**Movie Search App**

**The Movie Browser App is a Flutter application to provide users with an simple way to browse, search, and favorite movies using the TMDb API. The app uses Flutter for the front end, and SQLite for local storage of favorite movies.**

**Architecture**

**1. Models**

**Models represent the data structure of the application. The app includes two primary models:**

**- Movie: Represents the movie data fetched from the TMDb API.**

**- Genre: Represents the genre data fetched from the TMDb API.**

**2. Providers**

**Providers manage the state and business logic of the application. The app uses the Provider package for state management.**

**- MovieProvider: Manages the state of movies, including fetching popular movies, searching for movies, handling favorite movies, and fetching genres.**

**3. Screens**

**Screens represent the different user interfaces of the application. Each screen corresponds to a specific functionality of the app.**

**- MovieListScreen: Displays a list of popular or searched movies.**

**- MovieDetailScreen: Displays detailed information about a selected movie.**

**- FavoriteMoviesScreen: Displays a list of favorite movies saved locally.**

**Flow of Data**

**1. Fetching Movies**

**- When the app starts, MovieProvider fetches a list of popular movies from the TMDb API.**

**- The fetched movies are stored in the MovieProvider state and displayed in the MovieListScreen.**

**2. Searching for Movies**

**- When the user enters a search query, MovieProvider re-queries the TMDb API with the search term.**

**- The search results are stored in the MovieProvider state and displayed in the MovieListScreen.**

**3. Viewing Movie Details**

**- When the user selects a movie from the list, they are navigated to the MovieDetailScreen.**

**- MovieDetailScreen displays detailed information about the selected movie, including title, release date, poster, genres, and overview.**

**4. Favoriting Movies**

**- Users can favorite or unfavorite a movie from both the list view and the details screen.**

**- Favorite movies are stored locally using SQLite.**

**- MovieProvider manages the list of favorite movies and ensures they persist across app restarts.**

**Error Handling**

**- Network Issues: The app checks for network connectivity before making API requests. If there is no internet connection, an error message is displayed.**

**- Empty Results: If the API returns no results for a search query, an appropriate message is displayed.**

**- Loading State: While data is being fetched, a loading indicator is displayed to the user.**

**Folder Structure**

**lib/**

**├── models/**

**│ ├── genre.dart**

**│ └── movie.dart**

**├── providers/**

**│ └── movie\_provider.dart**

**├── screens/**

**│ ├── favorite\_movies\_screen.dart**

**│ ├── movie\_detail\_screen.dart**

**│ └── movie\_list\_screen.dart**

**├── main.dart**