

StreamSavvy: Your Ultimate Entertainment Destination

StreamSavvy is a web platform that highlights your expertise in creating engaging and interactive user experiences with React. By integrating external APIs, it delivers comprehensive details about movies and TV shows in a seamless and intuitive manner.

Key highlights include:

- **Dynamic Interface:** Developed with React.js, the app effectively handles data and provides smooth transitions between pages using React Router.
- **Modern Design:** Styled with CSS/SCSS, it ensures a responsive and attractive user experience across devices.
- **Integrated Data Features:** Fetches and displays key details like reviews, ratings, and trailers using Axios for efficient API integration.
- **Efficient State Management:** Utilizes React's state handling or tools like Redux to maintain a smooth flow of data and interactions.

This project not only emphasizes your technical proficiency with React but also reflects your ability to design a platform that enriches users' entertainment exploration.

Scenario based Use-case:

Sarah's StreamSavvy Experience:

- **Morning Discovery:**
Sarah, a 26-year-old professional, enjoys relaxing after work with movies and TV shows. One morning, while browsing a forum, she stumbles upon a recommendation for StreamSavvy and decides to give it a try.
- **First Impression:**
Sarah visits the StreamSavvy website and is immediately drawn to its modern and user-friendly layout. Excited, she starts exploring the platform's features.
- **Midday Browsing:**
During her lunch break, Sarah opens the app on her phone to search for her favorite shows and movies. She's impressed by how quickly she can find ratings, reviews, and trailers all in one place.
- **Discovering Recommendations:**
Exploring further, Sarah checks out the "Recommended for You" section. She loves how the platform suggests new content that matches her interests using its smart algorithm.
- **Evening Planning:**
After work, Sarah returns to StreamSavvy to plan her evening entertainment. She checks the platform's schedule for updates on her favorite TV shows.

- **Movie Night:**

Sarah picks a top-rated movie from the app. She reads user reviews and watches the trailer before deciding to settle in for a relaxing evening.

- **Interactive Engagement:**

While watching, Sarah rates the movie and writes a quick review on StreamSavvy. She enjoys being part of a community that shares her passion for entertainment.

- **Ending the Day:**

Before going to bed, Sarah reads the latest entertainment news on StreamSavvy. She's impressed by how much content the platform offers and shares it with her friends, encouraging them to try it out too.

Target Audience:

StreamSavvy is tailored for individuals who are passionate about movies and TV shows. It caters to users who:

- Appreciate an intuitive and well-organized platform for exploring entertainment.
- Want a single hub for accessing information such as ratings, reviews, and trailers.
- Enjoy discovering new content through personalized recommendations.
- Seek a reliable and user-friendly platform to keep up with the latest trends in entertainment.

Whether you're a casual viewer looking for suggestions or a dedicated fan searching for detailed insights, StreamSavvy is designed to meet your needs and elevate your entertainment experience.

PRE-REQUISITES

PRE-REQUISITES:

Here are the key prerequisites for developing a frontend application using React.js:

- **Node.js and npm:**

Node.js is a powerful JavaScript runtime environment that allows you to run JavaScript code on the local environment. It provides a scalable and efficient platform for building network applications.

Install Node.js and npm on your development machine, as they are required to run JavaScript on the server-side.

- Download: <https://nodejs.org/en/download/>
- Installation instructions: <https://nodejs.org/en/download/package-manager/>
- **The API Source:** The API source involves [LINK](#). These are the steps to get API key and put it into .env file.
 1. Sign up for the account
 2. Go to the settings in you account
 3. And move to api section
 4. Copy the _API_TOKEN_KEY and paste it in .env folder
- **Vite:** Vite is a new frontend build tool that aims to improve the developer experience for development with the local machine, and for the build of optimized assets for production (go live). Vite (or ViteJS) includes: a development server with ES _native_ support and Hot Module Replacement; a build command based on rollup.

```
npm create vite@latest
```

