FLIXON

A Netflix clone

Content

Introduction

The Netflix clone project(FLIXON) is a web-based application that emulates some of the features and functionalities of the popular streaming service, Netflix. The primary objective of this project was to provide users with a similar experience to the original Netflix platform. The application was built using HTML, CSS, JS, TMDB API, and Firebase.

The emergence of online streaming services has revolutionized the way we consume media, and Netflix has become one of the most popular streaming services worldwide. The success of the platform can be attributed to its vast library of movies and TV shows, its user-friendly interface, and the ability to watch content on multiple devices.

The Netflix clone project aimed to replicate some of these features by providing a user-friendly platform that allows users to search for movies and watch their trailers. In addition, the application offers a login and registration system that allows users to create accounts and access the platform from anywhere.

The use of TMDB API in this project allows the application to access a vast database of movie-related

information, including plot summaries, release dates, and ratings. This information is used to provide users with a comprehensive view of each movie, which includes detailed information about the movie's plot, cast, and director.

Firebase is used as a backend system that stores user information, such as login credentials and personal details. This system also allows the application to authenticate user accounts, which ensures that only authorized users can access the platform.

Overall, the Netflix clone project provides a unique opportunity for users to experience some of the features of the original Netflix platform without having to pay for a subscription. The project is an excellent example of how different technologies can be integrated to create a functional web-based application that provides a great user experience.

Project Goal

The main goal of the Netflix clone project was to develop a web-based application that emulates some of the features and functionalities of the popular streaming service, Netflix. The application was designed to provide users with a similar experience to the original Netflix platform, while also showcasing the skills and knowledge of the developer.

The primary objectives of the project were as follows:

User Authentication: The project aimed to provide a secure login and registration system that allows users to create accounts and access the platform from anywhere. The authentication system ensures that only authorized users can access the platform and that their personal information is kept secure.

Movie Search: The project aimed to provide users with the ability to search for movies and TV shows based on their titles, genres, and release dates. This feature allows users to easily find the content they are interested in and browse the available options.

Switch Between Movie and TV Series: The project aimed to allow users to switch between movies and TV series, which are two of the primary categories of content available on Netflix. This feature allows users to easily navigate through the platform and find the content they want to watch.

User Profile: The project aimed to provide users with the ability to create a user profile that displays their personal information, such as their name and profile picture. This feature allows users to personalize their experience on the platform and share information with other users.

Display Movies in Rows Like Netflix: The project aimed to replicate the layout of the Netflix platform by displaying movies and TV shows in rows that can be scrolled through horizontally. This feature allows users to easily browse through the available content and find the movies they want to watch.

Movie Details and Trailer: The project aimed to provide users with detailed information about each movie and TV show, including their plot summaries, cast, and director. This feature allows users to learn more about the content before deciding to watch it. In addition, the

project aimed to allow users to watch movie trailers by clicking on a "Watch" button, which redirects them to YouTube.

Overall, the goal of the project was to develop a functional and user-friendly web-based application that emulates some of the key features of the Netflix platform. The application was designed to showcase the developer's skills and knowledge, while also providing a valuable experience for users.

Technical Details

The Netflix clone project was built using a variety of technologies to achieve its functionality. Here are some of the technical details of the project:

HTML and CSS: The project used HTML and CSS to create the structure and styling of the application. HTML was used to create the page structure, while CSS was used to apply styles to the various elements of the page.

JavaScript: JavaScript was used to add interactivity to the application. This included functionality such as searching for movies and displaying movie details when a user clicks on a movie.

TMDB API: The project used the TMDB API to access a vast database of movie-related information. This information was used to provide users with a comprehensive view of each movie, including detailed information about the movie's plot, cast, and director.

Firebase: Firebase was used as a backend system to store user information, such as login credentials and personal details. Firebase also allowed the application

to authenticate user accounts, which ensures that only authorized users can access the platform.

Responsive Design: The project was designed to be responsive, which means that it can be viewed on a variety of devices, including desktop computers, tablets, and smartphones. This was achieved using media queries and flexible layout techniques.

Overall, the Netflix clone project was built using a variety of technologies and frameworks that allowed for the creation of a functional and user-friendly web-based application. The combination of HTML, CSS, JavaScript, TMDB API, Firebase and responsive design helped to achieve the project's functionality and technical goals.

Features and Functionality

1. **User Authentication:** The project provides a secure login and registration system that allows users to create accounts and access the platform from anywhere. The authentication system ensures that only authorized users can access the platform and that their personal information is kept secure.

2. **Movie Search:** The project allows users to search for movies and TV shows based on their titles, genres, and release dates. The search functionality is implemented using the TMDB API, which provides access to a vast database of movie-related information.

3. **Switch Between Movie and TV Series:** The project allows users to switch between movies and TV series, which are two of the primary categories of content available on Netflix. This feature is implemented using a toggle button that allows users to switch between the two categories.

4. **User Profile:** The project allows users to create a user profile that displays their personal

information, such as their name and profile picture. Users can also edit their profile information and change their profile picture. The user profile functionality is implemented using Firebase, which allows user information to be stored securely.

5. **Display Movies in Rows Like Netflix:** The project replicates the layout of the Netflix platform by displaying movies and TV shows in rows that can be scrolled through horizontally. Each row displays a set of movies or TV shows that share a common attribute, such as their genre or release date.

6. **Movie Details and Trailer**: The project allows users to view detailed information about each movie and TV show, including their plot summaries, cast, and director. Users can also watch movie trailers by clicking on a "Watch" button, which redirects them to YouTube.

7. **Watch History**: The project allows users to keep track of their watch history. When a user clicks on a movie or TV show to watch it, the project records this information and adds it to the user's

watch history. Users can view their watch history and resume watching from where they left off.

8. **Favorite Movies:** The project allows users to mark movies and TV shows as favorites. Users can view their list of favorite movies and TV shows and easily access them.

9. **Responsive Design:** The project is designed to be responsive, which means that it can be viewed on a variety of devices, including desktop computers, tablets, and smartphones. The design is optimized for each device type and adjusts to different screen sizes.

Challenges Faced

Understanding TMDB API: One of the biggest challenges of the project we faced in understanding the TMDB API and how to use it to retrieve movie-related information. The TMDB API provides a vast amount of information about movies and TV shows, including their titles, cast, plot summaries, and release dates. Understanding how to use the API to retrieve this information, as well as how to handle API errors and responses, have been a significant challenge.

Calling Specific URLs to Retrieve Data: Another challenge we have faced was determining how to call specific URLs to retrieve data from the TMDB API. The API provides various endpoints that return different types of data, such as movie information, TV show information, and cast information. Understanding how to call the correct endpoint and parse the returned data may have been a challenging task.

Advanced JavaScript: Creating a Netflix clone project requires advanced knowledge of JavaScript, including concepts such as asynchronous programming, DOM manipulation, and event handling. You we have

encountered difficulties in implementing certain features, such as the watch history functionality or displaying movies in rows, that required more advanced JavaScript skills.

Integrating the Project with a Database: Integrating the project with a database, such as Firebase, have been a challenging task. This required understanding how to create and manage databases, as well as how to store and retrieve user information securely. Additionally, implementing user authentication and authorization required advanced knowledge of security concepts and best practices.

Responsive Design: Another challenge we have faced was creating a responsive design that could be viewed on a variety of devices, including desktops, tablets, and smartphones. This required an understanding of CSS and how to use media queries to adjust the layout and styling of the site based on the device type and screen size.

Solution

1. **Understanding TMDB API**: To understand the TMDB API, we have studied the API documentation provided by TMDB. This documentation includes detailed information about the endpoints, request parameters, response formats, and error codes. we also watched tutorials or read online resources that explain how to use the API to retrieve movie-related information. By reading and experimenting with the documentation, we have gained a better understanding of how to use the API to retrieve data.
2. **Calling Specific URLs to Retrieve Data:** To call specific URLs to retrieve data from the TMDB API, we need to understand the structure of the API endpoints and how to pass parameters to them. We have experimented with different requests and tested the responses to see what data is returned. Additionally, you may have used JavaScript Fetch to handle API requests and responses. By doing so, we have learned how to retrieve the data we need and how to handle any errors that may occur.
3. **Advanced JavaScript:** To implement advanced JavaScript features, we have studied JavaScript documentation and online resources to learn about advanced concepts like Promises, asynchronous

programming, and event handling. You may also have watched video tutorials or read articles that explain these concepts in more detail. Additionally, you may have practiced writing code that uses these concepts and debugged any errors that occurred. By doing so, you would have developed a better understanding of advanced JavaScript techniques.

4. **Integrating the Project with a Database:** To integrate the project with a database like Firebase, you may have studied Firebase documentation and followed step-by-step tutorials to learn how to create a Firebase account, set up a Firebase project, and connect your website to Firebase. You may also have studied Firebase API documentation to learn how to store and retrieve data, handle user authentication, and manage database security. By following these tutorials and experimenting with Firebase, you would have learned how to integrate your project with a database and use Firebase to store user data securely.
5. **Responsive Design**: To create a responsive design for the project, you may have studied CSS techniques, including media queries, flexbox, and grid layouts. You may have also tested your website on different devices and screen sizes to ensure that it looks and functions correctly. Additionally, you may have received feedback from users or peers on how to

