NETFLIX-CLONE

Mini Project Report

submitted by

T ROHAN KINI
(NNM22CS187)
SPOORTHI S KOTIAN
(NNM22CS178)

Under the Guidance of

Mr. ASHWIN SHENOY M
Designation

In partial fulfillment of the requirements for the award of

the Degree of

Bachelor of Engineering in Computer Science and Engineering

from

NITTE (Deemed to be University), Mangalore

NITTE (Deemed to be University) | NMAM INSTITUTE OF TECHNOLOGY

2023-2024

NITTE (Deemed to be University) | NMAM INSTITUTE OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

Certified that the Mini project work entitled

"NETFLIX-CLONE"

is a bonafide work carried out by

T ROHAN KINI NNM22CS187

SPOORTHI S KOTIAN NNM22CS187

in partial fulfilment of the requirements for the award of

Bachelor of Engineering Degree in Computer Science and Engineering

prescribed by ***Nitte (Deemed to be) University***

during the year ***2023-2024.***

The project report has been approved as it satisfies the academic requirements in respect of the project work prescribed for the Bachelor of Engineering Degree.

_____

Signature of the Guide

ACKNOWLEDGEMENT

The satisfaction that accompanies the completion of any task would be incomplete without the mention of all the people, without whom this endeavour would have been a difficult one to achieve. Their constant blessings, encouragement, guidance and suggestions have been a constant source of inspiration.

I take this opportunity to express my gratitude and regards to my project guide, **Mr. Ashwin Shenoy M**, Assistant Professor Gd-II, Department of Computer Science Engineering for his guidance, monitoring, and encouragement throughout this Mini project**.**

I would like to express my deepest gratitude to **Dr. Jyothi Shetty**, Head of the Department of Computer Science and Engineering, for her invaluable guidance, unwavering support, and encouragement throughout my academic journey.

My sincere thanks to our beloved principal, **Dr. Niranjan N Chiplunkar** and vice principal **Dr. I.R.Mithanthaya** for permitting us to carry out this Mini project at our college and providing us with all needed facilities.

Also, I thank my parents and friends for the immense support, honest opinions and suggestions throughout the internship.

T ROHAN KINI (NNM22CS187)
SPOORTHI S KOTIAN (NNM22CS178)

TABLE OF CONTENTS

ABSTRACT

This project aims to develop a Netflix clone, a web-based streaming platform designed to emulate the user interface and functionality of the widely popular Netflix service. The clone will focus on delivering a seamless and engaging user experience, offering a vast library of movies, TV shows, and other video content.

Key Features:

1. **User Authentication:** Implement a secure user authentication system to allow users to create accounts, log in, and personalize their viewing preferences.
2. **Content Catalog:** Build a comprehensive content catalog with a diverse range of movies, TV series, documentaries, and other video content. Explore integration options for third-party APIs to source content information.
3. **Search and Recommendation Engine:** Develop an efficient search algorithm and recommendation engine to help users discover new content based on their viewing history, preferences, and ratings.
4. **Streaming Infrastructure:** Implement a robust video streaming infrastructure to ensure smooth playback across various devices. Consider adaptive streaming techniques for optimal performance.
5. **Responsive Design:** Design the platform with a responsive and user-friendly interface that adapts to different screen sizes and devices, including desktops, tablets, and smartphones.

INTRODUCTION

In an era dominated by digital entertainment, the demand for on-demand streaming platforms has surged exponentially. Thus, Netflix is such, one of the platform. This project(website) aims to visualise the replica of Netflix done using HTML5, CSS, JS, and SCSS trailer rendering through API'S.

1.1 OVERVIEW AND OBJECTIVES

The first page i.e. index.html consists of "Get started " section and " FAQS" section. Wherein the 'get started' section, the function called validateform() is defined in JS which basically responsible for form validation, which checks if the given defined variable(email), matches the input email.

The main page has header section with search bar designed using JS. And movie section with subsection like Netflix original, Netflix shows, Trending Now, Top Rated, Horror Movies, Romantic Movies, Comedy Movies, Action Movies.

The main objectives of developing a Netflix clone are to replicate the key features and functionalities of the original Netflix platform while possibly adding unique elements or customization. Here are the main objectives:

1. **Content Replication:**
    - Build a comprehensive library of movies, TV shows, documentaries, and other video content, similar to Netflix's extensive catalog.
2. **Search and Discovery:**
    - Develop an efficient search functionality to allow users to find specific content quickly.
    - Create a recommendation engine based on user preferences, watch history, and ratings to enhance content discovery.
3. **Video Streaming Infrastructure:**
    - Implement a reliable and scalable video streaming infrastructure to ensure seamless playback across various devices.

- Consider adaptive streaming techniques for optimal performance under different network conditions.
- ***We are showing Released date, Languages, Rating and Genre*.**

4. **Responsive Design:**
    - Design a responsive and user-friendly interface that adapts to different devices, including desktops, tablets, and smartphones.
5. **Admin Dashboard:**
    - Create an admin dashboard for platform administrators to manage user accounts, monitor content analytics, and update the content catalog.
6. **Security Measures:**
    - Implement encryption protocols to secure user data, login credentials, and payment information.
    - Regularly update security measures to protect against potential vulnerabilities and ensure a safe streaming environment.
7. **Adaptation and Scalability:**
    - Design the platform to be adaptable to evolving technological trends and scalable to accommodate a growing user base and expanding content library.

By achieving these objectives, a Netflix clone aims to provide users with a similar streaming experience while potentially offering unique features or improvements that cater to specific preferences or market demands.

REQUIREMENTS

2.1 Requirement and Used Technology

Netflix clone using JavaScript (JS), SCSS (a syntax of Sass), and HTML/CSS, you'll need to consider several requirements. Here's a breakdown of the key requirements:

1. **HTML Structure:**
    - Create the main HTML structure for the web pages, including sections for the header, navigation, main content area, and footer.
    - Design HTML templates for individual pages such as the homepage, content details page, user profiles, etc.
2. **Styling with SCSS/CSS:**
    - Use SCSS to write modular and maintainable stylesheets. SCSS allows you to use variables, mixins, and nested rules, improving the organization of your styles.
    - Implement a responsive design that adapts to different screen sizes and devices using media queries.
    - Apply styles to create a visually appealing and user-friendly interface, mimicking the Netflix aesthetic.
3. **Navigation:**
    - Design a navigation bar that includes menu items such as Home, Browse, Search, and User Profiles.
    - Ensure that the navigation is responsive and accessible, providing a smooth user experience on various devices.
4. **Content Display:**
    - Create sections to display featured content, trending shows, and personalized recommendations.
    - Implement grid layouts to showcase multiple content items on a single page.

5. **Content Details:**
    - Design a page layout for displaying detailed information about a specific movie or TV show, including its description, cast, ratings, and related content.
    - Consider incorporating modal dialogs or lightboxes for trailers and additional information.
6. **User Authentication:**
    - Develop forms for user registration and login using HTML forms and handle authentication using JavaScript.
    - Implement user profiles with the ability to customize avatars and personal preferences.
7. **Search Functionality:**
    - Create a search bar that allows users to search for specific content.
    - Implement JavaScript functionality to handle search queries and display relevant results.
8. **Video Playback:**
    - Integrate a video player using HTML5 video capabilities and trailers are rendered from Youtube.
    - Use JavaScript to control video playback, handle user interactions, and update the UI accordingly.
9. **Responsive Design:**
    - Ensure that the entire application is responsive, providing a seamless experience across devices.
    - Test and optimize the layout for various screen sizes, including desktops, tablets, and smartphones.
10. **API Integration:**

- Explore third-party APIs to fetch and display movie/TV show information. For example, use the TMDb (The Movie Database) API for content data.
- Implement AJAX requests in JavaScript to dynamically load content without refreshing the entire page.

11. **Subscription and Payment:**
    - Design subscription and payment forms for users to sign up, upgrade, or cancel their subscription plans.
    - Integrate payment gateways using JavaScript for secure transactions.
12. **Admin Dashboard:**
    - Create an admin dashboard using HTML, CSS, and JavaScript to manage user accounts, monitor analytics, and update the content catalog.
13. **Security:**
    - Implement security measures such as HTTPS, secure user authentication, and data encryption to protect user information.
14. **Testing:**
    - Perform thorough testing across browsers and devices to ensure a consistent and bug-free experience.
15. **Documentation:**
    - Document the code, especially for complex functionalities, and provide clear instructions for developers who may work on the project in the future.

By fulfilling these requirements, you can develop a Netflix clone that replicates the key features of the original platform using HTML, CSS (SCSS), and JavaScript.

2.1.1 COMPONENTS OF Netflix

To create a Netflix clone using JS (JavaScript), SCSS (Sass), HTML, and CSS, along with integration of The Movie Database (TMDb) API, you would typically structure your project into various components. Below, I'll outline the main components and their functionalities:

1. **Header Component:**
    - Contains the Netflix logo, navigation links, and user authentication.
    - May include a search bar for users to search for content.
2. **Banner Component:**
    - Displays a large, dynamically changing banner showcasing a featured movie or TV show.
    - May include additional information such as the title and a brief description.
3. **Row Component:**
    - Represents a row of content, such as "Trending Now" or "Top Rated."
    - Utilizes the TMDb API to fetch relevant content for the specified category.
    - Displays a horizontal list of movie or TV show posters that users can scroll through.
4. **Movie Component:**
    - Represents an individual movie or TV show within a row.
    - Displays the movie poster, and may include additional details such as the title and release date.
    - Allows users to click on a movie to view more information.
5. **Modal Component:**
    - Displays detailed information about a selected movie or TV show in a modal or overlay.

- May include a trailer, cast information, and a synopsis.
- Provides a close button for users to exit the modal.

6. **Footer Component:**
    - Contains the footer elements, such as links to terms of service, privacy policy, and the Netflix help center.
7. **Authentication Component:**
    - Manages user authentication, including sign-up, login, and logout functionalities.
    - Connects with a backend server for user authentication and authorization.
8. **API Integration:**
    - Handles communication with the TMDb API to fetch information about movies and TV shows.
    - Sends requests to the API to get details such as popular movies, top-rated shows, etc.
    - Parses the API responses and updates the UI accordingly.
9. **Styles:**
    - Contains global styles for the entire application.
    - Imports the individual SCSS files for each component.
10. **Scripts:**
    - Manages the overall functionality of the application.
    - Handles events such as user interactions, button clicks, and API requests.
    - Updates the DOM based on user actions and API responses.

This is a basic breakdown of how you might structure a Netflix clone using JS, SCSS, HTML, and CSS, incorporating the TMDb API for fetching movie and TV show data.

The actual implementation may vary based on specific design choices and requirements.

2.2 VISUAL STUDIO

The IDE that we used to write and test the code is visual studio.

Visual Studio integrated development environment (IDE) from Microsoft. It is used to develop computer programs including websites, web apps, web services and mobile apps. Visual Studio uses Microsoft software development platforms such as Windows API, Windows Forms, Windows Presentation Foundation, Windows Store and Microsoft Silverlight. It can produce both native code and managed code.

Visual Studio includes a code editor supporting IntelliSense (the code completion component) as well as code refactoring. The integrated debugger works as both a source-level debugger and as a machine-level debugger. Other built-in tools include a code profiler, designer for building GUI applications, web designer, class designer, and database schema designer. It accepts plug-ins that expand the functionality at almost every level—including adding support for source control systems (like Subversion and Git) and adding new toolsets like editors and visual designers for domain-specific languages or toolsets for other aspects of the software development lifecycle (like the Azure DevOps client: Team Explorer).

