POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Watchee: Movie Tracking and Social Platform

Design Document
Design and Implementation of Mobile Applications

Matteo Laini - 10683821
Matteo Macaluso - 10656537

Academic Year: 2024-25

Contents

1 Introduction **3**
- 1.1 Purpose . . . 3
- 1.2 Scope . . . 3
- 1.3 Definitions, Acronyms and Abbreviations . . . 4
  - 1.3.1 Definitions . . . 4
  - 1.3.2 Acronyms . . . 4

2 Requirements and Features **5**
- 2.1 Android features implemented . . . 6
  - 2.1.1 User Authentication . . . 6
  - 2.1.2 Movie Discovery and Search . . . 6
  - 2.1.3 Movie Information Display . . . 6
  - 2.1.4 Cast Member Details . . . 6
  - 2.1.5 Watchlists Management . . . 7
  - 2.1.6 Watchlist Features . . . 7
  - 2.1.7 Review System . . . 7
  - 2.1.8 Profile Customization . . . 7
  - 2.1.9 Social Interaction . . . 7
  - 2.1.10 Notification Management . . . 8
  - 2.1.11 Tablet Interface Optimization . . . 8
- 2.2 Non-Functional Requirements . . . 9
  - 2.2.1 Performance Requirements . . . 9
  - 2.2.2 Resilience Requirements . . . 9

3 Application architecture **10**
- 3.1 Overview . . . 10
- 3.2 Services . . . 11
- 3.3 Database overview . . . 12
- 3.4 Sequence diagrams . . . 13
  - 3.4.1 App Launch & Authentication Flow . . . 14
  - 3.4.2 Login process . . . 15
  - 3.4.3 Movie Search & Movie Details Retrieval . . . 16
  - 3.4.4 Watchlist Management . . . 17
  - 3.4.5 Notification System . . . 18

4 Application Design **19**
4.1 Authentication 20
4.2 Home Page 21
4.3 Movie Details 23
4.4 Cast Details 24
4.5 Profile 25
4.5.1 User profile 25
4.5.2 Edit personal information 26
4.5.3 Settings 27
4.5.4 Notifications 28
4.5.5 Notification page 29

5 Testing **30**
5.1 Unit Testing 30
5.2 Widget Testing 30
5.3 Coverage 31
5.4 Integration Testing 33

6 Conclusion **34**

1 | Introduction

1.1. Purpose

In today's digital age, the movie industry continues to grow with numerous streaming platforms offering vast libraries of content. With this abundance of options, users often find it challenging to keep track of movies they've watched, want to watch, or discover new ones that align with their interests. Watchee was developed to address these challenges by providing a comprehensive movie tracking and social platform.

The purpose of Watchee is to enhance users' movie-watching experience by enabling them to:

- Track movies they've watched or want to watch
- Share their opinions through ratings and reviews
- Follow other users and discover new movies through social interactions
- Create, manage and share custom watchlists
- Receive personalized movie recommendations
- Check available streaming providers for each movie by country

This document provides a detailed overview of the design decisions and software architecture that form the foundation of Watchee.

1.2. Scope

This document covers the requirements and specifications of the Watchee mobile app, along with its interaction with external services supporting its functionalities. The primary stakeholders are:

- End Users: Who interact with the service to track their movie-watching habits and engage with the community
- Content Providers: Who supply movie information through The Movie Database (TMDB) API
- System Administrators: Who maintain and monitor the application's infrastructure

1.3. Definitions, Acronyms and Abbreviations

1.3.1. Definitions

Flutter A UI toolkit developed by Google for building natively compiled applications across multiple platforms from a single codebase.

Firebase A platform developed by Google for creating mobile and web applications, offering various backend services.

TMDB The Movie Database, a community-built movie and TV database.

Watchlist A curated list of movies that users want to watch or share with others.

1.3.2. Acronyms

API Application Programming Interface

UI User Interface

HTTP Hypertext Transfer Protocol

BLoC Business Logic Component

DB Database

2 | Requirements and Features

In this chapter we illustrate the key requirements that were considered when developing Watchee, and how they reflected onto the features being implemented in the platform. In the following table we listed all the pillar requirements that guided us through our Watchee development journey.

ID	Name	Description
FRE1	User Authentication	The user shall be able to register and login using an email address or Google Account authentication.
FRE2	Search Functionality	The user shall be able to search for movies and cast members in real-time through the search bar on the home page.
FRE3	Movie Information	The user shall be able to access comprehensive movie details including title, runtime, plot, trailer and rating.
FRE4	Cast Information	The user shall be able to view complete cast member information including personal details, description and filmography.
FRE5	Movie List Management	The user shall be able to add and remove movies to predefined lists or personalized collections.
FRE6	Watchlist Features	The user shall be able to create and manage watchlists with functionality for adding/removing movies, ordering options, sharing and collaborator management.
FRE7	Review System	The user shall be able to post movie reviews and access reviews from followed users.
FRE8	Profile Customization	The user shall be able to personalize their profile by selecting favorite genres and setting a custom profile picture.
FRE9	Social Features	The user shall be able to search for other users, view their profiles and follow them.
FRE10	Notification System	The user shall receive notifications for new followers, watchlist collaboration invites, and upcoming releases of liked movies.
FRE11	Tablet Support	The user shall be able to access an optimized horizontal layout when using the application on tablet devices.

Table 2.1: Table of Requirements

2.1. Android features implemented

Given the above requirements, we then mapped them to higher-level features, representing what the final product had to include from both a functional and interaction perspective. When mapping implemented features to functional requirements, we strived to follow a classic chronological order based on a typical usage session of the app.

2.1.1. User Authentication

Users can sign in to Watchee by using their email address and password.
In case they do not have an account on the platform, they can choose to sign up providing an email address and password, or alternatively use their Google Account for a quicker registration.
During the initial registration phase, users are required to complete their profile by selecting a username, providing their full name, and choosing at least three favorite movie genres to personalize their experience.
This feature maps requirements FRE1 and FRE8.

2.1.2. Movie Discovery and Search

Users can explore content through a universal search bar located at the top of the home screen, which allows them to look for both movies and cast members in real-time. The home page displays trending movies and organizes content into different categories such as "Now Playing", "Top Rated", and "Upcoming". Additionally, a personalized "Recommended for You" section shows movies based on the user's selected favorite genres. Users can tap on any movie to access detailed information. This feature maps requirements FRE2 and FRE3.

2.1.3. Movie Information Display

Users can access comprehensive details about any movie by selecting it from the home page or search results. Each movie page displays essential information including the title, release date, runtime, plot overview, and average rating. A YouTube trailer can be watched directly within the app, and users can check streaming availability across different countries through a provider section. Below the main information, users find a scrollable cast list displaying actor photos and character names. This feature implements requirements FRE3 and FRE4.

2.1.4. Cast Member Details

When users tap on any cast member, they are taken to a detailed profile page showing the person's biographical information and complete filmography. The filmography section displays all movies the person has appeared in, sorted by release date, with their respective roles. Users can tap on any listed movie to view its details, creating an interconnected browsing experience. This feature satisfies requirement FRE4.

