Project Documentation - Streamify

1. Introduction

• **Project Title:** Streamify - The Ultimate OTT Platform

• **Team Members:**

- Rucha Sudhir Mene - Frontend Developer
- Pranjali Verma - UI/UX Designer & Testing Lead
- Lavanya Kumari - backend and documentation
- Monika Kumari- Research & Integration Support

2. Project Overview

❖ **Purpose:**
Streamify is designed to be an ultimate OTT (Over-The-Top) streaming platform that offers a seamless and immersive entertainment experience. The goal of the project is to provide users with easy access to a wide variety of movies and shows, categorized by genre, language, and popularity—all presented through a clean and responsive user interface.

❖ **Objectives:**

- To build an interactive and visually appealing frontend using React and TypeScript.
- To lay the groundwork for scalable backend integration using Node.js and Express.js.
- To design a user-centric experience that supports intuitive navigation, responsive layouts, and feature-rich components.

❖ **Key Features:**

- Clean and modern UI/UX design
- Browse and explore a curated list of movies and series
- Watchlist functionality (UI ready)
- Authentication interface for login/signup
- Search and basic filter features
- Designed for responsiveness across devices
- Built-in flexibility for future enhancements like streaming, payment integration, and user personalization

3. Architecture

❖ Frontend

The frontend of Streamify is built using **React** and **TypeScript**, following a **component-based architecture** to promote modularity and reusability. Each feature of the application—such as the movie cards, navbar, search bar, watchlist UI, and authentication forms—is implemented as an individual component, which helps in scaling and maintaining the codebase efficiently.

TypeScript adds static typing on top of JavaScript, reducing bugs during development and improving code quality. Styling is handled using CSS modules and responsive design techniques to ensure the platform looks visually consistent across devices. The layout adapts fluidly from desktop to mobile, maintaining an intuitive user experience.

❖ Backend

The backend is designed using **Node.js** and **Express.js**, providing a lightweight and efficient server environment for handling API requests. Express.js allows for quick routing and middleware integration, making it ideal for managing user sessions, authentication, and content delivery.

Although not yet connected, the backend is intended to expose a set of **RESTful APIs** for functionalities such as:

- User registration and login
- Fetching movie/show data
- Managing user watchlists
- Search and filter content

Authentication is planned using **JWT (JSON Web Tokens)** to secure user sessions and protect private routes.

❖ Database

We have chosen **MongoDB** as our NoSQL database to store unstructured and semi-structured data like:

- User details (username, email, encrypted password)
- Watchlist items linked to users
- Content data (movie titles, genres, descriptions, poster URLs, and ratings)

Mongoose, an Object Relational Mapper (ORM) for MongoDB, will be used to define schemas and interact with the database. This ensures data consistency and simplifies CRUD operations for both users and content.

The database schema is designed with scalability in mind, allowing us to easily extend the data model to support new features like user preferences, viewing history, and personalized recommendations.

4. Setup Instructions

❖ Prerequisites

Before setting up the project, ensure you have the following software installed on your system:

- **Node.js** – Required for running the JavaScript runtime and development server.
- **MongoDB** – Used as the database to store user data, watchlists, and content information.
- **npm** (Node Package Manager) or **Yarn** – To install and manage project dependencies.

❖ Installation

Follow these steps to set up and run the **Streamify** project on your local machine:

1. **Clone the repository**
   Open your terminal or command prompt and run:

   bash

   CopyEdit

   ```
   git clone <repository-url>
   ```

2. **Install dependencies**
   Navigate into the project folder. Run the following command in both the client and server directories to install all required dependencies:

   nginx

   CopyEdit

   ```
   npm install
   ```

3. **Set up environment variables**
   Create a .env file in the root of the server directory and add the required environment variables. For example:

   ini

   CopyEdit

   ```
   MONGODB_URI=<your-mongodb-connection-uri>

   JWT_SECRET=<your-secret-key>

   PORT=5000
   ```

4. **Run the application**

To start the frontend:

```
cd client

npm start
```

To start the backend:

```
cd server

npm start
```

The application should now be running locally, with the frontend on http://localhost:3000 and the backend (once connected) on http://localhost:5000.

5. Folder Structure

- ❖ **Client:** The client directory houses the frontend code built with **React** and **TypeScript**. It follows a modular and organized structure to ensure scalability and maintainability:
  - **/components**: Reusable UI components such as movie cards, navbars, search bars, authentication forms, etc.
  - **/pages**: Individual pages like Home, Login, Signup, Watchlist, etc., each handling specific routes and views.
  - **/assets**: Static files including images, icons, and styles.
  - **/utils**: Utility functions and helpers such as data formatters, validation logic, and API service methods.
  - **App.tsx**: Main application component that includes routing logic and renders page components.
  - **index.tsx**: Entry point of the React application which renders the App component into the DOM.
- ❖ **Server:** The server directory contains the backend code written in **Node.js** with **Express.js**. It is structured for clarity and extensibility:
  - **/routes**: Defines API endpoints grouped by functionality, such as user, watchlist, or content.
  - **/controllers**: Contains logic for handling incoming requests and returning appropriate responses.
  - **/models**: Mongoose models representing MongoDB collections like User, Content, and Watchlist.
  - **/middleware**: Includes custom middleware for tasks such as authentication, error handling, and request logging.
  - **server.js / index.js**: Entry point for the backend server where Express is initialized and connected to MongoDB.

