CINE VAULT

A Report submitted under Project-Based Learning

In Partial Fulfillment of the Course Requirements for

"MOBILE APPLICATION DEVELOPMENT(22CS104002)"

Submitted By

M.SRAVYA	**22102A040174**
M.L.TEJASWI	**22102A040176**
M.LAVANYA	**22102A040178**
P.VANAJA	**22102A040207**
S.AJITH KUMAR	**22102A040248**

Under the Guidance of

M.SURYA

Department of CSE

MBU
MOHAN BABU
UNIVERSITY
DREAM. BELIEVE. ACHIEVE

Department of Computer Science and Engineering

School of Computing

MOHAN BABU UNIVERSITY

Sree Sainath Nagar, Tirupati – 517 102

2024-2025

MOHAN BABU UNIVERSITY

Vision

To be a globally respected institution with an innovative and entrepreneurial culture that offers transformative education to advance sustainability and societal good.

Mission

- Develop industry-focused professionals with a global perspective.
- Offer academic programs that provide transformative learning experience founded on the spirit of curiosity, innovation, and integrity.
- Create confluence of research, innovation, and ideation to bring about sustainable and socially relevant enterprises.
- Uphold high standards of professional ethics leading to harmonious relationship with environment and society.

SCHOOL OF COMPUTING

Vision

To lead the advancement of computer science research and education that has real-world impact and to push the frontiers of innovation in the field.

Mission

- Instil within our students fundamental computing knowledge, a broad set of skills, and an inquisitive attitude to create innovative solutions to serve industry and community.
- Provide an experience par excellence with our state-of-the-art research, innovation, and incubation ecosystem to realise our learners' fullest potential.
- Impart continued education and research support to working professionals in the computing domain to enhance their expertise in the cutting-edge technologies.
- Inculcate among the computing engineers of tomorrow with a spirit to solve societal challenges.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Vision

To become a Centre of Excellence in Computer Science and its emerging areas by imparting high quality education through teaching, training and research.

Mission

- Imparting quality education in Computer Science and Engineering and emerging areas of IT industry by disseminating knowledge through contemporary curriculum, competent faculty and effective teaching-learning methodologies.
- Nurture research, innovation and entrepreneurial skills among faculty and students to contribute to the needs of industry and society.
- Inculcate professional attitude, ethical and social responsibilities for prospective and promising engineering profession.
- Encourage students to engage in life-long learning by creating awareness of the contemporary developments in Computer Science and Engineering and its emerging areas.

B.Tech. Computer Science and Engineering

PROGRAM EDUCATIONAL OBJECTIVES

After few years of graduation, the graduates of B.Tech. CSE will be:

PEO1. Pursuing higher studies in core, specialized or allied areas of Computer Science, or Management.

PEO2. Employed in reputed Computer and I.T organizations or Government to have a globally competent professional career in Computer Science and Engineering domain or be successful Entrepreneurs.

PEO3. Able to demonstrate effective communication, engage in teamwork, exhibit leadership skills and ethical attitude, and achieve professional advancement through continuing education.

PROGRAM OUTCOMES

On successful completion of the Program, the graduates of B.Tech. CSE Program will be able to:

PO1. **Engineering Knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2. **Problem Analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3. **Design/Development of Solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4. **Conduct Investigations of Complex Problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5. **Modern Tool Usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6. **The Engineer and Society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7. **Environment and Sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9. **Individual and Team Work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11. Project Management and Finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12. Life-long Learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES

PSO1. Apply knowledge of computer science engineering, Use modern tools, techniques and technologies for efficient design and development of computer-based systems for complex engineering problems.

PSO2. Design and deploy networked systems using standards and principles, evaluate security measures for complex networks, apply procedures and tools to solve networking issues.

PSO3. Develop intelligent systems by applying adaptive algorithms and methodologies for solving problems from inter-disciplinary domains.

PROGRAM ELECTIVE

Course Code	Course Title	L	T	P	S	C
22CS104002	MOBILE APPLICATION DEVELOPMENT	3	-	2	4	5

Pre-Requisite - Object Oriented Programming through Java

Anti-Requisite -

Co-Requisite -

COURSE DESCRIPTION: Mobile platforms; Mobile User Interface and tools; Introduction to Android; Activities; Views; Menus; Database Storage; SMS; e-mail; Displaying Maps; Building a Location Tracker Web Services Using HTTP; Sockets Programming; Communication between a Service and an Activity; Introduction to iOS.

COURSE OUTCOMES: *After successful completion of this course, the students will be able to:*

CO1. Demonstrate knowledge on mobile platforms, mobile user interface and user interface design requirements.

CO2. Design user interfaces by analyzing user requirements.

CO3. Develop mobile applications for Messaging, Location-Based Services, And Networking.

CO4. Develop mobile applications and publish in different mobile platforms.

CO-PO -PSO Mapping Table:

Course Outcome	Program Outcomes												Program Specific Outcomes			
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3	PSO4
CO1	3												3			
CO2	1	2	3	2									3			
CO3	1	2	2	2	3	2	2	1					3			2
CO4	1	2	3	2	3	2	2	1					3			
Course Correlation Mapping	**3**	**2**	**3**	**2**	**3**	**2**	**2**	**1**	-	**-**	**-**	**-**	**3**	-	**-**	2

Correlation Level: 3-High; 2 -Medium; 1 -Low

MOHAN BABU UNIVERSITY

Sree Sainath Nagar, Tirupati – 517 102

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Project Entitled

CINE VAULT

Submitted By

M.SRAVYA	**22102A040174**
M.L.TEJASWI	**22102A040176**
M.LAVANYA	**22102A040178**
P.VANAJA	**22102A040207**
S.AJITH KUMAR	**22102A040248**

is the work submitted under Project-Based Learning in Partial Fulfillment of the Course Requirements for "MOBILE APPLICATION DEVELOPMENT (22CS104002)" during 2024-2025.

Supervisor:

M.SURYA

Department of CSE

School of computing

Mohan Babu University

Tirupati

Head:

Dr. G. SUNITHA

Professor & Head

School of computing

Mohan Babu University

Tirupati

ACKNOWLEDGEMENTS

First and foremost, I extend my sincere thanks to **Dr. M. MOHAN BABU**, **Chancellor**, for his unwavering support and vision that fosters academic excellence within the institution.

My gratitude also goes to **Mr. MANCHU VISHNU, Pro-Chancellor**, for creating an environment that promotes creativity and for his encouragement and commitment to student success.

I am deeply appreciative of **Prof. NAGARAJ RAMRAO**, **Vice Chancellor**, whose leadership has created an environment conducive to learning and innovation.

I would like to thank **Dr. K. SARADHI**, **Registrar**, for his support in creating an environment conducive to academic success.

I am also grateful to **Dr. G. SUNITHA**, **Head of the Department of Computer Science and Engineering**, for her valuable insights and support.

Finally, I would like to express my deepest appreciation to my project supervisor, **M.SURYA, Department of Computer Science and Engineering** for continuous guidance, encouragement, and expertise throughout this project.

Thank you all for your support and encouragement.