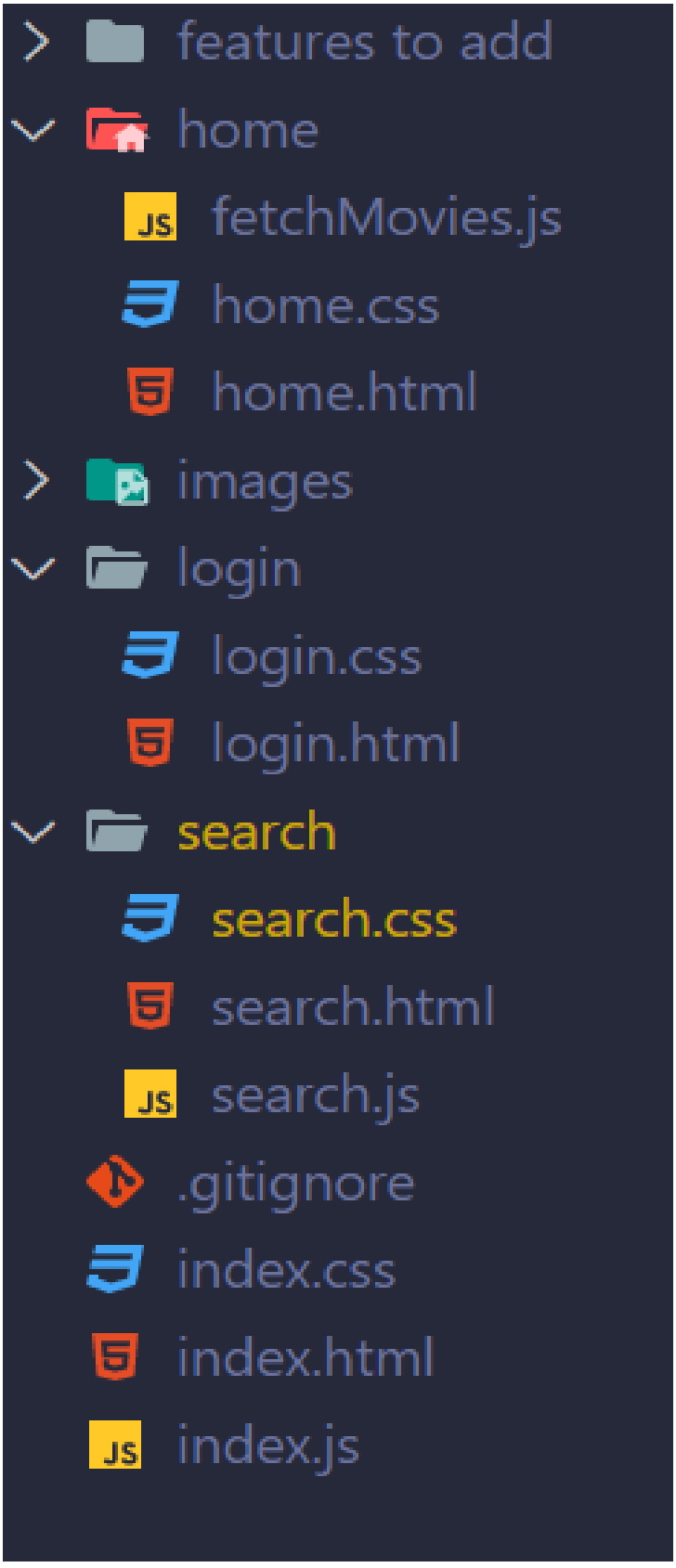
improve the website's design and usability. By doing so, we have developed a better understanding of how to create a website that is visually appealing and accessible on multiple devices.

Future Scope

1. **Implement a recommendation system:** Building a recommendation system would enhance the user experience and make the platform more engaging. Users could receive personalized movie and TV show recommendations based on their viewing history, ratings, and other factors.
2. **Add a social component**: Implementing a social component would allow users to share their favorite movies and TV shows with friends, create watchlists, and engage in discussions. This could increase user engagement and create a community around the platform.
3. **Implement a subscription system:** A subscription system could be added to the project to monetize the platform. Users could pay a monthly or yearly fee to access premium content, such as exclusive movies or TV shows.
4. **Integration with more external APIs:** The project could be expanded by integrating it with more external APIs, such as social media APIs, to create a more comprehensive user experience.

5. **Improve user interface and user experience:** The user interface and user experience can always be improved to make the platform more attractive and user-friendly. Adding new features, improving the layout, and enhancing the search functionality can all improve the user experience.

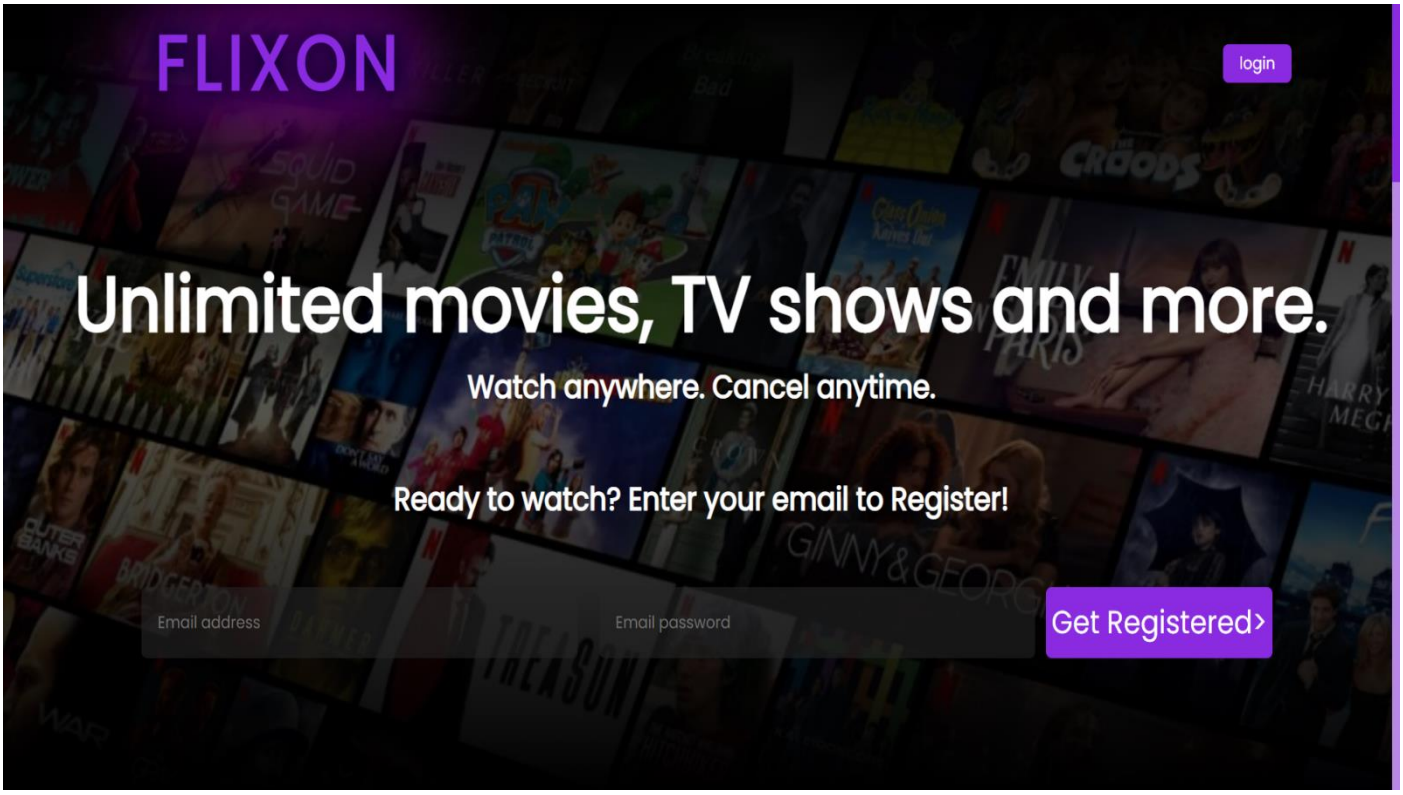
Project Structure



Source code

And

ScreenShots

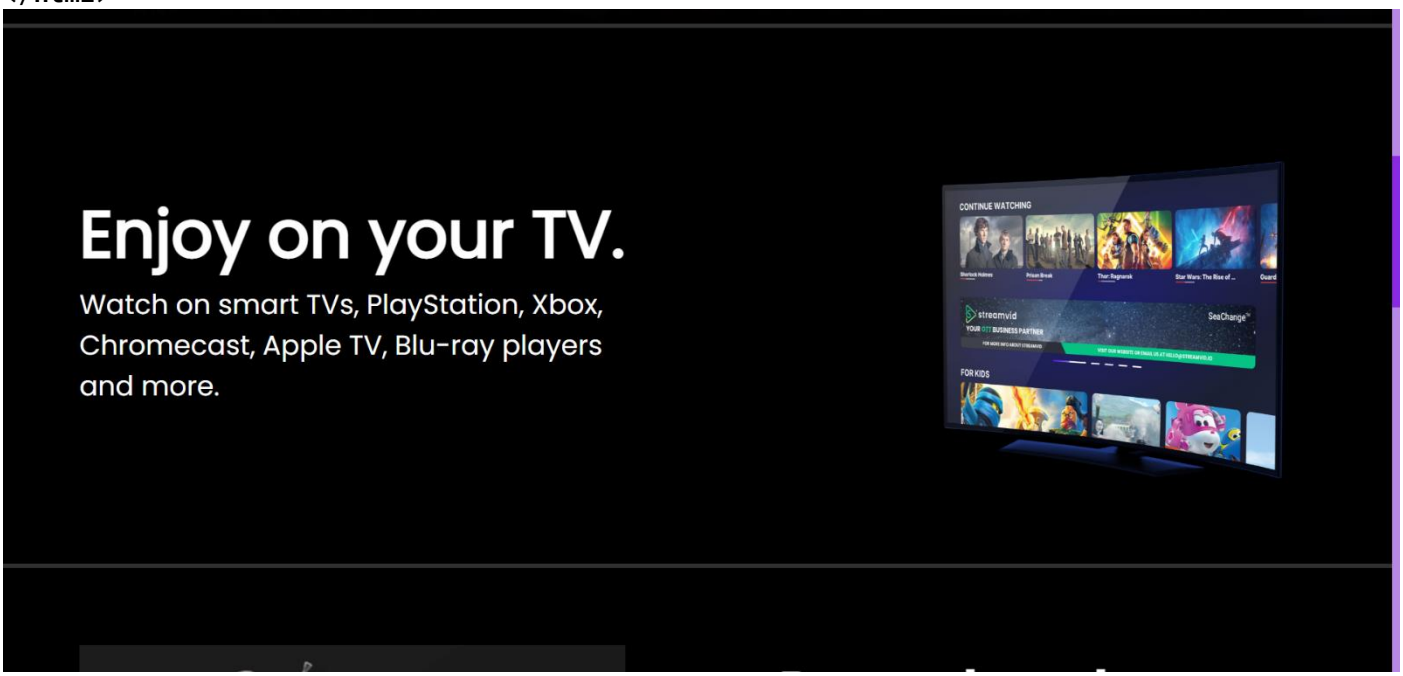


index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Register</title>
    <link rel="stylesheet" href="index.css">
    <!-- Firebase App -->
<script src="https://www.gstatic.com/firebasejs/9.6.8/firebase-app.js"></script>
<!-- Firebase Auth -->
<script src="https://www.gstatic.com/firebasejs/9.6.8/firebase-auth.js"></script>

</head>
<body>
    <section id="outer-hero">
    <div class="hero">
        <div class="navbar">
            <span id="logo" class="neon">FLIXON</span>


            <a href="login/login.html" ><button id="signin">login</button></a>
        </div>
        <div class="inner-hero">
            <h1>Unlimited movies, TV shows and more.</h1>
            <h2>Watch anywhere. Cancel anytime.</h2>
            <form class="email"  >
                <h3>Ready to watch? Enter your email to Register!</h3>
```