2.1.5. Watchlists Management

The app provides users with two default lists: "Liked Movies", "Seen Movies" and the possibility to create customized watchlists through the floating button in the lists page. Users can easily add or remove movies from any of these lists through a modal interface accessible from the Watchist page. The liked and seen lists are private to each user, while watchlists can be configured with different privacy settings. The app maintains these lists across sessions and synchronizes them with the cloud storage. This feature maps requirements FRE5 and FRE6.

2.1.6. Watchlist Features

Users can create multiple custom watchlists with specific names and privacy settings. Public watchlists are visible to anyone, while private ones remain accessible only to their creator. Watchlists also allow movie sorting through a simple sorting tile. The collaborative feature allows users to invite others to contribute to a watchlist, where all collaborators can add or remove movies. Users can also follow other users' public watchlists and share every public watchlist through an external link to anyone. This feature implements requirement FRE6.

2.1.7. Review System

Users can express their opinions about movies through a rating and review system. From any movie details page, users can assign a rating from 1 to 5 stars and write a text review of up to 160 characters. On each movie page, reviews from followed users are displayed prominently, allowing users to see what their friends think about the movie. Users can manage their own reviews through a dedicated interface in their profile, where they can edit or delete previous reviews. This feature maps requirement FRE7.

2.1.8. Profile Customization

Upon completing registration, users must set up their profile by choosing a username, providing their full name, and selecting at least three favorite movie genres. Users can optionally upload a profile picture from their device's gallery. These genre preferences are used to power the recommendation system on the home page. Users can modify their profile information and genre preferences at any time through the account management page. This feature implements requirement FRE8.

2.1.9. Social Interaction

Users can discover other movie enthusiasts by viewing their public profiles, which display their watchlists, reviews, and movie preferences. The search bar on the top of the people page allows users to find their friends by name or username. The follow system enables users to stay connected with others whose movie tastes they find interesting. When viewing another user's profile, they can see the public watchlists they maintain and any reviews they've written. A mutual follower indicator helps users identify shared

connections. Users can manage easily following and follow users in the people page. This feature satisfies requirements FRE8 and FRE9.

2.1.10. Notification Management

The app keeps users informed through a comprehensive notification system. Users receive alerts when someone follows them and when movies in their "liked movies" list are about to be released. For collaborative watchlists, notifications are sent when receiving invitations to collaborate. Users can manage their notification preferences through the settings page. This feature implements requirement FRE10.

2.1.11. Tablet Interface Optimization

When using the app on a tablet device horizontally, the interface automatically adapts to provide an optimized landscape layout. The home screen reorganizes the main widgets into a two-column view showing more content simultaneously. Movie details pages display information in a side-by-side layout, with the movie poster and details on one side and reviews or cast information on the other. The cast details page also displays information in a two column layout, allowing the user to scroll through their filmography while being able to read their biography. This tablet-specific interface takes full advantage of the larger screen space while maintaining all functionality of the phone version. This feature satisfies requirement FRE11.

2.2. Non-Functional Requirements

2.2.1. Performance Requirements

Performance requirements are critical for ensuring a smooth and responsive user experience across different devices and network conditions. The entire application is optimized to lower loading times through navigation. Watchlists structure has been modeled in order to avoid continuous and repetitive API calls without losing any real time information. However the response times related to TMDB API calls cannot be managed and affect mostly the home page loading time.

2.2.2. Resilience Requirements

The application shall behave correctly when internet connection interrupts:

1. No internet starting screen - Application launch on a specific page when unable to connect to internet - Button to reload the app status

2. In-App connection failure handling - The app displays a message when internet connection is not available - Retry button reloads the entire application - Image loading with placeholders

3 | Application architecture

3.1. Overview

Watchee was developed using Flutter, a cross-platform framework that enables the creation of natively compiled applications from a single codebase. The application was designed to be highly responsive across different Android devices, from phones to tablets, with layouts and components that automatically adapt to various screen sizes.

The architecture is structured into two layers distributed over two tiers. The application operates as a thick client, handling both the presentation and application logic directly on the device. A separate data layer is implemented server-side using Firebase services to provide persistence, authentication, and real-time functionality across user sessions and devices. The app also integrates with an external API (TMDB) to retrieve up-to-date movie information and streaming availability data.

The application uses the BloC pattern for state management and dependency injection, making it easy to share and update data across different parts of the app. This pattern enables efficient state management and helps maintain a clean separation between the UI components and the business logic.

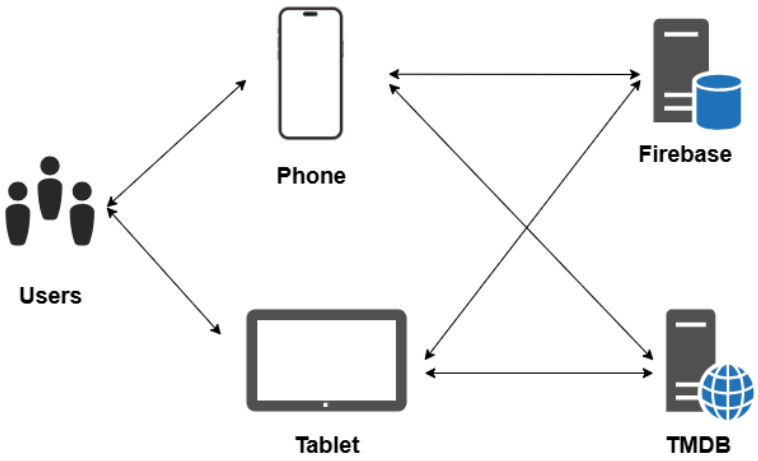


Figure 3.1: Watchee's high-level architecture overview

3.2. Services

The application is powered by both external and internal services that work together to deliver its core functionalities:

External Services:

- TMDB API: Serves as the primary source of movie-related data, providing comprehensive information about movies including details, ratings, cast information, and streaming availability.
- Firebase Authentication: Manages user authentication, supporting both email/password and Google Sign-In methods for secure user access.
- Cloud Firestore: Acts as the main data store, handling real-time synchronization of user data, watchlists, reviews, and social connections.
- Firebase Cloud Messaging (FCM): Powers the notification system, delivering real-time alerts for social interactions (new followers), watchlist invitations, and movie updates.

Internal Services:

- UserService: Manages user-related operations including profile updates, follower management, and user data synchronization with Firestore.
- WatchlistService: Handles the creation, updating, and sharing of watchlists, including collaborative features that allow multiple users to contribute to a single watchlist.
- Custom Authentication Services: Wraps Firebase Authentication functionality to provide a clean interface for handling user authentication flows within the app.
- FCM Service: Manages token refresh, notification permissions, and handles incoming notifications to display appropriate in-app updates.

The application employs a service-based architecture where each service is responsible for a specific domain of functionality. These services are injected into the application using Provider, allowing for easy access throughout the widget tree while maintaining a clear separation of concerns.

This architecture ensures that the application remains modular, maintainable, and scalable while providing a smooth user experience with real-time updates and offline capabilities. The thick client approach allows for responsive user interactions, while the Firebase backend ensures data consistency and enables social features across devices.

