



**MIT Art, Design and Technology  
University**

**MIT School of Computing, Pune**

**Department of Information Technology**

# **Lab Manual**

**Practical - Web Programming**

**Class - S.Y. (SEM-IV), DA**

**Batch - DA-II**

**Name of the Student**

**Mr. Shridhar Barde**

**A.Y. 2024 – 2025 (SEM-IV)**



Web Programming			
SEMESTER – IV			
Course Code:	23IT2008	Course Credits:	02
Teaching Hours / Week (L:T:P):	0:0:4	CA Marks:	25
Total Number of Teaching Hours:		END-SEM Marks:	25
<b>Course Pre-requisites:</b>			
<b>Course Description:</b>  <p>This course provides a comprehensive introduction to web technology, designed to help students develop a strong foundation in building and managing websites and web applications. The curriculum covers key topics such as HTML, CSS, and JavaScript, PHP, MySQL, which are essential for creating interactive, well-designed web pages. Students will also explore the principles of responsive design, ensuring that web applications are optimized for different devices and screen sizes.</p> <p>The course dives deeper into server-side technologies, including HTTP, web servers, and databases, allowing students to understand how websites function behind the scenes. Emphasis is placed on practical learning, and students will gain hands-on experience by working on projects that showcase their ability to design, develop, and deploy websites.</p> <p>By the end of the course, students will be proficient in using modern web technologies to create web applications. They will understand how to handle client-server interactions, manage user data, and implement various web technologies to enhance the functionality of their applications.</p>			
<b>Course Learning Objectives:</b> This course will enable the students to: <ol style="list-style-type: none"> <li>1. Understand fundamental concepts of front-end web development.</li> <li>2. Enable students to create basic web pages incorporating essential elements such as images, hyperlinks, lists, tables, and forms.</li> <li>3. Teach students how to use CSS to manage fonts, lists, colors, text alignment, and background images for a cohesive and aesthetically pleasing web design.</li> <li>4. Develop an understanding of JavaScript scopes to manage the visibility and lifetime of variables and functions effectively.</li> <li>5. Equip students with the skills to implement and handle JavaScript events, enabling enhanced user interactions through event-driven programming.</li> <li>6. Apply comprehensive knowledge of HTML, CSS, and JavaScript to develop a complete front-end application. Utilize project-based learning to showcase problem-solving skills and creativity in web development projects.</li> <li>7. Configure server environments with Apache/TOMCAT.</li> <li>8. Set up a PHP development environment and write basic PHP scripts.</li> <li>9. Master PHP programming constructs for web development tasks.</li> <li>10. Create and process HTML forms, and manage MySQL database operations.</li> <li>11. Develop comprehensive back-end applications using PHP and MySQL.</li> </ol>			

<b>Course Outcome:</b> After taking this course, Students will be able to :		
<ol style="list-style-type: none"> <li>1. Apply knowledge of HTML to create the structure of the webpage and CSS to style and layout the elements, making the application visually appealing.</li> <li>2. Apply comprehensive knowledge of HTML, CSS, and JavaScript to develop a complete front-end application and utilize project-based learning to showcase problem-solving skills and creativity in web development projects.</li> <li>3. Set up and configure a server environment using tools like Apache or TOMCAT and set up a PHP development environment. Write &amp; execute simple PHP scripts, understanding PHP syntax and basic features, create HTML forms to collect user data and integrate with PHP for processing.</li> <li>4. Design and develop a back-end application using PHP and MySQL, implementing CRUD operations to manage data effectively.</li> </ol>		
<b>UNIT – I</b>	<b>Introduction to HTML and Cascading Style Sheet</b>	<b>09 Hours</b>
Module 1 - Markup Language (HTML): Introduction to HTML, Formatting and Fonts, Commenting Code, Anchors, Backgrounds, Images, Hyperlinks, Lists, Tables, Frames, HTML Forms  Module 2 - CSS: Need for CSS, introduction to CSS, basic syntax and structure, Levels of style sheets, Style specification formats, BOX Model, Selector forms, Property value forms, Font properties, List properties, Color, Alignment of text, Background images		
<b>Pedagogy</b>	<b>ICT Teaching / PowerPoint Presentation and Videos:</b>  <b>Use tools like Visual Studio Code (free).</b>  <b>Videos:</b>  <a href="https://www.coursera.org/learn/html-css-javascript-for-web-developers">https://www.coursera.org/learn/html-css-javascript-for-web-developers</a>	
	<b>Self-study / Do it yourself /:</b>  <b>Practice creating basic HTML pages and enhancing them using CSS.</b>	
	<b>Experiential Learning Topics:</b>  <b>Design a simple webpage for coffee shop website</b>	
	<b>PBL - Project Based Learning:</b>  <b>Create a multi-page website (e.g., coffee shop website) using HTML and CSS.</b>	
<b>UNIT – II</b>	<b>Front-End Development</b>	<b>09 Hours</b>
Module 3 - Overview of JavaScript, including JS in an HTML (Embedded, External), Basic JS syntax, basic interaction with HTML  Module 4 - Core features of JavaScript: Data types, Control Structures, Arrays, Functions and Scopes		
<b>Pedagogy</b>	<b>ICT Teaching / PowerPoint Presentation and Videos:</b>  <b>Use tools like Visual Studio Code (free).</b>	

	<b>Videos:</b>  <a href="https://www.coursera.org/learn/javascript-basics">https://www.coursera.org/learn/javascript-basics</a>	
	<b>Self-study / Do it yourself /:</b>  Solve exercises on JavaScript syntax, control structures, and functions	
	<b>Experiential Learning Topics:</b>  Build a web page with interactive elements (e.g., a simple calculator).	
	<b>PBL - Project Based Learning:</b>  Develop an interactive webpage that uses JavaScript to validate form inputs or perform basic calculations.	
<b>UNIT – III</b>	<b>Advanced Front-End Development</b>	<b>09 Hours</b>
Module 5 - DOM: DOM levels, DOM Objects and their properties and methods, Manipulating DOM		
Module 6 - JavaScript Events: JavaScript Events, Types of JavaScript Events, Objects in JS, Event Handling		
<b>Pedagogy</b>	<b>ICT Teaching / PowerPoint Presentation and Videos:</b>  <a href="https://www.coursera.org/learn/building-interactive-web-pages-using-javascript">https://www.coursera.org/learn/building-interactive-web-pages-using-javascript</a>  Use tools like Visual Studio Code (free).	
	<b>Self-study / Do it yourself /:</b>  Practice exercises on DOM traversal and event handling.	
	<b>Experiential Learning Topics:</b>  Add dynamic behavior to a webpage using DOM and events (e.g., a to-do list app).	
	<b>PBL - Project Based Learning:</b>  Develop a web page with dynamic content (e.g., a task manager or interactive quiz) using DOM manipulation and event handling.	
<b>UNIT – IV</b>	<b>Server Side Scripting</b>	<b>09 Hours</b>
Module 7 - Set up and configure a server environment using tools like Apache or TOMCAT, set up a PHP development environment.		
Module 8 -Introduction to PHP: : Introduction to PHP, Server side scripting Vs Client side scripting, Basic Development Concepts (Mixing PHP with HTML), Creating, Writing & Running First PHP Script, PHP syntax, conditions & Loops, Functions, String manipulation, Arrays & Functions,		
Module 9 - Form handling with HTML and PHP: Designing of Forms using HTML, Form Handling using GET		

and POST methods of Form

Pedagogy	ICT Teaching / PowerPoint Presentation and Videos:  <a href="https://www.coursera.org/learn/web-applications-php">https://www.coursera.org/learn/web-applications-php</a>  Use tools like Visual Studio Code (free), XAMPP/WAMP for PHP server setup, and MySQL Workbench for database management	
	Self-study / Do it yourself /:  Practice exercises on form handling and server-side scripting with PHP.	
	Experiential Learning Topics:  Create a basic form for data submission and handle it using PHP (e.g., feedback form).	
	PBL - Project Based Learning:  Develop a small server-side application (e.g., a contact form with email validation and submission).	
UNIT – V	Working with Databases and Web Application Development	09 Hours
Module 10 - Working with databases using MySQL with PHP: MySQL database, create database, create table, primary key with AUTO_INCREMENT setting, Insert Data Into a Database Table, Select Data From a Database Table, Open or close a Connection to the MySQL Server.		
Module 11 - Web Application Development (Project): Develop the web application to handle client-server interactions, manage user data, and implement various web technologies to enhance the functionality of their applications. Example: Website for a Coffee Shop		
Pedagogy	ICT Teaching / PowerPoint Presentation and Videos:  Use tools like Visual Studio Code (free), XAMPP/WAMP for PHP server setup, and MySQL Workbench for database management  Videos:  <a href="https://www.coursera.org/learn/web-app">https://www.coursera.org/learn/web-app</a>	
	Self-study / Do it yourself /:  Exercises on creating and manipulating databases using PHP and MySQL.	
	Experiential Learning Topics:  Create a database and design a webpage to display its data dynamically.	
	PBL - Project Based Learning:	

	<b>Develop a fully functional web application (e.g., a Coffee Shop website or e-commerce platform) that integrates database functionality for data management.</b>
--	--

# Experiment No.1

## 1. Problem Statement

**Create the basic structure of an online movie shopping platform website, including a clean home page layout with a header, main content area, and footer. Ensure consistency, responsiveness, and usability across all key pages like login, registration, movie listings, cart, and contact.**

Prepare a common project website design and plan document for all assignments. Consider following points:

1. Brief information about the project.
2. Set the goals & deliverables.
3. Finalize the modules of the project.
4. Define the audience.
5. Describe pain points & the ideal experience (On the basis of existing systems)
6. Set the visual direction
7. Map out the Project structure.
8. Plan the content for each page.
9. Add ideas for content, images & layout.
10. Determine your site structure or Create content for your core website pages:
  - a. Home page
  - b. About page
  - c. Product/Service page
  - d. Support page
  - e. Starter blog posts
11. Create and collect design elements
12. These design elements define your brand personality and help customers feel what your brand represents through the use of:
  - a. Colors
  - b. Fonts and typography
  - c. Logos
  - d. Images and photos



## **2. Brief Information about the Project**

**Project Title: MoviesX – Online Movie Ticket & DVD Shopping**

**Description:**

**MoviesX is a dynamic movie platform that allows users to browse, buy, and review movies. It offers DVD purchasing, booking options, user registration, and an admin login. The platform includes cart functionality and database integration with PHP & MySQL.**

## **3. Goals & Deliverables**

**Goals:**

- **Develop a responsive multi-page movie shopping site**
- **Provide user login/registration and admin access**
- **Add cart functionality with PHP + MySQL integration**
- **Allow review/testimonial posting**
- **Maintain clean navigation and UI/UX**

**Deliverables:**

- **Home, About, Product (Movies), Cart, Contact pages**
- **Functional Login & Registration (User + Admin)**
- **PHP + MySQL-backed shopping cart system**
- **Connected database for login, register, and product operations**
- **Optional: Review/Testimonial feature**

## **4. Finalized Project Modules**

- **Header + Navigation**
- **Home Page**
- **About Page**
- **Movie Listing (Product Page)**
- **Cart Page**
- **Testimonial/Review Page**
- **Contact Page**
- **Login & Registration (User + Admin)**
- **MySQL Database + PHP Backend**

## **5. Target Audience**

- **Movie enthusiasts looking to buy DVDs**

- Users interested in booking online movie screenings
- Teens and adults aged 16–35
- Admins managing user and movie data

## 6. Pain Points & Ideal Experience

### Pain Points in Existing Movie Sites:

- Cluttered UI with pop-ups and ads
- Login/signup issues or confusing user flows
- No proper cart or order system for DVDs
- No review/testimonial section

### Ideal Experience:

- Straightforward login/registration
- Smooth cart and checkout system
- Responsive layout on mobile
- Movie browsing with visual previews
- Secure and fast interactions

## 7. Visual Direction

- **Tone:** Entertaining, bold, clean
- **Color Scheme:**
  - **Primary:** #E50914 (Netflix red)
  - **Secondary:** #141414 (Dark gray background)
  - **Accent:** #FFD700 (Gold highlights/buttons)
  - **Text:** #FFFFFF
- **Typography:**
  - **Headings:** "Playfair Display", serif
  - **Body:** "Inter", sans-serif
- **Imagery:**
  - High-res movie posters, DVD covers, cinema stills
  - Icons for cart, login, genre tags, etc.

## 8. Project Folder Structure

```

laptop
|
|— index.html
|— about.html
|— product.html
|— cart.html
|— contact.html
|— Movie.html <-- Current file open in the editor
|— register.html
|— test.html
|— /css/
|   |— design.css
|— /js/
|   |— web.js
|— /php/
|   |— register.php
|— /images/
|   |— product1.jpg
|   |— product2.jpg
|   |— product3.jpg
|   |— background.jpg <-- Likely image used in Movie.html

```

## 9. Content Plan for Pages

### Home Page:

- **Banner (latest movies or sales)**
- **Intro section for MoviesX**
- **Featured Movies with "Add to Cart" buttons**
- **Footer with social icons and site links**

### About Page:

- **Background of MoviesX**
- **Team or development story**
- **Mission & Vision**

### Product/Movie Listing Page:

- **Display list of movies (DVDs or streamable)**
- **Price, title, poster, and "Add to Cart" button**
- **Filter or sort options (genre/year – optional)**

### Cart Page:

- **List of selected movies**
- **Quantity selection**

- Total price calculation
- Checkout (mocked or PHP placeholder)

#### **Review/Testimonial Page:**

- User testimonials
- Ratings (stars)
- Featured user reviews

#### **Contact Page:**

- Contact form (Name, Email, Message)
- Map/embed (optional)
- PHP mail or DB storage (basic)

#### **Login/Registration Pages:**

- User login & registration
- Admin login (with separate validation)
- Error validation and success message
- Redirect to home after login

## **10. Content, Image & Layout Ideas**

#### **Image Ideas:**

- Movie posters (1920x1080 or 800x600 JPEGs)
- Icons for cart, heart (wishlist), genre tags
- Backgrounds with gradient overlays or blur
- Use Pexels, Pixabay, or Unsplash for royalty-free visuals

#### **Layout:**

- Grid-based for movie cards
- Flexbox for nav and footer
- Responsive navbar with burger menu on mobile
- Cards with shadow and hover effects

## **11. Design Elements**

### **a. Colors**

Role	Color Code	Description
Navbar/Background	#141414	Dark theme base
CTA Button	#E50914	Netflix-style red for action buttons
Button Hover	#B81D24	Slightly darker red
Text (Default)	#FFFFFF	White for contrast
Highlight/Badges	#FFD700	Gold for badges or star ratings

## b. Fonts & Typography

```
@import
url('https://fonts.googleapis.com/css2?family=Inter:wght@400;600&family=Playfair+Display:wght@600&display=swap');
```

```
body {
  font-family: 'Inter', sans-serif;
}

h1, h2, h3 {
  font-family: 'Playfair Display', serif;
}

code {
  font-family: 'Roboto Mono', monospace;
}
```

## c. Logo Design

- Wordmark style: “MoviesX” using "Playfair Display"
- Color Versions:
  - Light mode: Red on black
  - Dark mode: White on red or white on dark gray
- Logo includes a play icon inside the “O” of “MoviesX” (optional)

## d. Image Styling

```
img {
  border-radius: 8px;
  object-fit: cover;
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.15);
}
```

```
}
```

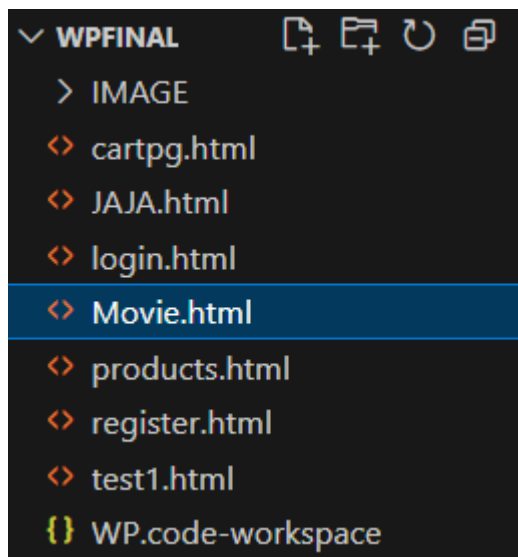
## 12. Implementation Suggestion (CSS Variables)

```
:root {  
  --primary-color: #E50914;  
  --secondary-color: #141414;  
  --accent-color: #FFD700;  
  --text-light: #FFFFFF;  
  --text-dark: #000000;  
  --background-dark: #121212;  
}
```

### Objective:

To design and develop the basic structure of the MoviesX movie shopping platform, focusing on user needs, clean UI, backend integration (PHP + MySQL), and modern web design elements.

### Output:



## Experiment No.2

### Problem Statement:

#### 2. HTML

- A. Create a detailed home page for the MoviesX website.
- B. Create a detailed movies page listing all currently available movies, categorized by genre or language.
- C. Create a cart or booking summary page that allows users to review selected movies and manage their ticket bookings.
- D. Create an about us page providing detailed information about MoviesX's mission, vision, and team.
- E. Create a contact page that allows users to get in touch with MoviesX through a simple contact form.
- F. Design and implement admin/user registration form for MoviesX.
- G. Design and implement admin/user login form for MoviesX.

#### Objective:

To create an online movie browsing and ticket booking platform webpage using HTML.

#### Theory:

HTML (HyperText Markup Language) is the core language used to structure and organize content on the web. It helps define web elements such as headings, paragraphs, tables, images, videos, buttons, links, and forms.

In this project, we are designing a MoviesX website using HTML. This platform serves as a movie discovery and booking portal, allowing users to explore current and upcoming movies, view details, and book tickets online. Such platforms are crucial in modern cinema as they allow convenient and instant access to movie schedules and ticket purchases from anywhere.