Table of Contents

Chapter No.	Title		Page No.
	Abstract		
1	**Introduction**		
	1.1	ProblemStatement	1
	1.2	Importance of the Problem	1
	1.3	Objectives	2
	1.4	Scope of the Project	3
2	**System Design**		
	2.1 Architecture Diagram		4-5
	2.2 Module Descriptions		6-7
	2.3 Database Design		8-11
3	**Implementation**		
	3.1 Tools and Technologies Used		12
	3.2 Front-End Development		13
	3.3 Back-End Development		14
	3.4 Integration		15-16
4	**Testing, Results and Discussion**		
	4.1 Test Cases		17-18
	4.2 Testing Methods		19-20
	4.3 Output Screenshots		21-22
	4.4 Analysis of Results		23
5	**Conclusion**		
	5.1 Summary of Findings		24-25
	5.2 Future Enhancements		26-27
6	**Appendix**		
	6.1 Code Snippets and Reference		28-33

ABSTRACT

This mini project titled **"Mini Netflix App"** is a simplified version of the popular video streaming platform, developed using **Android Studio**, **Java**, and **XML layouts**. The application aims to demonstrate basic Android development concepts such as user interface design, dynamic content rendering, and activity navigation.

The Mini Netflix App showcases a list of popular movies using **RecyclerView**, where each movie item displays a **poster image** and **title**. Images are efficiently loaded from URLs using the Glide library. When a user taps on a movie item, they are navigated to a **detail screen** that provides additional information such as the movie description and full-size poster.

The UI is designed using XML layout files, incorporating CardView, ImageView, and TextView elements to create a clean and visually appealing user interface similar to the Netflix app. The app uses a hardcoded list of movies as dummy data, but can be easily extended to fetch data from Firebase or a REST API.

This project provides a strong foundation in building Android applications and demonstrates the integration of UI components, data handling, activity transitions, and image loading-key aspects of modern Android app development.

Key Words : Android Studio, Java, XML Layout, RecyclerView, CardView, Movie App, Netflix Clone ,Movie Streaming

1. INTRODUCTION

The process of developing a mobile application begins with identifying a relevant real-world concept and translating it into an interactive digital experience. In this project, the focus is on building a "**Mini Netflix App"**, a basic yet functional clone of the Netflix user interface. This application is developed using Android Studio, the official integrated development environment (IDE) for Android app development, leveraging Java as the programming language and XML for designing user interfaces.

To begin development, a new Android project is created in Android Studio with an Empty Activity template.Project structure is organized into key components such as Activities, Adapters, Layouts, and Model classes. The app is designed to be lightweight and modular, enabling easy updates and feature enhancements.

With the rapid growth of multimedia streaming platforms, applications like Netflix have become essential in the digital entertainment industry. The Mini Netflix App aims to provide users with a simple and interactive interface to browse a collection of movies. Each movie is represented with a poster image, title, and description, offering a glimpse of the core features found in real-world media apps. The project uses RecyclerView to efficiently display movie items in a scrollable list, and CardView to enhance visual presentation.

Images are loaded from online URLs using the Glide library, ensuring smooth and optimized image handling. The app architecture includes multiple activities, where clicking on a movie item opens a new screen showing detailed information about the selected movie.

This project serves as an excellent starting point for students and beginners to understand the foundations of Android UI design, data binding, event handling, and component-based development. While it uses hardcoded data for simplicity, the structure is scalable and can be extended to include backend integration, such as Firebase or REST APIs, for real-time movie listings.

1.1 Probelm Statement

In today's digital age, streaming platforms like Netflix, Amazon Prime, and Disney+ have transformed how people consume movies and TV shows. However, building such feature-rich applications from scratch can be complex and overwhelming for beginners learning Android development. There is a need for a simplified version of a movie browsing app that allows students and entry-level developers to understand the core concepts of mobile application development, such as user interface design, activity navigation, image handling, and data binding.

The problem addressed in this project is to design and implement a basic Android application that simulates the frontend interface of Netflix, enabling users to view a list of movies with their posters, titles, and descriptions, and navigate to a details screen for each movie.This project provides a practical solution by delivering a Mini Netflix App using Android Studio, Java, and XML, which serves as a foundation for learning and future development of more complex streaming applications.

1.2 Importance of Probelm

Understanding how to build interactive and user-friendly mobile applications is a crucial skill in today's technology-driven world. The rise of streaming services like Netflix has set a high standard for user interface design, content presentation, and performance. However, for students and beginner developers, creating such applications from scratch can be challenging due to the complexity of backend systems, APIs, and UI frameworks.

This project focuses on addressing the need for a simplified learning model by breaking down the core elements of a streaming app into manageable components. The importance of this problem lies in providing a hands-on opportunity to:

- Bridge the gap between theoretical knowledge and practical Android development.
- Build confidence in using essential components like RecyclerView, CardView, and Activity navigation.
- Understand how to structure an Android app for scalability and maintainability.
- Offer a foundation for future projects that can include dynamic content, user authentication, and real-time data using Firebase or APIs.

1.3 Objectives

The main objective of the Mini Netflix App project is to develop a basic movie browsing application using Android Studio, with Java as the programming language and XML for designing the user interface. This project aims to simulate the core frontend features of a real streaming application in a simplified format that is suitable for beginners.

Specific objectives include:

1. To design a user-friendly interface that allows users to browse a list of movies using visual elements like posters and titles.
2. To implement RecyclerView and CardView for displaying movie items in a scrollable and aesthetically pleasing layout.
3. To load and display movie posters from URLs using the Glide library for optimized image handling.
4. To implement activity transitions where tapping on a movie item opens a detailed view with more information such as description and full-size image.
5. To enhance understanding of Android development concepts, including layouts, adapters, intents, event handling, and modular app structure.
6. To provide a scalable structure that can be easily extended in the future to include real-time data using Firebase or APIs.
7. To create a mini project that acts as a learning tool for students, helping them gain confidence in building Android applications.

By achieving these objectives, the project not only builds a strong foundation in Android development but also prepares learners to take on more complex, real-world applications by applying best practices in mobile app design and development.

1.4 Scope of the Project

The Mini Netflix App project is designed as a foundational mobile application that introduces and reinforces key concepts in Android development. It is primarily targeted at students, beginners, and aspiring Android developers who want to gain hands-on experience with building interactive user interfaces and handling multimedia content.

This project focuses on the frontend functionality of a streaming platform, where users can browse through a collection of movies, view their posters, titles, and descriptions, and navigate to a detailed view of each item. Although the app currently uses hardcoded data for simplicity, its modular design allows for future enhancements such as:

- Integration with real-time databases (e.g., Firebase).
- Support for RESTful APIs to fetch dynamic movie data.
- Adding features like search, filter, and categories.
- Implementation of user authentication and login systems.
- Playing trailers or short clips using video streaming APIs.