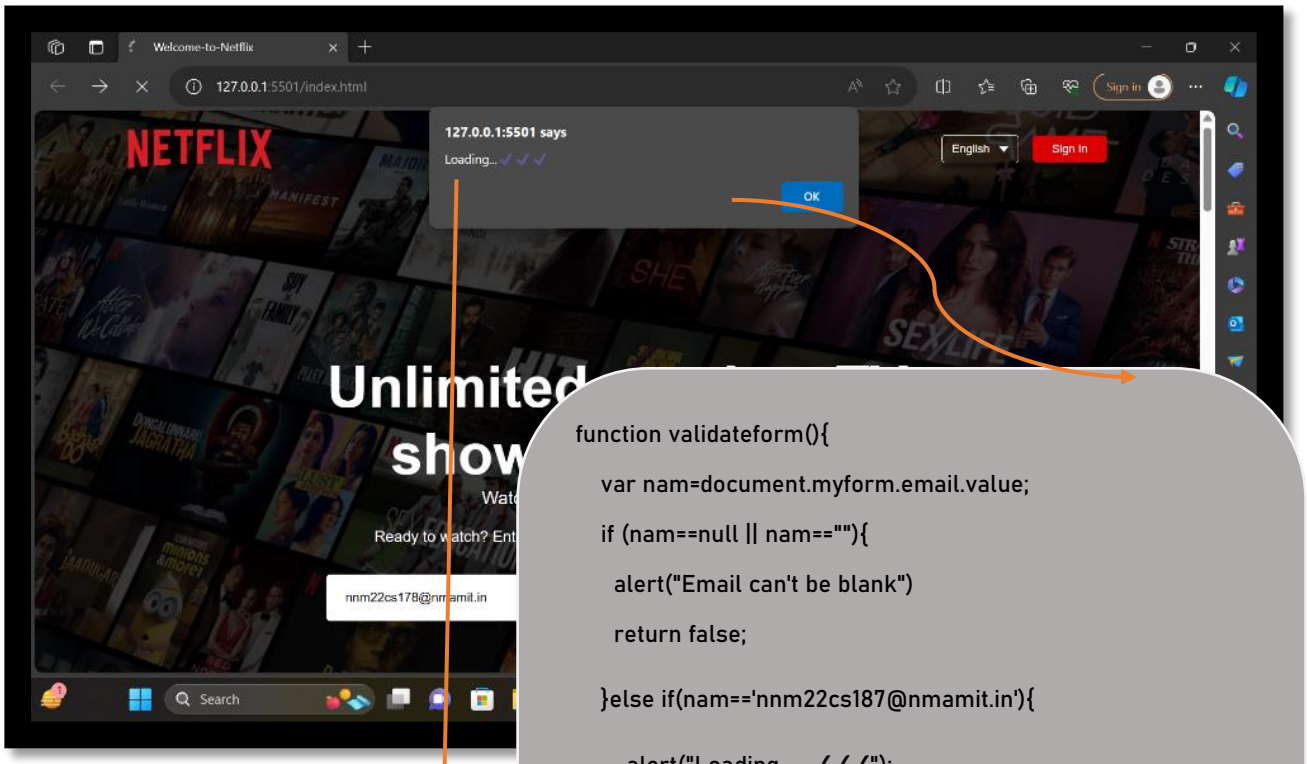
Visual Studio supports 36 different programming languages and allows the code editor and debugger to support (to varying degrees) nearly any programming language, provided a language-specific service exists. Built-in languages include C,[6] C++, C++/CLI, Visual Basic .NET, C#, F#,[7] JavaScript, TypeScript, XML, XSLT, HTML, and CSS. Support for other languages such as Python,[8] Ruby, Node.js, and M among others is available via plug-ins. Java (and J#) were supported in the past.

The most basic edition of Visual Studio, the Community edition, is available free of charge. The slogan for Visual Studio Community edition is "Free, fully featured IDE for students, open-source and individual developers". As of January 10, 2023, Visual Studio 2022 is a current production-ready version. Visual Studio 2013, 2015 and 2017 are on Extended Support, while 2019 is on Mainstream Support.

Design and Implementation

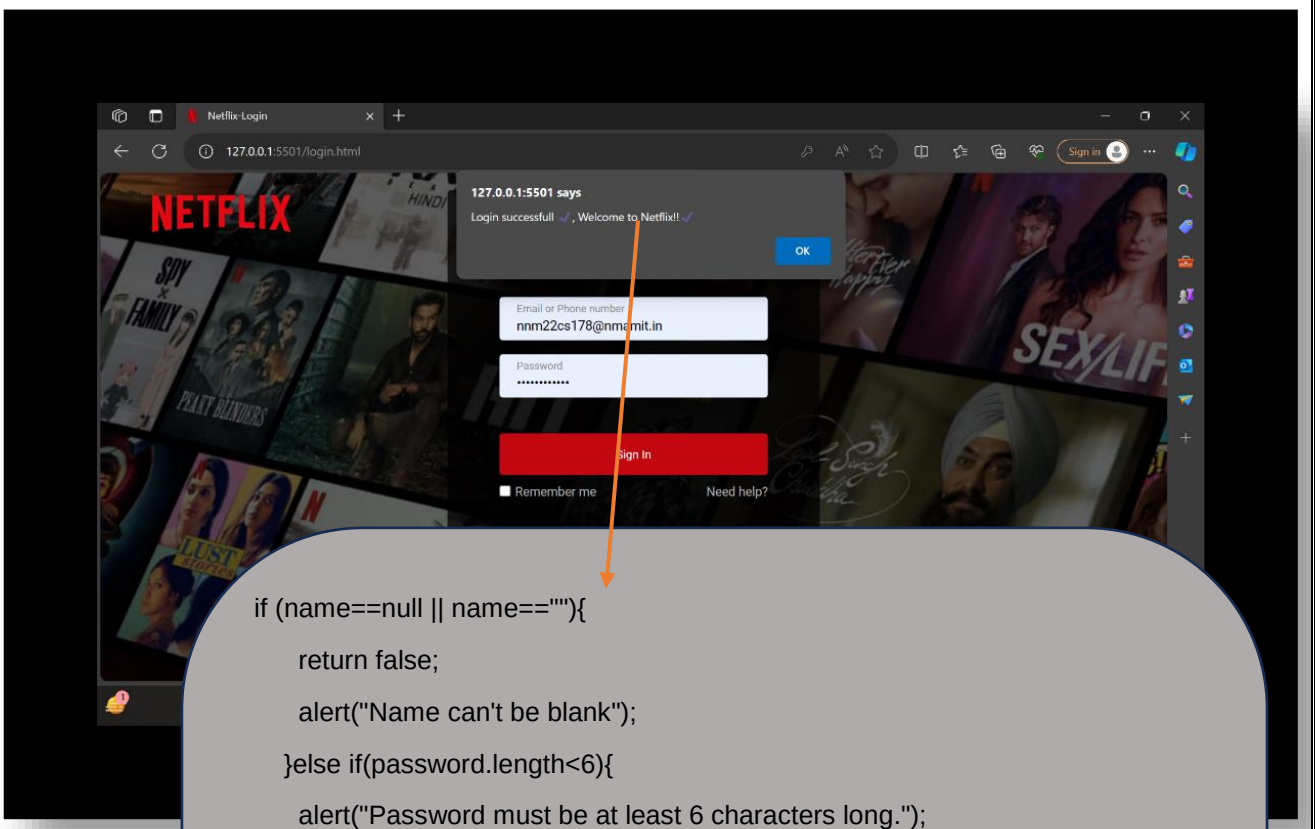
3.1 Method And PROTOTYPING

HOME PAGE(index.html)

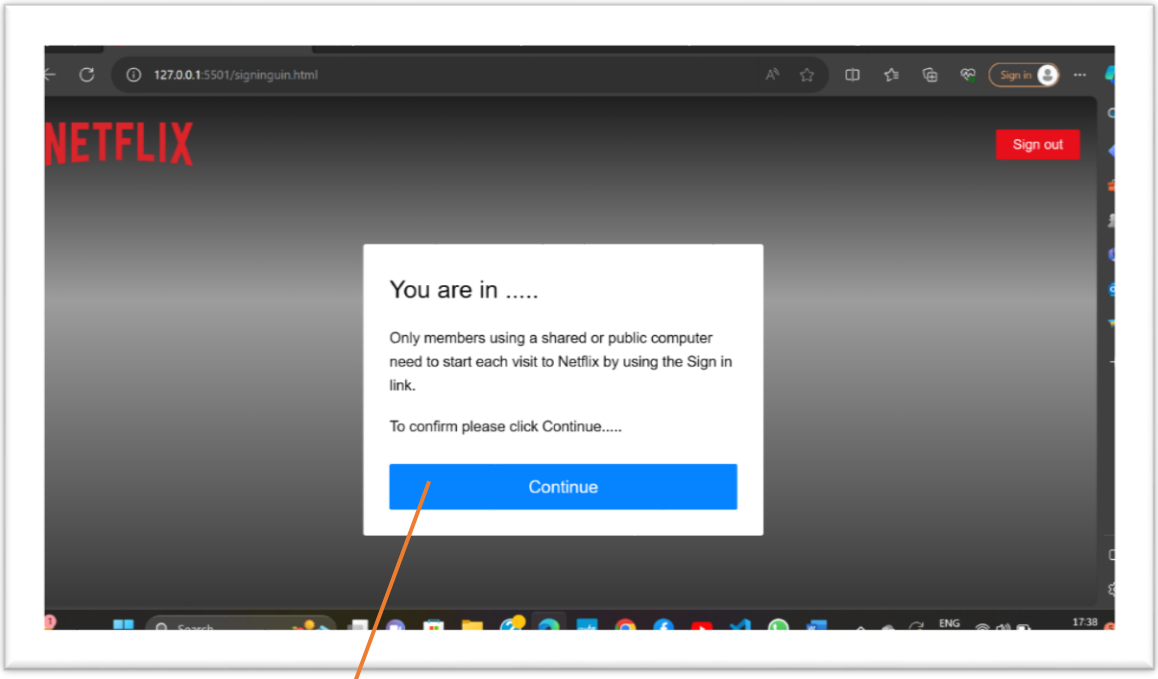



```
function validateform(){
   var nam=document.myform.email.value;
   if (nam==null || nam==""){
     alert("Email can't be blank")
     return false;
   }else if(nam=='nnm22cs187@nmamit.in'){
      alert("Loading.....✓✓✓");
      window.location.assign("login.html");
   }else if(nam=='nnm22cs178@nmamit.in'){
      alert("Loading...✓✓✓")
      window.location.assign("login.html");       }
   else
   {
      alert("❌Invalid Email❌")
      return false;
   }
}
```

