Mobile App Mini Report (UFCF7H-15-3)

Movie Ticket Booking App

Advisor: Prof. Le Duy Tan. Group Name: FlickFanatic

Student 1 Name: Phan Cong Bao

Student 1 Number: 24072184 (HCMIU ID: ITITWE21004)

Student 2 Name: Dao Ngoc Linh

Student 2 Number: 24072180 (HCMIU ID: ITITWE18031)

Student 3 Name: Le Thanh Phuong Nam

Student 3 Number: 23083609 (HCMIU ID: ITITWE19025)

GitHub link: https://github.com/linhmt13113/Booking-films

Demonstration video link: https://www.youtube.com/watch?v=ozVpHbQsl3M

Table of Contents

Contribution and Configuration

Member Contributions:

Name	Phan Cong Bao	Le Thanh Phuong Nam	Dao Ngoc Linh
Percent	33%	35%	32%
Signature	[illegible] Phan Cong Bao	[illegible] Lê Thanh Phương Nam	[illegible] Dao Ngoc Linh

Member 1: Dao Ngoc Linh:

I was to design and implement several key screens for the application. My other core responsibilities included allowing the user to register easily, making sure that every input field is correct and functions properly. I was also managing user authentication and logging in, thus allowing users to sign in with ease. I also designed an introduction screen that would route the user either to the login or registration page, depending on the user's needs.

I did the initial schema design for the database management of users. I made the first model that handled user accounts: creation of user information, verification of user logins, checking whether a user existed, and fetching user details by email. In the user interface and experience, I did a UI layout for every activity, ensuring that the screen elements are well-navigated and look nice. Also, I have edited and implemented various basic UI and UX tasks, such as adjusting layouts for different screen activities to ensure a smooth user experience.

Some of the key features I have implemented are user authentication using shared preferences for session persistence, where users will remain logged in between sessions; working on the processes of user registration and validation with password handling; and populating the initial database by using helper methods to manage the data of the users efficiently.

Member 2: Le Thanh Phuong Nam:

I managed the core navigation of the application and implemented some key screens. Among my first screens that I worked upon was the home interface; it provided clear access to users easily into other main features of the app. I created a screen on film details with options of booking tickets. Apart from this, I designed a seat selection and scheduling feature that enabled users to choose their seats and view the available booking time.

As for structuring the code, I have tried to put more emphasis on the organization of project folders and the modularization of components into activities, view models, adapters, and models, among others, to ensure the project was well-structured and scalable. In integrating the database, I further developed the functionalities of the database by creating models for films, seats, and other related entities to ensure smooth data management and retrieval.

Some of the key features introduced were RecyclerViews in home and information on film screens to efficiently work with datasets of dynamic or huge volume, a screen with the development for a seat listing interactive feature for immediate updating according to what each user chose for their seats, enhancements in end-user navigation to activity transition smoothness as seamless to users.

Member 3: Phan Cong Bao:

I designed and implemented the transaction-related screens, which generate and display booking confirmations and invoice details. I managed user-selected tickets, handling operations of carts. I developed features for profile and password management of users, including a list screen for profile management. I implemented functionality to let users change passwords and log out from their profiles.

I've done database integration through the extension of the database in supporting invoices and transaction data. Managed updates of data required for booked seats and invoice generation. Major features I've built include the invoice generation system, its integration to the database in the invoice screen. Then, I also built an editable cart with invoice management functionality in the paid-ticket screen.

Project Implementation Timeframe:

Implementation	Assigned Member	Functionality	Date
RegisterActivity.kt	Dao Ngoc Linh	Handles user registration with validation logic	21 Oct 2024
LoginActivity.kt	Dao Ngoc Linh	Manages user login and session handling	25 Oct 2024
DatabaseHelper.kt	Dao Ngoc Linh	Manages database creation and data operations	1 Nov 2024
IntroActivity.kt	Dao Ngoc Linh	App introduction and navigation to login/register	7 Nov 2024
MainActivity.kt	Le Thanh Phuong Nam	The home screen has navigation to film details, profile, and cart	22 Oct 2024
FilmDetailActivity.kt	Le Thanh Phuong Nam	Displays film details and options for booking tickets	3 Nov 2024
SeatListActivity.kt	Le Thanh Phuong Nam	Allows seat selection with pricing calculation	12 Nov 2024
InvoiceActivity.kt	Phan Cong Bao	Generates and displays invoice details post-booking	22 Oct 2024
ProfileActivity.kt	Phan Cong Bao	Allows profile management and password updates	1 Nov 2024
androidTest	Phan Cong Bao	Create test files for functions in some primary activities	7 Nov 2024

test [UnitTest]	Phan Cong Bao	Create test files for primary view models and database	15 Nov 2024