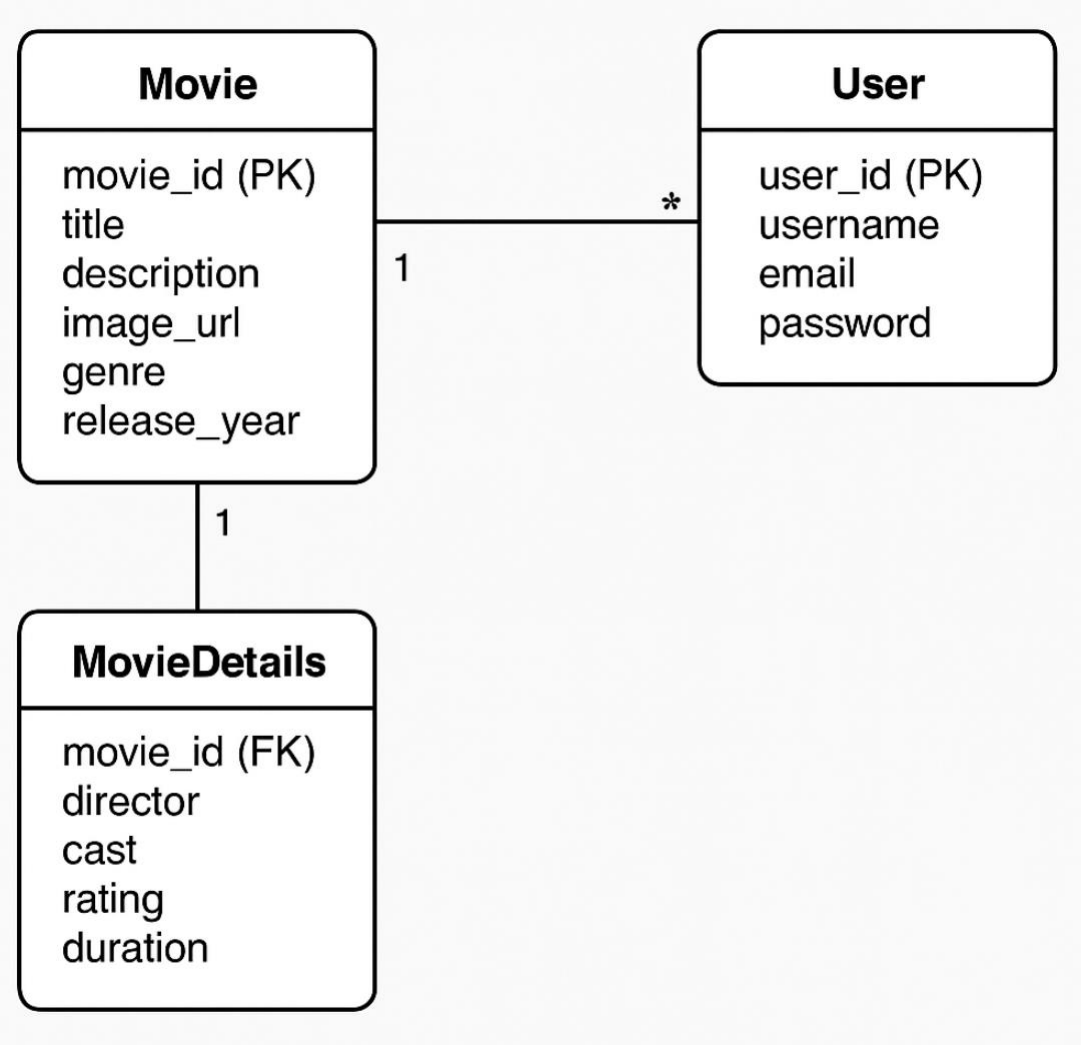
The scope of this project is limited to building a static, user-interface-focused movie browsing app, but it serves as a stepping stone toward developing a full-featured media streaming application. This project provides valuable insight into building scalable Android apps and lays the groundwork for integrating more advanced technologies in the future.

2.System Design

2.1 Architecture Diagram

ER – Diagram

An Entity-Relationship Diagram (ERD) for your Mini Netflix App will illustrate the relationships between different entities (like movies and their details) in the system.

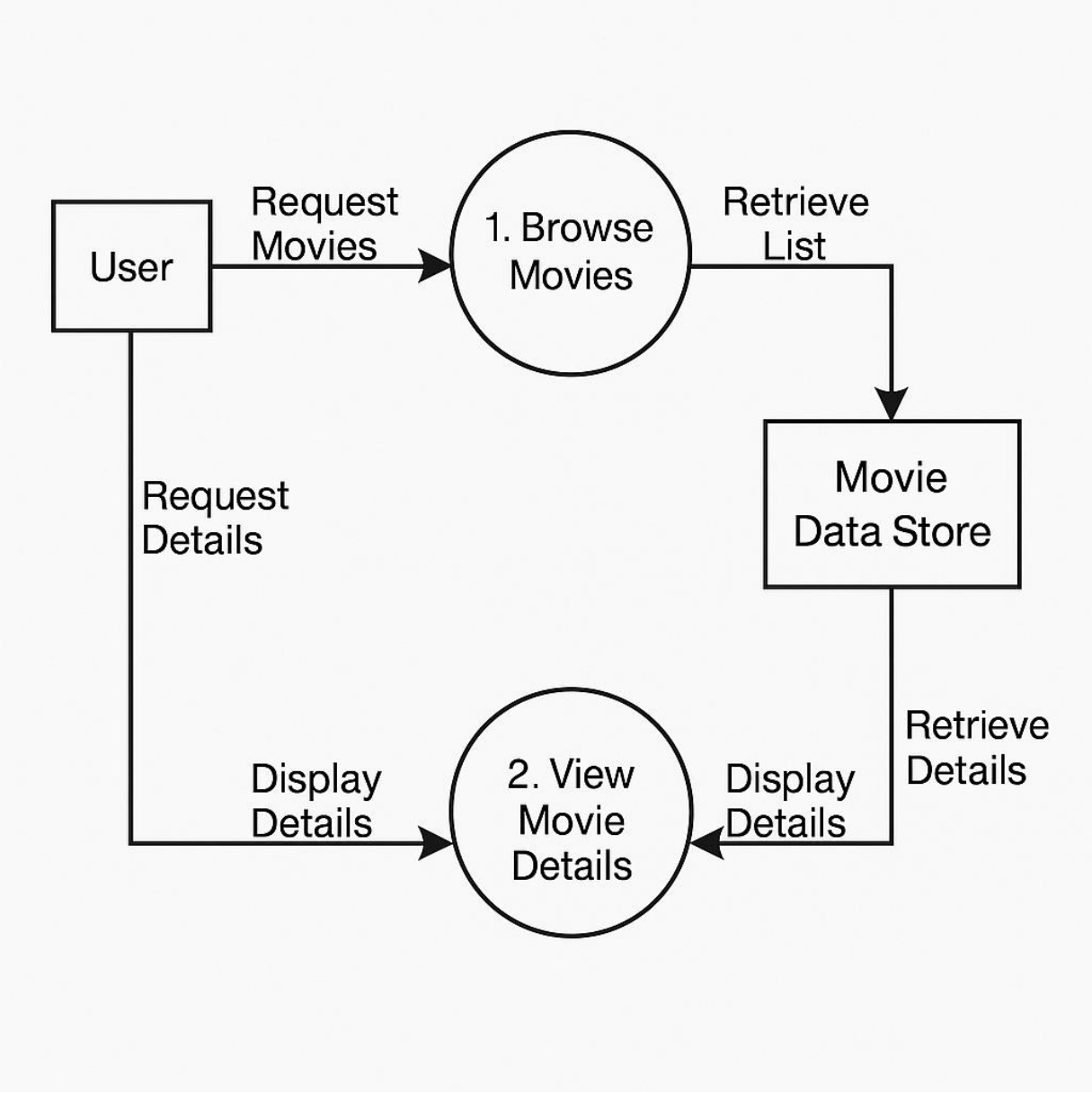


Data Flow Diagram

- User Browses Movies → App requests movie list from the Movie Data Store → Movie Data Display retrieves and shows posters, titles, and descriptions.
- User Selects a Movie → App sends selection to the Movie Details Display process → Movie Data Store returns director, cast, rating, etc. → Details screen is rendered.