6. Running the Application

❖ **Frontend**

Navigate into the client directory and run:

```
npm start
```

This will start the React development server. By default, the frontend will be accessible at:

http://localhost:3000

❖ **Backend**

Navigate into the server directory and run:

```
npm start
```

This will start the Express.js server. The backend will typically be running on:

http://localhost:5000

7. API Documentation

Below are the key RESTful API endpoints exposed by the backend. These are subject to further development and expansion:

❖ **User Routes**

- POST /api/register – Register a new user
- POST /api/login – Authenticate and log in a user

❖ **Movie/Show Data**

- GET /api/movies – Fetch all movies
- GET /api/movies/:id – Fetch details of a specific movie

❖ **Watchlist**

- POST /api/watchlist – Add a movie/show to a user's watchlist
- GET /api/watchlist – Get a user’s watchlist
- DELETE /api/watchlist/:id – Remove a specific item from the watchlist

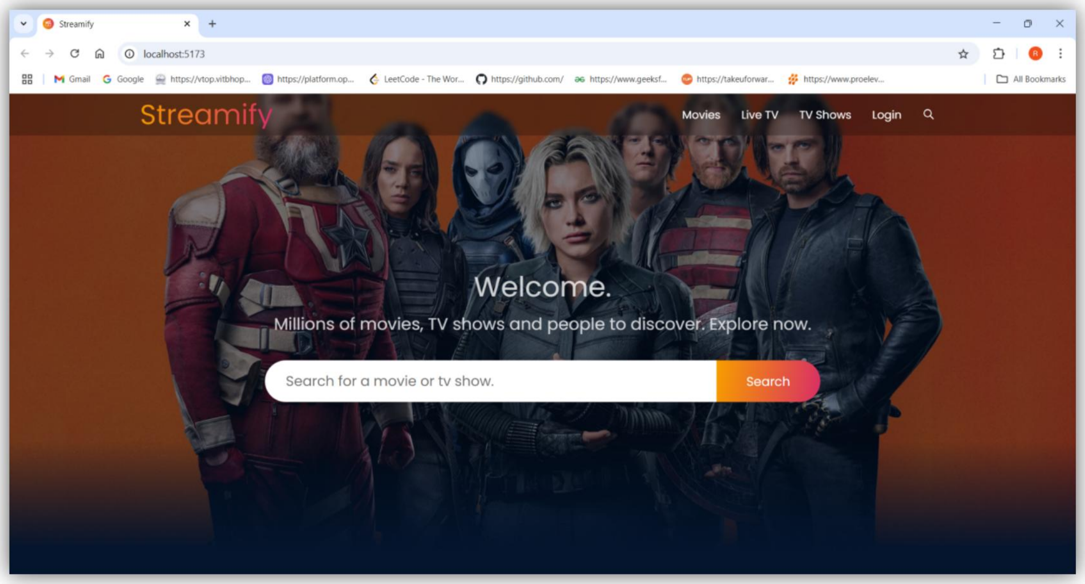
8. Authentication

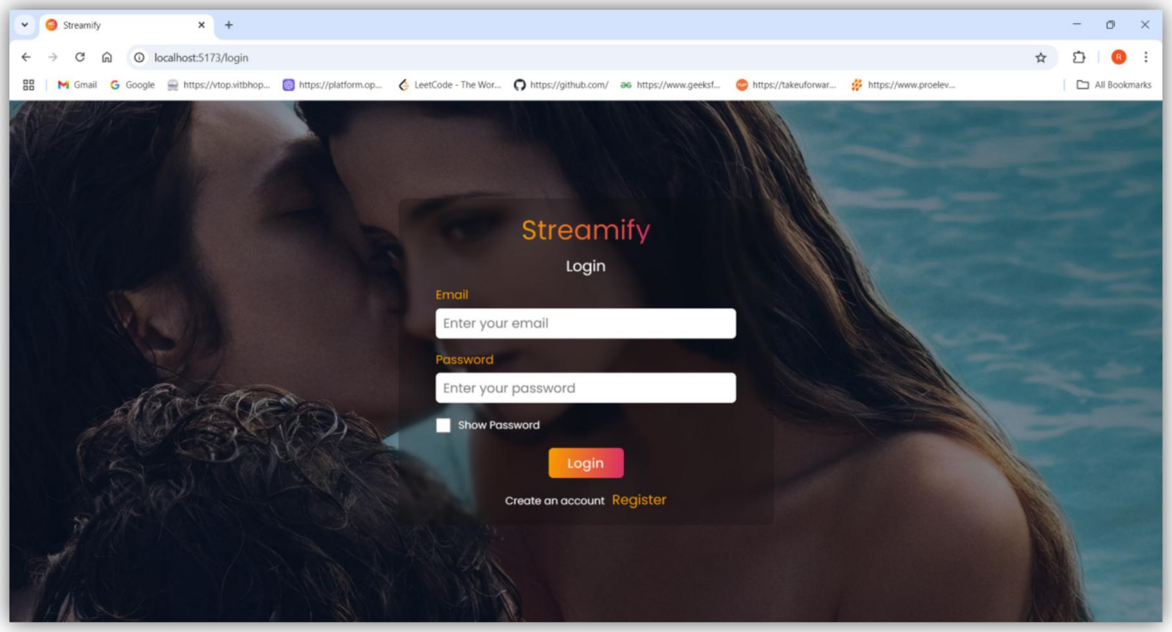
Streamify uses **JWT (JSON Web Tokens)** for secure user authentication and route protection.

