymän kautta. Näkymässä esitetään perustiedot kuten syntymävuosi, elämänkerta, sekä luettelo elokuvista ja sarjoista, jossa hän on näytellyt. Ohessa kuva näyttelijänäkymästä: (KUVA 10)

KUVA 10. Kuva sovelluksen näyttelijänäkymä (Kuva: Mikkonen Riku, 2025).

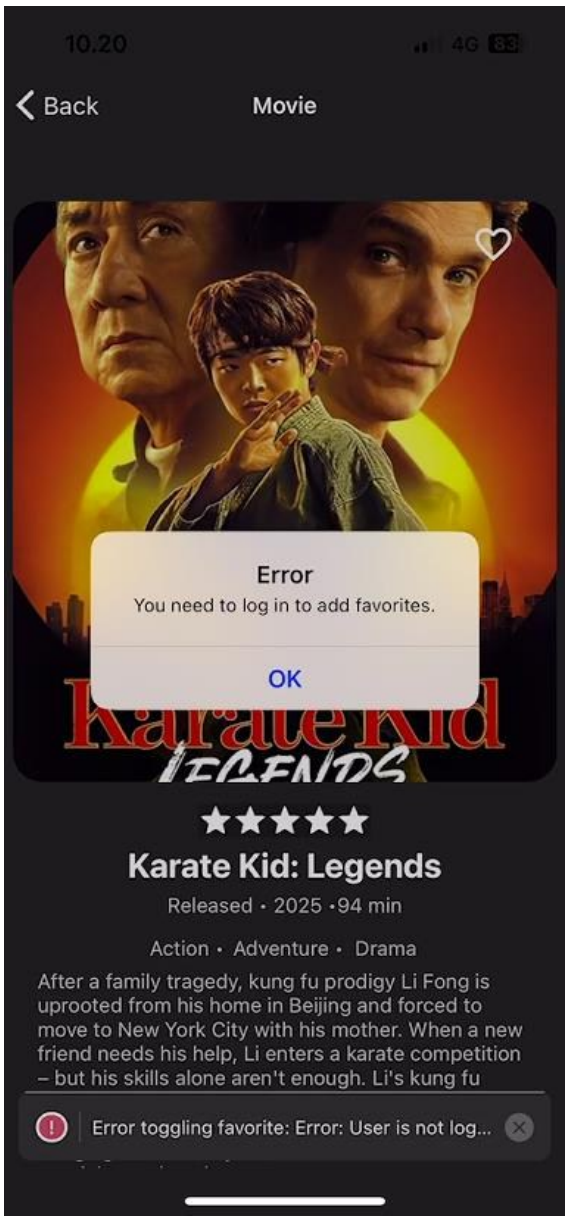
3.5 Suosikkien ja arvostelujen lisäys ja poisto

Suosikkien ja arvostelujen lisäys onnistuu ohjelmien tarkemmasta näkymästä. Arvostelut annetaan valitsemalla tähtimäärä, jolloin arvostelut tallentuvat käyttäjän tietoihin. Suosikin voi lisätä painamalla julisteen yläkulmasta olevasta sydänpainikkeesta. Sekä suosikkien ja arvostelut löytyvät omalta sivultaan, johon pääsee etusivun alareunan navigointipalkin kautta (Bottom Tabs). Ohessa näkymä, jossa käyttäjän suosikit ja arvostellut sisällöt on koottu yhteen listaan: (KUVA 11)



KUVA 11. Käyttäjän näkymä suosikki, arvostelluista elokuvista ja sarjoista (Kuva: Mikkonen Riku, 2025).

Henkilökohtaiset toiminnot, kuten suosikkien ja arvostelujen lisääminen ovat käytettävissä vain kirjautuneella käyttäjällä. Mikäli käyttäjä ei ole kirjautunut ja yrittää lisätä arvostelua tai tallentaa suosikkeihin sovellus näyttää ponnahdusikkunan, joka kehottaa kirjautumaan sisään. Ohessa kuva ponnahdusikkunan, joka näyttää kyseisen kehotuksen: (KUVA 12)



KUVA 12. Kuva kehotuksesta kirjautua, jos haluaa lisätä ohjelman suosikkeihin (Kuva: Mikkonen Riku, 2025).

3.6 Kohderyhmä

Vaikka Movie-App on tehty harjoitustyönä, sen mahdollisessa tuotantokäytössä kohderyhmänä olisivat kaikenikäiset elokuvista ja sarjoista kiinnostuneet henkilöt. Erityisesti nuoret aikuiset, jotka käyttävät aktiivisesti suoratoistopalveluita ja etsivät helppoa tapaa löytää uutta katsottavaa, muodostaisivat todennäköisesti suurimman kohderyhmän.

3.7 Käyttötarpeet ja tavoitteet

Movie-App sovelluksen käyttötarve korostuu erityisesti tilanteissa, jossa käyttäjä ei tiedä mitä elokuvaa tai sarjaa katsoisi seuraavaksi. Sovellus tarjoaa tähän apua ehdottamalla tällä hetkellä hyvin arvosteltuja ja suosiossa olevia elokuvia tai sarjoja. Käyttötarve ilmenee tilanteissa, jos käyttäjä haluaa etsiä tietoja näyttelijöistä tai lisätä elokuvia ja sarjoja omiin suosikkeihin, sekä arvostella ohjelmia mitä on katsonut.