Environment Configuration:

SDK and Tools: The project is using Android Studio Flamingo version 2022.2.1 Patch 2, and Kotlin version 1.8.0 was used for coding. Its Gradle version is 8.7.3, which goes well with the project dependencies. The target SDK of the project is Android 14, API Level 34, while the minimum supported SDK is Android 8.0 Oreo, API Level 26.

Libraries and Dependencies: Glide handles and displays images efficiently for seamless user experiences, while ChipNavigationBar makes for easy bottom navigation across different activities. Room Database takes care of persistence, acting like a local database that stores user information. ViewModel and LiveData handle UI-related data in a lifecycle-aware manner, while custom RecyclerView Adapters are developed to handle lists and grids.

Emulation/Testing Devices: The project has been executed on both virtual and physical devices to test and confirm its functionality. The virtual device emulated is that of a Pixel 8, which, by resolution, goes for 1080x2400 pixels and runs on Android 14, API Level 34. It also went through the physical test on the Samsung Galaxy S22 running on Android 13, API Level 33, to confirm its operability on different Android versions.

Development Features: This project includes database management systems that are strong enough to handle any user authentication or invoice processing-seat booking processes with ease. Architecture MVVM (Model-View-ViewModel) guarantees modularity and maintainability of code in this application, while smooth animation provides fluid screen transitions with intuitive navigation or consistent styling across themes for coherency.

Implementation

The project follows the Model-View-ViewModel (MVVM) architecture to separate the business logic and data management from the user interface (UI). Here's how the architecture is implemented:

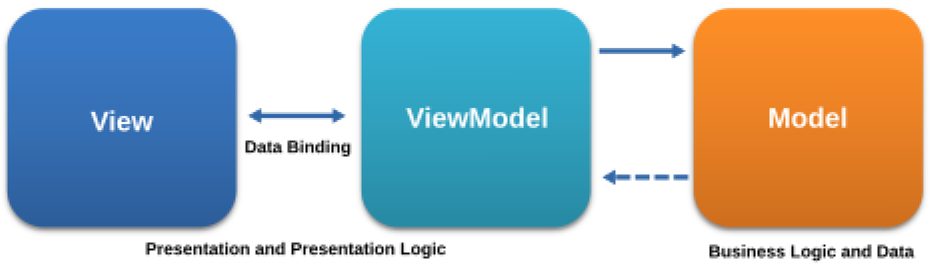


Figure 1. The MVVM architecture

Models:

Models define the structure of data used and are responsible for the representation of data entities used in the application.

Film.kt defines the structure that is to store information about the film, such as the title, description, IMDB rating, and other important information. User.kt is the class defining the format of the user data, including such important attributes as email, username, and password. Seat.kt describes the format of data in the case of seat selection-whether the seat is free, the name of it, and to what film it belongs. Invoice.kt specifies the layout adopted in storing information concerning an invoice that entails the movie's name, the time of show, choice of seats, theatre and pricing.

ViewModels:

View Models handle lifecycle aware UI related data and UI should able to respond to any event/change on those without leaking a data in config changes and also at application's lifetime events.

MainViewModel.kt will be responsible for such implementations as loading films, showing lists of upcoming movies, and managing user sessions. FilmDetailViewModel.kt is responsible for the details of a certain film, updating information about the selected film, and setting it. LoginViewModel.kt cares about the logic of user authorization, which means the login process is smooth and secure. Finally, RegisterViewModel.kt checks the user's registration information and controls the database interaction in order to create new user accounts.

