STREAMFLIX

A Project Report

Submitted by

S. JASWANTH SAI KRISHNA – AP23110010961

D.CHANDU – AP23110010995

S.RAMU – AP23110010004

A. RAJESH KUMAR – AP23110010996

Under the Supervision of

SYED ARSHAD

Assistant Professor
Department of Computer Science and Engineering
SRM University-AP

In partial fulfilment for the requirements of the project

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SRM UNIVERSITY-AP

NEERUKONDA

MANAGALAGIRI - 522503

ANDHRA PRADESH, INDIA

DECEMBER-2025

TABLE OF CONTENTS

1. Introduction

StreamFlix is a modern, React-based OTT (Online Movie Streaming) web application designed to provide users with a clean, intuitive, and dynamic platform for exploring movies, watching trailers, managing watchlists, and accessing detailed movie information. Built with Vite, Tailwind CSS, and a custom JSON-based backend, StreamFlix combines performance with usability, offering a smooth browsing experience for movie lovers.

In today's entertainment landscape, users depend on streaming platforms for instant access to movie collections. StreamFlix demonstrates how modern UI frameworks like React can be used to build scalable and responsive streaming interfaces. It uses reusable components, context-based state management, and REST-style API communication to create a complete full-stack-like environment suitable for learning and real-world applications.

At the backend, StreamFlix utilizes a **Node.js + Express API** and a JSON database to store movies, users, watchlists, and admin-managed movie entries. The frontend leverages **React Hooks**, **Context API**, **Protected Routes**, and **Tailwind CSS** to deliver a dynamic, mobile-friendly user experience.

StreamFlix is ideal for students, learners, and developers who want hands-on experience building a modern OTT-like platform with authentication, routing, state management, admin panels, and API integration.

2. Scenario-Based Intro

Imagine opening a streaming platform where you can instantly browse trending movies, watch trailers, explore categories, and manage your own watchlist. This is precisely what StreamFlix offers.

Consider yourself a movie enthusiast exploring the latest releases:

You open StreamFlix and land on the **Home Page**, where a Hero Banner showcases featured movies. Scrolling down reveals horizontally scrollable Movie Rows such as *Trending Now*, *Top Rated*, *Action Movies*, and *Comedy*. You click on a movie card and instantly see its detailed description, cast, genre, and a trailer preview. With one click, you add the movie to your Watchlist.

Later, you decide to explore movies manually and use the **Search Bar**, which instantly fetches matching titles using fuzzy search from the database.

If you are an admin, you access the **Movies Admin Panel**, where you can add, edit, or delete movies. You upload posters, enter descriptions, genres, release dates, and trailer links. With just a button click, your new movie appears instantly on the user side.

Throughout your experience, authentication ensures that your personal watchlist and preferences remain secure. The system validates data, manages user sessions through React Context, and ensures smooth functioning without page reloads.

By the end of the day, you've explored movies, managed your watchlist, searched titles, and seen detailed pages — all within StreamFlix.

3. Target Audience

StreamFlix is designed for:

• Movie Enthusiasts

Users who want a clean, simple movie browsing experience.

• Students & Learners

Individuals learning:

- React
- Tailwind
- Context API
- Routing
- REST APIs

• Beginner Developers

People building their first full-stack style project.

• OTT Platform Prototype Designers

Teams prototyping streaming platforms without complex backend infrastructure.

4. Project Goals & Objectives

The main goals of StreamFlix include:

1. Modern UI/UX for Movie Streaming

Provide a responsive, mobile-friendly interface.

2. Efficient Movie Management

Admin can add, edit, delete movies.

3. Personalized Experience

Users can maintain their own watchlist.

4. Smooth Navigation

Use React Router for:

- Home
- Search
- Movie Details
- Watchlist
- Admim Pages

5. Learning Real-World Development Workflow

Implement modern frontend patterns:

- Component reuse
- Props/state
- Context
- API integration
- Protected routes

5. Key Features

1. Browse Movies

Home page displays movie categories in rows.

2. Movie Details Page

Includes:

- Poster
- Description
- Genre
- Cast
- Trailer Modal

3. Watchlist

Users can:

- Add movies
- Remove movies
- View personal list

4. Search

Real-time search using local JSON + filtering.

5. Admin Module

Add, edit, delete movies from a dedicated dashboard.