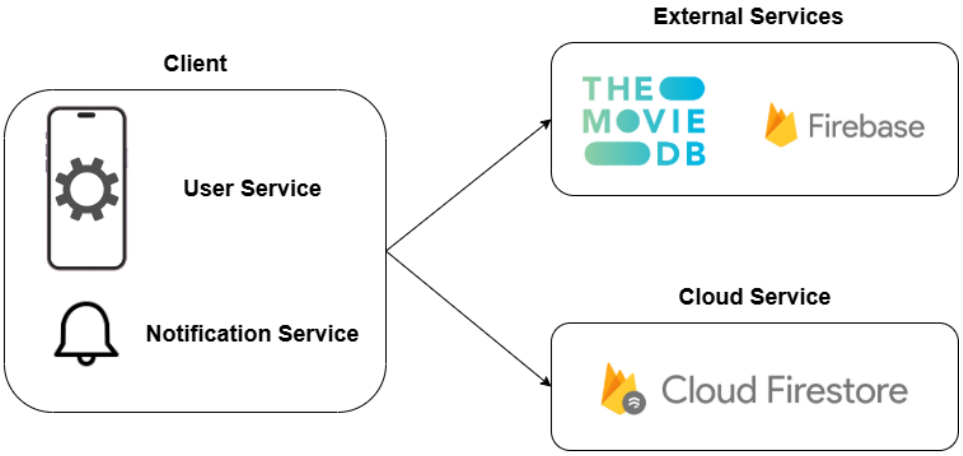


Figure 3.2: Watchee's services overview

3.3. Database overview

The following diagram shows the structure of Watchee's database.

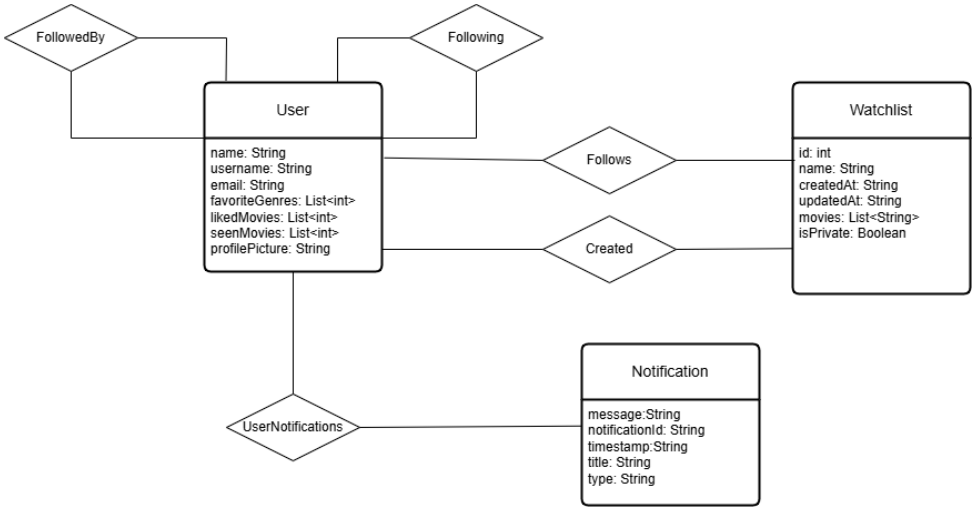


Figure 3.3: Watchee's database overview

These are the entities that we chose to store in Firestore:

- User: It contains every relevant information about a user (name, username, email) and also lists containing followed/following users, created/followed watchlists, and the 10 most recent received notifications.
- Watchlist: it contains details about a watchlist (movie list, timestamps of creation and latest update, whether it is private or public)
- Notification: it contains details about a notification.

3.4. Sequence diagrams

Sequence diagrams visualize the interactions between different components and services within Watchee. They help to understand the data flow and communication patterns across the application's architecture. These diagrams map out the chronological order of operations and show how different parts of the system work together to accomplish specific tasks. To keep the diagrams clear and focused, the function calls are represented at a high level, omitting internal implementation details. Service calls to external providers such as Firebase and TMDB are shown as direct interactions, although they actually pass through service layer abstractions in the actual implementation.

3.4.1. App Launch & Authentication Flow

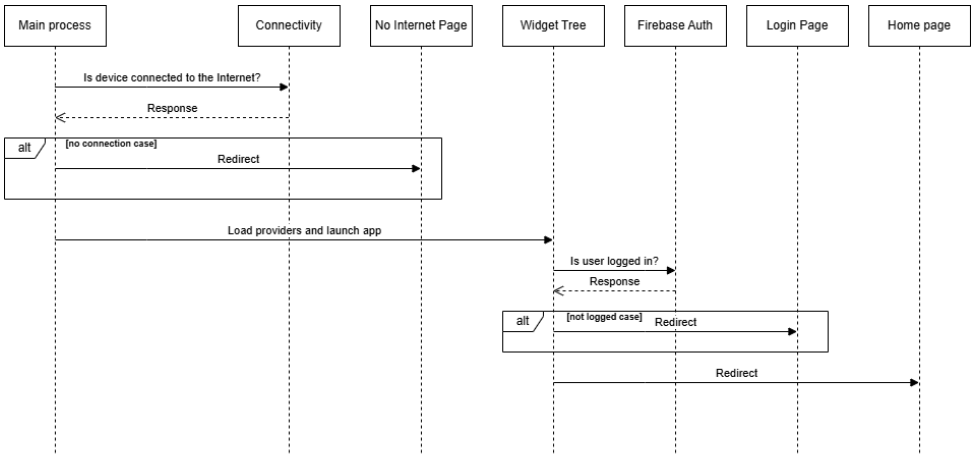


Figure 3.4: Sequence diagram about app launch

This diagram illustrates the initial flow when a user launches Watchee. The main process first checks authentication status with Firebase Auth Service. Based on this status:

- If not authenticated, the user is presented with the login page
- If authenticated, the user's data is retrieved from Firestore and the app proceeds to HomePage
- The diagram also shows the case where the app is launched but the device is not connected to the internet. In this case the app shows an error message and an option to reload the application

