Cinema App Documentation

1. Folder Structure

The project was organized into three folders:

1.1. Pages: Contains the code for the layout of the home page, movie details page, seat selection page and confirmation page.

1.2. Models: Contains the code for classes that are used to pass arguments between pages.

1.3. Utils: Container for constants (api keys, colors) that are used in various parts of the application

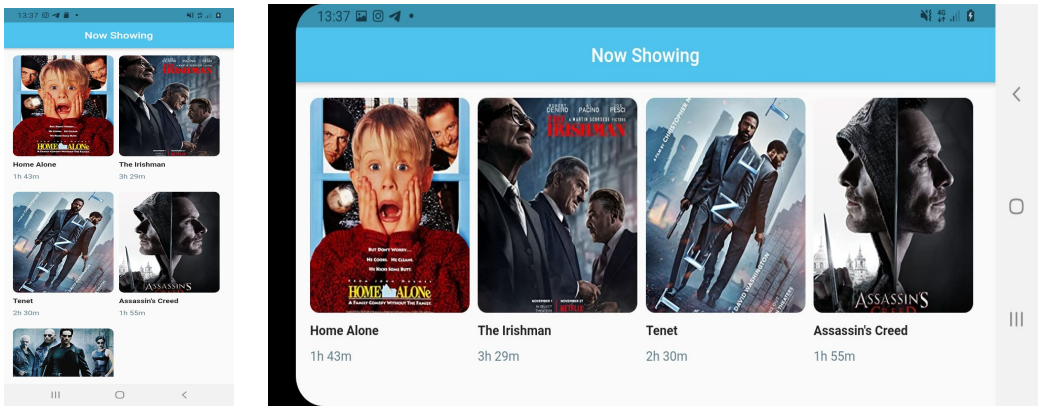
2. Fetching Movie Data

As per the challenge's instructions, movies were fetched from the IMDB database. Since the IMDB API is a premium service, data was instead fetched from the OMDB API, which uses the IMDB database. The OMDB API allows for 1,000 free API calls per day.

5 movies (Home Alone, The Irishman, Tenet, Assassin's Creed, The Matrix) were randomly selected and their IMDB IDs are stored in the Constants.dart file in the utils folder. Their data is fetched asynchronously from the API when the Application loads.

3. User Interface design

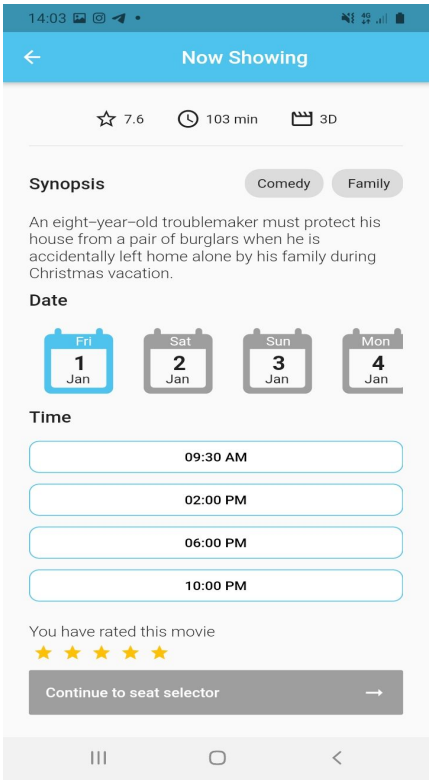
3.1. Home Page layout



The movies were rendered using a GridView widget. Care was taken to maintain an appropriate item aspect ratio for portrait and landscape orientations.