- **HTML, CSS, and JavaScript:** Basic knowledge of HTML for creating the structure of your app, CSS for styling, and JavaScript for client-side interactivity is essential.
- **Version Control:** Use Git for version control, enabling collaboration and tracking changes throughout the development process. Platforms like GitHub or Bitbucket can host your repository.
 - Git: Download and installation instructions can be found at: <https://git-scm.com/downloads>
- **Development Environment:** Choose a code editor or Integrated Development Environment (IDE) that suits your preferences, such as Visual Studio Code, Sublime Text, or WebStorm.
 - Visual Studio Code: Download from <https://code.visualstudio.com/download>
 - Sublime Text: Download from <https://www.sublimetext.com/download>
 - WebStorm: Download from <https://www.jetbrains.com/webstorm/download>

To clone and run the Application project from GitHub:

Follow below steps:

- **Get the code:**

- Download the code from the drive link given below: [LINK](#)

Install Dependencies:

- Navigate into the cloned repository directory and install libraries:

```
cd movies  
  
npm install
```

- **Start the Development Server:**

- To start the development server, execute the following command:

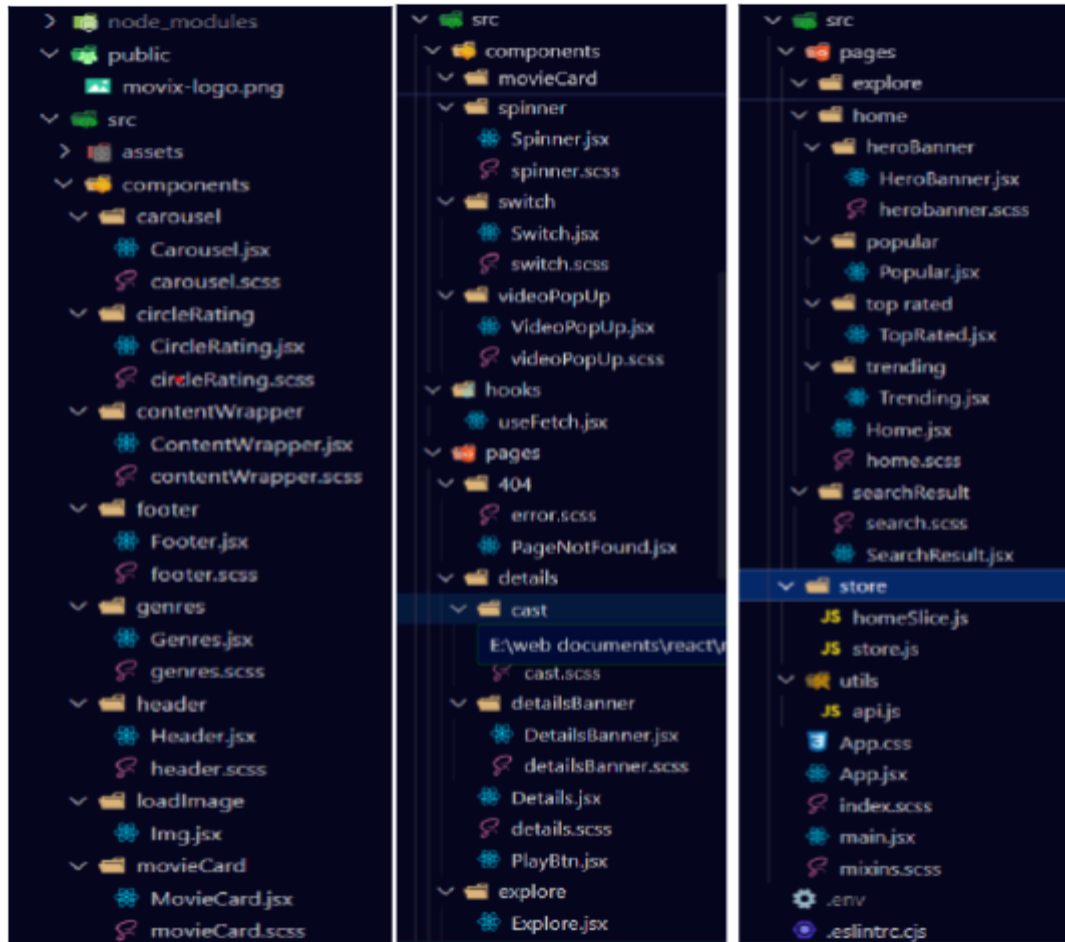
```
npm start
```

Access the App:

- Open your web browser and navigate to <http://localhost:5173/>.
- You should see the recipe app's homepage, indicating that the installation and setup were successful.

