VIDYA JYOTHI INSTITUTE OF TECHNOLOGY

(An Autonomous Institution)

(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)

B.Tech(CSE-DS) IV Year / I Semester (R22)

LAB MANUAL

Name of the Faculty	**KISHORE K**
Department	**CSE(Data Science)**
Year & Semester	**B.Tech-IV & I Sem**
Regulation	**R22**
Lab Name	**FULL STACK DEVELOPMENT (Professional Elective - III)**

DEPARTMENT OF CSE (Data Science)

VIDYA JYOTHI INSTITUTE OF TECHNOLOGY

(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)

An Autonomous Institution

AZIZ NAGAR, C B POST, HYDERABAD-500075

2025-2026

INDEX

Week-11 & 12		
11.	**CASE STUDY-1:** Create a Chat module/Interface using HTML CSS and JavaScript. The chat interface primarily consists of two segments: the message header and the chat box. **Message-Header-** The message header resides at the top of the chat box. It includes the user's name, avatar or profile image, and the user's last seen. Last seen is the last time the user was active. **The Chat-Box-** The chat box consists of the message page and the message bottom sections. • ***Message page-***The message page consists of incoming and outgoing messages, as well as the avatars of the senders. It also displays the time at which each message is sent. • ***The Message-Bottom-***This section contains an input field where the user can type in the messages and a send button to send them.	

1. **a)** Create a Web Page using HTML which contains a Heading, Image and 2 hyperlinks. Each hyperlink opens a new page in the same web browser. New page contains "Go Back" link that takes you to the main page.

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Main Page</title>
</head>
<body>
  <header>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            background-color: #f4f4f4;
        }
        .container {
            width: 50%;
            margin: 10px auto;
            background-color: #fff;
            padding: 20px;
            border-radius: 8px;
            box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
        }

            img {
              width: 100%;
            border: none;
            border-radius: 4px;
         }
.topnav {
  overflow: hidden;
  background-color: #3498db;
}
```

```
.topnav a {
  float: left;
  color: #f2f2f2;
  text-align: center;
  padding: 16px 18px;
  text-decoration: none;
  font-size: 19px;
}

.topnav a:hover {
  background-color: #ebdef0;
  color: black;
}

.topnav a.active {
  background-color: #f7f45d;
  color: black;
}
    </style>
  </header>
<body>
<div class="container">
        <img src="banner.JPG" />

     <div class="topnav">
  <a class="active" href="index.html">Home</a>
   <a href="registration.html">Registration</a>
  <a href="login.html">Login</a>

</div>
<div style="padding-left:16px">
  <h2>Welcome to VJIT, Hyderabad</h2>
  <p>Dept. of CSE(DS)</p>
</div>
    </div>
 </body>
</html>
```

login.html

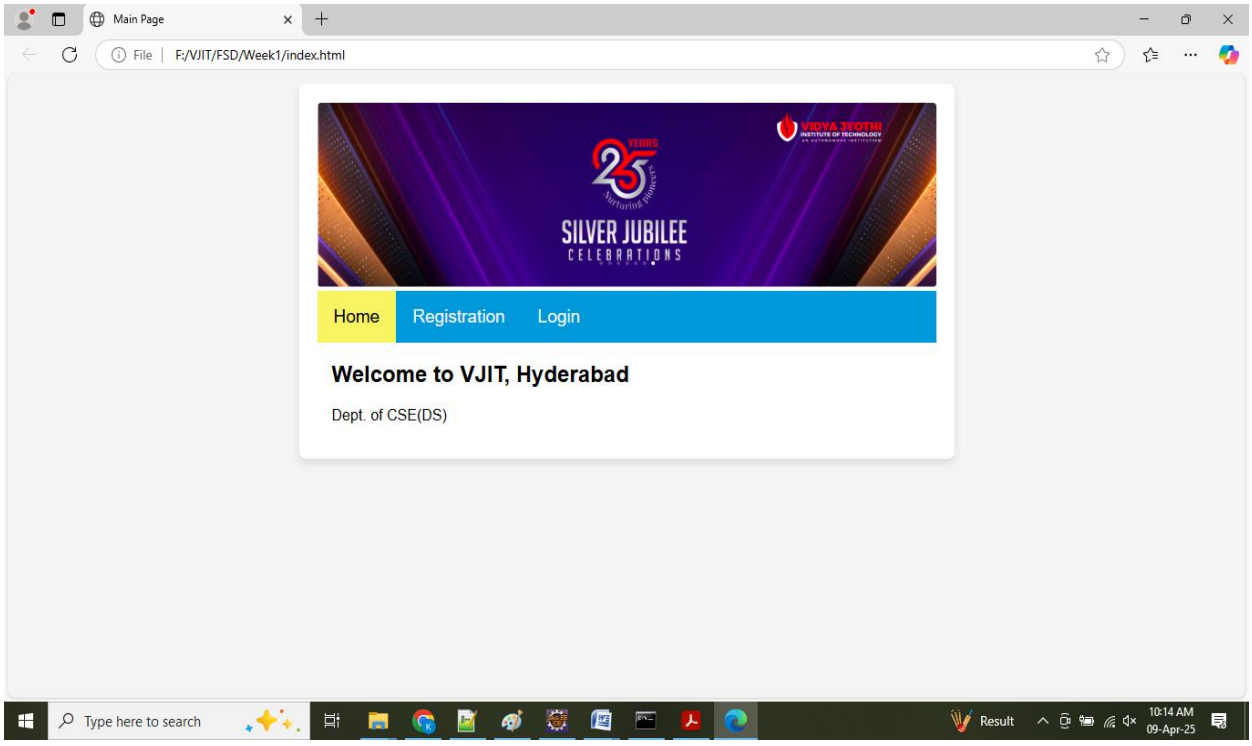
```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Login Page</title>
   <style>
      body {
         font-family: Arial, sans-serif;
         margin: 0;
         padding: 0;
         background-color: #f4f4f4;
      }
      .container {
         width: 50%;
         margin: 10px auto;
         background-color: #fff;
         padding: 20px;
         border-radius: 8px;
         box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
      }
      h2 {
         text-align: center;
         margin-bottom: 20px;
                  width: 98%;
         padding: 10px;
         background-color: #b91df5;
         color: white;
         font-size: 22px;
         border: none;
         border-radius: 4px;
      }
      img {
            width: 100%;
         border: none;
         border-radius: 4px;

      }
```

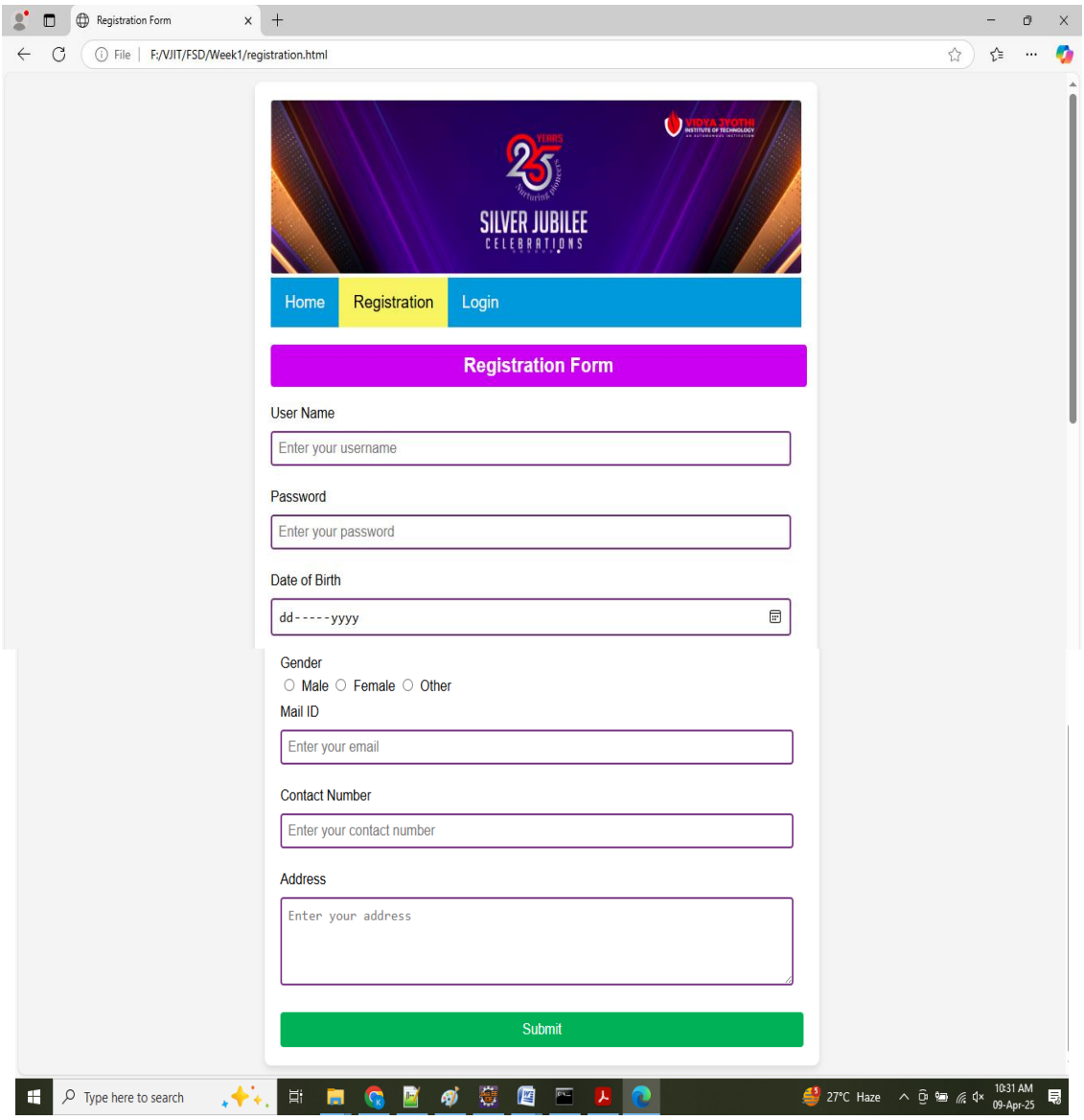
```
.topnav {
  overflow: hidden;
  background-color: #3498db;
}

.topnav a {
  float: left;
  color: #f2f2f2;
  text-align: center;
  padding: 16px 18px;
  text-decoration: none;
  font-size: 19px;
}

.topnav a:hover {
  background-color: #ebdef0;
  color: black;
}

.topnav a.active {
  background-color: #f7f45d;
  color: black;
}
        label {
            display: block;
            margin: 10px 0 5px;
        }

 input[type="text"], input[type="password"], input[type="email"], input[type="tel"],
input[type="date"], textarea, select {
            width: 95%;
            padding: 8px;
            margin: 5px 0 15px;
            border: 2px solid #ddd;
            border-radius: 4px;
                        border-color: #6c3483;
            font-size: 16px;
        }
```

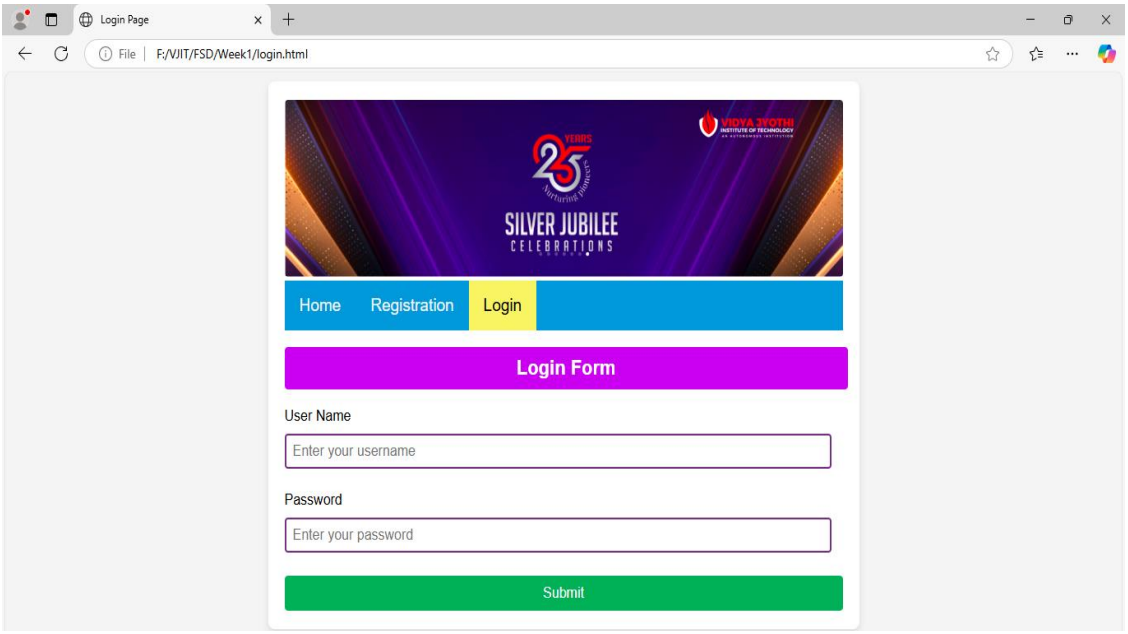
```
        .form-group {
            margin-bottom: 10px;
        }

        input[type="radio"], input[type="checkbox"] {
            margin-right: 5px;
        }

        .submit-btn{
            width: 100%;
            padding: 10px;
            background-color: #4CAF50;
            color: white;
            font-size: 16px;
            border: none;
            border-radius: 4px;
            cursor: pointer;
        }

        .submit-btn:hover {
            background-color: #45a049;
        }
    </style>
</head>
<body>

    <div class="container">
        <img src="banner.JPG" />
                <div class="topnav">
  <a  href="index.html">Home</a>
   <a href="registration.html">Registration</a>
  <a class="active" href="login.html">Login</a>

</div>
                <h2>Login Form</h2>
        <form action="#" method="POST">

            <div class="form-group">
                <label for="username">User Name</label>
```

```
            <input type="text" id="username" name="username" placeholder="Enter your username" required>
        </div>

        <div class="form-group">
            <label for="password">Password</label>
            <input type="password" id="password" name="password" placeholder="Enter your password" required>
        </div>


        <button type="submit" class="submit-btn">Submit</button>
    </form>
  </div>

</body>
</html>
```

Output




Registration Form
File | F:/VJIT/FSD/Week1/registration.html
SILVER JUBILEE
CELEBRATIONS
Home
Registration
Login
Registration Form
User Name
Enter your username
Password
Enter your password
Date of Birth
dd-----yyyy
Gender
Male
Female
Other
Mail ID
Enter your email
Contact Number
Enter your contact number
Address
Enter your address
Submit
Type here to search
27°C Haze
10:31 AM
09-Apr-25


Login Page
File | F:/VJIT/FSD/Week1/login.html
SILVER JUBILEE
CELEBRATIONS
Home
Registration
Login
Login Form
User Name
Enter your username
Password
Enter your password
Submit

1. b) Write a HTML program to create a Registration form, which contains User Name, Password, Date of Birth, Gender, Mail-id, Contact number, Address and submit button.

registration.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Registration Form</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            background-color: #f4f4f4;
        }

        .container {
            width: 50%;
            margin: 10px auto;
            background-color: #fff;
            padding: 20px;
            border-radius: 8px;
            box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
        }

        h2 {
            text-align: center;
            margin-bottom: 20px;
                        width: 98%;
            padding: 10px;
            background-color: #b91df5;
            color: white;
            font-size: 22px;
            border: none;
            border-radius: 4px;

        }
```