Opinnäytetyön yhtenä tavoitteena oli tarjota helppokäyttöinen. Toinen tavoite oli luoda sovellus, joka auttaa elokuvan katsojia löytämään uusia elokuvia, sarjoja sekä näyttelijätietoja saman sovelluksen kautta. Tarkoituksena on tarjota vaivaton tapa etsiä ohjelmia ja näyttää, mistä suoratoistopalvelusta ohjelma on katsottavissa. Tavoitteena on myös tarjota helppokäyttöinen sovellus, jonka kautta on helppo tapa arvostella ja tallentaa ohjelmia suosikkeihin.

4 TEKNINEN TOTEUTUS

Tässä luvussa käydään läpi Movie-App-sovelluksen tekninen toteutus. Luvussa esitellään sovelluksen rakennetta Backend-puolelta ja mitä teknologioita on käytetty. Lisäksi tarkastellaan, kuinka eri ominaisuudet on ohjelmoitu ja miten ne toimivat teknisestä näkökulmasta.

4.1 Sovelluksen rakenne ja navigointi

Movie-App-sovelluksen navigaatio on toteutettu hyödyntäen React Navigation-kirjastoa. Alapalkki on rakennettu käyttäen Bottom Tabs-komponenttia ja se näkyy päänäkymissä, kuten etusivu, haku, suosikit, arvostelut ja asetukset. Alapalkin avulla käyttäjä voi liikkua näiden näkymien välillä järkevästi. Ohessa kuva alapalkin koodista: (KUVA 13)

```
return (
  <Tab.Navigator
    screenOptions={({ route }) => ({
      tabBarIcon: ({ color, size }) => {
        let iconName;

        if (route.name === "Home") {
          iconName = "home";
        } else if (route.name === "MyLists") {
          iconName = "list";
        } else if (route.name === "Search") {
          iconName = "search";
        } else if (route.name === "Settings" || route.name === "Login") {
          iconName = isGuest ? "log-in" : "settings";
        }

        return <Ionicons name={iconName} size={size} color={color} />;
      },
      tabBarActiveTintColor: Colors.buttonColor,
      tabBarInactiveTintColor: Colors.status,
      headerShown: true,
      tabBarStyle: BottomTabsStyles.tabBar,
      headerStyle: BottomTabsStyles.header,
      headerTintColor: Colors.white,
    })}
  >
    <Tab.Screen
```

KUVA 13. Alapalkin (Bottom Tabs) toteutus käyttäen Tab.Navigator-komponenttia. Kuvassa määritellään eri näkymien ikonit ja tyylit (Kuva: Mikkonen Riku, 2025).

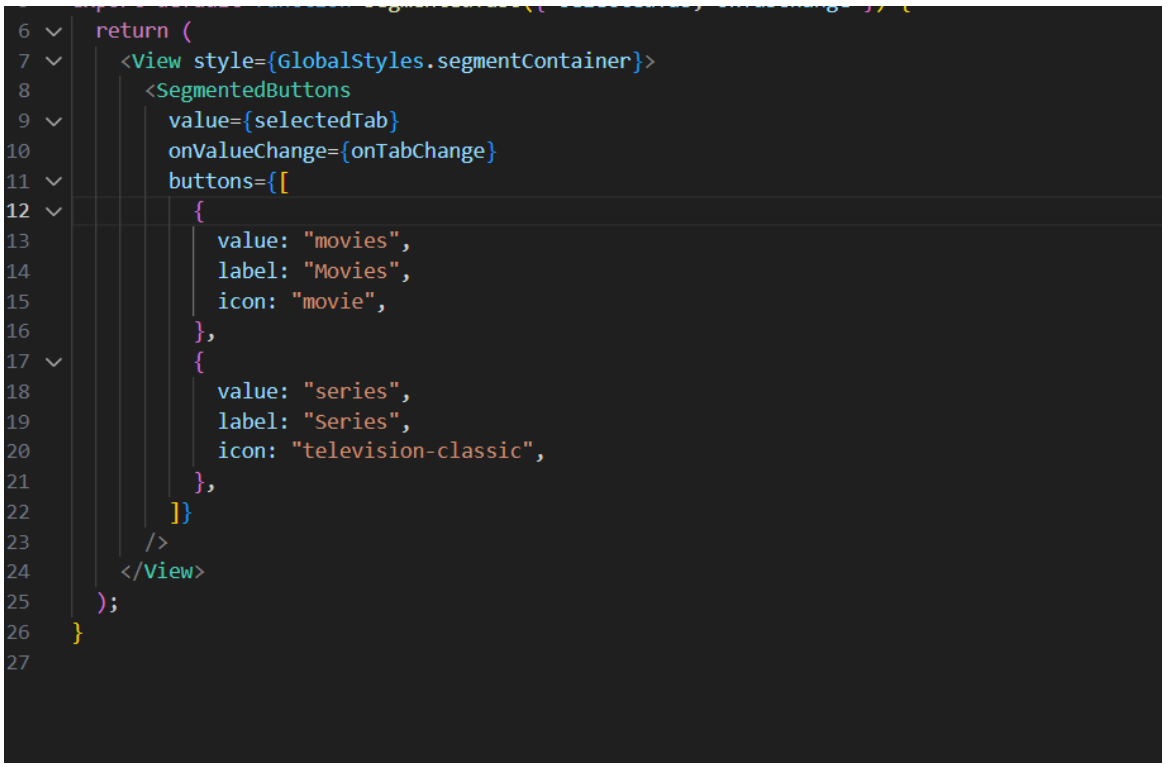
Muut siirtymät, kuten ohjelmien tarkemmat sivut tai näyttelijätiedot avautuvat Stack Navigation avulla. Näissä näkymissä alapalkki ei ole näkyvissä ja navigointi