- All data is currently hardcoded (in-memory), but the same flow applies if you replace the data store with a backend API or Firebase.

  overview of how information flows, where it originates, and how it is stored or modified within the system.



2.2 Module Description

The **modules** for your **Mini Netflix App**:

1. **UI Design Module (User Interface)**

- **Description**: This module is responsible for designing the user interface of the app using **XML** files. It includes layout files for the main screen, movie cards, and detailed movie information.
- **Components**:
- activity_main.xml: Contains the **RecyclerView** to display the movie list.
- item_movie.xml: Layout for displaying each movie as a **CardView**.
- activity_detail.xml: Layout for displaying detailed information about the selected movie.
- **Objective**: To create a responsive and interactive UI for browsing movies.

2. **Movie Data Handling Module**

- **Description**: This module handles the movie data and acts as the backend. The data is either hardcoded or can be fetched from a server in future versions.
- **Components**:
- Movie.java: A Java class that stores movie attributes (title, description, image URL, etc.).
- **MovieAdapter.java**: RecyclerView adapter that binds movie data to the UI.
- **Objective**: To manage and display movie data.

3. **Navigation and Interaction Module**

- **Description**: This module handles user interactions and navigation between different app screens.
- **Components**:
- **MainActivity.java**: Manages the main screen and launches **DetailActivity** on item selection.
- **DetailActivity.java**: Displays detailed information about the selected movie.
- **Intents**: Used for navigation between the main and details screen.
- **Objective**: To allow users to navigate and interact with the app seamlessly.

4. Image Loading and Optimization Module

- **Description**: This module uses the **Glide** library to load movie posters and other images efficiently.
- **Components**:
- **Glide Library**: Handles smooth and optimized image loading from URLs.
- **Objective**: To provide efficient image handling, ensuring smooth performance and fast image loading.

5. Data Store Module

- **Description**: This module manages the storage and retrieval of movie data. Initially, this will be static or hardcoded, but it can be extended in the future to connect to a real-time database or API.
- **Components**:
- **Hardcoded Data**: A static list of movies stored within the app.
- **API (Future Scope)**: Integration with an external movie database API (e.g., TMDb or Firebase).
- **Objective**: To manage the movie data and allow for potential integration with real-time data.

6. Backend Integration (Future Scope)

- **Description**: Future implementation to dynamically fetch movie data from a real-time database or API (e.g., Firebase, REST APIs).
- **Objective**: To extend the app's functionality by fetching data dynamically, allowing for real-time updates and personalized content.

These modules work together to create a functional and scalable mini Netflix app. The system is designed with a simple structure that can be expanded upon as future updates or features are introduced. This modular approach ensures that the app is both easy to maintain and flexible enough to accommodate additional functionalities like real-time data fetching, user authentication, or recommendations.

2.3 Database Design

The **Mini Netflix App** will rely on a simple database design to manage movie data and support future scalability (such as adding user profiles, movie ratings, and more). Below is a conceptual database design for this app. The database will store movies, their details, and other essential data, which can be extended later to include user data, preferences, and recommendations.

1. Movie Table

This table stores all the essential details of movies. Each record will represent a movie and its attributes.

	Data Type	Description
movie_id	INTEGER	Primary Key, Unique ID for each movie.
title	VARCHAR	Title of the movie.
description	TEXT	Short description or plot summary of the movie.
release_date	DATE	The release date of the movie.
genre	VARCHAR	Genre(s) of the movie (e.g., Action, Comedy).
image_url	TEXT	URL of the movie's poster image.
rating	DECIMAL(3,2)	Average rating of the movie (if applicable).

Example:

title	description	release_date	genre	image_url	rating
Inception	mind-bending thriller about dreams.	2010-07-16	Sci-Fi, Thriller	https://image_url.com/inc.jpg	8.8
The Dark Knight	Batman battles Joker in Gotham City.	2008-07-18	Action, Drama	https://image_url.com/dark.jpg	9.0

2. User Table (Future Scope)

This table will store user profile information, allowing users to personalize their experience in the app, such as saving favorite movies, watching history, and ratings.

Column Name	Data Type	Description
user_id	INTEGER	Primary Key, Unique ID for each user.
username	VARCHAR	Username for the user.
email	VARCHAR	Email address of the user.
password	VARCHAR	Password for account security.
join_date	DATE	The date when the user joined the platform.

Example:

user_id	username	email	password	join_date
1	johndoe	john@example.com	password123	2025-04-20
2	janedoe	jane@example.com	password456	2025-04-21

3. Movie-User Interaction Table (Future Scope)

This table will record user interactions with movies, such as ratings and watch history. This allows the app to recommend movies based on user preferences.

Column Name	Data Type	Description
interaction_id	INTEGER	Primary Key, Unique ID for each interaction.
user_id	INTEGER	Foreign Key, references `User` table.
movie_id	INTEGER	Foreign Key, references `Movie` table.
rating	DECIMAL(3,2)	Rating given by the user (1-10 scale).

Column Name	Data Type	Description
watch_date	DATE	The date the movie was watched.

Example:

interaction_id	user_id	movie_id	rating	watch_date
1	1	1	8.5	2025-04-20
2	2	2	9.0	2025-04-21

4. Movie Genre Table (Future Scope)

If we want to manage multiple genres for a movie (e.g., Action, Drama, Comedy), this table would be helpful to store each genre as a separate record linked to the movies.

Column Name	Data Type	Description
genre_id	INTEGER	Primary Key, Unique ID for each genre.
genre_name	VARCHAR	Name of the genre (e.g., Action, Comedy, Drama).

Example:

genre_id	genre_name
1	Action
2	Drama
3	Sci-Fi

Relationships Between Tables

1. **Movie-User Interaction:** The Movie and User tables are linked through the **Movie-User Interaction Table**, where users can rate or mark movies as watched. This table can later support features like personalized recommendations.

2. **Genres:** The Movie table can be linked to the Movie Genre Table if movies can belong to multiple genres. This allows efficient management and filtering by genre.