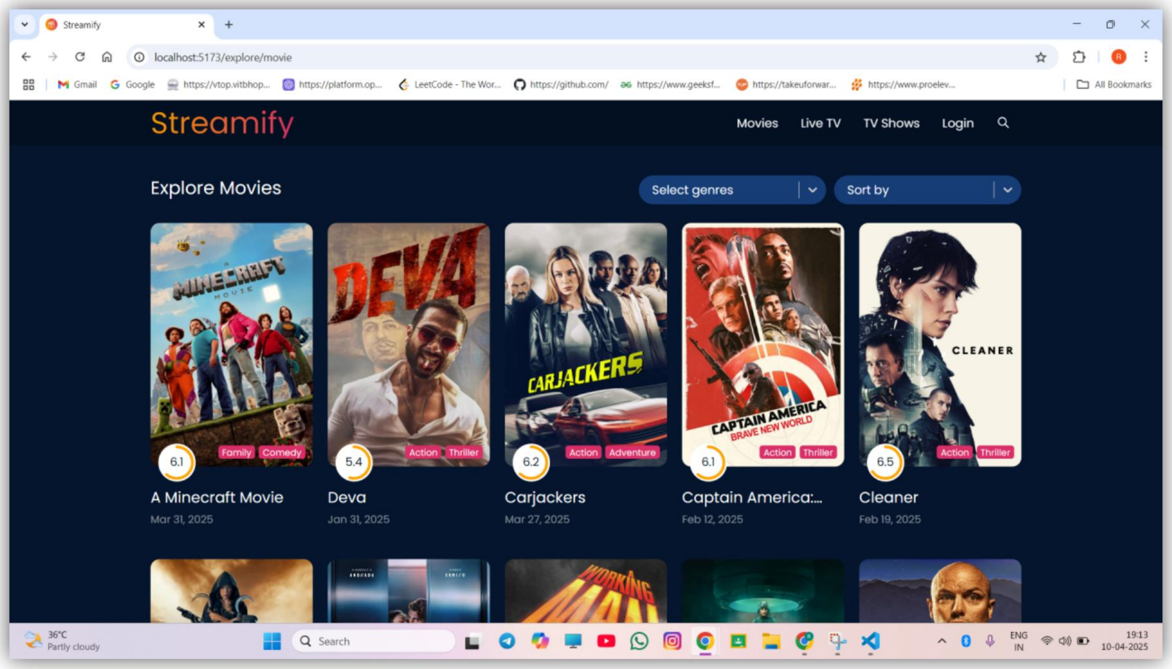
- **Key Features:**
  - On successful login, a token is generated and sent to the client.
  - Tokens are stored in **HTTP-only cookies**, which are not accessible through JavaScript, thereby mitigating XSS attacks.
  - Protected routes on the backend check for the presence and validity of the token before granting access.

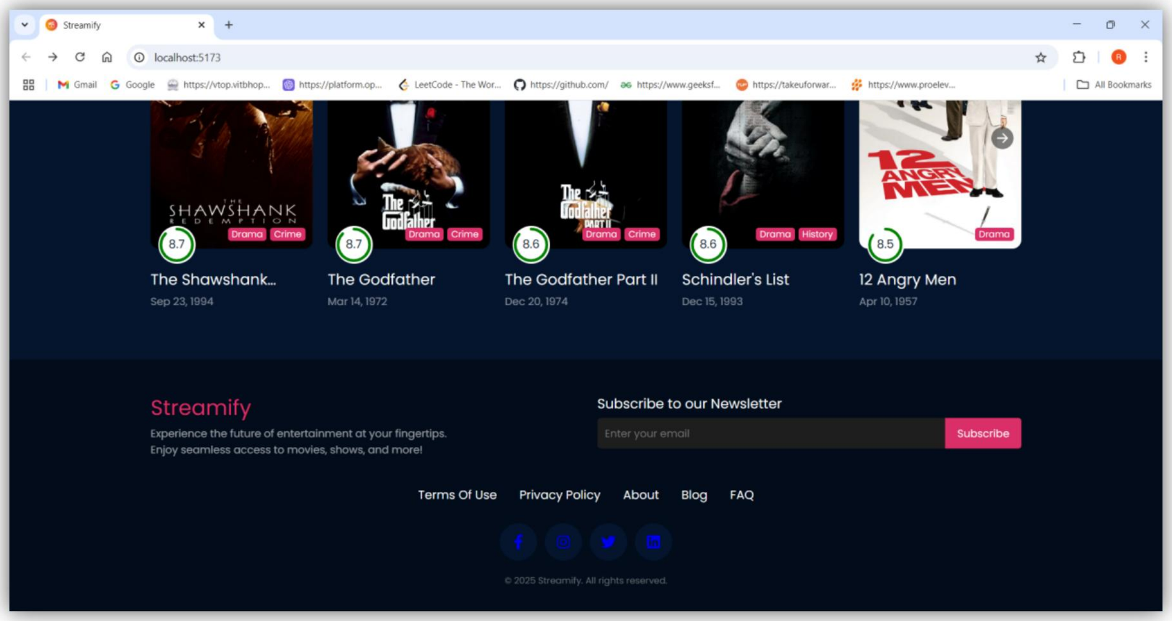
This setup ensures that sensitive operations such as watchlist modifications and personal data retrieval are performed only by authenticated users.