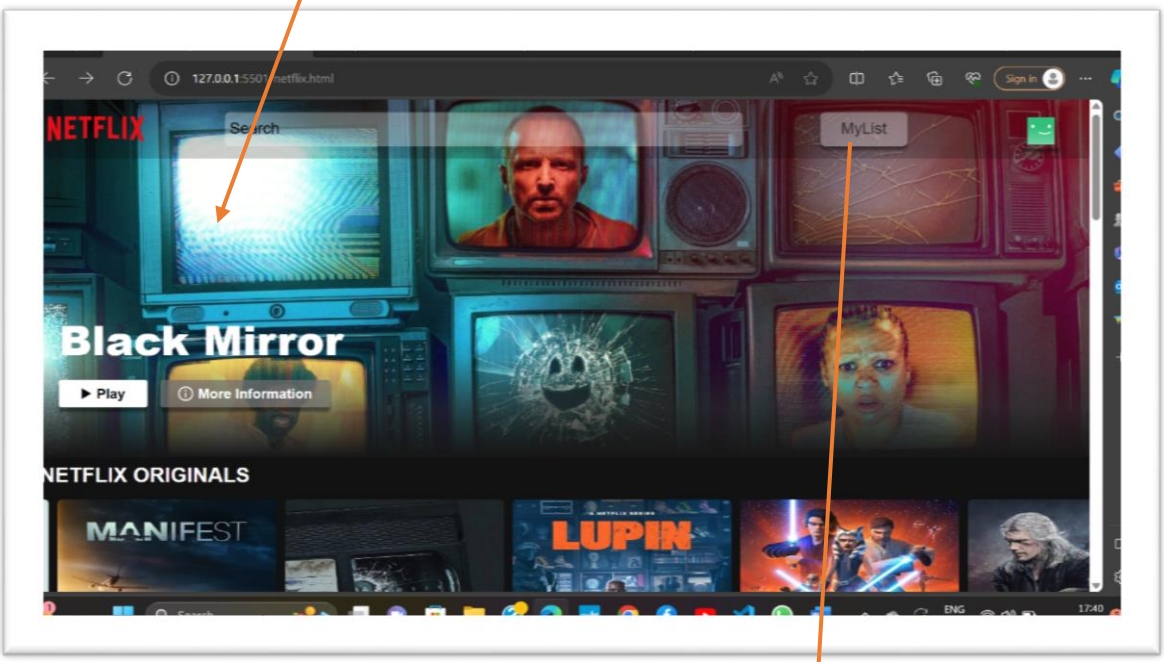
P.T.O

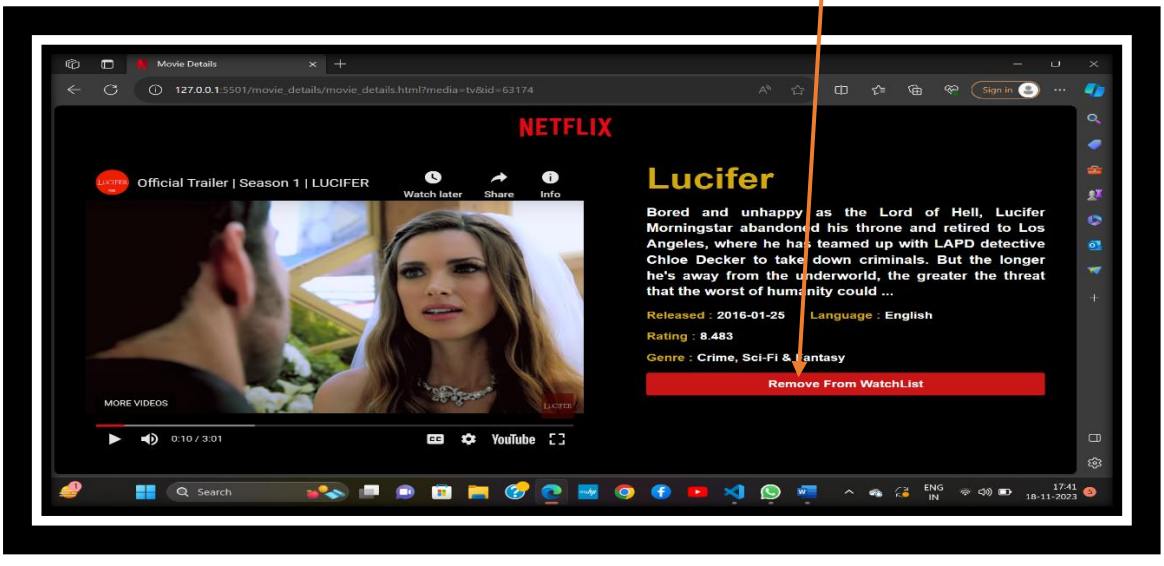



```
if (name==null || name==""){
    return false;
    alert("Name can't be blank");
  }else if(password.length<6){
    alert("Password must be at least 6 characters long.");
    return false;
    }else if(name=='nnm22cs187@nmamit.in' && password=='Rohan1234'){
      alert("Login successfull ✓, Welcome to Netflix!!✓")
      window.location.assign("signinguin.html");
  }else if(name=='nnm22cs178@nmamit.in' && password=='Spoorthi1234'){
      alert("Login successfull ✓, Welcome to Netflix!!✓")
      window.location.assign("signinguin.html");
  }
  else{
    alert("✕ Invalid Email or Password ✕");
    return false;
```

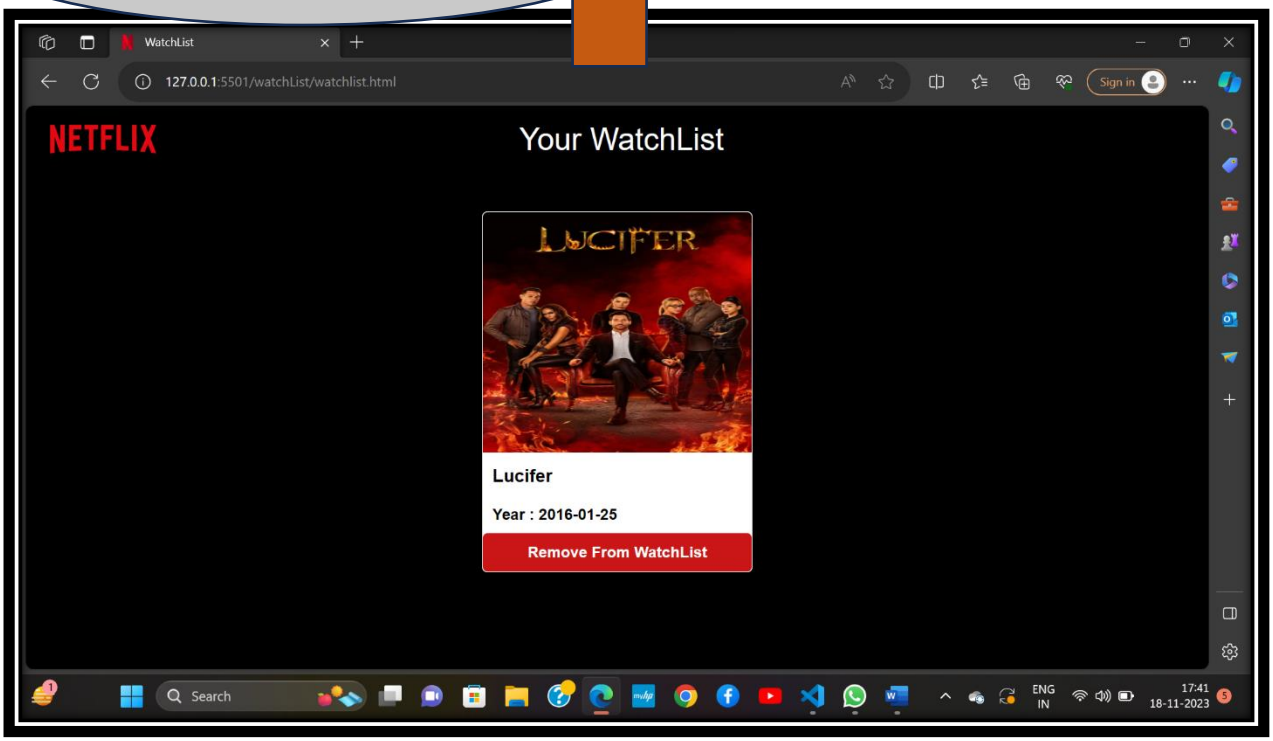


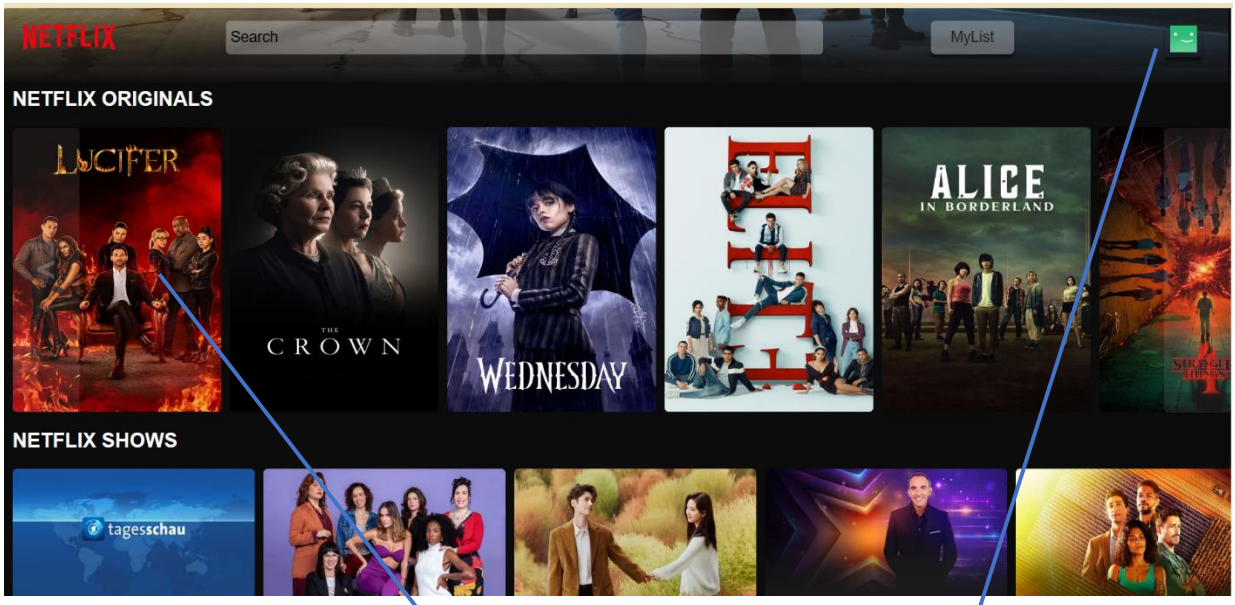
MAIN NETFLIX PAGE



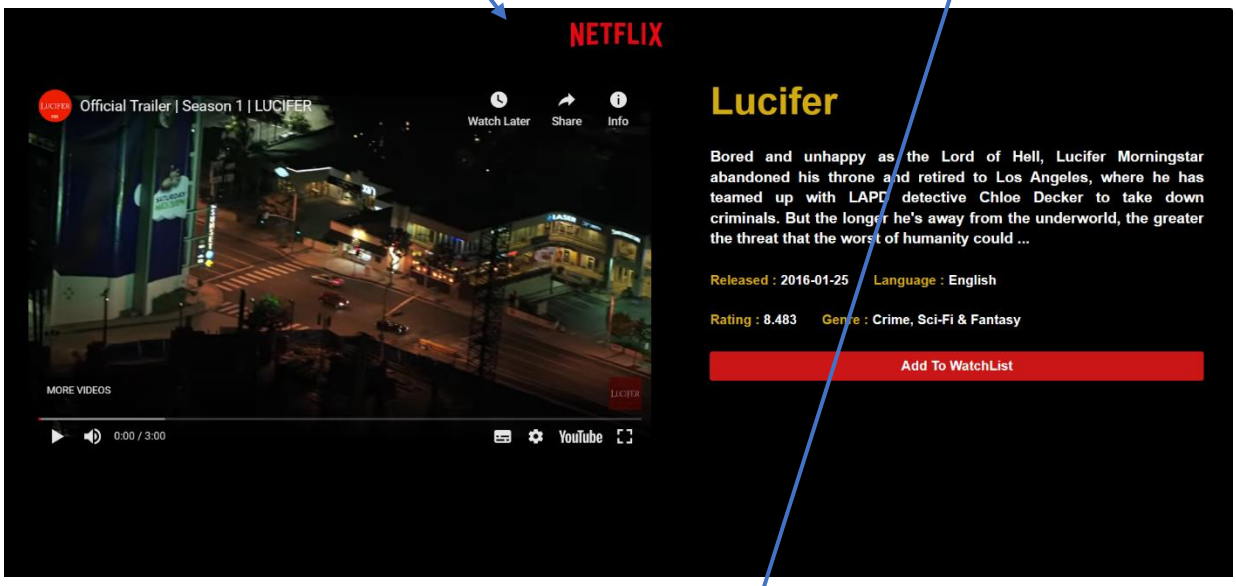


```
if (storedWatchList.length === 0) {
        const emptyMessage = document.createElement('p');
        emptyMessage.textContent = "It's lonely here. Add some Movies  or Tv shows to WatchList !";
        watchListItems.appendChild(emptyMessage);
    } else {
        storedWatchList.forEach(movie => {
            const shortenedTitle = movie.title || movie.name;
            const date = movie.release_date || movie.first_air_date;
            const watchList_Item = document.createElement('div');
            watchList_Item.classList.add('watchlist-item');
            watchList_Item.innerHTML = `<div class="search-item-thumbnail">
                    <img src="https://image.tmdb.org/t/p/w500${movie.poster_path}">
                </div>
                <div class="search-item-info">
                    <h3>${shortenedTitle}</h3>
                    <h4>Year : ${date}</h4>
                </div>
                <button class="removeBtn" id="${movie.id}">Remove From WatchList</button>`;
            watchListItems.appendChild(watchList_Item);
```