Repositories:

The Repositories are an abstraction of sources of data. They are actually responsible for managing the operations upon the data themselves.

MainRepository.kt is responsible for handling data of films, users, and upcoming movies, providing a unified place of access and management. LoginRepository.kt should check the correctness of login and password according to the database data. RegisterRepository.kt is responsible for creating a user, checking if one already exists, and allowing the creation of a new account.

ViewModel Factories:

View model factories are supposed to provide dependency injection so that ViewModels are created with its dependencies.

MainViewModelFactory.kt is used for creating instances of the MainViewModel with necessary context, allowing the ViewModel to be correctly initialized. Similarly, LoginViewModelFactory.kt provides instances of LoginViewModel, with the needed context to function as expected. In turn, RegisterViewModelFactory.kt does so for the RegisterViewModel, with the required context to handle user registration. Finally, FilmDetailViewModelFactory.kt creates instances of FilmDetailViewModel, providing the needed context to handle film-specific operations.

Main Application Flow:

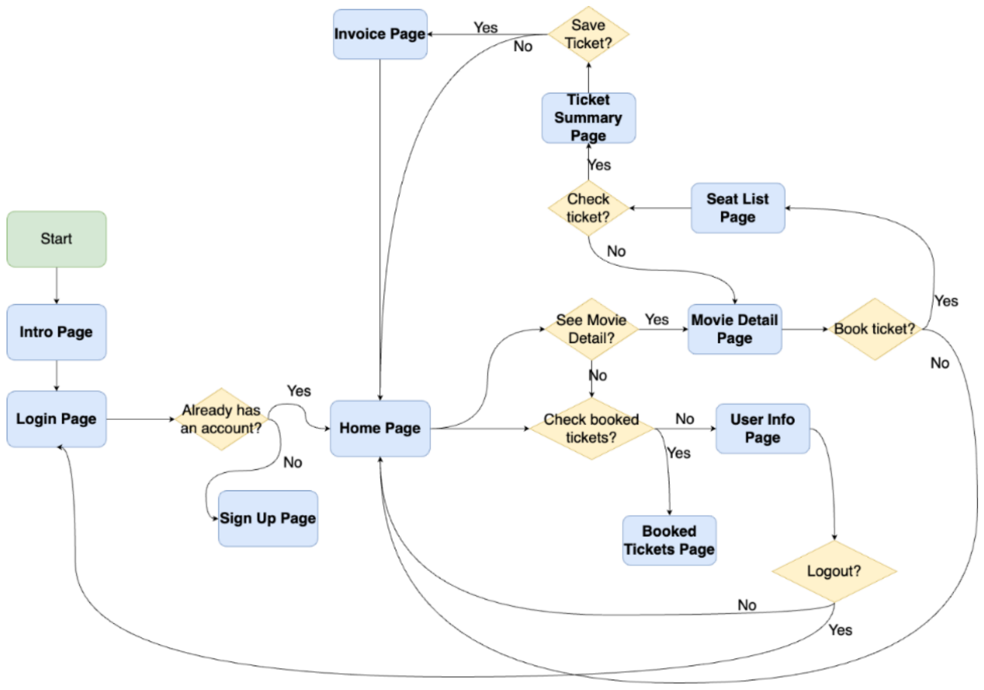


Figure 2. User Flow Diagram

The IntroActivity is the primary flow of the application. It's an introductory screen to an app, basically introducing the app and containing a "Get Started" button. On clicking that, it will take the user to the LoginActivity. In LoginActivity, users can log in into their existing accounts, and on successful login, are taken to the MainActivity. If the user wants to make a new account, he/she can click "Register Now" to move to RegisterActivity. In RegisterActivity users can create new account. This will then bring the user back to LoginActivity upon a successful registration or automatically signs up and take the user into MainActivity.

After logging in, the user is taken to the Main Screen, MainActivity, where the user can view movie lists, featured movies, and upcoming releases. The user is taken to FilmDetailActivity after clicking on any movie, where detailed information about a particular movie is shown, and the "Buy Ticket" button is clicked to proceed to SeatListActivity. The user can also use the navigation bar in MainActivity to access CartActivity for viewing their cart or ProfileActivity for account management.