```
                <div class="inner-email">
                    <input id="email" type="email" placeholder="Email address" required>
                    <input id="password" type="password" placeholder="Email password"
required>
                    <button id="register" type="submit">Get Registered></button>
                </div>
            </form>
        </div>
    </div>
</section>
<hr>
    <section class="features">
        <div>
            <h1>Enjoy on your TV.</h1>
            <p>Watch on smart TVs, PlayStation, Xbox, Chromecast, Apple TV, Blu-ray
players and more.</p>
        </div>
        <img src="../images/tv.png" height="400" width="400" />
    </section>
    <hr>
    <section class="features">
        <img src="../images/mob.webp" height="350" width="350" />
        <div>
            <h1>Download your shows to watch offline.</h1>
            <p>Save your favourites easily and always have something to watch.</p>
        </div>
    </section>
    <hr>
    <section class="features">
        <div>
            <h1>Watch everywhere.</h1>
            <p>Stream unlimited movies and TV shows on your phone, tablet, laptop, and
TV.</p>
        </div>
        <img src="../images/devices.webp" height="300" width="400" />
    </section>
    <hr>
    <section class="features">
        <img src="https://occ-0-2890-
2186.1.nflxso.net/dnm/api/v6/19OhWN2dO19C9txTON9tvTFtefw/AAAABYjXrxZKtrzxQRVQNn2aIByoomnlb
XmJ-uBy7du8a5Si3xqIsgerTlwJZG1vMpqer2kvcILy0UJQnjfRUQ5cEr7gQlYqXfxUg7bz.png?r=420"
height="400" width="400" />
        <div>
            <h1>Enjoy on your TV.</h1>
            <p>Watch on smart TVs, PlayStation, Xbox, Chromecast, Apple TV, Blu-ray
players and more.</p>
        </div>
    </section>
    <hr>
    <section class="footer">
        <ul>
            <li>faqs</li>
            <li>about</li>
            <li>speed test</li>
```

```
            <li>contact us</li>
        </ul>
    </section>
    <script src="index.js" type="module" ></script>
</body>
</body>
</html>
```

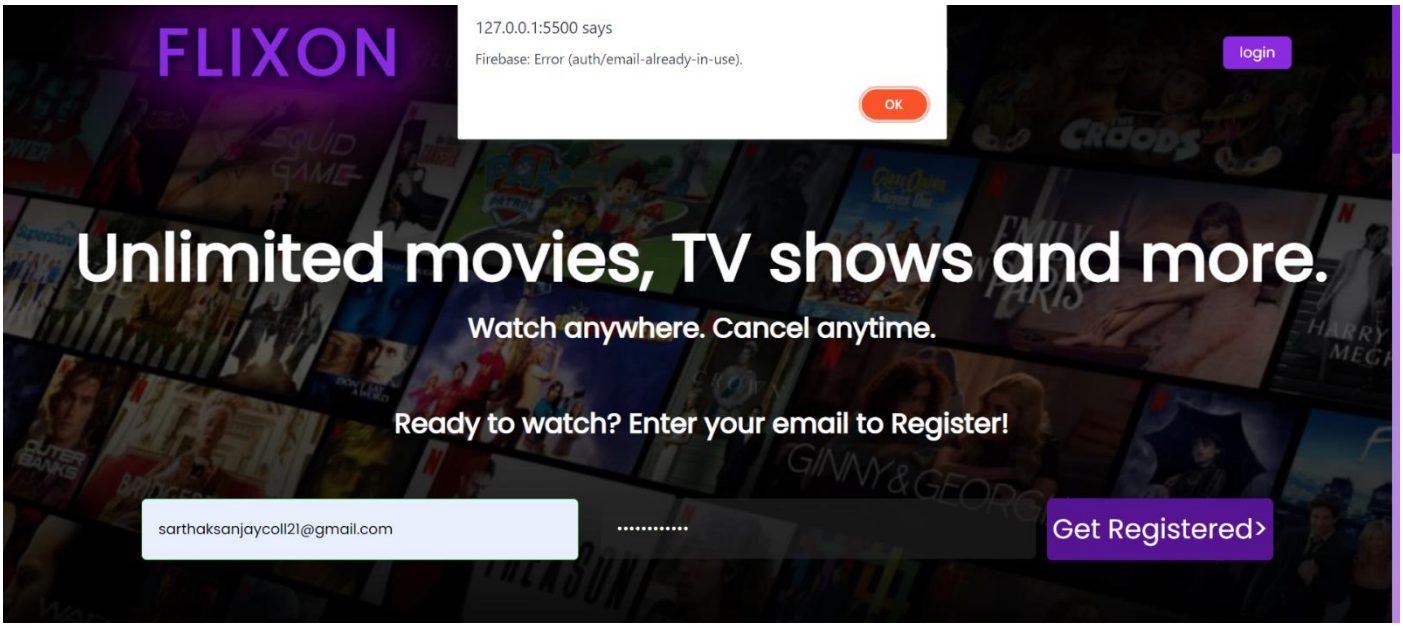


index.js

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "https://www.gstatic.com/firebasejs/9.18.0/firebase-app.js";
import { getAuth, createUserWithEmailAndPassword ,signInWithEmailAndPassword , signOut  }
from "https://www.gstatic.com/firebasejs/9.18.0/firebase-auth.js";
  // TODO: Add SDKs for Firebase products that you want to use
  // https://firebase.google.com/docs/web/setup#available-libraries

  // Your web app's Firebase configuration
  const firebaseConfig = {
    apiKey: "AIzaSyCxaCZsBFFrw9g21JmnyX4W2bNRMpRsaWA",
    authDomain: "flixon-9f63b.firebaseapp.com",
    projectId: "flixon-9f63b",
    storageBucket: "flixon-9f63b.appspot.com",
    messagingSenderId: "467380563632",
    appId: "1:467380563632:web:63df2d08a980ef1382d575"
  };

  // Initialize Firebase
const app = initializeApp(firebaseConfig);

const auth = getAuth(app);


 // Get the form element
const form = document.querySelector(".email");
```

```
// Add the submit event listener to the form
form.addEventListener("submit", function (event) {
  event.preventDefault();

  // Get the email and password values
  let email = document.getElementById("email").value;
  let password = document.getElementById("password").value;

  // Validate the email
  function isValidEmail(email) {
    const emailRegex = /\S+@\S+\.\S+/;
    return emailRegex.test(email);
  }

  // Check if the email is valid
  if (isValidEmail(email)) {
    document.getElementById("email").style.border = "1px solid green";
    console.log("valid");
  } else {
    document.getElementById("email").style.outline = "1px solid red";
    console.log("invalid email");
    return; // Stop the function execution if the email is invalid
  }

  // Create user with email and password
  createUserWithEmailAndPassword(auth, email, password)
    .then((userCredential) => {
      const user = userCredential.user;
      console.log(user);
      window.location.href = "/home/home.html";
      alert("you are now registered");
    })
    .catch((error) => {
      const errorCode = error.code;
      const errorMessage = error.message;
      console.log(errorMessage);
      alert(errorMessage);
      if (errorCode === "auth/email-already-in-use") {
        window.location.href = "login.html";
      }
    });
});
  function loginUser(email, password) {
    if (!email || !password) {
      alert("Please enter your email and password.");
      return;
    }

    signInWithEmailAndPassword(auth, email, password)
      .then((userCredential) => {
          // Signed in
      const user = userCredential.user;
      console.log(user);
      // Redirect to home page
```

```
      alert("you are now logged in");
      window.location.href = "/home/home.html";
      })
      .catch((error) => {
        const errorCode = error.code;
        const errorMessage = error.message;
        if (errorCode === "auth/user-not-found" || errorCode === "auth/wrong-password") {
          alert("Invalid email or password. Please try again.");
        } else if (errorCode === "auth/invalid-email") {
          alert("Invalid email format. Please enter a valid email.");
        } else if (errorCode === "auth/user-disabled") {
          alert("Your account has been disabled. Please contact support.");
        } else {
          console.log(errorMessage);
          alert(errorMessage);
        }      });
  }

  document.querySelector(".login").addEventListener("submit", function () {
    let email = document.getElementById("login_email").value;
    let password = document.getElementById("login_password").value;
    loginUser(email, password);
  });
```



index.css

```
@import
url('https://fonts.googleapis.com/css2?family=Poppins&family=Rubik+Iso&family=Titan+One&display=swap');
*{
    margin: 0;
    padding: 0;
    /* box-sizing: border-box; */
    font-family: 'Poppins', sans-serif;
    scroll-behavior: smooth;
```

```
}
:root{
    --light-text: white;
    --dark-text: #252525;
    --button-color: blueviolet;
}
::-webkit-scrollbar{
    width: 8px;
    background-color: rgba(134, 53, 211, 0.587);
}
::-webkit-scrollbar-thumb{
    background-color: var(--button-color);
}
body{
    overflow-x: hidden;
}
hr{
    color: #312f2f;
    height: 4px;
    background-color: #312f2f;
    /* border: 1px solid red; */
    border: none;
    width: 100vw;
}
h1{
    font-size: 40px;
    color: var(--light-text);
}

h2, h3, h4, h5, h6{
    font-size: 25px;
    color: var(--light-text);
}
/* section 1 */
#outer-hero{
    height: 100vh;
    width: 100vw;
    background-image: url(../images/background2.jpg);
    background-size: cover;
}
.hero{
    height: 100vh;
    width: 100vw;
    /* background-image: url(/images/background.jpg); */
    background-size: cover;
    /* filter: brightness(0.5); */
    background-image:linear-gradient(to top, rgba(0, 0, 0, 1.8) 0, rgba(0, 0, 0, 0) 50%,
rgba(0, 0, 0, 1.8) 100%);
    /* backdrop-filter: blur(5px); */
    background-color: #000000a7;
}

.navbar{
    display: flex;
```

```
    height: 15vh;
    width: 100vw;
    align-items: center;
    justify-content: space-between;
}
#logo{
    margin: 0px 140px;
    font-size: 60px;
    color: var(--button-color);
    font-weight: bolder;
    letter-spacing: 5px;
    /* -webkit-text-stroke: 1px white; */
}
.neon {
    font-size: 60px;
    color: transparent;
    text-shadow: 0 0 10px #002, 0 0 20px #002, 0 0 30px #002, 0 0 40px rgb(221, 0, 255), 0
0 70px rgb(221, 0, 255), 0 0 80px rgb(221, 0, 255), 0 0 100px rgb(221, 0, 255);
    animation: glow 1s ease-in-out infinite alternate;
  }

  @keyframes glow {
    from {
      text-shadow: 0 0 10px #002, 0 0 20px #002, 0 0 30px #002, 0 0 40px rgb(221, 0, 255),
0 0 70px rgb(221, 0, 255), 0 0 80px rgb(221, 0, 255), 0 0 100px rgb(221, 0, 255);
    }
    to {
      text-shadow: 0 0 20px #002, 0 0 30px #002, 0 0 40px #002, 0 0 50px rgb(221, 0, 255),
0 0 80px rgb(221, 0, 255), 0 0 90px rgb(221, 0, 255), 0 0 110px rgb(221, 0, 255);
    }
  }

/* signin button */
#signin{
    margin: 0px 100px;
    padding: 5px 15px;
    background-color: var(--button-color);
    color: var(--light-text);
    border-radius: 5px;
    outline: none;
    border: none;
}
#signin:hover{
    background-color: rgb(85, 21, 146);
}

.inner-hero{
    height: 85vh;
    width: 100vw;
    /* background-color: gray; */
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
```

