**MOVIE TRACKER**

| **App Name** | MovieTracker |
| --- | --- |
| **Team Name** | TeamA |
| **Course** | 7SENG999C Mobile Application Development |
| **Member 1 (Name)** | Diwanga Amasith |
| **Member 1 (IIT No / UOW No)** | 20241838 / W2151429 |
| **Member 2 (Name)** | Achini Jayasuriya |
| **Member 2 (IIT No/ UOW No)** | 20233201 / W2157858 |

## **1. INTRODUCTION**

The thorough study and design methodology for our suggested mobile application app named as “Movie Tracker” has been described in this publication. This offers an organized perspective on the key requirements along with the final design diagrams, screenshots and evaluation.

## **2. PROJECT OVERVIEW**

### 2.1 PROBLEM STATEMENT

Movie enthusiasts are struggling to track the watched films and missed the opportunities for customized recommendations according to their viewing preferences and discover the new movies that cater for their interests. Movie Tracker addresses this gap by providing comprehensive cinematic analytics and smart recommendations based on user preferences.

### 2.2 PROPOSED SOLUTIONS

Below are the main features which addresses the issues the movie enthusiasts faces,

* Watch list
* Smart Recommendations
* Stats Dashboard.

This app has been built with Swift and following the MVVM architecture pattern, the application will derive core data for offline persistence and integrate with the IMDB API to provide up-to-date movie information.

## **3. Requirements Analysis**

### 3.1 Functional Requirements

| **ID** | **Requirement** | **Priority** |
| --- | --- | --- |
| FR1 | Users can search and browse movies from the IMDB database | High |
| FR2 | Users can mark movies as watched or add them to the watchlist | High |
| FR3 | Users can create customized movie playlists according to their preferences | Medium |
| FR4 | The system provides personalized movie recommendations | High |
| FR5 | Users can view statistics about their viewing habits | Medium |
| FR6 | Users can filter movies by name, genre, and rating | Medium |
| FR7 | Users can access all features offline (using cached data) | High |
| FR8 | Users can customize app's appearance according to their preference (dark/light mode) | Low |

### 

### 3.2 Non-Functional Requirements

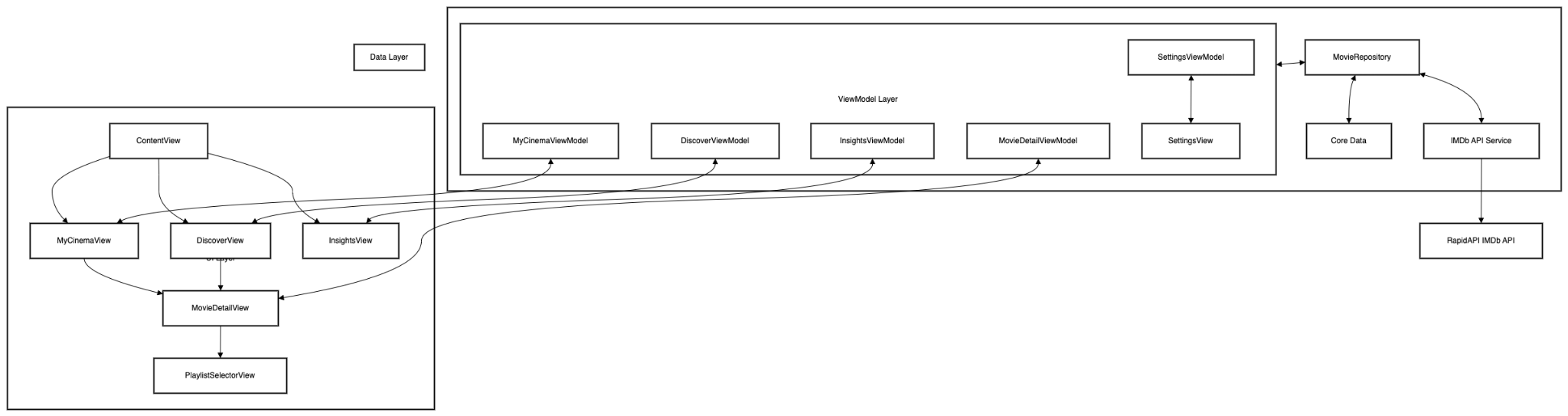
| **ID** | **Requirement** | **Description** |
| --- | --- | --- |
| NFR1 | Performance | App loads 1000+ movies in <1s using Core Data batch fetching |
| NFR2 | Offline Support | 100% functionality without an internet connection |
| NFR4 | Responsiveness | UI responds within 100ms of user interaction |
| NFR5 | Battery Efficiency | Optimized network calls and background processing |
| NFR6 | Security | Local data encryption for user watchlists and preferences |