#### Website Modules / Pages:

- Home Page  
Features highlights of latest movies, trending trailers, banners, or promotional offers.
- Movies Page  
Lists all currently running or upcoming movies, with categories like language, genre, or rating.
- Movie Details Page (optional enhancement)  
Offers a detailed description of each movie including cast, synopsis, trailer, duration, and rating.

- **About Us Page**  
Describes the background of the MoviesX platform, its purpose, and team members behind it.
- **User Registration and Login Pages**  
Let users securely sign up and log in to book tickets, track bookings, or save preferences.
- **Cart or Booking Page**  
Allows users to review selected tickets/movies, modify quantity or seats, and proceed to checkout.
- **Contact Page**  
Includes a user-friendly form for customer support, feedback, or business inquiries.

## Code:

### A. index page:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>MoviesX - Home</title>
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/font-awesome@6.5.1/css/all.min.css"/>
  <link href="https://fonts.googleapis.com/css2?family=Orbitron:wght@400;700&display=swap" rel="stylesheet"/>
  <style>
    body {
      font-family: 'Orbitron', sans-serif;
      margin: 0;
      padding: 0;
      background: url("C:/Users/Shrid/OneDrive/background.jpg") no-repeat center center fixed;
      background-size: cover;
      color: white;
      animation: fadeIn 1.2s ease-in-out;
    }

    @keyframes fadeIn {
      from { opacity: 0; }
      to { opacity: 1; }
    }

    header {
      background: linear-gradient(to right, cyan, red);
      text-align: center;
      padding: 20px;
      font-size: 44px;
      font-weight: bold;
    }

    .nav {
      display: flex;
      justify-content: space-between;
      align-items: center;
      background: black;
      padding: 15px;
      position: relative;
      color: white;
    }
```



```
.nav-left,
.nav-center,
.nav-right {
  display: flex;
  align-items: center;
}
.nav-center {
position: absolute;
left: 50%;
transform: translateX(-50%);
display: flex;
gap: 20px;
}

.nav-right {
  gap: 20px;
}

.nav a {
  color: white;
  text-decoration: none;
  font-size: 18px;
  padding: 10px 15px;
  transition: background 0.3s;
}

.nav a:hover {
  background: red;
  border-radius: 5px;
}

.profile-icon {
  position: relative;
  font-size: 24px;
  color: white;
  cursor: pointer;
  margin-left: 20px;
}

#profile-hover-text {
  display: none;
  position: absolute;
  bottom: 35px;
  left: -10px;
  background-color: black;
  color: white;
  padding: 5px 10px;
  border-radius: 5px;
  font-size: 14px;
  white-space: nowrap;
}

.profile-icon:hover #profile-hover-text {
  display: block;
}

#logout-dropdown {
  display: none;
  position: absolute;
  top: 35px;
  left: 0;
  background: black;
  color: white;
  padding: 10px;
```

```
border-radius: 5px;
font-size: 14px;
cursor: pointer;
z-index: 999;
}
```

```
.cart-icon {
font-size: 24px;
color: white;
cursor: pointer;
position: relative;
}
```

```
.cart-icon span {
display: none;
position: absolute;
bottom: 35px;
left: -50px;
background-color: black;
color: white;
padding: 5px 10px;
border-radius: 5px;
font-size: 14px;
white-space: nowrap;
}
```

```
.cart-icon:hover span {
display: block;
}
```

```
.hero {
height: 80vh;
display: flex;
align-items: center;
justify-content: center;
text-align: center;
background-color: rgba(0, 0, 0, 0.7);
}
```

```
.hero h1 {
font-size: 50px;
background: linear-gradient(to right, red, cyan);
padding: 20px 40px;
border-radius: 10px;
color: whitesmoke;
transition: all 0.5s ease-in-out;
}
```

```
.about-us, .team-section {
background-color: rgba(0, 0, 0, 0.9);
padding: 60px 20px;
text-align: center;
border-radius: 20px;
margin: 20px;
box-shadow: 0px 4px 15px rgba(0, 0, 0, 0.7);
}
```

```
.about-us h2, .team-section h2 {
color: cyan;
font-size: 36px;
}
```

```
.about-us p {
font-size: 18px;
```

```

    max-width: 800px;
    margin: 20px auto;
    line-height: 1.8;
}

.team {
    display: flex;
    justify-content: center;
    flex-wrap: wrap;
    gap: 30px;
    margin-top: 40px;
}

.team-member {
    text-align: center;
    background-color: rgba(255, 255, 255, 0.1);
    padding: 30px;
    border-radius: 20px;
    width: 250px;
    transition: transform 0.3s ease;
    box-shadow: 0px 4px 10px rgba(0, 0, 0, 0.4);
}

.team-member:hover {
    transform: translateY(-10px);
}

.team-member i {
    font-size: 80px;
    color: white;
}

.team-member h3 {
    margin: 10px 0 5px;
    color: red;
}

.team-member p {
    color: white;
    font-size: 16px;
}

footer {
    background: black;
    text-align: center;
    padding: 15px;
    margin-top: 20px;
}
</style>
</head>
<body>
<header>MoviesX</header>
<nav class="nav">
<div class="nav-left">
    <div class="profile-icon" id="profile-icon" style="display: none;">
        <i class="fa fa-user-circle"></i>
        <span id="profile-hover-text">Username</span>
        <div id="logout-dropdown">Logout</div>
    </div>
</div>

<div class="nav-center" id="nav-center">
    <a href="login.html" id="login-link">Login</a>
    <a href="register.html" id="register-link">Register</a>

```

```
<a href="products.html" id="products-link">Products</a>
</div>
```

```
<div class="nav-right">
  <a href="cartpg.html" class="cart-icon">
    <i class="fa fa-shopping-cart"></i>
    <span id="cart-hover-text">View Cart</span>
  </a>
</div>
</nav>
```

```
<div class="hero">
  <h1 id="hero-text">Discover Movies Rediscover Yourself</h1>
</div>
```

```
<section class="about-us">
  <h2>About Us</h2>
  <p>
    <strong>MoviesX</strong> is your one-stop platform to explore and shop exclusive movie merchandise.
    From iconic collectibles to stylish apparel, we bring fans closer to their favorite films.
    Designed by passionate developers, MoviesX combines sleek design, secure shopping, and a smooth user experience.
  </p>
</section>
```

```
<section class="team-section">
  <h2>Our Team</h2>
  <div class="team">
    <div class="team-member">
      <i class="fa fa-female"></i>
      <h3>Shreeya Salunkhe</h3>
      <p>UI/UX Designer</p>
    </div>
    <div class="team-member">
      <i class="fa fa-female"></i>
      <h3>Sanskriti Gupta</h3>
      <p>Frontend Developer</p>
    </div>
    <div class="team-member">
      <i class="fa fa-male"></i>
      <h3>Suraj Hippargi</h3>
      <p>Backend Developer</p>
    </div>
  </div>
</section>
```

```
<footer>
  <p>&copy; 2025 MoviesX. All Rights Reserved.</p>
</footer>
```

```
<script>
function updateCartHover() {
  const cart = JSON.parse(localStorage.getItem('cart')) || [];
  document.getElementById('cart-hover-text').textContent = `View Cart (${cart.length})`;
}
window.onload = updateCartHover;
</script>
```

```
<script>
window.addEventListener('DOMContentLoaded', () => {
  const userData = JSON.parse(localStorage.getItem("moviesXUser"));
  const heroText = document.getElementById("hero-text");

  if (userData && userData.username) {
```

```

    heroText.textContent = `Hello, ${userData.username}!`;
    setTimeout(() => {
        heroText.textContent = "Discover Movies Rediscover Yourself";
    }, 3000);
    }
    });
</script>

```

```

<script>
window.addEventListener('DOMContentLoaded', () => {
    const userData = JSON.parse(localStorage.getItem("moviesXUser"));
    const registerLink = document.getElementById("register-link");
    const loginLink = document.getElementById("login-link");
    const profileIcon = document.getElementById("profile-icon");
    const profileHoverText = document.getElementById("profile-hover-text");
    const logoutDropdown = document.getElementById("logout-dropdown");

    if (userData && userData.username) {
        registerLink.style.display = "none";
        loginLink.style.display = "none";
        profileIcon.style.display = "block";
        profileHoverText.textContent = userData.username;

        logoutDropdown.addEventListener("click", () => {
            localStorage.removeItem("moviesXUser");
            window.location.reload();
        });
    } else {
        profileIcon.style.display = "none";
        registerLink.style.display = "inline-block";
        loginLink.style.display = "inline-block";
    }
});

```

```

</script>

```

```

<script>
window.addEventListener('DOMContentLoaded', () => {
    const userData = JSON.parse(localStorage.getItem("moviesXUser"));
    const registerLink = document.getElementById("register-link");
    const loginLink = document.getElementById("login-link");
    const profileIcon = document.getElementById("profile-icon");
    const profileHoverText = document.getElementById("profile-hover-text");
    const logoutDropdown = document.getElementById("logout-dropdown");

    if (userData && userData.username) {
        registerLink.style.display = "none";
        loginLink.style.display = "none";
        profileIcon.style.display = "block";
        profileHoverText.textContent = userData.username;

        // Toggle dropdown on profile icon click
        profileIcon.addEventListener("click", (e) => {
            e.stopPropagation();
            logoutDropdown.style.display = logoutDropdown.style.display === "block" ? "none" : "block";
        });

        // Click outside hides the dropdown
        document.addEventListener("click", (e) => {
            if (!profileIcon.contains(e.target)) {
                logoutDropdown.style.display = "none";
            }
        });

        // Logout action
    }
});

```

```

logoutDropdown.addEventListener("click", () => {
    localStorage.removeItem("moviesXUser");
    window.location.reload();
});

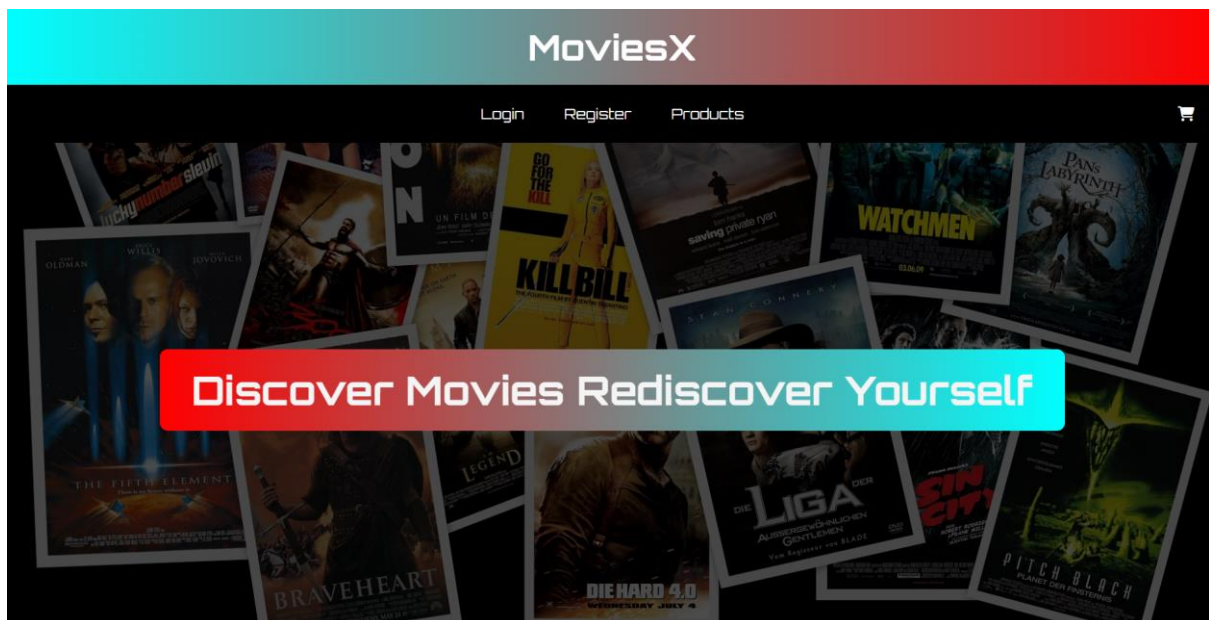
} else {
    profileIcon.style.display = "none";
    registerLink.style.display = "inline-block";
    loginLink.style.display = "inline-block";
}
});
</script>

</body>
</html>

```

## Output:

A. Index/Home page output:



## Code:

B.Product page:

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>MoviesX - Products</title>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.5.1/css/all.min.css">
    <link href="https://fonts.googleapis.com/css2?family=Orbitron:wght@400;700&display=swap" rel="stylesheet">

    <style>
    body {
        font-family: 'Orbitron', sans-serif;
        background: linear-gradient(to right, cyan, red);
        margin: 0;
        padding: 0;
    }

```

```
background-color: #1e1e1e;
color: white;
}
header {
background: linear-gradient(to right, cyan, red);
font-family: 'Orbitron', sans-serif;
text-align: center;
padding: 20px;
font-size: 44px;
font-weight: bold;
}
.nav {
display: flex;
justify-content: center;
background: black;
padding: 15px;
position: relative;
}
.nav a {
color: white;
text-decoration: none;
padding: 12px 20px;
font-size: 18px;
transition: 0.3s;
}
.nav a:hover {
background: red;
border-radius: 5px;
}
.cart-icon {
position: absolute;
right: 20px;
font-size: 24px;
color: white;
cursor: pointer;
}
.cart-icon span {
display: none;
position: absolute;
bottom: 35px;
left: -70px;
background-color: black;
color: white;
padding: 5px 10px;
border-radius: 5px;
font-size: 14px;
white-space: nowrap;
}
.cart-icon:hover span {
display: block;
}
.products-container {
display: grid;
grid-template-columns: repeat(3, 1fr);
gap: 20px;
max-width: 1000px;
margin: 40px auto;
}
.product-card {
background-color: #333;
padding: 20px;
border-radius: 10px;
text-align: center;
}
```

```
.product-card img {
  width: 100%;
  height: 300px;
  border-radius: 10px;
  object-fit: cover;
}
button {
  background: red;
  color: white;
  border: none;
  padding: 10px 20px;
  margin-top: 10px;
  cursor: pointer;
  border-radius: 5px;
}
button:hover {
  background: cyan;
}
.product-card h3 {
  font-size: 24px;
  color: white;
  transition: all 0.3s ease;
  position: relative;
  z-index: 1;
}

.product-card h3::after {
  content: "";
  position: absolute;
  left: 0;
  bottom: 0;
  height: 3px;
  width: 100%;
  background: cyan;
  transform: scaleX(0);
  transform-origin: right;
  transition: transform 0.3s ease;
  z-index: -1;
}

.product-card h3:hover {
  color: cyan;
  transform: scale(1.05);
}

.product-card h3:hover::after {
  transform: scaleX(1);
  transform-origin: left;
}
.product-card h3 {
  position: relative;
  display: inline-block;
  transition: 0.3s ease;
  cursor: pointer;
}

.product-card h3::after {
  content: "";
  position: absolute;
  left: 0;
  bottom: -5px;
  width: 100%;
  height: 2px;
  background: cyan;
}
```



```
transform: scaleX(0);
transform-origin: right;
transition: transform 0.3s ease;
}
```

```
.product-card h3:hover::after {
transform: scaleX(1);
transform-origin: left;
}
```

```
.product-card h3:hover {
color: cyan;
text-shadow: 0 0 10px cyan, 0 0 20px red;
}
.product-card {
background-color: #333;
padding: 20px;
border-radius: 10px;
text-align: center;
transition: transform 0.3s ease, box-shadow 0.3s ease;
cursor: pointer;
}
```

```
.product-card:hover {
transform: translateY(-10px) scale(1.03);
box-shadow: 0 10px 20px rgba(0, 255, 255, 0.4), 0 0 20px rgba(255, 0, 0, 0.4);
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<header>MoviesX - Products</header>
```

```
<nav class="nav">
```

```
<a href="Movie.html">Home</a>
```

```
<a href="cartpg.html" class="cart-icon" onclick="window.location.href='cartpg.html';">
```

```
<i class="fa fa-shopping-cart"></i>
```

```
<span id="cart-hover-text">View Cart (0)</span>
```

```
</a>
```

```
<a href="register.html">Register</a>
```

```
<a href="login.html">Login</a>
```

```
</nav>
```

```
<div class="products-container" id="products-container"></div>
```

```
<script>
```

```
const movies = [
```

```
{ name: 'Daredevil', genre: 'Action', price: '299', img: 'IMAGE/Daredevil.png' },
```

```
{ name: 'Interstellar', genre: 'Sci-Fi', price: '450', img: 'IMAGE/Interstellar.png' },
```

```
{ name: 'Oppenheimer', genre: 'Biopic', price: '300', img: 'IMAGE/image.png' },
```

```
{ name: 'Stree 2', genre: 'Horror Comedy', price: '120', img: 'IMAGE/Stree2.png' },
```

```
{ name: 'Avengers: Endgame', genre: 'Action', price: '350', img: 'IMAGE/Avengers.png' },
```

```
{ name: 'Inception', genre: 'Sci-Fi', price: '400', img: 'IMAGE/Inception.png' }
```

```
];
```

```
const container = document.getElementById('products-container');
```

```
movies.forEach(movie => {
```

```
const card = document.createElement('div');
```

```
card.className = 'product-card';
```

```
card.innerHTML = `
```

```