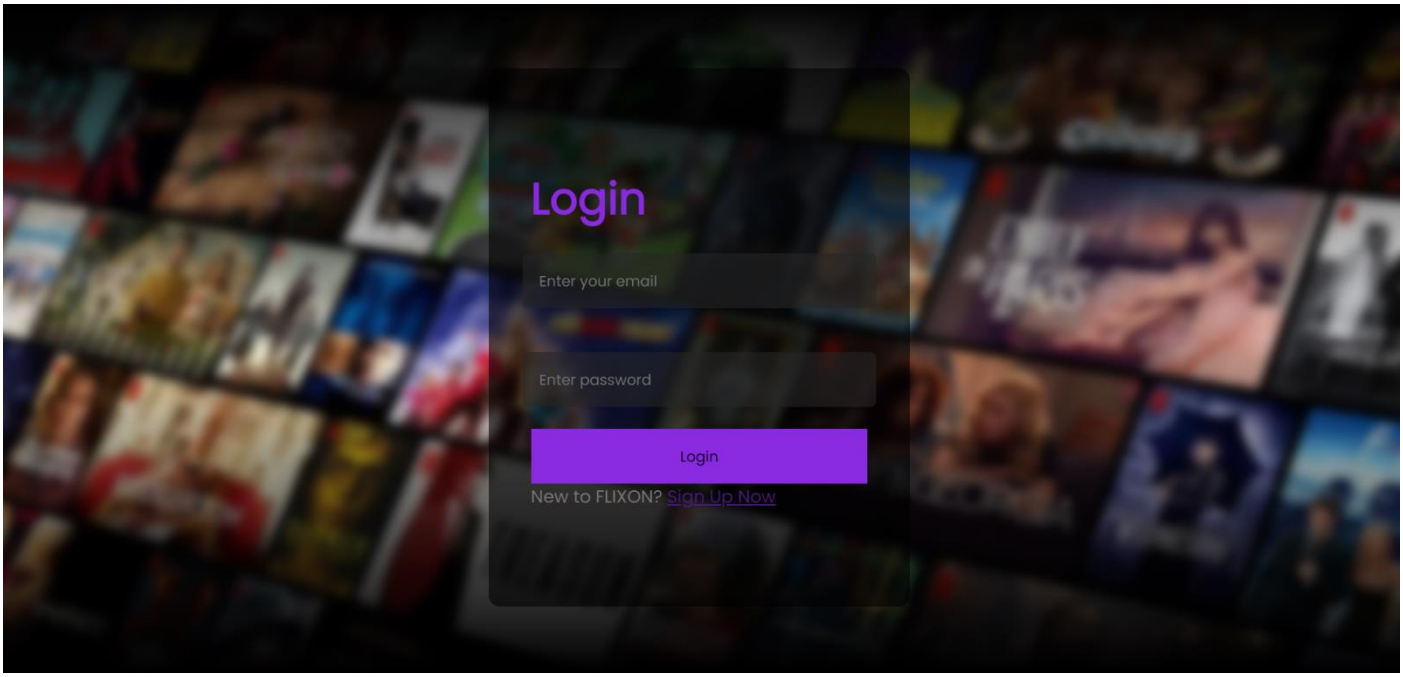
```
    color: #000;
}
.inner-hero h1{
    font-size: 60px;
}
.email{
    height: 200px;
}
.email h3{
    margin: 50px 0px;
    text-align: center;
}


.inner-email{
    display: flex;
    justify-content: center;
}
.inner-email input{
    width: 30vw;
    height: 55px;
    margin: 0 10px;
    padding-left: 15px;
    color: var(--light-text);
    background-color: #25252595;
    outline: none;
    border: none;
    border-radius: 5px;
    cursor: pointer;
}
.inner-email button{
    font-size: 25px;
    padding: 5px;
    background-color: var(--button-color);
    color: var(--light-text);
    border-radius: 5px;
    outline: none;
    border: none;
    cursor: pointer;
}
.inner-email button:hover{
    background-color: rgb(85, 21, 146);
}
.features{
    height: 80vh;
    width: 100vw;
    background-color: black;
    /* border: 1px solid red; */
    display: flex;
    justify-content: space-around;
    align-items: center;
    text-align: left;
}
.features img{
```

```
    width: 500px;
}
.features div{
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    /* border: 1px solid red; */
    /* margin-left: 30px; */
}
.features h1 , p{
    width: 500px;
    color: var(--light-text);
}
.features h1{
    font-size: 60px;
}
.features p{
    font-size: 25px;
}
.faqs{
    background-color: black;
    text-align: center;
    width: 100vw;
}
.footer{
    height: 20vh;
    width: 100vw;
    display: flex;
    justify-content: center;
    align-items: center;
    background-color: #000;
    color: var(--light-text);
}

.footer ul{
    display: flex;
}
.footer li {
    list-style: none;
    margin: 0px 20px;

}

@media only screen and (max-width:400px){
    .navbar{width: 95vw;
    padding: 0 2.5vw;}
    #logo{
        margin: 0px 14px;
        font-size: 40px;
    }
    /* signin button */
    #signin{
        margin: 0px 10px;
```

```
    padding: 0px 15px;
    font-size: 20px;

 }
.inner-hero{
 margin: 5vh 5vw;
 height: 70vh;
 width: 90vw;
 /* border: 1px solid red; */
 text-align: center;
}
 .inner-hero h1{
    font-size: 30px;
    text-align: center;
 }
 .inner-hero h2{
    font-size: 20px;
 }
 .email h3{
    margin: 10px 0px;
    font-size: 18px;
 }
 .inner-email{
    display: flex;
    width: 100%;
    flex-direction: column;
 }
 .inner-email input{
    width: 95%;
    height: 50px;
    margin: 0;
    padding: 10px;

 }
 .inner-email input::placeholder{
    width: 100%;
    padding-left: 10px;
    color: white;
 }
 .features{
    flex-direction: column;
    width: 100vw;
    height: 90vh;
 }
 .features div{
    width: 100%;

 }
 .features img{
    width: 100%;
    height: 60%;

 }
 .features h1 {
```

```
        width: 90vw;
        font-size: 25px;
    }
    .features p {
        width: 90vw;
        font-size: 15px;
    }

}
```

Login



login.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>login</title>
    <link rel="stylesheet" href="login.css">
    <!-- Firebase App -->
<script src="https://www.gstatic.com/firebasejs/9.6.8/firebase-app.js"></script>
<!-- Firebase Auth -->
<script src="https://www.gstatic.com/firebasejs/9.6.8/firebase-auth.js"></script>

</head>
<body>
   <section class="signup">
    <div class="hero">
        <form class="login">
            <h1>Login</h1>
            <input id="login_email" type="email" placeholder="Enter your email" />
```

```
            <input id="login_password" type="password" placeholder="Enter password" />
            <button id="btn" type="submit" >Login</button>
            <p>New to FLIXON? <a href="/index.html">Sign Up Now</a></p>
        </form>
    </div>
   </section>
   <script src="/index.js" type="module"></script>
</body>
</html>
```

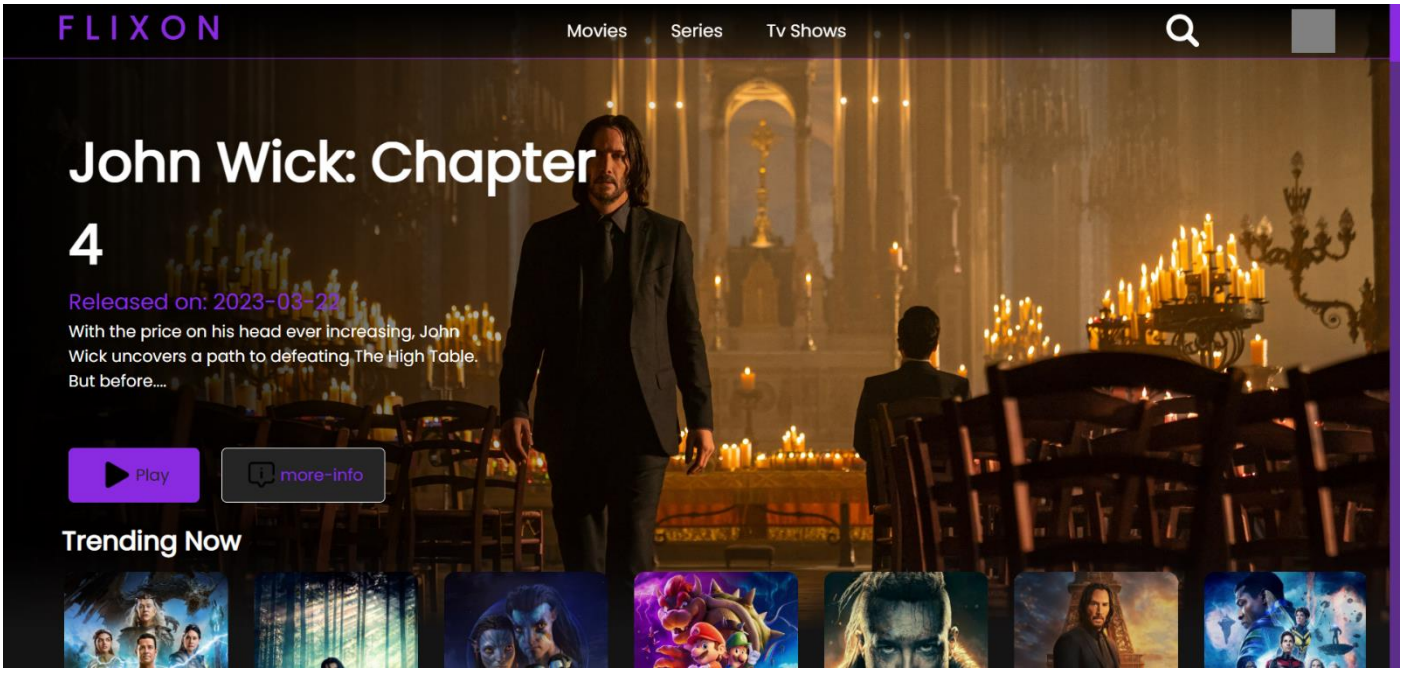
login.css