## 

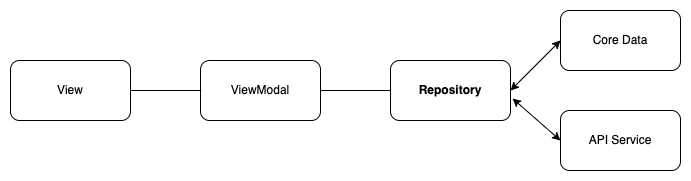
## **4. Design Documentation**

### 4.1 Use Case Diagram

### 4.2 Architecture diagram

Link: [Architecture Diagram](https://drive.google.com/file/d/1AixXf1zef3CTV5I7ka8MurE6XM4RBeHP/view?usp=sharing)

### 4.3 Architecture Overview

Movie Tracker implements the MVVM (Model-View-ViewModel) architecture pattern along with a repository layer for the data management:

#### Key Design Patterns:

1. **MVVM Pattern**: Clear separation of UI (Views), business logic (ViewModels), and data (Models)
2. **Repository Pattern**: Abstraction layer for data operations
3. **Protocol-Oriented Programming**: Interfaces for dependency injection and testing

### Technologies

* **SwiftUI**: Modern declarative UI framework
* **Core Data**: Persistent storage for movie information and user data
* **RapidAPI IMDb API**: External data source for movie information

### Core Data Model

* **Movie**: Stores movie details from the API
* **Playlist**: Represents user-created collections and default lists (Completed/Watchlist)
* **UserStats**: Tracks user viewing statistics

### 

### 

### 

### 

### 4.4 Class Diagram



**5. Technical Implementation**

### 5.1 Core Data Implementation

### 5.2 UI Implementation

* SwiftUI with MVVM binding for reactive updates
* LazyVGrid with prefetching for smooth scrolling of large lists
* Custom animations for transitions between states
* Dark/light mode support with dynamic color assets

### 

**All 10 UI Designs**

### 

### 

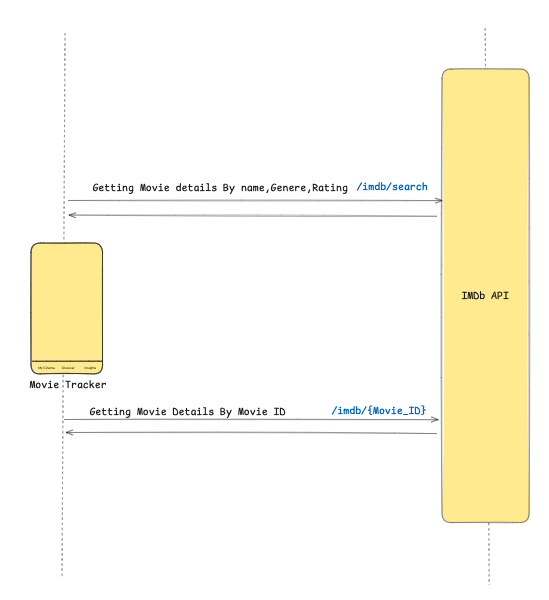
### 

### 

### 5.3 API Integration

The application integrates with the IMDB API:

* Response models with Codable for type-safe parsing



**API Data Flow**

### 

### 

### 

### 

### 

### 

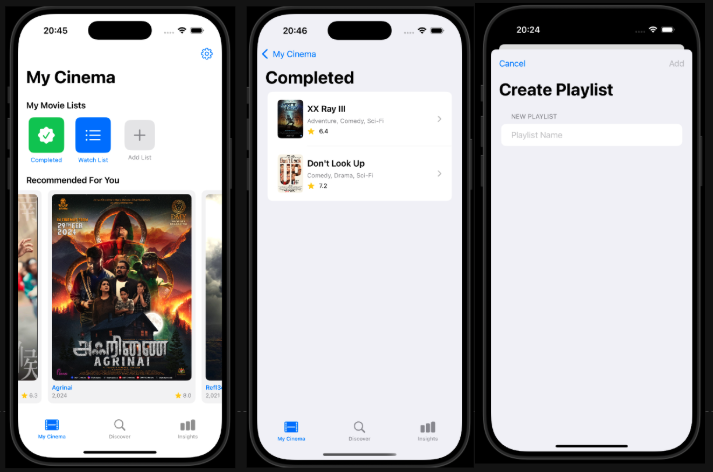
### 5.4 Navigation Flow

## 

## **6. Key Features**

### 6.1 My Cinema

* Personal movie collection management
* Create custom playlists
* Mark movies as watched
* View auto-suggested similar films based on watching history



**My Cinema View**

### 

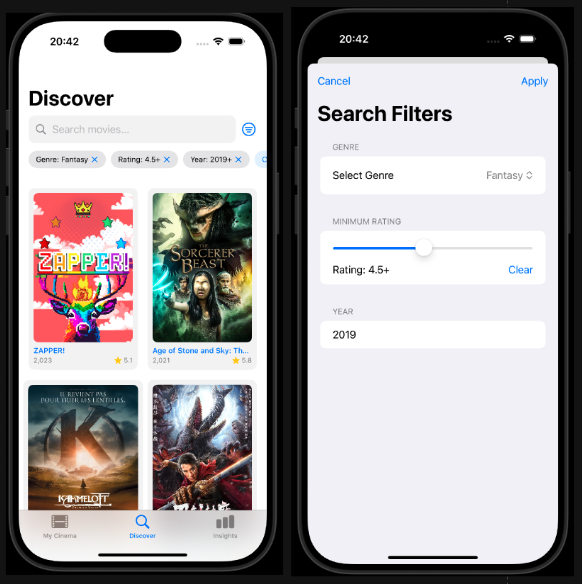
### 

### 

### 

### 6.2 Discover

* Movie search with real-time filtering
* Advanced filters by genre, rating, and release year from.
* Trending and popular movie sections



**Movie Search and Filter Views**

### 

### 

### 6.3 Movie Details View

* Details view showing comprehensive movie information.
* Route by clicking the Movie Tile from Discover or My Cinema Views.
* The User Can Add a Movie to the Default 2 Playlists or custom playlists.