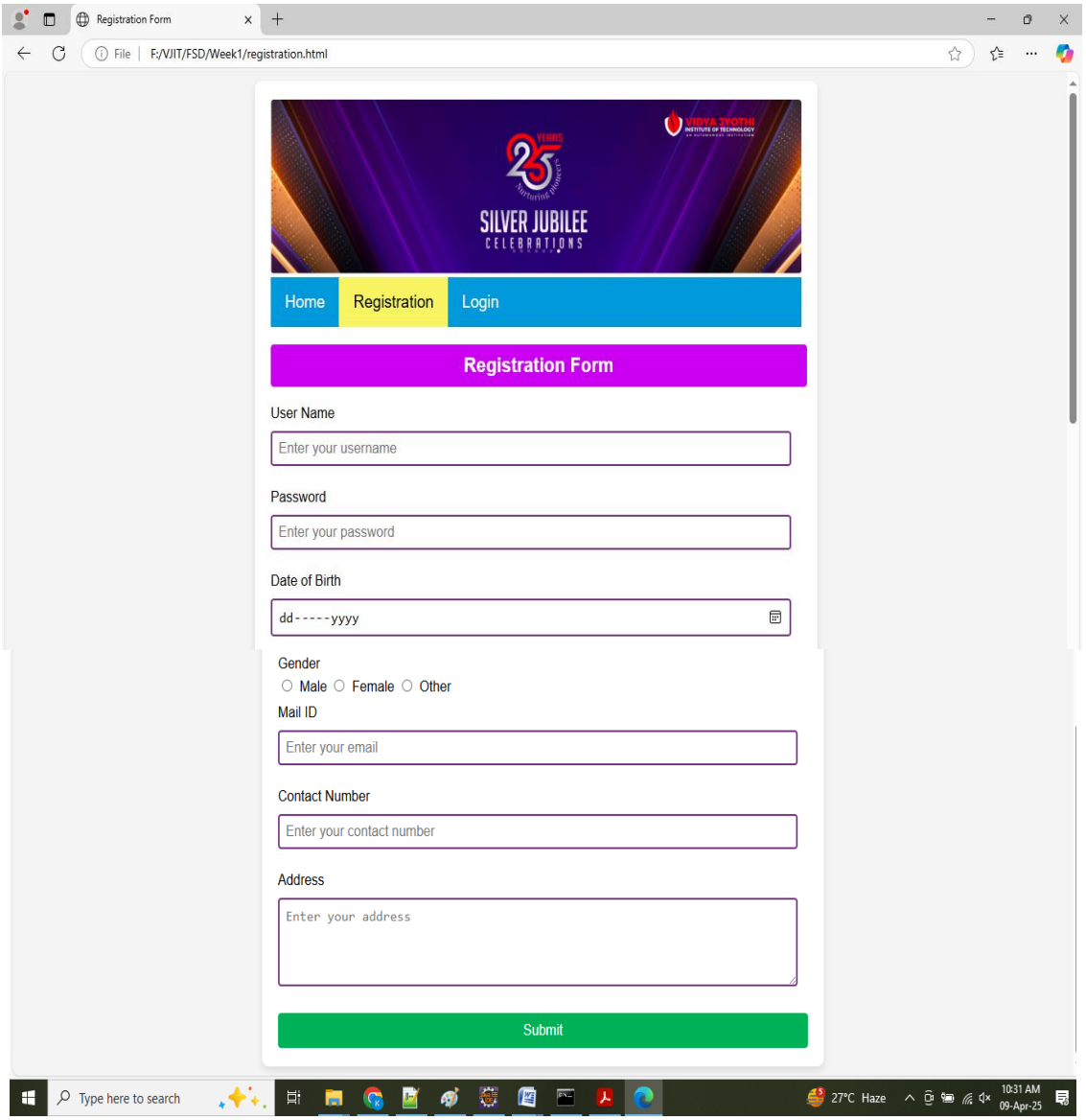
```
        img {
                width: 100%;
            border: none;
            border-radius: 4px;
        }
                .topnav {
  overflow: hidden;
  background-color: #3498db;
}
.topnav a {
  float: left;
  color: #f2f2f2;
  text-align: center;
  padding: 16px 18px;
  text-decoration: none;
  font-size: 19px;
}
.topnav a:hover {
  background-color: #ebdef0;
  color: black;
}
.topnav a.active {
  background-color: #f7f45d;
  color: black;
}
        label {
            display: block;
            margin: 10px 0 5px;
        }
        input[type="text"], input[type="password"], input[type="email"],
input[type="tel"], input[type="date"], textarea, select {
            width: 95%;
            padding: 8px;
            margin: 5px 0 15px;
            border: 2px solid #ddd;
            border-radius: 4px;
                    border-color: #6c3483;
            font-size: 16px;
        }
```

```
        .form-group {
            margin-bottom: 10px;
        }

        input[type="radio"], input[type="checkbox"] {
            margin-right: 5px;
        }

        .submit-btn{
            width: 100%;
            padding: 10px;
            background-color: #4CAF50;
            color: white;
            font-size: 16px;
            border: none;
            border-radius: 4px;
            cursor: pointer;
        }

        .submit-btn:hover {
            background-color: #45a049;
        }
    </style>
</head>
<body>

    <div class="container">
        <img src="banner.JPG" />
                <div class="topnav">
  <a  href="index.html">Home</a>
   <a class="active" href="registration.html">Registration</a>
  <a href="login.html">Login</a>

</div>
                <h2>Registration Form</h2>
        <form action="#" method="POST">

            <div class="form-group">
                <label for="username">User Name</label>
```

```
            <input type="text" id="username" name="username" placeholder="Enter your username" required>
        </div>

        <div class="form-group">
            <label for="password">Password</label>
            <input type="password" id="password" name="password" placeholder="Enter your password" required>
        </div>

        <div class="form-group">
            <label for="dob">Date of Birth</label>
            <input type="date" id="dob" name="dob" required>
        </div>

        <div class="form-group">
            <label>Gender</label>
            <input type="radio" id="male" name="gender" value="Male" required> Male
            <input type="radio" id="female" name="gender" value="Female"> Female
            <input type="radio" id="other" name="gender" value="Other"> Other
        </div>

        <div class="form-group">
            <label for="email">Mail ID</label>
            <input type="email" id="email" name="email" placeholder="Enter your email" required>
        </div>

        <div class="form-group">
            <label for="contact">Contact Number</label>
            <input type="tel" id="contact" name="contact" placeholder="Enter your contact number" required>
        </div>

        <div class="form-group">
            <label for="address">Address</label>
```

```
            <textarea id="address" name="address" rows="4" placeholder="Enter your address" required></textarea>
        </div>

        <button type="submit" class="submit-btn">Submit</button>
    </form>
  </div>

</body>
</html>
```

Output:



2. a) Create a web page to demonstrate Position Property in CSS.

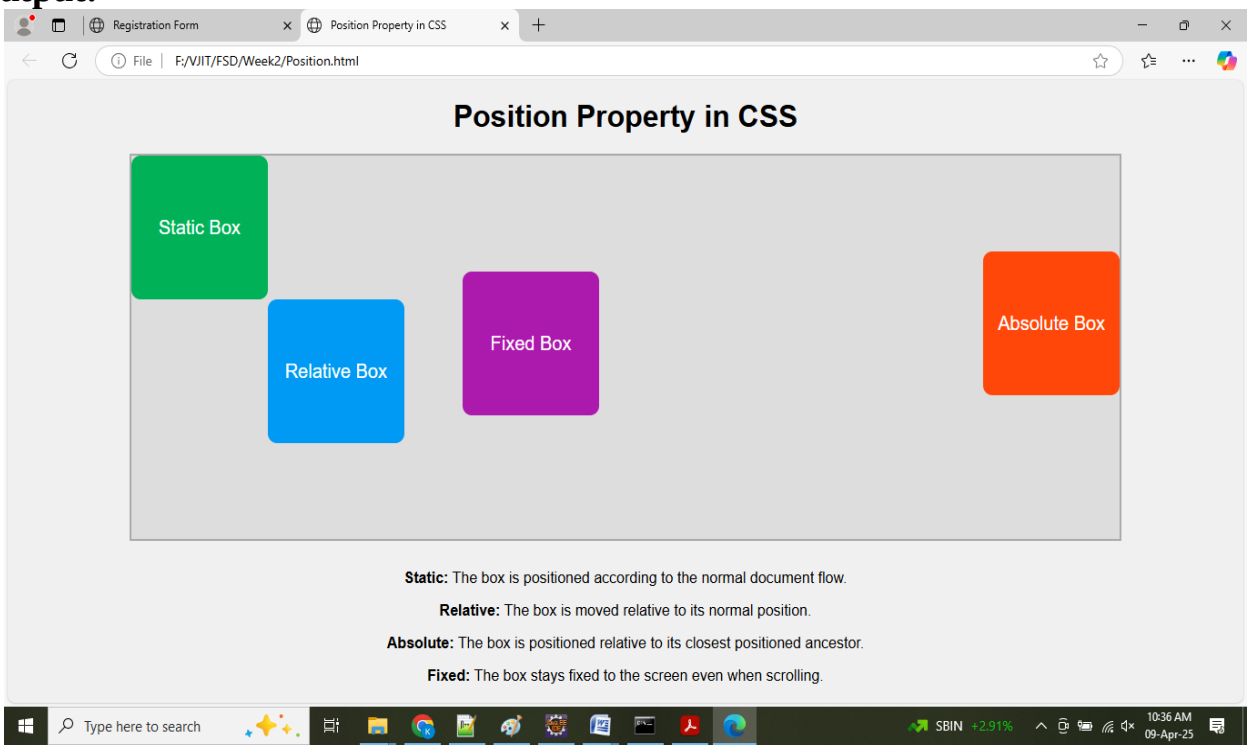
Position.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Position Property in CSS</title>
   <style>
      body {
         font-family: Arial, sans-serif;
         margin: 0;
         padding: 0;
         background-color: #f0f0f0;
      }

      h1 {
         text-align: center;
         margin: 20px 0;
      }

      .container {
         width: 80%;
         margin: 0 auto;
         position: relative;
         height: 400px;
         background-color: #ddd;
         border: 2px solid #aaa;
      }

      .box {
         width: 150px;
         height: 150px;
         color: white;
         text-align: center;
         line-height: 150px;
         font-size: 20px;
         border-radius: 10px;
         position: absolute;
      }
      /* Static Position */
      .static-box {
         background-color: #4CAF50;
      }
```

```
        /* Relative Position */
        .relative-box {
            background-color: #2196F3;
            position: relative;
            top: 150px;
            left: 150px;
        }

        /* Absolute Position */
        .absolute-box {
            background-color: #FF5733;
            position: absolute;
            top: 100px;
            right: 0;
        }

        /* Fixed Position */
        .fixed-box {
            background-color: #9C27B0;
            position: fixed;
            top: 200px;
            left: 500px;
        }

        .info {
            text-align: center;
            margin-top: 30px;
        }
    </style>
</head>
<body>

    <h1>Position Property in CSS</h1>

    <div class="container">
        <!-- Static position: Default positioning -->
        <div class="box static-box">Static Box</div>

        <!-- Relative position: Positioned relative to its normal position -->
        <div class="box relative-box">Relative Box</div>

    <!-- Absolute position: Positioned relative to its closest positioned ancestor -->
        <div class="box absolute-box">Absolute Box</div>
    </div>
```

```
        <!-- Fixed Position: Positioned relative to the browser window -->
        <div class="box fixed-box">Fixed Box</div>

        <div class="info">
    <p><strong>Static:</strong> The box is positioned according to the normal document flow.</p>
        <p><strong>Relative:</strong> The box is moved relative to its normal position.</p>
        <p><strong>Absolute:</strong> The box is positioned relative to its closest positioned ancestor.</p>
      <p><strong>Fixed:</strong> The box stays fixed to the screen even when scrolling.</p>
        </div>

        </body>
        </html>
```

Output:



2.b) Create a Newspaper Style Design to print minimum 2 articles using HTML and CSS.

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Newspaper Style Design</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>

  <div class="newspaper-container">
  <img src="banner.JPG"/>
    <!-- Header Section -->
    <header class="newspaper-header">
      <h1>Daily News</h1>
      <p class="date">March 26, 2025</p>
    </header>

    <!-- Articles Section -->
    <div class="articles-container">
      <article class="article">
        <h2 class="article-title">Breaking News: Technology Revolutionizing the World</h2>
        <p class="article-date">By John Doe, March 26, 2025</p>
        <p class="article-content">
          In the past decade, technology has drastically changed the way we live, work, and communicate. From smartphones to artificial intelligence, the rapid advancements have paved the way for new industries, businesses, and career opportunities. Experts predict that in the coming years, technology will continue to shape our society in unimaginable ways, impacting everything from healthcare to transportation.
        </p>
      </article>

      <article class="article">
        <h2 class="article-title">Climate Change: A Growing Concern for Future Generations</h2>
        <p class="article-date">By Jane Smith, March 26, 2025</p>
        <p class="article-content">
          As global temperatures rise and extreme weather events become more frequent, the issue of climate change is gaining increasing attention worldwide. Governments,
```

scientists, and environmental groups are calling for immediate action to mitigate the damage and protect the planet for future generations. Experts emphasize the need for sustainable practices, renewable energy, and global cooperation to combat climate change and its devastating effects.

```
        </p>
      </article>
    </div>

    <!-- Footer Section -->
    <footer class="newspaper-footer">
      <p>&copy; 2025 Daily News. All rights reserved.</p>
    </footer>
  </div>
</body>
</html>
```

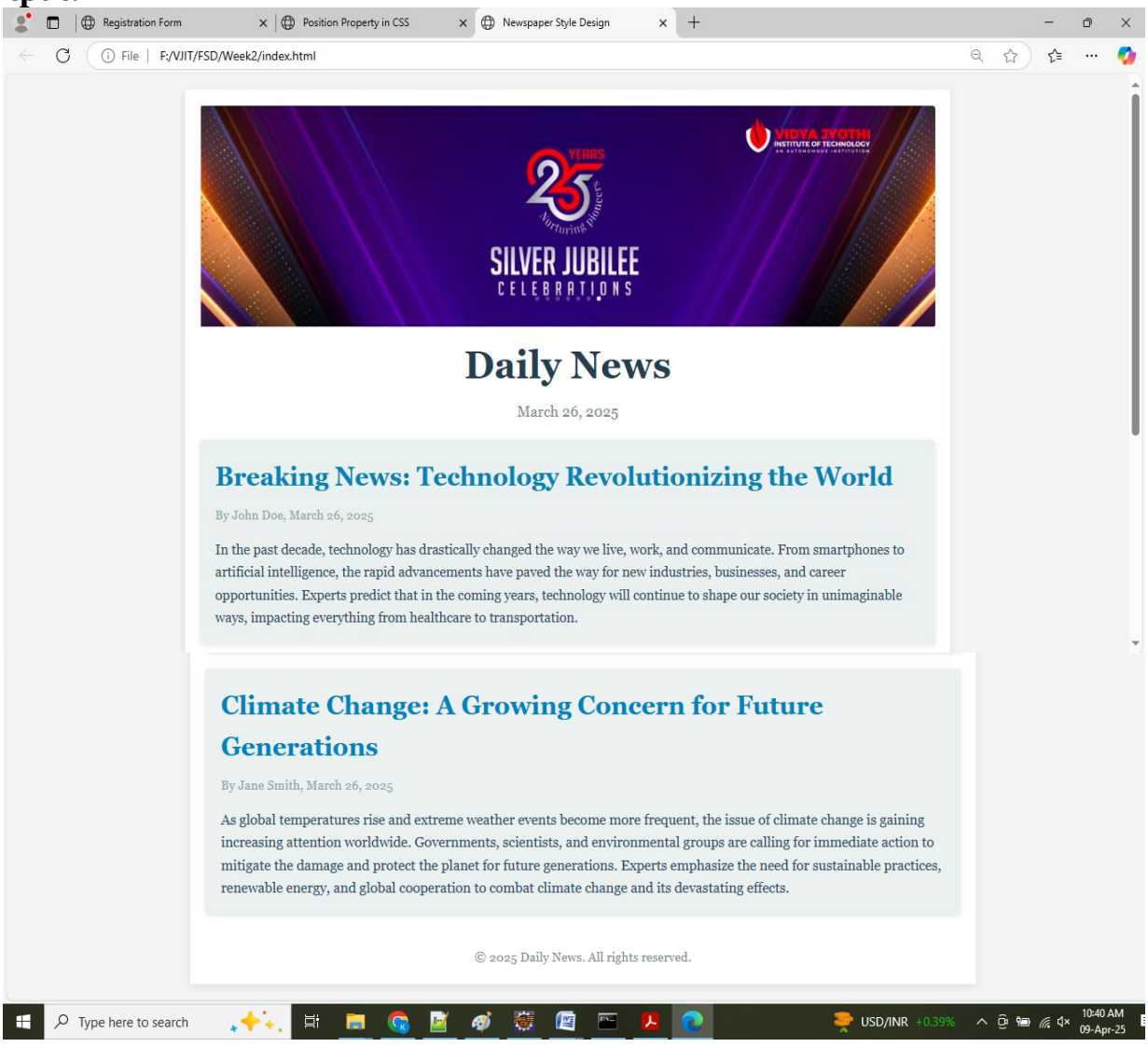
Style.css

```
/* Reset default margin and padding */
* {  margin: 0;
  padding: 0;
  box-sizing: border-box;
}
/* Global Styles */
body {
  font-family: 'Georgia', serif;
  background-color: #f4f4f4;
  color: #333;
  line-height: 1.6;
  padding: 20px;
}
img {
          width: 100%;
            border: none;
            border-radius: 4px;

        }
/* Container for the entire newspaper */
.newspaper-container {
  width: 70%;
  margin: 0 auto;
  background-color: white;
  box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
  padding: 20px;
}
```

```
/* Header section */
.newspaper-header {
  text-align: center;
  margin-bottom: 20px;
}

.newspaper-header h1 {
  font-size: 3rem;
  font-weight: bold;
  color: #2c3e50;
}

.newspaper-header .date {
  font-size: 1.2rem;
  color: #7f8c8d;
  margin-top: 5px;
}

/* Articles container */
.articles-container {
  display: flex;
  flex-direction: column;
  gap: 20px;
}

/* Each article style */
.article {
  padding: 20px;
  background-color: #ecf0f1;
  border-radius: 5px;
  box-shadow: 0 1px 5px rgba(0, 0, 0, 0.1);
}

.article-title {
  font-size: 2rem;
  font-weight: bold;
  color: #2980b9;
  margin-bottom: 10px;
}

.article-date {
  font-size: 1rem;
  color: #95a5a6;
  margin-bottom: 15px;
}
```

```
.article-content {
  font-size: 1.1rem;
  color: #34495e;
}

/* Footer section */
.newspaper-footer {
  text-align: center;
  margin-top: 40px;
  font-size: 1rem;
  color: #7f8c8d;
}
```