In FilmDetailActivity, users are provided with in-depth details about the movie and a button to buy tickets. When the "Buy Ticket" button is clicked, users are taken to SeatListActivity, where they can choose show dates, times, and seats. After selecting their preferred options, they can click the "Download Ticket" button to move on to TicketSummaryActivity.

The TicketSummaryActivity shows the details of the selected ticket, and one can confirm the ticket to go ahead to InvoiceActivity or cancel the process and go back to MainActivity. In InvoiceActivity, the user views the final invoice, and the ticket gets saved in the database. Confirming the invoice takes one to the MainActivity.

CartActivity shows the booked tickets. The user can either go back to the MainActivity via the bottom navigation bar or navigate to ProfileActivity. ProfileActivity finally lets the user manage account details, change password, or log out. The log-out functionality brings him/her back to the IntroActivity for a fresh restart.

User-Friendly and Consistent UI/UX Design:

The application offers a user-friendly and consistent UI/UX design, intuitive to use. There are transition effects while going between screens-like slide_in_right and slide_out_left-that will make the usage of this app seamless. A ChipNavigationBar is implemented in MainActivity, CartActivity, and ProfileActivity. Thus, access to main features is pretty simple and will provide easy navigation.

The interface is consistent, using a well-combined colour palette, icons, and typography throughout the screens. XML layouts, such as activity_main.xml, activity_film_detail.xml, and activity_seat_list.xml, have been designed in such a way that visuals across different sections of the app are maintained.

These steps are reasonably aligned in such a sequence that would follow a practical logical flow-one following the other; for instance, movie selection for booking, paying. It will then give ample notification and instruct at every touch point, giving proper insight on every step-one would have gone ahead with login errors and confirmation against the seats selected, after having logged in.

User Authentication:

User authentication is the process through which users gain access to the application by log-in using account credentials. For a user to authenticate, they have to input their username and password. The application then checks the entered details against information stored in the database for authenticity.

This whole process starts with an activity class called the LoginActivity, from an implementation perspective, where a user gives in his or her credentials. The provided details are sent to the SQLite for authentication. In case the provided login details match correctly, he shall be authenticated and the application switches to the main interface, which is the MainActivity. Otherwise, if wrong details are given, an error message will be thrown for display to the user.

Once the verification process is completed, the app transitions to the home screen, which is the MainActivity. If authentication fails, the user remains in the LoginActivity. An error is displayed asking for a correction of their login details.