6. Authentication

Login + Register pages using Context API.

7. Protected Route

Only logged-in users can access:

- Watchlist
- Admin panels

8. Trailer Modal

Embedded YouTube video popup

6. Prerequisites

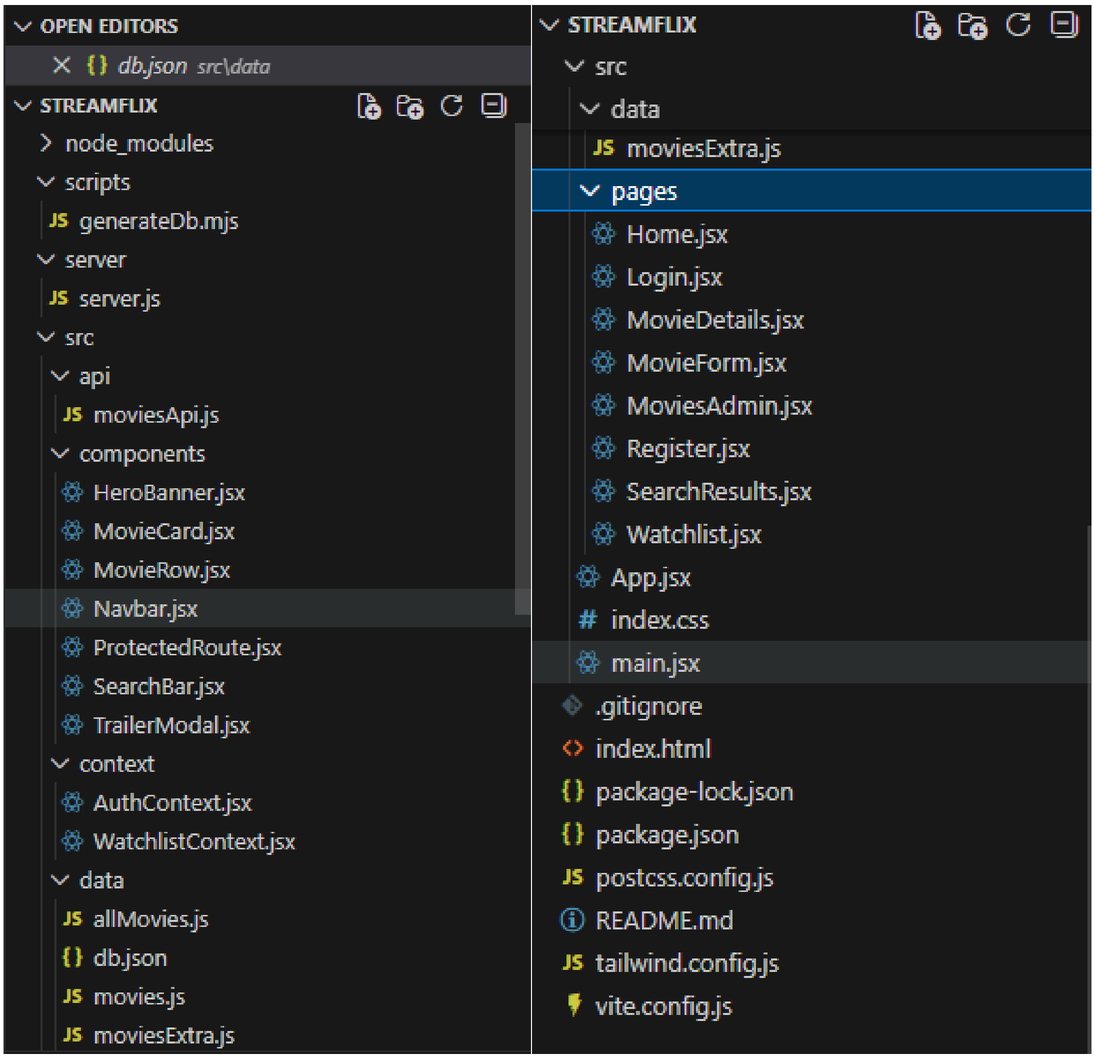
Tools & Technologies Required

- Node.js and npm
- Vite + React
- Tailwind CSS
- Express.js backend
- JSON database (db.json)
- VS Code
- Git

Users must know:

