Project Report (web programming)

Student Name: SURAJ KUMAR **UID:** 24MCA20013
Branch: General MCA **Section/Group:** 3_B
Semester: 1st **Date of Performance:** 03/11/2024
Subject Name: Web Programming Lab **Subject Code:** 24 CAP-605

1. Aim of the project: Create a video streaming platform.

2. Algorithm/Flowchart:

Creating a Video Streaming Platform like Netflix involves several processes from the moment a user visits the website to streaming video content. Below is a high-level algorithm and an outline for the flowchart that you can use for your platform.

1. User Registration/Login

- Step 1: User visits the platform's homepage.
- Step 2: User is presented with options to log in or sign up.
- Step 3: If the user chooses to sign up, they must provide personal details (email, password, etc.).
- Step 4: If the user chooses log in, they provide credentials (email and password).
- Step 5: If the credentials are valid, the user is granted access to the platform.
- Step 6: If the credentials are invalid, display an error message.

2. Homepage/Content Discovery

- Step 1: After successful login, the user is directed to the homepage.
- Step 2: Homepage displays a personalized content feed based on user preferences, browsing history, or subscriptions.
- Step 3: Users can browse content via categories such as Trending, Genres, New Releases, or Recommended.
- Step 4: The system also provides a search functionality for users to find specific videos, TV shows, or movies.
- Step 5: The user can click on a video thumbnail to view its details (description, rating, etc.).

3. Video Playback

- Step 1: After the user selects a video, the system checks if the user has the required subscription level or if the video is available in their region.
- Step 2: If eligible, the video player is loaded.
- Step 3: The video player buffers the content by fetching data from the server and starts the playback.
- Step 4: The user can control playback via play/pause, volume control, and fullscreen options.
- Step 5: The system continuously monitors the network speed and adjusts video quality (e.g., HD, 720p, 1080p) for seamless streaming.
- Step 6: If the user chooses to exit the video or reach the end of the content, the system suggests similar videos or returns the user to the homepage or previous screen.

4. Subscription/Payment Process

- Step 1: The user is prompted to choose a subscription plan (e.g., Basic, Standard, Premium).
- Step 2: The user enters payment details (credit card, PayPal, etc.).
- Step 3: If payment is successful, the system activates the subscription and grants access to premium content.
- Step 4: If payment fails, the user is notified, and the process is aborted.

5. Recommendations System

- Step 1: As the user watches content, the system records user activity (watched shows, genres, ratings).
- Step 2: The system applies a recommendation algorithm to suggest new content based on viewing habits (collaborative filtering, content-based filtering, or hybrid models).
- Step 3: Personalized recommendations are shown on the homepage or as part of the video details page.

6. Logout

- Step 1: The user chooses to log out.
- Step 2: The system logs the user out and redirects to the homepage or login page.

3. Code for the Project :

HTML:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link rel="stylesheet" href="style.css" />
    <title>Movie App</title>
  </head>
  <body>
    <header>
      <form id="form">
        <input type="text" id="search" class="search" placeholder="Search">
      </form>
    </header>
    <main id="main"></main>
    <script src="script.js"></script>
  </body>
</html>
```

CSS:

```
@import url('https://fonts.googleapis.com/css2?family=Poppins:wght@200;400&display=swap');
:root {
  --primary-color: #22254b;
  --secondary-color: #373b69;
}

* {
  box-sizing: border-box;
}

body {
  background-color: var(--primary-color);
  font-family: 'Poppins', sans-serif;
  margin: 0;
}

header {
  padding: 1rem;
  display: flex;
  justify-content: flex-end;
  background-color: var(--secondary-color);
}

.search {
  background-color: transparent;
  border: 2px solid var(--primary-color);
  border-radius: 50px;
  font-family: inherit;
  font-size: 1rem;
  padding: 0.5rem 1rem;
  color: #fff;

}

.search::placeholder {
  color: #7378c5;
}

.search:focus {
  outline: none;
  background-color: var(--primary-color);
}

main {
  display: flex;
  flex-wrap: wrap;
  justify-content: center;
}

.movie {
  width: 300px;
  margin: 1rem;
  background-color: var(--secondary-color);
  box-shadow: 0 4px 5px rgba(0, 0, 0, 0.2);
  position: relative;
  overflow: hidden;
  border-radius: 3px;
}

.movie img {
  width: 100%;
}
```

```
.movie-info {
  color: #eee;
  display: flex;
  align-items: center;
  justify-content: space-between;
  gap:0.2rem;
  padding: 0.5rem 1rem 1rem;
  letter-spacing: 0.5px;
}

.movie-info h3 {
  margin-top: 0;
}

.movie-info span {
  background-color: var(--primary-color);
  padding: 0.25rem 0.5rem;
  border-radius: 3px;
  font-weight: bold;
}

.movie-info span.green {
  color: lightgreen;
}

.movie-info span.orange {
  color: orange;
}

.movie-info span.red {
  color: red;
}

.overview {
  background-color: #fff;
  padding: 2rem;
  position: absolute;
  left: 0;
  bottom: 0;
  right: 0;
  max-height: 100%;
  transform: translateY(101%);
  overflow-y: auto;
  transition: transform 0.3s ease-in;
}

.movie:hover .overview {
  transform: translateY(0);
}
```

JS:

```
const API_URL =
'https://api.themoviedb.org/3/discover/movie?sort_by=popularity.desc&api_key=3fd2be6f0c70a2a598f084ddfb75487c&page=1'
const IMG_PATH = 'https://image.tmdb.org/t/p/w1280'
const SEARCH_API = 'https://api.themoviedb.org/3/search/movie?api_key=3fd2be6f0c70a2a598f084ddfb75487c&query="'

const main = document.getElementById('main')
const form = document.getElementById('form')
const search = document.getElementById('search')

// Get initial movies
getMovies(API_URL)

async function getMovies(url) {
  const res = await fetch(url)
  const data = await res.json()

  showMovies(data.results)
}

function showMovies(movies) {
  main.innerHTML = ''

  movies.forEach((movie) => {
    const { title, poster_path, vote_average, overview } = movie

    const movieEl = document.createElement('div')
    movieEl.classList.add('movie')

    movieEl.innerHTML = `
      <img src="${IMG_PATH + poster_path}" alt="${title}">
      <div class="movie-info">
     <h3>${title}</h3>
     <span class="${getClassByRate(vote_average)}">${vote_average}</span>
      </div>
      <div class="overview">
     <h3>Overview</h3>
     ${overview}
    </div>
    `
    main.appendChild(movieEl)
  })
}

function getClassByRate(vote) {
  if(vote >= 8) {
    return 'green'
  } else if(vote >= 5) {
    return 'orange'
  } else {
    return 'red'
  }
}
```

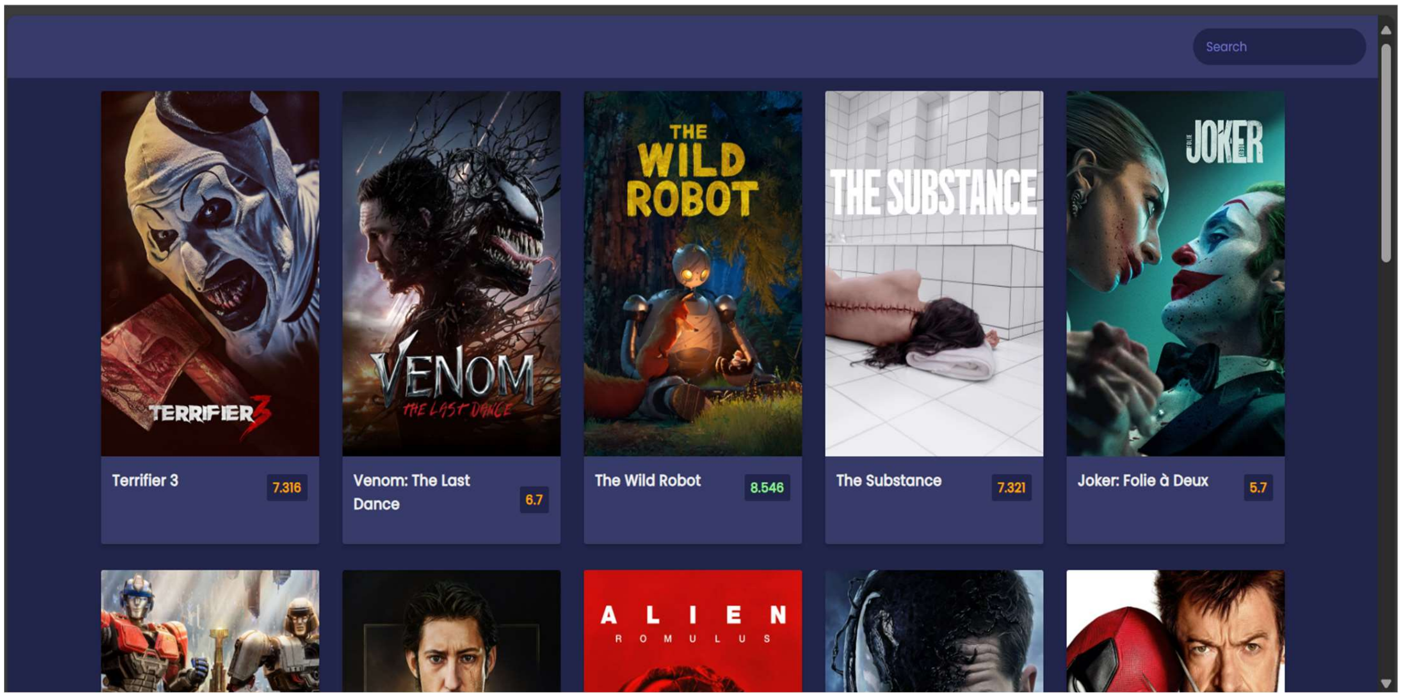
```
form.addEventListener('submit', (e) => {
    e.preventDefault()

    const searchTerm = search.value

    if(searchTerm && searchTerm !== '') {
        getMovies(SEARCH_API + searchTerm)

        search.value = ''
    } else {
        window.location.reload()
    }
})
```

4. Output:



5. Learning Outcomes:

- Learn how to structure and style web pages, creating a responsive and dynamic user interface for web applications.
- Gain proficiency in client-side scripting for interactivity, such as implementing video playback controls (play, pause, skip, volume control).
- Understand how to use frameworks like **React** or **Vue.js** to build efficient and dynamic front-end interfaces, and how to leverage libraries like **Axios** or **Fetch** for API calls.
- Learn to implement **responsive web design** to ensure the application works smoothly on a variety of devices (e.g., smartphones, tablets, and desktops).
- Gain experience with server-side programming languages such as **Node.js**, **Python (Flask/Django)**, or **Ruby on Rails** for handling requests, serving video content, and managing user authentication.