3.4.2. Login process

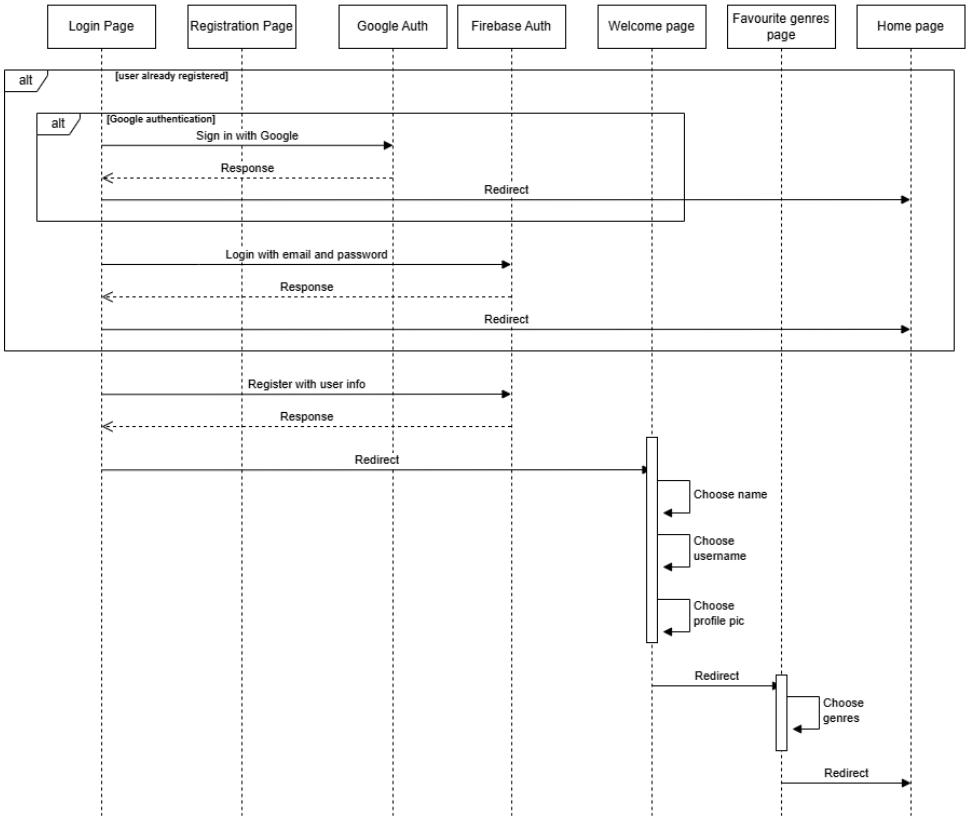


Figure 3.5: Sequence diagram about login process

The login sequence showcases the authentication flow when a user attempts to sign in:

- User provides credentials through the login form.
- Credentials are validated and sent to Firebase Auth.
- Upon successful authentication, user data is fetched from Firestore.
- FCM token is generated and stored for notifications.
- The user is then redirected to the main application flow.
- If the user is not registered, the app will show an option to register.

3.4.3. Movie Search & Movie Details Retrieval

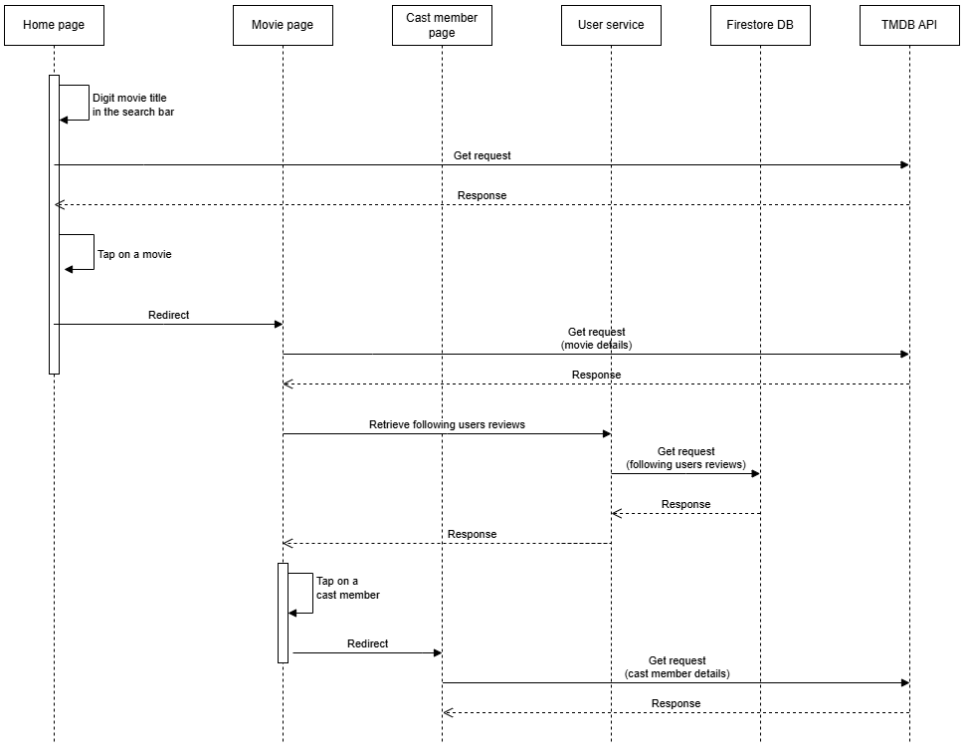


Figure 3.6: Sequence diagram about movie search and movie details retrieval

This diagram demonstrates the interaction flow when users search for movies and view details (the same happens when a user taps on a movie from the home page or from a watchlist):

- Search queries are sent to TMDB API.
- Results are displayed in the search interface.
- When a movie is selected, detailed information is fetched.
- Additional data like streaming providers and recommended movies are retrieved.
- Reviews from followed users are fetched from Firestore.
- All this information is compiled and presented in the film details view.

3.4.4. Watchlist Management

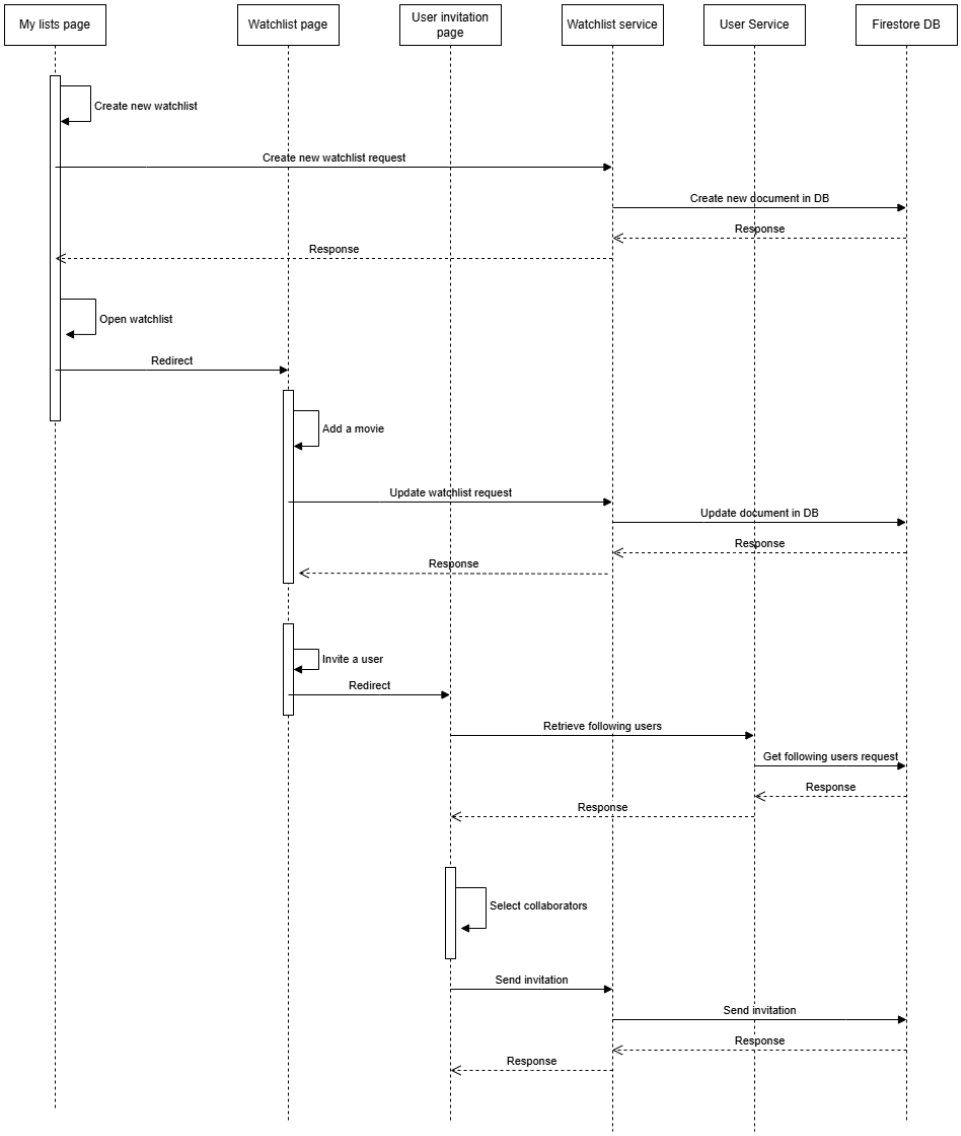


Figure 3.7: Sequence diagram about watchlist management

The watchlist sequence shows how the app handles watchlist-related operations:

- Creating and modifying watchlists.
- Adding/removing movies from watchlists.
- Synchronizing watchlist data with Firestore.
- Handling collaborative features like sharing and following watchlists.

