SEC 160: FULL STACK DEVELOPMENT

STREAMFLIX

Submitted by
M Neeraj Kumar (AP23110011290)
T Shyam Kumar (AP23110011358)
K Anusha Chowdary (AP23110011363)
G Teja Reddy (AP23110011378)

SEM: V

SECTION: H

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

OF

SCHOOL OF ENGINEERING AND SCIENCES

SRM University-AP
Neerukonda, Andhra Pradesh 522240
December 2025

INTRODUCTION

1.Project Overview

Streamflix is a modern, feature-rich movie streaming web application designed to deliver a seamless and engaging entertainment experience. The platform focuses primarily on Telugu cinema while also supporting global movie exploration through API integration.

Built as a **Single Page Application (SPA)**, Streamflix utilizes **React.js** to offer a highly responsive interface with smooth navigation and real-time interaction. Unlike traditional static websites, the application provides dynamic updates without page reloads, closely resembling the performance of native apps.

2.Core Objectives

The Streamflix project aims to bridge the gap between local data management and real-time data fetching from external sources. The key objectives include:

->Explore: Provide users access to a curated movie collection stored in a local database.

->**Search:** Integrate IMDb API functionality to enable global movie searches.

->Personalize: Allow users to maintain a watchlist, track viewing history, and receive recommendations.

->Experience: Deliver a visually appealing, modern interface using a premium dark theme built with Tailwind CSS.

These features collectively create a hybrid ecosystem combining local storage efficiency with external data flexibility.

3.Technological Foundation

Streamflix is powered by a modern and scalable technology stack tailored for performance and modularity:

Frontend:*React 18* with *Vite*, offering fast builds and an optimized development environment.

Routing:*React Router DOM (v6)* to enable smooth, client-side navigation without page reloads.

Styling:*Tailwind CSS*, enabling responsive and customizable UI components through utility-first styling.

Data Management:*Context API* for managing global state, covering Authentication, Theme Mode, and Viewing History.

Axios for efficient handling of HTTP requests and API communication.

Backend Simulation:*JSON-Server* to simulate a fully functional REST API supporting all CRUD operations, enabling realistic backend testing and development.

SCENARIO-BASED INTRODUCTION

In the digital age, viewers expect quick access to entertainment without navigating multiple platforms. Consider a typical user who loves watching Telugu movies after a long day of work or study. They open their browser, but they face challenges like:

Scattered websites with incomplete movie details

Slow-loading pages that interrupt the viewing experience

No centralized place to save favorites or track what they watched

Difficulty discovering new content based on personal taste

This gap reveals a clear need for a **fast, intuitive, and personalized movie browsing platform**.

Streamflix was conceptualized to address exactly this problem.

Imagine a user opening Streamflix:

The homepage instantly loads trending Telugu films from a local curated database.

If the user wants to explore movies beyond the local collection, the built-in IMDb API search provides global access.

With one click, the user can add movies to a personal watchlist for later viewing.

The app records viewing history, helping the user pick up right where they left off.

The dark, modern UI creates a theater-like experience, making browsing enjoyable even at night.

This scenario reflects how Streamflix transforms traditional movie exploration into a smooth, interactive, and personalized journey.

Streamflix is a feature-rich movie streaming web application designed to offer a unified entertainment experience. Focused on Telugu cinema while supporting global search, the platform combines local data storage with real-time API integration.

Built as a **Single Page Application (SPA)** using **React.js**, it ensures fluid navigation, instant updates, and a native-app-like feel.

The Streamflix project aims to deliver:

Exploration: Access to locally stored curated movie data

Search: Integrated IMDb API for global movie discovery

Personalization: Watchlist management, viewing history tracking, and tailored recommendations

Experience: A sleek, dark-themed interface powered by Tailwind CSS.

Streamflix uses a robust and modern stack:

Frontend: React 18 + Vite

Routing: React Router DOM (v6)

Styling: Tailwind CSS

Data Management: Context API + Axios

Backend Simulation: JSON-Server for complete CRUD operations

Target Audience

Who Is It For:

Streamflix is designed for movie lovers who want a fast, modern, and personalized platform to explore films—especially Telugu cinema. The application caters to students, young adults, entertainment enthusiasts, and regular OTT users aged **16–40** who enjoy browsing movies, discovering new content, and maintaining their personal watchlists. It also appeals to users who appreciate smooth UI interactions and quick access to curated and global movie data.