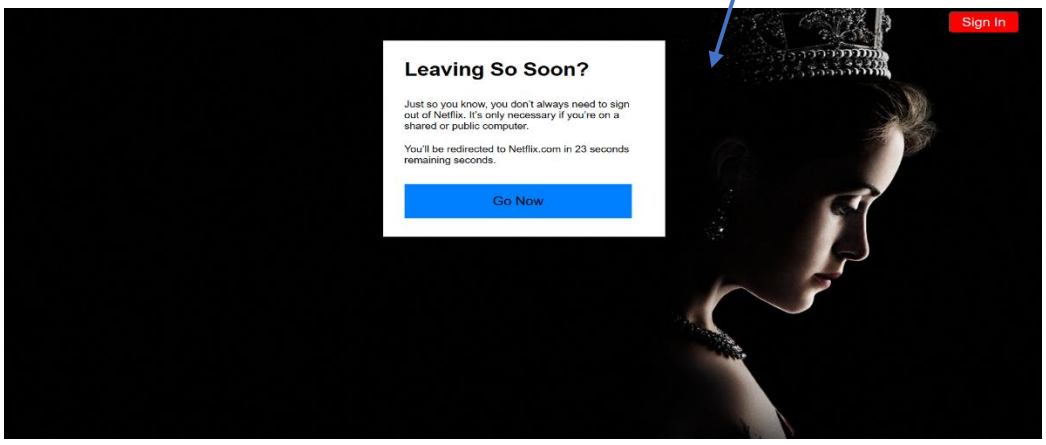




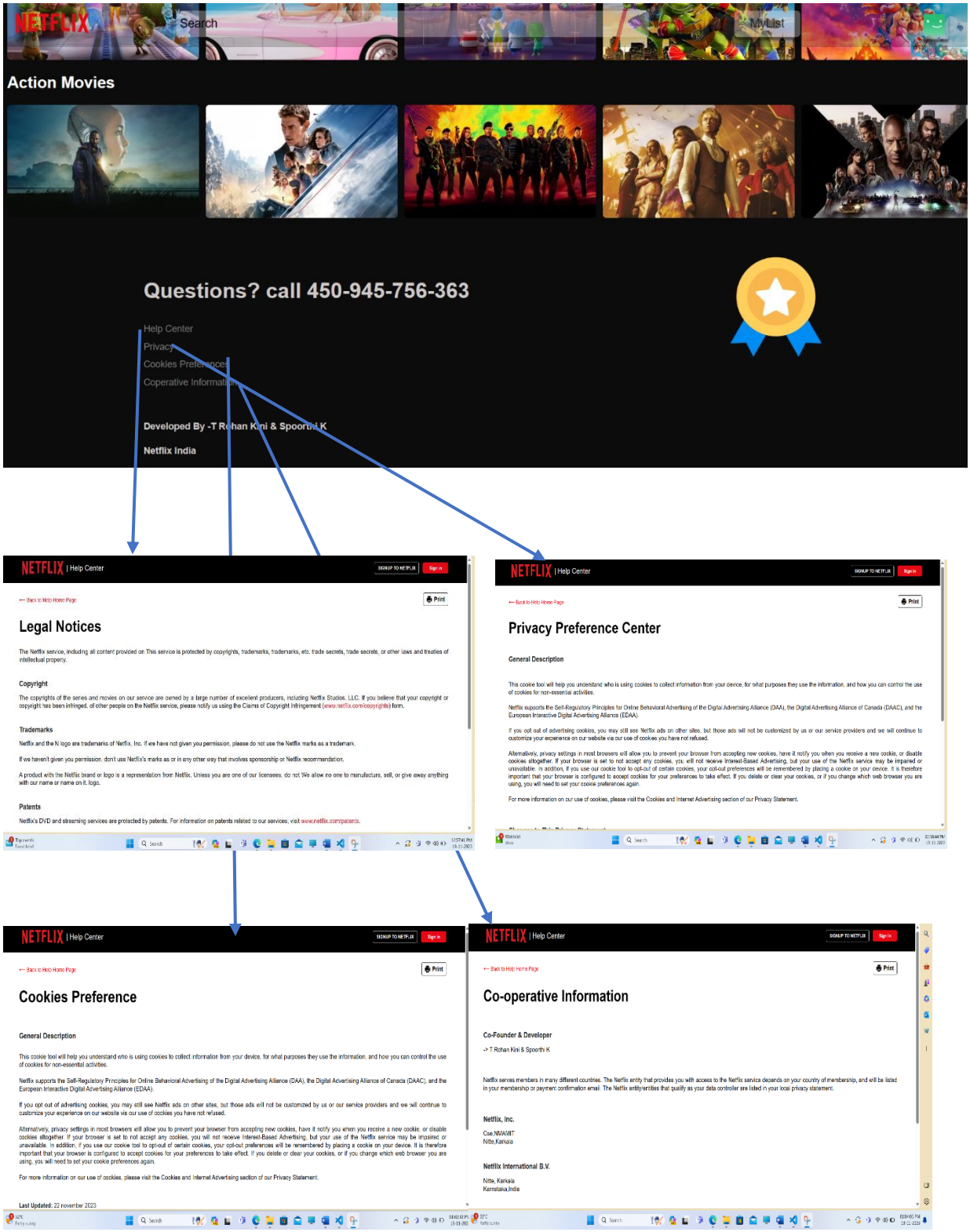
#below wishlist.html



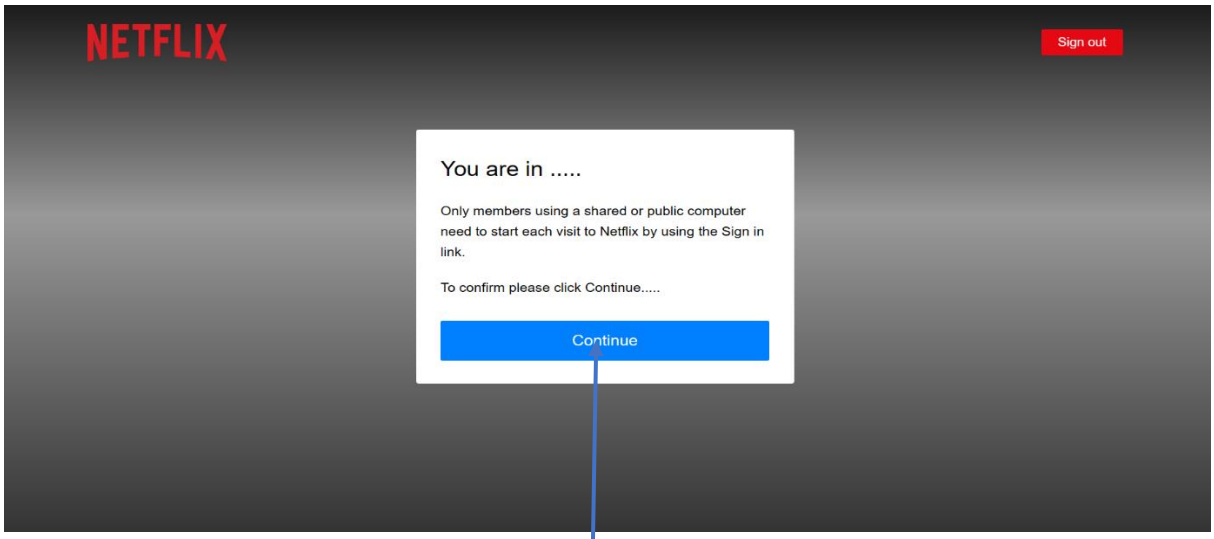
Signout.html



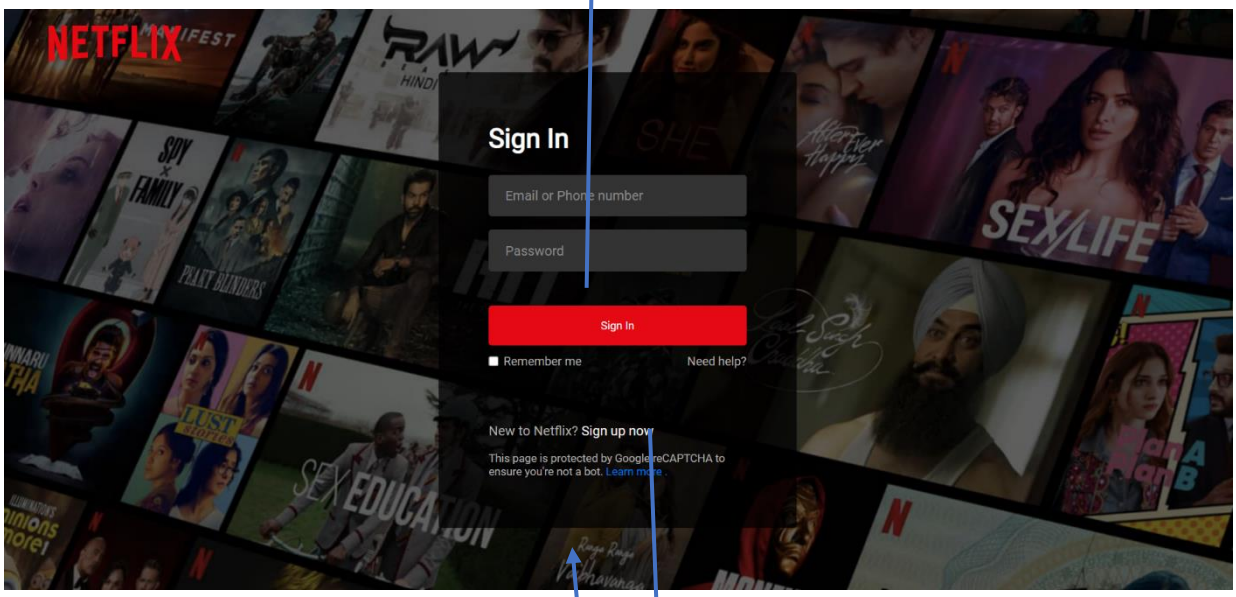
Netflix.html



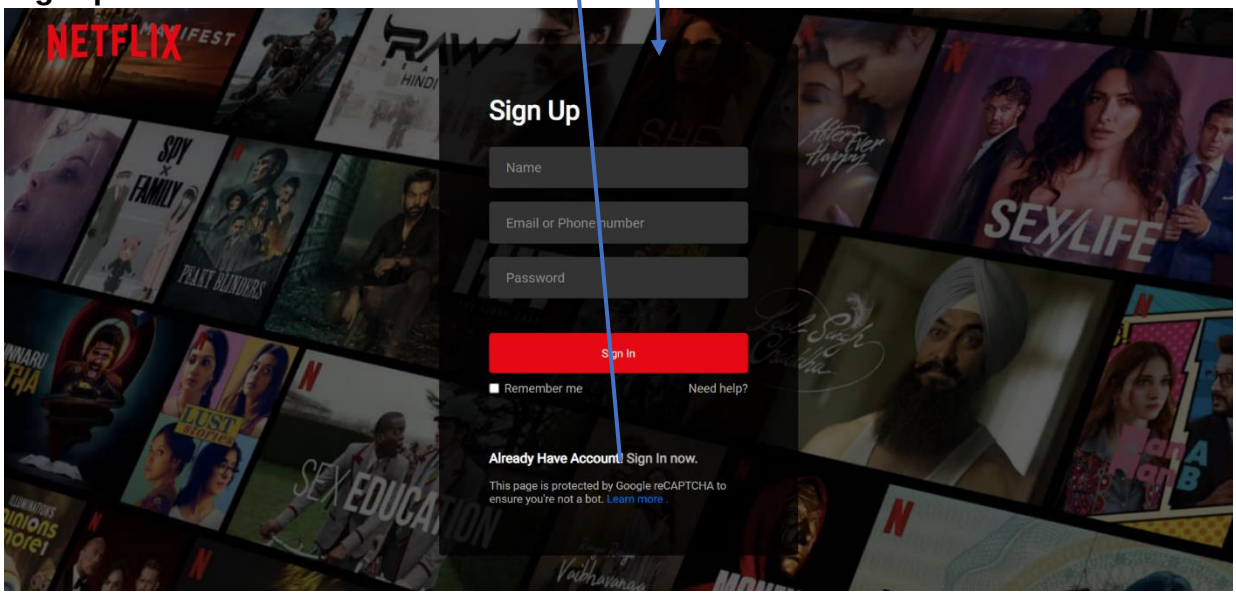
#below signinguin.html



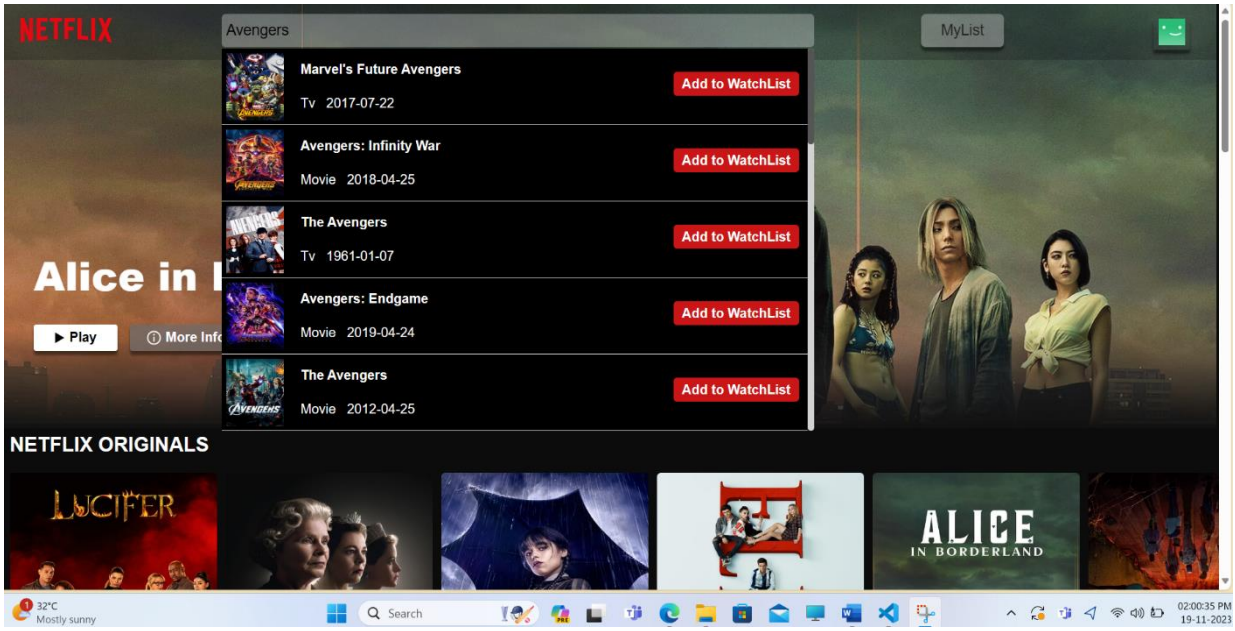
Login.html



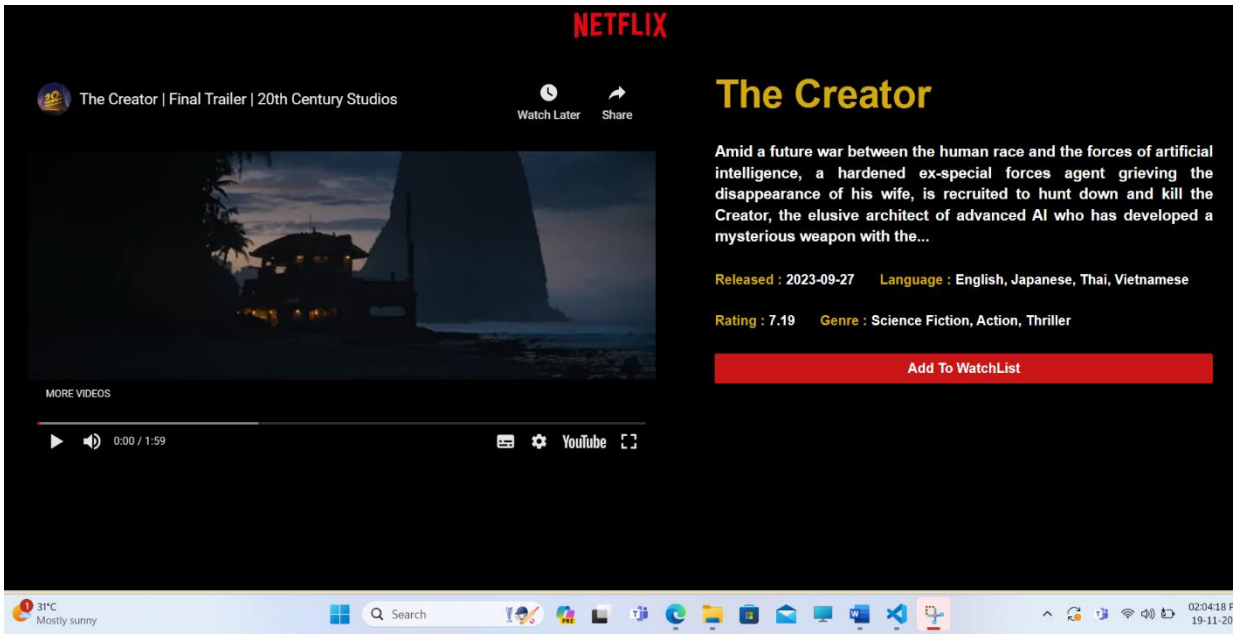
Signup.html



#search bar features



#We are showing Released date, Languages, Rating and Genre. As shown below:



3.2 Code Implements

Wishlist.html

```
const logo = document.querySelector('.logo');
const watchListItems = document.getElementById('watchList-Items');

// Function to display the list of WatchList
function showWatchListItems() {
    // Retrieve WatchList from local storage or use an empty array
    const storedWatchList = JSON.parse(localStorage.getItem('watchlist')) ||
[];

    if (storedWatchList.length === 0) {
        const emptyMessage = document.createElement('p');
        emptyMessage.textContent = "It's lonely here. Add some Movies  or Tv
shows to WatchList !";
        watchListItems.appendChild(emptyMessage);
    } else {
        storedWatchList.forEach(movie => {
            const shortenedTitle = movie.title || movie.name;
            const date = movie.release_date || movie.first_air_date;
            const watchList_Item = document.createElement('div');
            watchList_Item.classList.add('watchlist-item');
            watchList_Item.innerHTML = `<div class="search-item-thumbnail">
                    <img
src="https://image.tmdb.org/t/p/w500${movie.poster_path}">
                </div>
                <div class="search-item-info">
                    <h3>${shortenedTitle}</h3>
                    <h4>Year : ${date}</h4>
                </div>
                <button class="removeBtn" id="${movie.id}">Remove From
WatchList</button>`;
            watchListItems.appendChild(watchList_Item);

            // Add a click event listener to the remove button
            const removeBtn = watchList_Item.querySelector('.removeBtn');
            removeBtn.addEventListener('click', () =>
removeMovieFromWatchList(movie.id));

            // Add a click event listener to navigate to respestive movie
details page
            const thumbnail = watchList_Item.querySelector('.search-item-
thumbnail');
            thumbnail.addEventListener('click', () => {
                // Construct the URL for the movie details page with the TMDb
ID and Media_Type as a parameter
```

```
                const movieDetailsURL =
`../movie_details/movie_details.html?media=${movie.media_type}&id=${movie.id}`;
                window.location.href = movieDetailsURL;
            });
        });
    }
}

// Function to remove a movie from the WatchList
function removeMovieFromWatchList(movieId) {
    let storedWatchList = JSON.parse(localStorage.getItem('watchlist')) || [];

    // Find the index of the movie with the given ID in the stored array
    const movieIndex = storedWatchList.findIndex(movie => movie.id ===
movieId);

    if (movieIndex !== -1) {
        storedWatchList.splice(movieIndex, 1);

        // Update the local storage with the modified array
        localStorage.setItem('watchlist', JSON.stringify(storedWatchList));

        // Remove the corresponding DOM element
        const movieElement = document.getElementById(movieId);
        if (movieElement) {
            movieElement.parentElement.remove(); // Remove the entire movie
item
        }

        // If no movies/Tv shows are left, show the empty message
        if (storedWatchList.length === 0) {
            watchListItems.innerHTML = "";
            const emptyMessage = document.createElement('p');
            emptyMessage.textContent = "It's lonely here. Add some Movies  or
Tv shows to WatchList!";
            watchListItems.appendChild(emptyMessage);
        }
    }
}