edelliseen näkymään tapahtuu käyttöliittymästä löytyvän takaisinpainikkeen avulla. Ohessa kuva Stack Navigator-koodista: (KUVA 14)

```
    }}
  />
  <Stack.Screen
    name="MainTabs"
    component={BottomTabs}
    options={{ headerShown: false }}
  />
  <Stack.Screen
    name="Movie"
    component={MovieScreen}
    options={{
      headerBackTitle: "Back",
      title: "Movie",
      headerShown: true,
    }}
  />

  <Stack.Screen
    name="Person"
    component={PersonScreen}
    options={{
      headerBackTitle: "Back",
      title: "Person",
      headerShown: true,
    }}
  />
  <Stack.Screen
    name="Series"
    component={SeriesHomeScreen}
    options={{
      headerBackTitle: "Back",
      title: "TV Series",
      headerShown: true,
    }}
```

(KUVA 14). Näkymien välinen siirtymä toteutettu Stack.Screen-rakenteella. Tätä käytetään ohjelmasivuilla ja näyttelijäsivulla (Kuva: Mikkonen Riku, 2025).

Elokuvien ja sarjat-sivun välillä käytetään Segmented Buttons-komponenttia, jossa on kaksi painiketta elokuvat (Movies) ja sarjat (Series). Painikkeilla käyttäjä voi vaihtaa näkymää helposti sisällön mukaan ilman sivun vaihtamista. Ohessa kuva Segmented Buttons koodista: (KUVA 15)

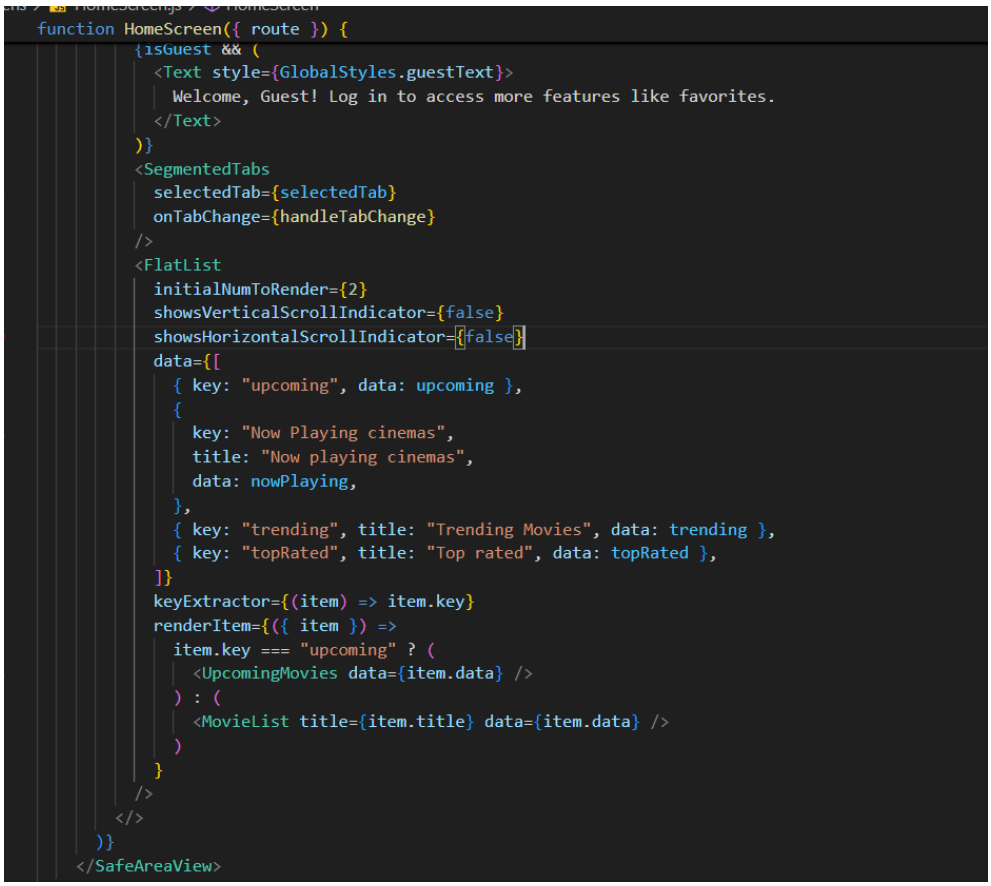


```
return (
  <View style={GlobalStyles.segmentContainer}>
    <SegmentedButtons
      value={selectedTab}
      onValueChange={onTabChange}
      buttons={[
        {
          value: "movies",
          label: "Movies",
          icon: "movie",
        },
        {
          value: "series",
          label: "Series",
          icon: "television-classic",
        },
      ]}
    />
  </View>
);
}
```

(KUVA 15). Elokuvien ja sarjojen siirtyminen toteutettu käyttäen Segmented Buttons-komponenttia (Kuva: Mikkonen Riku, 2025).

