Technical Document: Movies App

Arbnor Rama CSY21054

This is a technical documentation for the project coursework on the class Mobile Computing 2024, we were tasked to create a mobile application using android studio, this application would let users browse a selection of movies from an api provided to us furthermore it would allow the user to favorite movies and browse offline these movies. This document will explain how the backend including API request, database were handled.

1. API

To handle API requests a helper is created `APIRequest ` it utilizes the OkHttp library for network requests and Gson for JSON parsing. This class was created to be modular and be able to be used very easily in different parts of the code.

There are two functions only accessible inside this class called `sendRequest` and `performRequest`, the first one implements the async effect of the request so it happens on the background without blocking the ui, where as the other function performs the real request to the api url and parses the response.

The provided API for the assignment had four endpoint therefore four public methods were implemented respectively to the api endpoints, the provided endpoints are as follow;

"https://app-vpigadas.herokuapp.com/api/movies/demo/",
"https://app-vpigadas.herokuapp.com/api/movies/",
"https://app-vpigadas.herokuapp.com/api/movies/" + id,
"https://app-vpigadas.herokuapp.com/api/movies/demo/" + id

All these functions were implemented to allow easier extension of the application in the future.

2. Offline Functionality

This application was designed to work offline as well as online, so to allow the user to browse movies offline we implemented a database system which stores movie information locally on the users device to be accessed when the device doesn't have an internet connection.

Firstly a movie database was created using the `movieDatabase` file alongside it there were tables implemented for movies and favorite movies using the classes `MovieEntity` and `FavoriteEntity` respectively. To perform CRUD operations DAOs were implemented having some simple functionalities such saving single rows and reading multiple rows.
Using the movieDao on the start of application when the device was connected to the internet the incoming data was stored in the database so it would be accessible later on offline. It is worth mentioning that whenever these DAO functions were called the function calling them was implemented in an async way so it does not block the UI thread.

3. Favorite Movies

The user has the option to favorite a movie once they read the movie's detail page, to implement this these data were needed to be stored locally on the users machine. Implementing this was straightforward, we created a table to store the id of the movies that were favorited and once the user wants to remove a favorite movie it is deleted from the table. FavoriteEntity was used to create the favorite table and the favorteDAO was used to do the CRUD operations on this table, to showcase all the favorite movies there was created a new page called favorites but for the implementation of that is talked about on the "frontend" technical documentation.