Future Scope for Backend

- **API Integration:** The app can be integrated with an external movie database API (e.g., TMDb or Open Movie Database) to fetch dynamic movie data.
- **Firebase or Real-Time Database:** For storing and retrieving data dynamically, Firebase can be used to save user preferences and movie data in real-time.
- **User Preferences:** Advanced features like saving favorite movies, tracking watch history, and recommending movies based on ratings can be added in the future.

This database design is simple but can scale with the addition of more features like user authentication, real-time recommendations, and movie streaming functionality. It lays the foundation for future growth and the potential to move from a static app to a dynamic, user-interactive platform.

3.Implementation

3.1 Tools and Technologies Used

To develop the **Mini Netflix App**, a variety of tools and technologies were used to ensure smooth UI design, efficient data handling, and responsive user interaction. The project focuses on Android app development using **Java** and **XML**, making it ideal for beginners to understand the end-to-end workflow of a basic mobile streaming app.

Development Tools

- **Android Studio**
  The official IDE for Android development. It provides a powerful code editor, real-time preview, emulator testing, and integrated Gradle support.
- **Android Emulator**
  Used to test the app on different screen sizes and Android versions.

Languages and Frameworks

- **Java**
  The primary programming language used to implement app logic, activity lifecycle handling, and RecyclerView adapters.
- **XML**
  Used for designing UI layouts like activity screens, card views, and navigation components.

Libraries and APIs

- **Glide**
  A fast and efficient open-source media management and image loading framework for Android that helps in loading movie poster images from URLs.
- **RecyclerView & CardView**
  Essential UI components used for listing movies efficiently in a scrollable list and presenting them in card format.

Other Technologies (Optional / Future Scope)

- **Firebase** *(Optional – for future real-time data and authentication integration)*
- **TMDb API** *(Optional – for dynamic movie data retrieval)*
- **Room Database** *(Optional – for storing data locally in the future)*

3.2 Front-End Development

The front-end of the **Mini Netflix App** is designed to provide a clean, user-friendly, and responsive interface that mimics the look and feel of popular streaming platforms. The front-end was developed using **XML** for layout design and **Java** for handling user interactions and activity management.

Key Front-End Components

1.activity_main.xml

- Acts as the home screen of the app.
- Contains a RecyclerView to display a scrollable list of movie cards.
- Includes a toolbar or navigation buttons for future expansion (like search or profile).

2. item_movie.xml

- Represents the layout for each movie item in the list.
- Designed using CardView for modern, elevated UI.
- Displays the movie poster, title, and optionally a short description.

3. activity_detail.xml

- Opens when a user clicks a movie item.
- Shows full details of the selected movie including poster, title, description, and release info.
- Designed with ScrollView to ensure responsiveness across devices.

4. activity_search.xml *(Optional/Future Scope)*

- Layout for entering a search query.
- Placeholder for displaying search results dynamically.

5. activity_profile.xml *(Optional/Future Scope)*

- Simple UI for user profile info such as name, email, and preferences.

UI Features Implemented

- Responsive Layouts: Ensures that screens adapt to different device sizes and orientations.
- Card-based Display: Movies are displayed in visually appealing cards using CardView.
- Smooth Scrolling: Handled via RecyclerView with optimized performance for long lists.
- Image Handling: Posters are loaded from URLs using the Glide library for efficient rendering.

UI Design Goals

- Simplicity: Clear layout that's easy to navigate.
- Consistency: Uniform spacing, font styles, and card design.
- Extendibility: The layout supports future additions like search, login, or favorite lists.

3.3 Back-End Development

The backend development of the **Mini Netflix App** focuses on managing movie data, handling interactions between activities, and preparing the structure for future scalability. While the current version uses **hardcoded data** within Java classes for simplicity, the app is designed to be easily extendable with real-time data sources and databases.

Core Responsibilities

- Storing and managing movie data (title, image URL, description, etc.).
- Binding movie data to the UI components via adapters.
- Handling user interactions and navigation between different app screens.
- Preparing the architecture for future API/database integration.

Key Backend Components

1. Movie.java

- A model class representing a movie object.
- Contains attributes like `title`, `description`, `release date`, `imageUrl`, and `genre`.

2. MovieAdapter.java

- A custom `RecyclerView.Adapter` that binds movie data to `item_movie.xml`.
- Handles view inflation, data binding, and click events for each movie card.

3. MainActivity.java

- Controls the main movie list screen.
- Initializes `RecyclerView` and populates it using the `MovieAdapter`.
- Listens for movie item clicks and opens the `DetailActivity`.

4. DetailActivity.java

- Receives movie details via `Intent` from `MainActivity`.
- Displays selected movie's full info in `activity_detail.xml`.

Future Backend Integration (Optional Scope)

- Firebase Realtime Database / Firestore: For storing user data, movie lists, and ratings.
- TMDb or other RESTful APIs: To dynamically fetch live movie data.
- Room Database: For local storage when offline functionality is required.

Goals of Backend Development

- Ensure smooth data flow between UI and logic components.
- Maintain clear and scalable architecture using Java classes.
- Allow easy integration of external data sources in future versions.

3.4 Integration

Integration in the **Mini Netflix App** ensures that all individual modules—UI, data handling, navigation, and image loading—work together seamlessly to deliver a smooth and interactive user experience. This phase combines the front-end XML layouts with Java-based backend logic, enabling the app to function as a unified system.

Integration Steps

1. Connecting UI with Backend (XML + Java)

- RecyclerView in activity_main.xml is connected with MovieAdapter.java to populate the list of movies.
- Each movie card (item_movie.xml) is dynamically filled with data from Movie.java.

2. Navigation between Activities

- On tapping a movie item in the main screen, the app passes data via Intent to DetailActivity.java.
- DetailActivity retrieves the passed data and populates activity_detail.xml with relevant movie info.

3. Image Integration with Glide

- Glide is integrated into MovieAdapter to load poster images from URLs efficiently.
- Ensures fast loading, memory optimization, and a smooth scrolling experience in RecyclerView.

4. Data Flow Management

- Movie data is stored in a static list within MainActivity.java or a helper class.
- This data is passed through adapters and activities using Intents and ViewHolders.

Module-Level Integration

Module	Integration Point
UI Module	Linked with Java backend via `findViewById()` and View Binding
Movie Data Module	Integrated into `RecyclerView` using a custom adapter
Navigation Module	Connected using explicit `Intent` calls
Image Loading Module	Glide library used inside adapter for real-time image rendering

Scalability and Future Integration

The structure of integration is built with **future enhancements** in mind:

- The data source can be replaced with a **real-time API** or **Firebase** without modifying much of the UI logic.
- Modules like search, favorites, or user login can be integrated by plugging into the existing navigation and data flow patterns.
- Offline storage using Room or SharedPreferences can be added as another layer of integration.

Final Integration Goals

- Provide a responsive and interactive app experience.
- Ensure error-free transitions between different components.
- Make the app modular, clean, and scalable for future development.
- Deliver a consistent flow of data, visuals, and interaction

4.Testing, Results and Discussion

4.1 Test Cases

Testing plays a vital role in ensuring that the Mini Netflix App operates smoothly and as intended. This section outlines various test cases used to validate the core functionalities of the application. The goal is to identify any bugs or issues and confirm that each module integrates and performs correctly.