You have successfully installed and set up the application on your local machine. You can now proceed with further customization, development, and testing as needed.

Project Structure



These are the screenshots of all folder structures that are used in this project. This shows how components and other folder structures are used for the smooth building of projects.

- Here we can see that inside the src folder we have components folder, store folder, utils folder. Along with App.jsx file, App.css, etc.
- Inside the components folder we have carousel folder, circleRating folder, contentWrapper folder, footer folder, genres folder, header folder, loadImage folder, movieCard folder, spinner folder, switch folder, and videoPopUp folder.
- Inside the pages folder we have 404 folder, details folder, explore folder, home folder, and searchResult folder.

Project Flow

Project demo:

Before starting to work on this project, let's see the demo.

Demo link: [LINK](#)

Use the code: [LINK](#)

Let's Proceed with the project flow for the project development phase Project setup and configuration:

- Setup React Application:
 - Create a React app in the client folder.
 - Install required libraries
 - Create required pages and components and add routes.
- Design UI components:
 - Create Components.
 - Implement layout and styling.
 - Add navigation.
- Implement frontend logic:
 - Integration with API endpoints.
 - Implement data binding.

Project Setup And Configuration

- **Installation of required tools:**

1. Open the project folder to install the necessary tools In this project, we use:

- React Js
- React Router Dom
- Bootstrap/tailwind css
- Material UI

- For further reference, use the following resources

- <https://react.dev/learn/installation>
- <https://react-bootstrap-v4.netlify.app/getting-started/introduction/>
- <https://reactrouter.com/en/main/start/tutorial>

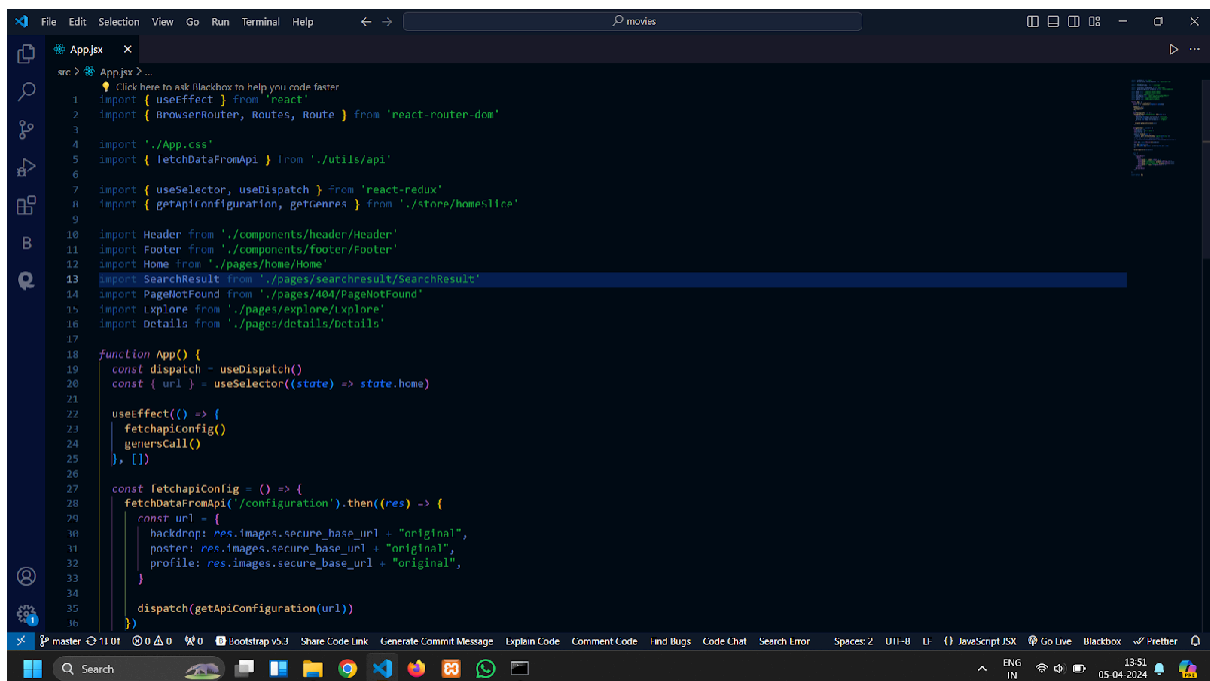
Project Development

- Setup the Routing paths

Set the clear routing paths to access various files in the application.