// Add a window load event listener to show the WatchList when the page loads
window.addEventListener('load', () => {
    showWatchListItems();
});

// event listener to logo to navigate to the index page
logo.addEventListener('click', () => {
    window.location.href = '../netflix.html';});
```

moviedetils.html

```
const logo = document.querySelector('.logo');
logo.addEventListener('click', () => {
    window.location.href = '../index.html';
});

// Selecting various elements on the page for displaying movie details
const movieTitle = document.getElementById('movieTitle');
const moviePoster = document.getElementById('moviePoster');
const movieYear = document.getElementById('movieYear');
const rating = document.getElementById('rating');
const genre = document.getElementById('genre');
const plot = document.getElementById('plot');
const language = document.getElementById("language");
const iframe = document.getElementById("iframe");
const watchListBtn = document.querySelector('.watchListBtn');
const watchlist = JSON.parse(localStorage.getItem('watchlist')) || [];

// API key for TMDB API
const api_Key = '4626200399b08f9d04b72348e3625f15';

// Retrieve the TMDb ID and Media from the URL parameter
const params = new URLSearchParams(window.location.search);
const id = params.get('id');
const media = params.get("media");

// Function to fetch detailed information using its TMDb ID
async function fetchMovieDetails(id) {
    const response = await
fetch(`https://api.themoviedb.org/3/${media}/${id}?api_key=${api_Key}`);
    const data = await response.json();
    return data;
}

// Display the movie details on the page
async function displayMovieDetails() {
    try {
        const movieDetails = await fetchMovieDetails(id);

        var spokenlanguage = movieDetails.spoken_languages.map(language =>
language.english_name)
        language.textContent = spokenlanguage.join(', ');

        var genreNames = movieDetails.genres.map(genre => genre.name);
        genre.innerText = genreNames.join(', ');

        movieDetails.overview.length > 290
            ? plot.textContent = `${movieDetails.overview.substring(0,
290)}...`
```

```
            : plot.textContent = movieDetails.overview;

        movieTitle.textContent = movieDetails.name || movieDetails.title;
        moviePoster.src =
`https://image.tmdb.org/t/p/w500${movieDetails.backdrop_path}`;
        movieYear.textContent = `${movieDetails.release_date ||
movieDetails.first_air_date}`;
        rating.textContent = movieDetails.vote_average;

        // Updating the favorite button text and adding a click event listener
to toggle favorites
        if (watchlist.some(favoriteMovie => favoriteMovie.id ===
movieDetails.id)) {
            watchListBtn.textContent = "Remove From WatchList";
        } else {
            watchListBtn.textContent = "Add To WatchList";
        }
        watchListBtn.addEventListener('click', () =>
toggleFavorite(movieDetails));

    } catch (error) {
        movieTitle.textContent = "Details are not available right now! Please
try after some time."
    }

    try {
        const videoDetails = await fetchVideoDetails(id);
        const trailer = videoDetails.find(video => video.type === 'Trailer');
        if (trailer) {
            iframe.src =
`https://www.youtube.com/embed/${trailer.key}?autoplay=1`;
            moviePoster.style.display = "none";
        } else {
            iframe.style.display = "none";
        }
    } catch (error) {
        iframe.style.display = "none";
    }
}

// Function to toggle adding/removing from favorites
function toggleFavorite(movieDetails) {
    const index = watchlist.findIndex(movie => movie.id === movieDetails.id);
    if (index !== -1) {
        watchlist.splice(index, 1);
        watchListBtn.textContent = "Add To WatchList";
    } else {
        watchlist.push(movieDetails);
        watchListBtn.textContent = "Remove From WatchList";
```