```
@import
url('https://fonts.googleapis.com/css2?family=Poppins&family=Rubik+Iso&family=Titan+One&di
splay=swap');
*{
    margin: 0;
    padding: 0;
    /* box-sizing: border-box; */
    font-family: 'Poppins', sans-serif;
    scroll-behavior: smooth;
}
:root{
    --light-text: white;
    --dark-text: #252525;
    --button-color: blueviolet;
}
::-webkit-scrollbar{
    width: 10px;
    background-color: rgba(134, 53, 211, 0.587);
}
::-webkit-scrollbar-thumb{
    background-color: var(--button-color);
}
body{
    overflow-x: hidden;
}
hr{
    color: #312f2f;
    height: 4px;
    background-color: #312f2f;
    /* border: 1px solid red; */
    border: none;
    width: 100vw;
}
h1{
    font-size: 40px;
    color: var(--light-text);
}

h2, h3, h4, h5, h6{
    font-size: 25px;
    color: var(--light-text);
}
```

```css
.signup{

    background-image: url(../images/background2.jpg);
    background-size: cover;
    width: 100vw;
    height: 100vh;

}

.hero{
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    width: 100vw;
    background-image:linear-gradient(to top, rgba(0, 0, 0, 1.8) 0, rgba(0, 0, 0, 0) 50%,
rgba(0, 0, 0, 1.8) 100%);
    backdrop-filter: blur(5px);
}

form{
    height: 80vh;
    width: 30vw;
    background-color: rgba(0, 0, 0, 0.589);
    display: flex;
    justify-content: center;
    align-items: center;
    flex-direction: column;
    border-radius: 10px;
}
form h1{
    color: var(--button-color);
    width: 80%;
    text-align: left;
}
form input{
    margin: 20px 0px;
    height: 50px;
    width: 80%;
    border: 1px solid;
    border-radius: 5px;
    padding-left: 15px;
    color: var(--light-text);
    background-color: #25252595;
    outline: none;
    border: none;
    border-radius: 5px;
}
#btn{
    background-color: var(--button-color);
    width: 80%;
    height: 50px;
    border: none;
}
```

```
#btn:hover{
    background-color: rgb(85, 21, 146);
}
form p {
    width: 80%;
    text-align: left;
    color: #7e7878;
}
```

Home page/app

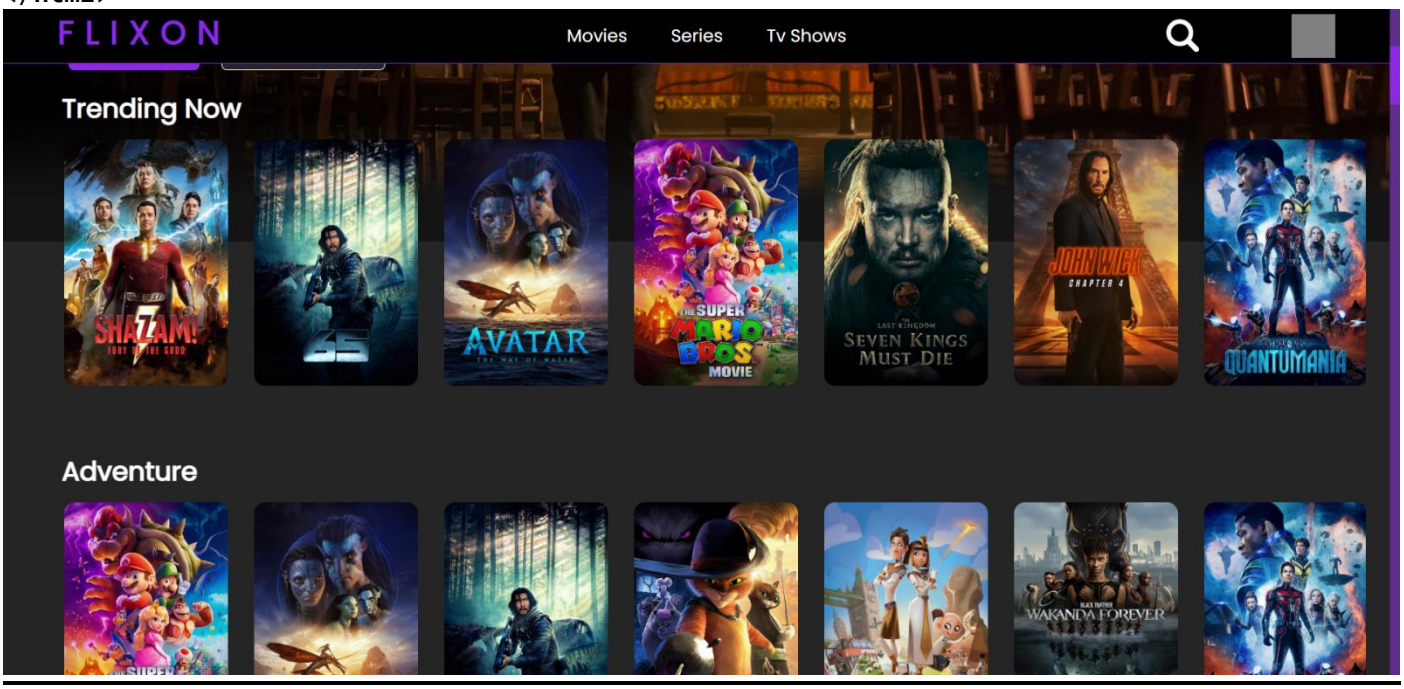


home.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>FLIXON</title>
  <link rel="stylesheet" href="home.css">
</head>

<body>
  <section class="hero">
    <div class="navbar">
      <h1>FLIXON</h1>
      <div id="nav-items">
        <span class="nav-item" id="movies">Movies</span>
        <span class="nav-item" id="series">Series</span>
        <span class="nav-item" id="tvs">Tv Shows</span>
      </div>
      <button id="search_btn"><img src="/images/search.png"/></button>
      <div>
```

```
          <div id="user_profile">
            <div id="inner-profile">
              <p>Profile</p>
              <p>History</p>
             <button id="signout">Logout</button>
            </div>
          </div>
      </div>

    </div>
    <div id="loading">
      <div id="spinner"></div>
    </div>

    <!-- banner -->
    <div class="banner" id="bannerCont">
      <div class="fade"></div>
    </div>
    <!-- banner -->
    <div class="movieContainer" id="movieContainer"></div>

  </section>
  <script type="module">
    // Import the functions you need from the SDKs you need
    import { initializeApp } from "https://www.gstatic.com/firebasejs/9.18.0/firebase-
app.js";
    import { getAuth, createUserWithEmailAndPassword, signInWithEmailAndPassword, signOut
} from "https://www.gstatic.com/firebasejs/9.18.0/firebase-auth.js";
    // TODO: Add SDKs for Firebase products that you want to use
    // https://firebase.google.com/docs/web/setup#available-libraries

    // Your web app's Firebase configuration
    const firebaseConfig = {
      apiKey: "AIzaSyCxaCZsBFFrw9g21JmnyX4W2bNRMpRsaWA",
      authDomain: "flixon-9f63b.firebaseapp.com",
      projectId: "flixon-9f63b",
      storageBucket: "flixon-9f63b.appspot.com",
      messagingSenderId: "467380563632",
      appId: "1:467380563632:web:63df2d08a980ef1382d575"
    };

    // Initialize Firebase
    const app = initializeApp(firebaseConfig);


    //   console.log(app)


    const auth = getAuth(app);
    document.getElementById("signout").addEventListener("click", function (event) {
      event.preventDefault();
      console.log("signout button clicked")
      signOut(auth)
        .then(() => {
```

```
            // Sign-out successful.
            console.log('User signed out successfully');
            // Redirect to the login page or home page
            window.location.href = "/login/login.html";
          })
          .catch((error) => {
            // An error happened.
            console.error('Error signing out:', error);
          });
        // window.location.href = "login.html";
      });

  </script>
  <script src="fetchMovies.js"></script>
  <!-- <script src="./scripts/home.js"></script> -->
  <!-- <script src="index.js" type="module"></script> -->
</body>

</html>
```