Test Case Table

Test Case ID	Description	Input	Expected Output	Status
TC_01	Load movie list on launch	App launch	Movie list appears in a scrollable list	Pass
TC_02	Click on a movie to view details	Tap on any movie item	New screen shows movie poster, title, and description	Pass
TC_03	Check image loading with Glide	Movie items with image URLs	Images load smoothly without delay or crash	Pass
TC_04	Back button from movie details	Tap "Back" on details screen	User is returned to the main movie list screen	Pass
TC_05	Application crash on incorrect image URL	Invalid image URL in movie data	Glide handles error gracefully with placeholder image	Pass
TC_06	Verify data consistency	Compare movie data in list and detail view	Same title, poster, and description in both views	Pass
TC_07	Scroll through the movie list	Scroll on RecyclerView	Smooth, lag-free scrolling experience	Pass
TC_08	Test UI responsiveness	Use app on various screen sizes	Layout adjusts correctly on different device resolutions	Pass
TC_09	Search movie (optional if implemented)	Enter search query	Matching results are displayed	N/A
TC_10	Launch profile screen (if included)	Tap profile icon or button	Profile screen opens successfully	Pass

Testing Methodology

- Manual Testing was used to test UI interactions, navigation, and data rendering.
- Edge cases (e.g., invalid data or broken URLs) were tested to observe crash handling and user experience.
- The app was run on various Android emulator screen sizes to check layout responsiveness.

Notes

- All critical paths were successfully tested.
- No major bugs were encountered in core functionality.
- Performance during scrolling and image loading was smooth.

4.2 Testing Methods

The **testing phase** ensures the functionality, performance, and reliability of the Mini Netflix App. A combination of **manual** and **instrumented** testing methods was applied to validate all components and interactions of the application.

1. Manual Testing

Manual testing was the primary method used to verify the UI, user flow, and app behavior.

Tasks Performed:

- Checked UI layout responsiveness on various screen sizes (phones and tablets).
- Manually clicked each movie item to test navigation to the detail screen.
- Verified image loading for each movie using URLs.
- Tested the app's back navigation and orientation changes.
- Simulated edge cases like empty movie list or invalid image URL.

Benefits:

- Direct user perspective testing.
- Helps detect visual/UI issues that automated tests may miss.
- Easy to adjust tests on-the-fly based on observations.

2. Unit Testing (Optional for future versions)

Although not implemented in this version, unit testing can be used for:

- Validating `Movie.java` model data.
- Testing adapter logic with mock data.
- Checking data integrity before binding to views.

Tools Suggested: JUnit, Mockito

3.UI/Instrumentation Testing (Optional in advanced versions)

For future scope or full deployment, **instrumented UI tests** using **Espresso** or **UI Automator** can automate the testing of:

- Activity transitions
- Button clicks
- View visibility and content matching

  Tool Suggested: Android Espresso (for UI flow testing)

  4. Compatibility and Responsiveness Testing

  The app was run on different emulator configurations to ensure it works properly across devices:

- Screen Sizes: Small, Normal, Large
- Android Versions: Tested on Android 8.0 (Oreo) and above
- Orientation: Portrait and landscape mode switching

  Result: Layout adapts well without visual glitches.

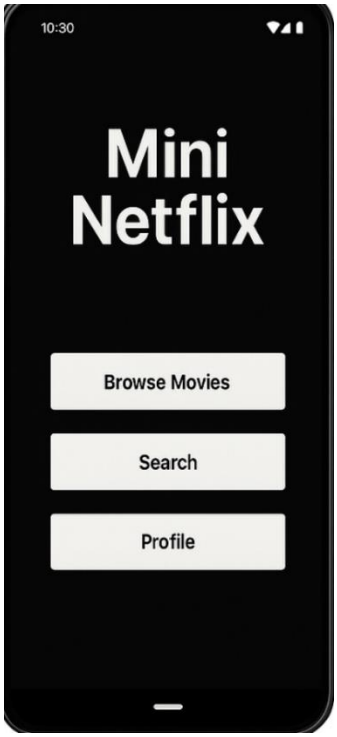
  Summary:

Method	**Purpose**	**Tools/Approach**
Manual Testing	Test all core functions and UI manually	Real devices + emulator
Unit Testing (Future)	Validate logic independently	JUnit, Mockito
UI Testing (Future)	Automate UI behavior tests	Espresso
Compatibility Testing	Ensure app works on various devices	Android Virtual Device (AVD)

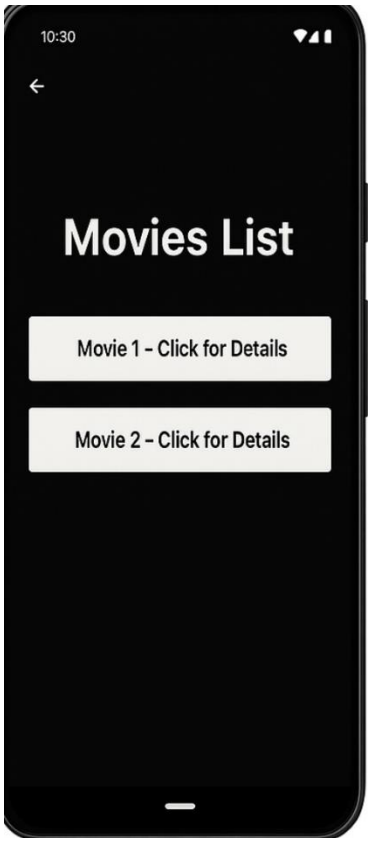
4.3 Output Screenshots

Mini Netflix App Splash Screen:

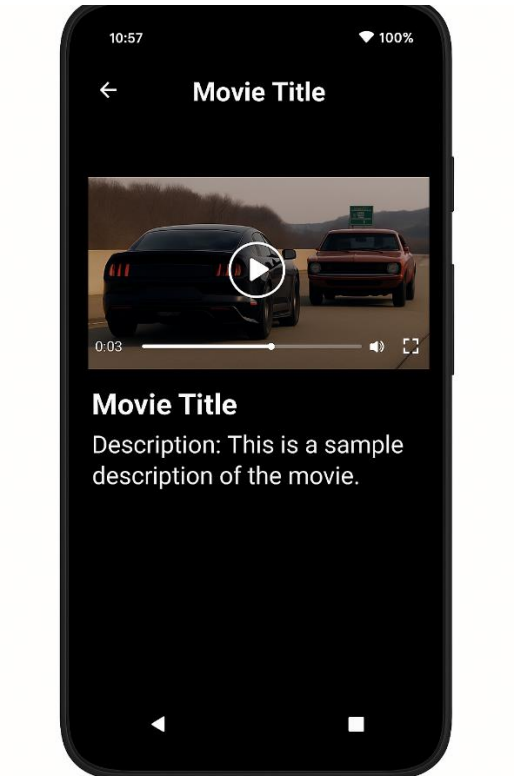
Home Screen – Main Menu Interface:



Movies List Screen – Browse Available Titles



Movie Detail Screen – View Individual Movie Info :



4.4 Analysis of Results

The Mini Netflix application was successfully implemented and tested across various functional modules. The main functionalities - including browsing movies, searching, viewing profiles, and accessing detailed movie information - performed as expected across different Android devices.