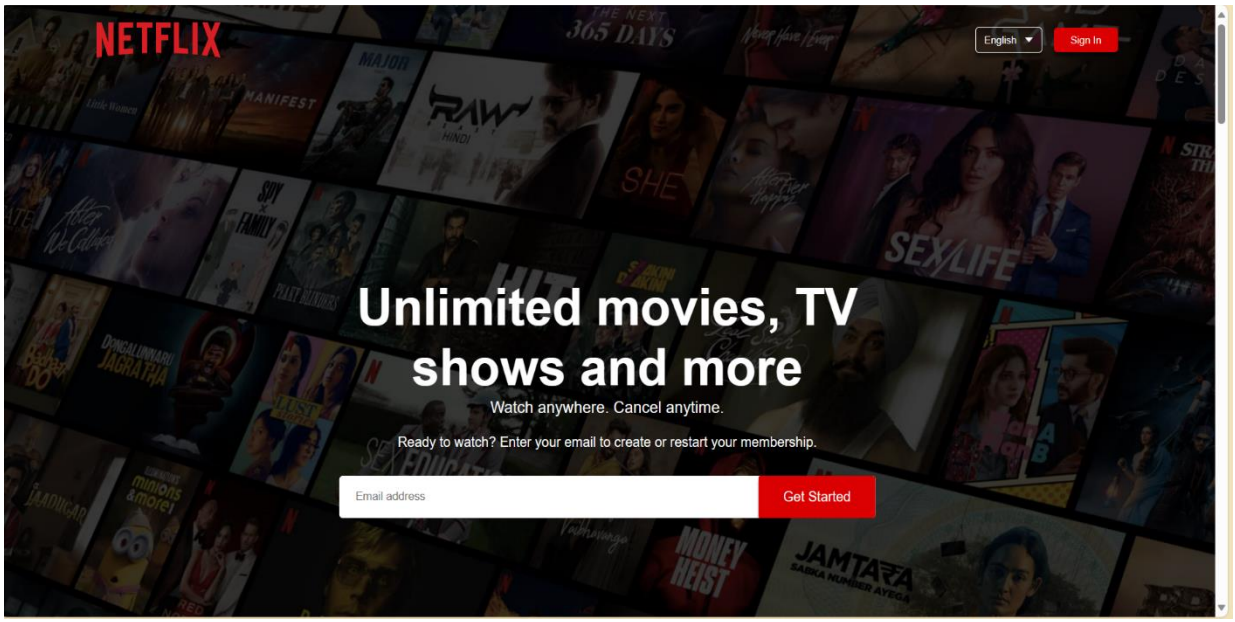
```
    }
    localStorage.setItem('watchlist', JSON.stringify(watchlist));
}

// Call the function to display movie details when the page loads
window.addEventListener('load', () => {
    displayMovieDetails();
});


// Function to fetch video details (trailers) for a movie or TV show
async function fetchVideoDetails(id) {
    const response = await
fetch(`https://api.themoviedb.org/3/${media}/${id}/videos?api_key=${api_Key}`);
    const data = await response.json();
    return data.results;
}
```

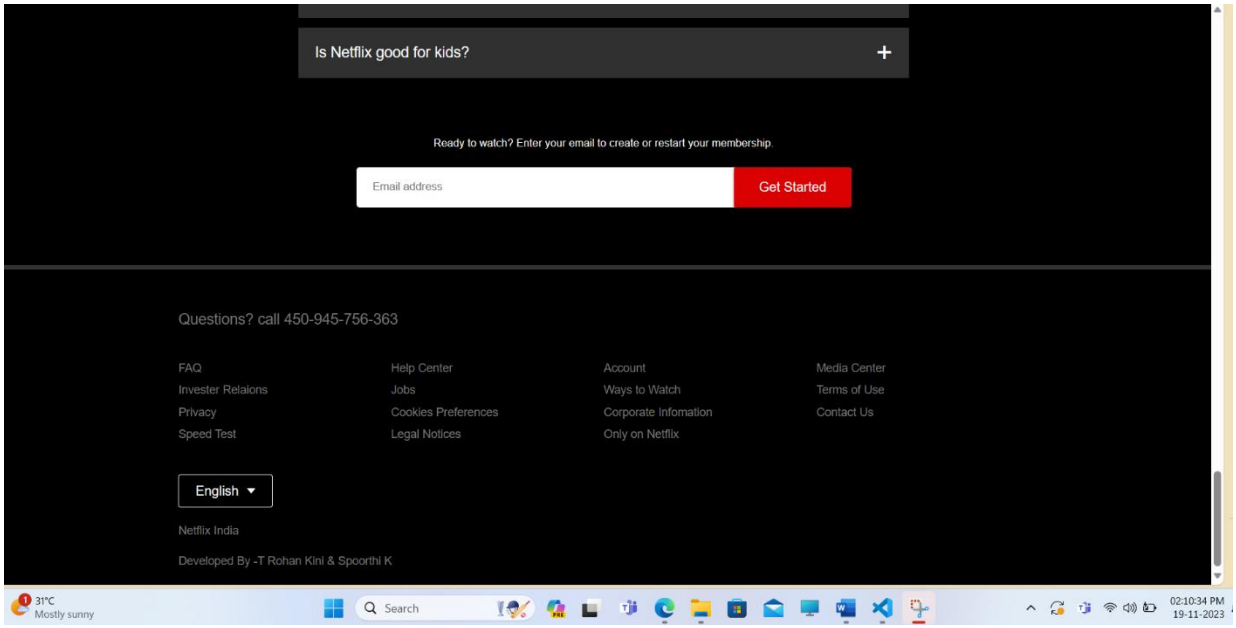
Api.js

```
import secret from '../secrets.json';
import Api from '../utils/api';

const API_BASE = 'https://api.themoviedb.org/3';
const API_KEY =  secret.TMDB_API_KEY;


const tmdb =  new Api({
    baseUrl: API_BASE,
    searchParams: { api_key: API_KEY },
})

export default tmdb;
```