Output:



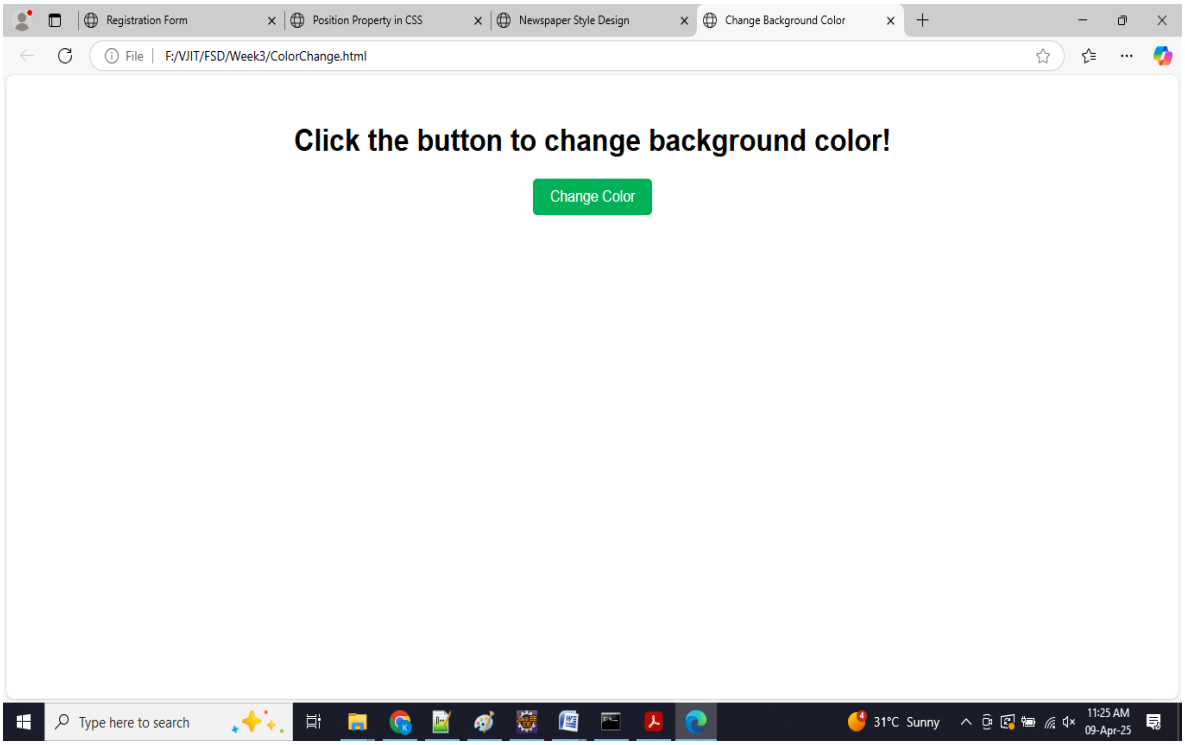
3. a) Write a JavaScript program to change the background color after clicking "change color" button.

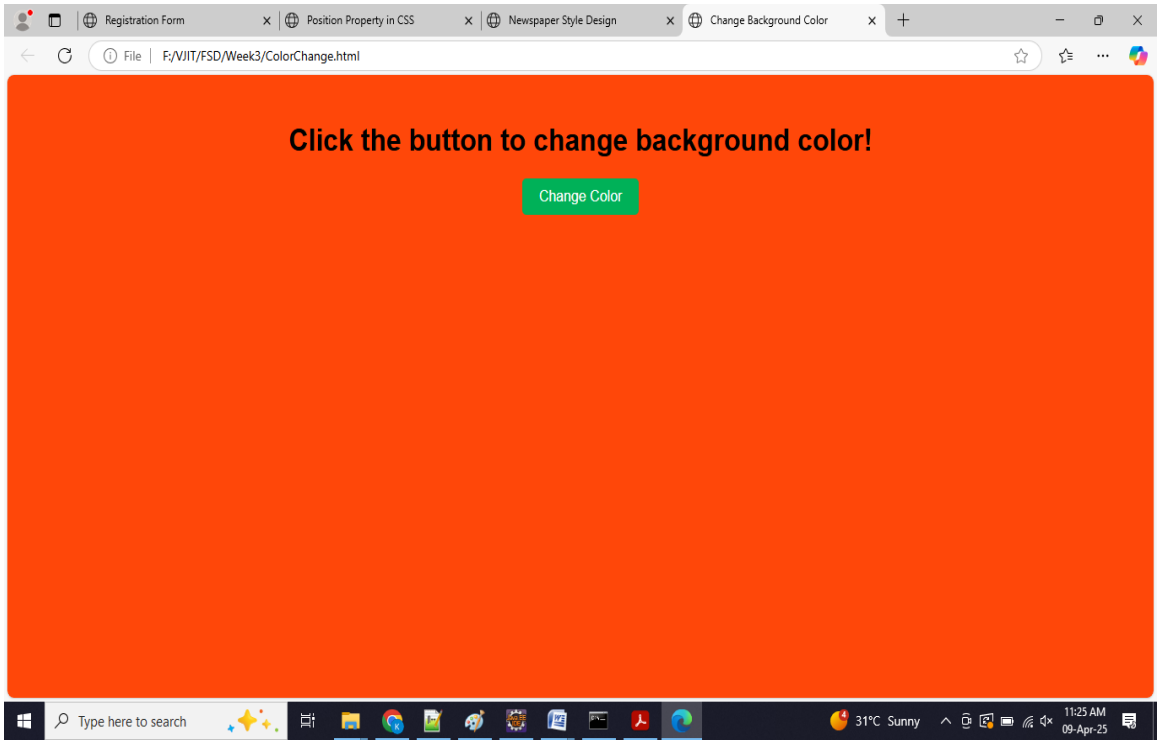
ColorChange.html

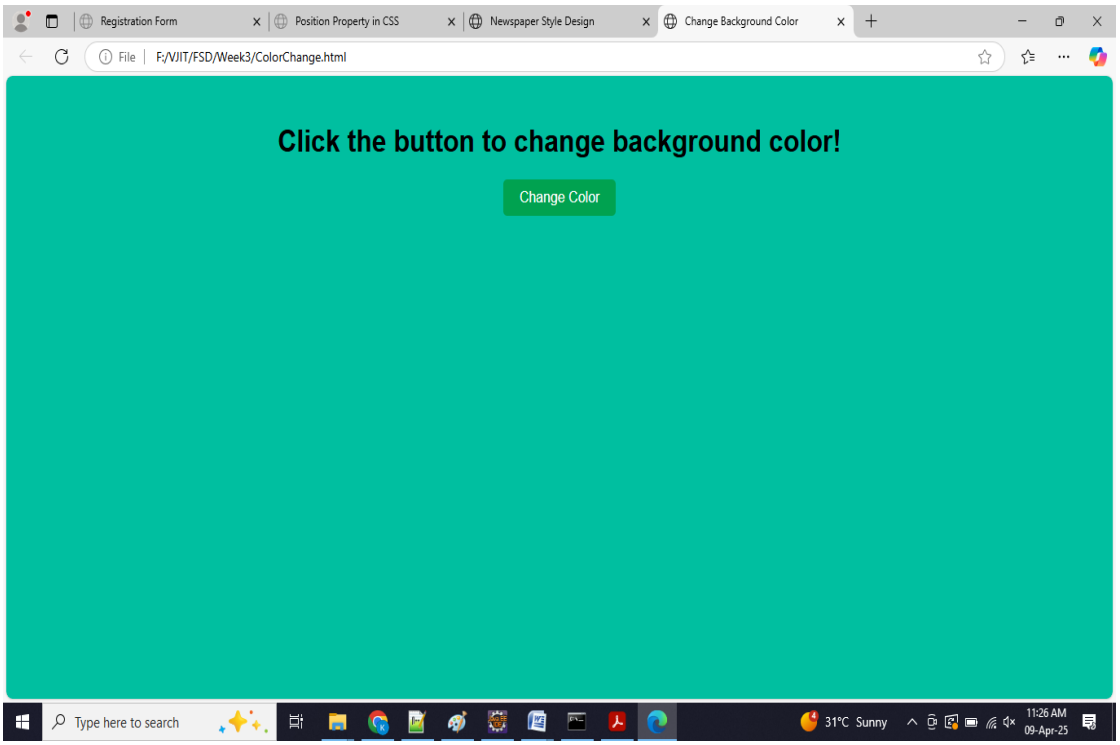
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Change Background Color</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            text-align: center;
            margin-top: 50px;
        }
        #colorButton {
            padding: 10px 20px;
            font-size: 16px;
            cursor: pointer;
            background-color: #4CAF50;
            color: white;
            border: none;
            border-radius: 5px;
        }
        #colorButton:hover {
            background-color: #45a049;
        }
    </style>
</head>
<body>
    <h1>Click the button to change background color!</h1>
    <button id="colorButton">Change Color</button>
    <script>
        // Get the button element by its ID
        const button = document.getElementById('colorButton');
        // Define an array of colors to change the background to
        const colors = ['#FF5733', '#33FF57', '#5733FF', '#F1C40F', '#8E44AD',
'#1ABC9C'];

        // Add an event listener for the 'click' event
        button.addEventListener('click', function() {
            // Generate a random index to choose a color from the colors array
            const randomColor = colors[Math.floor(Math.random() * colors.length)];
```

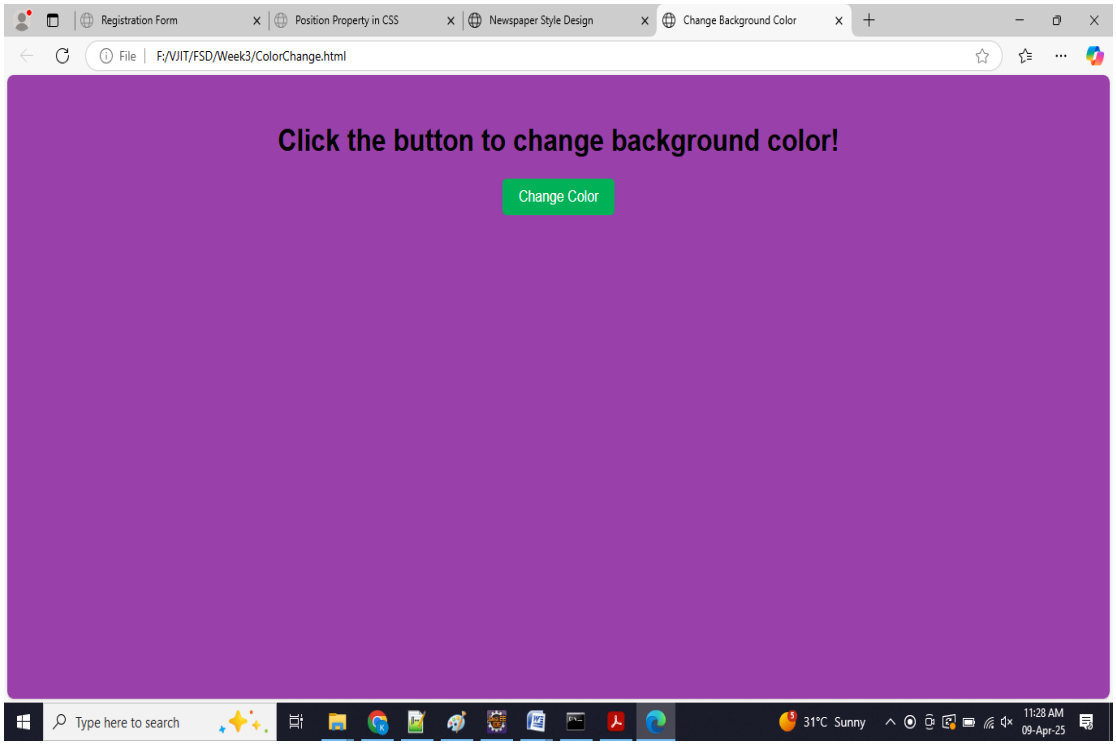
```
            // Change the background color of the body
            document.body.style.backgroundColor = randomColor;
        });
    </script>
</body>
</html>
```

Output:






Registration Form
Position Property in CSS
Newspaper Style Design
Change Background Color
File | F:/VJIT/FSD/Week3/ColorChange.html
Click the button to change background color!
Change Color
Type here to search
31°C Sunny
11:26 AM
09-Apr-25


Registration Form
Position Property in CSS
Newspaper Style Design
Change Background Color
File | F:/VJIT/FSD/Week3/ColorChange.html
Click the button to change background color!
Change Color
Type here to search
31°C Sunny
11:28 AM
09-Apr-25

3. b) Write a JavaScript program to validate registration page using regular expression.

RegistrationValidation.html

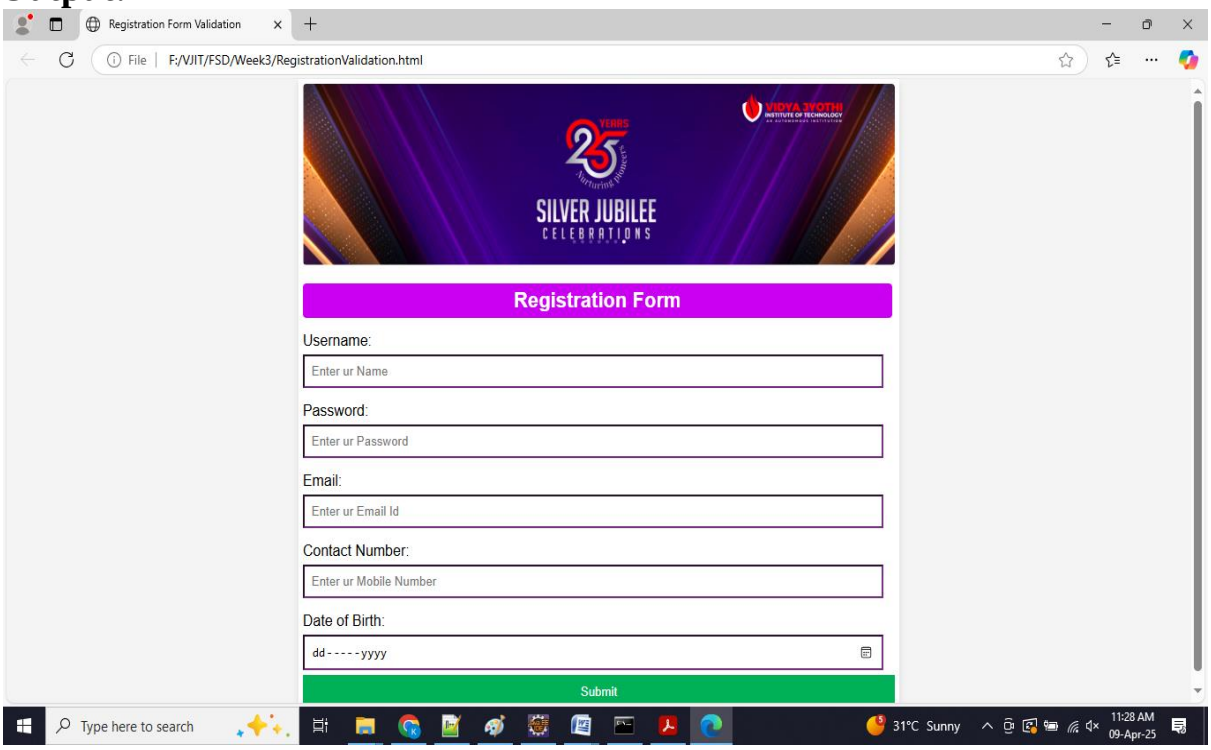
```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Registration Form Validation</title>
   <style>
      body {
         font-family: Arial, sans-serif;
         margin: 0;
         padding: 0;
         background-color: #f4f4f4;
      }
      .container {
         width: 50%;
         margin: 0 auto;
         background-color: white;
         padding: 5px;
         box-shadow: 0 4px 8px rgba(0,0,0,0.1);
      }
      h1 {
         text-align: center;

                        width: 98%;
         padding: 5px;
         background-color: #b91df5;
         color: white;
         font-size: 22px;
         border: none;
         border-radius: 4px;
      }
               img {
        width: 100%;
         border: none;
         border-radius: 4px;