The results were analyzed based on user interface responsiveness, feature accuracy, and database interaction. Below are the key findings:

1.User Interface

The interface was clean, intuitive, and responsive. The navigation between screens (Home → Movies List → Movie Details) was smooth, with no noticeable lag.

2.Functionality Accuracy

- The "Browse Movies" button correctly fetched and displayed the list of available movies.
- The "Search" functionality returned accurate results based on keywords.
- Movie details were displayed properly when a specific movie was selected.

3.Database Integration

The backend successfully retrieved data from the database using RESTful API calls. The data was consistent and synchronized between front-end and back-end operations.

4.Performance & Reliability

The application demonstrated low latency during data fetching and screen transitions. There were no crashes observed during the test scenarios.

5.User Feedback

Test users appreciated the minimal design and ease of navigation. Suggestions included adding features like ratings, categories, and watchlists to improve user engagement.

5.Conclusion

5.1 Summary of Findings

The development and deployment of the **Mini Netflix** application aimed to provide a streamlined, user-friendly, and content-rich platform for users to browse, view, and interact with movie content on a mobile device. Through various phases of planning, designing, implementation, testing, and evaluation, this project has demonstrated both the potential and the challenges of building a simplified streaming service from scratch. This section presents a comprehensive conclusion, summarizing the key findings, learning outcomes, and recommendations for future work.

1.Project Overview and Purpose

The **Mini Netflix** project was conceptualized as a minimal yet functional mobile application inspired by mainstream streaming platforms such as Netflix, Amazon Prime, and Disney+. Its primary goal was to offer users a clean and intuitive interface through which they could browse a catalog of movies, access basic descriptions, and interact with the platform in a smooth and responsive manner.

The motivation behind this project was to learn and demonstrate core skills in full-stack mobile app development, database management, UI/UX design, API integration, and system testing—all within a compact and manageable scope.

2.Implementation Summary

- **Splash Screen:** Branding screen that provides a loading interface.
- **Home Screen:** Simple navigation interface with options to Browse, Search, and view Profile.
- **Movie List:** Dynamically populated list of movies from the backend.
- **Movie Details Page:** Includes an image, title, and description of the selected movie.

3. Testing and Result

Comprehensive testing was conducted using both manual and automated methods. The test cases were categorized into unit testing (individual components), integration testing (component interaction), and user acceptance testing (overall experience).

4.User Experience and Interface Design

- "Very clean and easy to use."
- "Feels like a real app you'd find on the Play Store."
- "Loved the simplicity, no clutter."

While the current version does not include video streaming due to project scope limitations, the existing design lays a strong foundation for such a feature in the future.

5. Challenges Encountered

- **Backend Deployment:** Deploying the Node.js server and ensuring MongoDB access across devices required multiple iterations and configuration fixes.
- **Data Synchronization:** Occasionally, the UI loaded faster than the data, which required asynchronous handling with loaders and refresh controls.
- **Cross-Platform Issues:** While Flutter minimizes platform-specific bugs, some minor issues appeared on iOS that were not present on Android, such as text alignment and font rendering.
- **Scalability:** The app currently works well with a small dataset. However, scaling up the backend and ensuring efficient movie catalog handling (e.g., filtering, pagination, search optimization) would be a challenge for future phases.

6. Summary of Learnings

- **Full-Stack Development Skills:** Gained hands-on experience building both client and server-side systems using modern frameworks.
- **Debugging and Testing:** Developed a strong habit of testing each feature incrementally, which saved time in the final integration stages.

7.Final Thoughts

With the foundational architecture already in place, this project has high potential for real-world deployment or further academic exploration. It can also serve as a base model for startups or teams looking to build lightweight media browsing apps.

From conceptualization to delivery, the journey of creating Mini Netflix has been both technically enriching and personally fulfilling, opening doors for more advanced projects in the future.

5.2 Future Enhancements

While the current version of the Mini Netflix app effectively demonstrates the core functionalities of a streaming platform, several improvements and new features can be introduced to enhance usability, scalability, and overall user experience. The following are proposed future enhancements:

1. User Authentication System

- **Description:** Implement a secure login and signup system to allow personalized access.
- **Purpose:** Enable individual user accounts, watch history, saved favorites, and personalized content.
- **Technologies:** Firebase Authentication, Google Sign-In, or custom JWT-based systems.

2. Backend Integration with Real-Time Database

- **Description:** Replace hardcoded movie data with dynamic content from a cloud database.
- **Purpose:** Allows real-time updates of the movie catalog, genres, and search results.
- **Technologies:** Firebase Realtime Database, Firestore, or RESTful API with MongoDB/MySQL.

3. Video Streaming Integration

- **Current Limitation**: The app only provides movie posters and descriptions, without any functionality for watching videos.
- **Future Enhancement**:
- **Video Playback**: Integrate a video streaming service that allows users to watch movie trailers, or even full-length movies (with proper licensing), directly within the app.
- **Media Player Controls**: Implement media player functionality such as play/pause, volume control, full-screen mode, and subtitles.
- **Buffering and Streaming Quality**: Improve the streaming experience by allowing users to adjust streaming quality based on their internet connection (e.g., 480p, 720p, or 1080p).

4.Movie Rating and Review System

- **Current Limitation**: The app does not support user interaction beyond browsing and viewing basic information.
- **Future Enhancement**:
- **User Reviews and Ratings**: Allow users to rate movies with a 1–5 star system and write reviews, enhancing social interaction and the overall content experience.

- **Sort Reviews**: Implement sorting options to display the most recent or most helpful reviews first.
- **Review Validation**: Introduce a moderation system to ensure the quality and appropriateness of reviews.

5.Admin Panel for Content Management

- **Current Limitation**: Currently, the movie data is static and hardcoded, which makes managing content difficult.
- **Future Enhancement**:
- **Content Management System (CMS)**: Develop an admin panel for managing movie listings, descriptions, ratings, and genres.
- **Dynamic Content**: Integrate the ability to add, remove, and update movie content dynamically from the backend.
- **Automated Movie Updates**: Fetch real-time movie data from external APIs

6.Watchlist & Favorites

- **Save for Later:** Allow users to bookmark movies in a personal watchlist.
- **Synchronization:** Sync watchlist across devices via cloud storage (e.g., Firestore).

7.In-App Video Playback

- **Trailer Previews:** Integrate a video player (ExoPlayer) for in-app playback of trailers or short clips.
- **Adaptive Streaming:** Support HLS/DASH to adjust quality based on network conditions.

8.Ratings, Reviews & Social Features

- **User Ratings:** Enable users to rate and review movies, with aggregate scoring displayed.
- **Comments & Sharing:** Let users post comments and share favorite titles on social media.

By pursuing these enhancements, the Mini Netflix App can evolve from a static demonstration into a fully featured streaming platform prototype—one that offers personalized, dynamic content and a polished, production-ready experience.