- HTML, CSS, JS basics
- ES6+
- React Hooks
- Routing

7. Project Structure



```
OPEN EDITORS
  × {} db.json src\data
STREAMFLIX
  > node_modules
  v scripts
      generateDb.mjs
  v server
      server.js
  v src
    v api
        moviesApi.js
    v components
        HeroBanner.jsx
        MovieCard.jsx
        MovieRow.jsx
        Navbar.jsx
        ProtectedRoute.jsx
        SearchBar.jsx
        TrailerModal.jsx
    v context
        AuthContext.jsx
        WatchlistContext.jsx
    v data
        allMovies.js
        db.json
        movies.js
        moviesExtra.js
```

```
STREAMFLIX
  v src
    v data
        moviesExtra.js
    v pages
        Home.jsx
        Login.jsx
        MovieDetails.jsx
        MovieForm.jsx
        MoviesAdmin.jsx
        Register.jsx
        SearchResults.jsx
        Watchlist.jsx
      App.jsx
      index.css
      main.jsx
    .gitignore
    index.html
    package-lock.json
    package.json
    postcss.config.js
    README.md
    tailwind.config.js
    vite.config.js
```

8. PROJECT FLOW

Before deep technical implementation begins, it is important to understand how StreamFlix operates from a user perspective and how the different modules interact internally. let's see the demo.

Demo link: https://drive.google.com/file/d/1EsWGtl4HhmtbnmQP4hCay6rNg0vQjeNW/view?usp=sharing

Use the code in:https://drive.google.com/file/d/10_jbdYoam_kJaBVWr5e-suQXWiIJpVpN/view?usp=sharing

User Flow

1. User opens StreamFlix homepage.
2. User browses categories displayed via horizontal movie rows.
3. User clicks on a movie → navigates to MovieDetails page.
4. User can watch trailer (modal), add to watchlist, or explore related titles.
5. If the user is not logged in, certain actions redirect to Login.
6. Logged-in users see Watchlist, Admin pages (if authorized), etc.
7. Admin can add new movies, edit existing movies, or delete them.

System Flow

- Frontend retrieves movie data from backend (moviesApi.js)
- AuthContext manages login & user sessions
- WatchlistContext synchronizes watchlist operations
- ProtectedRoute ensures authenticated pages
- Admin pages use MovieForm.jsx for CRUD actions
- All data stored in db.json through Express server

9.Project Setup and Configuration

1. Install Required Tools

Node.js & npm

StreamFlix requires Node.js for JavaScript runtime and npm to install dependencies.

Download:
https://nodejs.org/en/download/

After installation, verify:

node -v : v22.21.0

npm -v : 10.9.4

2. Create Vite + React Project

```
npm create vite@latest streamflix

cd streamflix

npm install
```

3. Install Core Libraries

StreamFlix uses:

Frontend:

- React
- React Router DOM
- Tailwind CSS
- Context API
- Vite development server

Backend:

- JSON-based database (db.json)
- Custom APIs

Install libraries:

```
npm install react-router-dom
npm install axios
npm install express
npm install cors
npm install nodemon --save-dev
```

4. Tailwind CSS Installation

As per official docs:

```
npm install -D tailwindcss postcss autoprefixer
npx tailwindcss init -p
```

Add Tailwind directives in index.css:

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

5. Setup Backend (server.js)

A custom server handles routes:

```
server/
 server.js
```

6. Setup db.json

A JSON database is created in:

src/data/db.json

Contains:

- movies ,users ,watchlist

10.Web Development

This milestone covers building the frontend UI, components, routing, API, and admin panel.

1. Setup React Application

main.jsx

Mounts the React app:

```
ReactDOM.createRoot(document.getElementById('root')).render(
   <AuthProvider>
     <WatchlistProvider>
       <Router>
         <App />
       </Router>
     </WatchlistProvider>
   </AuthProvider>
```

2. Configure Routing (App.jsx)

Routes:

- / → Home
- /login → Login
- /register → Register
- /movie/:id → MovieDetails
- /search → SearchResults
- /watchlist → Protected Route
- /admin → MoviesAdmin
- /admin/new → MovieForm
- /admin/edit/:id → MovieForm

ProtectedRoute ensures authentication:

```
<Route path="/watchlist" element={
  <ProtectedRoute><Watchlist /></ProtectedRoute>
} />
```

3. Reusable UI Components

Navbar.jsx

Contains:

- Search bar
- Login/Register buttons
- User avatar
- Navigation links

HeroBanner.jsx

Shows featured movie + CTA.