4.2 Etusivun listaus

Etusivun elokuvien ja sarjojen listaus on toteutettu käyttäen FlatList-komponenttia, joka jakaa sisällön eri kategorioihin, kuten tulevat elokuvat (Upcoming Movies) ja suosituimmat elokuvat (Trending Movies). Jokaiselle kategoriatyypille FlatList renderöi vaaka-asentoiseen kuvakarusellin, jossa yksittäiset elokuvat esitetään visuaalisesti. Renderöinti tapahtuu komponenttien UpcomingMovies ja MovieList kautta. Sarjat sivulla käytetään samaa toteutustapa, mutta tiedot haetaan SeriesList-komponentin kautta, jolloin FlatList esittää sarjat vastaavalla tavalla visuaalisesti. Ohessa kuva elokuva näkymän koodista: (KUVA 16)



```
function HomeScreen({ route }) {
          {isGuest && (
            <Text style={GlobalStyles.guestText}>
              Welcome, Guest! Log in to access more features like favorites.
            </Text>
          )}
          <SegmentedTabs
            selectedTab={selectedTab}
            onTabChange={handleTabChange}
          />
          <FlatList
            initialNumToRender={2}
            showsVerticalScrollIndicator={false}
            showsHorizontalScrollIndicator={false}
            data={[
              { key: "upcoming", data: upcoming },
              {
                key: "Now Playing cinemas",
                title: "Now playing cinemas",
                data: nowPlaying,
              },
              { key: "trending", title: "Trending Movies", data: trending },
              { key: "topRated", title: "Top rated", data: topRated },
            ]}
            keyExtractor={(item) => item.key}
            renderItem={({ item }) =>
              item.key === "upcoming" ? (
                <UpcomingMovies data={item.data} />
              ) : (
                <MovieList title={item.title} data={item.data} />
              )
            }
          />
        </>
      )}
    </SafeAreaView>
```

KUVA 16. FlatList-komponentin toteutus elokuva näkymässä (Kuva: Mikkonen Riku, 2025).

FlatList-komponentti saa tietonsa API-kutsujen kautta, jotka tehdään esimerkiksi ApiParsing.js-tiedoston funktiolla kuten fetchUpcoming ja fetchtrending. Näillä funktiolla haetaan eri kategorioiden ohjelmatiedot TMDB-rajapinnasta ja ne välitetään FlatList-komponentille esitettäväksi käyttöliittymässä. Sarjat sivulla tietojen haku toimii samalla periaatteella, mutta siinä käytetään omia funktioita sarjojen tietojen hakemiseen. Ohessa kuva hakufunktiosta: (KUVA 17)

```
const getUpcomingMovies = async () => {
  const data = await fetchUpcoming();
  if (data?.results) setUpcoming(data.results);
  setLoading(false);
};

const getTrendingMovies = async () => {
  const data = await fetchTrending();
  if (data?.results) setTrending(data.results);
};

const getRatedMovies = async () => {
  const data = await fetchRated();
  if (data?.results) setRated(data.results);
};
```

KUVA 17. Elokuvatietojen hakufunktiot, jotka käyttävät TMDB-rajapintaa tiedon noutamiseen käyttöliittymään (Kuva: Mikkonen Riku, 2025).

4.3 Hakutoiminto

Sovelluksen hakutoiminto on toteutettu käyttämällä TextInput-komponenttia, johon on liitetty onChangeText-tapahtuma. Tämä tallentaa käyttäjän antaman hakusanan tilamuuttujaan ja käynnistää haun viiveellä (debounce), jotta turhilta API-kutsuilta vältytään. Ohessa kuva TextInput-komponentista: (KUVA 18)

```
const handleTextDebounce = useCallback(debounce(handleSearch, 400), []);
return (
  <View style={SearchStyles.container}>
    <View style={SearchStyles.search}>
      <TextInput
        style={SearchStyles.textinput}
        placeholder="Search"
        onChangeText={handleTextDebounce}
        placeholderTextColor="white"
      />
      <TouchableOpacity onPress={() => navigation.navigate("Home")}>
        <MaterialIcons style={SearchStyles.icon} size={38} name="close" />
      </TouchableOpacity>
    </View>
```

KUVA 18. TextInput-komponentti, joka yhdistyy debounce-tekniikkaan käyttäjän syötteen perusteella tehtävien hakujen optimoimiseksi. Sulkemisnappi palauttaa käyttäjän kotinäkymään (Kuva: Mikkonen Riku, 2025).

Kun käyttäjä kirjottaa hakusanan, sovellus suorittaa samanaikaisesti kolme API-kutsua TMDB-rajapintaan. Näissä käytetään erillisiä funktioita: searchMovies, searchSeries ja searchPeople, jotka hakevat vastaavat tulokset elokuvista, sarjoista ja henkilöistä. Ohessa kuva handleSearch-funktiosta: (KUVA 19)

```
const handleSearch = async (value) => {
  if (value && value.length > 2) {
    setLoading(true);

    try {
      const [moviesData, seriesData, peopleData] = await Promise.all([
        searchMovies({
          query: value,
          include_adult: "false",
          language: "en-US",
          page: "1",
        }),
        searchSeries({
          query: value,
          include_adult: "false",
          language: "en-US",
          page: "1",
        }),
        searchPeople({
          query: value,
          include_adult: "false",
          language: "en-US",
          page: "1",
        }),
      ]);
```

KUVA 19. Esimerkki handleSearch-funktiosta, jossa suoritettaan kolme API-kutsua samanaikaisesti (Kuva: Mikkonen Riku, 2025).