```

```
<h3>${movie.name}</h3>
```

```
<p>Genre: ${movie.genre}</p>
```

```
<p>Price: ₹${movie.price}</p>
```

```
<button onclick="addToCart('${movie.name}', '${movie.genre}', '${movie.price}')">Add to Cart</button>
```

```
`;
```

```

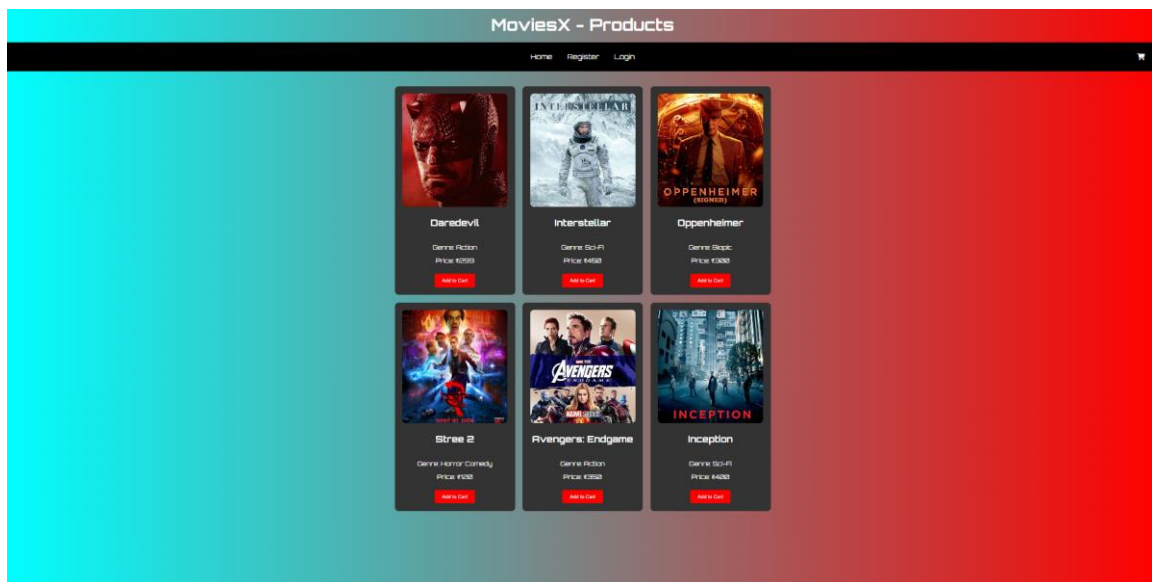
        container.appendChild(card);
    });

    function addToCart(name, genre, price) {
        const movie = { name, genre, price };
        let cart = JSON.parse(localStorage.getItem('cart')) || [];
        cart.push(movie);
        localStorage.setItem('cart', JSON.stringify(cart));
        alert(`${name} has been added to your cart!`);
        updateCartHover();
    }
    function updateCartHover() {
        const cart = JSON.parse(localStorage.getItem('cart')) || [];
        const cartCount = cart.length;
        document.getElementById('cart-hover-text').textContent = `View Cart (${cartCount})`;
    }
    window.onload = updateCartHover;
</script>
</body>
</html>

```

## Output:

### B. Page output:



## Code:

### B.Cart page:

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>MoviesX - Cart</title>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.5.1/css/all.min.css">
    <link href="https://fonts.googleapis.com/css2?family=Orbitron:wght@400;700&display=swap" rel="stylesheet">

<style>
    body {

```

```
font-family: 'Orbitron', sans-serif;
background: linear-gradient(to right, cyan, red);
margin: 0;
padding: 0;
background-color: #1e1e1e;
color: white;
}
header {
background: linear-gradient(to right, cyan, red);
font-family: 'Orbitron', sans-serif;
text-align: center;
padding: 20px;
font-size: 44px;
font-weight: bold;
}
.nav {
display: flex;
justify-content: center;
background: black;
padding: 15px;
position: relative;
}
.nav a {
color: white;
text-decoration: none;
padding: 12px 20px;
font-size: 18px;
transition: 0.3s;
}
.nav a:hover {
background: red;
border-radius: 5px;
}
.cart-icon {
position: absolute;
right: 20px;
font-size: 24px;
color: white;
cursor: pointer;
}
.cart-icon span {
display: none;
position: absolute;
top: -30px;
right: 0;
background-color: black;
color: white;
padding: 5px 10px;
border-radius: 5px;
font-size: 14px;
white-space: nowrap;
}
.cart-icon:hover span {
display: block;
}
.cart-container {
max-width: 800px;
margin: 40px auto;
background-color: #333;
padding: 20px;
border-radius: 10px;
text-align: center;
}
#total-price {
```

```

    font-size: 24px;
    margin-top: 20px;
}
button {
    background: red;
    font-family: 'Orbitron', sans-serif;
    color: white;
    border: none;
    padding: 10px 20px;
    margin-top: 10px;
    cursor: pointer;
    border-radius: 5px;
    margin-right: 10px;
}
button:hover {
    background: cyan;
}
}
</style>
</head>
<body>
<header>MoviesX - Cart</header>
<nav class="nav">
    <a href="Movie.html">Home</a>
    <a href="products.html">Products</a>
    <a href="register.html">Register</a>
    <a href="login.html">Login</a>
    <div class="cart-icon" id="cart-icon">
        <i class="fa fa-shopping-cart"></i>
        <span id="cart-hover">View Cart</span>
    </div>
</nav>

<div class="cart-container" id="cart-container">
    <h2>Your Cart</h2>
    <div id="cart-info"></div>
    <div id="total-price"></div>
    <button onclick="clearCart()">Clear Cart</button>
    <button onclick="checkout()">Checkout</button>
</div>

<script>
function loadCart() {
    const cart = JSON.parse(localStorage.getItem('cart')) || [];
    const cartInfo = document.getElementById('cart-info');
    const totalPriceElement = document.getElementById('total-price');
    const cartHover = document.getElementById('cart-hover');

    cartHover.textContent = cart.length > 0 ? `View Cart (${cart.length} items)` : 'View Cart';

    if (cart.length === 0) {
        cartInfo.innerHTML = '<p>Your cart is empty.</p>';
        totalPriceElement.textContent = '';
        return;
    }

    let totalPrice = 0;
    cartInfo.innerHTML = '';

    cart.forEach(movie => {
        cartInfo.innerHTML += `<p>Movie: ${movie.name}</p><p>Genre: ${movie.genre}</p><p>Price:
₹${movie.price}</p><hr>`;
        totalPrice += parseInt(movie.price);
    });

```

```

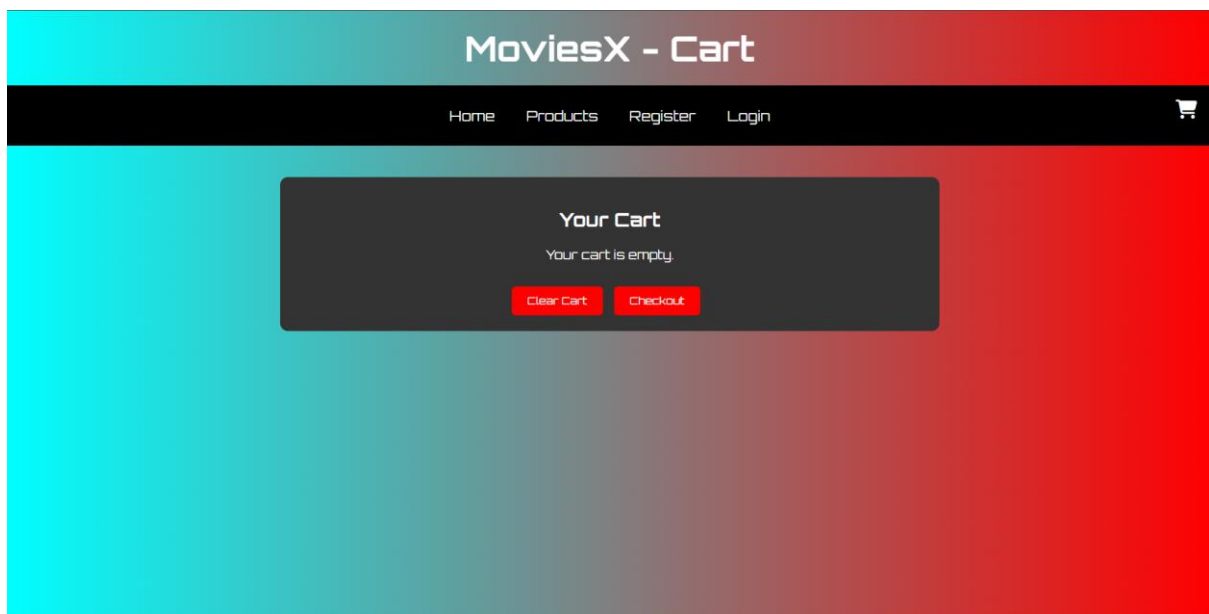
        totalPriceElement.textContent = `Total Price: ₹${totalPrice}`;
    }

    function clearCart() {
        localStorage.removeItem('cart');
        alert('Cart has been cleared!');
        loadCart();
    }
    function checkout() {
        const cart = JSON.parse(localStorage.getItem('cart')) || [];
        if (cart.length === 0) {
            alert('Your cart is empty. Add items to proceed with checkout.');
            return;
        }
        alert('Checkout successful! Thank you for your purchase.');
        clearCart();
    }
    window.onload = loadCart;
</script>
</body>
</html>

```

## Output:

C. Cart Page output:



## Code:

D. About us page:

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>MoviesX - Home</title>
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.5.1/css/all.min.css"/>

```

```

<link href="https://fonts.googleapis.com/css2?family=Orbitron:wght@400;700&display=swap" rel="stylesheet" />
<section class="about-us">
  <h2>About Us</h2>
  <p>
    <strong>MoviesX</strong> is your one-stop platform to explore and shop exclusive movie merchandise.
    From iconic collectibles to stylish apparel, we bring fans closer to their favorite films.
    Designed by passionate developers, MoviesX combines sleek design, secure shopping, and a smooth user experience.
  </p>
</section>

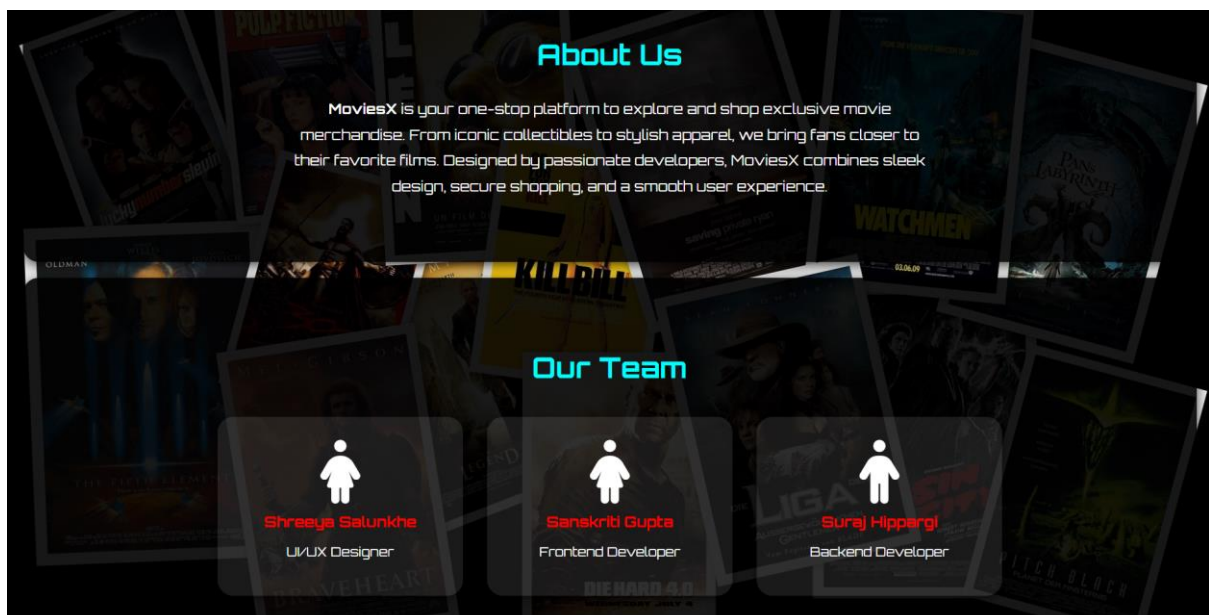
<section class="team-section">
  <h2>Our Team</h2>
  <div class="team">
    <div class="team-member">
      <i class="fa fa-female"></i>
      <h3>Shreeya Salunkhe</h3>
      <p>UI/UX Designer</p>
    </div>
    <div class="team-member">
      <i class="fa fa-female"></i>
      <h3>Sanskriti Gupta</h3>
      <p>Frontend Developer</p>
    </div>
    <div class="team-member">
      <i class="fa fa-male"></i>
      <h3>Suraj Hippargi</h3>
      <p>Backend Developer</p>
    </div>
  </div>
</section>

<footer>
  <p>&copy; 2025 MoviesX. All Rights Reserved.</p>
</footer>
</body>
</html>

```

## Output:

D. About us Page output:



## E. Register Page:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>MoviesX - Register</title>
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.5.1/css/all.min.css"/>
  <link href="https://fonts.googleapis.com/css2?family=Orbitron:wght@400;700&display=swap" rel="stylesheet" />
</style>
body {
  font-family: 'Orbitron', sans-serif;
  background: linear-gradient(to right, cyan, red);
  margin: 0;
  padding: 0;
  height: 100vh;
  display: flex;
  justify-content: center;
  align-items: center;
}

.nav {
  display: flex;
  justify-content: center;
  background: black;
  padding: 15px;
  position: fixed;
  top: 0;
  width: 100%;
  z-index: 1000;
}

.nav a {
  color: white;
  text-decoration: none;
  padding: 12px 20px;
  font-size: 18px;
  transition: 0.3s;
}

.nav a:hover {
  background: red;
  border-radius: 5px;
}

.register-container {
  background: black;
  padding: 30px;
  border-radius: 10px;
  color: white;
```

```
width: 350px;
text-align: center;
margin-top: 80px;
}
```

```
input, select, button {
width: 100%;
padding: 10px;
margin: 10px 0;
border-radius: 5px;
border: 2px solid transparent;
font-family: 'Orbitron', sans-serif;
background: #111;
color: white;
}
```

```
input.valid, select.valid {
border-color: #00ff99;
box-shadow: 0 0 8px #00ff99;
}
```

```
input.invalid, select.invalid {
border-color: #ff4d4d;
box-shadow: 0 0 8px #ff4d4d;
}
```

```
button {
background: red;
color: white;
cursor: pointer;
}
```

```
button:hover {
background: cyan;
color: black;
}
```

```
.back {
color: white;
text-decoration: none;
display: block;
margin-top: 10px;
}
```

```
.back a {
color: cyan;
}
```

```
.error {
color: red;
font-size: 12px;
text-align: left;
margin-bottom: -5px;
}
```



```

.password-container {
  position: relative;
}

.password-container i {
  position: absolute;
  right: 10px;
  top: 35%;
  cursor: pointer;
  color: white;
}

input::placeholder,
select {
  color: rgba(255, 255, 255, 0.6);
}

input[type="date"]::-webkit-datetime-edit {
  color: rgba(255, 255, 255, 0.6);
}

select:invalid {
  color: rgba(255, 255, 255, 0.6);
}
</style>
</head>
<body>

<nav class="nav">
  <a href="Movie.html">Home</a>
  <a href="products.html">Products</a>
  <a href="login.html">Login</a>
</nav>

<div class="register-container">
  <h2>Create Your MoviesX Account</h2>
  <form id="registerForm">
    <input type="text" id="username" placeholder="Username" required />
    <div id="username-error" class="error"></div>

    <input type="email" id="email" placeholder="Email" required />
    <div id="email-error" class="error"></div>

    <div class="password-container">
      <input type="password" id="password" placeholder="Password" required />
      <i class="fa fa-eye" id="togglePassword"></i>
    </div>
    <div id="password-error" class="error"></div>

    <input type="date" id="dob" placeholder="Date of Birth" required />
    <div id="dob-error" class="error"></div>

    <select id="genre" required>

```

```

    <option value="">Favorite Genre</option>
    <option>Action</option>
    <option>Comedy</option>
    <option>Drama</option>
    <option>Sci-Fi</option>
    <option>Horror</option>
    <option>Romance</option>
    <option>Thriller</option>
    <option>Animation</option>
  </select>
  <div id="genre-error" class="error"></div>

  <button type="submit">Register</button>
</form>
<span class="back">Already have an account? <a href="login.html">Login</a></span>
</div>

<script>
const form = document.getElementById('registerForm');
const togglePassword = document.getElementById('togglePassword');
const passwordInput = document.getElementById('password');

togglePassword.addEventListener('click', () => {
  const type = passwordInput.type === 'password' ? 'text' : 'password';
  passwordInput.type = type;
  togglePassword.classList.toggle('fa-eye');
  togglePassword.classList.toggle('fa-eye-slash');
});

function setValidationState(input, isValid) {
  input.classList.remove('valid', 'invalid');
  input.classList.add(isValid ? 'valid' : 'invalid');
}

form.addEventListener('input', (event) => {
  const username = document.getElementById('username');
  const email = document.getElementById('email');
  const password = document.getElementById('password');
  const dob = document.getElementById('dob');
  const genre = document.getElementById('genre');

  if (event.target.id === 'username') {
    const isValid = username.value.length >= 6 && /^[!@#$$%^&*()_."':{}|<>]/g.test(username.value);
    document.getElementById('username-error').textContent = isValid ? '' : 'Username must be at least 6 characters and contain a special character.';
    setValidationState(username, isValid);
  }

  if (event.target.id === 'email') {
    const isValid = /@/g.test(email.value);
    document.getElementById('email-error').textContent = isValid ? '' : 'Email must contain @';
    setValidationState(email, isValid);
  }

  if (event.target.id === 'password') {

```

```

const isValid = password.value.length >= 8 && /[0-9]/.test(password.value) &&
/[a-z]/.test(password.value) && /[!@#$$%^&*()_,.?":{}|<>]/.test(password.value);
document.getElementById('password-error').textContent = isValid ? '' : 'Password must be at least 8 characters, with one digit,
one lowercase letter, and one special character.';
setValidationState(password, isValid);
}

if (event.target.id === 'dob') {
const isValid = dob.value !== '';
document.getElementById('dob-error').textContent = isValid ? '' : 'Please enter your Date of Birth.';
setValidationState(dob, isValid);
}

if (event.target.id === 'genre') {
const isValid = genre.value !== '';
document.getElementById('genre-error').textContent = isValid ? '' : 'Please select a genre.';
setValidationState(genre, isValid);
}
});

form.addEventListener('submit', (event) => {
event.preventDefault();

const errors = [
document.getElementById('username-error').textContent,
document.getElementById('email-error').textContent,
document.getElementById('password-error').textContent,
document.getElementById('dob-error').textContent,
document.getElementById('genre-error').textContent
];