MovieCard.jsx

Displays movie poster & title.

MovieRow.jsx

Horizontal row for categories such as:

- Trending Now
- Top Rated
- Action
- Comedy

TrailerModal.jsx

YouTube video popup.

4. Context API

AuthContext.jsx

Handles:

- Login
- Register

- Logout
- User session

Uses localStorage.

WatchlistContext.jsx

Manages:

- Add to watchlist
- Remove from watchlist
- Load watchlist on login

5. API Layer

api/moviesApi.js

Handles:

- Fetch all movies
- Fetch single movie
- Create movie
- Update movie
- Delete movie

```
export const getMovies = () => axios.get("/movies");
```

Backend URL is configured via proxy in vite.config.js.

6. Admin Development

Admin has full CRUD capabilities using these pages:

MoviesAdmin.jsx

List of all movies with:

- Edit button
- Delete button
- Add new movie link

MovieForm.jsx

The same component is used for:

- Add

- Edit

Fields:

- title
- genre
- poster URL
- trailer URL
- description

11. Code Descriptions

This section matches the reference PDF — describing key code components in detail.

A. Navbar.jsx – Code Description

• Imports React hooks and contexts
• Contains search bar and navigation links
• Uses conditional rendering:

{user ? <Logout/> : <Login/>}

• Fully responsive using Tailwind CSS
• Links to Home, Watchlist, Admin
• Search triggers navigation to /search?q=

B. MovieRow.jsx – Code Description

• Accepts title and movies props
• Renders horizontal scrollable movie cards
• Uses Tailwind:

flex overflow-x-scroll gap-4

• Each card clickable and navigates to MovieDetails

C. MovieDetails.jsx – Code Description

• Fetches movie details using id from params
• Displays:

- Poster
- Title
- Genre
- Description
  • Buttons:
- Watch Trailer

- Add to Watchlist
  • Modal:
- TrailerModal opens on click

Uses useEffect:

```
useEffect(() => {
  fetchMovie();
}, [id]);
```

D. SearchResults.jsx – Code Description

• Gets search query from URL
• Filters movies:

```
movies.filter(movie =>
  movie.title.toLowerCase().includes(query.toLowerCase())
)
```

• Displays results in grid layout

E. Watchlist.jsx – Code Description

• Uses WatchlistContext
• Displays list of saved movies
• Remove option:

```
removeFromWatchlist(movie.id)
```

F. AuthContext.jsx – Code Description

Contains functions:

```
login(email, password)

register(userData)

logout()
```

Stores user in localStorage:

```
localStorage.setItem("streamflix-user", JSON.stringify(user));
```

Provides global access to user data.

G. server.js – Code Description

Express server:

- Imports Express, CORS
- Loads db.json as local database
- Provides REST endpoints:

GET /movies

GET /movies/:id

POST /movies

PATCH /movies/:id

DELETE /movies/:id

- Used by moviesApi.js

12. Project Execution – StreamFlix

Step 1: Install Dependencies

```
npm install
```

Step 2: Start Backend

```
nodemon server/server.js
```

Backend runs on:

http://localhost:5000

Step 3: Start Frontend

```
npm run dev
```

Frontend runs on:

http://localhost:5173

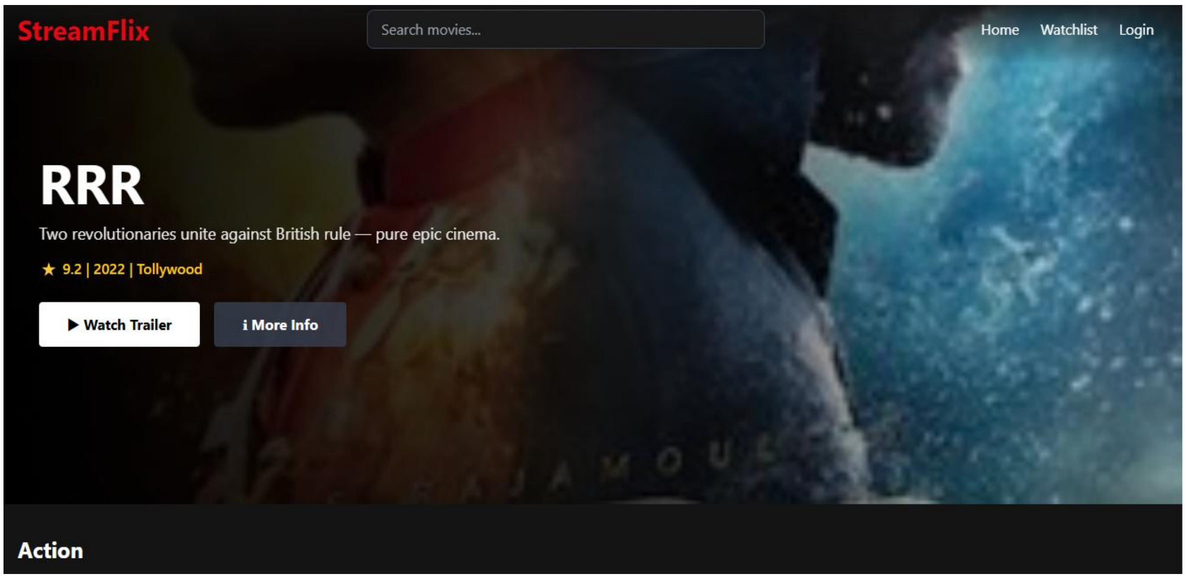
Step 4: Access Application

- Home page
- Login/Register
- Browse movies
- Watchlist
- Search
- Admin

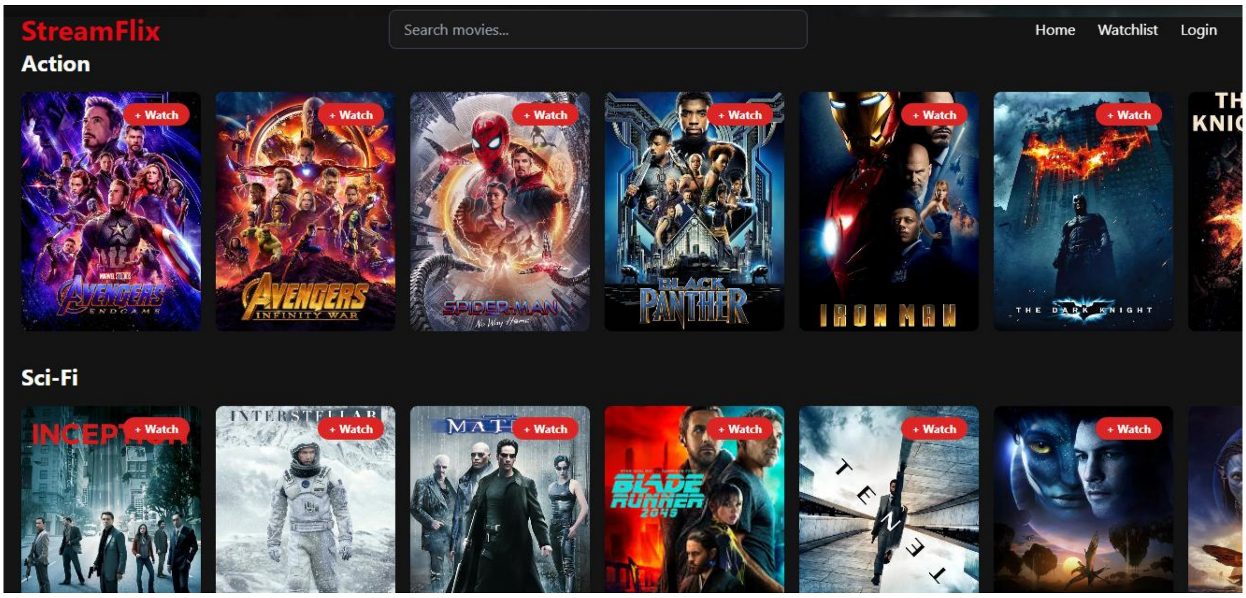
13. OUTPUT (Screens / Features)