3.4.5. Notification System

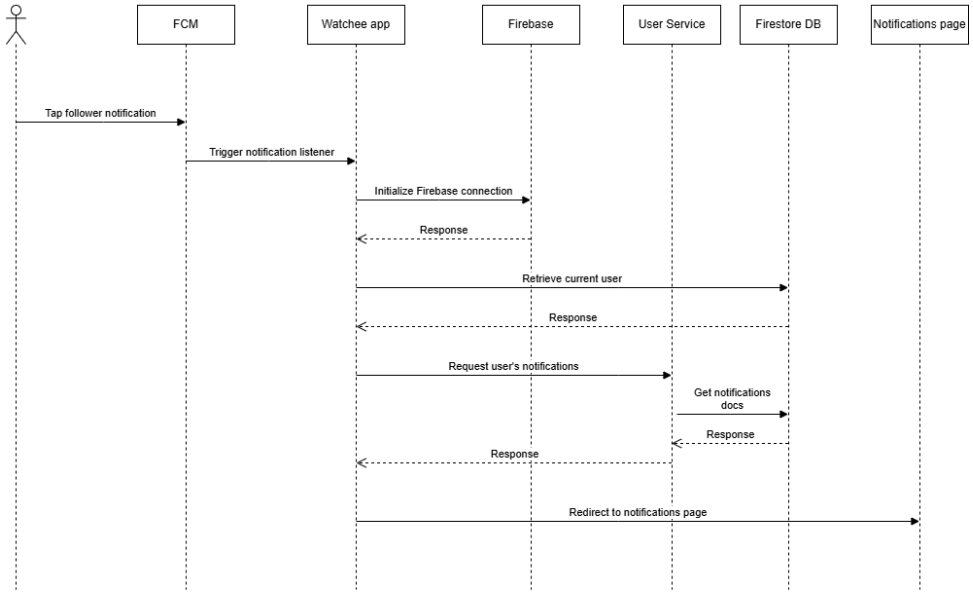


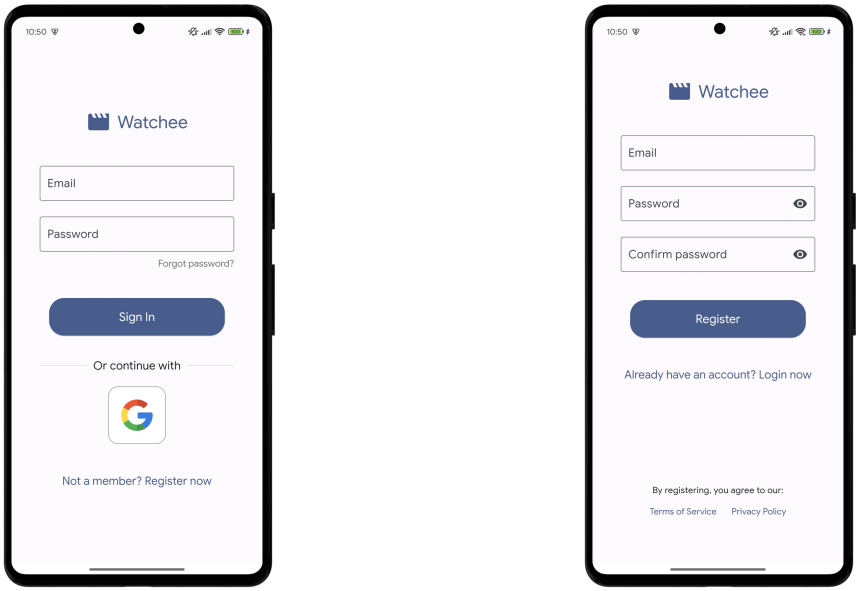
Figure 3.8: Sequence diagram about notification system

This sequence illustrates how the notification system operates. When a user taps on a new follower notification, the app is launched and it will initialize the Firebase connection. The user's latest notifications are retrieved from Firestore and displayed on the notifications page.

4 | Application Design

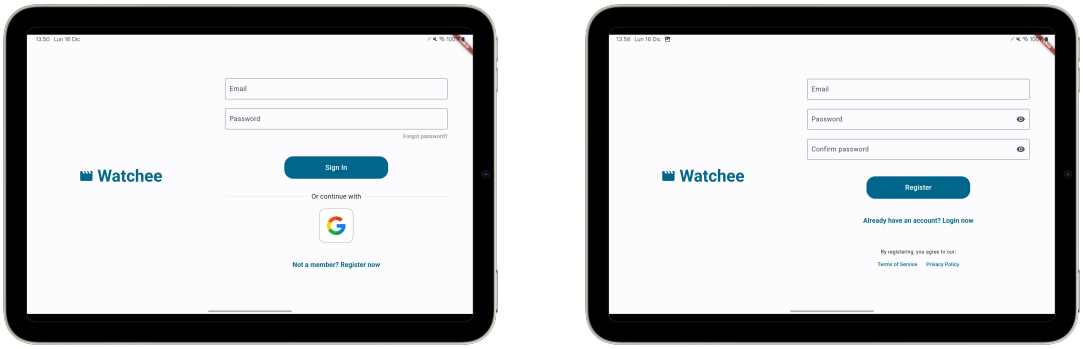
This section showcases the design of the key screens in Watchee. Pages layouts change according to the device, to better exploit screen dimension. More details are shown to users at the same time when using a tablet to take advantage of the landscape mode. Watchee also supports dark mode and dynamic color theme using the Monet feature integrated in the Android system.

4.1. Authentication



(a) Login Screen (b) Registration Screen

Figure 4.1: Login and Registration Screens on phone



(a) Login Screen (b) Registration Screen

Figure 4.2: Login and Registration Screens on tablet

These screens are displayed when the user is not authenticated or after they have signed out. A text button is used to switch from the login to the registration form and vice versa. When using a larger display, we decided to make the form bigger by using a two-column layout, an example of which can be seen in tablet screenshots.