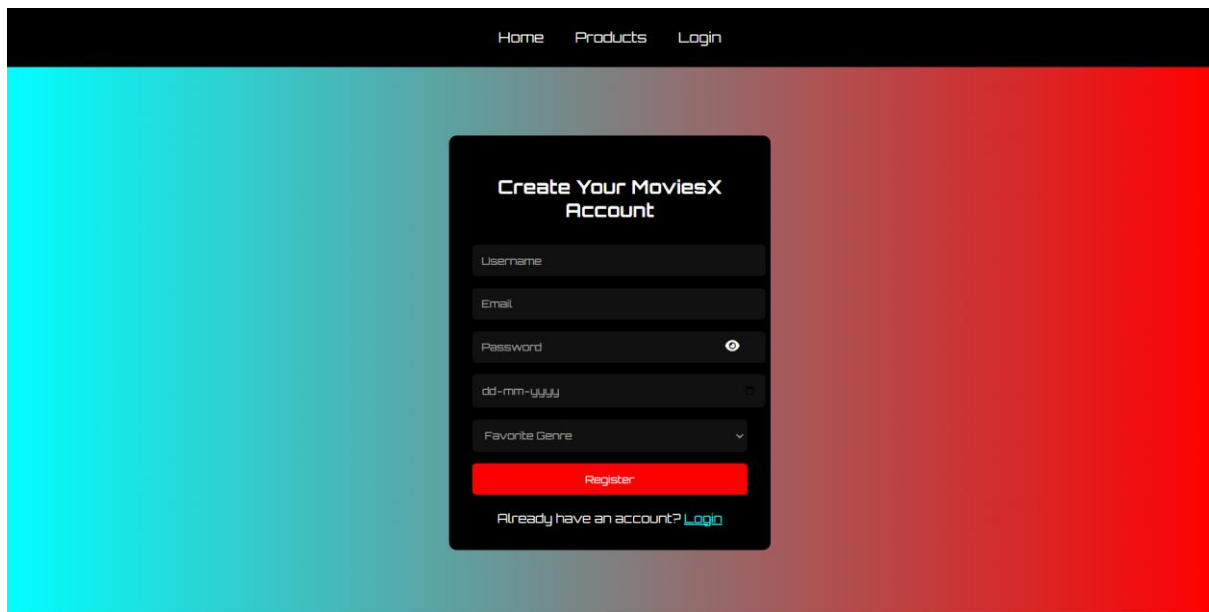
if (errors.some(err => err !== '')) {
alert("Please fix the errors before submitting.");
} else {
const userData = {
username: document.getElementById('username').value,
email: document.getElementById('email').value,
dob: document.getElementById('dob').value,
genre: document.getElementById('genre').value
};
localStorage.setItem("moviesXUser", JSON.stringify(userData));

// Redirect to home page
window.location.href = "Movie.html";
}
});
</script>

</body>
</html>

```

Output:



The screenshot displays a web application interface. At the top, a black navigation bar contains the links 'Home', 'Products', and 'Login' in white text. The main content area has a background with a horizontal gradient from cyan on the left to red on the right. Centered in this area is a dark gray registration form titled 'Create Your MoviesX Account'. The form includes input fields for 'Username', 'Email', 'Password' (with a toggle icon), and a date field labeled 'dd-mm-yyyy'. Below these is a 'Favorite Genre' dropdown menu. A prominent red 'Register' button is positioned below the dropdown. At the bottom of the form, it says 'Already have an account? [Login](#)'.

## **Conclusion:**

This experiment focused on creating a basic structure of an online MOVIES platform using HTML. We developed key pages like the home, about, contact, registration, and login pages. The project helped us understand how to build a clean, user-friendly layout and organize content effectively, forming a solid foundation for future web development.

# Experiment No.3

## Problem Statement:

Enhance the layout of the MoviesX website using CSS Grid for the homepage. Use CSS Grid to structure the movie listings and organize categories like genres, featured movies, and recommended films with headings, spacing, images, summaries, and links for more details.

## Theory:

### CSS Grid for MoviesX – Online Movie Browsing Platform

#### Introduction to CSS Grid

CSS Grid Layout is a two-dimensional system that helps design responsive web layouts efficiently. Unlike Flexbox (which works in one direction), CSS Grid operates across both rows and columns, making it ideal for websites that display structured content like movie listings and categories.

#### Why CSS Grid for an Online Movie Platform?

An online movie platform like MoviesX needs a clean and organized layout to display movie details effectively. CSS Grid helps in:

- Creating responsive and flexible layouts for movie cards.
- Arranging sections like "Featured Movies," "Trending Now," "Movie Genres," or "Top Rated."
- Ensuring consistency in the alignment of movie posters, titles, genres, and ratings.
- Supporting smooth transitions between mobile and desktop views for a clean user experience.

#### 1. Homepage Layout with CSS Grid

The homepage includes:

- A full-width navigation bar with links to different movie categories.
- A hero section featuring a banner or introductory content.
- A movie grid showcasing featured or trending movies.
- A testimonial or review section.
- A footer with additional links and social media.

Benefits of Using CSS Grid:

- Better control over spacing, layout, and structure.
- Easy scalability for adding new sections like genres or movie collections.
- Consistent and clean user experience across various screen sizes.

#### 2. Movie Listing Layout Using CSS Grid

Each movie is shown in a card with:

- A movie poster/image.

- Title of the movie.
- A short summary or description.
- Rating or review score.
- "More Details" link to the movie's detailed page.

Example CSS:

CSS

CopyEdit

```
.movie-grid {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
  gap: 30px;
  padding: 20px;
}
```

### Additional Styling Concepts for MoviesX Website

- **Category Headings:** Use larger fonts, background colors, or borders to differentiate movie categories (e.g., "Action," "Comedy," "Drama").
- **Separators:** Use borders or background colors to create visual separation between different sections, like "Featured Movies" and "Top Rated."
- **Hover Effects:** Add hover effects like animations on movie cards to make the interface interactive and engaging.
- **Responsive Design:** The use of auto-fit and minmax() in grid layout makes the website mobile-friendly, ensuring that the movie cards adjust to different screen sizes.

### Mobile Responsiveness with CSS Grid

CSS Grid enables seamless transitions on smaller screens, providing a flexible and responsive layout. For a movie platform:

- 1–2 column layout on mobile devices to ensure movie cards are easy to browse.
- Tap-friendly spacing for users to easily interact with movie details or select movies to watch.
- Fast, clean mobile browsing with movies neatly arranged in a grid.

## Code:

```
<style>
body {
  font-family: 'Orbitron', sans-serif;
  margin: 0;
  padding: 0;
  background: url("C:/Users/Shrid/OneDrive/background.jpg") no-repeat center center fixed;
  background-size: cover;
  color: white;
  animation: fadeIn 1.2s ease-in-out;
}
```

```
@keyframes fadeIn {
  from { opacity: 0; }
  to { opacity: 1; }
}

header {
  background: linear-gradient(to right, cyan, red);
  text-align: center;
  padding: 20px;
  font-size: 44px;
  font-weight: bold;
}

.nav {
  display: flex;
  justify-content: space-between;
  align-items: center;
  background: black;
  padding: 15px;
  position: relative;
  color: white;
}

.nav-left,
.nav-center,
.nav-right {
  display: flex;
  align-items: center;
}

.nav-center {
  position: absolute;
  left: 50%;
  transform: translateX(-50%);
  display: flex;
  gap: 20px;
}

.nav-right {
  gap: 20px;
}

.nav a {
  color: white;
  text-decoration: none;
  font-size: 18px;
  padding: 10px 15px;
  transition: background 0.3s;
}

.nav a:hover {
  background: red;
  border-radius: 5px;
}

.profile-icon {
  position: relative;
  font-size: 24px;
  color: white;
  cursor: pointer;
  margin-left: 20px;
}
```

```
#profile-hover-text {
  display: none;
  position: absolute;
  bottom: 35px;
  left: -10px;
  background-color: black;
  color: white;
  padding: 5px 10px;
  border-radius: 5px;
  font-size: 14px;
  white-space: nowrap;
}

.profile-icon:hover #profile-hover-text {
  display: block;
}

#logout-dropdown {
  display: none;
  position: absolute;
  top: 35px;
  left: 0;
  background: black;
  color: white;
  padding: 10px;
  border-radius: 5px;
  font-size: 14px;
  cursor: pointer;
  z-index: 999;
}

.cart-icon {
  font-size: 24px;
  color: white;
  cursor: pointer;
  position: relative;
}

.cart-icon span {
  display: none;
  position: absolute;
  bottom: 35px;
  left: -50px;
  background-color: black;
  color: white;
  padding: 5px 10px;
  border-radius: 5px;
  font-size: 14px;
  white-space: nowrap;
}

.cart-icon:hover span {
  display: block;
}

.hero {
  height: 80vh;
  display: flex;
  align-items: center;
  justify-content: center;
  text-align: center;
  background-color: rgba(0, 0, 0, 0.7);
}
```



```
}

.hero h1 {
  font-size: 50px;
  background: linear-gradient(to right, red, cyan);
  padding: 20px 40px;
  border-radius: 10px;
  color: whitesmoke;
  transition: all 0.5s ease-in-out;
}

.about-us, .team-section {
  background-color: rgba(0, 0, 0, 0.9);
  padding: 60px 20px;
  text-align: center;
  border-radius: 20px;
  margin: 20px;
  box-shadow: 0px 4px 15px rgba(0, 0, 0, 0.7);
}

.about-us h2, .team-section h2 {
  color: cyan;
  font-size: 36px;
}

.about-us p {
  font-size: 18px;
  max-width: 800px;
  margin: 20px auto;
  line-height: 1.8;
}

.team {
  display: flex;
  justify-content: center;
  flex-wrap: wrap;
  gap: 30px;
  margin-top: 40px;
}

.team-member {
  text-align: center;
  background-color: rgba(255, 255, 255, 0.1);
  padding: 30px;
  border-radius: 20px;
  width: 250px;
  transition: transform 0.3s ease;
  box-shadow: 0px 4px 10px rgba(0, 0, 0, 0.4);
}

.team-member:hover {
  transform: translateY(-10px);
}

.team-member i {
  font-size: 80px;
  color: white;
}

.team-member h3 {
  margin: 10px 0 5px;
  color: red;
```

```

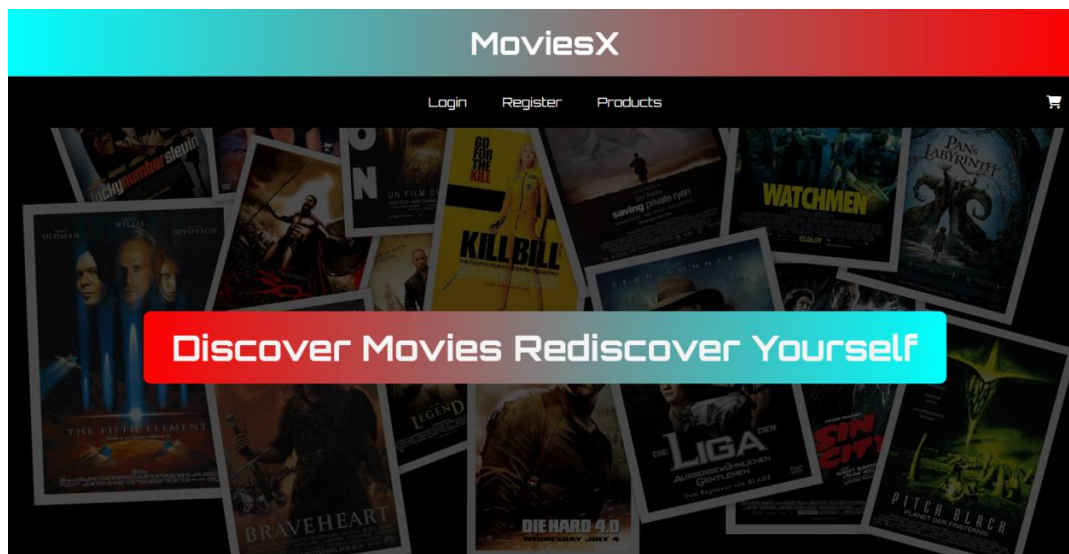
}

.team-member p {
  color: white;
  font-size: 16px;
}
}
footer {
  background: black;
  text-align: center;
  padding: 15px;
  margin-top: 20px;
}
</style>