Haun tulokset yhdistetään yhdeksi listaksi, jossa jokaiselle tulokselle lisätään media_type-kenttä (esim. movie, tv, person). Lista lajitellaan siten, että täsmälliset ja hakusanan sisältävät osumat nousevat listan kärkeen. Ohessa esimerkkikuva koodista: (KUVA 20)

```
const results = [
  ...(moviesData?.results || []).map((item) => ({
    ...item,
    media_type: "movie",
  })),
  ...(seriesData?.results || []).map((item) => ({
    ...item,
    media_type: "tv",
  })),
  ...(peopleData?.results || []).map((item) => ({
    ...item,
    media_type: "person",
  })),
];

const sortedResults = results.sort((a, b) => {
  const aTitle = (a.title || a.name || "").toLowerCase();
  const bTitle = (b.title || b.name || "").toLowerCase();
  const searchValue = value.toLowerCase();

  const aExactMatch = aTitle === searchValue ? 1 : 0;
  const bExactMatch = bTitle === searchValue ? 1 : 0;

  if (aExactMatch !== bExactMatch) {
    return bExactMatch - aExactMatch;
  }

  const aIncludes = aTitle.includes(searchValue) ? 1 : 0;
  const bIncludes = bTitle.includes(searchValue) ? 1 : 0;

  if (aIncludes !== bIncludes) {
    return bIncludes - aIncludes;
  }
  return (b.popularity || 0) - (a.popularity || 0);
});
```

KUVA 20. Esimerkkikuva haun tuloksien yhdistämiseksi yhdeksi listaksi ja lajitellaan niin, että täsmälliset ja hakusanan sisältävät osumat tulevat kärkeen (Kuva: Mikkonen Riku, 2025).

Kun käyttäjä klikkaa hakutulosta, TouchableWithoutFeedback-komponentin onPress-tapahtuma käynnistää siirtymisen tarkempaan näkymään. Navigointi toteutetaan navigation.push-funktioilla, joka ohjaa oikeaan näkymään sisällön tyypin mukaan (esimerkiksi elokuvan, sarjan tai näyttelijän sivulle). Ohessa kuva koodista: (KUVA 21)

```
<TouchableWithoutFeedback
  onPress={() => {
    if (item.media_type === "movie") {
      navigation.push("Movie", item);
    } else if (item.media_type === "tv") {
      navigation.push("SeriesDetails", item);
    } else if (item.media_type === "person") {
      navigation.push("Person", item);
    }
  }}
>
  <View>
```

KUVA 21. Kuva esimerkki koodista, jossa toteutetaan siirtymä ohjelman sivulle (Kuva: Mikkonen Riku, 2025).

4.4 Suosikkien ja arvostelujen lisääminen ja poisto

Suosikkien lisäys ja poisto tapahtuu elokuvien ja sarjojen tarkemassa näkymässä. Näkymän avautuessa fetchFavoriteStatus-funktio tarkistaa onko ohjelma jo valmiiksi lisätty suosikkeihin. Kun käyttäjä lisää tai poistaa ohjelman suosikeista, handleToggleFavorite-funktio lähettää API-kutsun ja näyttää käyttäjälle ilmoituksen toimenpiteen onnistumisesta. Alla oleva kuva havainnollistaa toimintaa: (KUVA 22)

```
const fetchFavoriteStatus = async (movieId) => {
  try {
    const favorites = await fetchFavorites();
    const isMovieFavorite =
      favorites.movies?.some((favorite) => favorite.id === movieId) || false;
    setIsFavorite(isMovieFavorite);
  } catch (error) {
    console.error("Error fetching favorite status:", error);
  }
};
const handleToggleFavorite = async () => {
  try {
    const newFavoriteStatus = !isFavorite;
    const response = await toggleFavorite(
      movie.id,
      newFavoriteStatus,
      "movie",
    );
    Alert.alert(
      "Favorites",
      newFavoriteStatus
        ? `${movie.title} has been added to your favorites!`
        : `${movie.title} has been removed from your favorites!`,
    );
    if (response.success) {
      setIsFavorite(newFavoriteStatus);
    } else {
      console.error("Failed to toggle favorite:", response);
    }
  } catch (error) {
    console.error("Error toggling favorite:", error);
  }
};
return (
```

KUVA 22. Kuvassa näkyy, kuinka fetchFavoriteStatus tarkistaa suosikkitilanteen näkymän avautuessa ja handleToggleFavorite reagoi käyttäjän painallukseen päivittämällä suosikkitiedon (Kuva: Mikkonen Riku, 2025).

Myös arvostelut annetaan ohjelmien tarkemassa näkymässä. Arvostelujen lisäys tai poisto on toteutettu omassa CustomRating-komponentissa, joka näyttää tähtiarvostelun ja hakee mahdollisen käyttäjän aiemman arvostelun TMDB-rajapinnasta. (getRating tai getTVRating). Ohessa kuva kyseisistä toiminnoista: (KUVA 23)

```
  const fetchRating = async () => {
    if (sessionId && id) {
      const savedRating =
        type === "movie"
          ? await getRating(id, sessionId)
          : await getTvRating(id, sessionId);
      setRating(savedRating / 2);
    }
  };

  fetchRating();
}, [id, sessionId, type]);

const handleRating = async (value) => {
  if (!sessionId) {
    Alert.alert("Error", "You need to be logged in to rate.");
    return;
  }

  try {
    const scaledRating = value * 2;
    if (value === 0) {
      await handleRemoveRating();
      return;
    }
```

KUVA 23. Kuvassa näkyy, kuinka fetchRating-funktio hakee vanhat arvostelut ja handleRating-funktio lähettää käyttäjän antaman arvostelun (Kuva: Mikkonen Riku, 2025).