Home Page

- Hero Banner

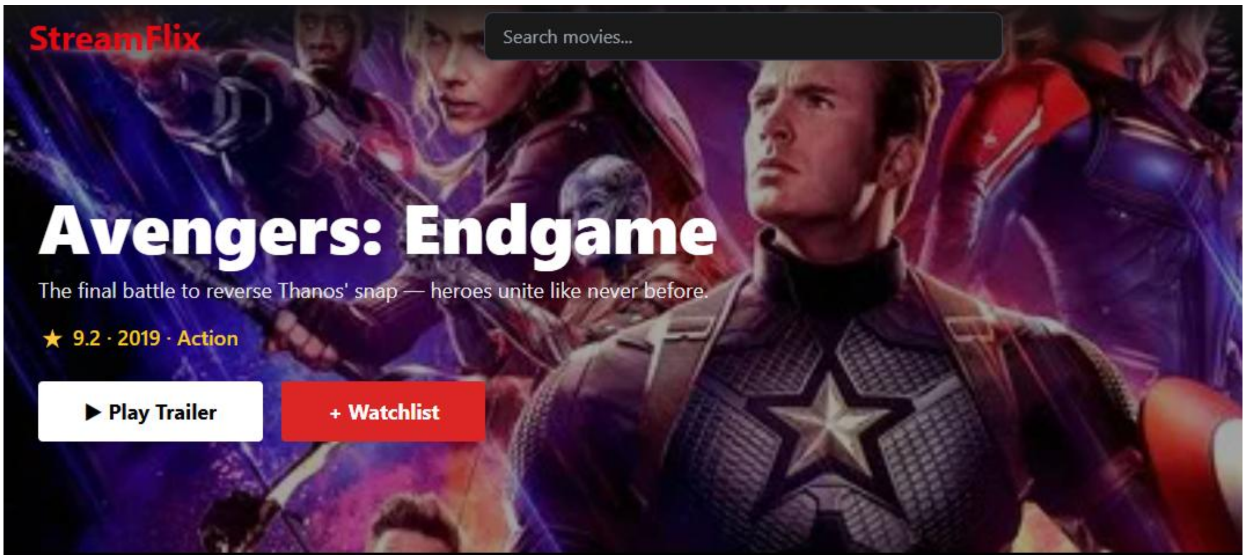


- Multiple movie rows



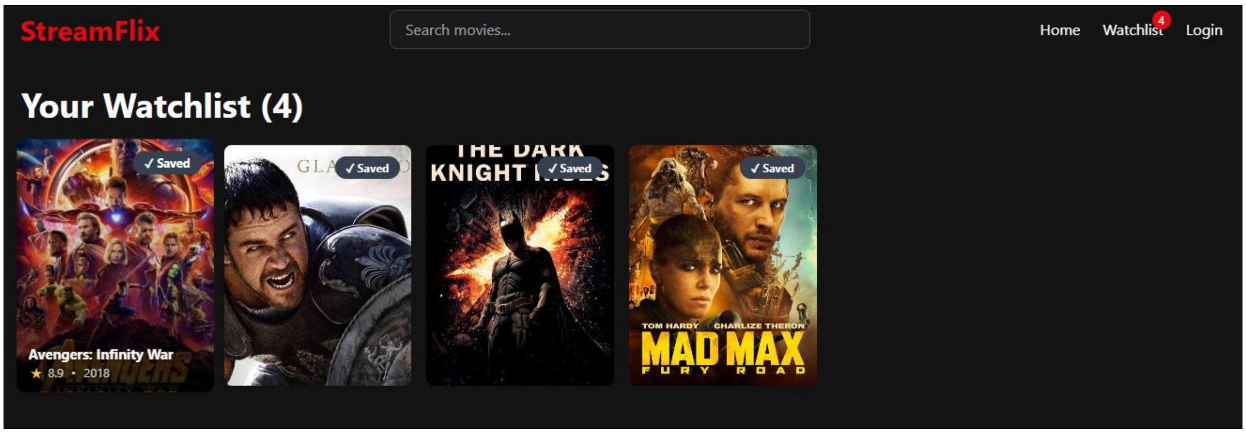
Movie Details Page

- Poster
- Title
- Genre
- Trailer



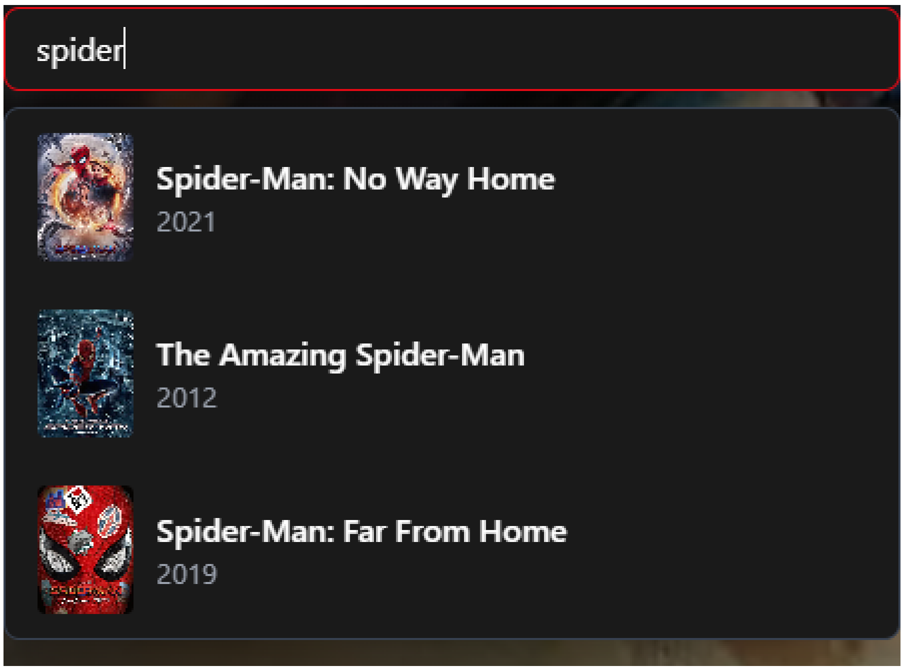
Watchlist Page

- Saved movies grid



Search Page

- Dynamic results



spider

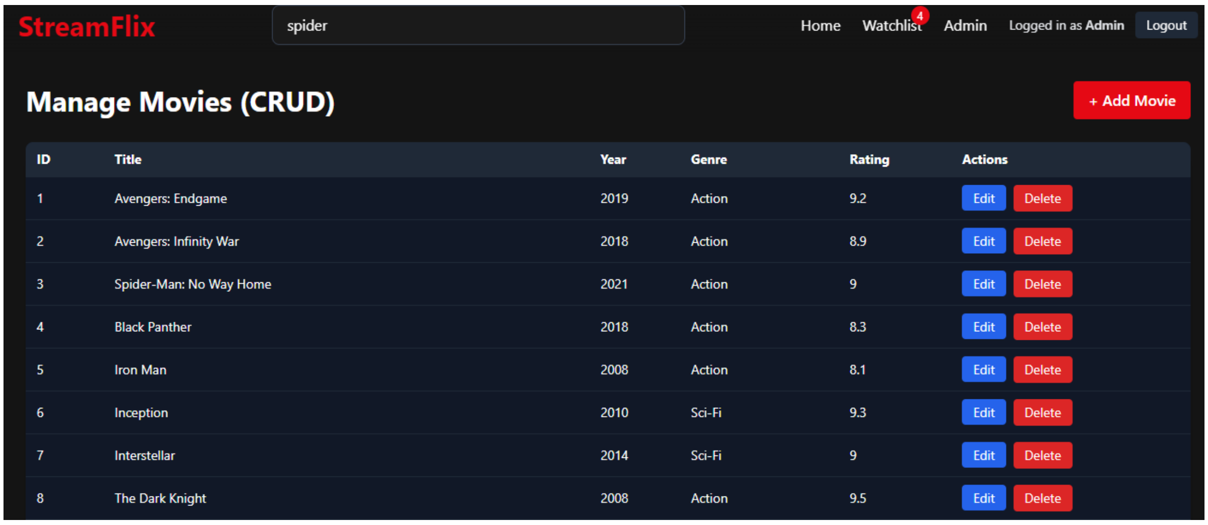