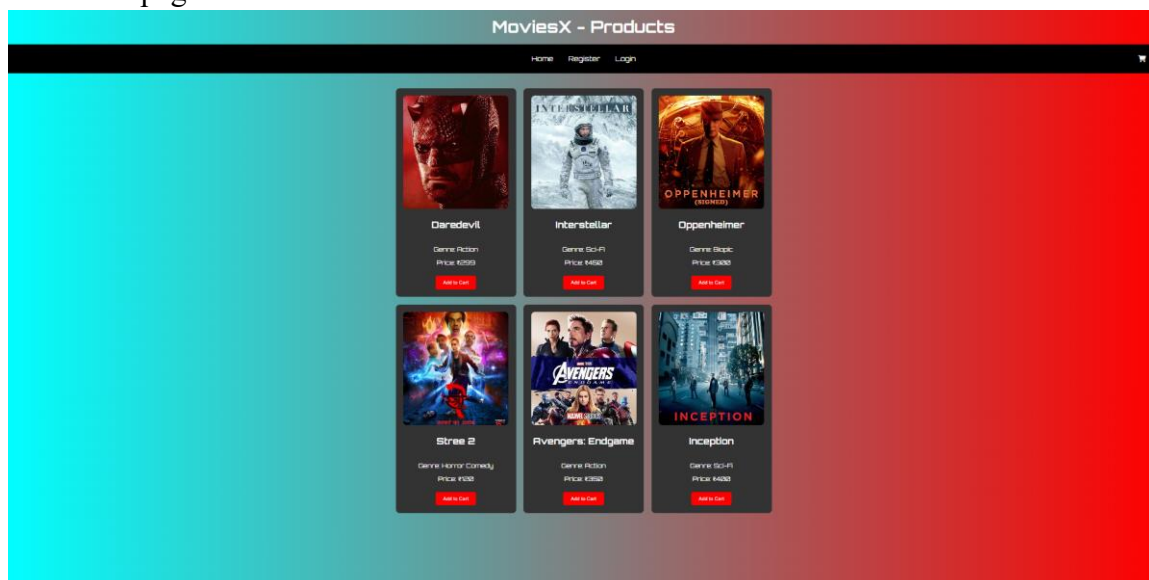
```

## Output:

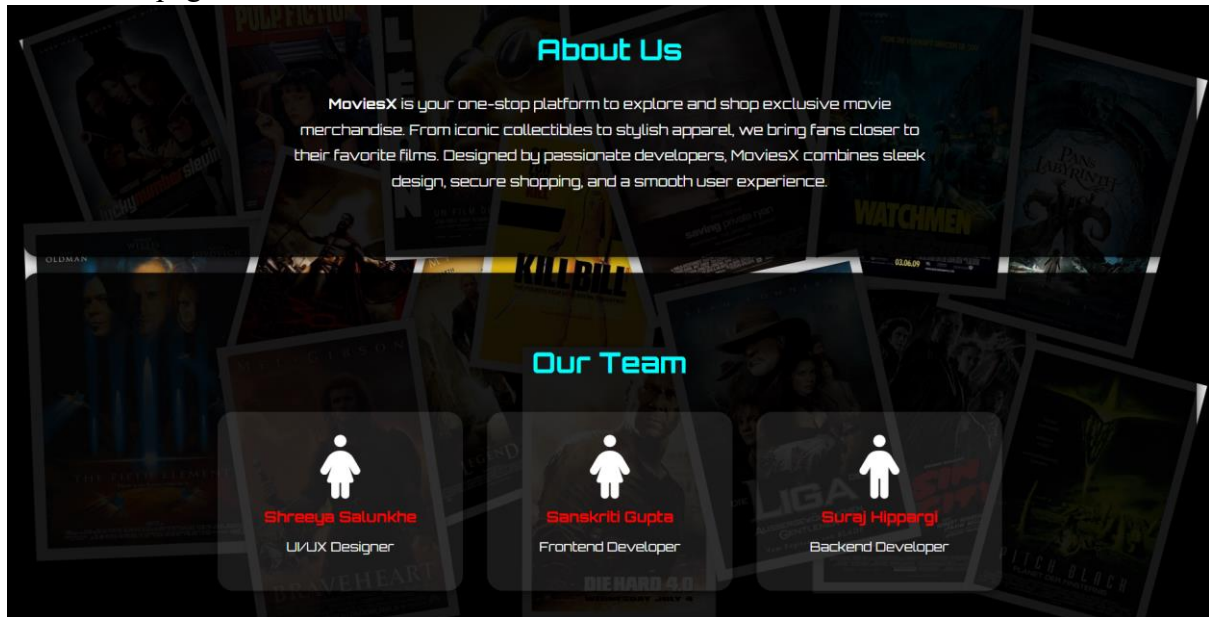
\*Home page:



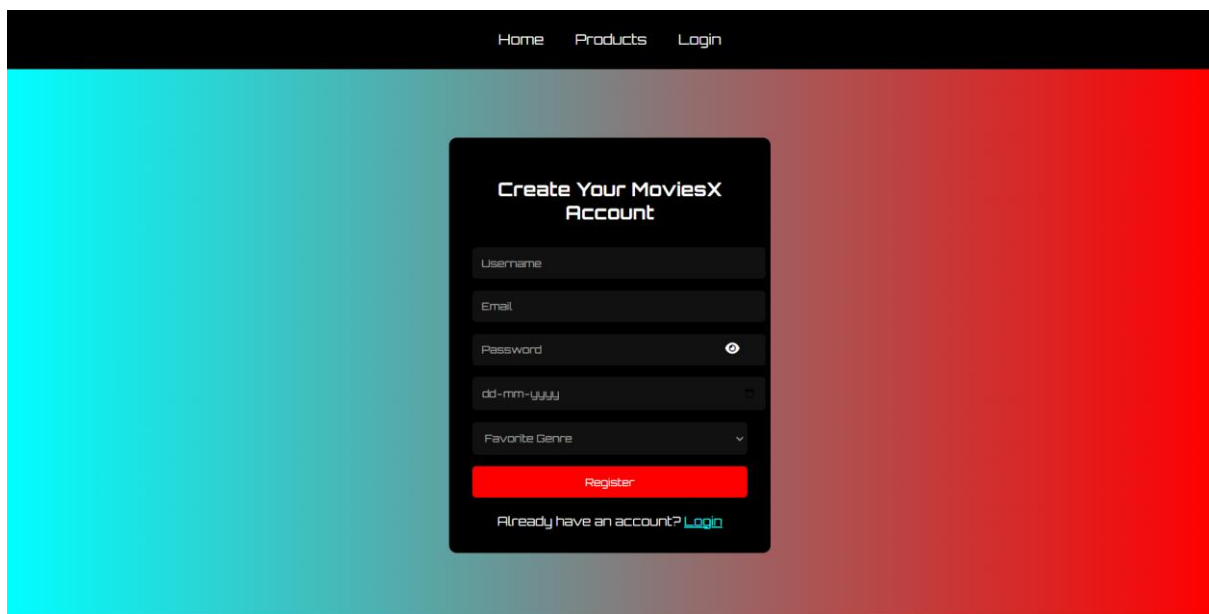
\* Product page:



\*About us page:



\*Register page



## Conclusion

Implementing **CSS Grid** for the online MOVIES platform offers a powerful and clean way to organize content visually. It enhances layout flexibility, ensures responsiveness, and provides a better **user experience** across devices. Whether it's displaying shopping posts, testimonials, or categories, CSS Grid allows developers to build scalable and intuitive designs—making site a professional, easy-to-navigate shopping space for both writers and readers.

# Experiment No.4

## Problem Statement:

Enhance the layout of the MoviesX website using CSS Grid for the homepage. Use CSS Grid to structure movie listings and organize categories with headings, spacing, images, summaries, and reading links.

CSS Theory: Enhancing and Styling Key Pages in the MoviesX Platform

### 1. Why CSS Styling Matters in MoviesX Platforms

For a platform like MoviesX, the first impression counts. A clean layout, good spacing, and appealing visuals can greatly enhance readability and engagement. Whether it's the "Featured Movies" section, "About Us" page, or login form, CSS styling:

- Improves readability, making it easy for users to explore movies and their details.
- Creates visual hierarchy, making sections like "Trending Movies", "Top Rated", and "Genres" easily identifiable.
- Enhances accessibility and navigation, ensuring users can effortlessly interact with the site.
- Increases user trust and time-on-site, encouraging users to explore more content.

Page-wise CSS Styling Theory for MoviesX Website

### 1. Add to Watchlist / Saved Movies Page (Cart Equivalent)

This page allows users to add movies they wish to watch or save to their watchlist.

Key Styling Techniques:

- Spacing and margin around each movie item to make the section clean and organized.
- Display movie title, genre, rating, and release year with ample padding for a comfortable reading experience.
- Use "Remove" buttons with hover effects for easy management of watchlist items.
- Highlight the movie title using bold fonts for better emphasis.
- Use a card-style layout with subtle shadows for each movie listing for better clarity and focus.

Result:

A clean, well-spaced section that makes the user experience intuitive and efficient.

### 2. About Us Page

This page tells the story behind MoviesX, its mission, and the platform's values.

Key Styling Techniques:

- Increase line height and padding for paragraphs for better readability.
- Space out sections like "Mission," "Team," and "Vision" for better structure and clarity.

- Use grid or flexbox to align team photos side-by-side, creating a professional look.
- Round team photos and include hover text or tooltips for interactivity.
- Style quote blocks or core values using soft boxes or colored sections to visually highlight them.

Result:

A professional and engaging layout that strengthens trust with visitors and helps users connect with the platform's story.

### 3. User/Admin Registration Form

For new users or site administrators to create an account.

Key Styling Techniques:

- Group inputs into clear sections: Personal Info and Login Credentials.
- Use descriptive placeholders and labels for better clarity.
- Maintain equal spacing and alignment for all form fields to enhance usability.
- Apply a form card background color, box shadow, and rounded borders for a clean look.
- Implement real-time validation (e.g., password strength, email format check) for a more user-friendly experience.

Result:

A professional, secure-looking form that builds trust and makes registration easy to navigate.

### 4. User/Admin Login Form

The entry point for users to access or manage their MoviesX account.

Key Styling Techniques:

- Centrally position the login box for a clean and streamlined experience.
- Provide enough padding between fields to avoid visual clutter.
- Style error messages clearly (e.g., red for error, green for success) to guide users.
- Use minimalist fonts, slight shadow effects, and clean input boxes for a modern design.
- Include a “Show Password” toggle and a forgot password link for better usability.

Result:

A smooth and efficient login interface that feels secure and user-friendly.

### Bonus: Styling Tips Across All Pages

- CSS Variables: Use for consistent colors throughout the site (e.g., --primary-color, --bg-color, etc.) to maintain a cohesive theme.
- Typography: Opt for clean, modern fonts like Inter, Open Sans, or Roboto to enhance readability.
- Responsive Layouts: Use CSS Grid or Flexbox for a mobile-first design that ensures a seamless experience on any device.
- Light/Dark Mode: Implement CSS themes to allow users to toggle between light and dark modes for a more personalized experience.

## Code:

```
<style>
body {
  font-family: 'Orbitron', sans-serif;
  margin: 0;
  padding: 0;
  background: url("C:/Users/Shrid/OneDrive/background.jpg") no-repeat center center fixed;
  background-size: cover;
  color: white;
  animation: fadeIn 1.2s ease-in-out;
}

@keyframes fadeIn {
  from { opacity: 0; }
  to { opacity: 1; }
}

header {
  background: linear-gradient(to right, cyan, red);
  text-align: center;
  padding: 20px;
  font-size: 44px;
  font-weight: bold;
}

.nav {
  display: flex;
  justify-content: space-between;
  align-items: center;
  background: black;
  padding: 15px;
  position: relative;
  color: white;
}

.nav-left,
.nav-center,
.nav-right {
  display: flex;
  align-items: center;
}

.nav-center {
  position: absolute;
  left: 50%;
  transform: translateX(-50%);
  display: flex;
  gap: 20px;
}

.nav-right {
  gap: 20px;
}

.nav a {
  color: white;
  text-decoration: none;
  font-size: 18px;
  padding: 10px 15px;
```

```
    transition: background 0.3s;
  }

.nav a:hover {
  background: red;
  border-radius: 5px;
}

.profile-icon {
  position: relative;
  font-size: 24px;
  color: white;
  cursor: pointer;
  margin-left: 20px;
}

#profile-hover-text {
  display: none;
  position: absolute;
  bottom: 35px;
  left: -10px;
  background-color: black;
  color: white;
  padding: 5px 10px;
  border-radius: 5px;
  font-size: 14px;
  white-space: nowrap;
}

.profile-icon:hover #profile-hover-text {
  display: block;
}

#logout-dropdown {
  display: none;
  position: absolute;
  top: 35px;
  left: 0;
  background: black;
  color: white;
  padding: 10px;
  border-radius: 5px;
  font-size: 14px;
  cursor: pointer;
  z-index: 999;
}

.cart-icon {
  font-size: 24px;
  color: white;
  cursor: pointer;
  position: relative;
}

.cart-icon span {
  display: none;
  position: absolute;
  bottom: 35px;
  left: -50px;
  background-color: black;
  color: white;
  padding: 5px 10px;
```

```
border-radius: 5px;
font-size: 14px;
white-space: nowrap;
}
```

```
.cart-icon:hover span {
display: block;
}
```

```
.hero {
height: 80vh;
display: flex;
align-items: center;
justify-content: center;
text-align: center;
background-color: rgba(0, 0, 0, 0.7);
}
```

```
.hero h1 {
font-size: 50px;
background: linear-gradient(to right, red, cyan);
padding: 20px 40px;
border-radius: 10px;
color: whitesmoke;
transition: all 0.5s ease-in-out;
}
```

```
.about-us, .team-section {
background-color: rgba(0, 0, 0, 0.9);
padding: 60px 20px;
text-align: center;
border-radius: 20px;
margin: 20px;
box-shadow: 0px 4px 15px rgba(0, 0, 0, 0.7);
}
```

```
.about-us h2, .team-section h2 {
color: cyan;
font-size: 36px;
}
```

```
.about-us p {
font-size: 18px;
max-width: 800px;
margin: 20px auto;
line-height: 1.8;
}
```

```
.team {
display: flex;
justify-content: center;
flex-wrap: wrap;
gap: 30px;
margin-top: 40px;
}
```

```
.team-member {
text-align: center;
background-color: rgba(255, 255, 255, 0.1);
padding: 30px;
border-radius: 20px;
width: 250px;
```



```

transition: transform 0.3s ease;
box-shadow: 0px 4px 10px rgba(0, 0, 0, 0.4);
}

.team-member:hover {
transform: translateY(-10px);
}

.team-member i {
font-size: 80px;
color: white;
}

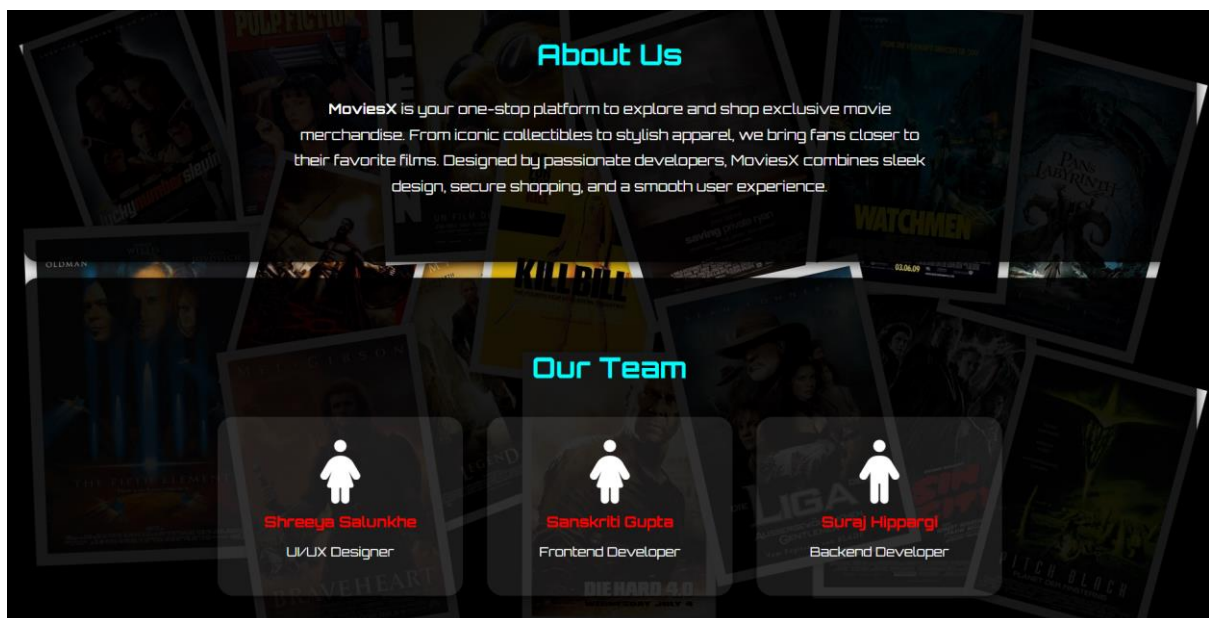
.team-member h3 {
margin: 10px 0 5px;
color: red;
}

.team-member p {
color: white;
font-size: 16px;
}
footer {
background: black;
text-align: center;
padding: 15px;
margin-top: 20px;
}

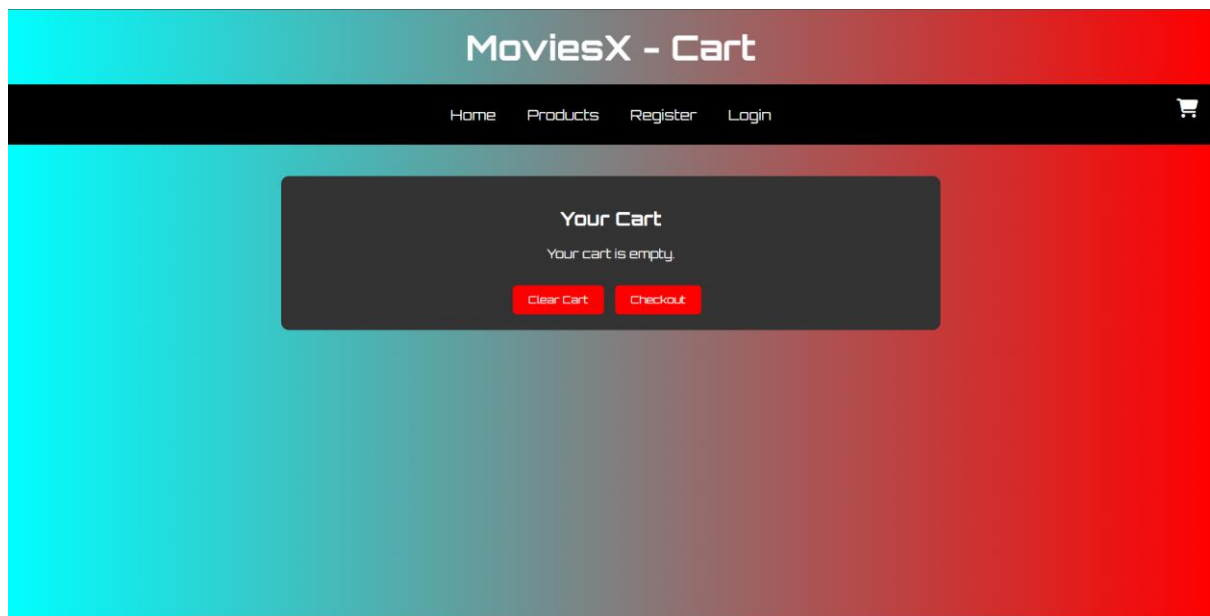
```

Output:

About us page:



\*Cart page:



## Conclusion

The success of a movie platform like **MoviesX** doesn't rely solely on its content—it also depends heavily on its **design, structure, and overall user experience**. By utilizing **CSS Grid** and modern styling techniques, we can transform basic movie listing pages into **visually engaging, well-structured, and responsive** layouts.

By styling key pages—such as the **Watchlist (Favorites), About Us, Contact, Registration, and Login** forms—we significantly enhance **usability, accessibility, and user trust**. From **organized movie displays, clear spacing, and hover effects** to **responsive grids, smooth forms, and clean typography**, every CSS enhancement adds to a more **professional and immersive movie browsing experience**.

Ultimately, thoughtful CSS design for **MoviesX** is not just about appearance—it's about creating a **welcoming, functional, and enjoyable** environment for every movie enthusiast who visits the site.

## Experiment No. 5

### Problem Statement:

JavaScript Theory: User Registration, Login, Validation, and Bookmarking Functionality for MoviesX – Online Movie Platform

### Introduction

In the digital entertainment era, a movie platform like **MoviesX** thrives on **user engagement**, **personalized content**, and a **seamless user experience**. JavaScript plays a crucial role in enhancing client-side functionality—such as **registration**, **login**, **form validation**, and **bookmarking** favorite movies—making the platform more **interactive**, **dynamic**, and **user-centric**.

### 1. User Registration and Login Forms

These forms are essential for establishing user identity and offering a personalized movie dashboard.

#### Registration Form

The registration form collects essential data like **username**, **email**, and **password** to create a new MoviesX user account.

#### JavaScript Functions in Registration:

- Ensures no input field is left blank
- Validates **email format** using RegEx
- Checks **password strength** and matches with confirm password
- Displays **real-time error messages** and hints for user assistance

#### Login Form

Allows users to securely log in and access their **saved movies**, **watchlists**, or **account settings**.

#### JavaScript Functions in Login:

- Validates that fields are not left empty
- Verifies input credentials against stored data
- Provides error messages on incorrect login attempts
- Redirects to the **home page or dashboard** after successful login

### 2. JavaScript Form Validations

Form validation improves user experience by preventing **incorrect**, **incomplete**, or **malicious entries**.

#### Common Validations Include:

- Required field checking

- **Valid email** format verification
- Password length, special characters, and digit checks
- Confirm password must **match original password**
- In-line error messages and color-coded feedback for better guidance

Client-side validation ensures a smoother experience by reducing server load and enhancing data quality.

### 3. Bookmark / Favorite Movie Feature

Instead of a shopping cart, **MoviesX** allows users to **bookmark or save** their favorite movies using a **"heart" icon or Save button**.

#### JavaScript Responsibilities:

- Clicking the **bookmark icon toggles** the save/unsave state
- Adds the movie **ID or title** to an array in **localStorage**
- Displays the saved movie list on the **"My Watchlist"** page
- Enables real-time removal of movies from the saved list dynamically
- Uses **icons and visual indicators** to show the bookmark status

#### Code:

```
<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0"/>

<title>MoviesX - Register</title>

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/font-awesome@6.5.1/css/all.min.css"/>

<link href="https://fonts.googleapis.com/css2?family=Orbitron:wght@400;700&display=swap" rel="stylesheet"/>