4.2. Home Page

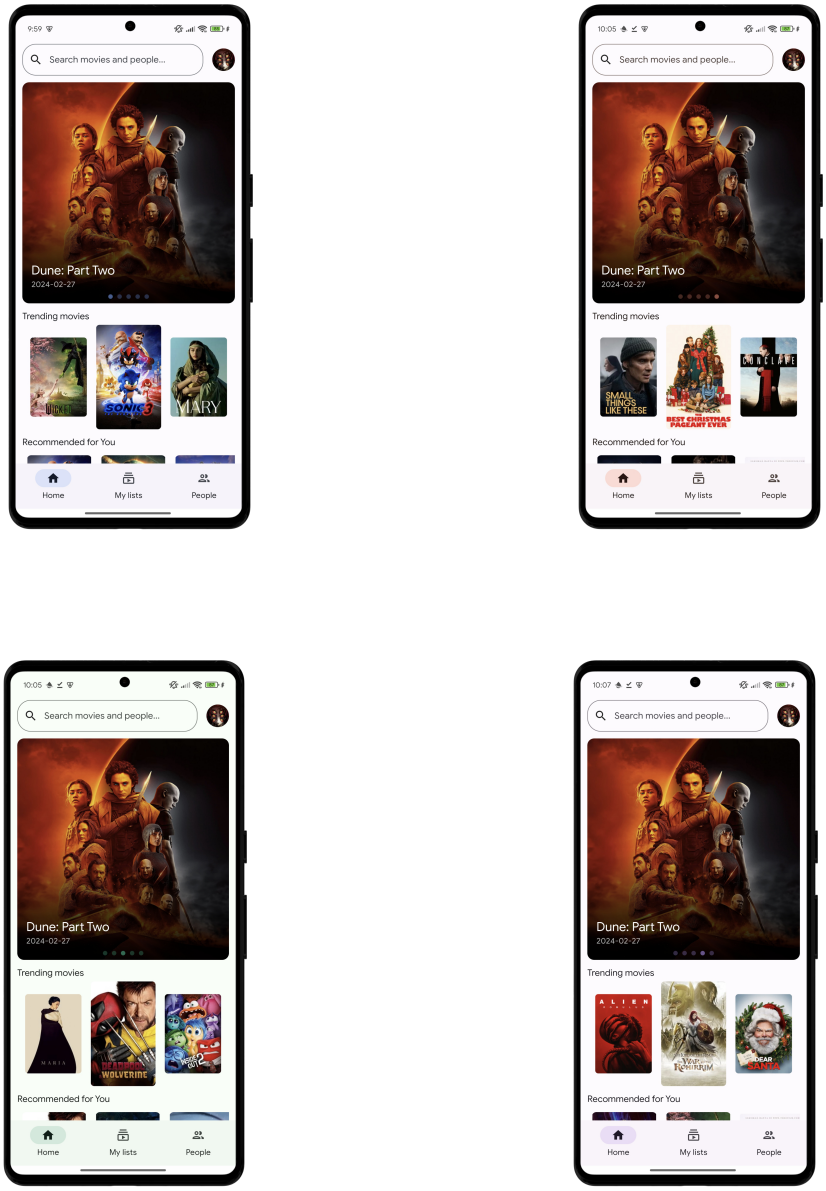


Figure 4.4: Home Screens with different color accents on phone

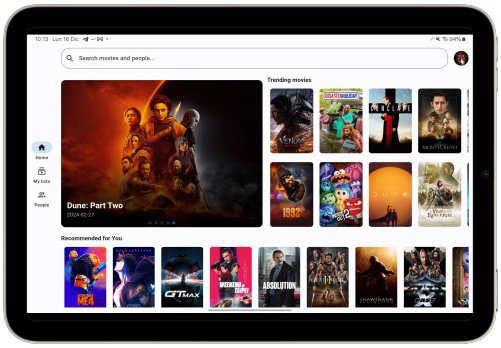
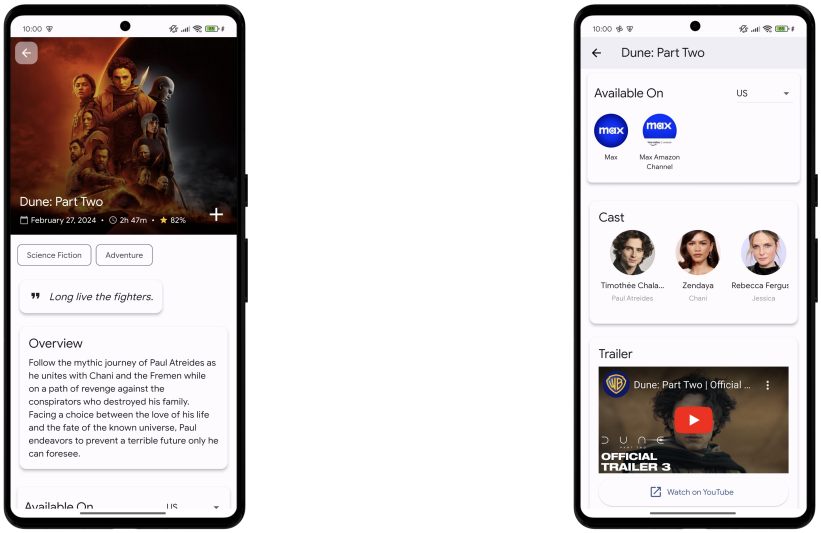


Figure 4.5: Home Screen on tablet

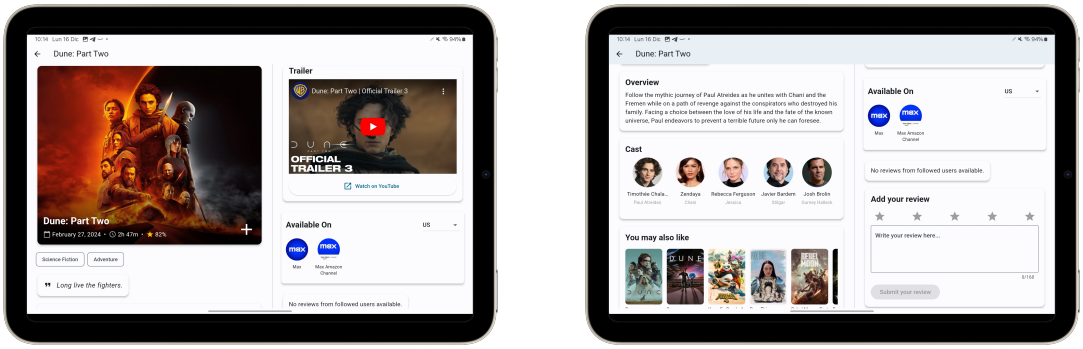
These screens showcase the home page of Watchee. It shows a custom squared widget that contains five of the most popular trending movies. Then it lists other horizontal scrolling widgets, each one focusing on a specific movie category. Home page also includes a personalized slider with recommended movies based on genre preferences previously set by the user. For tablet optimization we decided to create a double row horizontal slider and place it alongside the main one.