Kun käyttäjä antaa uuden arvosanan handleRating-funktio lähettää uuden arvion TMDB:lle käyttäen submitRating- tai submitTvRating-funktiota. Jos käyttäjä valitsee arvosanaksi 0, kutsutaan handleRemoveRating-funktioita, joka poistaa arvostelun. Arvosana näytetään asteikolla 0–10, mutta käyttäjä valitsee sen tähtinä (1–5 tähteä).

4.5 Näyttelijätiedot

Näyttelijän tietonäkymä on toteutettu PersonScreen-komponentissa, joka avautuu omassa näkymässä, kun käyttäjä valitsee henkilön sovelluksessa. Komponentti hakee henkilön tiedot tekemällä kolme erillistä API-kutsua: fetcPersonDetails, fetchPersonMovies ja fetchPersonSeries, jotka noutavat henkilön perustiedot, elokuvaroolit ja tv-sarjarooleihin liittyvät tiedot.

Haetut tiedot tallennetaan komponentin tilaan useState-hookien avulla. UseState on Reactin tarjoama funktio, jonka avulla komponentti säilyttää ja päivittää tietoja tilamuuttujassa sovelluksen käytön aikana. Elokuvat ja sarjaroolit esitetään flatList-komponentin avulla visuaalisessa listamuodossa.

Henkilön perustiedot kuten: nimi, sukupuoli ja syntymäpäivä näytetään omissa tekstikentissä ja elämänkerta (biography) esitetään omassa osiossaan. Elokuvat - ja sarjarooleihin liittyvät tiedot näytetään omissa MovieList - ja SerieList-komponenteissa. Ohessa kuva koodista: (KUVA 24)

```
        }}
      />
    </View>
  </View>
  <View style={PersonStyles.titleContainer}>
    <Text style={PersonStyles.title}>{person?.name}</Text>
    <Text style={PersonStyles.textStatus}>
      {person?.place_of_birth}
    </Text>
  </View>
  <View style={PersonStyles.centerContainer}>
    <View style={PersonStyles.textContainer}>
      <Text style={PersonStyles.centreText}>Gender</Text>
      <Text style={PersonStyles.centres}>
        {person?.gender === 1 ? "female" : "Male"}
      </Text>
    </View>

    <Divider style={PersonStyles.divider} />
    <View style={PersonStyles.textContainer}>
      <Text style={PersonStyles.centreText}>Birthday</Text>
      <Text style={PersonStyles.centres}>{person?.birthday}</Text>
    </View>
```

KUVA 24. Henkilön perustietojen esittely erillisinä tekstikenttinä (Kuva: Mikkonen Riku, 2025).

4.6 Tiedon haku ja käsittely

Tietojenhaku tapahtuu keskitetysti ApiParsing.js-tiedostossa määritetyillä funktioilla, jotka muodostavat valmiita pyyntöjä TMDB-rajapintaan. Esimerkiksi tulevien, suosittujen ja parhaiten arvosteltujen elokuvien tiedot haetaan kutsumalla funktioita: fetchUpcoming, fetchTrending ja fetchRated. Näissä funktioissa hyödynnetään Axiois-kirjastoa ja API-avaimen käsittely hoidetaan expo-constants-kirjaston avulla turvallisesti ympäristömuuttujista. Jokainen kutsu palauttaa tiedon, joka tallennetaan komponenttien tilaan (useState) ja esitetään käyttöliittymässä. Ohessa kuva toiminnoista: (KUVA 24)

```
};

export const fetchUpcoming = () => {
  return apiCall(upcoming);
};

export const fetchTrending = () => {
  return apiCall(trendingMovie);
};

export const fetchRated = () => {
  return apiCall(topRated);
};
```

KUVA 24. Kuvassa esitetään, kuinka funktiot hakevat tiedot TMDB-rajapinnasta (Kuva: Mikkonen Riku, 2025).

4.7 Kirjautuminen

Sovelluksen kirjautumislogiikka toteutetaan LoginScreen-komponentissa. Kirjautumisen yhteydessä sovellus hakee TMDB-rajapinnasta kertakäyttötunnuksen käyttäen fetchRequestToken-funktiota ja validateWithLogin-funktio tarkistaa onko kirjautumistunnus oikea. Kirjautumisen yhteydessä createSession-funktio luo session_id:n ja tallentaa sen paikallisesti AsyncStorage-kirjaston avulla myöhempää käyttöä varten. Ohessa kuva kirjautumistoiminnoista: (KUVA 25)