- some important code snips

1. App.jsx file: this is the counter file where all routes and all redux components are declared and used, to share data among all components and files. All Importing statements like `import {useEffect} from 'react'`, `import './App.css'`, etc are present in this file only.
2. It contains function like **fetchApiConfig** to configure to the API endpoints which can be used to connect through the redux present in the redux folder.



```
1 import { useEffect } from 'react'
2 import { BrowserRouter, Routes, Route } from 'react-router-dom'
3
4 import './App.css'
5 import { fetchDataFromApi } from './utils/api'
6
7 import { useSelector, useDispatch } from 'react-redux'
8 import { getApiConfiguration, getGenres } from './store/homeSlice'
9
10 import Header from './components/header/Header'
11 import Footer from './components/footer/Footer'
12 import Home from './pages/home/Home'
13 import SearchResult from './pages/searchresult/SearchResult'
14 import PageNotFound from './pages/404/PageNotFound'
15 import Explore from './pages/explore/Explore'
16 import Details from './pages/details/Details'
17
18 function App() {
19   const dispatch = useDispatch()
20   const { url } = useSelector((state) => state.home)
21
22   useEffect(() => {
23     fetchApiConfig()
24     genresCall()
25   }, [])
26
27   const fetchApiConfig = () => {
28     fetchDataFromApi('/configuration').then(res) => {
29       const url = {
30         backdrop: res.images.secure_base_url + "original",
31         poster: res.images.secure_base_url + "original",
32         profile: res.images.secure_base_url + "original",
33       }
34       dispatch(getApiConfiguration(url))
35     }
36   }
```

3. This file returns all the components and routes, params routing, etc. It contains `BrowserRouter`, `Router`, and `Route` which are imported from `react-router-dom` above in code. `Header` and `Footer` components are present which are imported from their respective folders.

```
src > App > App.jsx > @ fetchApi.config > then() > then() > @ url > @ profile
18 function App() {
19   const genresColl = async () => {
20     let allGenres = []
21
22     endpoints.forEach(url => {
23       promises.push(fetchDataFromApi(`/genre/${url}/list`))
24     })
25     // for (let i = 1; i <= endpoints.length; i++) {
26     //   promises.push(fetchDataFromApi(`/genre/${endpoints[i]}/list`))
27     // }
28     const data = await Promise.all(promises)
29
30     data.map((genres) => {
31       return genres.map((item) => allGenres[item.id] = item)
32     })
33
34     dispatch(getGenres(allGenres))
35   }
36
37   return (
38     <>
39     <BrowserRouter>
40     <Header />
41     <Routes>
42       <Route path="/" element=<Home /> />
43       <Route path="/:mediaType/:id" element=<Details /> />
44       <Route path="/search/:query" element=<SearchResult /> />
45       <Route path="/explore/:mediaType" element=<Explore /> />
46       <Route path="*" element=<PageNotFound /> />
47     </Routes>
48     </BrowserRouter>
49   )
50 }
51
52 export default App
```