Categories Covered

Telugu Movies (Primary Focus)

Global Movies via IMDb API Search

Watchlist Management

Viewing History

Project Goals and Objectives

Main Goal

To create a modern, user-friendly movie streaming interface that combines local movie data with real-time global search, offering users a smooth, interactive, and personalized entertainment experience.

Key Objectives

Provide a curated list of Telugu movies from a local database

Enable global movie search using the **IMDb API**

Allow users to **add and manage watchlists**

Track and display **viewing history**

Offer **personalized recommendations** based on user interaction

Build a visually appealing dark-themed UI using **Tailwind CSS**

Ensure fast performance with **React 18** and **Vite**

Manage global state using **Context API**

Simulate backend operations using **JSON-Server** for full CRUD functionality

User-Friendly Interface

Design Concept

Streamflix features a sleek, modern interface built with a premium dark theme that enhances the viewing experience. The design incorporates smooth transitions, soft shadows, and a visually rich layout inspired by cinematic platforms. This approach ensures that users can browse movies comfortably for long periods without visual strain.

Key Interface Features

Switch between different movie browsing layouts for a personalized experience

Add movies to a watchlist and view previously watched titles

Filter and explore movies by categories and search inputs

Enjoy smooth animations and intuitive navigation throughout the application

Accessibility

Fully responsive across devices, including mobiles, tablets, and desktops

Clear, readable visuals designed for ease of use and comfort

Modern Tech Stack

Streamflix is built with **React 18** and **Vite**, ensuring fast development, quick load times, and smooth runtime performance. **React Router DOM (v6)** manages client-side navigation with seamless transitions between pages.

The design is created using **Tailwind CSS**, offering a modern, dark-themed aesthetic with clean layouts. Global states—such as theme, history, and watchlist—are managed efficiently using the **Context API**, while **Axios** handles API communication for fetching global movie data through IMDb.

JSON-Server functions as a mock backend, providing full CRUD operations for watchlist, history, and user interactions. The interface also uses custom components and reusable UI elements to maintain consistency across the application.

Core Features

✓ Movie Search & Discovery

Fetch movies in real time from the IMDb API, enabling users to explore global content alongside the local curated database.

✓ Watchlist Management

Easily save favorite movies with one-click additions and maintain watchlists across sessions.

✓ Viewing History

Automatically track recently watched movies and provide quick access to past interactions.

✓ Personalized Dashboard

Display recommendations, watchlists, and history with a tailored user experience based on activity.

✓ Layout and Navigation

A responsive navbar, clean routing system, and intuitive page structure allow users to move effortlessly throughout the app.

✓ User Profile Interactions (if included in your app)

View basic user stats, manage preferences, and customize theme settings.

✓ Modern UI/UX Design

A dark-themed interface with smooth animations, responsive design, and aesthetically pleasing visual elements.

PREREQUISITES AND SETUP

System Requirements

To run the Streamflix application smoothly, the following system specifications are recommended:

Node.js: Version 16 or higher (recommended 18 or 20)

npm: Included with Node.js for managing project dependencies

Web Browser: Latest version of Chrome, Firefox, Edge, or Safari

Internet Connection: Required for installing packages and fetching movie data from the IMDb API

External Services & API Keys

IMDb/OMDb API Key (if used): Required for accessing global movie search results. Add the key in your environment file (.env) or directly inside API request files.

Backend Simulation

Streamflix uses **JSON-Server** to simulate backend functionality such as watchlist, history, and CRUD operations.

This server handles user interactions and stores data locally during development.

Development Tools

Code Editors: Visual Studio Code, WebStorm, or any preferred IDE

Git: For version control and project management

Browser DevTools: Useful for debugging UI, network calls, and component states

Installation Steps

1. **Clone or download** the project repository.
2. Navigate to the project directory:
3. cd Streamflix-main
4. Install all dependencies:
5. npm install
6. Start JSON-Server in a separate terminal:
7. npx json-server --watch data/db.json --port 3000
8. Add your API key to the .env file if external movie search is enabled.
9. Run the development server:
10. npm run dev