9. User Interface


Streamify
localhost:5173
Streamify
Movies
Live TV
TV Shows
Login
Welcome.
Millions of movies, TV shows and people to discover. Explore now.
Search for a movie or tv show.
Search


Streamify
localhost:5173/login
Streamify
Login
Email
Enter your email
Password
Enter your password
Show Password
Login
Create an account Register


Streamify
localhost:5173/explore/movie
Streamify
Movies
Live TV
TV Shows
Login
Explore Movies
Select genres
Sort by
6.1 Family Comedy
A Minecraft Movie
Mar 31, 2025
5.4 Action Thriller
Deva
Jan 31, 2025
6.2 Action Adventure
Carjackers
Mar 27, 2025
6.1 Action Thriller
Captain America:...
Feb 12, 2025
6.5 Action Thriller
Cleaner
Feb 19, 2025
36°C Partly cloudy
Search
ENG IN
19:13
10-04-2025



Streamify

Experience the future of entertainment at your fingertips.
Enjoy seamless access to movies, shows, and more!

© 2025 Streamify. All rights reserved.

10. Testing

Thorough testing is essential to ensure the reliability, functionality, and stability of the **Streamify** platform. The project uses a combination of automated and manual testing tools to validate both frontend and backend components.

❖ Frontend Testing

- **Tools Used**:
  - Jest: A powerful JavaScript testing framework used for writing and running unit tests.
  - React Testing Library: A library that focuses on testing React components from the user's perspective.
- **Testing Focus**:
  - Component rendering and UI behavior (e.g., MovieCard, Navbar, SearchBar)
  - Event handling (e.g., form submissions, button clicks)
  - Routing and navigation flow
  - Conditional rendering based on authentication state

- **Sample Test Case**:

```
test('renders movie card component', () => {
  render(<MovieCard title="Inception" />);
  expect(screen.getByText(/Inception/i)).toBeInTheDocument();
});
```