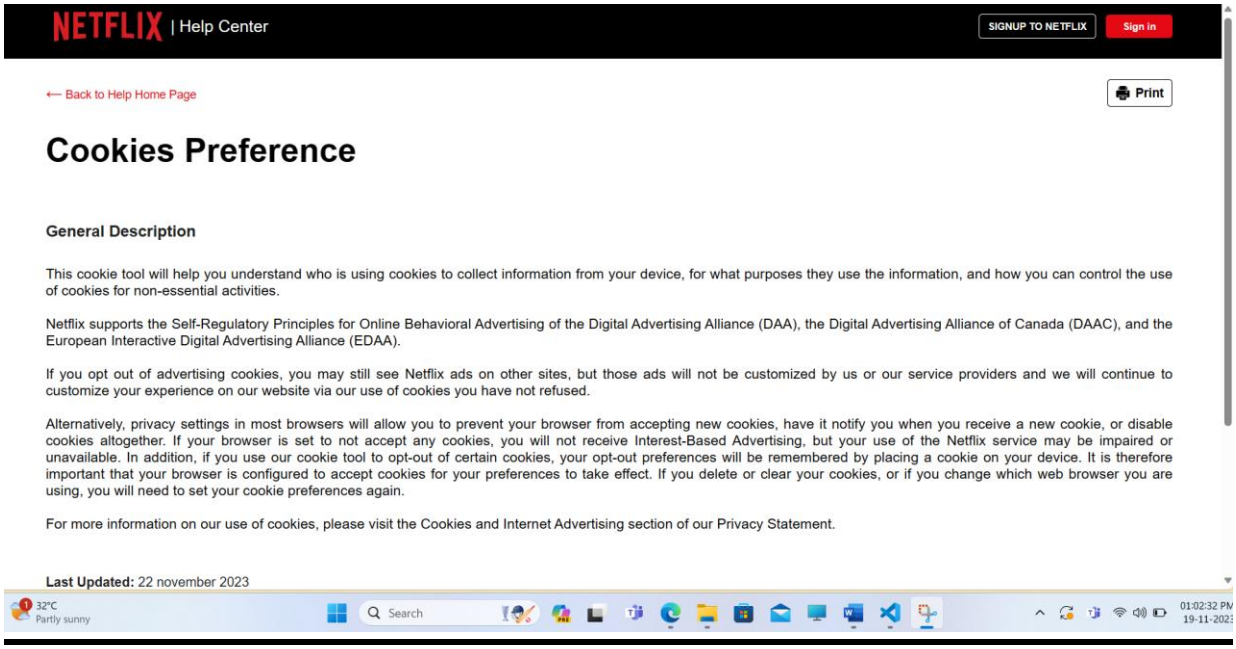
RESULT

Index.html

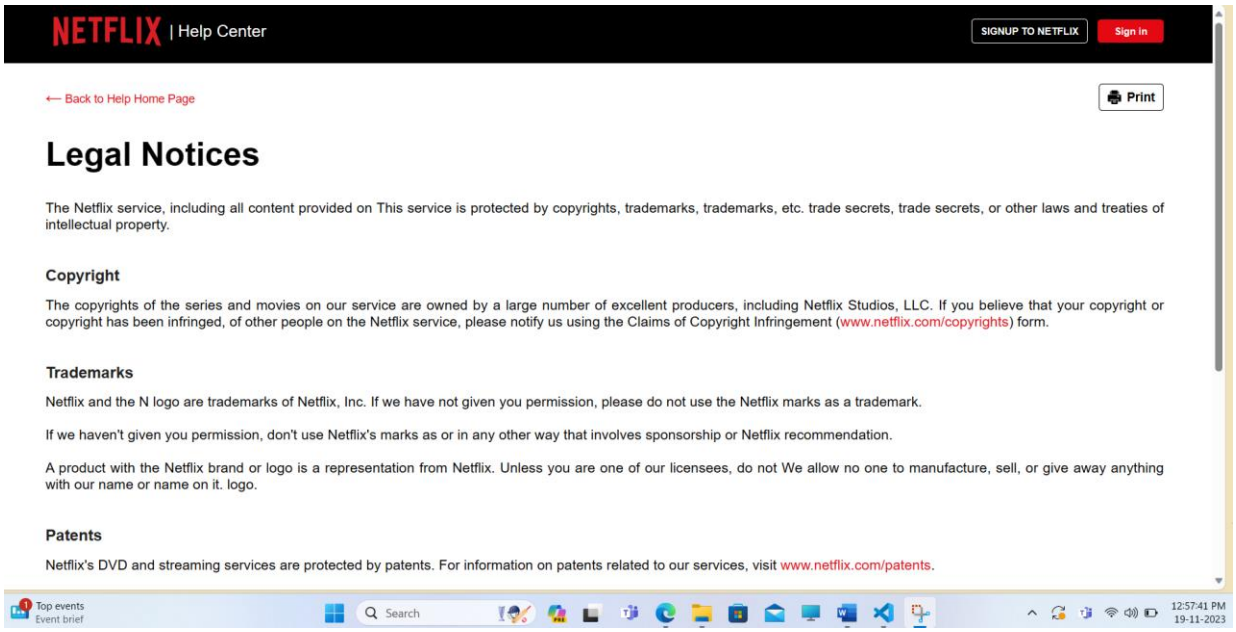




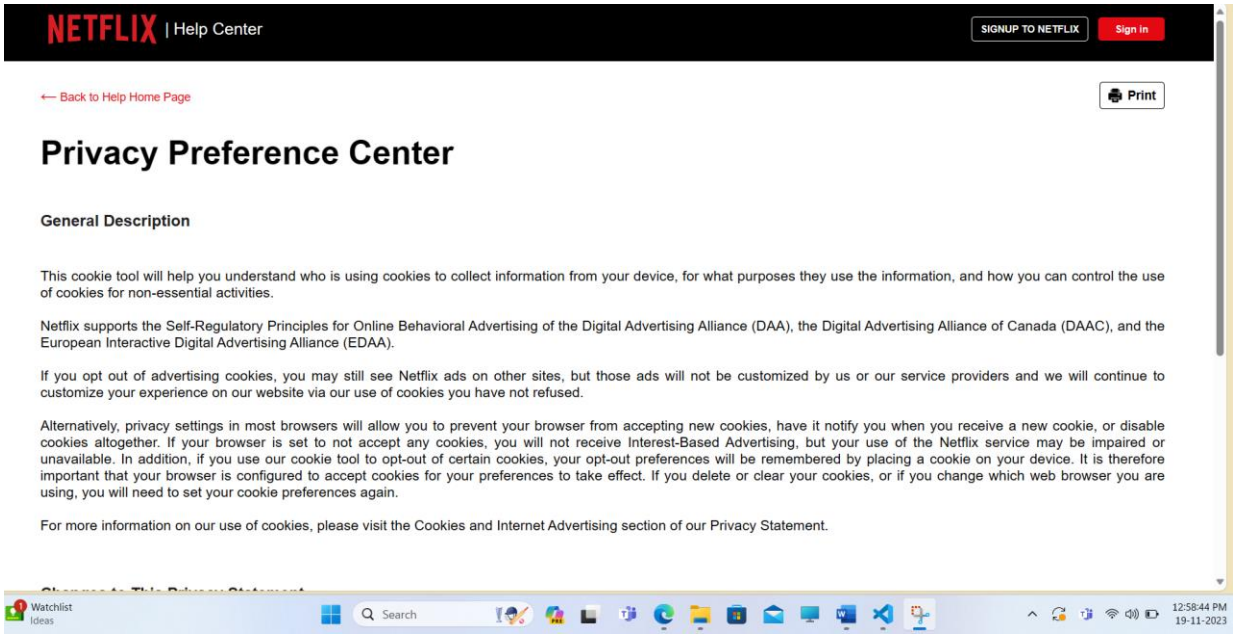
cookies_preference.html



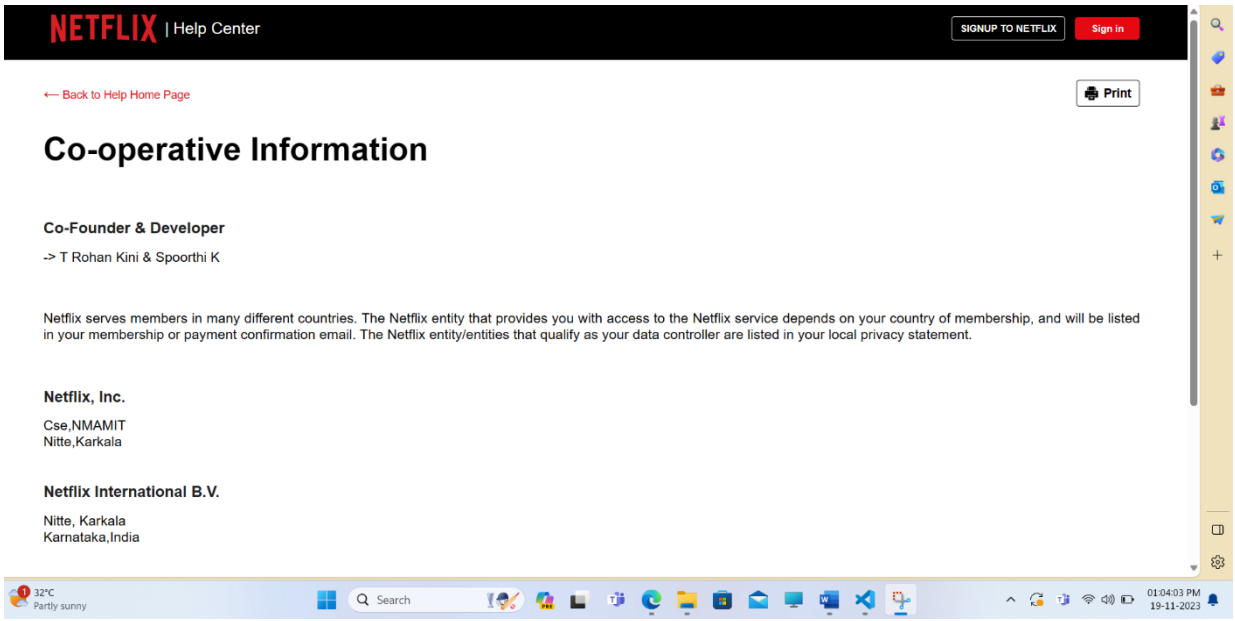
Legal-notices.html



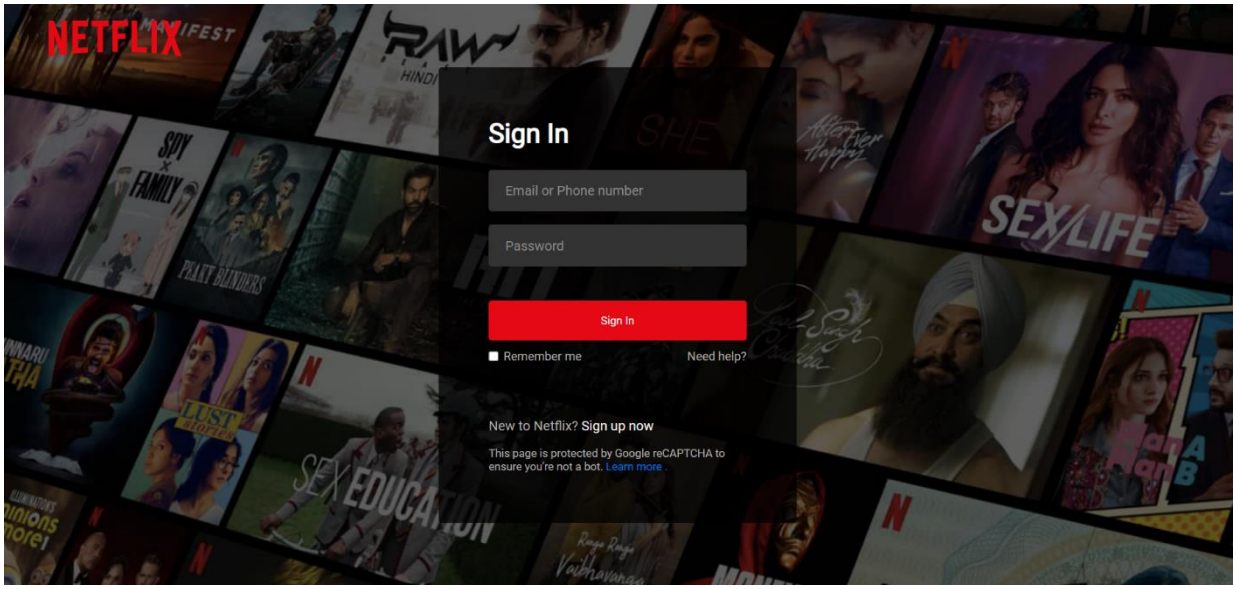
Privacy.html



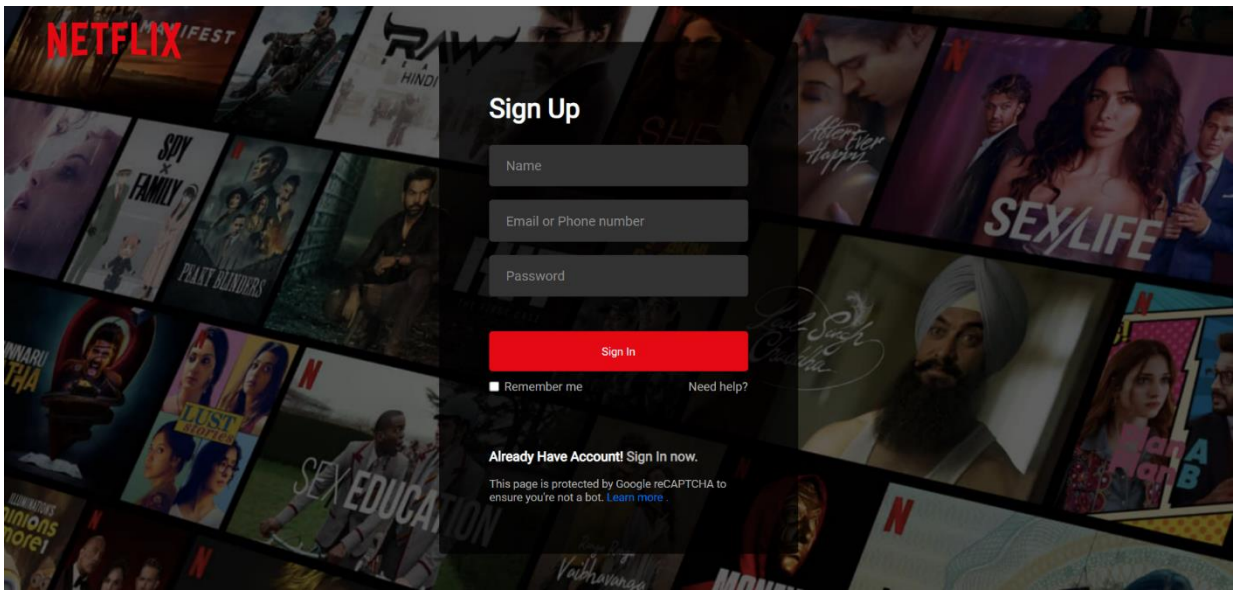
co_operative.html



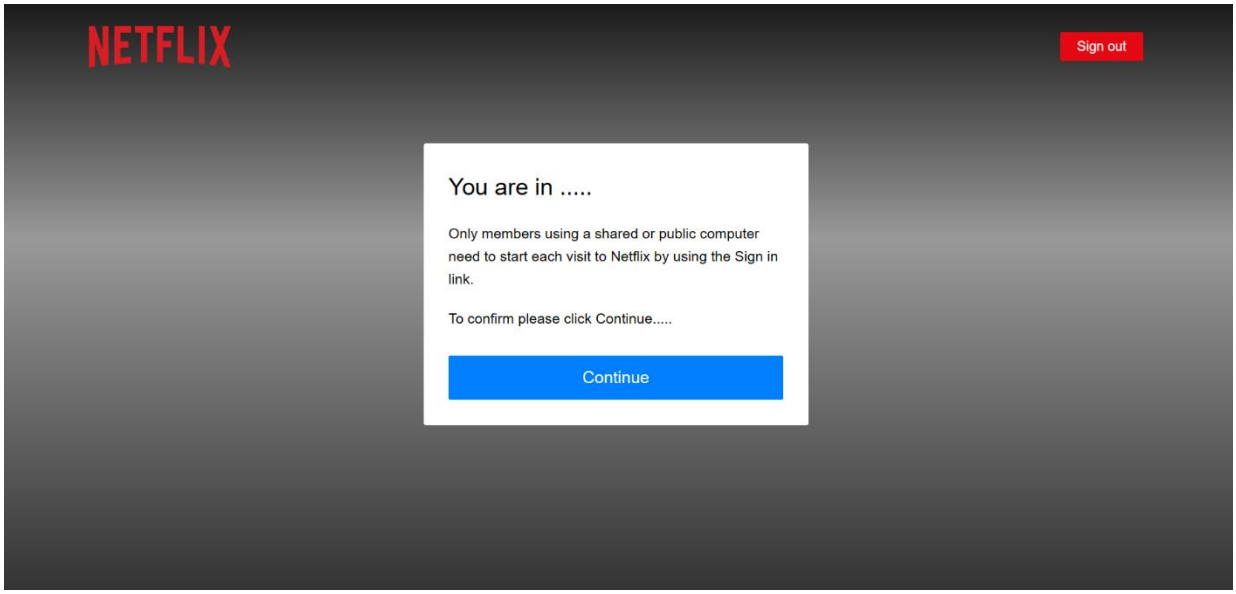
login.html



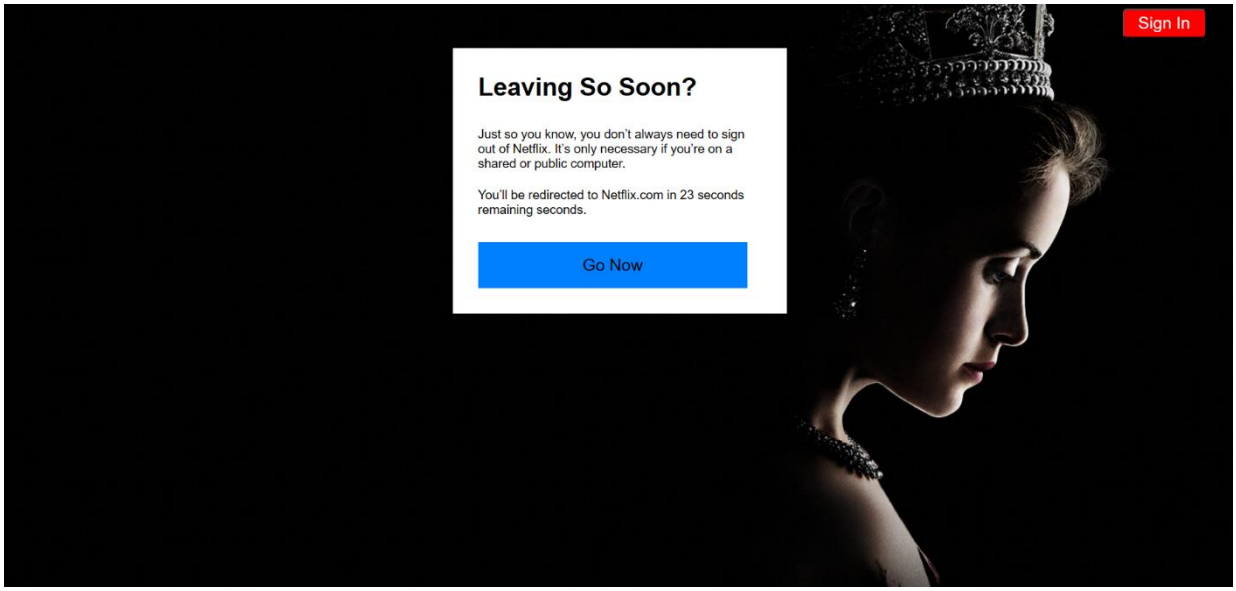
signup.html



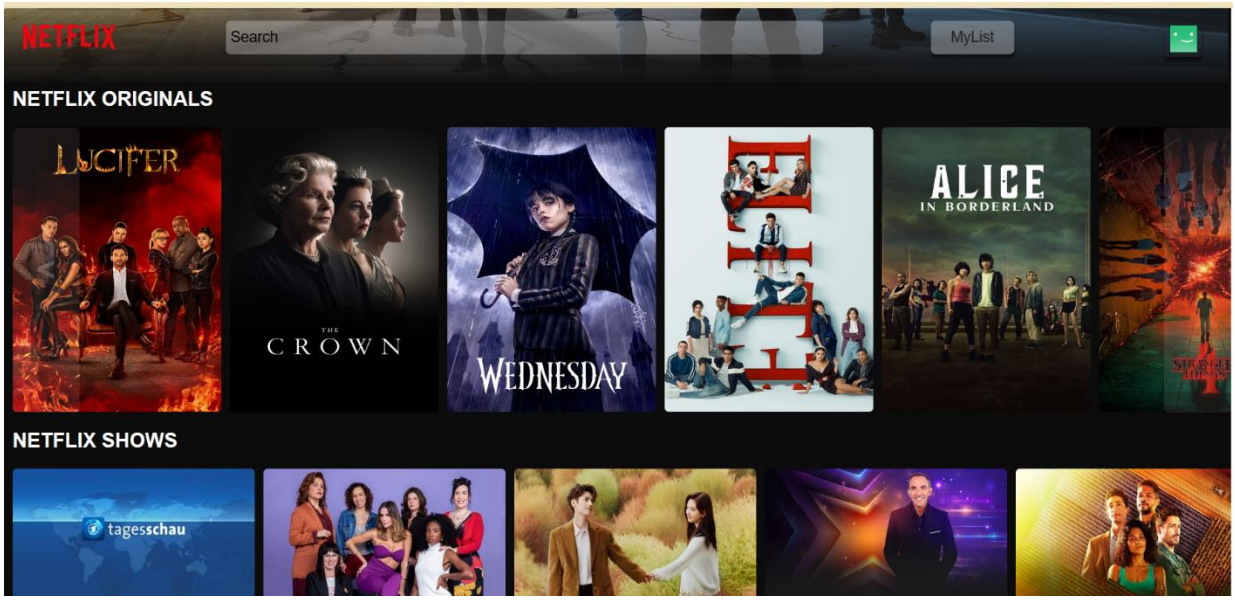
signinguin.html

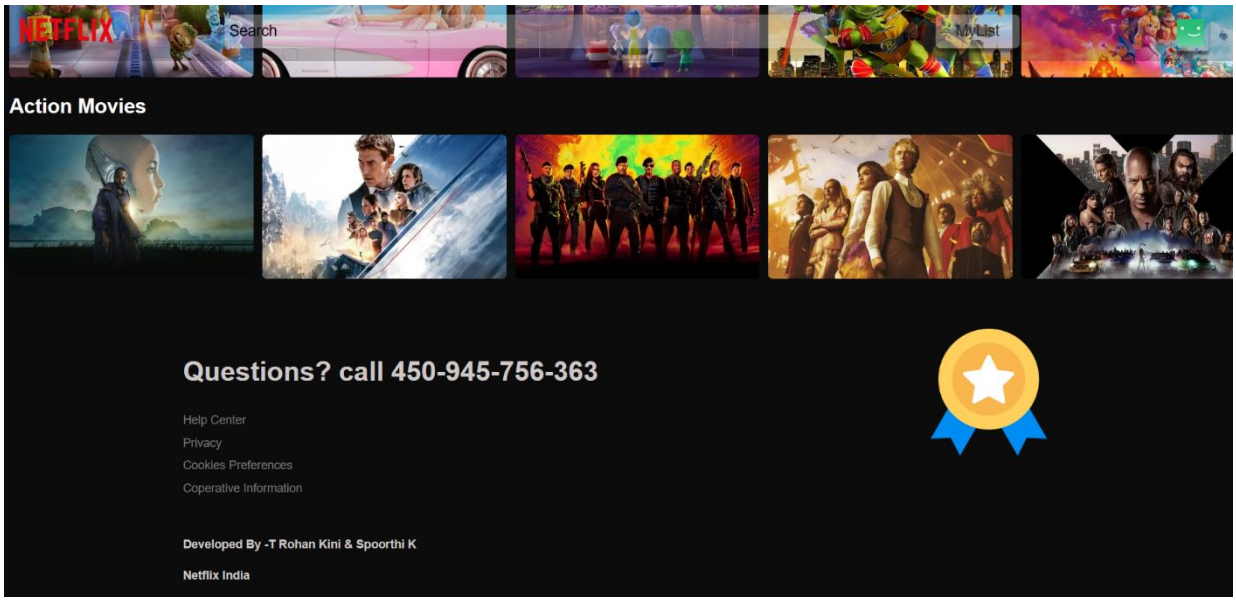


signout.html



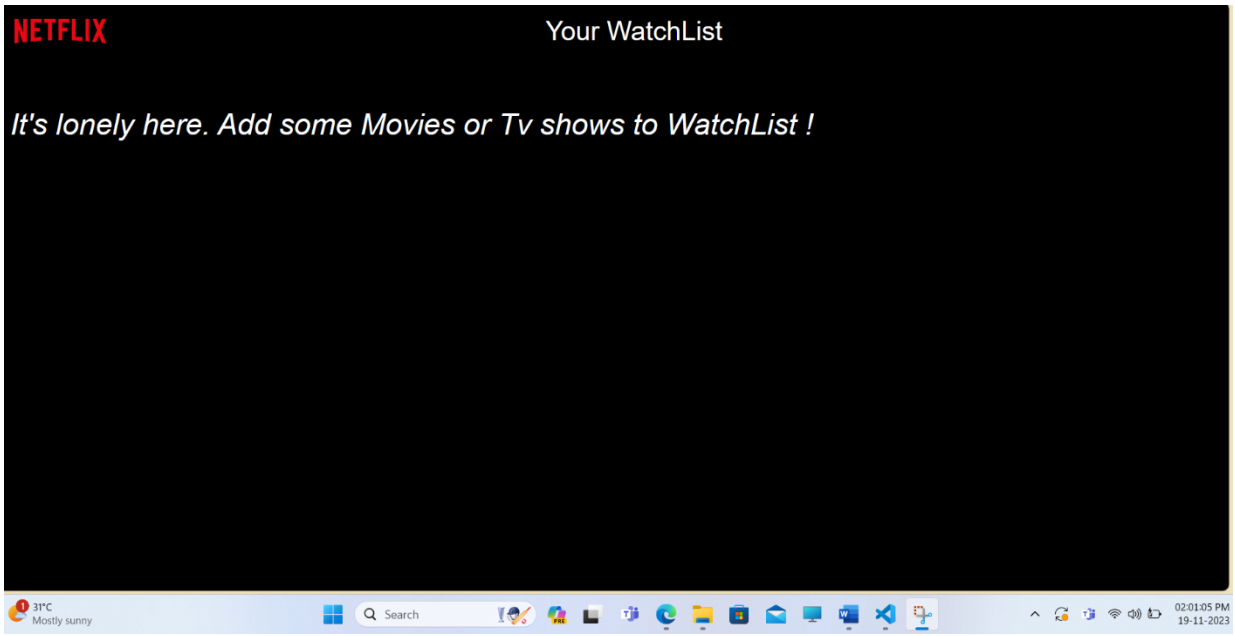
Netflix.html



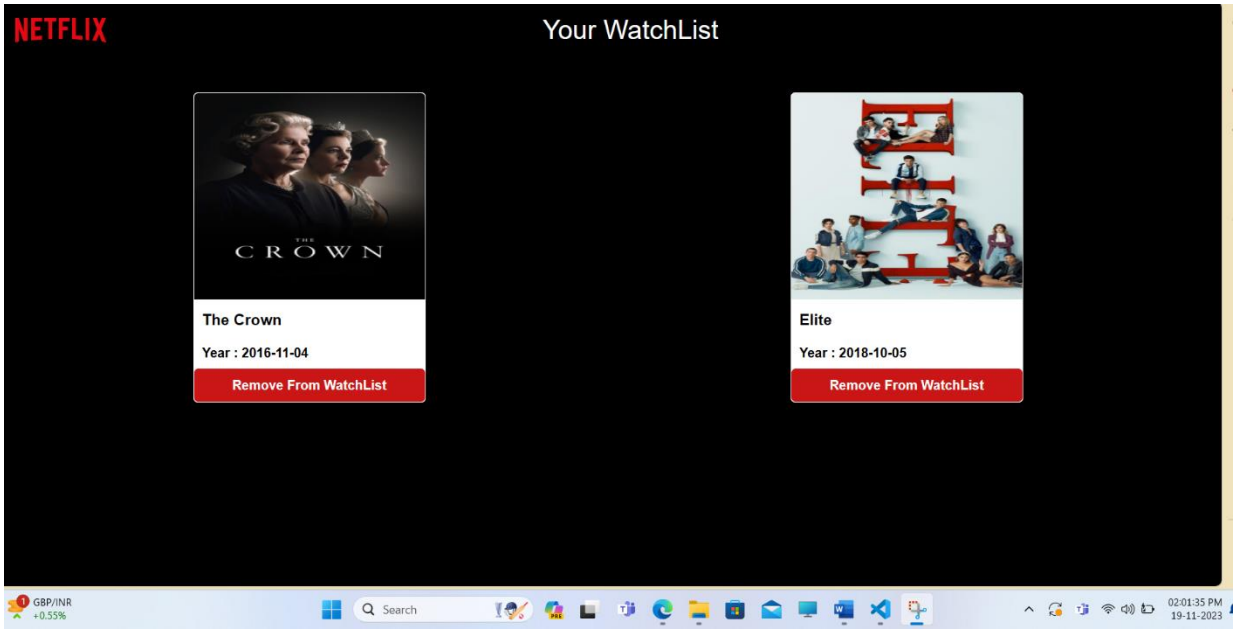


Movie_details.html

Wishlist.html
#empty wishlist



#set of movies in wishlist



CONCLUSION

Creating a Netflix clone using JavaScript and HTML is a great project that allows you to apply and enhance your web development skills. In conclusion, this project not only provides a hands-on experience in front-end development but also allows you to explore various aspects of web design, interactivity, and user experience. Here are some key points for conclusion:

1. **Learning Experience:**
    - Developing a Netflix clone involves working with HTML for structure and layout, and JavaScript for interactivity. This hands-on experience has likely deepened your understanding of these core web development technologies.
2. **Responsive Design:**
    - Ensuring that your Netflix clone is responsive is crucial for providing a seamless user experience across different devices. You may have explored and implemented responsive design principles, making your clone accessible on various screen sizes.
3. **Dynamic Content:**
    - Implementing dynamic content loading and organization is a key feature of a Netflix clone. This involves fetching and displaying data dynamically, simulating the real-time nature of a streaming platform.