4.3. Movie Details



(a) Before scrolling (b) After scrolling

Figure 4.6: Movie details page on phone

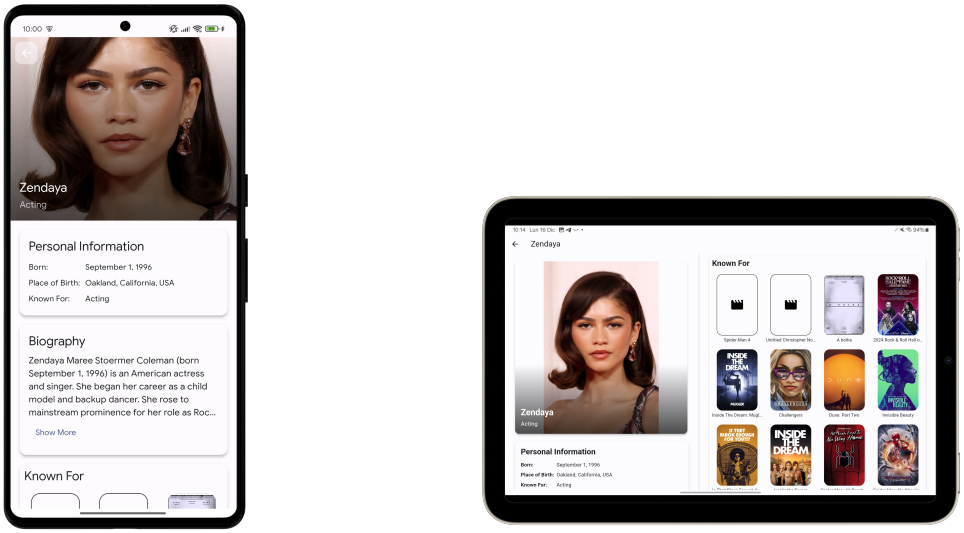


(a) Before scrolling (b) After scrolling

Figure 4.7: Login and Registration Screens on tablet

These screenshots showcase the movie details page, which presents the user key information regarding a given movie. This page can be accessed by tapping on a movie anywhere in the application. The plus button easily allows the user to add the movie to a predefined or personalized list or to remove it. The page contains a widget that shows the availability of the movie on specified platforms in different countries. The user can easily choose the country by tapping on the tabview. The trailer is also embedded and it could be played directly inside the application, or externally through the Youtube link. Tapping on a cast member redirects the user to the corresponding cast details page. In landscape view, we decided to rearrange the page optimizing the horizontal space with two independently scrollable columns, that allow the user to have everything in sight.

4.4. Cast Details



(a) Cast details on phone

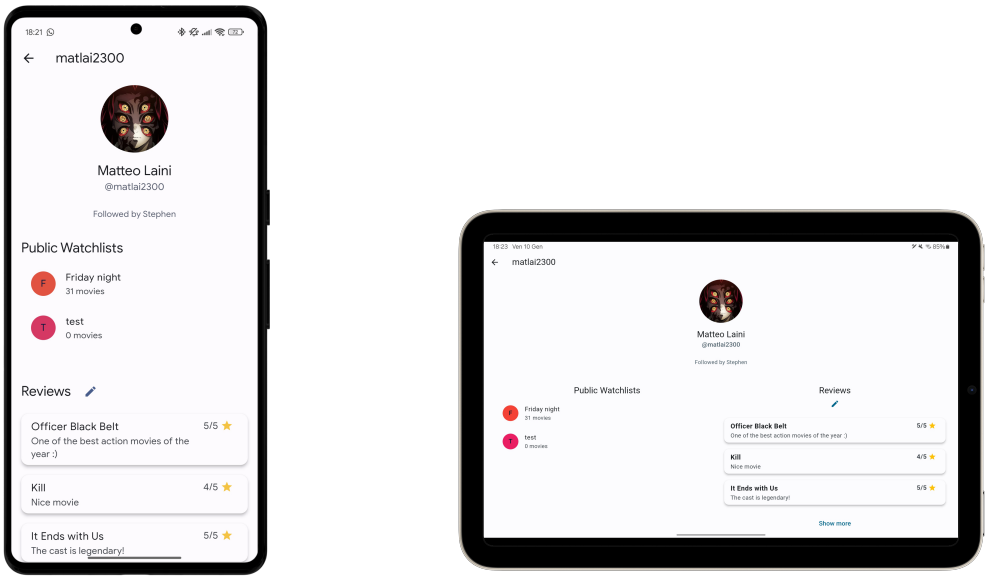
(b) Cast details on tablet

Figure 4.8: Cast details page

These screenshots showcase the cast details page, which presents the user key information regarding a cast member. Apart from biography and personal information, the page includes a widget containing all the filmography of this person. By tapping on a movie icon, the user is redirected to the corresponding movie detail page, granting a seamless in-app experience. Landscape optimization follows the same concept as in the movie details page.

4.5. Profile

4.5.1. User profile



(a) User profile on phone

(b) User profile on tablet

Figure 4.9: User profile page

This page shows all the public user information, such as name, username, public watchlists and reviews. The edit icon is only shown on the logged user profile page and allows to modify or delete personal reviews.

4.5.2. Edit personal information

Figure 4.10: Edit user profile page

This screenshot showcases all the personal information that the user can edit after registering to the application. The user has to choose at least three favourite genres to personalize the Watchee experience and get custom recommended movies.

4.5.3. Settings

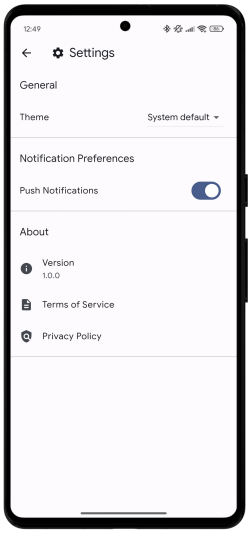
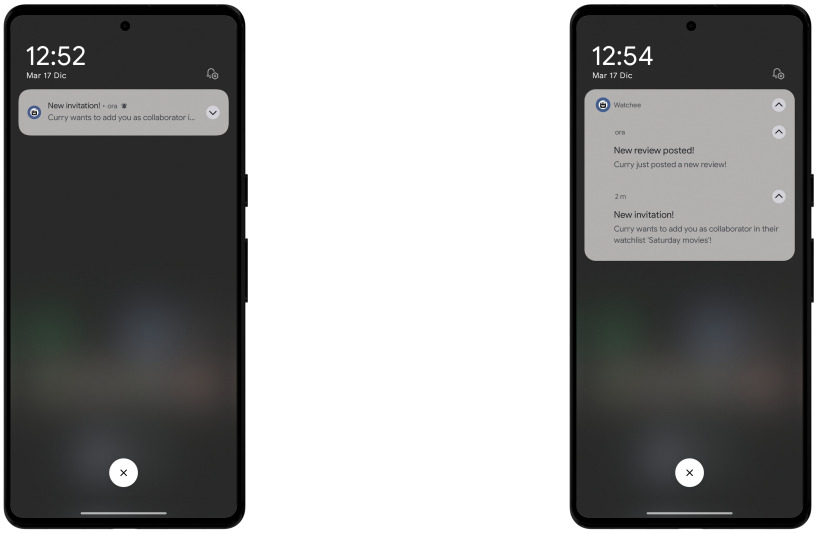


Figure 4.11: Settings page

These screenshot shows the settings interface, which gives users control over their app experience. Users can switch between themes (light, dark or system default), manage their push notification preferences, and access important app information like version details and legal documents.

4.5.4. Notifications



(a) Single notification (b) Multiple notifications

Figure 4.12: Notifications on phone

These screenshots showcase how the user can be informed of new invitation, new review and new follower even when the app is closed. User will also get a notification on the release date of a previously liked movie. Tapping on a notification redirects the user to the notification page.