      }
      label {
         display: block;
         margin-top: 10px;
      }
```

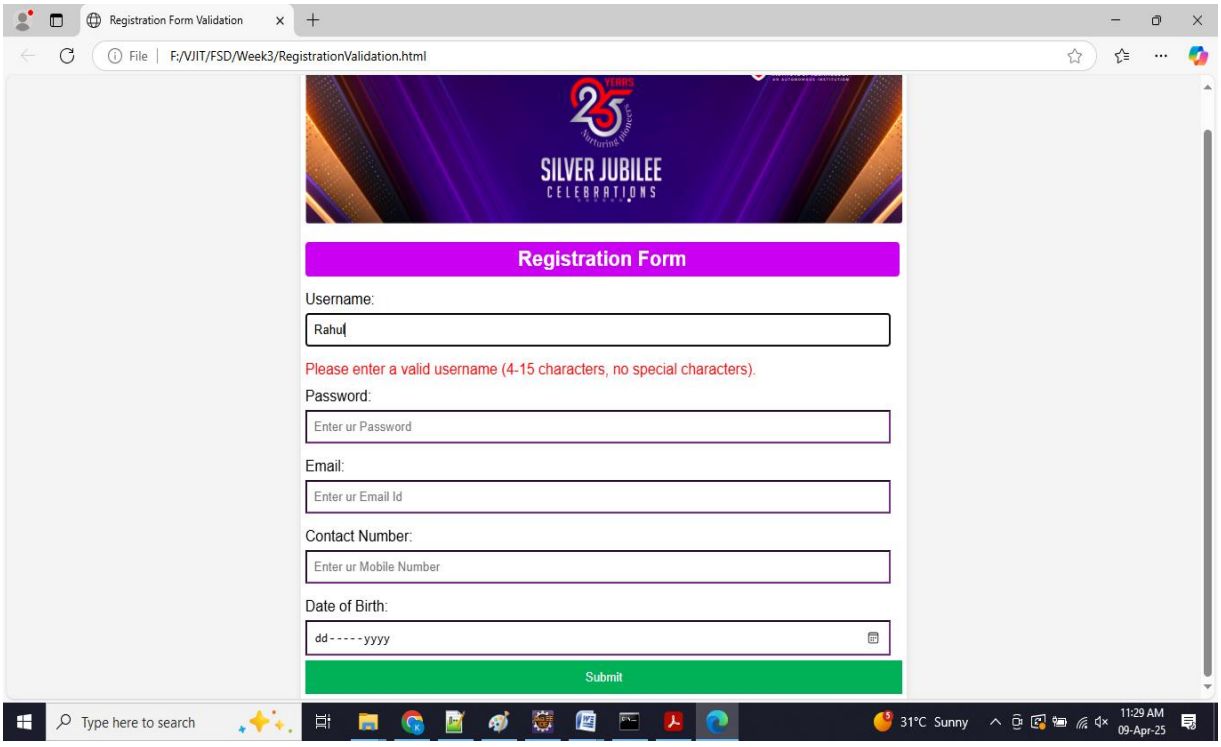
```
input {
        width: 98%;
        padding: 8px;
        margin: 5px 0;
        box-sizing: border-box;
                border-color: #6c3483;
    }
    .error {
        color: red;
        margin-top: 10px;
        display: none;
    }
    button {
        padding: 10px 15px;
        background-color: #4CAF50;
        color: white;
        border: none;
        cursor: pointer;
        width: 100%;
    }
    button:hover {
        background-color: #45a049;
    }
  </style>
</head>
<body>

<div class="container">
 <img src="banner.JPG" />
  <h1>Registration Form</h1>
  <form id="registrationForm" onsubmit="return validateForm()"
method="post">
    <label for="username">Username:</label>
    <input type="text" id="username" name="username" placeholder="Enter
ur Name">
    <div class="error" id="usernameError">Please enter a valid username (4-
15 characters, no special characters).</div>

    <label for="password">Password:</label>
    <input type="password" id="password" name="password"
placeholder="Enter ur Password">
    <div class="error" id="passwordError">Please enter a valid password (6-12
characters, at least one number).</div>
```

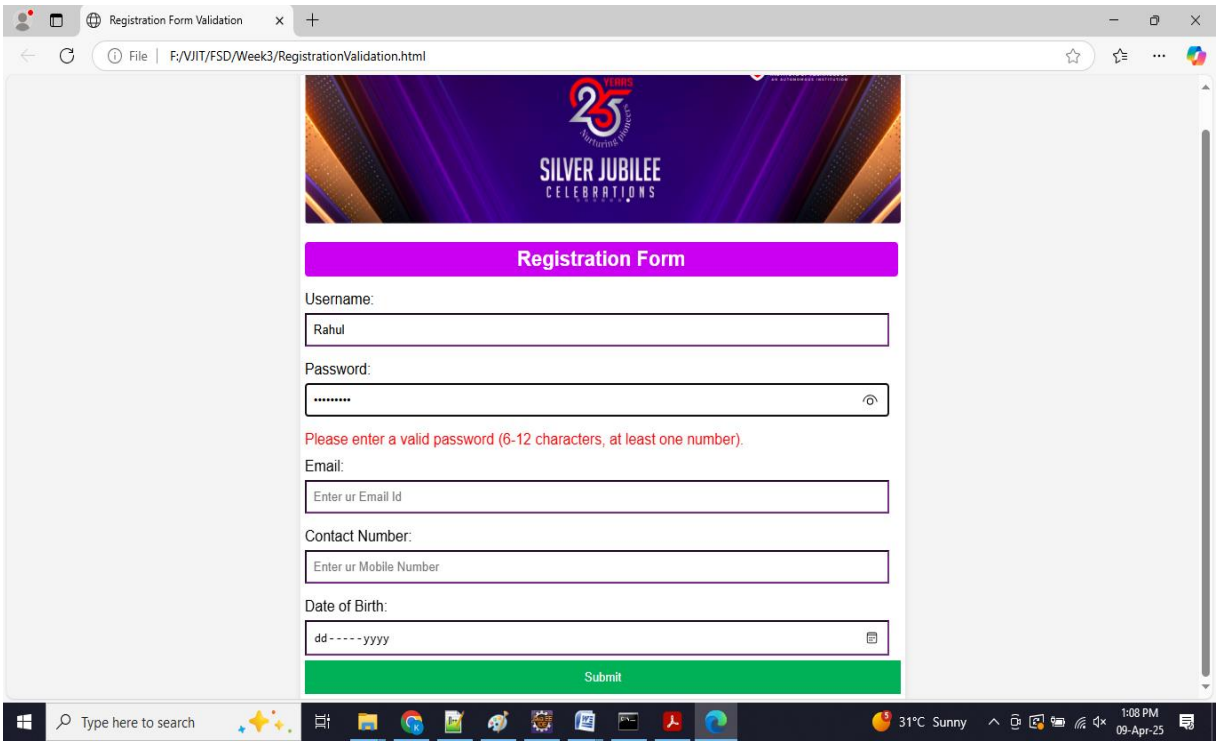
```
<label for="email">Email:</label>
        <input type="email" id="email" name="email" placeholder="Enter ur
Email Id">
        <div class="error" id="emailError">Please enter a valid email
address.</div>

        <label for="contact">Contact Number:</label>
        <input type="text" id="contact" name="contact" maxlength=10
placeholder="Enter ur Mobile Number">
        <div class="error" id="contactError">Please enter a valid 10-digit contact
number.</div>

        <label for="dob">Date of Birth:</label>
        <input type="date" id="dob" name="dob">
        <div class="error" id="dobError">Please enter your date of birth.</div>

        <button type="submit">Submit</button>
    </form>
</div>

<script>
    function validateForm()
  {
        // Clear previous errors
        const errors = document.querySelectorAll('.error');
        errors.forEach(error => error.style.display = 'none');

        // Username validation (4-15 characters, no special characters)
        const username = document.getElementById('username').value;
        const usernamePattern = /^[a-zA-Z0-9]{4,15}$/;
        if (!username.match(usernamePattern))
       {
            document.getElementById('usernameError').style.display = 'block';
            return false;
        }

        // Password validation (6-12 characters, at least one number)
        const password = document.getElementById('password').value;
        const passwordPattern = /^(?=.*\d)[A-Za-z\d]{6,12}$/;
        if (!password.match(passwordPattern))
       {
            document.getElementById('passwordError').style.display = 'block';
            return false;
        }
```

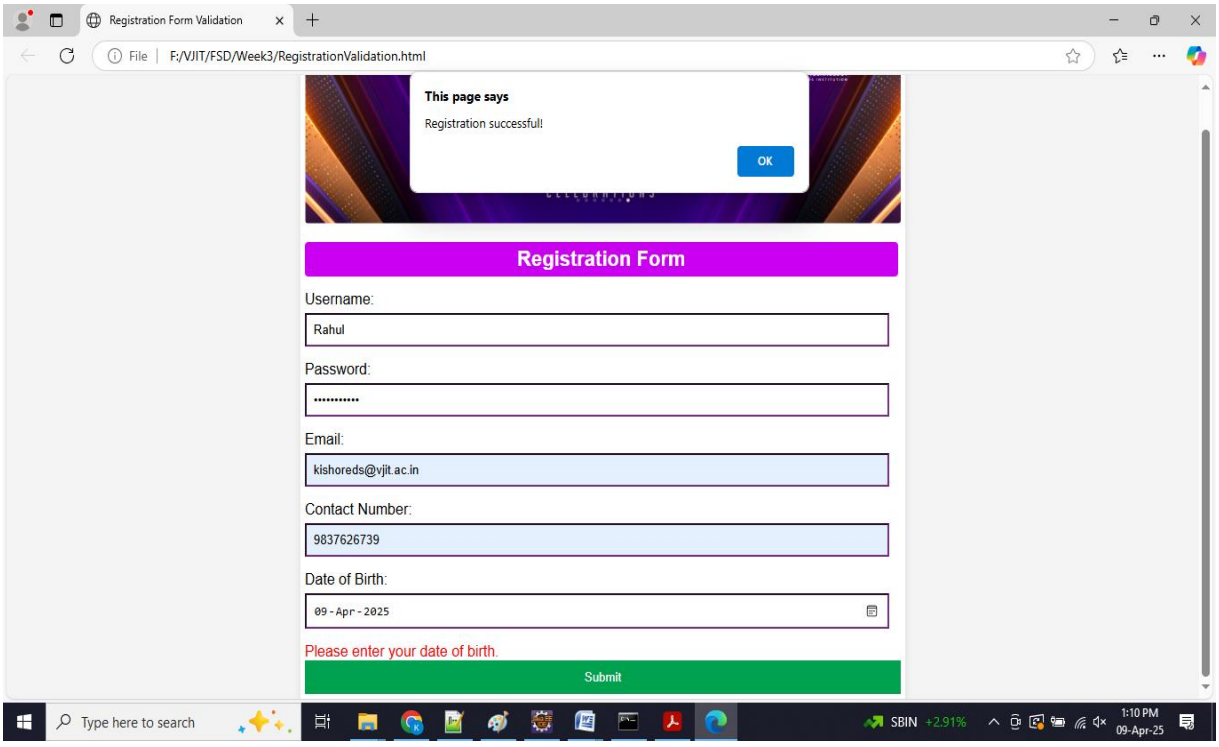
```
            // Email validation
            const email = document.getElementById('email').value;
            const emailPattern = /^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-
        Z]{2,}$/;
            if (!email.match(emailPattern))
           {
               document.getElementById('emailError').style.display = 'block';
               return false;
            }

            // Contact Number validation (exactly 10 digits)
            const contact = document.getElementById('contact').value;
            const contactPattern = /^\d{10}$/;
            if (!contact.match(contactPattern))
           {
               document.getElementById('contactError').style.display = 'block';
               return false;
            }

            // Date of Birth validation (should not be empty)
            const dob = document.getElementById('dob').value;
            if (!dob)
           {
               document.getElementById('dobError').style.display = 'block';
               return false;
            }

            alert("Registration successful!");
            return true;
         }
        </script>

        </body>
</html>
```

Output:


Registration Form Validation
File | F:/VJIT/FSD/Week3/RegistrationValidation.html
VIDYA JYOTHI INSTITUTE OF TECHNOLOGY
25 YEARS
SILVER JUBILEE
CELEBRATIONS
Registration Form
Username:
Enter ur Name
Password:
Enter ur Password
Email:
Enter ur Email Id
Contact Number:
Enter ur Mobile Number
Date of Birth:
dd-----yyyy
Submit
Type here to search
31°C Sunny
11:28 AM
09-Apr-25


Registration Form Validation
File | F:/VJIT/FSD/Week3/RegistrationValidation.html
25 YEARS
SILVER JUBILEE
CELEBRATIONS
Registration Form
Username:
Rahul
Please enter a valid username (4-15 characters, no special characters).
Password:
Enter ur Password
Email:
Enter ur Email Id
Contact Number:
Enter ur Mobile Number
Date of Birth:
dd-----yyyy
Submit
Type here to search
31°C Sunny
11:29 AM
09-Apr-25


Registration Form Validation
File | F:/VJIT/FSD/Week3/RegistrationValidation.html
SILVER JUBILEE
CELEBRATIONS
Registration Form
Username:
Rahul
Password:
Please enter a valid password (6-12 characters, at least one number).
Email:
Enter ur Email Id
Contact Number:
Enter ur Mobile Number
Date of Birth:
dd-----yyyy
Submit
Type here to search
31°C Sunny
1:08 PM
09-Apr-25

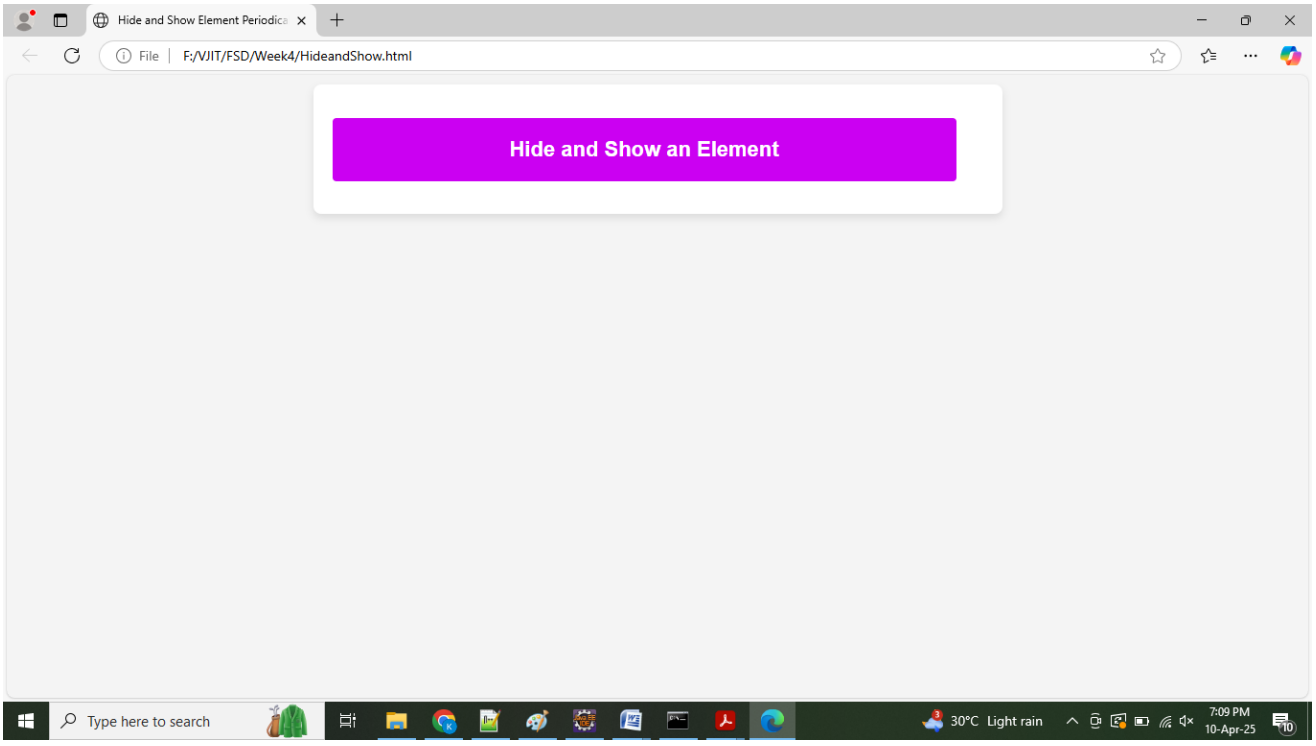

Registration Form Validation
File | F:/VJIT/FSD/Week3/RegistrationValidation.html
This page says
Registration successful!
OK
Registration Form
Username:
Rahul
Password:
Email:
kishoreds@vjit.ac.in
Contact Number:
9837626739
Date of Birth:
09-Apr-2025
Please enter your date of birth.
Submit
Type here to search
SBIN +2.91%
1:10 PM
09-Apr-25

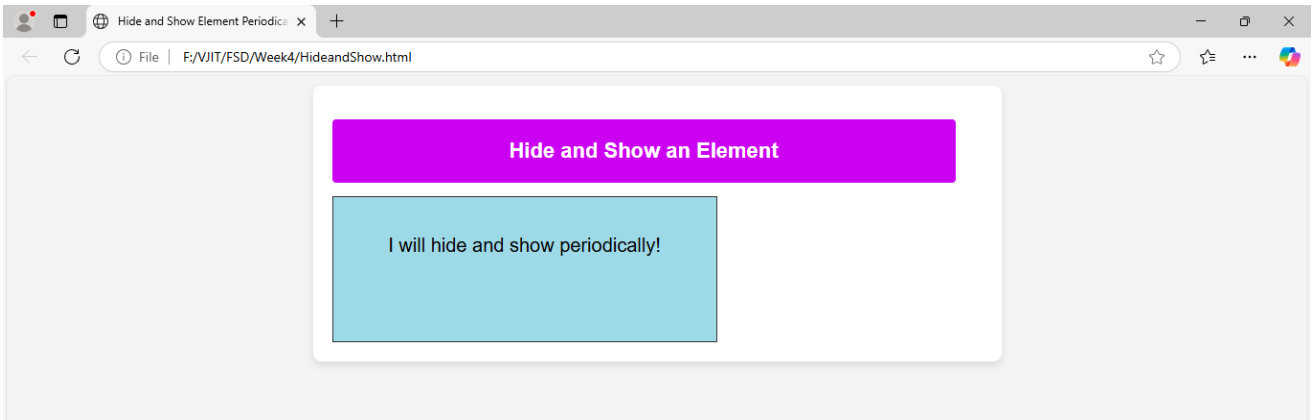
4. a) Write a code to hide and show an element in a periodic interval without any action from the user using JQuery.

HideandShow.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Hide and Show Element Periodically</title>
   <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
   <style>
        body {
          font-family: Arial, sans-serif;
          margin: 0;
          padding: 0;
          background-color: #f4f4f4;
       }
       .container {
          width: 50%;
          margin: 10px auto;
          background-color: #fff;
          padding: 20px;
          border-radius: 8px;
          box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
       }
         h1 {
          text-align: center;
          width: 90%;
          padding: 20px;
          background-color: #b91df5;
          color: white;
          font-size: 22px;
          border: none;
          border-radius: 4px;
       }
       #myElement {
          width: 400px;
          height: 150px;
          background-color: lightblue;
          text-align: center;
          line-height: 100px;
          font-size: 20px;
          border: 1px solid #333;        }
   </style>
```

```
</head>
<body>
<div class="container">
<h1>Hide and Show an Element</h1>
   <div id="myElement">I will hide and show periodically!</div>
</div>
   <script>
      $(document).ready(function() {
         setInterval(function() {
            $('#myElement').fadeToggle(); // Hide or show the element
         }, 2000); // 2000 milliseconds = 2 seconds
      });
   </script>

</body>
</html>
```