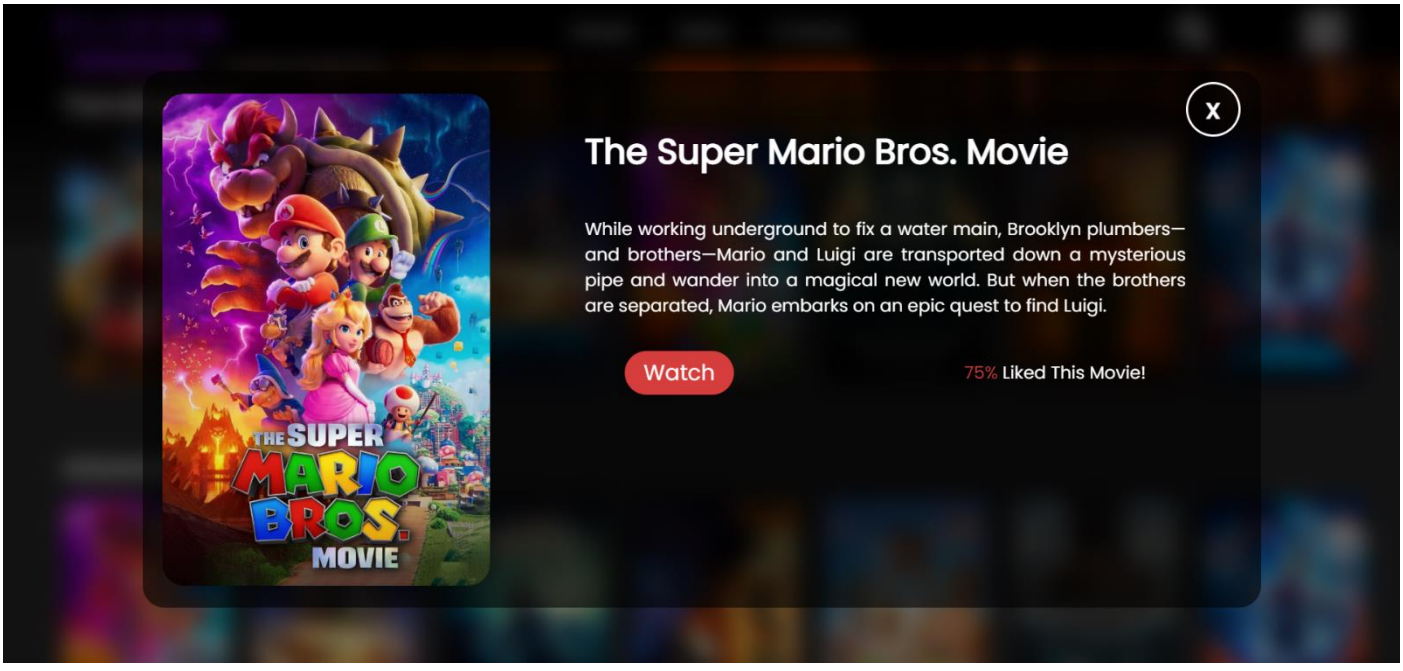


fetchMovies.js

```
//https://api.themoviedb.org/3/movie/550?api_key=9dafab561b71196c6c57491f4cd20519
// Set the API key and base URL for the MovieDB API
const apiKey = "9dafab561b71196c6c57491f4cd20519"
const baseUrl = "https://api.themoviedb.org/3"
const imgPath = "https://image.tmdb.org/t/p/original"
let contentType = "movie"



// Define the API paths for different data we want to fetch
const apiPaths = {
    fetchCategories: `${baseUrl}/genre/${contentType}/list?api_key=${apiKey}`,
```

```
    fetchMovieList: (id) =>
`${baseUrl}/discover/${contentType}?api_key=${apiKey}&with_genres=${id}`,
    fetchTrending: `${baseUrl}/trending/${contentType}/day?api_key=${apiKey}&language-en-
US`,
}



// Define an async function to fetch data from the API
const fetchData = async () => {
    // Use the fetch function to make a GET request to the API
    let data = await fetch(apiPaths.fetchCategories)
    // Use the json function to parse the response data as JSON
    let fetchData = await data.json()
    // Return the fetched data
    return fetchData
}
// Define a function to fetch and build all sections of the page



function fetchAndBuildAllSections() {
    // Use the fetch function to make a GET request to the API
    fetch(apiPaths.fetchCategories)
        // Use the json function to parse the response data as JSON
        .then(res => res.json())
        // Extract the movie genres from the response data
        .then(res => {
            const categories = res.genres
            if (Array.isArray(categories) && categories.length)
                categories.forEach(category => {
                    fetchAndBuildMovieSection(apiPaths.fetchMovieList(category.id),
category.name)
                })
        })
        // Log any errors to the console
        .catch(err => console.log(err))
}



function fetchAndBuildMovieSection(fetchData, category) {
    return fetch(fetchData)
        .then(res => res.json())
        .then(res => {
            const movies = res.results
            if (Array.isArray(movies) && movies.length) {
                buildMoviesSection(movies, category)
            }
            return movies
        })
        .catch(err => console.log(err))
}
```

```
async function buildMoviesSection(list, categoryName) {

    const moviesContainer = document.getElementById("movieContainer")
    const moviesListHTML = await Promise.all(list.map(async (item) => {
      const trailerUrl = await getMovieTrailer(item.id);
      let imgSrc = `${imgPath}${item.poster_path}`
      let description = item.overview
      let rating = Math.floor(item.vote_average * 10)
      let title = item.title
      return `
      <div class="movie-item" id="movie-item"
onclick="createAndDisplayMovieDetailPopup('${item.title }' ,' ${imgSrc}' ,
'${description}','${rating}','${trailerUrl}'  )">
         <img  class="move-item-img" src='${imgSrc}' alt='${item.title }'   )"  />
      </div>
      `
    }));

    const movieSectionHTML = `
      <h1 id="title">${categoryName}</h1><span>Explore</span>
      <div class="movies-row">
        ${moviesListHTML.join('')}
      </div>
    `
    const div = document.createElement("div")
    div.id = "moviePanel"
    div.innerHTML = movieSectionHTML
    //append div into movies container
    moviesContainer.append(div)
  }
```

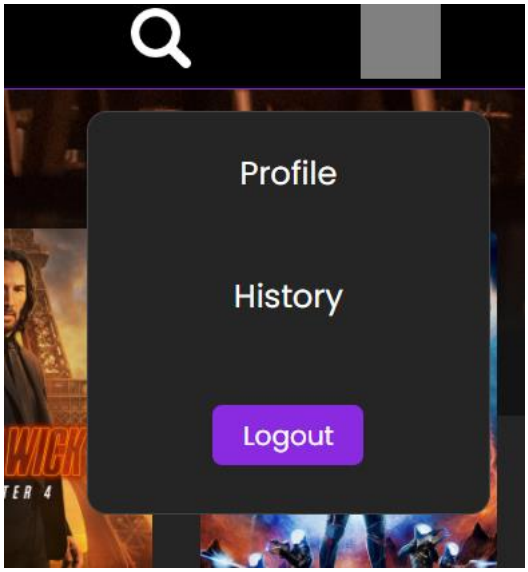



```
async function createAndDisplayMovieDetailPopup(title, movieImage, desc , rating ,
videoUrl ) {
    console.log(videoUrl)
    const popup = document.createElement("div")
```

```
    popup.className = "pop-up-div"
    popup.innerHTML = `
    <div class="popup-content">
        <div class='popup-movie-image'>
            <img src="${movieImage}" />
        </div>
        <div class="popup-movie-details">
            <button id='closeBtn'>x</button>
            <h1 class="movie-title">${title}</h1>
            <p class="desc">${desc}</p>
            <div id="btnAndRating">
             <a href="${videoUrl}"><button class="watchBtn">Watch</button></a>
         <p id='like'><span id="rating">${rating}%</span> Liked This Movie!</p>
         </div>


         </div>
        </div>
    </div>
    `
    disableScroll()
    document.body.appendChild(popup)
    const closeBtn = document.getElementById('closeBtn')
    closeBtn.addEventListener('click', () => {
        document.body.removeChild(popup)
        console.log("close button clicked")
        enableScroll()
    })



}


// Disable scrolling
function disableScroll() {
    document.body.style.overflow = 'hidden';
    // document.body.style.position = 'fixed';
}

// Enable scrolling
function enableScroll() {
    document.body.style.overflow = null;
    // document.body.style.position = null;
}



function fetchTrendingMovies() {
    fetchAndBuildMovieSection(apiPaths.fetchTrending, "Trending Now")
        .then(list => {
            // console.log(list)
            let movieItem = Math.floor(Math.random() * list.length)
            buildBannerSection(list[movieItem])
```

```
        }).catch(err => { console.log(err) })
}

 async function buildBannerSection(movie) {
    console.log(movie)
    const bannerContainer = document.getElementById('bannerCont')
    let imageUrl = `${imgPath}${movie.backdrop_path}`
    let posterUrl = `${imgPath}${movie.poster_path}`
    let bannerVideoUrl = await getMovieTrailer(movie.id)
    let rating = Math.floor(movie.vote_average * 10)
    bannerContainer.style.backgroundImage = `url(${imageUrl})`
    // console.log(movie)
    const contentDiv = document.createElement('contentDiv')
    contentDiv.id = "banner-content"
    if (movie.original_language === "en") {
        mtitle = movie.original_title ? movie.original_title : movie.name
    }
    else {
        mtitle = movie.title ? movie.title : movie.original_name
    }
    contentDiv.innerHTML = `
    <h1 id="movie-title">${mtitle.split(" ").slice(0,5).join(" ")}</h1>
    <p id="releaseDate">Released on: ${movie.release_date ? movie.release_date :
movie.first_air_date
    }</p>
    <p class="movie-desc">${movie.overview.split(" ").slice(0, 20).join(" ")}....</p>
    <div class="banner-btn">
    <a href="${bannerVideoUrl}">  <button id="play"><img class="icon"
src="/images/play.png" />Play</button></a>
    <button id="more-info" onclick="createAndDisplayMovieDetailPopup('${mtitle }' ,'
${posterUrl}' , '${movie.overview}','${rating}','${bannerVideoUrl}'  )" ><img class="icon"
src="/images/info.png" />more-info</button>
    </div>
    `
    bannerContainer.appendChild(contentDiv)
}
//movie trailer

function getMovieTrailer(movieId) {
    // Construct the API request URL to fetch the videos for the specified movie
    const url = `https://api.themoviedb.org/3/movie/${movieId}/videos?api_key=${apiKey}`;

    // Make the API request and parse the response as JSON
    return fetch(url)
      .then(response => response.json())
      .then(data => {
        // Extract the trailer URL from the response (if available)
        let trailerUrl = "";
        for (let video of data.results) {
          if (video.type === "Trailer") {
            trailerUrl = `https://www.youtube.com/watch?v=${video.key}`;
            // console.log(trailerUrl)
            break;
          }
```

```
        }

        // Return the trailer URL (or an empty string if no trailer is available)
        return trailerUrl;
      })
      .catch(error => {
        console.error("Error fetching movie videos:", error);
        return ""; // Return an empty string if an error occurs
      });
  }



//movie trailer
```




```
let profilepopup = false

document.getElementById("user_profile").addEventListener('click' , ()=>{
  if (!profilepopup) {
    document.getElementById("inner-profile").style.display = "flex"
    profilepopup = true
} else {
    document.getElementById("inner-profile").style.display = "none"
    profilepopup = false
  }
})

document.getElementById("search_btn").addEventListener("click" , ()=>{
    window.location.href = "/search/search.html"
})

// Add an event listener to run the fetchAndBuildAllSections function when the page loads



    window.addEventListener('load', () => {
        const loadingIndicator = document.getElementById('spinner');
        loadingIndicator.style.display = 'block';
        fetchTrendingMovies()
        fetchAndBuildAllSections()
```

```
        setTimeout(() => {
            loadingIndicator.style.display = 'none';
        }, 1000)
    })


window.addEventListener("scroll", function () {
    const navBar = document.querySelector(".navbar")
    if (window.scrollY > 5) {
        navBar.style.background = 'black'

    } else {
        navBar.style.background = ' linear-gradient(to bottom, black, rgba(0, 0, 0,
0.457))'
    }
})
```