<style>

body {

font-family: 'Orbitron', sans-serif;

background: linear-gradient(to right, cyan, red);

margin: 0;

padding: 0;

height: 100vh;

display: flex;

justify-content: center;

align-items: center;
```

```
}
```

```
.nav {  
  display: flex;  
  justify-content: center;  
  background: black;  
  padding: 15px;  
  position: fixed;  
  top: 0;  
  width: 100%;  
  z-index: 1000;  
}
```

```
.nav a {  
  color: white;  
  text-decoration: none;  
  padding: 12px 20px;  
  font-size: 18px;  
  transition: 0.3s;  
}
```

```
.nav a:hover {  
  background: red;  
  border-radius: 5px;  
}
```

```
.register-container {  
  background: black;  
  padding: 30px;  
  border-radius: 10px;  
  color: white;  
  width: 350px;  
  text-align: center;  
  margin-top: 80px;  
}
```

```
input, select, button {  
  width: 100%;  
  padding: 10px;
```

```
margin: 10px 0;

border-radius: 5px;

border: 2px solid transparent;

font-family: 'Orbitron', sans-serif;

background: #111;

color: white;

}
```

```
input.valid, select.valid {

border-color: #00ff99;

box-shadow: 0 0 8px #00ff99;

}
```

```
input.invalid, select.invalid {

border-color: #ff4d4d;

box-shadow: 0 0 8px #ff4d4d;

}
```

```
button {

background: red;

color: white;

cursor: pointer;

}
```

```
button:hover {

background: cyan;

color: black;

}
```

```
.back {

color: white;

text-decoration: none;

display: block;

margin-top: 10px;

}
```

```
.back a {

color: cyan;

}
```

```
.error {  
  color: red;  
  font-size: 12px;  
  text-align: left;  
  margin-bottom: -5px;  
}
```

```
.password-container {  
  position: relative;  
}
```

```
.password-container i {  
  position: absolute;  
  right: 10px;  
  top: 35%;  
  cursor: pointer;  
  color: white;  
}
```

```
input::placeholder,  
select {  
  color: rgba(255, 255, 255, 0.6);  
}
```

```
input[type="date"]::-webkit-datetime-edit {  
  color: rgba(255, 255, 255, 0.6);  
}
```

```
select:invalid {  
  color: rgba(255, 255, 255, 0.6);  
}
```

</style>

</head>

<body>

<nav class="nav">

<a href="Movie.html">Home</a>

<a href="products.html">Products</a>

<a href="login.html">Login</a>

</nav>

<div class="register-container">

<h2>Create Your MoviesX Account</h2>

<form id="registerForm">

<input type="text" id="username" placeholder="Username" required />

<div id="username-error" class="error"></div>

<input type="email" id="email" placeholder="Email" required />

<div id="email-error" class="error"></div>

<div class="password-container">

<input type="password" id="password" placeholder="Password" required />

<i class="fa fa-eye" id="togglePassword"></i>

</div>

<div id="password-error" class="error"></div>

<input type="date" id="dob" placeholder="Date of Birth" required />

<div id="dob-error" class="error"></div>

<select id="genre" required>

<option value="">Favorite Genre</option>

<option>Action</option>

<option>Comedy</option>

<option>Drama</option>

<option>Sci-Fi</option>

<option>Horror</option>

<option>Romance</option>

<option>Thriller</option>

<option>Animation</option>

</select>

<div id="genre-error" class="error"></div>

<button type="submit">Register</button>

</form>

<span class="back">Already have an account? <a href="login.html">Login</a></span>

</div>



```

<script>

const form = document.getElementById('registerForm');

const togglePassword = document.getElementById('togglePassword');

const passwordInput = document.getElementById('password');


togglePassword.addEventListener('click', () => {

    const type = passwordInput.type === 'password' ? 'text' : 'password';

    passwordInput.type = type;

    togglePassword.classList.toggle('fa-eye');

    togglePassword.classList.toggle('fa-eye-slash');

});


function setValidationState(input, isValid) {

    input.classList.remove('valid', 'invalid');

    input.classList.add(isValid ? 'valid' : 'invalid');

}


form.addEventListener('input', (event) => {

    const username = document.getElementById('username');

    const email = document.getElementById('email');

    const password = document.getElementById('password');

    const dob = document.getElementById('dob');

    const genre = document.getElementById('genre');


    if (event.target.id === 'username') {

        const isValid = username.value.length >= 6 && /^[!@#$$%^&*(),.?":{}|<>]/g.test(username.value);

        document.getElementById('username-error').textContent = isValid ? " : 'Username must be at least 6 characters and contain a special character.'" : 'Username must be at least 6 characters and contain a special character.';

        setValidationState(username, isValid);

    }

    if (event.target.id === 'email') {

        const isValid = /^[@]/.test(email.value);

        document.getElementById('email-error').textContent = isValid ? " : 'Email must contain @'." : 'Email must contain @';

        setValidationState(email, isValid);

    }

    if (event.target.id === 'password') {

        const isValid = password.value.length >= 8 && /[0-9]/.test(password.value) &&

            /[a-z]/.test(password.value) && /^[!@#$$%^&*(),.?":{}|<>]/.test(password.value);

        document.getElementById('password-error').textContent = isValid ? " : 'Password must be at least 8 characters, with one digit, one lowercase letter, and one special character.'" : 'Password must be at least 8 characters, with one digit, one lowercase letter, and one special character.';

    }

});

```

```

        setValidationState(password, isValid);
    if (event.target.id === 'dob') {
        const isValid = dob.value !== "";
        document.getElementById('dob-error').textContent = isValid ? " : 'Please enter your Date of Birth.';
        setValidationState(dob, isValid);
    }

    if (event.target.id === 'genre') {
        const isValid = genre.value !== "";
        document.getElementById('genre-error').textContent = isValid ? " : 'Please select a genre.';
        setValidationState(genre, isValid);
    }
});

form.addEventListener('submit', (event) => {
    event.preventDefault();

    const errors = [
        document.getElementById('username-error').textContent,
        document.getElementById('email-error').textContent,
        document.getElementById('password-error').textContent,
        document.getElementById('dob-error').textContent,
        document.getElementById('genre-error').textContent
    ];

    if (errors.some(err => err !== "")) {
        alert("Please fix the errors before submitting.");
    } else {
        const userData = {
            username: document.getElementById('username').value,
            email: document.getElementById('email').value,
            dob: document.getElementById('dob').value,
            genre: document.getElementById('genre').value
        };

        localStorage.setItem("moviesXUser", JSON.stringify(userData));

        // Redirect to home page
        window.location.href = "Movie.html";
    }
});

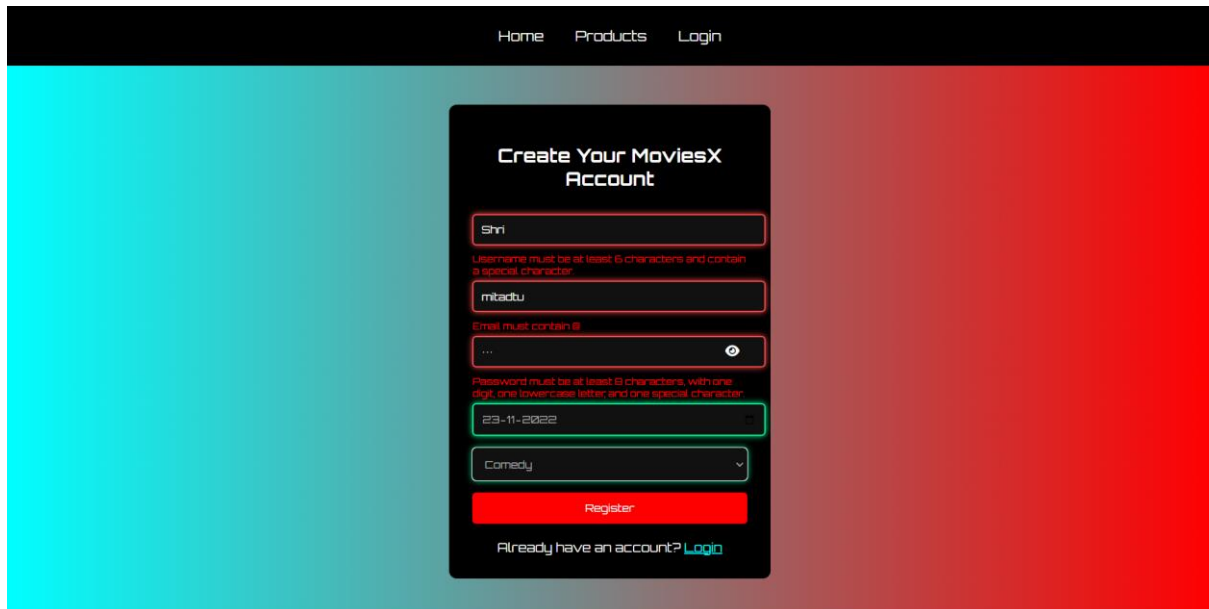
```

</script>

</body>

</html>

Output:

The image shows a web application interface for creating a new account. At the top, there is a black navigation bar with the links 'Home', 'Products', and 'Login' in white text. The main content area has a background with a blue-to-red gradient. In the center, there is a dark gray registration form titled 'Create Your MoviesX Account'. The form contains several input fields: a username field with the value 'Shri' and a red error message 'Username must be at least 6 characters and contain a special character'; an email field with the value 'mitabu' and a red error message 'Email must contain @'; a password field with a red error message 'Password must be at least 8 characters, with one digit, one lowercase letter, and one special character'; a date field with the value '23-11-2022'; and a dropdown menu for 'Comedy'. Below these fields is a red 'Register' button. At the bottom of the form, there is a link that says 'Already have an account? Login'.

## Conclusion

Implementing registration, login, validation, and bookmark features using JavaScript is essential for creating an interactive and personalized movie platform like MoviesX.

These JavaScript functionalities:

- Enhance user engagement by offering smooth and responsive interactions
- Improve usability through real-time form validations and feedback
- Empower users to save and revisit their favorite movies effortlessly via bookmarks or watchlists

Together, these dynamic features make MoviesX more intuitive, enjoyable, and user-focused—delivering a premium movie browsing experience.

## Experiment No.6

### JavaScript Theory: Persistent Login and Bookmarking using Web Storage API in MoviesX Platform

#### Introduction

A smooth and engaging user experience is critical for a movie platform like **MoviesX**. JavaScript's Web Storage API, which includes `localStorage` and `sessionStorage`, plays a vital role in maintaining **user session state and movie preferences**, such as bookmarks or watchlists—without requiring a backend database or server session.

#### 1. Persistent Login using `localStorage` / `sessionStorage`

The login system enables users to authenticate and continue accessing personalized movie content without logging in repeatedly.

##### JavaScript Implementation:

- **On successful login:**
  - Store `userEmail` and an `isLoggedIn` flag in `localStorage`
- **On every page load:**
  - JavaScript checks for login data in `localStorage`
  - If present, the user is auto-logged in and redirected to the homepage or movie dashboard
- **On logout:**
  - Clear `localStorage` to end the session

##### Benefits:

- Enhances usability with auto-login features
- Supports session continuity for frequent movie visitors
- Ideal for static prototypes like **MoviesX** that don't rely on backend systems

#### 2. Bookmark / Favourite Movies using `localStorage`

Instead of a traditional shopping cart, **MoviesX** uses a **bookmark/watchlist system** that lets users save movies they want to revisit or watch later.

##### Implementation Highlights:

- When a user clicks "Bookmark":

- The movie ID or title is saved to a **bookmarks array** in localStorage
- On the “Watchlist” or “My Favorites” page:
  - JavaScript retrieves and displays all saved movie entries from localStorage
- Bookmarks **persist across sessions** even after the browser is closed or refreshed

### Benefits:

- Increases engagement by allowing users to curate their personal movie lists
- Offers a highly personalized movie discovery and planning experience
- Saves preferences locally, ideal for simple platforms like **MoviesX**

### Bonus Use Cases for MoviesX:

- Save light/dark mode preference using localStorage
- Maintain video playback position (watch progress) for movies or trailers
- Store unfinished movie reviews or comment drafts

### Code:

#### \*Cart:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>MoviesX - Cart</title>
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.5.1/css/all.min.css">
  <link href="https://fonts.googleapis.com/css2?family=Orbitron:wght@400;700&display=swap" rel="stylesheet">

<style>
  body {
    font-family: 'Orbitron', sans-serif;
    background: linear-gradient(to right, cyan,red);
    margin: 0;
    padding: 0;
    background-color: #1e1e1e;
    color: white;
  }
  header {
    background: linear-gradient(to right, cyan,red);
    font-family: 'Orbitron', sans-serif;
    text-align: center;
    padding: 20px;
    font-size: 44px;
    font-weight: bold;
  }
  .nav {
    display: flex;
    justify-content: center;
    background: black;
    padding: 15px;
    position: relative;
  }
```

```

.nav a {
  color: white;
  text-decoration: none;
  padding: 12px 20px;
  font-size: 18px;
  transition: 0.3s;
}
.nav a:hover {
  background: red;
  border-radius: 5px;
}
.cart-icon {
  position: absolute;
  right: 20px;
  font-size: 24px;
  color: white;
  cursor: pointer;
}
.cart-icon span {
  display: none;
  position: absolute;
  top: -30px;
  right: 0;
  background-color: black;
  color: white;
  padding: 5px 10px;
  border-radius: 5px;
  font-size: 14px;
  white-space: nowrap;
}
.cart-icon:hover span {
  display: block;
}
.cart-container {
  max-width: 800px;
  margin: 40px auto;
  background-color: #333;
  padding: 20px;
  border-radius: 10px;
  text-align: center;
}
#total-price {
  font-size: 24px;
  margin-top: 20px;
}
button {
  background: red;
  font-family: 'Orbitron', sans-serif;
  color: white;
  border: none;
  padding: 10px 20px;
  margin-top: 10px;
  cursor: pointer;
  border-radius: 5px;
  margin-right: 10px;
}
button:hover {
  background: cyan;
}
</style>
</head>
<body>
<header>MoviesX - Cart</header>
<nav class="nav">
  <a href="Movie.html">Home</a>
  <a href="products.html">Products</a>
  <a href="register.html">Register</a>
  <a href="login.html">Login</a>
  <div class="cart-icon" id="cart-icon">
    <i class="fa fa-shopping-cart"></i>

```

```

        <span id="cart-hover">View Cart</span>
    </div>
</nav>

<div class="cart-container" id="cart-container">
    <h2>Your Cart</h2>
    <div id="cart-info"></div>
    <div id="total-price"></div>
    <button onclick="clearCart()">Clear Cart</button>
    <button onclick="checkout()">Checkout</button>
</div>

<script>
function loadCart() {
    const cart = JSON.parse(localStorage.getItem('cart')) || [];
    const cartInfo = document.getElementById('cart-info');
    const totalPriceElement = document.getElementById('total-price');
    const cartHover = document.getElementById('cart-hover');

    cartHover.textContent = cart.length > 0 ? `View Cart (${cart.length} items)` : 'View Cart';

    if (cart.length === 0) {
        cartInfo.innerHTML = '<p>Your cart is empty.</p>';
        totalPriceElement.textContent = '';
        return;
    }

    let totalPrice = 0;
    cartInfo.innerHTML = '';

    cart.forEach(movie => {
        cartInfo.innerHTML += `<p>Movie: ${movie.name}</p><p>Genre: ${movie.genre}</p><p>Price: ₹${movie.price}</p><hr>`;
        totalPrice += parseInt(movie.price);
    });

    totalPriceElement.textContent = `Total Price: ₹${totalPrice}`;
}

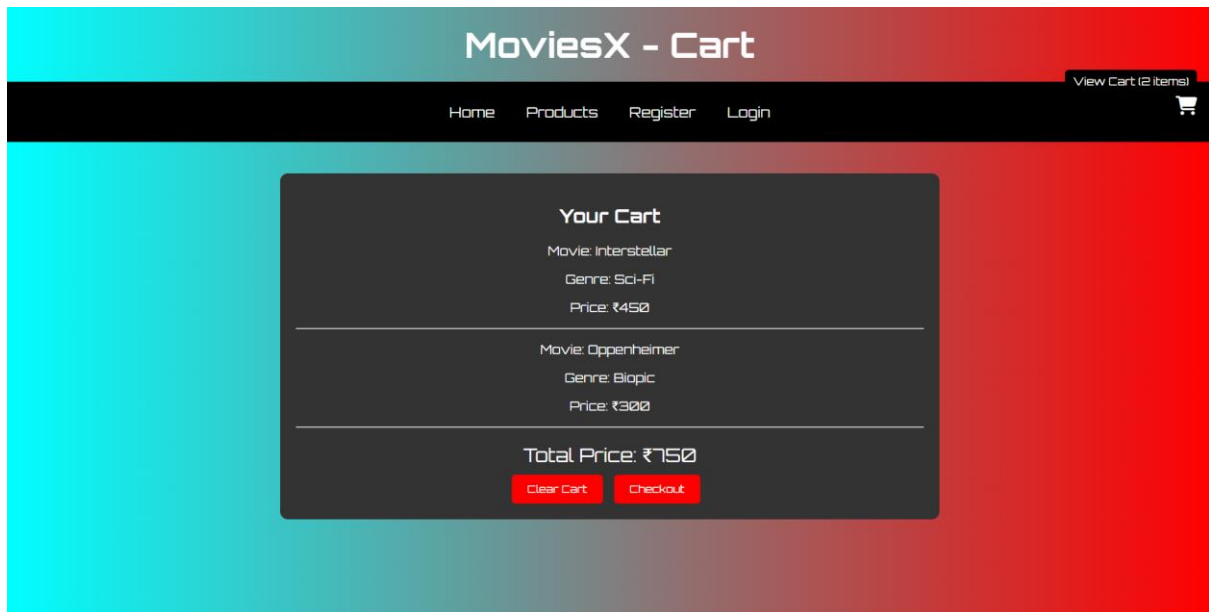
function clearCart() {
    localStorage.removeItem('cart');
    alert('Cart has been cleared!');
    loadCart();
}

function checkout() {
    const cart = JSON.parse(localStorage.getItem('cart')) || [];
    if (cart.length === 0) {
        alert('Your cart is empty. Add items to proceed with checkout.');
        return;
    }
    alert('Checkout successful! Thank you for your purchase.');
    clearCart();
}

window.onload = loadCart;
</script>
</body>
</html>

```

Output:



## Conclusion

Using JavaScript's **Web Storage API** significantly elevates the interactivity of **MoviesX**. Persistent login allows users to seamlessly access their movie dashboards, while the bookmarking feature enhances the content experience by allowing users to save and revisit their favorite movies.