11. Open the app in your browser at:

    http://localhost:3000

Important Notes

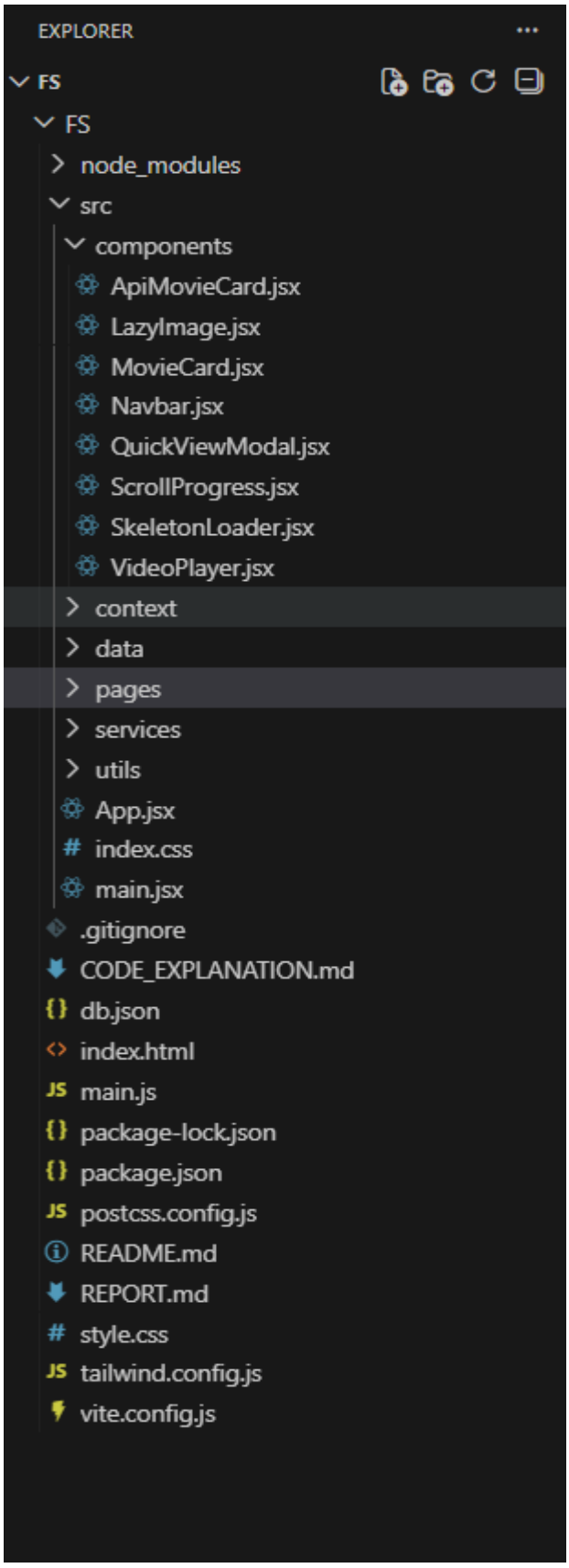
JSON-Server must run on **port 3000** for storing watchlist and history data.

API key is required if you are using IMDb/OMDb search functionality.

Ensure ports **3000** and **3010** are not in use by other applications.

All dependencies are listed in **package.json** and are installed automatically during npm install.

Project Structure



Project Flow

Project Source

- **Source Code Link:**

  PROJECT_FILES_GITHUB

  PROJECT_VIDEOS

Milestone 1: Project Setup and Configuration

Focus: Establishing the foundational structure, environment setup, and initial configuration required for the application.

Key Activities

- **Application Initialization:**
  - `src/main.jsx` initializes and renders the application.
  - `src/App.jsx` defines all routing paths and protected routes.
  - `LoginToggle.jsx` manages switching between login and registration components.
- **Authentication Flow:**
  - `Login.jsx` and `Signin.jsx` handle user login and registration.
  - JSON Server (`data/db.json`) stores user credentials.
  - Sessions and authentication states are maintained using **localStorage**.
- **Protected Route Setup:**
  - `ProtectedRoute` inside `App.jsx` ensures only authenticated users can access internal pages.
  - `DashboardLayout.jsx` wraps protected pages and includes the Navbar for consistent layout.
- **Data Configuration:**
  - LocalStorage manages user sessions and login state.
  - JSON Server handles backend-like operations for user data.
  - NewsData.io API is integrated for real-time news fetching.