home.css

```
@import
url('https://fonts.googleapis.com/css2?family=Poppins&family=Rubik+Iso&family=Titan+One&di
splay=swap');

* {
    margin: 0;
    padding: 0;
    font-family: 'Poppins', sans-serif;
    color: white;
}

body {
    background-color: #252525;
    overflow-x: hidden;
}

::-webkit-scrollbar {
    width: 10px;
    background-color: rgba(134, 53, 211, 0.587);
}

::-webkit-scrollbar-thumb {
    background-color: blueviolet;
}
.hero {
    height: 100vh;
    width: 100vw;
}

.navbar {
    position: fixed;
    top: 0;
    left: 0;
    right: 0;
```

```
    /* display: grid; */
    height: 50px;
    /* grid-template-columns: 50px 1fr 200px 100px; */
    border-bottom: 1px solid blueviolet;
    /* background-color: #252525; */
    background-image: linear-gradient(to bottom, black, rgba(0, 0, 0, 0.457));
    z-index: 6;
    display: flex;
    justify-content: space-around;

}
.navbar h1 {
    margin: 0px 50px;
    z-index: 1;
    color: blueviolet;
    letter-spacing: 10px;
    font-size: 30px
}

#nav-items {
    width: 100%;
    text-align: center;
    display: flex;
    justify-content: center;
    align-items: center;
}

.nav-item {
    margin: 0px 20px;
    cursor: pointer;
    transition: all 1s;
}

.nav-item:hover {
    text-decoration: blueviolet underline;

}

#search_btn, #signout{
    margin: 5px 20px;
    padding: 5px 15px;
    background-color:blueviolet;
    color: white;
    border-radius: 5px;
    outline: none;
    border: none;
  }
 #search_btn , #signout:hover{
    background-color: rgb(85, 21, 146);
  }
  #search_btn{
    width: fit-content;
    height: 40px;
    cursor: pointer;
```

```
    background: none;
    /* margin-top: 5px; */
}
#search_btn img{
    width: 30px;
}
#user_profile{
    height: 40px;
    width: 40px;
    background-color: gray;
    margin: 5px 50px;
    cursor: pointer;
}
#inner-profile{
    position: fixed;
    min-height: 200px;
    height: fit-content ;
    width: 200px;
    background-color: #252525;
    top: 10%;
    right: 2%;
    z-index: 20;
    border: 1px solid #454545;
    border-radius: 10px;
    display: none;
    flex-direction: column;
    justify-content: space-around;
    align-items: center;
}
/* banner */
.banner {
    height: 100vh;
    width: 100vw;
    display: flex;
    flex-direction: column;
    background-repeat: no-repeat;
    background-size: cover;
}

#banner-content {
    height: 100vh;
    width: 50vw;
    /* border: 1px solid greenyellow; */
    display: flex;
    flex-direction: column;
    justify-content: center;
    /* background-color:  */
    background-image: linear-gradient(to right, black, rgba(0, 0, 0, 0.792), rgba(0, 0, 0,
0.342), transparent);
}

#movie-title {
    font-size: 50px;
    width: 40vw;
```

```
    margin-left: 60px;
}

#releaseDate {
    margin-left: 60px;
    font-size: 20px;
    color: blueviolet;
}

.movie-desc {
    margin-left: 60px;
    font-size: 15px;
    width: 30vw;
}

.banner-btn {
    display: flex;
    width: 100%;
    margin: 50px;

}
.banner-btn a{
    text-decoration: none;
}
.banner-btn button{
    display: flex;
    align-items: center;
    justify-content: center;
}
.icon{
    width: 24px;
    margin: 0px 5px;

}
#play {
    font-size: 15px;
    width: 120px;
    height: 50px;
    background-color: blueviolet;
    color: #252525;
    border-radius: 5px;
    border: none;
    margin: 0px 10px;
    cursor: pointer;

}

#play:hover {
    background-color: rgb(88, 44, 130);
    color: white;
}

#more-info {
    width: 150px;
```

```
    height: 50px;
    font-size: 15px;
    background-color: #252525;
    color: blueviolet;
    border-radius: 5px;
    border: 1px solid white;
    margin: 0px 10px;
    cursor: pointer;
    /* padding: 0px 10px; */


}

#more-info:hover {
    background-color: #000000c7;

}

/* banner */
#movieContainer {
    width: 95vw;
    min-width: 100vh;
    margin: -200px auto;
    z-index: 5;

    /* border: 1px solid white; */

}

#title {
    margin: 0px 22px;
    /* width: fit-content; */
    display: inline;
}

#movieContainer span {
    display: none;
}

#moviePanel {
    width: 95vw;
    margin: 60px auto;
    position: relative;
    z-index: 5;
}

.movies-row {
    display: flex;
    width: 95vw;
    overflow-y: hidden;
    overflow-x: scroll;
}

.movies-row::-webkit-scrollbar {
```

```
    background-color: transparent;
    display: none;
}
.movie-item{
    width: 150px;
    height: 225px;
    border-radius: 10px;
    margin: 5px 12px;
    transition: transform 0.2s ease-in-out;
    position: relative;

}
.movieContainer img {
    width: 150px;
    height: 225px;
    border-radius: 10px;
    margin: 5px 12px;
}

.movie-item:hover {
    transform: scale(1.3 , 1.3);
}

#trending {
    height: 30vh;
    width: 95vw;
    margin: 20px auto;
    display: flex;
}

.pop-up-div{
    position: fixed;
    height: 100vh;
    width: 100vw;
    z-index: 10;
    top: 0;
    display: flex;
    justify-content: center;
    align-items: center;
    background-color: rgba(0, 0, 0, 0.5);
    backdrop-filter: blur(10px) saturate(150%);
    /* display: none; */
}
.popup-content{
    min-height: 80%;
    width: 80%;
    background-color: #00000097;
    border-radius: 20px;
    display: flex;
    flex-direction: row;
}

.popup-movie-image{
    display: flex;
```

```
    align-items: center;
    justify-content: center;
    width: 500px;
}

.popup-movie-image img{
    border-radius: 20px;
    /* height: 400px; */
    width: 300px;
}

.popup-movie-details{
    width: 100%;
}

#closeBtn{
    height: 50px;
    width: 50px;
    border: 2px solid white;
    color: white;
    outline: none;
    background:none;
    border-radius: 50%;
    font-size: 25px;
    font-weight: bolder;
    position: relative;
    top: 2%;
    left:90%;
}

#closeBtn:hover{
    background-color: #f1f1f1;
    border: 2px solid rgba(255, 248, 248, 0.788);
    color: #000000;
}

.movie-title{
    width: 90%;
    margin-left: 10%;
}

.desc{
    width: 80%;
    margin: 5% 10% 0 10%;
    text-align: justify;
}

#btnAndRating{
    display: flex;
    width: 100%;
    height: 100px;
    justify-content: space-around;
    align-items: center;
}
```

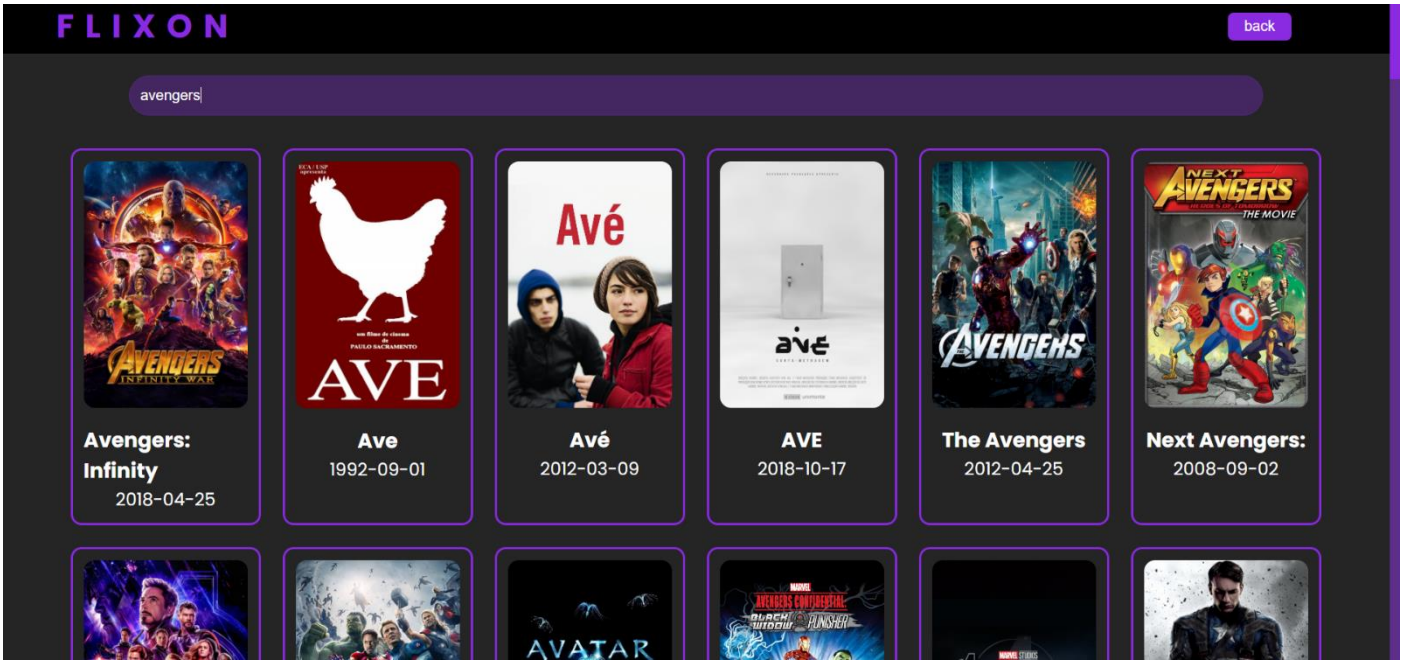
```css
.watchBtn{
    background-color: rgb(214, 62, 62);
    border: 1px solid rgb(214, 62, 62);
    outline: none;
    height:40px ;
    width: 100px;
    font-size: 20px;
    border-radius: 20px;
    /* margin: 50px 100px; */
    cursor: pointer;
}

.watchBtn:hover{
    background-color: rgb(188, 54, 54);
}

#rating{
    color: rgb(214, 62, 62);
}

.fade {
    width: 100vw;
    height: 20vh;
    position: absolute;
    top: 80%;
    z-index: 1;
    background-image: linear-gradient(180deg, hsla(0, 0%, 8%, 0) 0, hsla(0, 0%, 8%, .15) 15%, hsla(0, 0%, 8%, .35) 29%, hsla(0, 0%, 8%, .58) 44%, #141414 68%, #141414);
}

#loading{
    position: fixed;
    width: 100vw;
    top: 10vh;
}

#spinner {
    margin: 0px auto;
    z-index: 100;
    border: 4px solid #f3f3f3;
    border-top: 4px solid #000000;
    border-radius: 50%;
    width: 20px;
    height: 20px;
    animation: spin 1s linear infinite;
  }

  @keyframes spin {
    0% {
      transform: rotate(0deg);
    }
    100% {
      transform: rotate(360deg);
```

```
  }
}

#popup {
  position: relative;
  top: 0;
  /* left: 100%; */
  width: 200px;
  padding: 10px;
  background-color: #f1f1f1;
  border: 1px solid #ccc;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);
  display: none;
}

.movieContainer img:hover .popup {
  display: block;
}

/* Responsiveness */

@media (max-width: 767px) {
  /* CSS rules for mobile devices */
  .navbar{
      height: 50px;
      background-color: #000000;
  }
  #nav-items {
    display: none;
  }
  .banner{
      display: none;
  }
}

@media (min-width: 768px) and (max-width: 991px) {
  /* CSS rules for tablets */
}
```