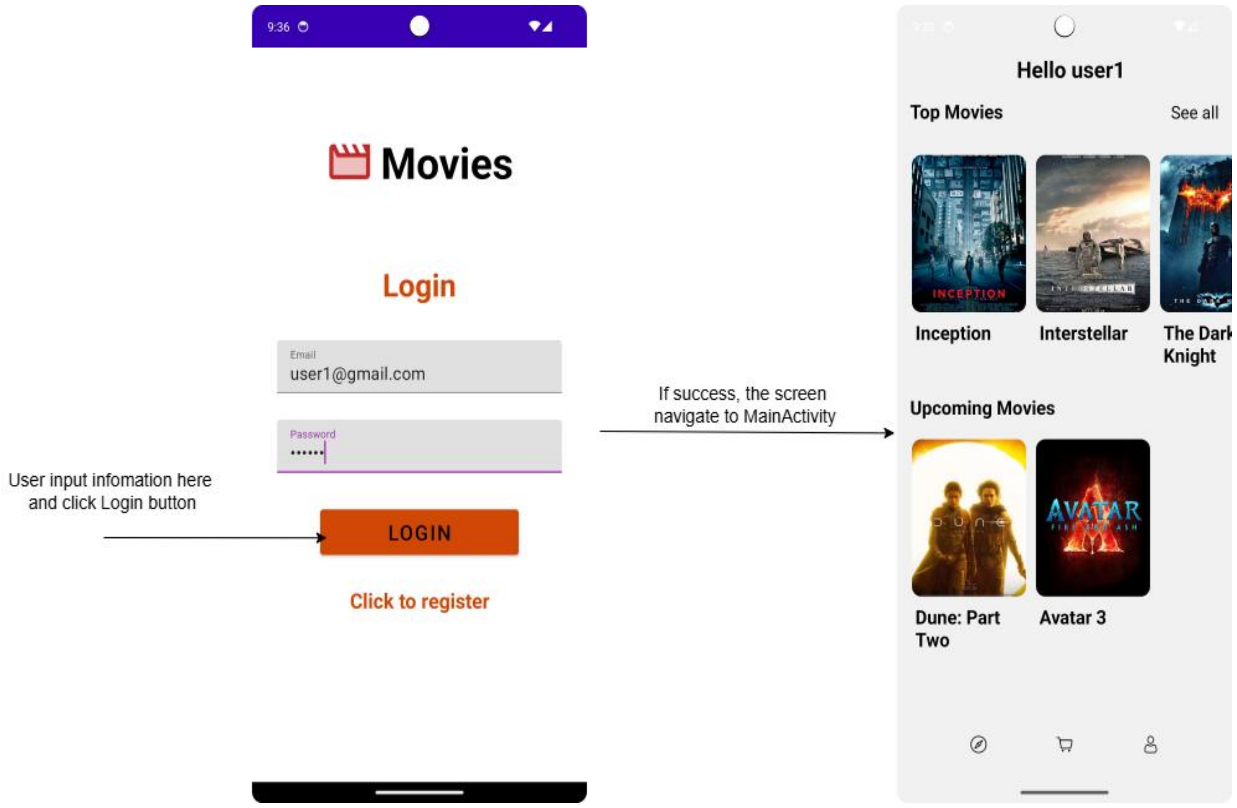


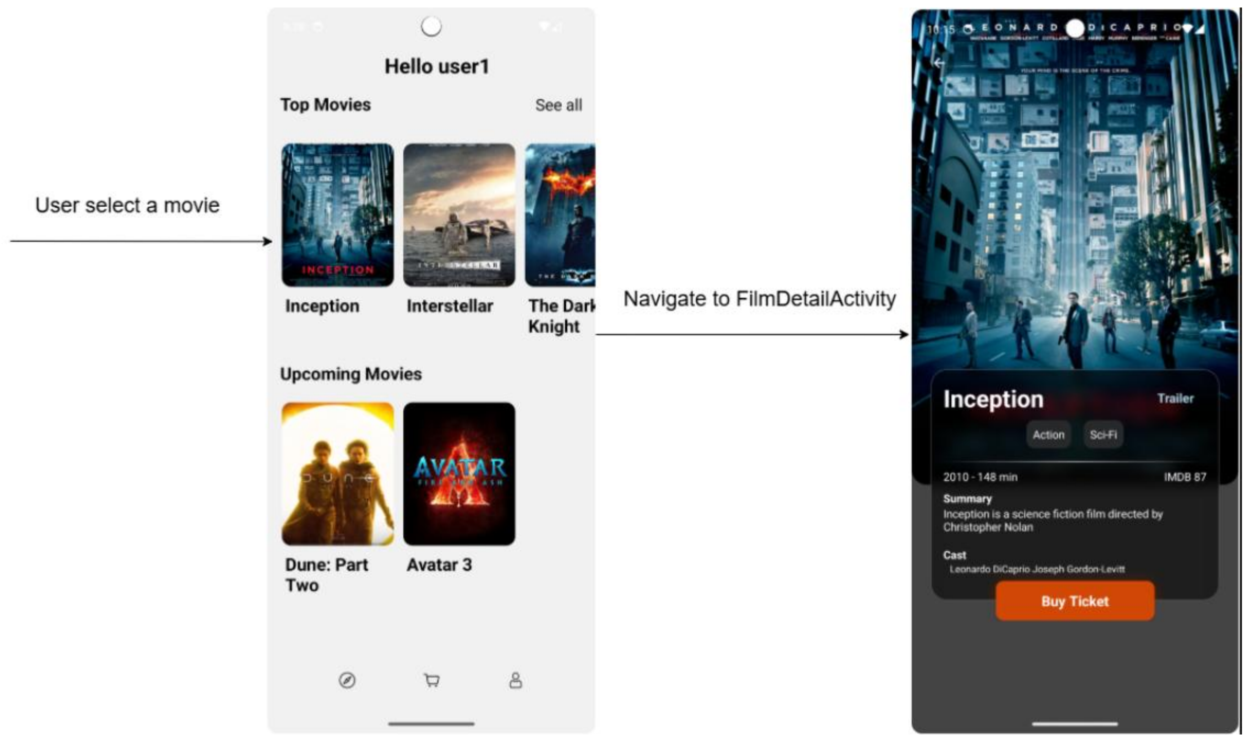
Figure 3. User Authentication Process

Main Interface:

The main interface is the first screen that will appear before them immediately after their successful login, from where they can browse the listed movies, read their detailed information, and initiate a movie ticket booking process.

Its implementation includes the presentation of the list of available movies in MainActivity, data presentation based on RecyclerView, together with an adapter, to handle data and view presentation of movie objects. Furthermore, when any movie is clicked in the movie listing screen, the application will directly enter into the FilmDetailActivity in order to give detailed information about the particular selected movie.

After a movie is selected, the app navigates to the FilmDetailActivity to show the detailed view of the selected movie.



Inception is a science fiction film directed by Christopher Nolan

Leonardo DiCaprio Joseph Gordon-Levitt

Figure 4. Movie Selection Process

Movie Details:

The movie details screen will provide the user with all the information about the movie they selected on the main interface: the name of the movie, a short description, show times, and other important details.

It would be implemented that FilmDetailActivity is to display all detailed information concerning the selected movie. Information is to be presented by using a combination of TextView elements for text (description, showtimes) and ImageView for visual elements (cover image of the movie).

After the film details have been shown, the user will be brought to either return to the MainActivity or to SeatListActivity in order to proceed with their booking of seats.

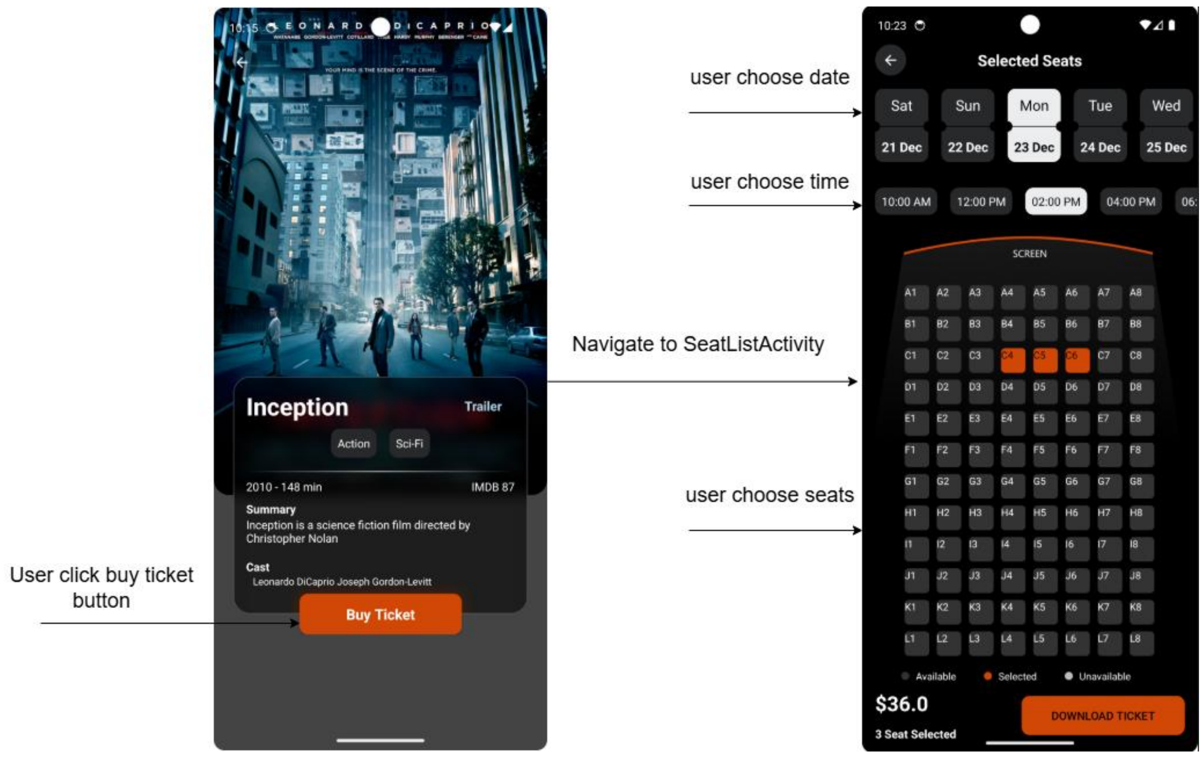


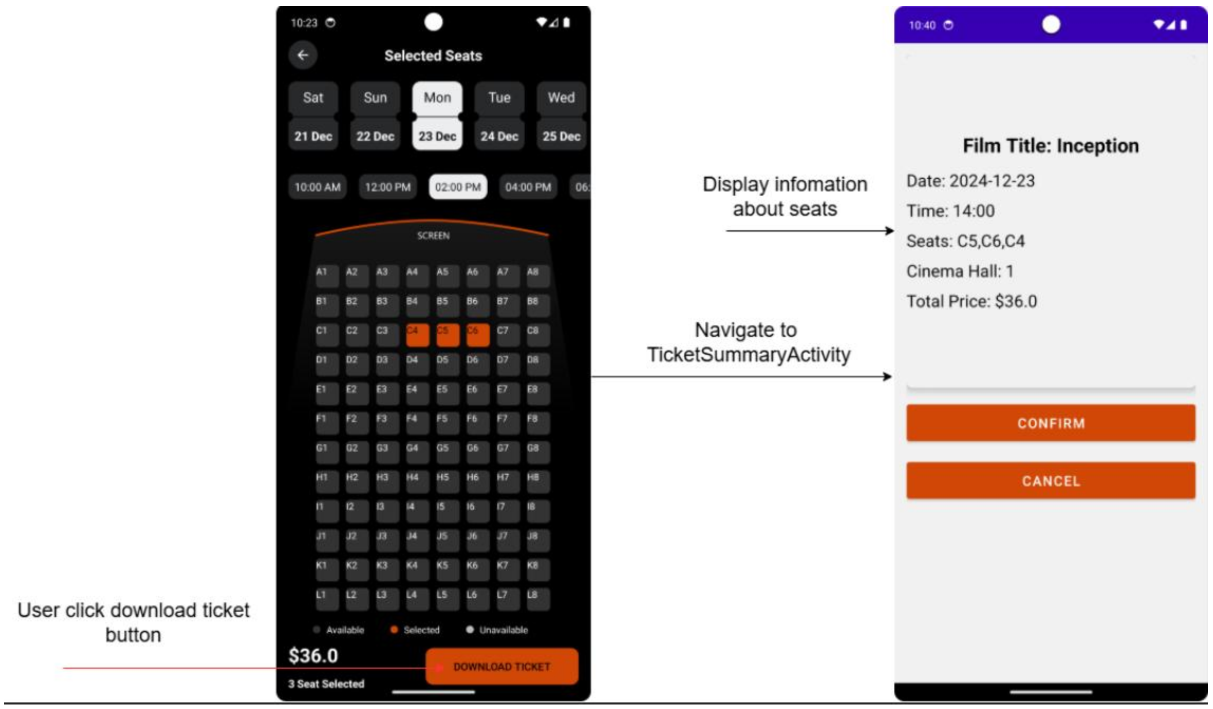
Figure 5. Ticket Buying Process

Seat Booking:

The ticket booking function enables users to choose their seats in the cinema and pay for the tickets for the movie they wish to watch.

In terms of implementation, the SeatListActivity allows the user to select a seat from an interactive seat map. Once a seat is chosen, the details of the selected seat are stored, and the user is moved to the next step in the booking process—the confirmation screen.

After selecting a seat, the user is directed to the TicketSummaryActivity, where they can review and confirm their ticket information before completing the booking.