Milestone 2: Web Development

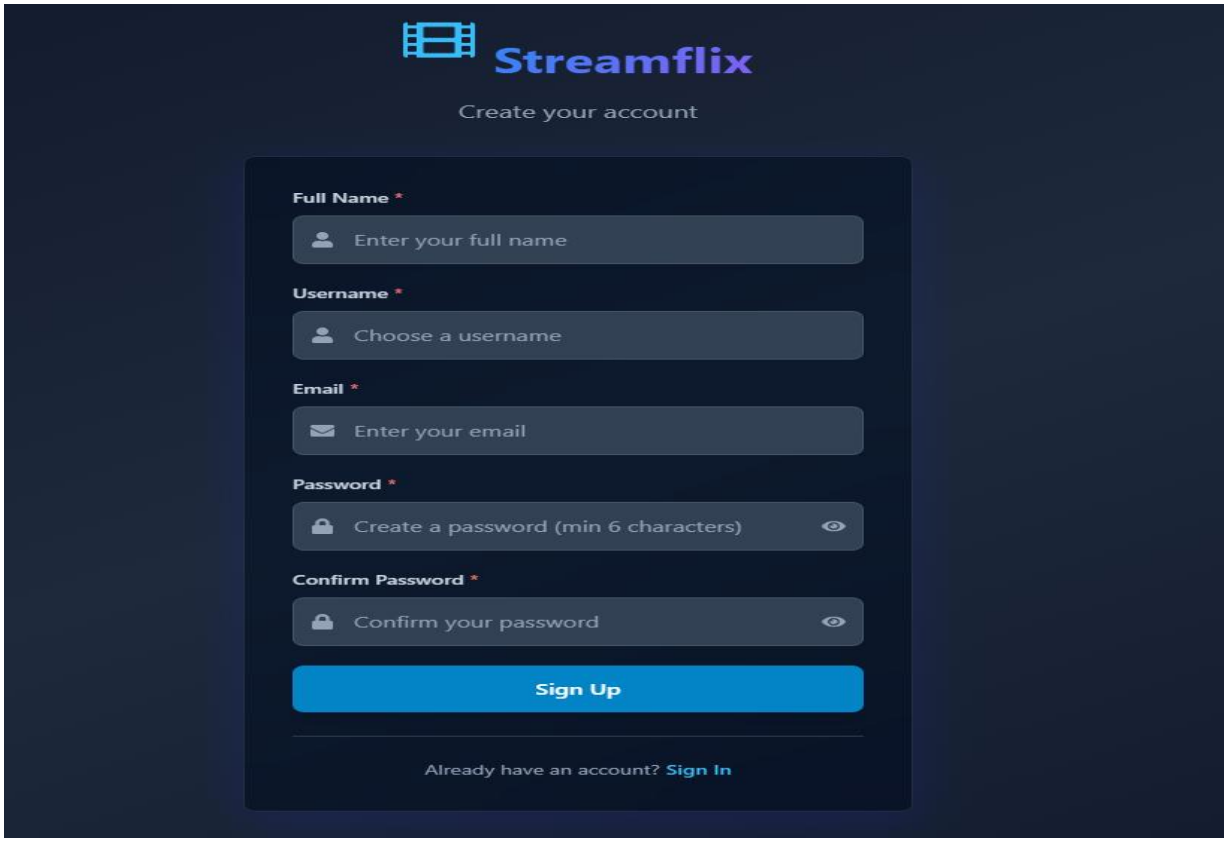
Focus: Developing the core user interface, handling interactions, and establishing smooth data flow across the application.