4. **User Interface (UI) and User Experience (UX):**
    - Designing an intuitive and visually appealing user interface is essential for any web application. Consider discussing the design choices you made, such as layout, color schemes, and navigation, and how they contribute to a positive user experience.
5. **API Integration:**

- If you used APIs to fetch movie or series data, highlight how this integration adds real-world data to your application. Discuss any challenges you faced and how you overcame them.

6. **Challenges and Problem-Solving:**
    - Share any challenges you encountered during the development process and the solutions you implemented. This can be related to coding issues, design decisions, or any unexpected roadblocks.
7. **Future Improvements:**
    - Suggest potential improvements or features that could be added to enhance the functionality and user experience of your Netflix clone. This could include features like user authentication, personalized recommendations, or a more extensive content library.
8. **Conclusion:**
    - In conclusion, creating a Netflix clone using JavaScript and HTML is a rewarding project that combines technical skills with creativity. It provides valuable insights into modern web development practices and sets a foundation for more advanced projects in the future.

Git-hub link: Rohankini007/Netflix.fewd.miniproject: Netflix mini project-FEWD (github.com)

Host-link : https://rohankini007.github.io/Netflix.fewd.miniproject/

REFERENCES

- https://youtu.be/jp5j0AcaoLU?si=wyI-cEFqkpJLTyMh
- https://www.youtube.com/live/XtMThy8QKqU?si=U4eeM6fxZUxMA2f8
- https://youtu.be/U27R_0aNH0I?si=8cPZaJvBPFA6_TjT
- Create Netflix Clone Using HTML CSS & JavaScript - DEV Community