4. b) Write a program to create and Build a star rating system using JQuery.

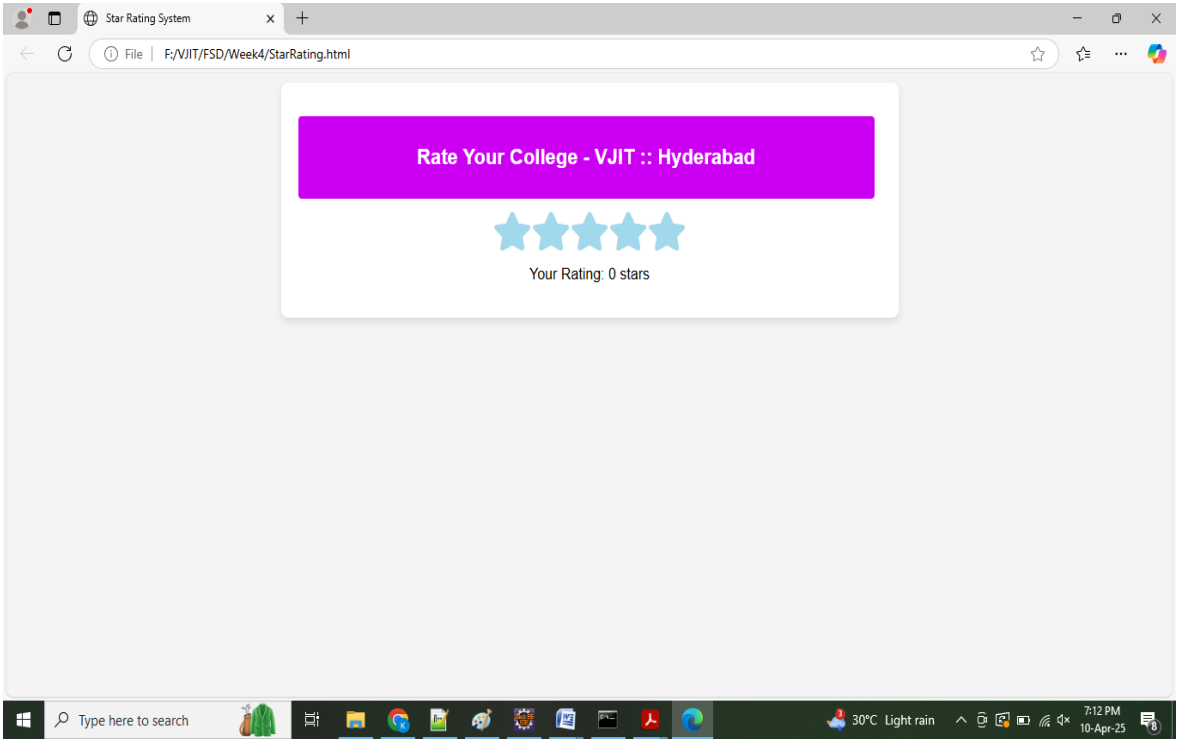
StarRating.html

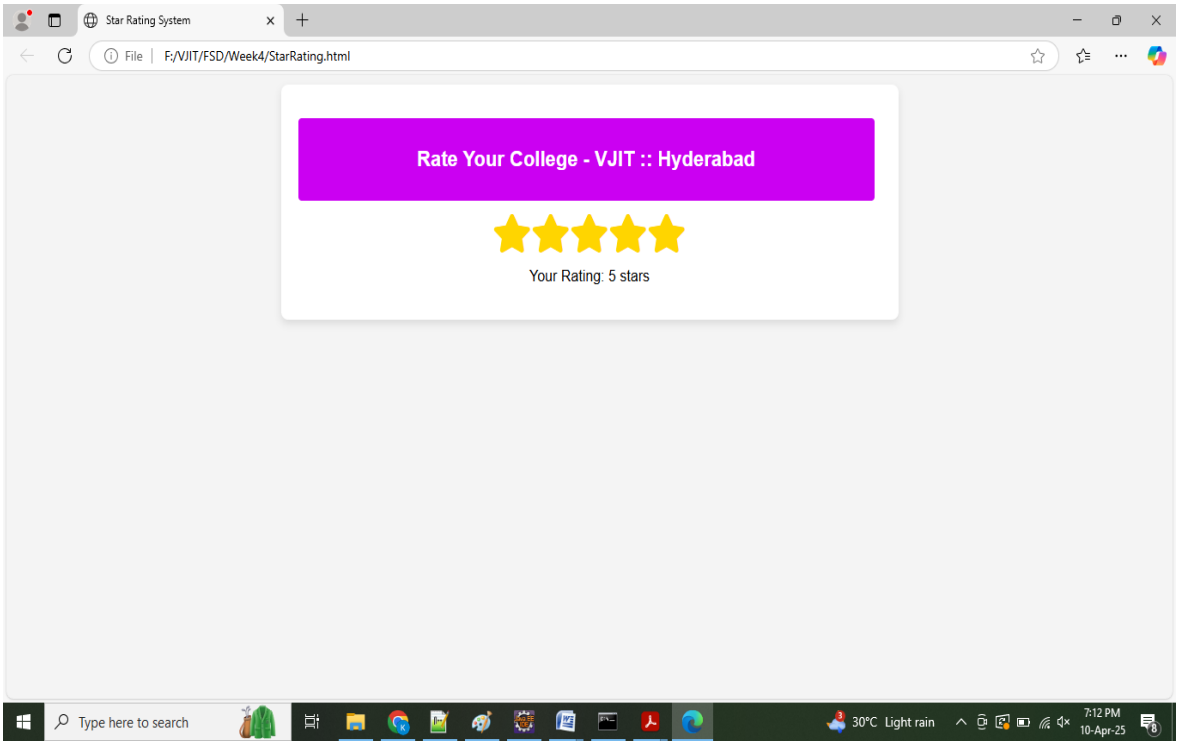
```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Star Rating System</title>
   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.3/css/all.min.css">
   <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
   <style>
        body {
          font-family: Arial, sans-serif;
          margin: 0;
          padding: 0;
          background-color: #f4f4f4;
        }
        .container {
          width: 50%;
          margin: 10px auto;
          background-color: #fff;
          padding: 20px;
          border-radius: 8px;
          box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
        }
         h1 {
          text-align: center;
          width: 90%;
          padding: 30px;
          background-color: #b91df5;
          color: white;
          font-size: 22px;
          border: none;
          border-radius: 4px;
        }
        .star-rating {
          display: flex;
                        justify-content: center;
          cursor: pointer;

        }
```

```
  .star-rating i {
        font-size: 40px;
        color: #afd7ea;

      }
      .star-rating .selected {
        color: gold;
      }
    </style>
</head>
<body>

<div class="container">
<h1>Rate Your College - VJIT :: Hyderabad</h1>

    <div class="star-rating">
      <i class="fa fa-star" data-index="1"></i>
      <i class="fa fa-star" data-index="2"></i>
      <i class="fa fa-star" data-index="3"></i>
      <i class="fa fa-star" data-index="4"></i>
      <i class="fa fa-star" data-index="5"></i>
    </div>

    <p align="center">Your Rating: <span id="rating-value">0</span> stars</p>
</div>

<script>

  $(document).ready(function()
  {
        var rating = 0;
        // Hover effect to show the stars
        $('.star-rating i').hover(function() {
          var index = $(this).data('index');
          // Highlight stars based on hover position
          $('.star-rating i').each(function() {
            if ($(this).data('index') <= index) {
              $(this).addClass('selected');
            } else {
              $(this).removeClass('selected');
            }
          });
        },
```

```
                function()
                 {
                 // Reset hover effect
                 $('.star-rating i').removeClass('selected');
                 // Highlight selected rating
                 $('.star-rating i').each(function() {
                    if ($(this).data('index') <= rating) {
                       $(this).addClass('selected');
                    }
                 });
              });

              // Click event to set the rating
              $('.star-rating i').click(function()
               {
                 rating = $(this).data('index');
                 $('#rating-value').text(rating);

                 // Save the rating and highlight the selected stars
                 $('.star-rating i').each(function() {
                    if ($(this).data('index') <= rating) {
                       $(this).addClass('selected');
                    } else {
                       $(this).removeClass('selected');
                    }
                 });
              });
           });
        </script>

    </body>
</html>
```

Output:


Star Rating System
File | F:/VJIT/FSD/Week4/StarRating.html
Rate Your College - VJIT :: Hyderabad
Your Rating: 0 stars
Type here to search
30°C Light rain
7:12 PM
10-Apr-25


Star Rating System
File | F:/VJIT/FSD/Week4/StarRating.html
Rate Your College - VJIT :: Hyderabad
Your Rating: 5 stars
Type here to search
30°C Light rain
7:12 PM
10-Apr-25

5. a) Write a program to demonstrate ReactJS Class and Instance.

Node.js Installation

1. Check if Node.js and npm are Installed

First, verify whether Node.js and npm are installed on your system by running the following commands:

```
node -v
npm -v
```

If these commands return the version of Node.js and npm, it means they are installed. If not, follow the next steps.

2. Install Node.js and npm

To install Node.js (which includes npm), follow these steps:

1. **Download and Install Node.js**:
   - Go to the official Node.js website.
   - Download the **LTS (Long Term Support)** version, as it is the most stable version.
   - Follow the installation prompts to install Node.js and npm.
2. **Verify Installation**: After installation, verify again by running:

```
node -v
npm -v
```

These commands should now output the versions of Node.js and npm, confirming the installation.

3. Check npm Path Configuration

If you have installed Node.js and npm but are still facing the issue, it could be because the **npm path is not correctly set** in your system environment variables.
To fix this:

- Open **System Properties**.
- Click on **Environment Variables**.
- Under **System Variables**, find the Path variable and ensure the following paths are added:
  - For Windows, it should include the following (based on your installation location):

```
C:\Program Files\nodejs\
```

Once added, restart your terminal to make sure the changes take effect.

Steps:

1. Set Up React Application
2. Install React Router for Navigation
3. Create Pages: Registration, Login, Contact, About
4. Set Up Routing to Navigate Between Pages

1. Set Up the React Application

If you haven't created a React app yet, use **Create React App** to set up a basic project.

```
npx create-react-app class-instance-demo
cd class-instance-demo
npm start
```

2. Install React Router

You need **React Router** to implement routing and navigation between different pages. Install it using npm:

```
npm install react-router-dom
```

3. Implement Routing

Update App.js to set up routing and implement the navigation between pages.

Folder Structure:

```
src/
  |-- App.js
  |--  App.css
  |-- Person.js
  |--  index.js
```

App.js

```
import React, { Component } from 'react';
import './App.css';
import Person from './Person';

// Main App component to render the Person component
class App extends Component {
  render() {
    return (
      <div className="App">
          <div class="container">
        <h1>React Class and Instance Demo</h1>
        {/* Create an instance of the Person component */}
        <Person />
                </div>
      </div>
    );
  }
}

export default App;
```

Person.js

```
import React, { Component } from 'react';

// Class component definition
class Person extends Component {
  // Constructor for initializing state
  constructor(props) {
    super(props);
    this.state = {
      name: 'Rahul',
      age: 35,
    };
  }

  // Method to change the name
  changeName = () => {
    this.setState({ name: 'Rohit Sharma' });
  };
```

```
  // Method to increase age
  increaseAge = () => {
    this.setState({ age: this.state.age + 1 });
  };

  // Rendering the component
  render() {
    return (
      <div>
        <h4>Person Component</h4>
        <p>Name: {this.state.name}</p>
        <p>Age: {this.state.age}</p>
        <button onClick={this.changeName}>Change Name</button>
        <button onClick={this.increaseAge}>Increase Age</button>
      </div>
    );
  }
}

export default Person;
```