Key Activities

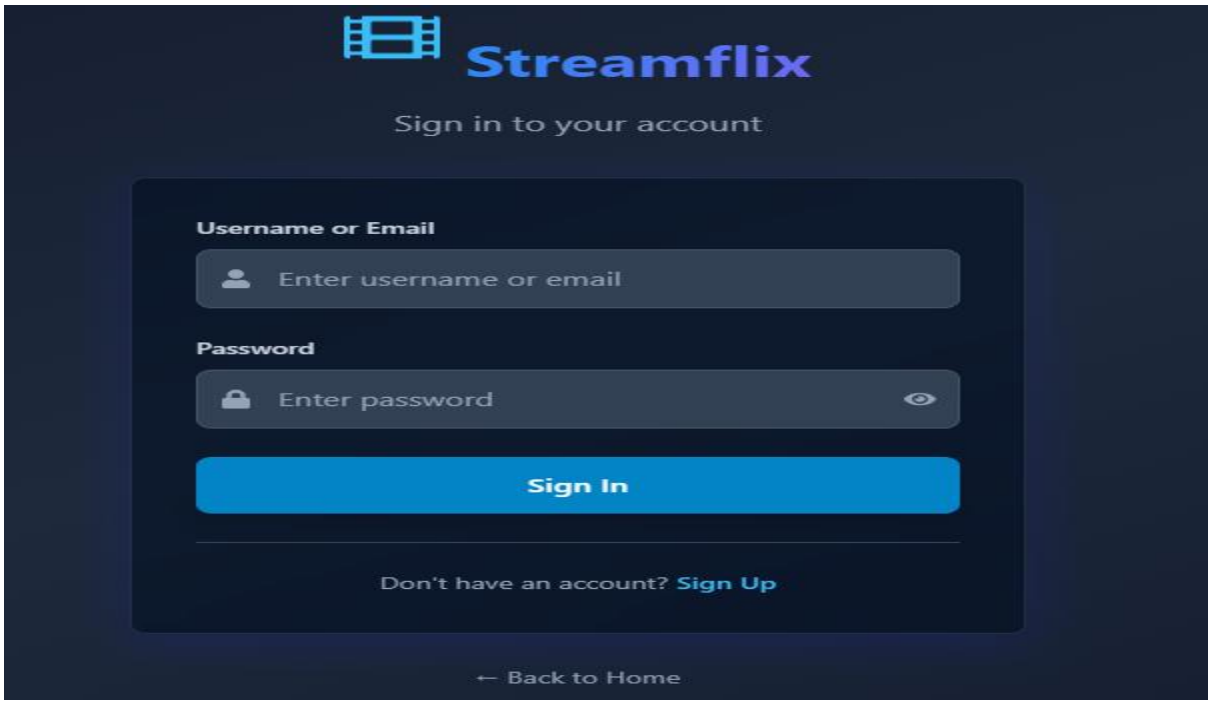
- **Dashboard & Navigation:**
  - `Home.jsx`: Displays category-based news and highlights.
  - `News.jsx`: Shows the latest headlines and filtered news results.
  - `Bookmarks.jsx`: Lists all saved articles.
  - `History.jsx`: Displays the user's reading history.
  - `Profile.jsx`: Shows user information, statistics, and profile actions.
- **User Interactions & State Management:**
  - User actions (bookmarking, viewing, filtering, etc.) are handled within respective pages.
  - State is managed through **useState**, with **Axios** used for API calls.
- **Navigation Components:**
  - `Navbar.jsx` provides top-level navigation across pages and includes logout functionality.
- **Error Handling & Lifecycle Management:**
  - All pages implement error handling for API calls.
  - `useEffect` ensures proper lifecycle management for fetching data and updating UI.
  - The Navbar includes logout handling to safely end user sessions.