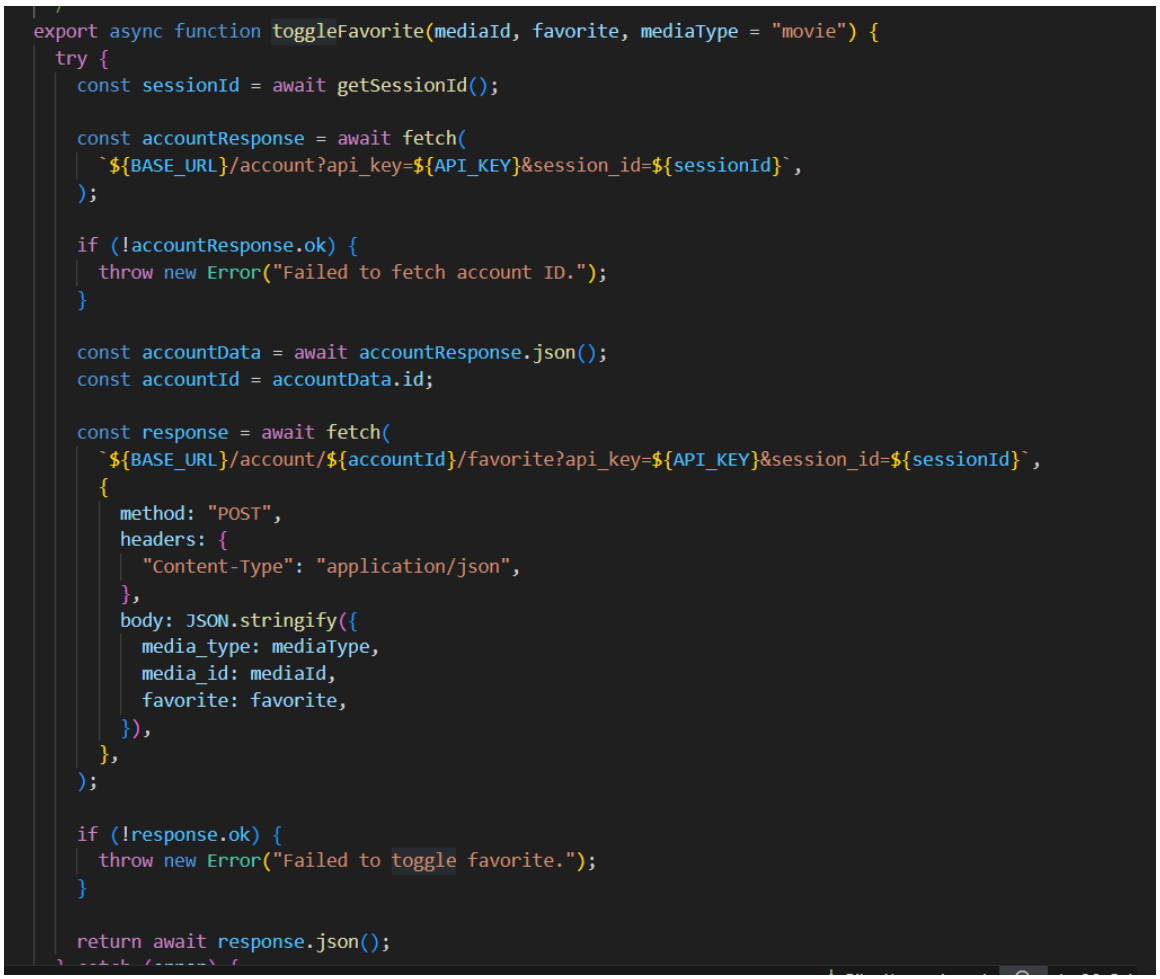
```
const handleLogin = async () => {
  try {
    const tokenData = await fetchRequestToken();
    if (!tokenData.success) {
      alert("Failed to get request token");
      return;
    }

    const loginData = await validateWithLogin(
      username,
      password,
      tokenData.request_token,
    );
    if (!loginData.success) {
      Alert.alert("Invalid username or password");
      return;
    }

    const sessionData = await createSession(tokenData.request_token);
    if (!sessionData.success) {
      Alert.alert("Failed to create session");
      return;
    }

    await AsyncStorage.setItem("session_id", sessionData.session_id);
```

KUVA 25. Kuva handeLogin-funktiosta, joka hoitaa kirjautumisen (Kuva: Mikkonen Riku, 2025).

Käyttäjän tunnistus perustuu tähän sessiontunnukseen (session_id), joka on välttämätön tietyissä toiminnoissa, kuten arvostelujen antamisessa ja suosikkien tallentamisessa omiin tietoihin. Nämä toiminnot käyttävät sessiotunnusta varmistaakseen, että toiminnot kohdistuvat oikeaan käyttäjään.

4.8 Suosikkien ja arvostelujen hallinta

Suosikkien ja arvostelujen haku ja lisäys tapahtuu omien tiedostojen kautta. Suosikkitoiminnossa hyödynnetään toggleFavorite-funktiota, jolla voidaan lisätä ja poistaa ohjelmia suosikeista. Suosikit haetaan käyttäen fetchFavorites-funktiota, joka palauttaa erikseen elokuvat ja sarjat käyttäjän suosikkilistalle. Molemmat funktiot käyttävät getSessionId-funktiota noutaakseen käyttäjän session-id:n AsyncStoragesta ja varmistaakseen onko käyttäjä kirjautunut sisään. Ohessa kuva edellä mainituista toiminnoista: (KUVA 26)



```
export async function toggleFavorite(mediaId, favorite, mediaType = "movie") {
  try {
    const sessionId = await getSessionId();

    const accountResponse = await fetch(
      `${BASE_URL}/account?api_key=${API_KEY}&session_id=${sessionId}`,
    );

    if (!accountResponse.ok) {
      throw new Error("Failed to fetch account ID.");
    }

    const accountData = await accountResponse.json();
    const accountId = accountData.id;

    const response = await fetch(
      `${BASE_URL}/account/${accountId}/favorite?api_key=${API_KEY}&session_id=${sessionId}`,
      {
        method: "POST",
        headers: {
          "Content-Type": "application/json",
        },
        body: JSON.stringify({
          media_type: mediaType,
          media_id: mediaId,
          favorite: favorite,
        }),
      },
    );

    if (!response.ok) {
      throw new Error("Failed to toggle favorite.");
    }

    return await response.json();
```

KUVA 26. Kuva toggleFavorite-funktiosta (Kuva: Mikkonen Riku, 2025).