4.5.5. Notification page

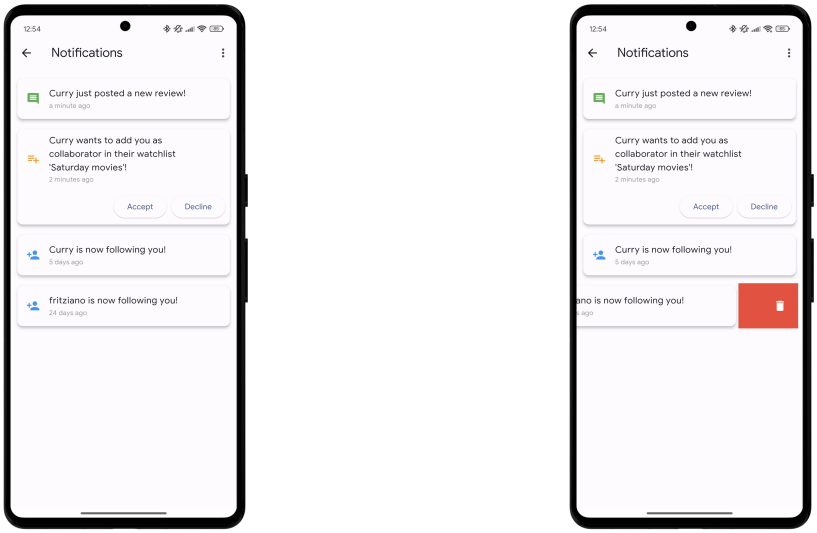


Figure 4.13: Notification page on phone

Notification page groups all the incoming notification of the user. Upon tapping on a follower or a review notification, the user is redirected to the corresponding user profile page. The user can accept or decline a pending invite to join a watchlist as a collaborator directly from this page. Notifications can also be deleted by sliding them to the left.

5 | Testing

Testing played a crucial role in ensuring Watchee's reliability and functionality. Our testing strategy encompassed three primary levels: unit tests, widget tests, and integration tests, with each level serving distinct purposes in validating the application's behavior.

5.1. Unit Testing

In our unit testing efforts we committed to thoroughly testing individual components and their interactions. We utilized Visual Studio Code with the official Flutter extension as our testing environment, which provided a streamlined development and testing workflow. Our unit testing approach covered all major pages and functionality within the application, including:

- Authentication Flow: Login and registration pages were tested to ensure proper validation, error handling, and successful authentication scenarios
- Core Features: Home page, movie details, and review management received thorough testing to verify correct data display and user interactions
- User Management: Profile pages and user settings were tested to confirm proper handling of user data and preferences
- List Management: Tests for watchlist creation, modification, and sharing functionality ensured reliable list management
- Social Features: Notification handling and user interaction components were validated through dedicated test cases

5.2. Widget Testing

With widget testing we focused on UI components and their interactions, verifying:

- Component Rendering: Proper display of movie information, user profiles, and lists
- User Input Handling: Form validation and submission across various screens
- Navigation: Correct routing between different sections of the application
- State Updates: Appropriate UI updates in response to user actions and data changes

5.3. Coverage

Our testing approach yielded high coverage metrics across both unit and widget tests. For unit testing, we achieved 98.3% code coverage across all core functionality, with particularly strong coverage in critical areas.

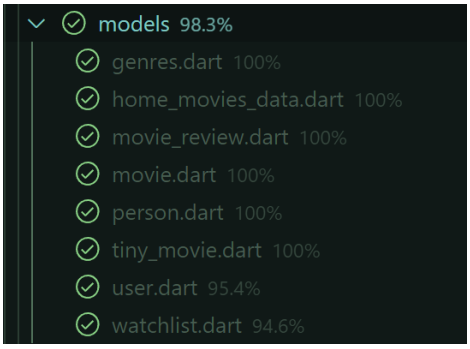


Figure 5.1: Unit tests coverage

Widgets and pages testing similarly demonstrated robust coverage at 90.6% and 76.1% overall.

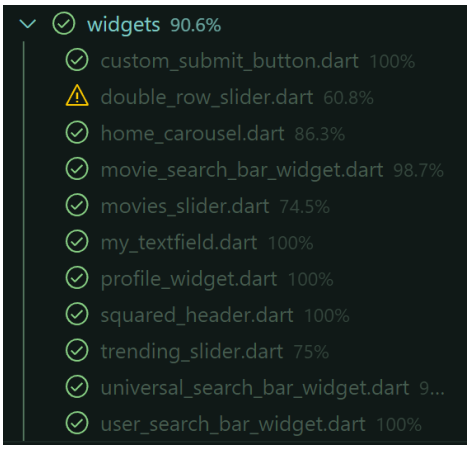


Figure 5.2: Widgets tests coverage

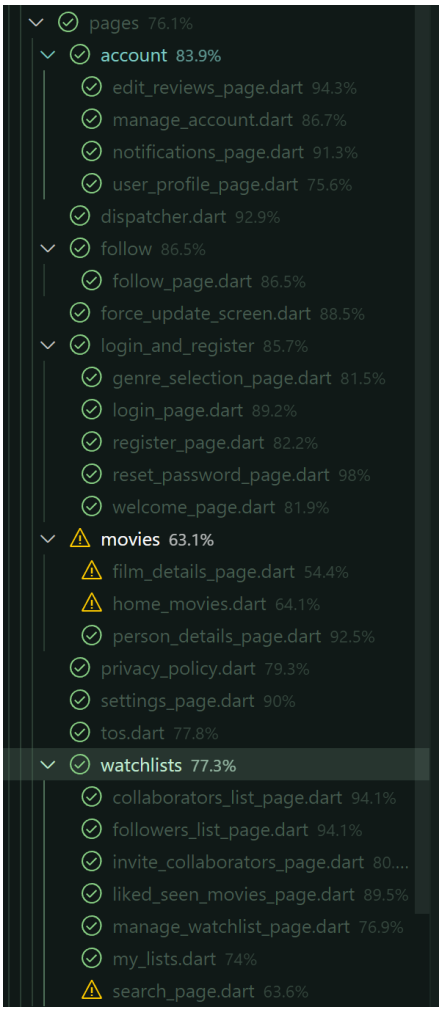


Figure 5.3: Pages tests coverage

While achieving high coverage was important, we remained focused on testing quality and edge cases rather than pursuing coverage metrics alone.

5.4. Integration Testing

Our integration testing strategy prioritized comprehensive edge case coverage over achieving high code coverage metrics.
This approach ensured that Watchee would handle real-world usage scenarios reliably, even in unexpected situations.
Testing was performed using the Flutter emulator to simulate real device conditions.
We focused on four critical user flows:

1. **Authentication Process**
    - Complete login flow including error cases and validation.
    - Registration process with all possible field combinations.
    - Password recovery and account verification scenarios.
2. **User following Process**
    - Following/unfollowing users
    - Interaction with followed users' content
    - Notification handling for social interactions
3. **Watchlist management**
    - Creation of new watchlists
    - Adding and removing movies
    - Sharing watchlists with other users
    - Collaborative editing scenarios
4. **Movie interaction flow**
    - Navigation through movie detail page
    - Adding and removing movie from liked list
    - Navigation through cast member profile

For each flow, we identified and tested edge cases such as:

- Network connectivity issues
- Invalid or unexpected input data
- Permission boundary conditions

This comprehensive approach to integration testing helped ensure that Watchee would provide a robust and reliable user experience across all its core functionalities.

6 | Conclusion

This design document outlines the architecture, features, and implementation details of the Watchee application. It serves as a comprehensive guide for development, maintenance, and future enhancements of the platform.