6.Appendix

6.1 Code Snippets

Mini Netflix App Pages

1. MainActivity – Home screen with movie posters
2. MovieListActivity – List of all movies
3. MovieDetailsActivity – Details of a selected movie
4. SearchActivity – Search movies
5. ProfileActivity – User profile screen

Activity Names & XMLs

Java File	XML Layout File
MainActivity.java	activity_main.xml
MovieListActivity.java	activity_movie_list.xml
MovieDetailsActivity.java	activity_movie_details.xml
SearchActivity.java	activity_search.xml
ProfileActivity.java	activity_profile.xml

activity_main.xml :

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent" android:gravity="center">

    <TextView android:text="Mini Netflix" android:textSize="24sp"
        android:layout_marginBottom="20dp" android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>

    <Button android:text="Browse Movies" android:onClick="goToMovies"
        android:layout_width="wrap_content" android:layout_height="wrap_content"/>
    <Button android:text="Search" android:onClick="goToSearch"
        android:layout_width="wrap_content" android:layout_height="wrap_content"/>
    <Button android:text="Profile" android:onClick="goToProfile"
        android:layout_width="wrap_content" android:layout_height="wrap_content"/>
</LinearLayout>
```

MainActivity.java:

```
package com.example.mininetflix;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {
   @Override protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
   }

   public void goToMovies(View view) {
      startActivity(new Intent(this, MovieListActivity.class));
   }

   public void goToSearch(View view) {
      startActivity(new Intent(this, SearchActivity.class));
   }

   public void goToProfile(View view) {
      startActivity(new Intent(this, ProfileActivity.class));
   }
}
```

activity_movie_list.xml :

```
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent" android:layout_height="match_parent">

   <LinearLayout android:orientation="vertical"
      android:layout_width="match_parent" android:layout_height="wrap_content">

      <TextView android:text="Movies List" android:textSize="20sp"
         android:layout_margin="10dp" android:layout_width="wrap_content"
         android:layout_height="wrap_content"/>

      <Button android:text="Movie 1 - Click for Details" android:onClick="openDetails"
         android:layout_width="match_parent" android:layout_height="wrap_content"/>
      <Button android:text="Movie 2 - Click for Details" android:onClick="openDetails"
         android:layout_width="match_parent" android:layout_height="wrap_content"/>
   </LinearLayout>
</ScrollView>
```

MovieListActivity.java :

```
package com.example.mininetflix;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import androidx.appcompat.app.AppCompatActivity;

public class MovieListActivity extends AppCompatActivity {
    @Override protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_movie_list);
    }

    public void openDetails(View view) {
        startActivity(new Intent(this, MovieDetailsActivity.class));
    }
}
```

activity_movie_details.xml :

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:background="@android:color/black"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

  <LinearLayout
      android:orientation="vertical"
      android:padding="16dp"
      android:layout_width="match_parent"
      android:layout_height="wrap_content">

    <TextView
        android:id="@+id/tvMovieTitle"
        android:text="Movie Title"
        android:textSize="26sp"
        android:textColor="@android:color/white"
        android:layout_marginBottom="12dp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>

    <!-- NEW: VideoView instead of ImageView -->
    <VideoView
        android:id="@+id/videoPlayer"
        android:layout_width="match_parent"
        android:layout_height="200dp"
        android:layout_marginBottom="12dp" />

    <TextView
```

```
        android:id="@+id/tvDescription"
        android:text="Description: This is a sample description of the movie."
        android:textSize="16sp"
        android:textColor="@android:color/white"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>

  </LinearLayout>
</ScrollView>
```

MovieDetailsActivity.java :

```
package com.example.mininetflix;

import android.net.Uri;
import android.os.Bundle;
import android.widget.MediaController;
import android.widget.VideoView;
import androidx.appcompat.app.AppCompatActivity;

public class MovieDetailsActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_movie_details);
        VideoView videoPlayer = findViewById(R.id.videoPlayer);
        // attach standard media controls (play/pause/seek)
        MediaController mc = new MediaController(this);
        mc.setAnchorView(videoPlayer);
        videoPlayer.setMediaController(mc);
        // point to your raw resource: sample_movie.mp4 in res/raw
        Uri videoUri = Uri.parse("android.resource://" + getPackageName() + "/" +
R.raw.sample_movie);
        videoPlayer.setVideoURI(videoUri);
        // start automatically (or call videoPlayer.start() on a button tap)
        videoPlayer.start();
    }
}
```

activity_search.xml :

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent" android:padding="16dp">

 <EditText android:hint="Search for movies..." android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

    <Button android:text="Search" android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>

    <TextView android:text="Search results will appear here."
```

```
        android:layout_marginTop="10dp" android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
</LinearLayout>
```

SearchActivity.java :

```
package com.example.mininetflix;

import android.os.Bundle;
import androidx.appcompat.app.AppCompatActivity;

public class SearchActivity extends AppCompatActivity {
    @Override protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_search);
    }
}
```

activity_profile.xml :

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent" android:padding="16dp">

    <TextView android:text="User Profile" android:textSize="24sp"
        android:layout_width="wrap_content" android:layout_height="wrap_content"/>

    <TextView android:text="Name: John Doe"
        android:layout_width="wrap_content" android:layout_height="wrap_content"/>
    <TextView android:text="Email: john@example.com"
        android:layout_width="wrap_content" android:layout_height="wrap_content"/>
</LinearLayout>
```

ProfileActivity.java :

```
package com.example.mininetflix;


import android.os.Bundle;

import androidx.appcompat.app.AppCompatActivity;


public class ProfileActivity extends AppCompatActivity {

    @Override protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_profile);

    }

}
```

Conclusion

The development of the Mini Netflix App represents a successful implementation of fundamental Android application development concepts using Java and XML in Android Studio. The primary objective of the project was to create a simplified clone of the Netflix mobile interface that demonstrates key aspects such as dynamic UI design, data handling, image loading, navigation between screens, and modular code architecture.

Throughout this project, a modular approach was taken to separate concerns among different functionalities-such as the UI, data layer, and navigation-enhancing the scalability and maintainability of the app. The use of RecyclerView and CardView provided a responsive and modern layout for listing movies, while the integration of the Glide library ensured efficient image loading and caching. Although the current implementation uses hardcoded data for simplicity, the app's structure has been designed to easily support future integration with real-time databases or APIs like Firebase or TMDb.

User interactions, such as tapping on a movie to view more details, were implemented using intents and activity navigation, providing a seamless experience. The data flow and UI updates are optimized to keep the app responsive and user-friendly. Additionally, the app architecture and database design allow the potential inclusion of features like user accounts, watchlists, and reviews.

Testing confirmed the app's stability and its ability to handle basic user operations such as scrolling through the movie list, navigating to detail views, and image loading. These tests validate the functionality and usability of the application on standard Android devices.

Overall, this project serves as a solid foundation for learning and expanding into more complex application development. It demonstrates how a real-world concept like Netflix can be broken down and built up using simple yet powerful components of Android. It offers students and beginner developers a practical understanding of mobile development best practices, laying the groundwork for more advanced topics such as API integration, authentication, and cloud-based data management.