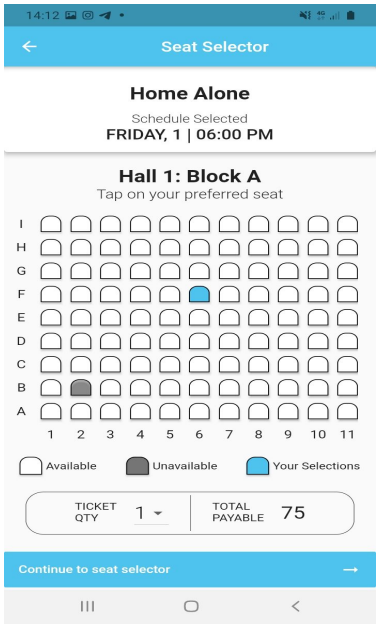
3.2. **Movie Details Page**

The UI for the movie details page mostly conforms to the design specifications provided. The sole exception was the addition of a rating mechanism as that was not included in the instructions. On this page, the user can see various information about the movie (all of which is fetched in real time from IMDB), the available dates this movie is airing on, as well as the times available.

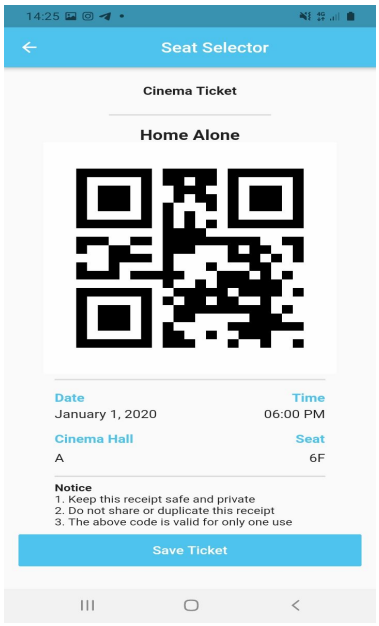


3.3. Seat Selector Page

The Seat selector page was more challenging to implement than the other pages. The most difficult aspect was mapping the seat names (like ‘8A’)to an integer index and vice versa in order to be able to render them appropriately on a GridView and required a bit of mathematical computation to do so. Available seats can be selected until the total number of selected seats exceeds the selected quantity. Selected seats can also be unselected.



3.4. Confirmation Page



The user can reserve the seats that they selected and save the confirmation ticket as an image on this page.

4. Scheduling

A demo schedule for the movies was saved in XML format in schedule.xml. The xml file was initially stored as an asset and is written onto the Application Documents directory (which is the AppData directory in android) when the app is installed for the first time.

The schedule XML file has the following format:

```
<movies>
    <movie title=”The Matrix” time=”04:30 PM” date=”Fri,Jan 15”>
        <reserved seat=”8B” />
        <reserved seat=”2C” />
        <reserved seat=”9K” />
    </movie>
    <movie title=”The Irishman” time = “06:30 PM” date=”Fri,Jan 15”>
        <reserved seat=”1A” />
        <reserved seat=”1B” />
    </movie>
</movies>
```

An XML Parser package was used to parse the XML document, traverse across the XML nodes and extract attributes. To fetch available dates, all <movie> XML elements with title attributes equal to the selected movie were fetched and their time and date attributes were parsed and stored in a Map data structure for constant time lookup. The Map matched a date with all the available time slots and had the following structure:

```
{
    “Fri,Jan 12”: [“09:30 AM”, “10:30 AM”, “12:00 PM”],
    “Sat,Jan 13”: [“11:00 AM”, “7:00 PM”]
    ...
}
```

Once the user selects a time and day and navigates to the seat selector page, the child attributes of the selected XML element are parsed. All reserved seats

are marked on the page and the user cannot interact with them. The user selected seats are displayed in blue and can be unselected.
When the user decides on the seats he/she wants reserved and confirms, new <reserved /> XML elements are created and appended as children of the aforementioned <movie> element. Then the changes are saved in the schedule.xml document.

5. Rating

Ratings were saved in a similar mechanism as the schedule. Initially a ratings xml file with the following format is saved in the Application Documents directory:

```
<ratings>
    <rating movie=”Home Alone” rate=”unrated” />
    <rating movie=”Tenet” rate=”unrated” />
    …
</ratings>
```

When the user navigates to the Movie Details page, ratings.xml is opened and its contents parsed to check whether or not the user has previously rated this movie. If that happens to be the case, the rating value is fetched and displayed. Otherwise, the user is urged to provide a rating if he/she has previously seen the movie. When the user rates a movie, the changes made are immediately saved on the ratings.xml file.