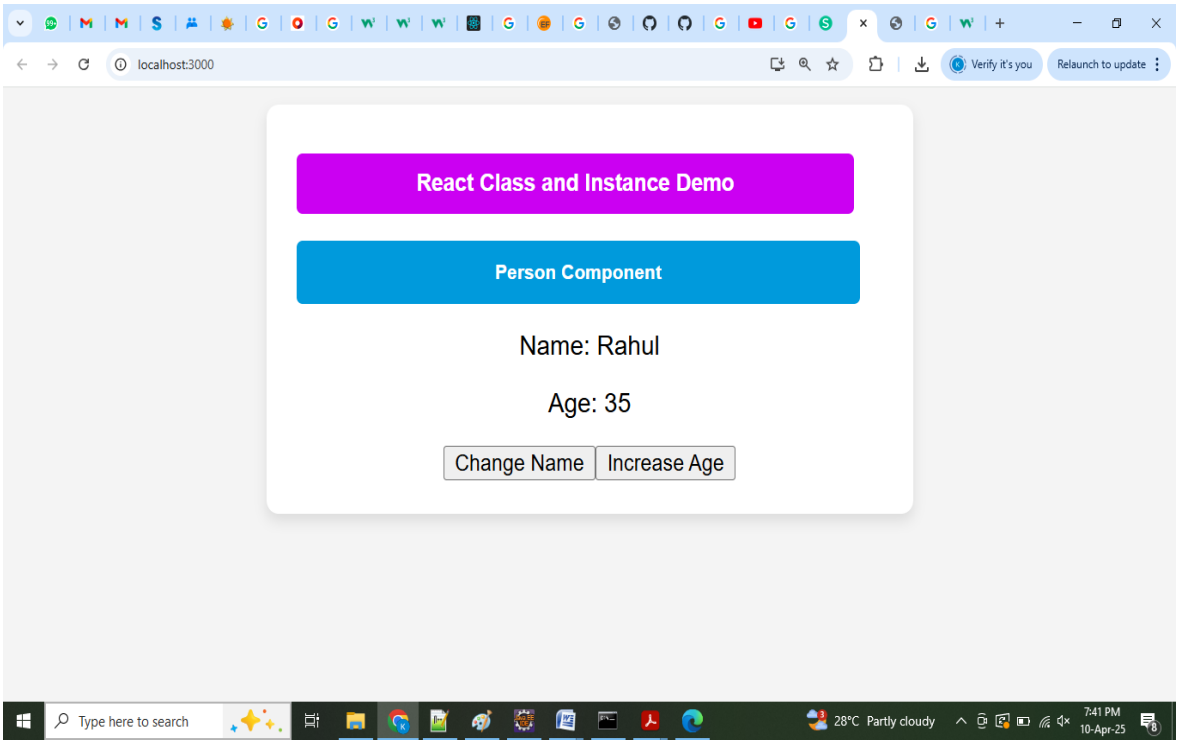
App.css

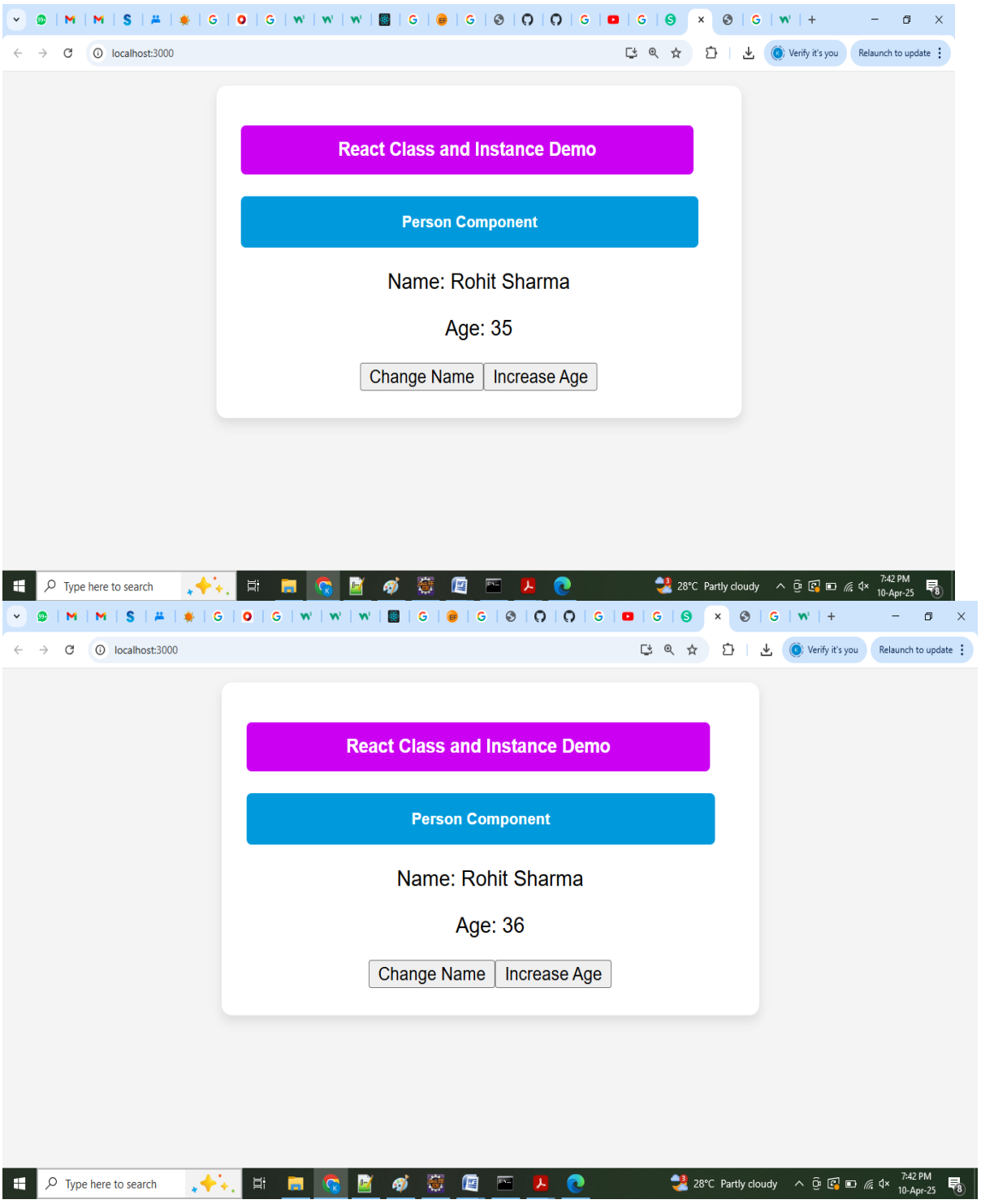
```
.App {
  text-align: center;
}

.App-logo {
  height: 40vmin;
  pointer-events: none;
}

@media (prefers-reduced-motion: no-preference) {
  .App-logo {
    animation: App-logo-spin infinite 20s linear;
  }
}

.App-header {
  background-color: #282c34;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
```

```
  justify-content: center;
  font-size: calc(10px + 2vmin);
  color: white;
}

.App-link {
  color: #61dafb;
}

@keyframes App-logo-spin {
  from {
    transform: rotate(0deg);
  }
  to {
    transform: rotate(360deg);
  }
}
body {
        font-family: Arial, sans-serif;
        margin: 0;
        padding: 0;
        background-color: #f4f4f4;
    }

    .container {
        width: 50%;
        margin: 10px auto;
        background-color: #fff;
        padding: 20px;
        border-radius: 8px;
        box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    }
      h1 {
        text-align: center;

                    width: 90%;
        padding: 10px;
        background-color: #b91df5;
        color: white;
        font-size: 14px;
        border: none;

        border-radius: 4px;
    }
```

```
h4 {
        text-align: center;
            width: 90%;
        padding: 12px;
        background-color: #3498db;
        color: white;
        font-size: 12px;
        border: none;

        border-radius: 4px;
}
```

Output:




localhost:3000
Verify it's you
Relaunch to update
React Class and Instance Demo
Person Component
Name: Rohit Sharma
Age: 35
Change Name
Increase Age
Type here to search
28°C Partly cloudy
7:42 PM
10-Apr-25
localhost:3000
Verify it's you
Relaunch to update
React Class and Instance Demo
Person Component
Name: Rohit Sharma
Age: 36
Change Name
Increase Age
Type here to search
28°C Partly cloudy
7:42 PM
10-Apr-25

5. b) Write a program to create a basic calculator to perform arithmetic operations using ReactJS.

1. Set Up the React Application

If you haven't created a React app yet, use **Create React App** to set up a basic project.

```
npx create-react-app calculator-app
cd calculator-app
npm start
```

2. Implement Routing

Update App.js to set up routing and implement the navigation between pages.

Folder Structure:

```
src/
  |-- App.js
  |--  App.css
  |-- Calculator.js
  |--  index.js
```

App.js

```
import React from 'react';
import './App.css';
import Calculator from './Calculator';

function App() {
  return (
    <div className="App">
         <div className="calculator">
      <h1>Simple React Calculator</h1>
          </div>
      <Calculator />
    </div>
  );
}

export default App;
```

Calculator.js

```
import React, { useState } from 'react';

const Calculator = () => {
  const [input, setInput] = useState(''); // To store the current input value
  const [result, setResult] = useState(null); // To store the result of calculations

  // Function to handle number and operator button clicks
  const handleButtonClick = (value) => {
    setInput(input + value); // Append the clicked button's value to the input string
  };

  // Function to handle clear button click
  const handleClear = () => {
    setInput('');
    setResult(null);
  };

  // Function to handle the calculation and show the result
  const handleEvaluate = () => {
    try {
      setResult(eval(input)); // Evaluate the input string and set the result
      setInput('');
    } catch (error) {
      setResult('Error');
    }
  };

  return (
    <div className="calculator">
      <div className="display">
        <input type="text" value={input} readOnly />
        {result !== null && <div className="result">Result: {result}</div>}
      </div>
      <div className="buttons">
        <button onClick={() => handleButtonClick('1')}>1</button>
        <button onClick={() => handleButtonClick('2')}>2</button>
        <button onClick={() => handleButtonClick('3')}>3</button>
        <button onClick={() => handleButtonClick('+')}>+</button>
        <button onClick={() => handleButtonClick('4')}>4</button>
        <button onClick={() => handleButtonClick('5')}>5</button>
        <button onClick={() => handleButtonClick('6')}>6</button>
        <button onClick={() => handleButtonClick('-')}>-</button>
```

```
            <button onClick={() => handleButtonClick('7')}>7</button>
            <button onClick={() => handleButtonClick('8')}>8</button>
            <button onClick={() => handleButtonClick('9')}>9</button>
            <button onClick={() => handleButtonClick('*')}>*</button>
            <button onClick={() => handleButtonClick('0')}>0</button>
            <button onClick={() => handleButtonClick('.')}>.</button>
            <button onClick={handleClear}>C</button>
            <button onClick={() => handleButtonClick('/')}>/</button>
            <button onClick={handleEvaluate}>=</button>
          </div>
        </div>
      );
    };

export default Calculator;
```

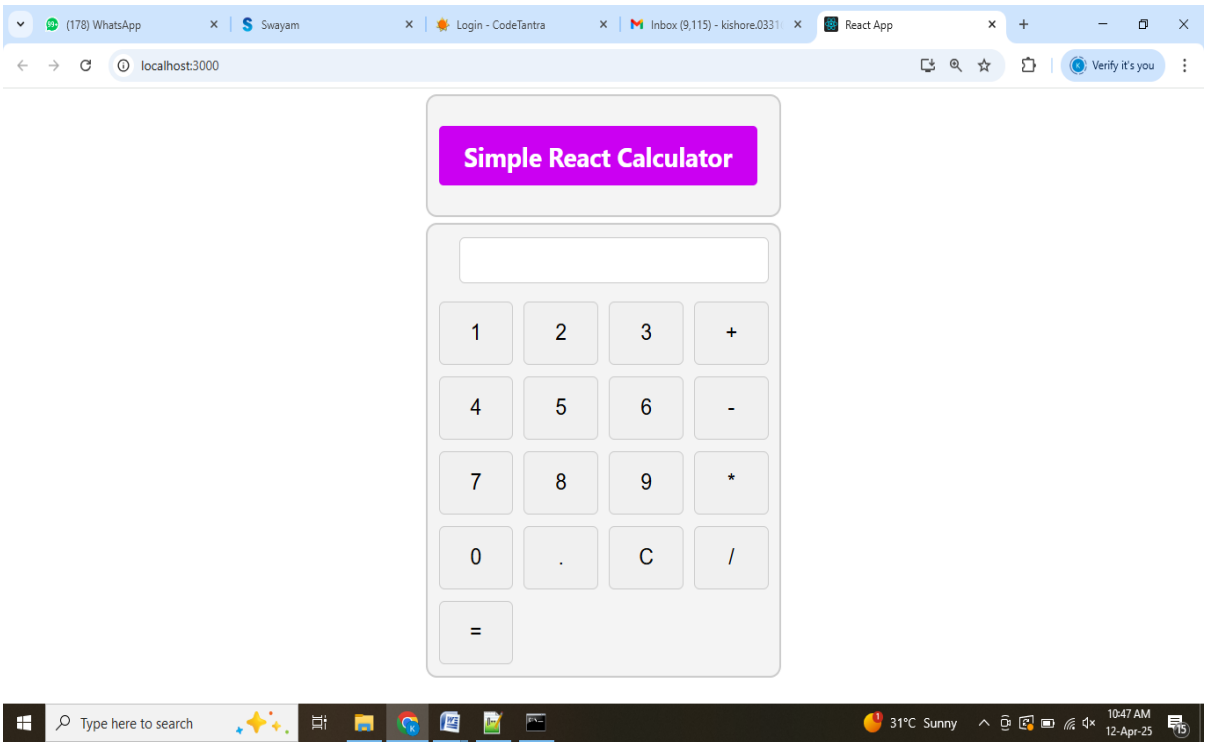
App.css

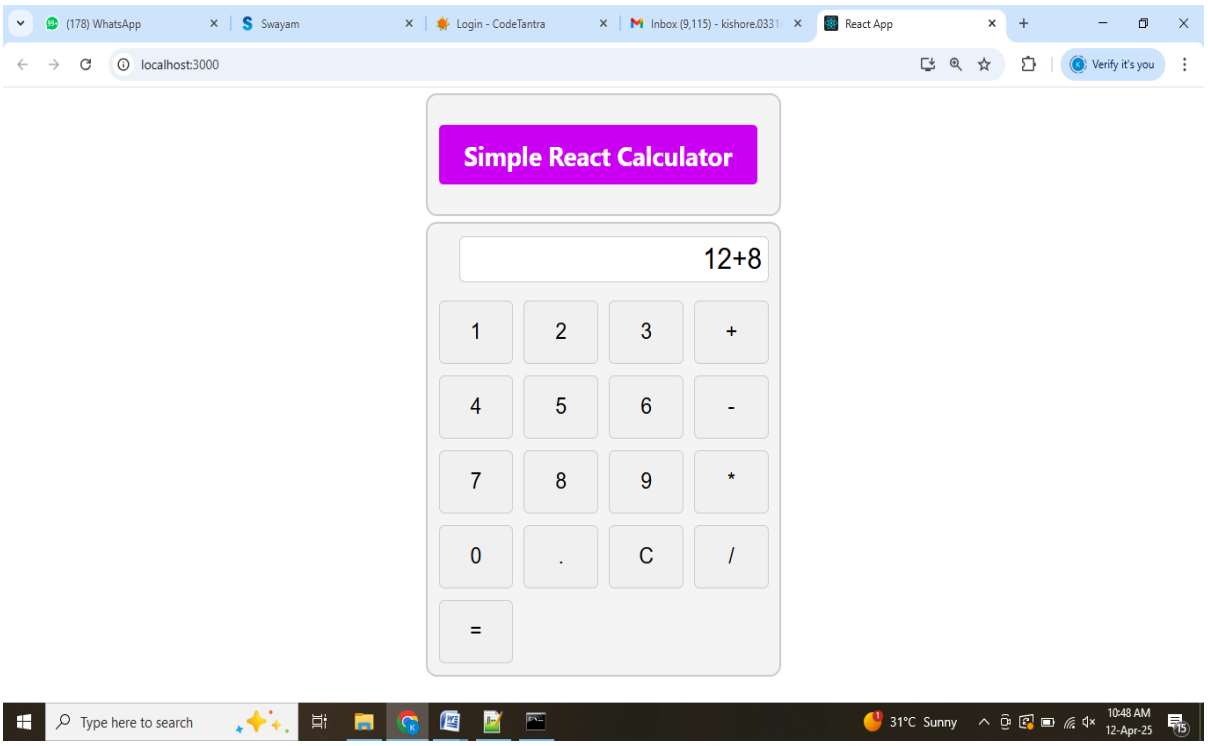
```
.App {
  text-align: center;
}

.App-logo {
  height: 40vmin;
  pointer-events: none;
}
@media (prefers-reduced-motion: no-preference) {
  .App-logo {
    animation: App-logo-spin infinite 20s linear;
  }
}
.App-header {
  background-color: #282c34;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  font-size: calc(10px + 2vmin);
  color: white;
}

.App-link {
  color: #61dafb;
}
```