These functionalities:

- Increase return visits and engagement by offering personalized movie lists
- Provide a smooth experience without relying on a server or backend database
- Enable powerful features in a front-end-heavy platform like **MoviesX**

In summary, JavaScript not only powers the interactivity of **MoviesX** but also ensures user preferences and sessions persist effectively—making the platform more personalized, efficient, and user-focused.



## Experiment No.7

### PHP Theory: User Registration Script for MoviesX – Online Movie Platform

#### Problem Statement

Develop a PHP script to handle user registration for the **MoviesX** platform website. The script should accept input from users for their name, email address, password, etc., validate the data, and store it securely. It should also provide appropriate feedback or error handling for user experience.

#### Introduction

User registration is a fundamental component of any interactive website, especially on a dynamic platform like **MoviesX**. PHP is a widely-used server-side language that facilitates form processing, database storage, and secure input validation.

On **MoviesX**, user registration enables personalized features like:

- Managing movie lists
- Bookmarking favorite movies
- Creating and updating user profiles
- Accessing personalized movie recommendations

#### Core Elements of the PHP Registration Script

##### 1. Form Handling

- The form captures details like name, email, password, and confirms the password.
- Data is submitted using the POST method to the PHP script.

##### 2. Validation

- Fields must not be empty.
- Email is validated with regex to ensure proper formatting.
- Password is checked for a minimum length and complexity.
- Password and Confirm Password must match.

##### 3. Password Hashing

- Before storing, passwords are hashed using `password_hash()` to ensure secure storage and prevent plaintext password exposure.

#### 4. Database Interaction

- The script uses MySQLi or PDO to connect with the MySQL database and store user data into the users table.
- Ensure to check if the email already exists to avoid duplicate accounts.

#### 5. Error Handling

- Displays messages for missing/invalid input, duplicate accounts, or connection failures.
- Detailed error messages should be shown if there is an issue with database insertion.

#### 6. User Feedback

- If registration is successful, the user is redirected to the login page or shown a success message.
- On failure, the script should display appropriate error messages (e.g., "Email already exists", "Password mismatch", etc.).

#### Code:

```
<?php

// Database connection

$servername = "localhost";

$username = "root";

$password = "";

$dbname = "moviesx";


// Create connection

$conn = new mysqli($servername, $username, $password, $dbname);


// Check connection

if ($conn->connect_error) {

    die("Connection failed: " . $conn->connect_error);

}


if ($_SERVER["REQUEST_METHOD"] == "POST") {

    // Collect and sanitize user inputs

    $name = mysqli_real_escape_string($conn, $_POST['name']);

    $email = mysqli_real_escape_string($conn, $_POST['email']);

    $password = mysqli_real_escape_string($conn, $_POST['password']);
```

```

$confirm_password = mysqli_real_escape_string($conn, $_POST['confirm_password']);

// Basic validation
if (empty($name) || empty($email) || empty($password) || empty($confirm_password)) {
    $error = "All fields are required!";
} elseif (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
    $error = "Invalid email format!";
} elseif ($password !== $confirm_password) {
    $error = "Passwords do not match!";
} elseif (strlen($password) < 6) {
    $error = "Password must be at least 6 characters!";
} else {
    // Check if email already exists
    $sql = "SELECT * FROM users WHERE email = '$email'";
    $result = $conn->query($sql);

    if ($result->num_rows > 0) {
        $error = "Email is already registered!";
    } else {
        // Hash the password
        $hashed_password = password_hash($password, PASSWORD_DEFAULT);

        // Insert user data into the database
        $sql = "INSERT INTO users (name, email, password) VALUES ('$name', '$email', '$hashed_password')";

        if ($conn->query($sql) === TRUE) {
            $success = "Registration successful! You can now log in.";
        } else {
            $error = "Something went wrong. Please try again.";
        }
    }
}

?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">

```

```
<title>Register - MoviesX</title>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<style>

  body {

    font-family: Arial, sans-serif;

    background-color: #f4f4f4;

    margin: 0;

    padding: 0;

  }

  .form-container {

    max-width: 400px;

    margin: 50px auto;

    background: white;

    padding: 30px;

    box-shadow: 0 0 15px rgba(0,0,0,0.1);

    border-radius: 10px;

  }

  h2 {

    text-align: center;

    color: #333;

  }

  label {

    display: block;

    margin-top: 15px;

    font-weight: bold;

  }

  input[type="text"], input[type="email"], input[type="password"], input[type="tel"] {

    width: 100%;

    padding: 10px;

    margin-top: 5px;

    border: 1px solid #ccc;

    border-radius: 5px;

  }

  button {

    margin-top: 20px;

    width: 100%;

    padding: 12px;

    background-color: #28a745;

    color: white;
```

```
        border: none;

        border-radius: 6px;

        font-size: 16px;

        cursor: pointer;
    }

    button:hover {

        background-color: #218838;

    }

    .message {

        margin-top: 15px;

        text-align: center;

        font-size: 14px;

    }

    .success { color: green; }

    .error { color: red; }

</style>

</head>

<body>

<div class="form-container">

    <h2>Create Your Account</h2>

    <!-- Display PHP success/error messages -->

    <?php if (isset($success)): ?>

        <div class="message success"><?= $success ?></div>

    <?php elseif (isset($error)): ?>

        <div class="message error"><?= $error ?></div>

    <?php endif; ?>

    <form method="POST" action="">

        <label for="name">Full Name</label>

        <input type="text" id="name" name="name" required>

        <label for="email">Email Address</label>

        <input type="email" id="email" name="email" required>

        <label for="password">Password</label>

        <input type="password" id="password" name="password" required>
```

```
<label for="confirm_password">Confirm Password</label>
```

```
<input type="password" id="confirm_password" name="confirm_password" required>
```

```
<button type="submit">Register</button>
```

```
</form>
```

```
<div class="message" style="text-align:center; margin-top: 15px;">
```

```
<a href="login.php">Already have an account? Login here</a>
```

```
</div>
```

```
</div>
```

```
</body>
```

```
</html>
```

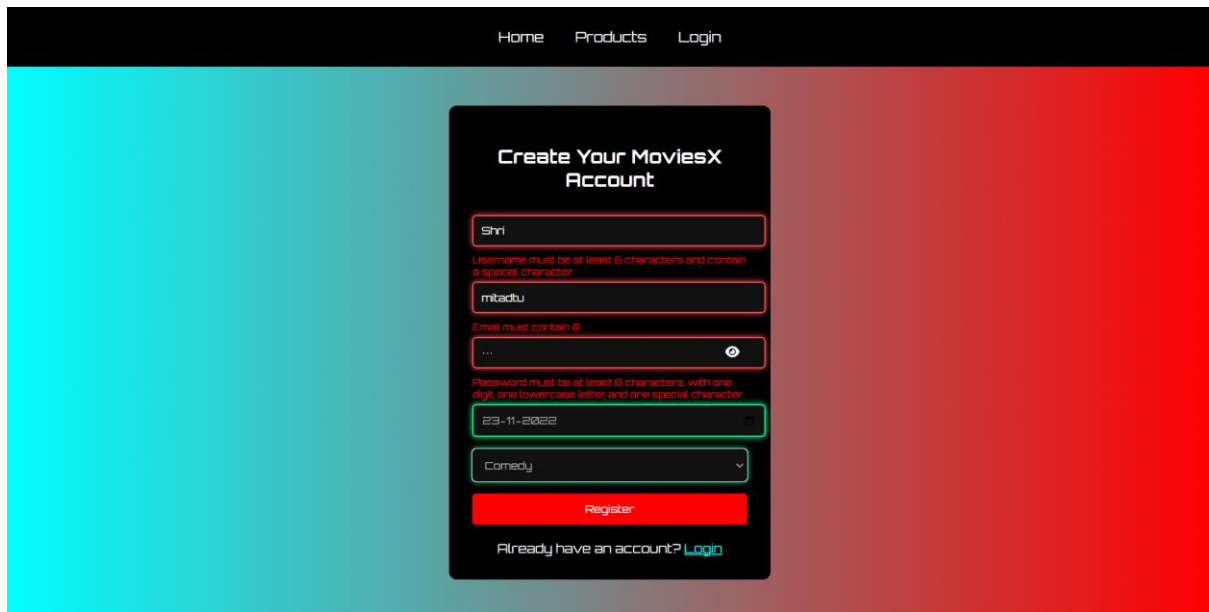
```
<?php
```

```
// Close database connection
```

```
$conn->close();
```

```
?>
```

## Output:



The screenshot displays a web registration form titled "Create Your MoviesX Account". The form is centered on a background with a blue-to-red gradient. At the top, a black navigation bar contains the links "Home", "Products", and "Login". The registration form itself is a dark grey card with the following fields and features:

- Username:** A text input field containing "Shri". Below it, a red error message states: "Username must be at least 6 characters and contain a special character".
- Email:** A text input field containing "mibadu". Below it, a red error message states: "Email must contain @".
- Password:** A text input field containing "23-11-2022". Below it, a red error message states: "Password must be at least 8 characters, with one digit, one lowercase letter and one special character".
- Gender:** A dropdown menu currently showing "Comedy".
- Register Button:** A prominent red button labeled "Register".
- Login Link:** A link labeled "Login" with the text "Already have an account?" preceding it.

## Conclusion:

Implementing a PHP-based registration system is essential for managing user identity on **MoviesX**. A well-structured registration script ensures secure account creation, input validation, and database integration, enabling a personalized experience for every user.

Through this PHP implementation:

- Users can register with valid credentials (name, email, password).
- Server-side validation guarantees that no incomplete or incorrect data is stored.
- Password hashing protects user data from breaches.
- Database integration (MySQL) ensures persistent storage of user details.
- Clear error messages and success feedback enhance user interaction and trust.

For **MoviesX**, this registration system lays the foundation for:

- User-authored movie reviews and ratings
- Commenting systems for movie discussions
- Personalized user dashboards
- Bookmarking or favouriting movies
- Saving preferences and watchlists, and more

This registration system ensures that users can seamlessly interact with the platform and access features that cater to their personalized needs.

## Experiment No. 8

PHP Theory: User Login Script for **MoviesX** – Online Movie Platform

### Problem Statement

Develop a PHP script to handle user login for the **MoviesX** platform.

The script should accept input for login credentials (email and password), validate the input, compare it with database records, and start a user session upon success. It must also provide feedback for success or failure.

### Introduction

The login system is a crucial feature of the **MoviesX** platform, enabling registered users to access personalized movie recommendations, post reviews, comment on movie discussions, and save their favorite movies. Using PHP for login management ensures secure backend communication, user session tracking, and robust error handling.

PHP login implementation involves:

- Capturing login credentials through a form
- Validating and sanitizing input
- Verifying credentials against the MySQL database
- Starting a session upon successful login
- Providing success or failure feedback

### Key Components of the Login System

#### 1. Form Handling

- Accepts email and password from the login form using the POST method.

#### 2. Input Validation

- Checks if fields are filled
- Verifies if the email format is correct
- Prevents SQL injection using prepared statements

#### 3. Password Verification

- Passwords are hashed during registration using `password_hash()`
- At login, `password_verify()` checks if the entered password matches the hash stored in the database

#### 4. Database Authentication



- Connects to MySQL
- Searches for a user with the provided email
- Verifies the password against the stored hash

## 5. Session Management

- Upon successful login, session\_start() stores the user's session data
- Enables access to personalized content like movie recommendations, reviews, and watchlists

## 6. User Feedback

- **If successful:** Redirects to the homepage or dashboard, displaying a welcome message with personalized content
- **If failed:** Displays relevant error messages, such as "Invalid credentials"

This login system ensures that **MoviesX** users can securely log in and access personalized features while maintaining a high level of security and user experience.

### Code:

```
<?php

// Registration script for MoviesX


// Include the database connection file
include('db_connection.php');


// Initialize variables
$username = $email = $password = $confirm_password = $phone = "";
$username_err = $email_err = $password_err = $confirm_password_err = $phone_err = "";


// Handle form submission
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Validate username
    if (empty(trim($_POST["username"]))) {
        $username_err = "Please enter your full name.";
    } else {
        $username = trim($_POST["username"]);
    }

    // Validate email
    if (empty(trim($_POST["email"]))) {
```

```

        $email_err = "Please enter your email address.";
    } else {
        $email = trim($_POST["email"]);
    }

    // Validate phone number
    if (empty(trim($_POST["phone"]))) {
        $phone_err = "Please enter your phone number.";
    } else {
        $phone = trim($_POST["phone"]);
    }

    // Validate password
    if (empty(trim($_POST["password"]))) {
        $password_err = "Please enter a password.";
    } elseif (strlen(trim($_POST["password"])) < 6) {
        $password_err = "Password must have at least 6 characters.";
    } else {
        $password = trim($_POST["password"]);
    }

    // Validate confirm password
    if (empty(trim($_POST["confirm_password"]))) {
        $confirm_password_err = "Please confirm your password.";
    } else {
        $confirm_password = trim($_POST["confirm_password"]);
        if ($password !== $confirm_password) {
            $confirm_password_err = "Passwords do not match.";
        }
    }

    // Check for any errors before inserting into the database
    if (empty($username_err) && empty($email_err) && empty($password_err) && empty($confirm_password_err) &&
    empty($phone_err)) {
        // Hash the password
        $password_hash = password_hash($password, PASSWORD_DEFAULT);

        // Prepare the SQL query
        $sql = "INSERT INTO users (username, email, password, phone) VALUES (?, ?, ?, ?)";
    }

```

```

if($stmt = $conn->prepare($sql)) {
    // Bind the parameters

    $stmt->bind_param("ssss", $username, $email, $password_hash, $phone);

    // Execute the statement
    if ($stmt->execute()) {
        // Redirect to login page if registration is successful
        header("Location: login.php");
    } else {
        echo "<div class='message error'>Something went wrong. Please try again.</div>";
    }

    // Close the statement
    $stmt->close();
}

// Close the connection
$conn->close();
}
?>

```

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Register - MoviesX</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            margin: 0;
            padding: 0;
        }
        .form-container {
            max-width: 400px;
            margin: 50px auto;

```

```
background: white;

padding: 30px;

box-shadow: 0 0 15px rgba(0,0,0,0.1);

border-radius: 10px;
}

h2 {

text-align: center;

color: #333;

}

label {

display: block;

margin-top: 15px;

font-weight: bold;

}

input[type="text"], input[type="email"], input[type="password"], input[type="tel"] {

width: 100%;

padding: 10px;

margin-top: 5px;

border: 1px solid #ccc;

border-radius: 5px;

}

button {

margin-top: 20px;

width: 100%;

padding: 12px;

background-color: #28a745;

color: white;

border: none;

border-radius: 6px;

font-size: 16px;

cursor: pointer;

}

button:hover {

background-color: #218838;

}

.message {

margin-top: 15px;

text-align: center;

font-size: 14px;
```

```
}

.success { color: green; }

.error { color: red; }

</style>

</head>

<body>

<div class="form-container">

  <h2>Create Your MoviesX Account</h2>

  <!-- Error messages -->

  <?php if (!empty($username_err)) { echo "<div class='message error'>$username_err</div>"; } ?>

  <?php if (!empty($email_err)) { echo "<div class='message error'>$email_err</div>"; } ?>

  <?php if (!empty($phone_err)) { echo "<div class='message error'>$phone_err</div>"; } ?>

  <?php if (!empty($password_err)) { echo "<div class='message error'>$password_err</div>"; } ?>

  <?php if (!empty($confirm_password_err)) { echo "<div class='message error'>$confirm_password_err</div>"; } ?>

  <form method="POST" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]); ?>">

    <label for="username">Full Name</label>

    <input type="text" id="username" name="username" value="<?php echo $username; ?>" required>

    <label for="email">Email</label>

    <input type="email" id="email" name="email" value="<?php echo $email; ?>" required>

    <label for="password">Password</label>

    <input type="password" id="password" name="password" required>

    <label for="confirm_password">Confirm Password</label>

    <input type="password" id="confirm_password" name="confirm_password" required>

    <label for="phone">Phone Number</label>

    <input type="tel" id="phone" name="phone" value="<?php echo $phone; ?>" required>