Arvioiden lähetys, haku ja poisto tapahtuu omilla erillisillä funktioillaan. Esimerkiksi submitRating-funktiolla käyttäjä voi lähettää oman arvostelun valitsemalle elokuvalle ja getRating-funktiolla haetaan, onko käyttäjä jo arvostellut kyseisen elokuvan. Molemmat funktiot hyödyntävät käyttäjän session_id:ttä varmistuakseen, että käyttäjä on oikea. Vastaavat funktiot löytyvät myös sarjojen käsittelyyn.

5 POHDINTA

Työn tekemisen aikana opin paljon mobiilisovellusten kehittämisestä React Nativella. Aiemmin olin ohjelmoinut mobiilisovelluksen Android-puhelimelle Kotlin-koodikielellä, jolloin käytin suoraan Androidin natiiveja komponentteja. React native-projektissa opin kuitenkin kokonaan uudenlaisen alustariippumattoman tavan rakentaa sovelluksia Expo-kehitysympäristössä. Kehitystyössä hyödynsin valmiita kirjastoja muun muassa mobiilinavigaation kuten Stack ja Tab navigation sekä kolmannen osapuolen rajapinnan TMDB API käyttämiseen. Nämä toiminnot ovat olemassa myös natiivialustoilla, mutta React Nativessa niiden hyödyntäminen tapahtuu usein erillisen kirjaston kautta.

Erityisen hyödylliseksi ja opettavaiseksi koin kirjautumistoiminnon toteuttamisen sekä sen, että sain luotua kirjautuneelle käyttäjälle mahdollisuuden arvostella elokuvia ja sarjoja ja tallentaa niitä suosikkeihin. Näiden toimintojen toteuttaminen auttoi ymmärtämään sessiohallintaa ja tietojen tallentamista AsyncStorageen. Opin myös jakamaan koodin selkeämmin pienempiin komponentteihin, mikä paransi koodin luettavuutta ja helpotti ylläpitoa.

Olen tyytyväinen projektin lopputulokseen ja koen työn olleen hyödyllinen käytännön ohjelmointitaitojen sekä ongelmaratkaisun kannalta. Jatkossa voisin kehittää sovellukseen esimerkiksi tummatilaominaisuuden ja profiilinäkymän. Kaiken kaikkiaan tämä työ vahvisti kiinnostustani mobiilikehitykseen ja ohjelmointiin yleisesti ja antoi konkreettisen näytön osaamisestani, jota voin hyödyntää esimerkiksi työnhaussa.

Sovelluksen kehittäminen tarjosi mahdollisuuden yhdistää käytännön ohjelmointiosaaminen konkreettiseen tarpeeseen. Halusin luoda palvelun, joka tuo elokuvat ja sarjat helposti saataville yhdessä paikassa ja uskon, että tämä toteutus vastaa hyvin kohdekäyttäjien tarpeisiin. Projektin kautta pääsin syventämään osaamistani mobiilikehityksessä, ja lopputulos toimii myös erinomaisena esimerkkinä taidoistani tulevaisuuden projekteja tai työnhakua ajatellen.

LÄHTEET

Expo. 2024b EAS build. Verkkosivu. Viitattu 6.5.2025
https://docs.expo.dev/build/introduction/

Expo. 2025. Develop an app with Expo. Verkkosivu. Viitattu 5.5.2025
https://docs.expo.dev/workflow/overview/

Expo. 2024a Introduction to Expo. Verkkosivu. Viitattu 5.5.2025
https://docs.expo.dev/get-started/introduction/

GitHub. 2022. A global community of developers. Verkkosivu. Viitattu 6.5.2025
https://octoverse.github.com/2022/global-tech-talent

GitHub Docs. n.d. a. About Github and Git. Verkkosivu Viitattu 6.5.2025
https://docs.github.com/en/get-started/start-your-journey/about-github-and-git

GitHub Docs. n.d. b. about branches. Verkkosivu. Viitattu 6.5.2025
https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/about-branches

JSON. n.d. Introducing JSON. Verkkosivu. Viitattu 13.5.2025
https://www.json.org/json-en.html

Kokalitcheva, K. 2016. Quora's Co-Founder Is Back in the Startup Game at Y Combinator. Verkkosivu. Viitattu 5.5.2025
https://fortune.com/2016/08/23/charlie-cheever-y-combinator/

Microsoft. n.d. Visual Studio. Verkkosivu. Viitattu 7.5.2025
https://visualstudio.microsoft.com/#vscode-section

Patel Rushi.2023. Why Choose React Native for App Development. Verkkosivu. Viitattu 30.04.2025.
https://www.mindinventory.com/blog/why-choose-react-native-for-app-development

React Native. 2025a. Getting Started. Verkkosivu. Viitattu 24.4.2025
https://reactnative.dev/docs/getting-started

React Native. 2025b. Create native apps for Android, iOS, and more using React. Verkkosivu, Viitattu 22.5.2025
https://reactnative.dev/

Sakniuk Mikhail, Boduch Adam. 2024. React and React Native. Viitattu 05.05.2025. E-kirja. Vaati kirjautumisen.
https://learning.oreilly.com/library/view/react-and-react/9781805127307/Text/Chapter_15.xhtml#_idParaDest-199

TMDB. 2024. FAQ. Verkkosivu. Viitattu 9.5.2025
https://developer.themoviedb.org/docs/faq

Visual Studio Code. 2025. Tutorial: Get started with Visual Studio Code. Verkkosivu. Viitattu 09.05.2025
https://code.visualstudio.com/docs/getstarted/getting-started