Project Output

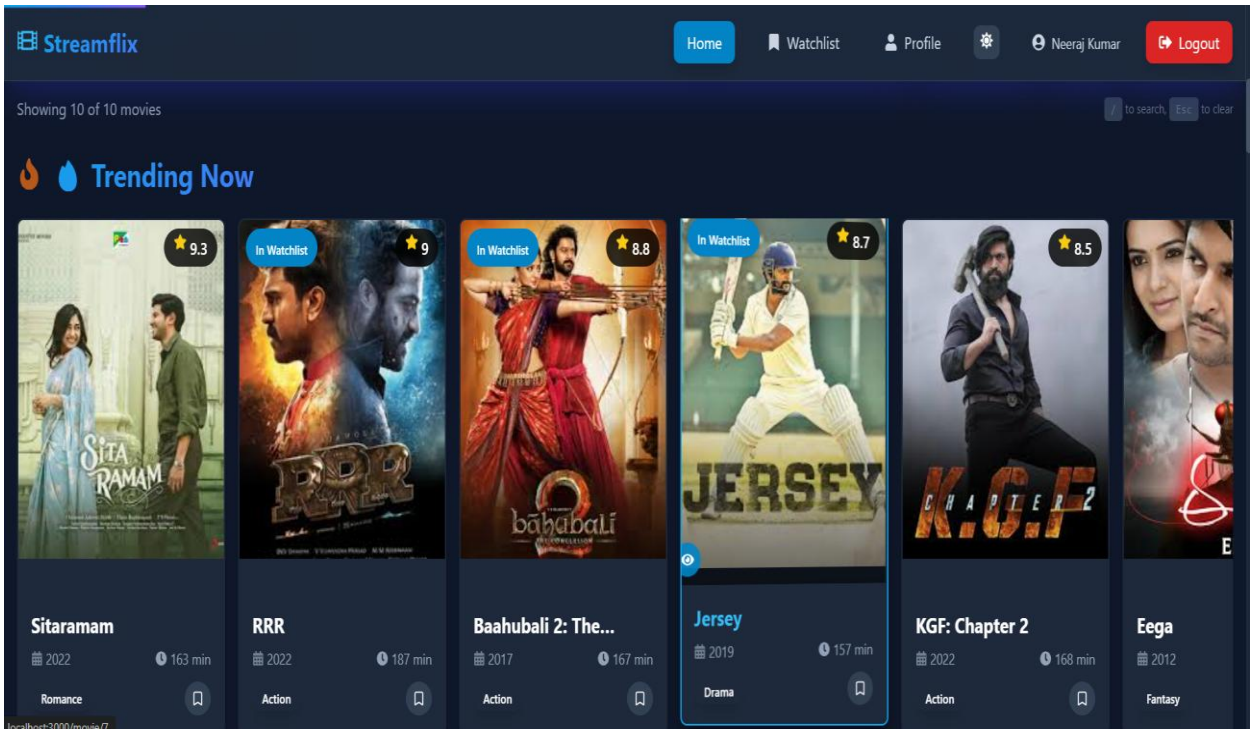
1) Signup Page:



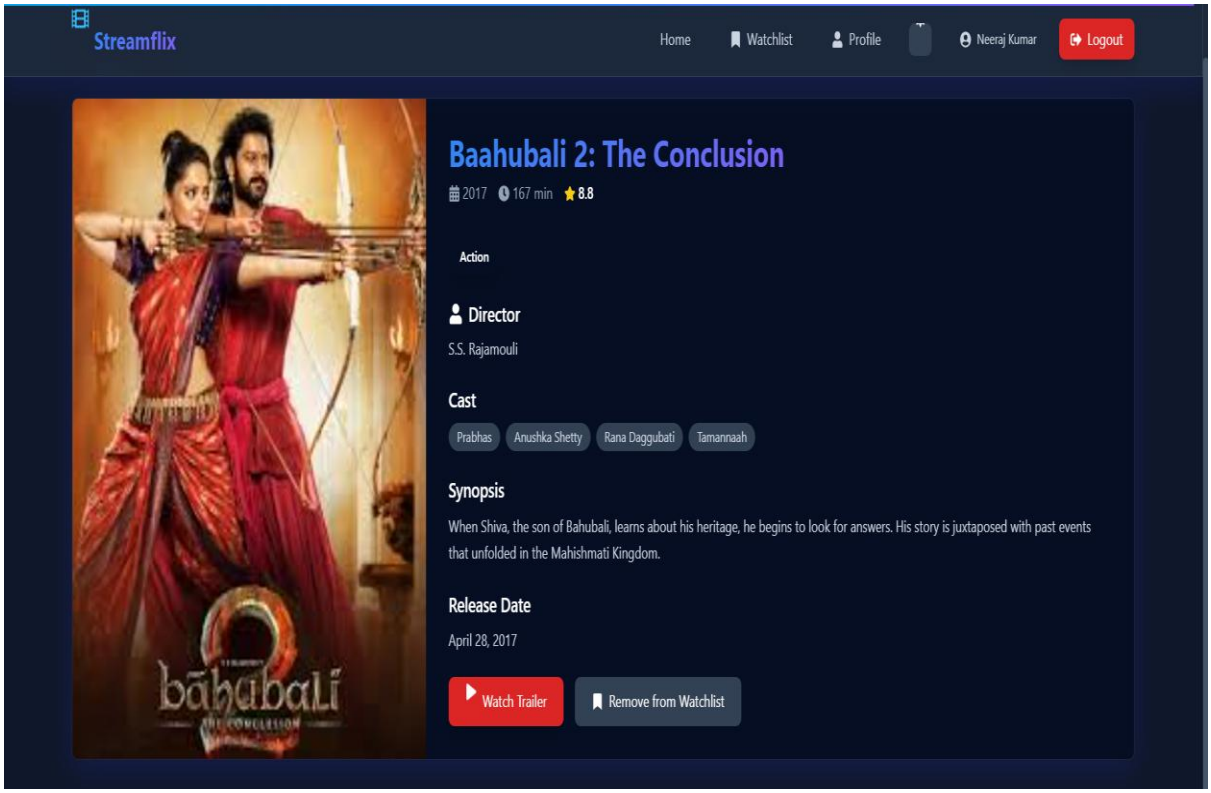
2) Login Page:



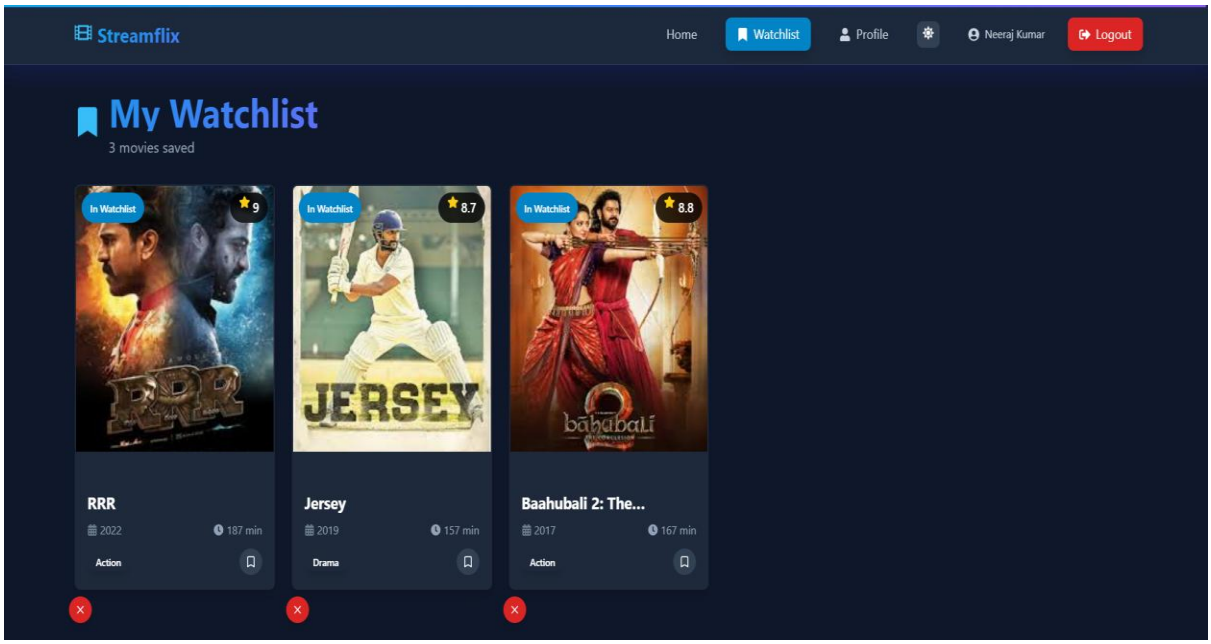
3) Main Home Page:



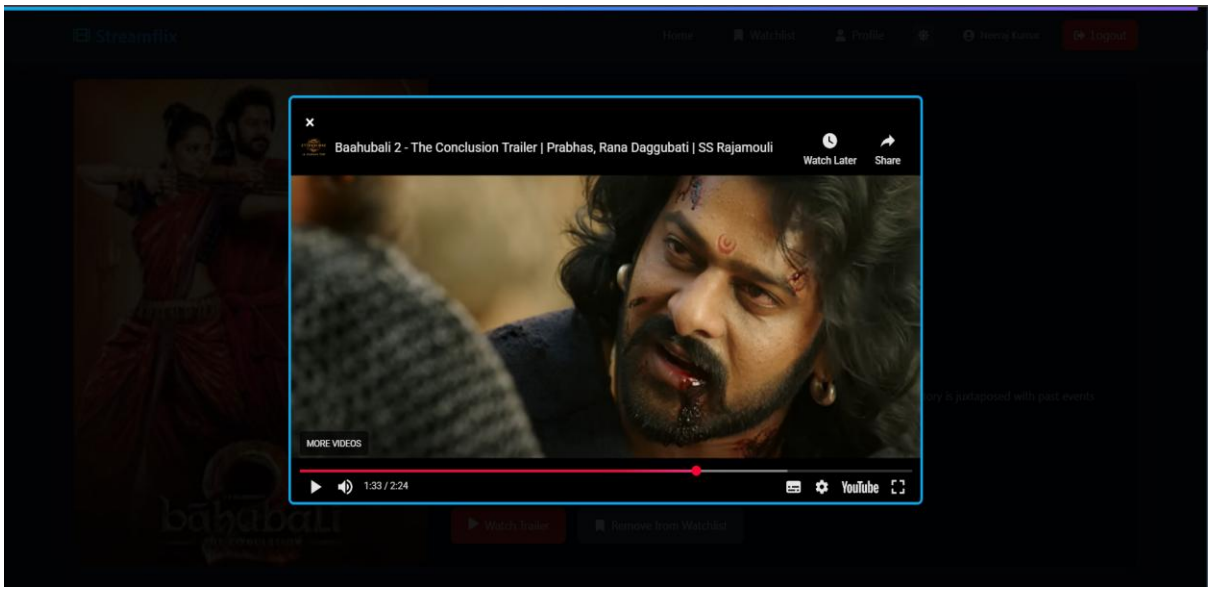
4) Movie card page:



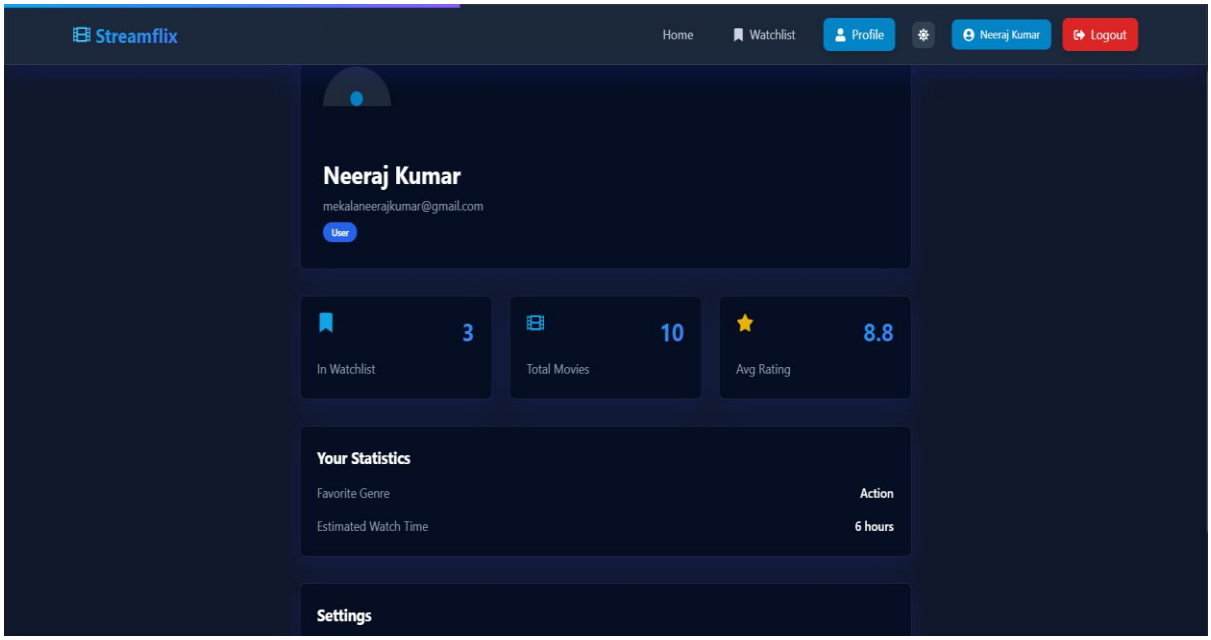
5) Watch list Page:



6) Trailer Page:



7) Profile & settings:



PROJECT FILES LINK

PROJECT_FILES_GITHUB

PROJECT_VIDEOS

THANK YOU!