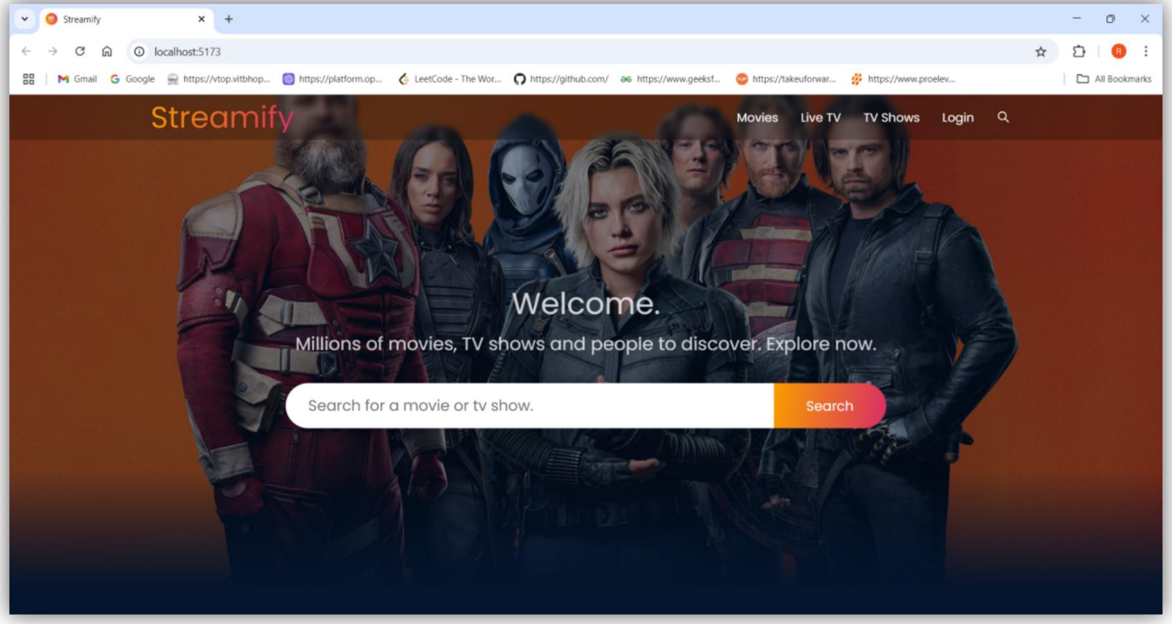
❖ **Backend Testing**

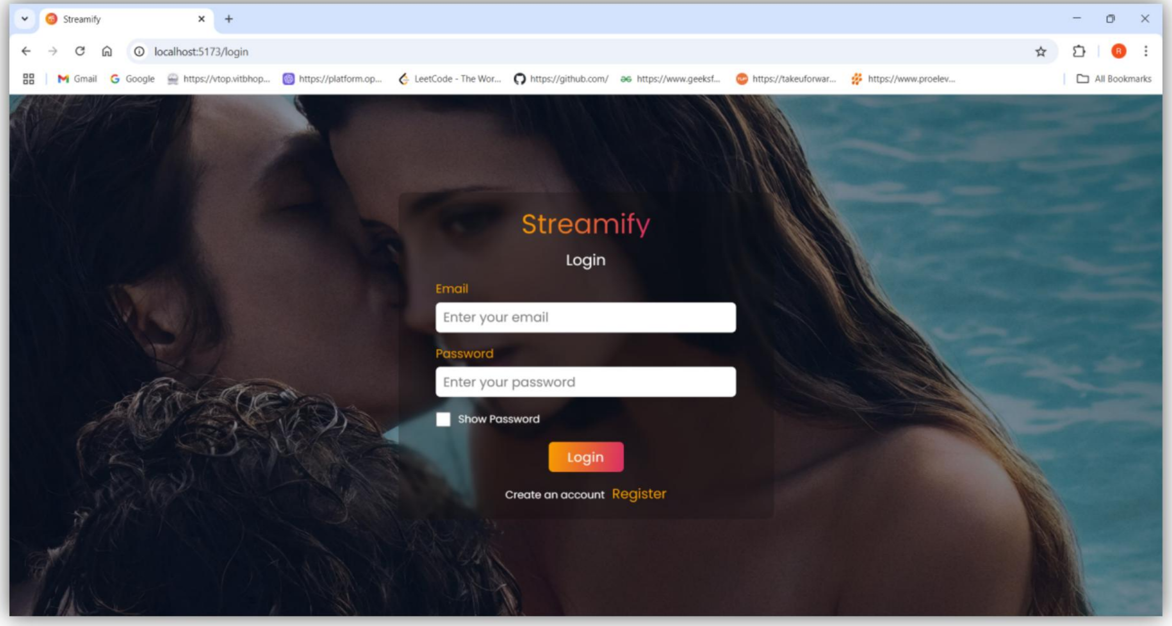
- **Tool Used**:
  - Postman: A manual testing tool for API endpoints.
- **Testing Focus**:
  - API functionality (e.g., login, register, fetch movies)
  - Input validation and error handling
  - Response structure and status codes
  - Authentication protection on private routes
- **Example Test**:
  - Method: POST
  - Endpoint: /api/login
  - Payload:

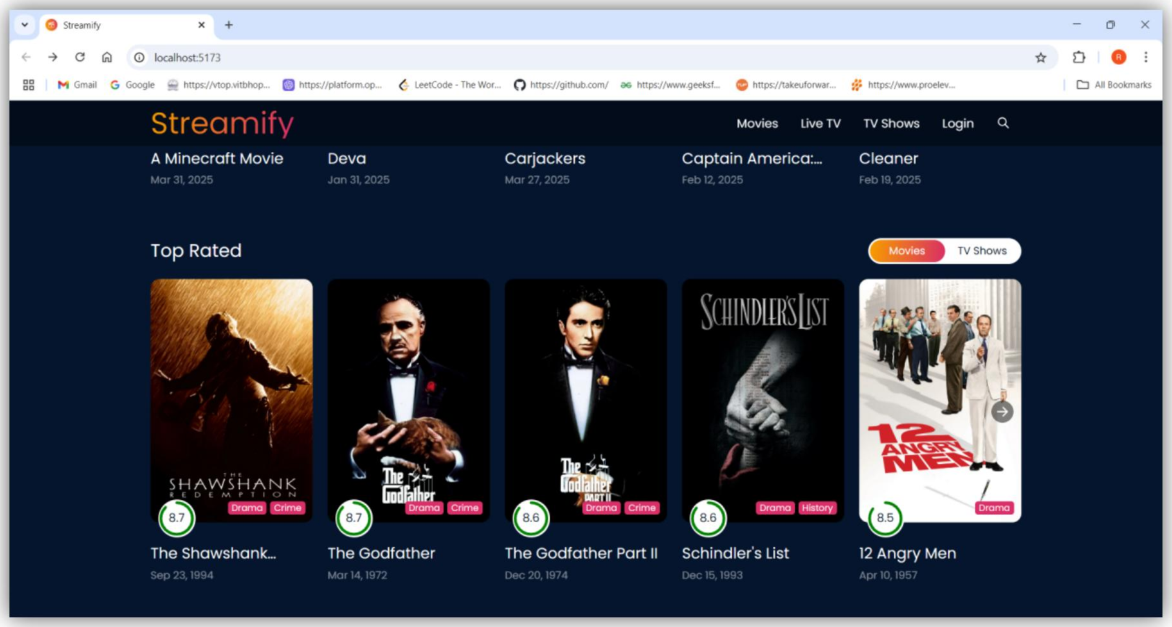
```
{
  "email": "testuser@example.com",
  "password": "password123"
}
```

Postman collections can be saved and shared for easy collaboration during development and debugging.