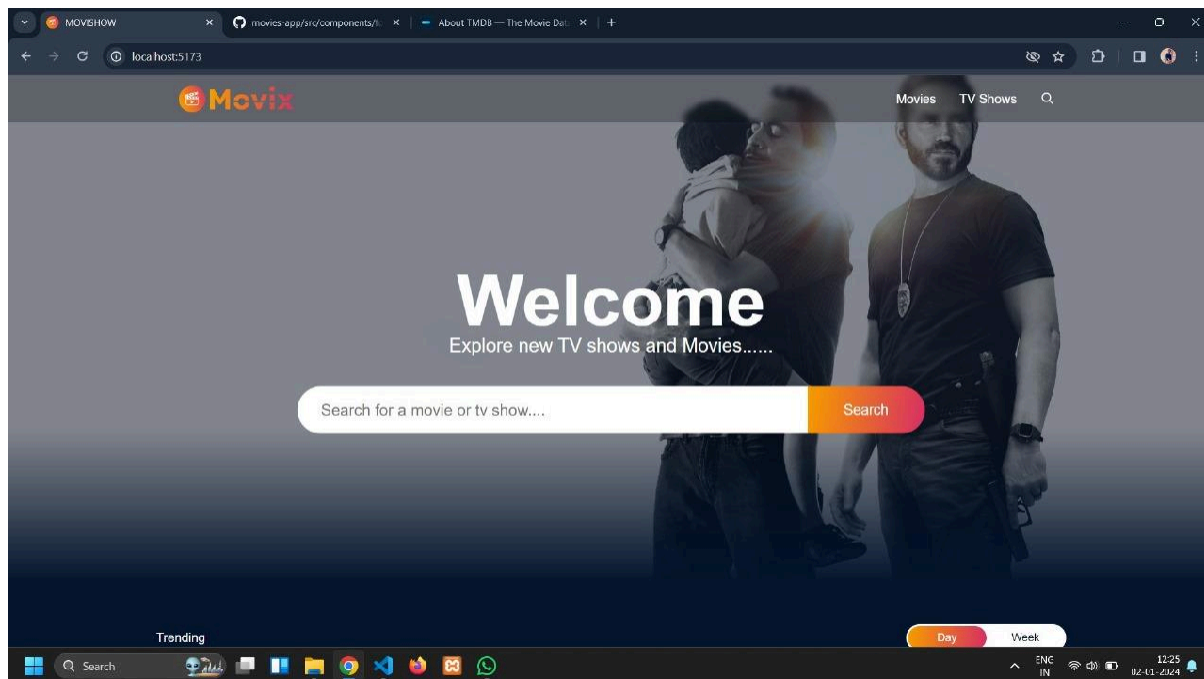
4. API file where a custom hook is defined for the project to fetch data that is coming from the API and other useful variables like BASE_URL = "API", TMDB_TOKEN = "token", and returning data that coming from the API fetching using **fetchDataFromApi** function.

```
src > api > api.js > @ fetchDataFromApi
Click here to add Bookmarks to help you code faster
1 import axios from "axios";
2
3 const BASE_URL = "https://api.themoviedb.org/3";
4
5 const TMDB_TOKEN = import.meta.env.VITE_APP_TMDB_TOKEN;
6
7 const headers = {
8   Authorization: "bearer " + TMDB_TOKEN,
9 };
10
11 export const fetchDataFromApi = async (url, params) => {
12   try {
13     const { data } = await axios.get(BASE_URL + url, {
14       headers,
15       params,
16     });
17     return data;
18   } catch (error) {
19     console.log(error);
20     return error;
21   }
22 }
23
24
25
```