    <button type="submit">Register</button>

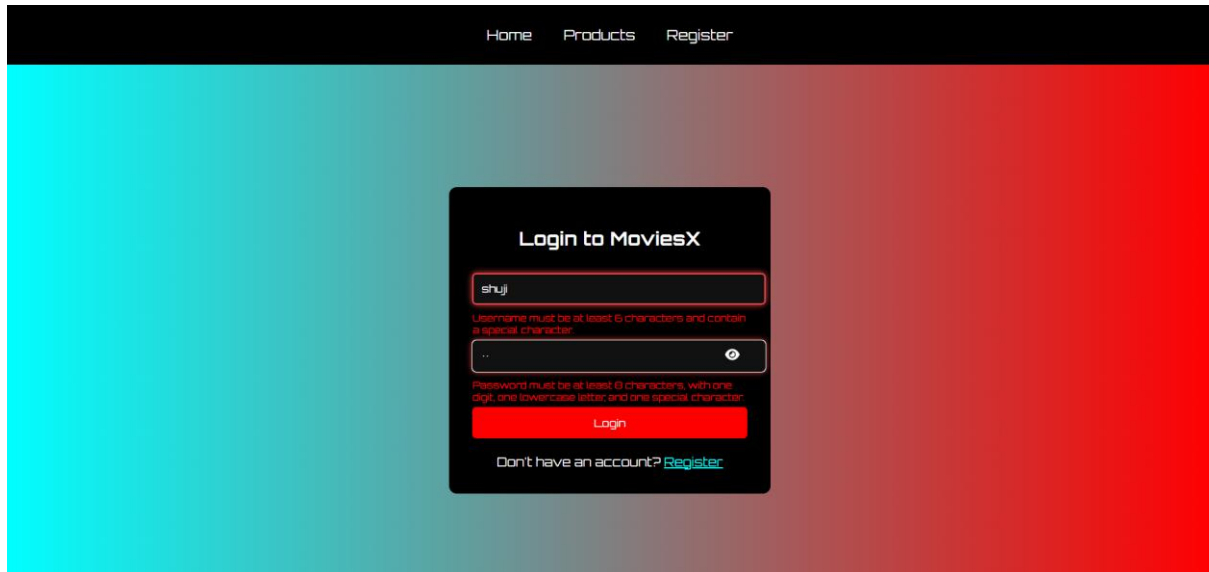
  </form>

</div>

</body>

</html>
```

Output:



## Conclusion

Implementing a PHP-based login system on MoviesX is critical for managing user access and enabling personalized functionality like movie reviews, bookmarking favorite movies, and customizing user dashboards.

With this PHP login system:

- **Users securely log in** using their credentials (email and password).
- **Sessions** maintain their authenticated state across pages.
- **Passwords are verified** using hashed encryption for enhanced security.
- Users are provided with **immediate feedback**, improving usability.

The login system also serves as the foundation for additional features like posting comments, interacting with content, and managing personal settings within MoviesX.

## **Experiment No. 9**

### **PHP Theory: Bookmark (Watch Later) Management System for MoviesX**

#### **A. Problem Statement**

**Develop a PHP system for MoviesX that allows users to:**

- **Add movie posts to their bookmark or "Watch Later" list.**
- **View saved movie posts that they have bookmarked.**
- **Remove bookmarked posts if desired.**

**The bookmark data can be stored temporarily via session or permanently in a MySQL database for logged-in users.**

#### **Theory: Bookmark (Watch Later) Feature in MoviesX**

**Bookmarks are an essential component of content platforms like MoviesX. They allow users to save their favorite movies for later viewing, revisit them conveniently, and manage their watchlist easily.**

#### **Two Bookmark Management Methods for MoviesX**

##### **A. Session-Based Bookmarking (Without MySQL)**

**This method uses PHP \$\_SESSION to temporarily store bookmarked movie IDs.**

##### **Key Characteristics:**

- **Bookmarks are stored in session memory.**
- **Quick and doesn't require user login.**
- **Data is lost once the session expires or the browser is closed.**

##### **Operations Supported:**

- **Add: Save movie ID to session.**
- **View: Display details using stored movie IDs.**
- **Remove: Unset movie ID from session array.**

##### **Pros:**

- **Easy to implement.**
- **Good for guest users who don't want to log in.**

##### **Cons:**

- **Not persistent across sessions or devices.**
- **Cannot be used for logged-in users to access their bookmarks across devices.**

## **B. Database-Based Bookmarking (With MySQL)**

**A persistent solution using a bookmarks table in MySQL for logged-in users.**

### **Key Characteristics:**

- **Requires user login (user ID linkage).**
- **Saves movie ID and user ID in the database.**
- **Persistent across sessions and devices.**

### **Operations Supported:**

- **Add: Insert movie ID for logged-in user.**
- **View: Fetch details using SQL JOIN with movies table.**
- **Remove: Delete entry by user ID and movie ID.**

### **Pros:**

- **Persistent and scalable across user sessions and devices.**
- **Supports personalized bookmark management, even across different devices.**

### **Cons:**

- **Requires login system for users to track their bookmarks.**
- **Needs proper error and session handling to ensure data integrity.**

### **Code:**

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Shopping Cart</title>

  <style>

    body {

      font-family: Arial, sans-serif;

      margin: 0;

      padding: 0;

      text-align: center;
```



```
background-color: #f5f5f5;  
}
```

```
.navbar {  
  background-color: #333;  
  padding: 15px;  
}
```

```
.navbar a {  
  color: white;  
  text-decoration: none;  
  padding: 14px 20px;  
  display: inline-block;  
}
```

```
.navbar a:hover {  
  background-color: #575757;  
}
```

```
.cart-container {  
  width: 60%;  
  margin: 40px auto;  
  padding: 20px;  
  background: white;  
  box-shadow: 0 0 10px rgba(0,0,0,0.1);  
  border-radius: 8px;  
  text-align: left;  
}
```

```
.cart-container h1 {  
  text-align: center;  
}
```

```
.cart-item {  
  display: flex;  
  justify-content: space-between;  
  align-items: center;  
  padding: 12px;  
  border-bottom: 1px solid #ddd;
```

```
}
```

```
.cart-item:last-child {  
    border-bottom: none;  
}
```

```
.total {  
    font-size: 1.2em;  
    font-weight: bold;  
    padding-top: 20px;  
    text-align: right;  
}
```

```
.remove-button {  
    background-color: red;  
    color: white;  
    border: none;  
    padding: 6px 12px;  
    border-radius: 4px;  
    cursor: pointer;  
}
```

```
.remove-button:hover {  
    background-color: darkred;  
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<div class="navbar">
```

```
<a href="index.html">Home</a>
```

```
<a href="product.html">Products</a>
```

```
<a href="about.html">About Us</a>
```

```
<a href="register.html">Login/Register</a>
```

```
<a href="cart.html">Cart (<span class="cart-count">0</span>)</a>
```

```
</div>
```

```
<div class="cart-container">
```

```
<h1>Your Shopping Cart</h1>
```

```

<div id="cart-items"></div>

<p class="total">Total: $<span id="total-price">0.00</span></p>

</div>

<script>

let cart = JSON.parse(localStorage.getItem('cart')) || [];

const cartCount = document.querySelector('.cart-count');

const cartItemsContainer = document.getElementById('cart-items');

const totalPriceEl = document.getElementById('total-price');

function updateCartDisplay() {

  cartItemsContainer.innerHTML = '';

  let totalPrice = 0;

  cart.forEach((item, index) => {

    totalPrice += item.price;

    const cartItem = document.createElement("div");

    cartItem.classList.add("cart-item");

    cartItem.innerHTML = `

      <span>${item.name} - ${item.price.toFixed(2)}</span>

      <button class="remove-button" onclick="removeItem(${index})">Remove</button>

    `;

    cartItemsContainer.appendChild(cartItem);

  });

  totalPriceEl.textContent = totalPrice.toFixed(2);

  cartCount.textContent = cart.length;

  localStorage.setItem('cart', JSON.stringify(cart));

}

function removeItem(index) {

  cart.splice(index, 1);

  updateCartDisplay();

}

window.onload = updateCartDisplay;

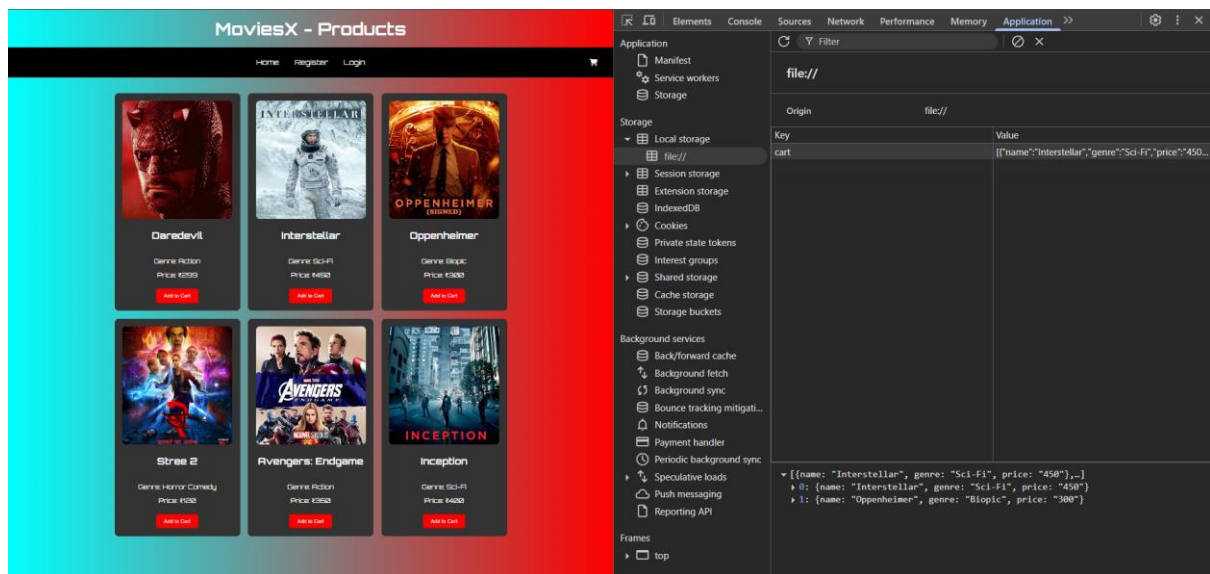
</script>

</body>

</html>

```

Output:



## Conclusion – MoviesX Watch Later System

The “**Watch Later**” feature enhances the **MoviesX platform** by:

- Allowing users to save movies or shows they’re interested in watching later
- Encouraging repeat visits and longer user engagement
- Supporting content personalization and improved discovery

**Session-based bookmarks** work well for guest users looking for a quick way to mark content temporarily.

In contrast, **MySQL-based bookmarks** offer a permanent, personalized solution for logged-in users.

For a real-world movie streaming platform like **MoviesX**, the **database-driven (MySQL) approach** is strongly recommended to ensure data persistence, scalability, and a tailored user experience across sessions and devices.

## Experiment No. 10

### PHP Theory: Movie Submission (Watchlist or Publish Analogy) for MoviesX Platform

#### A. Problem Statement

Develop a **PHP script** that allows users to **submit movie entries** (e.g., suggestions, reviews, or watchlist items), validating and storing the data in a **MySQL database**. Feedback should be displayed for both **successful and failed submissions**.

#### Theory: Movie Post Submission System

In **MoviesX**, the movie submission process functions like a **checkout system**—it transitions a drafted or composed movie entry into **published content** visible to others or stored in a user's profile.

#### Two Types of Movie Submission

##### A. Session-Based Draft Movie Submission

- The user creates a movie post (e.g., review or suggestion)
- Content is temporarily stored in **PHP session variables**
- On final submit, the post may be **previewed** and then optionally **stored into the database**

##### Pros:

- Simple, suitable for demos or preview drafts
- No immediate need for a database connection

##### Cons:

- Data is lost when the session ends or browser closes
- Not suitable for permanent or multi-device storage

##### B. Database-Based Submission

Movie posts are **inserted into a MySQL table** on submission. This enables long-term storage, editing, and moderation.

##### Workflow:

1. **Check if user is logged in** (using session or token)
2. **Validate inputs** such as title, description, genre, or movie ID

3. **Sanitize inputs** to protect against **XSS** or **SQL Injection**
4. **Insert into the movie\_post (or suggestions/reviews) table** with a reference to the user's ID
5. **Return success or error response** and provide feedback

#### **Benefits:**

- Persistent and structured
- Ideal for logged-in users and community interaction
- Allows feature enhancements like editing, reporting, or rating posts
- 

#### **Code:**

```
-- Create users table

CREATE TABLE users (

  id INT AUTO_INCREMENT PRIMARY KEY,

  email VARCHAR(100) NOT NULL UNIQUE,

  password VARCHAR(255) NOT NULL

);

-- Create movies table

CREATE TABLE movies (

  id INT AUTO_INCREMENT PRIMARY KEY,

  title VARCHAR(100) NOT NULL,

  director VARCHAR(100),

  genre VARCHAR(50),

  release_year INT,

  description TEXT,

  poster_url VARCHAR(255),

  price DECIMAL(10, 2) NOT NULL,

  quantity INT DEFAULT 0,

  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP

);

<?php

// Start session

session_start();

// Database connection

$host = "localhost";

$user = "root";    // Your DB username

$pass = "";        // Your DB password
```

```

$db = "moviesx"; // Your DB name

$conn = new mysqli($host, $user, $pass, $db);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Handle POST request for user registration
if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST['register'])) {
    // Fetch and sanitize inputs
    $name = trim($_POST["name"]);
    $email = trim($_POST["email"]);
    $password = password_hash($_POST["password"], PASSWORD_BCRYPT);
    $phone = trim($_POST["phone"]);

    // Basic validation
    if (!empty($name) && !empty($email) && !empty($password)) {
        // Insert query for user registration
        $sql = "INSERT INTO users (name, email, password, phone) VALUES (?, ?, ?, ?)";
        $stmt = $conn->prepare($sql);
        $stmt->bind_param("ssss", $name, $email, $password, $phone);

        if ($stmt->execute()) {
            echo "<script>
                document.getElementById('successMsg').style.display = 'block';
                setTimeout(() => window.location.href = 'login.html', 2000);
            </script>";
        } else {
            echo "<script>document.getElementById('errorMsg').style.display = 'block';</script>";
        }
        $stmt->close();
    } else {
        echo "<script>document.getElementById('errorMsg').style.display = 'block';</script>";
    }
}

// Handle POST request for adding a movie

```

```

if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST['add_movie'])) {

    // Fetch and sanitize inputs

    $title      = trim($_POST["title"]);
    $director   = trim($_POST["director"]);
    $genre      = trim($_POST["genre"]);
    $release_year = intval($_POST["release_year"]);
    $description = trim($_POST["description"]);
    $poster_url  = trim($_POST["poster_url"]);
    $price       = floatval($_POST["price"]);
    $quantity    = intval($_POST["quantity"]);


    // Basic validation

    if (!empty($title) && !empty($genre) && !empty($price) && !empty($quantity)) {

        // Insert query for adding movie

        $sql = "INSERT INTO movies (title, director, genre, release_year, description, poster_url, price, quantity)
                VALUES (?, ?, ?, ?, ?, ?, ?, ?)";

        $stmt = $conn->prepare($sql);

        $stmt->bind_param("ssssssdi", $title, $director, $genre, $release_year, $description, $poster_url, $price, $quantity);

        if ($stmt->execute()) {

            echo "<script>

                document.getElementById('successMsg').style.display = 'block';

                setTimeout(() => window.location.href = 'movies_list.html', 2000);

            </script>";

        } else {

            echo "<script>document.getElementById('errorMsg').style.display = 'block';</script>";

        }

        $stmt->close();

    } else {

        echo "<script>document.getElementById('errorMsg').style.display = 'block';</script>";

    }

}

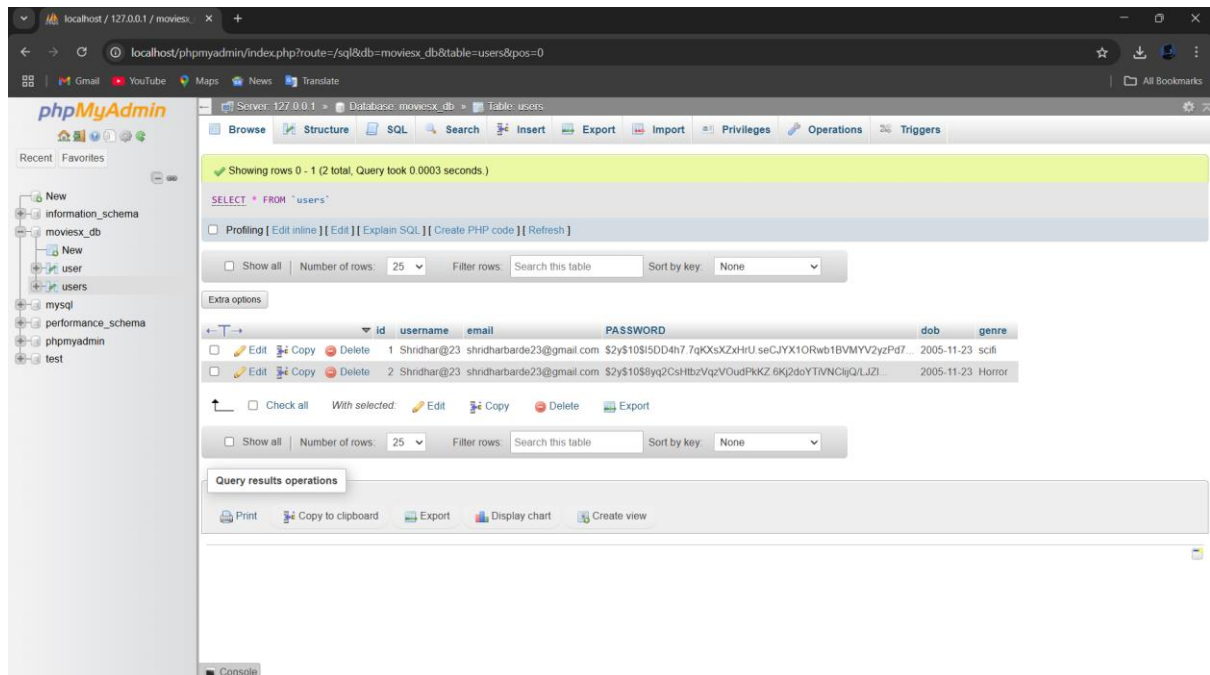
$conn->close();

?>

```



Output:



## Conclusion – MoviesX Submission System

The **movie post submission system** is central to the MoviesX platform, allowing content creators (movie directors, producers, and users) to:

- **Submit and manage movie details securely**
- **Store movie entries as drafts or publish them immediately**
- **Receive confirmation of successful submission**

By utilizing a **database-driven approach**, all movie data is:

- **Securely stored** in the database
- **Linked to individual users** for ownership and easy management
- **Ready for display** on listing pages and available for browsing by other users

This process transforms MoviesX from a static movie catalog into a **dynamic, interactive platform**, where users can not only browse movies but also contribute content, creating an engaging experience for all users.