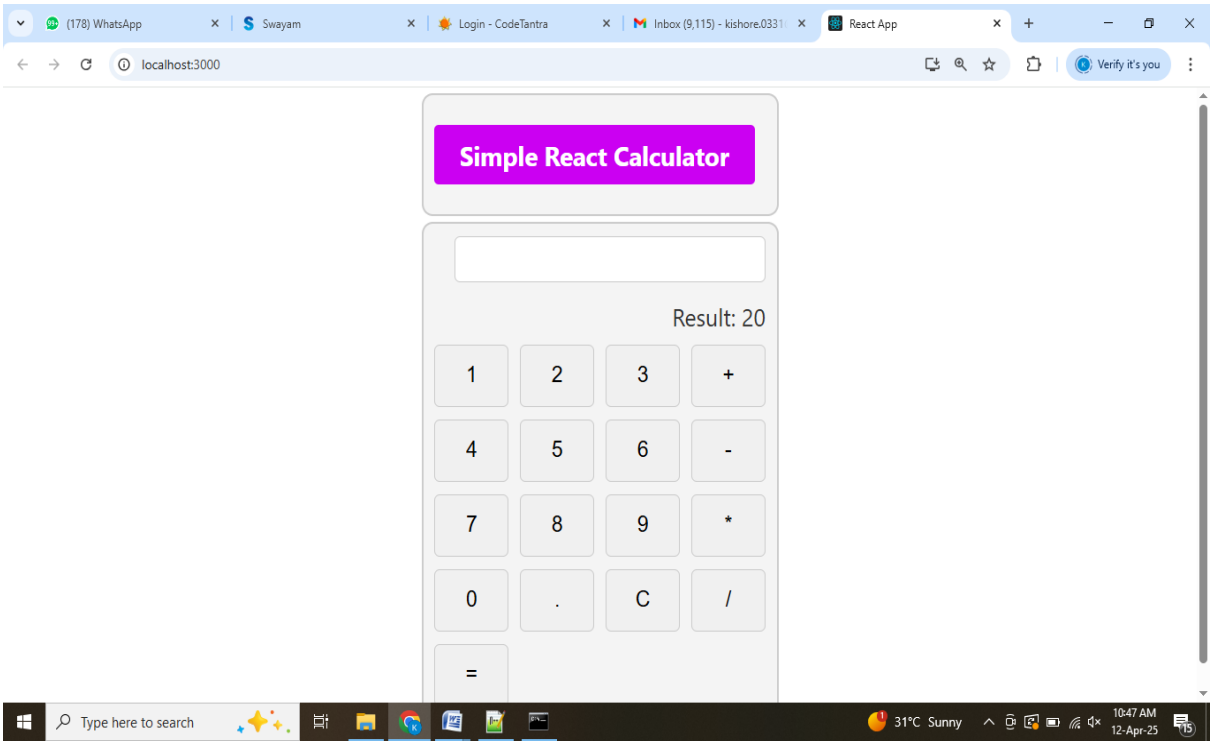
```
@keyframes App-logo-spin {
  from {
    transform: rotate(0deg);
  }
  to {
    transform: rotate(360deg);
  }
}
h1 {
        text-align: center;

                    width: 90%;
        padding: 10px;
        background-color: #b91df5;
        color: white;
        font-size: 22px;
        border: none;

        border-radius: 4px;
      }

.calculator {
  width: 300px;
  margin: 5px auto;
  padding: 10px;
  border: 2px solid #ccc;
  border-radius: 10px;
  background-color: #f4f4f4;
}

.display {
  text-align: right;
  margin-bottom: 10px;
}

.display input {
  width: 90%;
  padding: 5px;
  font-size: 24px;
  text-align: right;
  border: 1px solid #ccc;
  border-radius: 5px;
  margin-bottom: 5px;
}
```

```
.result {
  font-size: 20px;
  margin-top: 10px;
  color: #333;
}

.buttons {
  display: grid;
  grid-template-columns: repeat(4, 1fr);
  gap: 10px;
}

button {
  padding: 15px;
  font-size: 18px;
  background-color: #f0f0f0;
  border: 1px solid #ccc;
  border-radius: 5px;
  cursor: pointer;
}

button:hover {
  background-color: #ddd;
}
```

Output:




(178) WhatsApp
Swayam
Login - CodeTantra
Inbox (9,115) - kishore.0331
React App
localhost:3000
Verify it's you
Simple React Calculator
12+8
1 2 3 +
4 5 6 -
7 8 9 *
0 . C /
=
Type here to search
31°C Sunny
10:48 AM
12-Apr-25


(178) WhatsApp
Swayam
Login - CodeTantra
Inbox (9,115) - kishore.0331
React App
localhost:3000
Verify it's you
Simple React Calculator
Result: 20
1 2 3 +
4 5 6 -
7 8 9 *
0 . C /
=
Type here to search
31°C Sunny
10:47 AM
12-Apr-25

6. a) Demonstrate simple event handling example using ReactJS.

1. Set Up the React Application

If you haven't created a React app yet, use **Create React App** to set up a basic project.

```
npx create-react-app event-handling-demo
cd event-handling-demo
npm start
```

2. Implement Routing

Update App.js to set up routing and implement the navigation between pages.

Folder Structure:

```
src/
  |-- App.js
  |--  App.css
  |--  index.js
```

App.js

```
import React, { useState } from 'react';
import './App.css';

function App() {
  // State to store the paragraph text
  const [text, setText] = useState("Click the button to change the text.");
  const [inputText, setInputText] = useState("");

  // Function to handle the button click event
  const handleButtonClick = () => {
    setText("Welcome to VJIT,Hyd!");
  };

  // Function to handle input field change
  const handleInputChange = (event) => {
    setInputText(event.target.value);
  };
```

```
    return (
      <div className="App">
           <div className="header">
        <h1>Event Handling in React</h1>

        {/* Button click event */}
        <button onClick={handleButtonClick}>Change Text</button>

        {/* Paragraph text that changes on button click */}
        <p><font size="4" color="red">{text}</font></p>

        {/* Input field event */}
        <input
          type="text"
          value={inputText}
          onChange={handleInputChange}  // On change event updates input text
          placeholder="Type something"
        />

        {/* Displaying input text */}
        <p>You Typed: <font size="4" color="red">{inputText}</font></p>
           </div>
      </div>
    );
  }

export default App;
```

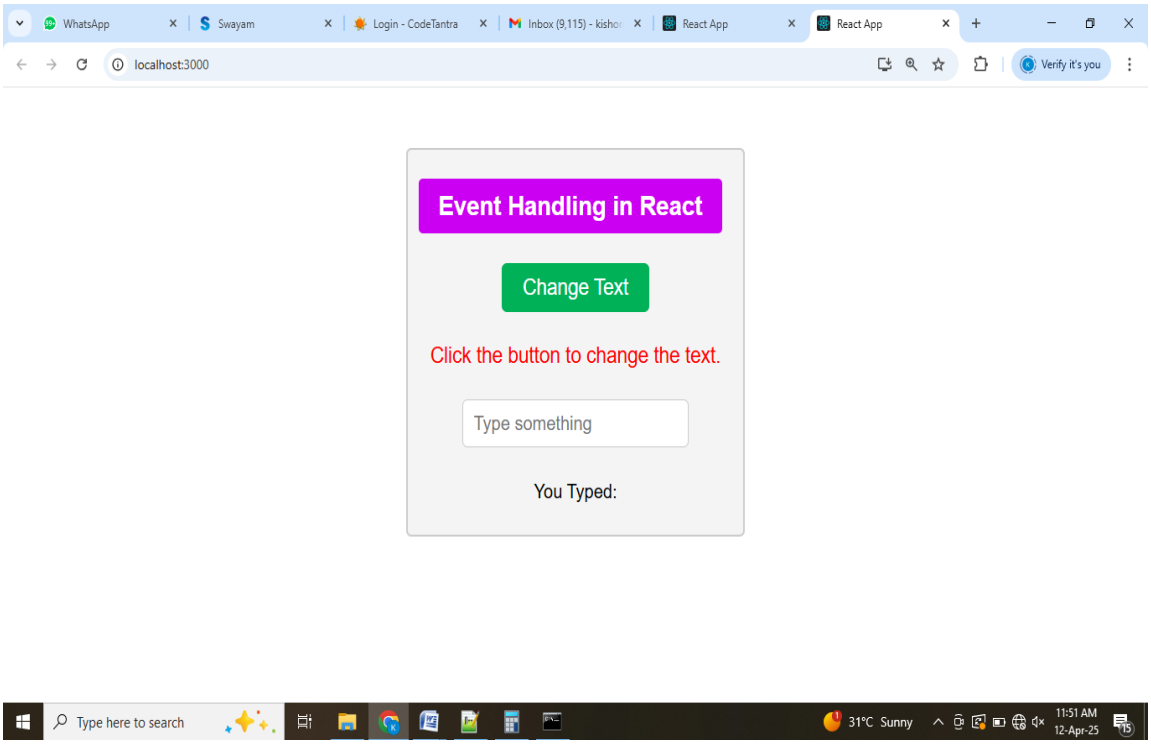
App.css

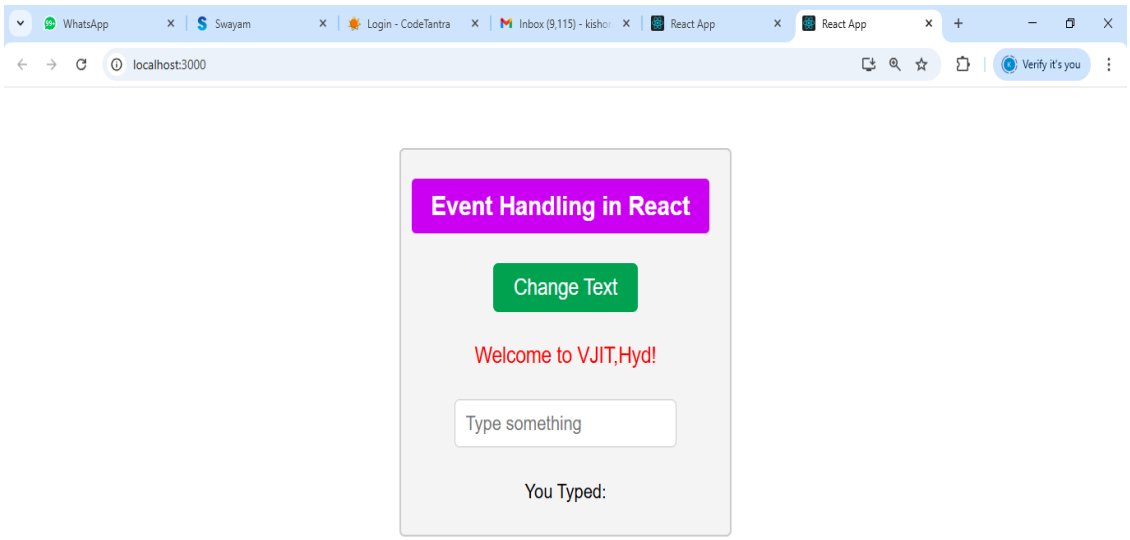
```
.App {
  text-align: center;
}

.App-logo {
  height: 40vmin;
  pointer-events: none;
}

@media (prefers-reduced-motion: no-preference) {
  .App-logo {
    animation: App-logo-spin infinite 20s linear;
  }
}
```

```
.App-header {
  background-color: #282c34;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  font-size: calc(10px + 2vmin);
  color: white;
}
.App-link {
  color: #61dafb;
}
@keyframes App-logo-spin {
  from {
    transform: rotate(0deg);
  }
  to {
    transform: rotate(360deg);
  }
}
.App {
  text-align: center;
  font-family: Arial, sans-serif;
  margin-top: 50px;
}
.header {
  width: 300px;
  margin: 5px auto;
  padding: 10px;
  border: 2px solid #ccc;
  border-radius: 5px;
  background-color: #f4f4f4;
}
h1 {
        text-align: center;
        width: 90%;
        padding: 10px;
        background-color: #b91df5;
        color: white;
        font-size: 22px;
        border: none;

        border-radius: 4px;
      }
```

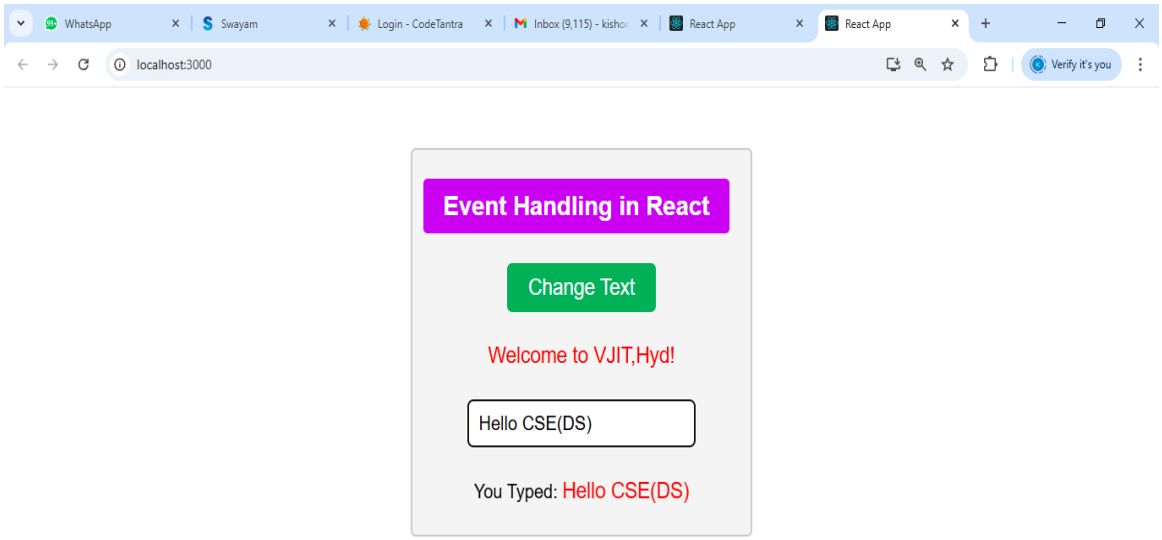
```
        button {
          padding: 10px 20px;
          font-size: 18px;
          margin: 10px;
          cursor: pointer;
          background-color: #4CAF50;
          color: white;
          border: none;
          border-radius: 5px;
        }

        button:hover {
          background-color: #45a049;
        }

        input {
          padding: 10px;
          font-size: 16px;
          margin: 10px;
          border-radius: 5px;
          border: 1px solid #ccc;
}
```

Output:




WhatsApp
Swayam
Login - CodeTantra
Inbox (9,115) - kisho
React App
React App
localhost:3000
Verify it's you
Event Handling in React
Change Text
Welcome to VJIT,Hyd!
Type something
You Typed:


Type here to search
31°C Sunny
11:56 AM
12-Apr-25


WhatsApp
Swayam
Login - CodeTantra
Inbox (9,115) - kisho
React App
React App
localhost:3000
Verify it's you
Event Handling in React
Change Text
Welcome to VJIT,Hyd!
Hello CSE(DS)
You Typed: Hello CSE(DS)


Type here to search
31°C Sunny
11:56 AM
12-Apr-25

6. b) Write a program to create a simple voting application system using ReactJS.

1. Set Up the React Application

If you haven't created a React app yet, use **Create React App** to set up a basic project.

```
npx create-react-app voting-app
cd voting-app
npm start
```

2. Implement Routing

Update App.js to set up routing and implement the navigation between pages.

Folder Structure:

```
src/
  |-- App.js
  |--  App.css
  |-- VotingApp.js
  |--  index.js
```

App.js

```
import React from 'react';
import './App.css';
import VotingApp from './VotingApp';

function App() {
  return (
    <div className="App">
    <VotingApp />
    </div>
  );
}

export default App;
```

VotingApp.js

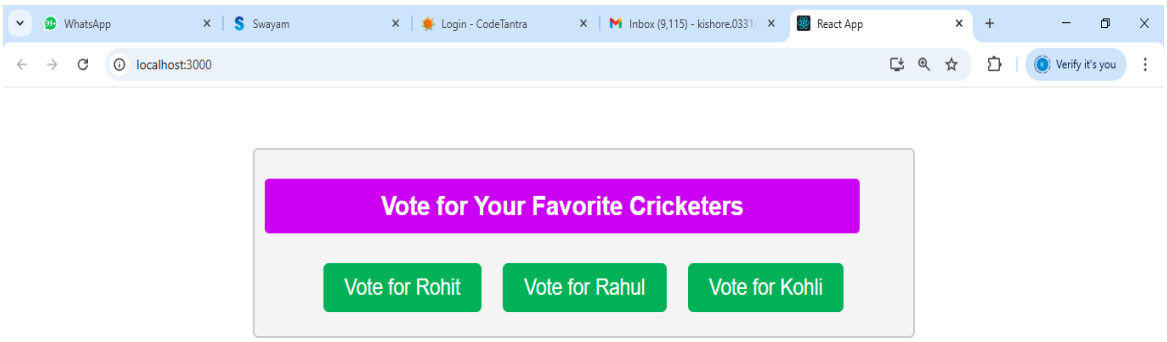
```
import React, { useState } from 'react';
import './App.css';

const VotingApp = () => {
  // State to store votes for each cricketer
  const [votes, setVotes] = useState({
    Rohit: 0,
    Rahul: 0,
    Kohli: 0,
  });

  // Function to handle voting
  const handleVote = (cricketers) => {
    // Update the vote count for the selected cricketers
    setVotes({
      ...votes,
      [cricketers]: votes[cricketers] + 1,
    });
  };

  return (
    <div className="voting-app">
      <div className="header">
      <h1>Vote for Your Favorite Cricketers</h1>
      <div className="vote-options">
        <button onClick={() => handleVote('Rohit')}>Vote for Rohit</button>
        <button onClick={() => handleVote('Rahul')}>Vote for Rahul</button>
        <button onClick={() => handleVote('Kohli')}>Vote for Kohli</button>
      </div>
</div>
      <h2>Vote Results:</h2>
      <div className="vote-results">
        <p>Rohit:<font size="4" color="red"> {votes.Rohit}</font> votes</p>
        <p>Rahul:<font size="4" color="red"> {votes.Rahul}</font> votes</p>
        <p>Kohli: <font size="4" color="red">{votes.Kohli} </font>votes</p>
      </div>
    </div>
  );
};

export default VotingApp;
```