11. Screenshots or Demo


Streamify
localhost:5173
Streamify
Movies
Live TV
TV Shows
Login
Welcome.
Millions of movies, TV shows and people to discover. Explore now.
Search for a movie or tv show.
Search

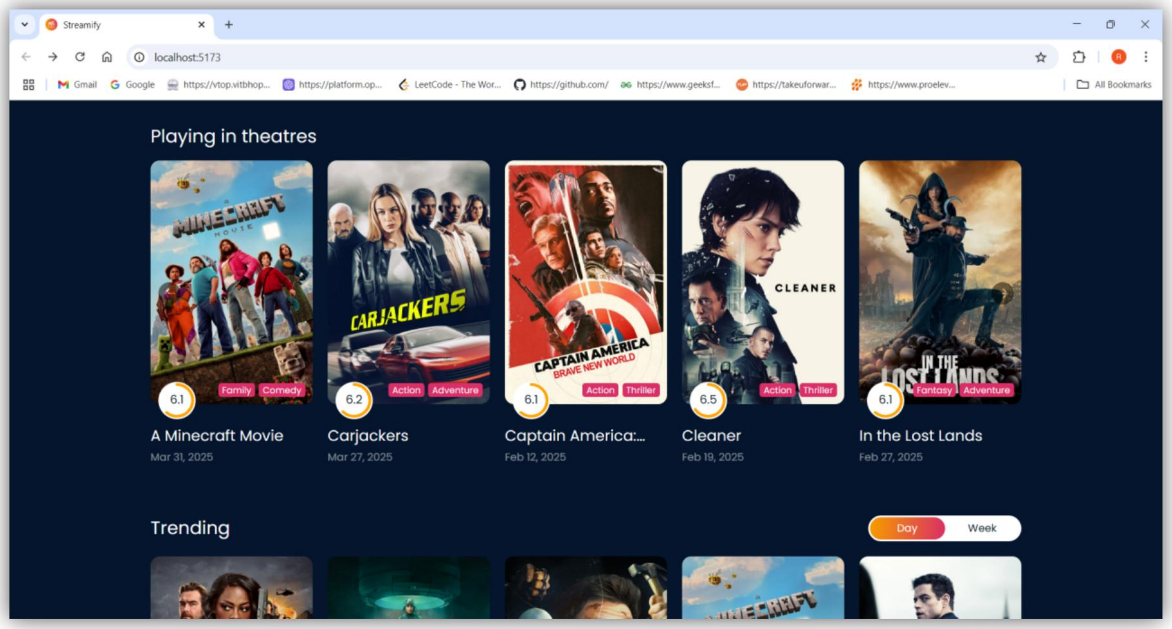




A Minecraft Movie Mar 31, 2025
Deva Jan 31, 2025
Carjackers Mar 27, 2025
Captain America:... Feb 12, 2025
Cleaner Feb 19, 2025

Top Rated

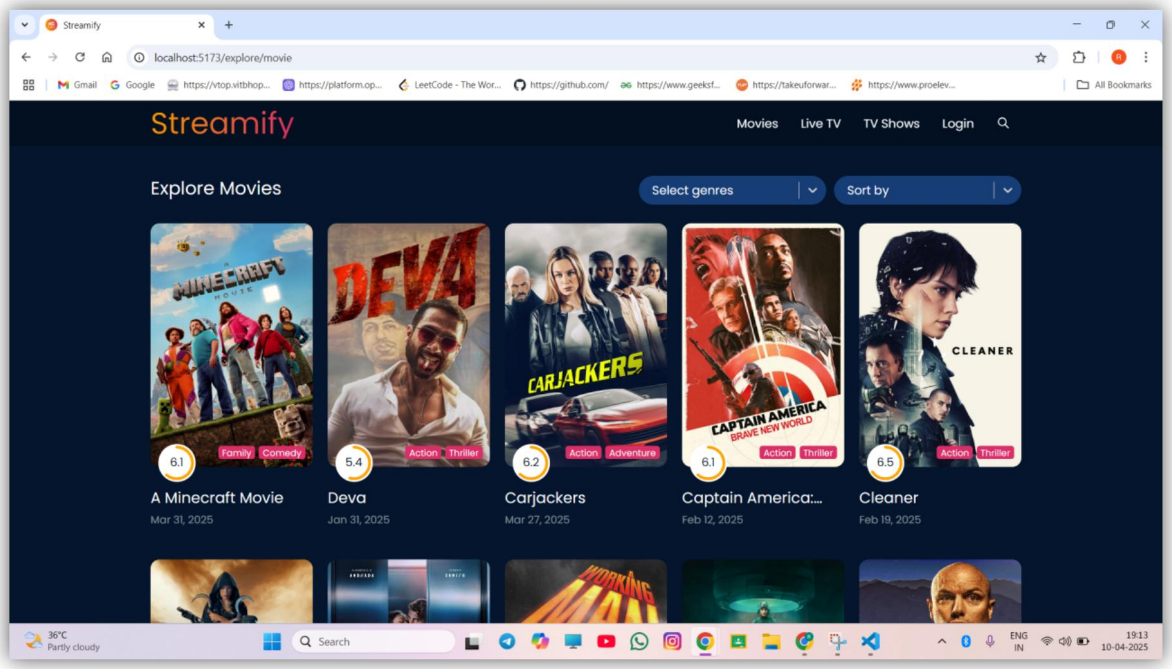
8.7 The Shawshank... Sep 23, 1994
8.7 The Godfather Mar 14, 1972
8.6 The Godfather Part II Dec 20, 1974
8.6 Schindler's List Dec 15, 1993
8.5 12 Angry Men Apr 10, 1957