Spider-Man: No Way Home
2021

The Amazing Spider-Man
2012

Spider-Man: Far From Home
2019

Admin Panel

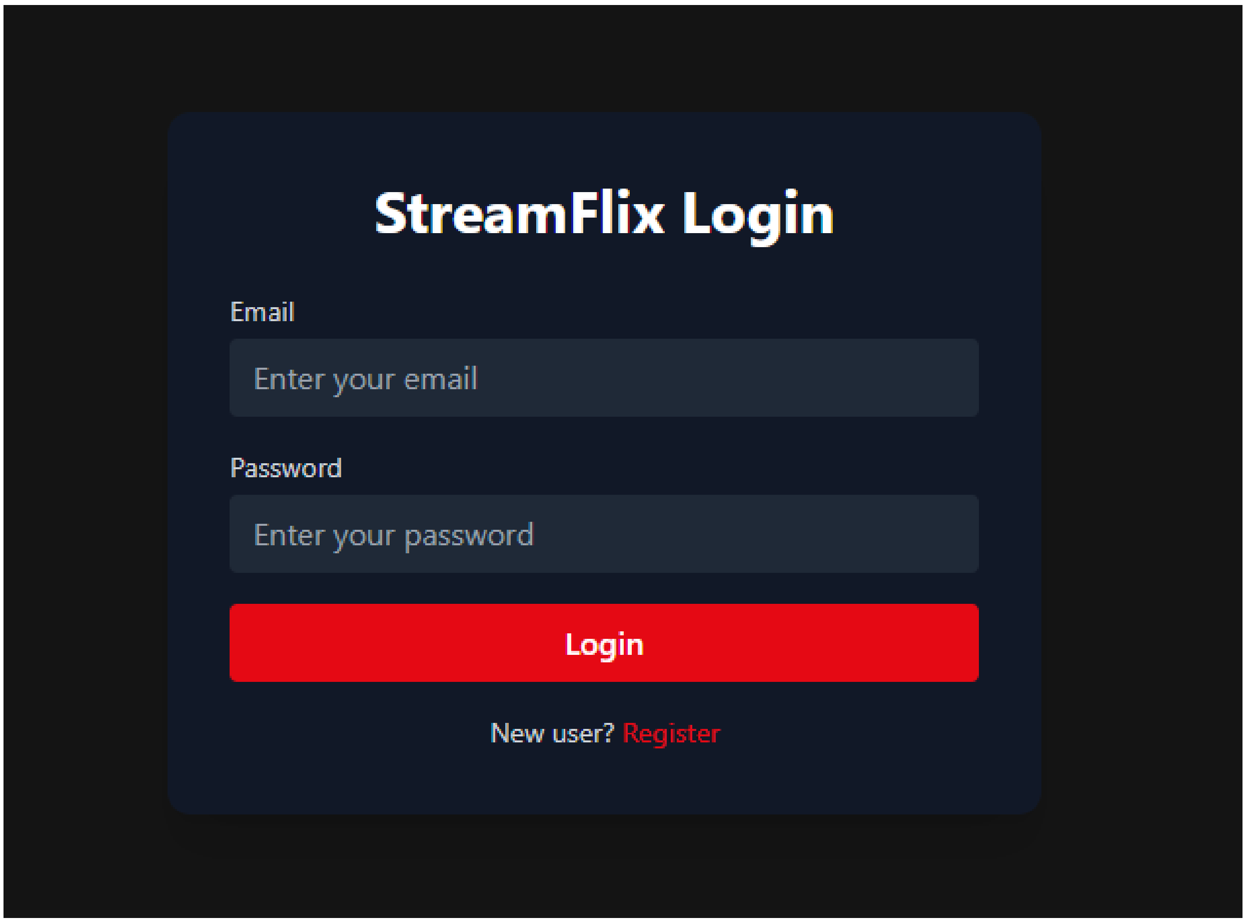
- Add movie
- Edit movie
- Delete movie



Manage Movies (CRUD)

ID	Title	Year	Genre	Rating	Actions
1	Avengers: Endgame	2019	Action	9.2	Edit Delete
2	Avengers: Infinity War	2018	Action	8.9	Edit Delete
3	Spider-Man: No Way Home	2021	Action	9	Edit Delete
4	Black Panther	2018	Action	8.3	Edit Delete
5	Iron Man	2008	Action	8.1	Edit Delete
6	Inception	2010	Sci-Fi	9.3	Edit Delete
7	Interstellar	2014	Sci-Fi	9	Edit Delete
8	The Dark Knight	2008	Action	9.5	Edit Delete

Login / Register


StreamFlix Login
Email
Enter your email
Password
Enter your password
Login
New user? Register