Search

search.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>search movies...</title>
    <link rel="stylesheet" href="search.css">
</head>
<body>
    <div class="navbar">
        <h1>FLIXON</h1>
        <button id="back">back</button>

    </div>
    <div class="search-container">
        <input type="text" id="search-input" placeholder="Search for a movie..." />
        <div id="search-results"></div>
      </div>
      <script src="search.js"></script>
</body>
</html>
```

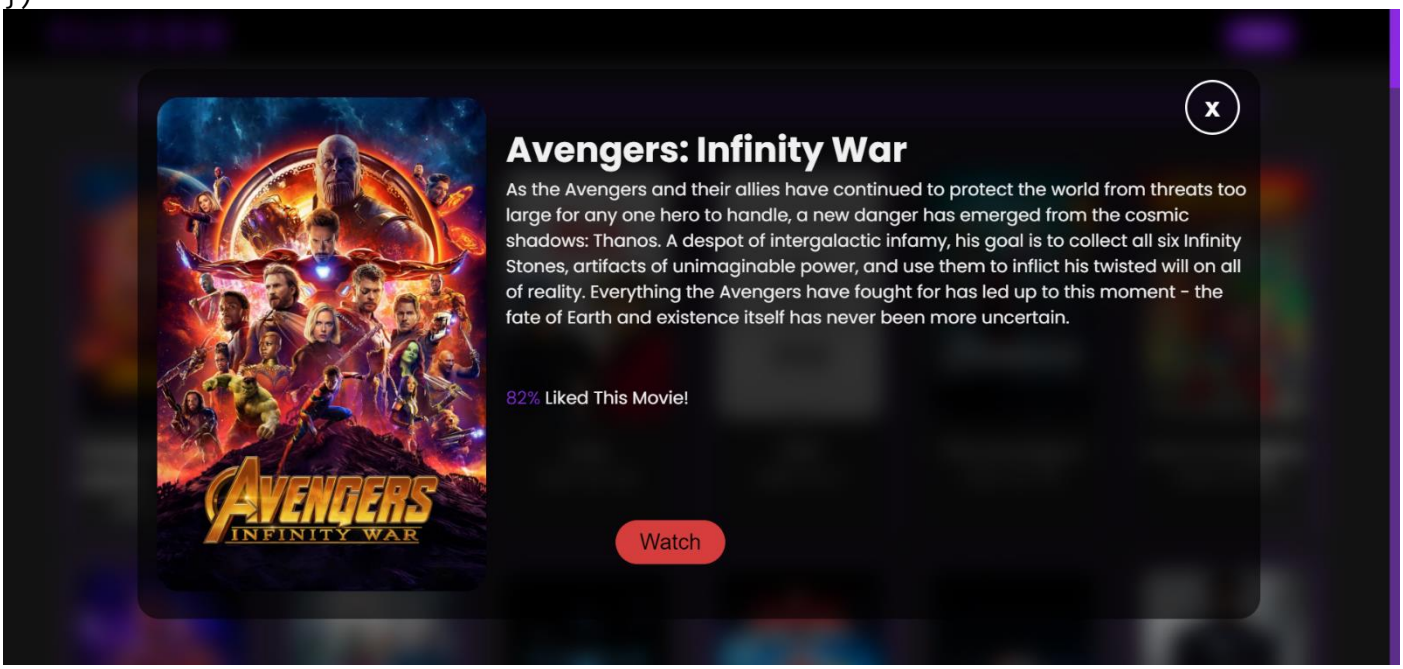


search.js

```
// Replace YOUR_API_KEY with your TMDb API key
const apiKey = '9dafab561b71196c6c57491f4cd20519';

// Get references to the search input and result container elements
const searchInput = document.getElementById('search-input');
const searchResults = document.getElementById('search-results');

// Add event listener to the search input to trigger the search
searchInput.addEventListener('input',async (e) => {
  // Get the search query from the input value
  const query = e.target.value;
  // Only search if the query is not empty
  if (query.trim() !== '') {
    // Encode the query to URL format
    const urlEncodedQuery = encodeURIComponent(query);
    // Make an HTTP request to the TMDb API search endpoint
    fetch(`https://api.themoviedb.org/3/search/movie?api_key=${apiKey}&query=${urlEncodedQ
uery}`)
      .then(response => response.json())
      .then(data => {
        // Clear the previous search results
        searchResults.innerHTML = '';
        // Iterate through the search results and create a card for each movie
        data.results.forEach(async (movie) => {
          let rating = Math.floor(movie.vote_average * 10)
          let videoUrl = await getMovieTrailer(movie.id)
         let imageUrl= `https://image.tmdb.org/t/p/original${movie.poster_path}`
```

```
          const card = document.createElement('div');
          card.className = 'card';
          card.innerHTML = `
             <div class="movie-item" id="movie-item"
onclick="createAndDisplayMovieDetailPopup('${movie.title}', '${imageUrl}','${
movie.overview}' , '${rating}')">
           <img  src="${imageUrl}" alt='${movie.title }' ,"${videoUrl}   )"  />
         </div>
            <h3>${movie.title.split(" ").slice(0,2).join(" ")}</h3>
            <p>${movie.release_date}</p>
          `;
          // Add the card to the search results container
          searchResults.appendChild(card);
        });
      })
      .catch(error => console.error(error));
  } else {
    // Clear the search results if the query is empty
    searchResults.innerHTML = '';
  }
});


async function createAndDisplayMovieDetailPopup(title, movieImage, desc ,rating ,video  )
{
  console.log("clicked")
  const popup = document.createElement("div")
  popup.className = "pop-up-div"
  popup.innerHTML = `
  <div class="popup-content">
      <div class='popup-movie-image'>
          <img src="${movieImage}" />
      </div>
      <div class="popup-movie-details">
          <button id='closeBtn'>x</button>
          <h1 class="movie-title">${title}</h1>
          <p class="desc">${desc}</p>
          <div id="btnAndRating">
          <p id='like'><span id="rating">${rating}%</span> Liked This Movie!</p>
          <a href="${video}"><button class="watchBtn">Watch</button></a>
       </div>


       </div>
      </div>
  </div>
  `
  // disableScroll()
  document.body.appendChild(popup)
  const closeBtn = document.getElementById('closeBtn')
  closeBtn.addEventListener('click', () => {
      document.body.removeChild(popup)
      console.log("close button clicked")
      enableScroll()
```

```
  })
}
async function getMovieTrailer(movieId) {
  // Construct the API request URL to fetch the videos for the specified movie
  const url = `https://api.themoviedb.org/3/movie/${movieId}/videos?api_key=${apiKey}`;

  // Make the API request and parse the response as JSON
  return fetch(url)
    .then(response => response.json())
    .then(data => {
      // Extract the trailer URL from the response (if available)
      let trailerUrl = "";
      for (let video of data.results) {
        if (video.type === "Trailer") {
          trailerUrl = `https://www.youtube.com/watch?v=${video.key}`;
          // console.log(trailerUrl)
          break;
        }
      }

      // Return the trailer URL (or an empty string if no trailer is available)
      return trailerUrl;
    })
    .catch(error => {
      console.error("Error fetching movie videos:", error);
      return ""; // Return an empty string if an error occurs
    });
}

document.getElementById('back').addEventListener("click",()=>{
  window.location.href = "/home/home.html"
})
```



search.css

```
@import
url('https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600;700&display=swa
p');
*{
  padding: 0;
  margin: 0;
}
body{
    background-color: #252525;
    font-family: 'Poppins', sans-serif;
    overflow-x: hidden;
}
.navbar {
height: 50px;
width: 100vw;
display: flex;
justify-content: space-between;
background-color: #000000;
align-items: center;

}



.navbar h1 {
  margin: 0px 50px;
  z-index: 1;
  color: blueviolet;
  letter-spacing: 10px;
  font-size: 30px
}
#back{
  margin: 0px 100px;
  padding: 5px 15px;
  background-color:blueviolet;
  color: white;
  border-radius: 5px;
  outline: none;
  border: none;
}
#back:hover{
  background-color: rgb(85, 21, 146);
}

.search-container {
    display: flex;
    flex-direction: column;
    align-items: center;
  }

  #search-input {
    width: 80%;
    padding: 10px;
    margin-top: 20px;
```

```
    border-radius: 20px;
    border: 1px solid rgba(137, 43, 226, 0.269);
    background-color: rgba(137, 43, 226, 0.308);
    outline: none;
    color: white;
  }

  .card {
    display: flex;
    flex-direction: column;
    align-items: center;
    margin: 10px;
    padding: 10px;
    border-radius: 10px;
    max-width: 150px;
    border: 2px solid blueviolet;
    color: white;
    cursor: pointer;
}

.card img {
    max-width: 100%;
    height: auto;
    margin-bottom: 10px;
    border-radius: 10px;
  }
.card h3{

}
.card p{

}
  #search-results {
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
    margin-top: 20px;
  }
  .pop-up-div{
    position: fixed;
    height: 100vh;
    width: 100vw;
    z-index: 10;
    top: 0;
    display: flex;
    justify-content: center;
    align-items: center;
    background-color: rgba(0, 0, 0, 0.5);
    backdrop-filter: blur(10px) saturate(150%);
    /* display: none; */
}
.popup-content{
    min-height: 80%;
    width: 80%;
```

```
    background-color: #00000097;
    border-radius: 20px;
    display: flex;
    flex-direction: row;
    color: #f1f1f1;

}
.popup-movie-image{
    display: flex;
    align-items: center;
    justify-content: center;
    width: 500px;
}
.popup-movie-image img{
    border-radius: 20px;
    /* height: 400px; */
    width: 300px;
}
.popup-movie-details{
    width: 100%;
}
#closeBtn{
    height: 50px;
    width: 50px;
    border: 2px solid white;
    color: white;
    outline: none;
    background:none;
    border-radius: 50%;
    font-size: 25px;
    font-weight: bolder;
    position: relative;
    top: 2%;
    left:90%;
}
#closeBtn:hover{
    background-color: #f1f1f1;
    border: 2px solid rgba(255, 248, 248, 0.788);
    color: #000000;
}
#rating{
  color: blueviolet;
}
#like{
  margin: 50px 0px;
}

.watchBtn{
    background-color: rgb(214, 62, 62);
    border: 1px solid rgb(214, 62, 62);
    outline: none;
    height:40px ;
    width: 100px;
    font-size: 20px;
```

```
    border-radius: 20px;
    margin: 50px 100px;
    cursor: pointer;
}

.watchBtn:hover{
    background-color: rgb(188, 54, 54);
}
::-webkit-scrollbar {
  width: 10px;
  background-color: rgba(134, 53, 211, 0.587);
}

::-webkit-scrollbar-thumb {
  background-color: blueviolet;
}
```