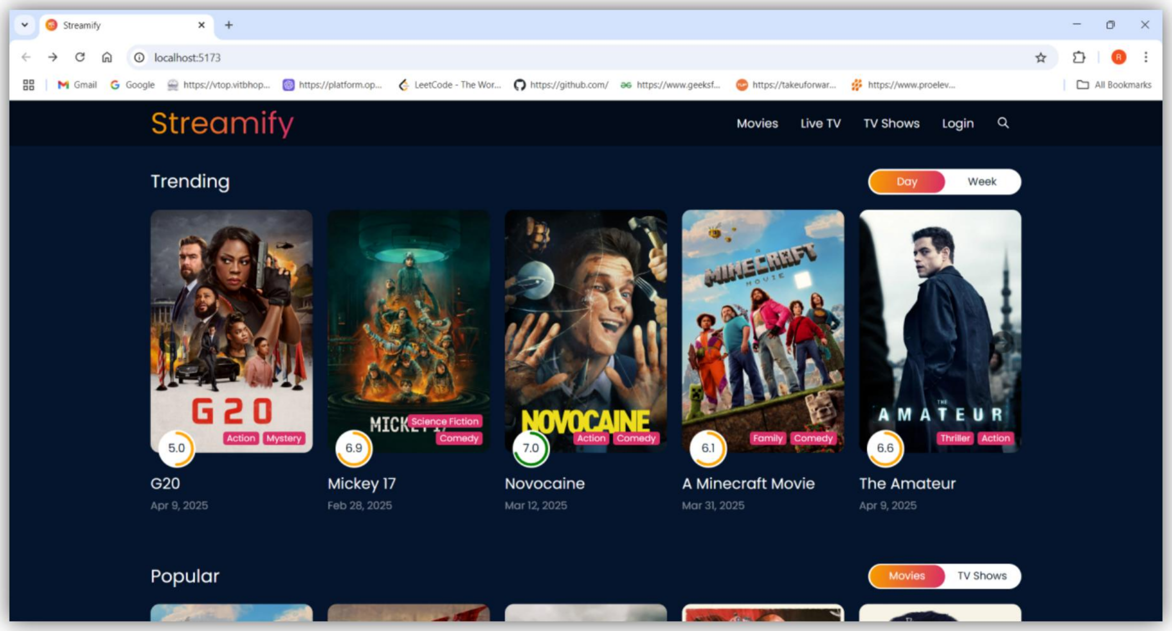


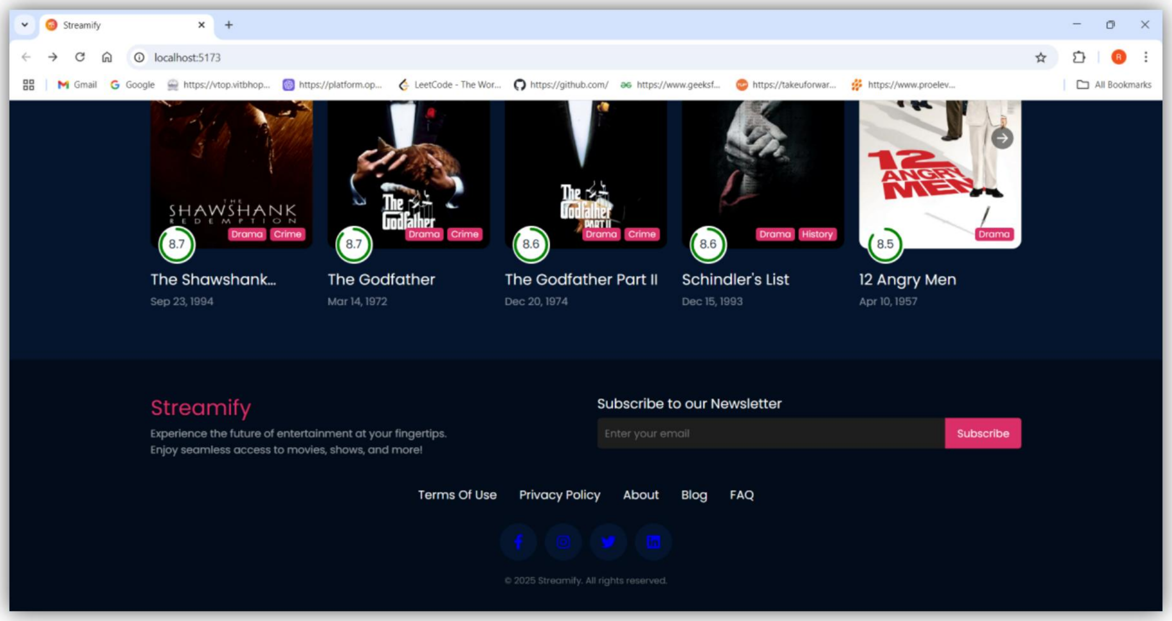
Playing in theatres

6.1 A Minecraft Movie Mar 31, 2025
6.2 Carjackers Mar 27, 2025
6.1 Captain America:... Feb 12, 2025
6.5 Cleaner Feb 19, 2025
6.1 In the Lost Lands Feb 27, 2025

Trending


Streamify
localhost:5173/explore/movie
Movies
Live TV
TV Shows
Login
Explore Movies
Select genres
Sort by
6.1 Family Comedy
A Minecraft Movie
Mar 31, 2025
5.4 Action Thriller
Deva
Jan 31, 2025
6.2 Action Adventure
Carjackers
Mar 27, 2025
6.1 Action Thriller
Captain America:...
Feb 12, 2025
6.5 Action Thriller
Cleaner
Feb 19, 2025


Streamify
localhost:5173
Movies
Live TV
TV Shows
Login
Trending
Day
Week
5.0 Action Mystery
G20
Apr 9, 2025
6.9 Science Fiction Comedy
Mickey 17
Feb 28, 2025
7.0 Action Comedy
Novocaine
Mar 12, 2025
6.1 Family Comedy
A Minecraft Movie
Mar 31, 2025
6.6 Thriller Action
The Amateur
Apr 9, 2025
Popular
Movies
TV Shows



Experience the future of entertainment at your fingertips.
Enjoy seamless access to movies, shows, and more!

© 2025 Streamify. All rights reserved.

12. Known Issues

❖ **Limited Data and Static Content**
The current version of Streamify uses mock or static data for movies, shows, and user profiles. This restricts the ability to test real-world interactions like filtering, recommendations, and user-specific features.

❖ **Authentication Flow Needs Improvements**
Although the UI for login and signup is present, the logic for secure token handling, password hashing, and route protection is still a work in progress. Edge cases like token expiration or login from multiple devices haven't been fully handled yet.

❖ **Search and Filter Functionality is Basic**
The search bar currently performs only basic string matches. It doesn't support advanced filtering (like genre, language, release year, rating) or typo correction, which can affect user experience when browsing for specific content.

❖ **No Error Handling or User Feedback**
The current UI lacks proper error messages or feedback for invalid inputs, failed actions, or loading states. This can confuse users when something doesn't work as expected.

.

13. Future Enhancements

- **Personalized Recommendations**