Film Title: Inception
Date: 2024-12-23
Time: 14:00
Seats: C5,C6,C4
Cinema Hall: 1
Total Price: $36.0

Figure 6. Seat Booking Process

Adapters and RecyclerViews:

FilmListAdapter.kt shows the list of films in the main screen (MainActivity). It uses Glide for efficient image loading by applying transformations to improve appearance, such as rounded corners. The adapter dynamically binds film data for a seamless navigation experience upon opening FilmDetailActivity when a user selects any film. It also uses the submitList method to update the dataset efficiently and refresh the UI.

The SeatListAdapter.kt is designed to manage the grid layout for seat selection in the SeatListActivity. This adapter handles different seat availability statuses—AVAILABLE, SELECTED, and UNAVAILABLE—and updates the grid dynamically based on user interactions. It keeps track of the selected seats and calculates the total price in real time. Furthermore, the adapter provides a callback interface, SelectedSeat, which passes the seat selection data back to the parent activity for smooth communication.

The DateAdapter.kt renders a horizontal list of available dates for the show and marks the selected date. It notifies a UI update, highlighting the selected date differently from others. The adapter sends back the selected date to the parent activity so it can load appropriate time slots to be presented for the user's choice.

TimeAdapter.kt is responsible for displaying available time slots in a horizontal list, which is dynamically filtered with respect to the selected date and current time. Notify parent activity of the adapter upon selecting time; parent activity updates the seat availability so that user can choose a time that works for them.

SQLite Database Integration:

SQLite acts as the backbone of the app, persisting data between runs. This gives way to a more complicated system that can allow for seat booking and invoice generation, among other features. At the core of this integration is the DatabaseHelper.kt file, which will manage the storage and retrieval of data within the application. It creates, reads, updates, and deletes various types of entities such as users, films, seats, and invoices.

The DatabaseHelper.kt file communicates with nearly all the tables in the schema, namely, films, seats, invoices, and users; hence, the queries have been optimized. Some of the important methods implemented in this file are the getBookedSeats(filmTitle, showDate, showTime) method, which fetches the booked seats for a particular show to keep real-time availability; insertInvoice(invoice: Invoice) is used to save booking details into the database for future reference. Also, some of the other important methods, like getAllFilms() and getUpcomingMovies(), which retrieve movie data, are used to populate the main screen of the app.

Our Final Thoughts

Challenges:

During development, the movie ticket booking mobile application has been worked out with a series of challenges, such as problems of time limitation, resources management, and even external commitments. Among these obstacles, the foremost one was its end duration time, which fell at approximately three months, hence placing many pressures on being able to timely release the deadline at hand. In addition, this project is taking less and less time due to scheduling conflicts with other academic subjects, often becoming the cause for lagging behind.

Moreover, team members combine their work in the development of the app with part-time jobs. This further complicates the time factor, and in some cases, the jobs have taken priority over the work on the app development, which delayed the beginning of certain processes within the app development. Another challenge has been that at times, the unavailability of the project advisor because of his natural work obliged the team to independently handle various tasks without immediate guidance. Besides, the project faced financial constraints since the development of the app was without funding. Not getting the required economic support meant designing, developing, and maintenance of the application were all volunteer-based. The effect was that there could be minimal tools and extension of resources done to the app to add its functionalities.

Future Work:

For further development, the movie ticket booking application would be developed and enhanced with the following features. First and foremost, the app will be developed by using the help of Firebase in the management of the database and the authentication of users. This would provide a secure, scalable backend for storing user data related to booking history and personal preferences, along with a reliable authentication system to manage accounts.

There is also an indication in the direction of enhancements for improving the user experience, for example, adding features that make users more friendly such as reward points. The users would gain points from booking tickets, which could then be used for some discounts or special offers. This will help not only in increasing user activity but also retaining customers over a longer period. Another important future step would be integrating the methods of payment into the app. By adding secure payment gateways, users will be able to buy tickets directly within the app, which would facilitate the booking process and provide users with more seamless transaction experiences.

Future development should pursue another important feature: integration with the website of the mobile application. It should be easy to switch between platforms with access to user accounts, booking history, and preferences. The synchronization of data on both the app and website should ensure continuity and therefore make using the app more convenient.

List of figures