### 

### 

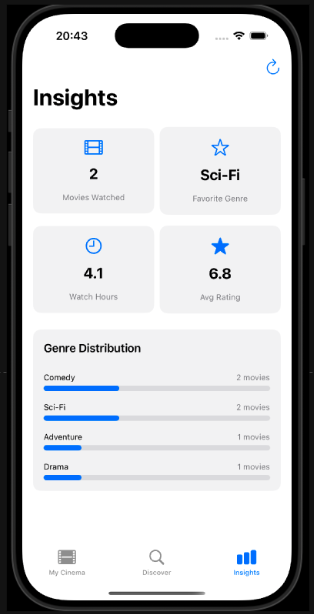
### 

### 

### 

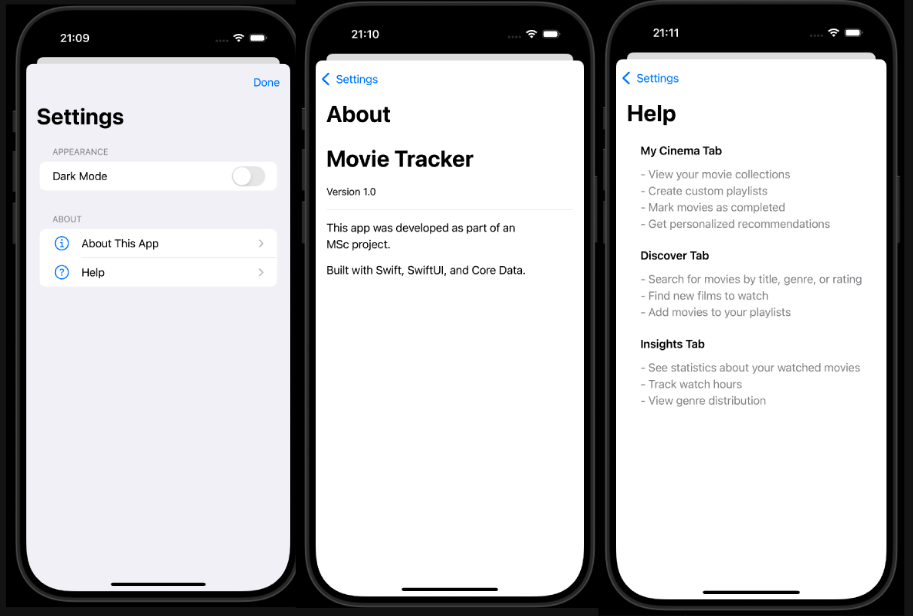
### 6.4 Insights

* Comprehensive viewing statistics:
  + Visualize viewing habit analytics
  + Track the total movies watched and hours
  + See favorite genres and average ratings
  + View genre distribution charts



**Insights View**

### 6.4 Other Views



### 

### 6.4 Dark Mode UI

* All UIs are Support Dark Mode and can be togel from Setting.



**7. Setup Instructions**

### Prerequisites

* Xcode 14.0+
* iOS 16.0+
* RapidAPI Account with IMDb API subscription

### Installation Steps

1. Clone the repository

git clone https://github.com/yourusername/movie-tracker.git

cd movie-tracker

1. Open the project in Xcode

open MovieTracker.xcodeproj

1. Set up your RapidAPI Key
   * Create a configuration file called APIConfig.swift in the Network folder
   * Add your RapidAPI key as follows:

struct APIConfig {

static let rapidAPIKey = "YOUR\_RAPIDAPI\_KEY"

static let rapidAPIHost = "imdb236.p.rapidapi.com"

}

1. Build and run the app on your device or simulator

**8. Critical Evaluation**

The *MovieTracker* app can be considered as a one stop solution for movie enthusiasts by addressing theis common challenges like tracking watched content and discovering new recommendations as per their tastes. Use of SwiftUI, MVVM architecture, and Core Data will be a modern and scalable approach to iOS app development. Key features such as personalized recommendations, user statistics, and offline support will enhance the app’s utility and user experience. Also, the integration with IMDb and the focus on user customization are commendable

However, the app could benefit from more robust consideration of usability testing, error handling. Further, the challenges such as the absence of details on data validation, user feedback mechanisms, and few potential limitations may impact the real-world performance.

Overall, *MovieTracker* shows strong potential, but would be further improved by more emphasis on testing, user engagement strategies, and a complete visual design package.

### 9**. Appendix**

### 9.1 Link to Presentation

* [Movie Tracker Presentation](https://docs.google.com/presentation/d/1YWMKramEHsHgwoRVp-WFrEClYmj1iAt-sBsVf_Cv7xU/edit?usp=sharing)