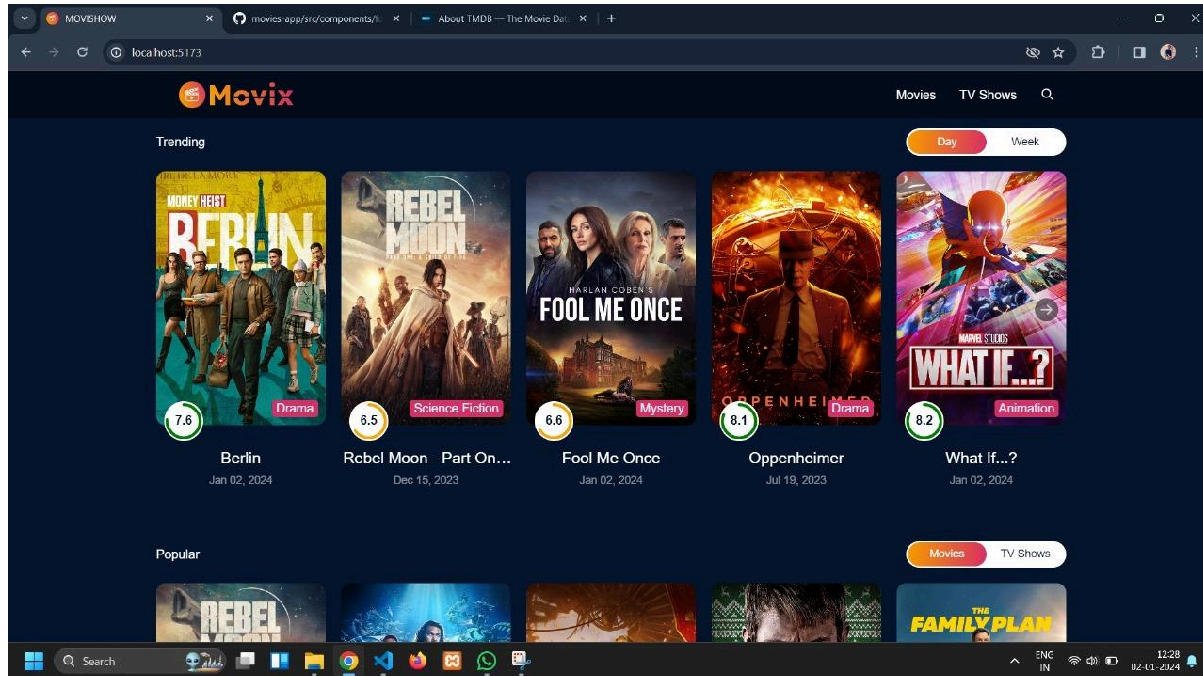

Project Implementation & Execution

Home Page

1. Here we can see the Background Image.
2. We also see a search bar.
3. There is a Navbar which can navigate us to Movies and TV Shows sections.




- We see the cards of the Movies along with their ratings, their posters as well as their titles. We can see that the navbar is still present in the page.



- Then when we click on our favourite movie card, we can see that the poster is aligned on the left hand side and the info is displayed on the right hand side along with it's ratings and watch trailer button.

MOVESHOW movies app/src/components/... About TMDb - The Movie Data... localhost:5173/movie/848326

Movix Movies TV Shows



Rebel Moon - Part One: A Child of Fire (2023)

There are no heroes. Only rebels.

Science Fiction

6.5

Watch Trailer

Overview

When a peaceful colony on the edge of the galaxy finds itself threatened by the armies of the tyrannical Regent: Balisarius, they dispatch Kora, a young woman with a mysterious past, to seek out warriors from neighboring planets to help them take a stand.

Status: Released **Release Date:** Dec 15, 2023 **Run Time:** 2h 14m

Director: Zack Snyder


Writer: Zack Snyder, Zack Snyder, Kurt Johnstad, Shay Hatten

Top Cast

Windows taskbar: Search, File Explorer, Edge, Teams, Outlook, WhatsApp, 11:28, 8/2-11-2024

MOVESHOW movies app/src/components/... About TMDb - The Movie Data... localhost:5173/movie/848326

Movix Movies TV Shows



When a peaceful colony on the edge of the galaxy finds itself threatened by the armies of the tyrannical Regent: Balisarius, they dispatch Kora, a young woman with a mysterious past, to seek out warriors from neighboring planets to help them take a stand.

Status: Released **Release Date:** Dec 15, 2023 **Run Time:** 2h 14m

Director: Zack Snyder

Writer: Zack Snyder, Zack Snyder, Kurt Johnstad, Shay Hatten

Top Cast

| Actor | Character |
|-----------------|---------------|
| Sofia Boutella | Kora |
| Michiel Huisman | Gunnar |
| Ed Skrein | Atticus Noble |
| Djimon Hounsou | Tilus |
| Bae Doona | Nemesis |
| Staz Nair | Tarak |

Windows taskbar: Search, File Explorer, Edge, Teams, Outlook, WhatsApp, 11:29, 8/2-11-2024

- We also see that clicking on the "watch trailer" button opens up the trailer of the selected movie.