  We plan to integrate a recommendation system that suggests movies and shows based on a user's watch history, preferences, and ratings. This could use collaborative filtering or machine learning models to enhance user engagement and retention.

- **Offline Streaming Feature**
  Users will be able to download content and watch it later without an internet connection. This will involve secure download options with limited-time access and encrypted file storage to prevent piracy.

- **Multi-language and Subtitle Support**
  Streamify will support content in multiple languages along with subtitle options. This will make the platform more inclusive and accessible to a wider audience.

- **Smart Search and Voice Assistant Integration**
  A smart search feature using natural language processing (NLP) and filters will be implemented. Additionally, voice assistant support (using tools like Web Speech API or third-party integrations) will make navigation more intuitive.

- **Payment Gateway Integration**
  We plan to integrate secure payment gateways such as Razorpay, Stripe, or PayPal for subscription-based access. Users can choose between monthly, quarterly, or annual plans.

- **Parental Controls and Kid Mode**
  A dedicated mode for children with age-appropriate content and restricted access to adult content will be introduced. Parental controls will allow guardians to monitor and restrict access based on age ratings.

- **Live Streaming and Premieres**
  Enabling live broadcasts for events or premieres will add real-time interaction and increase platform engagement. This may involve real-time servers and third-party streaming services.

- **User Reviews and Ratings**
  A review and rating system will be added to allow users to share feedback and rate movies/shows, creating a more community-driven experience.

- **Multi-device Sync and Continue Watching**
  Users will be able to switch between devices and continue watching from where they left off. This will involve session sync across devices using persistent user state management.

- **Admin Dashboard for Content Management**
  A robust admin panel will be developed to manage content uploads, categories, user subscriptions, and view analytics, ensuring efficient content lifecycle management.