App.css

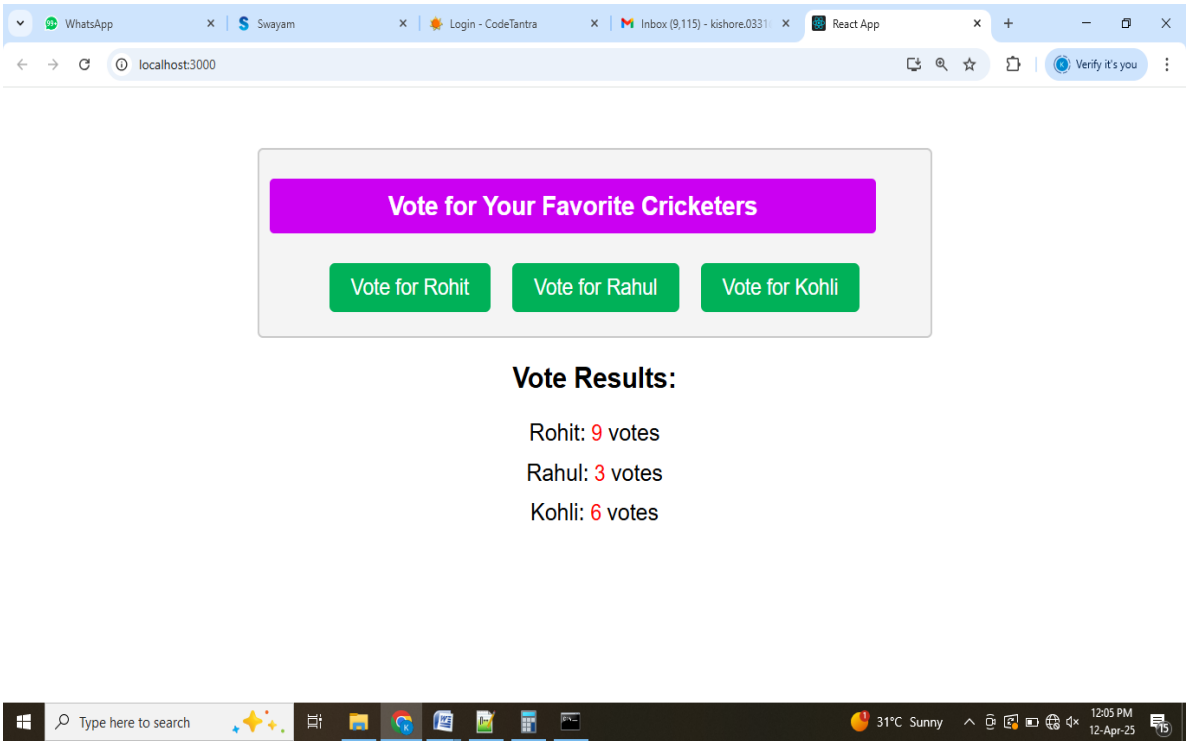
```
.App {
  text-align: center;
}

.App-logo {
  height: 40vmin;
  pointer-events: none;
}

@media (prefers-reduced-motion: no-preference) {
  .App-logo {
    animation: App-logo-spin infinite 20s linear;
  }
}

.App-header {
  background-color: #282c34;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  font-size: calc(10px + 2vmin);
  color: white;
}

.App-link {
  color: #61dafb;
}

@keyframes App-logo-spin {
  from {
    transform: rotate(0deg);
  }
  to {
    transform: rotate(360deg);
  }
}
.header {
  width: 600px;
  margin: 5px auto;
  padding: 10px;
```

```
  border: 2px solid #ccc;
  border-radius: 5px;
  background-color: #f4f4f4;
}
h1 {
        text-align: center;

                width: 90%;
        padding: 10px;
        background-color: #b91df5;
        color: white;
        font-size: 22px;
        border: none;

        border-radius: 4px;
      }
.voting-app {
  text-align: center;
  font-family: Arial, sans-serif;
  margin-top: 50px;
}

.vote-options button {
  padding: 10px 20px;
  font-size: 18px;
  margin: 10px;
  cursor: pointer;
  background-color: #4CAF50;
  color: white;
  border: none;
  border-radius: 5px;
  transition: background-color 0.3s;
}

.vote-options button:hover {
  background-color: #45a049;
}

.vote-results p {
  font-size: 20px;
  margin: 10px;
}
```

Output:


WhatsApp
Swayam
Login - CodeTantra
Inbox (9,115) - kishore.0331
React App
localhost:3000
Verify it's you
Vote for Your Favorite Cricketers
Vote for Rohit
Vote for Rahul
Vote for Kohli
Vote Results:
Rohit: 0 votes
Rahul: 0 votes
Kohli: 0 votes


Type here to search
31°C Sunny
12:04 PM
12-Apr-25


WhatsApp
Swayam
Login - CodeTantra
Inbox (9,115) - kishore.0331
React App
localhost:3000
Verify it's you
Vote for Your Favorite Cricketers
Vote for Rohit
Vote for Rahul
Vote for Kohli
Vote Results:
Rohit: 9 votes
Rahul: 3 votes
Kohli: 6 votes

Type here to search
31°C Sunny
12:05 PM
12-Apr-25

7. a) Create a webpage to display "Hello World" using SERVLET.

1. Set Up to Develop Java Web Application using Servlets

- **JDK** (Java Development Kit) – e.g., Java 8 or above
- **Eclipse IDE for Enterprise Java Developers**
- **Apache Tomcat** (Servlet container)

Step-by-Step: Create "Hello World" Servlet in Eclipse

Step 1: Set Up Eclipse with Tomcat

1. **Open Eclipse**
2. Go to **Window** → **Preferences** → **Server** → **Runtime Environments**
3. Click **Add**
4. Choose **Apache Tomcat v9.0** (or any version you have), click **Next**
5. Browse to the **Tomcat installation directory**, click **Finish**

Step 2: Create a Dynamic Web Project

1. Go to **File** → **New** → **Dynamic Web Project**
2. Enter project name: **HelloWorldServlet**
3. Target runtime: Select **Apache Tomcat**
4. Configuration: Leave default (Dynamic web module version 3.1 or 4.0)
5. Click **Finish**

Eclipse will generate a standard project structure for Java EE:

```
HelloWorldServlet/
├── src/
├── Webapp/
│   └── WEB-INF/
│       └── web.xml
```

Step 3: Create a Servlet

1. Right-click **src** → **New** → **Servlet**
2. Name: **HelloWorld**
3. Package: **com.helloworld**
4. Click **Finish**

HelloWorldServlet.java

```
package com.helloworld;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloWorldServlet extends HttpServlet
 {
   protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
 {
       // Set the content type of the response
       response.setContentType("text/html");

       // Get the output stream to send the response to the client
       PrintWriter out = response.getWriter();

       // Write the HTML content to the response
       out.println("<html><body>");
       out.println("<h1><font size=\"4\" color=\"red\">&quot;Hello World&quot; from
Servlet!</font></h1>");
       out.println("</body></html>");
   }
}
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd" id="WebApp_ID"
version="4.0">
  <display-name>HelloWorldServlet</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.jsp</welcome-file>
    <welcome-file>default.htm</welcome-file>
  </welcome-file-list>
```

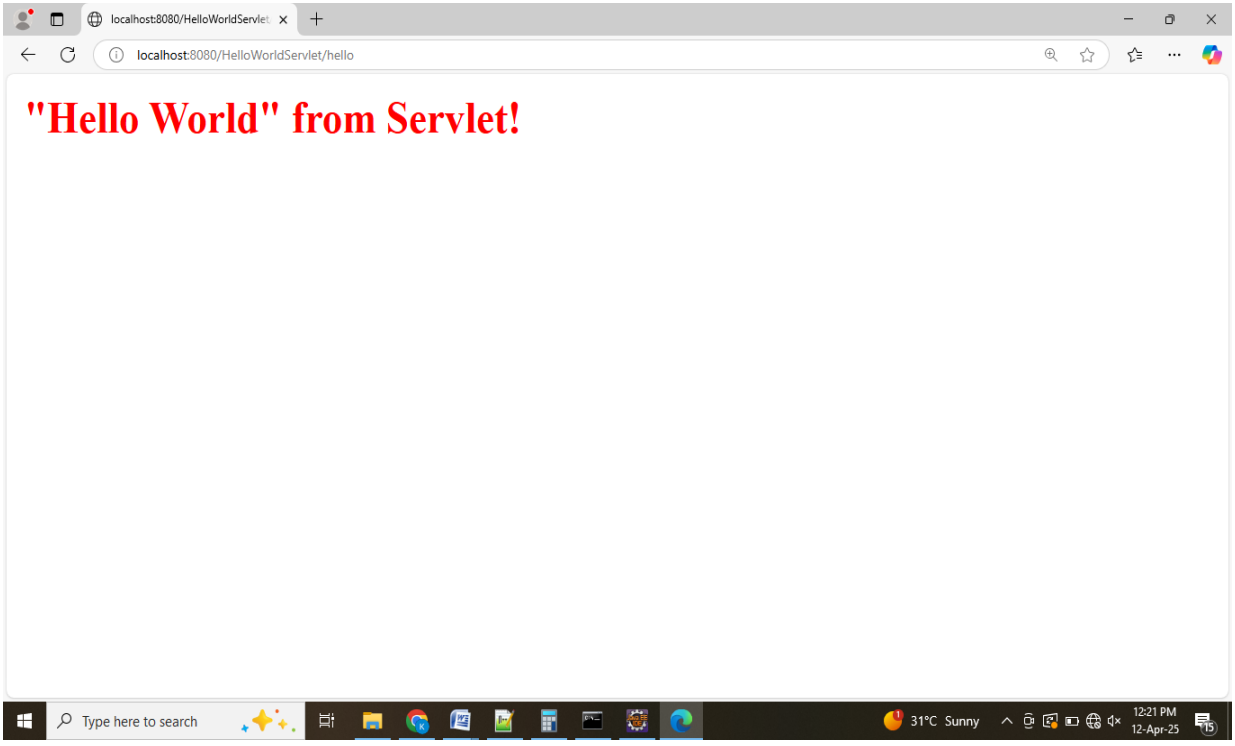
```
<servlet>
        <servlet-name>HelloWorldServlet</servlet-name>
        <servlet-class>com.helloworld.HelloWorldServlet</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>HelloWorldServlet</servlet-name>
        <url-pattern>/hello</url-pattern>
    </servlet-mapping>
</web-app>
```

Step 4: Run the Servlet

1. Right-click the project → **Run As** → **Run on Server**
2. Select **Apache Tomcat**, click **Finish**
3. The browser opens with:

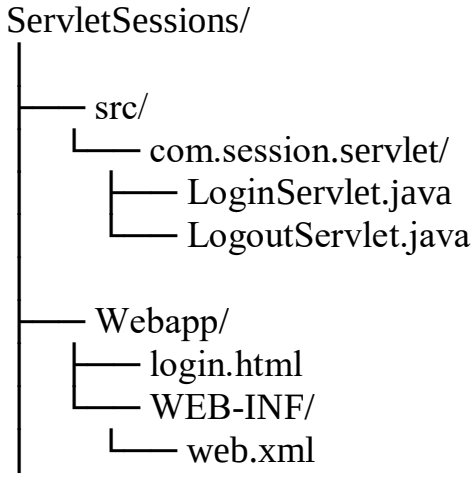
   http://localhost:8080/HelloWorldServlet/hello

Output:



7.b) Implement a web application using SERVLET, which takes a name as input and on submitting it, shows a hello <name> page. It shows start time at the right top comer of the page and provides a logout button. On clicking logout button, it should show a logout page with Thank You <name> message with the duration of usage (hint: Use session to store name and time).

Project Structure



LoginServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.time.LocalTime;

public class LoginServlet extends HttpServlet
 {

   protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
 {
      // Get the name from the request
      String name = request.getParameter("name");

      // Create a new session or get the existing one
      HttpSession session = request.getSession();

      // Store the name and current time in the session
      session.setAttribute("name", name);
      session.setAttribute("startTime", System.currentTimeMillis()); // Current time in
milliseconds
```

```
        // Set content type and response writer
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        // Generate the welcome page
        out.println("<html>");
        out.println("<head><title>Welcome Page</title></head>");
        out.println("<body>");
        out.println("<h1>Hello " + name + "</h1>");

        // Display the start time in the top-right corner
        out.println("<div style='position: absolute; top: 10px; right: 10px;'>Start Time: "
+ LocalTime.now() + "</div>");

        // Button for logging out
        out.println("<form action='LogoutServlet' method='POST'>");
        out.println("<input type='submit' value='Logout'>");
        out.println("</form>");

        out.println("</body>");
        out.println("</html>");
    }
}
```

LogoutServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class LogoutServlet extends HttpServlet
 {

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
 {
        // Get the session
        HttpSession session = request.getSession(false);

        if (session != null)
       {
            // Get the user's name and session start time from the session
            String name = (String) session.getAttribute("name");
            long startTime = (Long) session.getAttribute("startTime");
```

```
        // Calculate the session duration in seconds
        long duration = (System.currentTimeMillis() - startTime) / 1000;  // in seconds

        // Invalidate the session (log out)
        session.invalidate();

        // Set content type and response writer
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        // Display the Thank You message with the session duration
        out.println("<html>");
        out.println("<head><title>Logout Page</title></head>");
        out.println("<body>");
        out.println("<h1>Thank You " + name + "</h1>");
        out.println("<p>Your session lasted for " + duration + " seconds.</p>");
        out.println("</body>");
        out.println("</html>");
    }
  }
}
```

login.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Login Page</title>
  <style>
    body
    {
        font-family: Arial, sans-serif;
    }
    .form-container
    {
        text-align: center; margin-top: 50px;
    }
    input[type="text"]
    {
        padding: 8px; font-size: 16px;
    }
    input[type="submit"]
     {
```

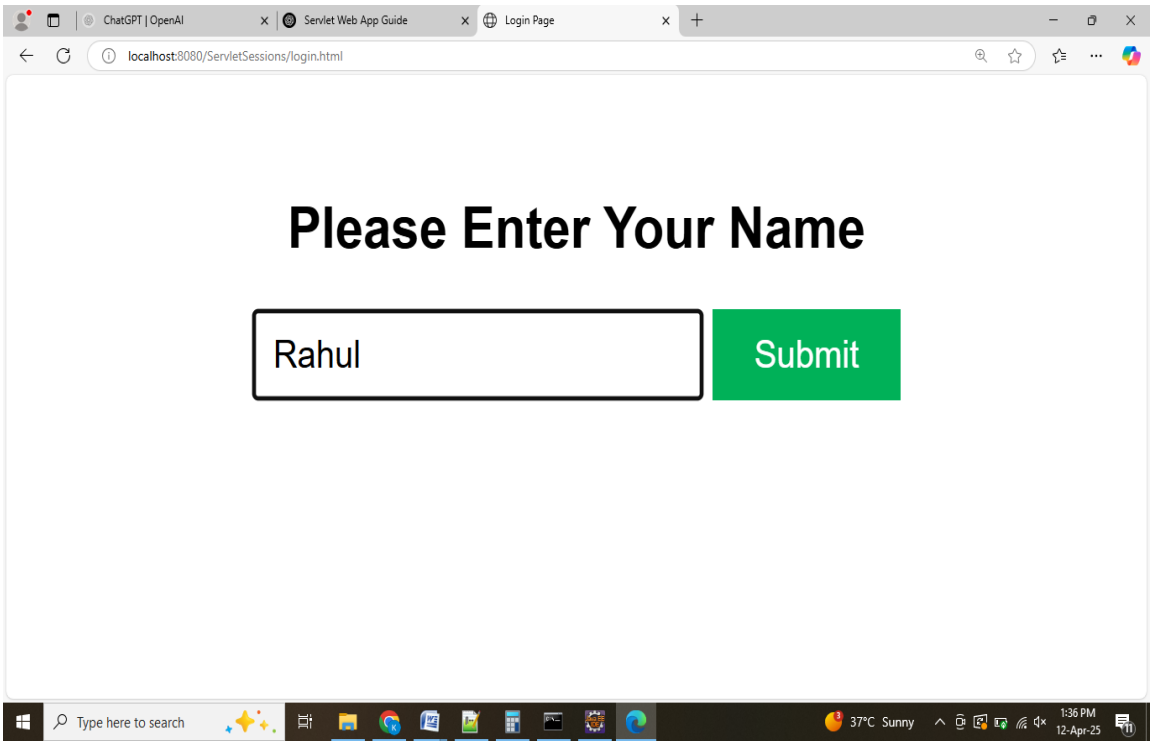
```
                padding: 10px 20px; font-size: 16px; background-color: #4CAF50;
color: white; border: none; cursor: pointer;
        }
    </style>
</head>


<body>
    <div class="form-container">
        <h2>Please Enter Your Name</h2>
        <form action="LoginServlet" method="POST">
            <input type="text" name="name" required placeholder="Enter your name">
            <input type="submit" value="Submit">
        </form>
    </div>
</body>
</html>
```

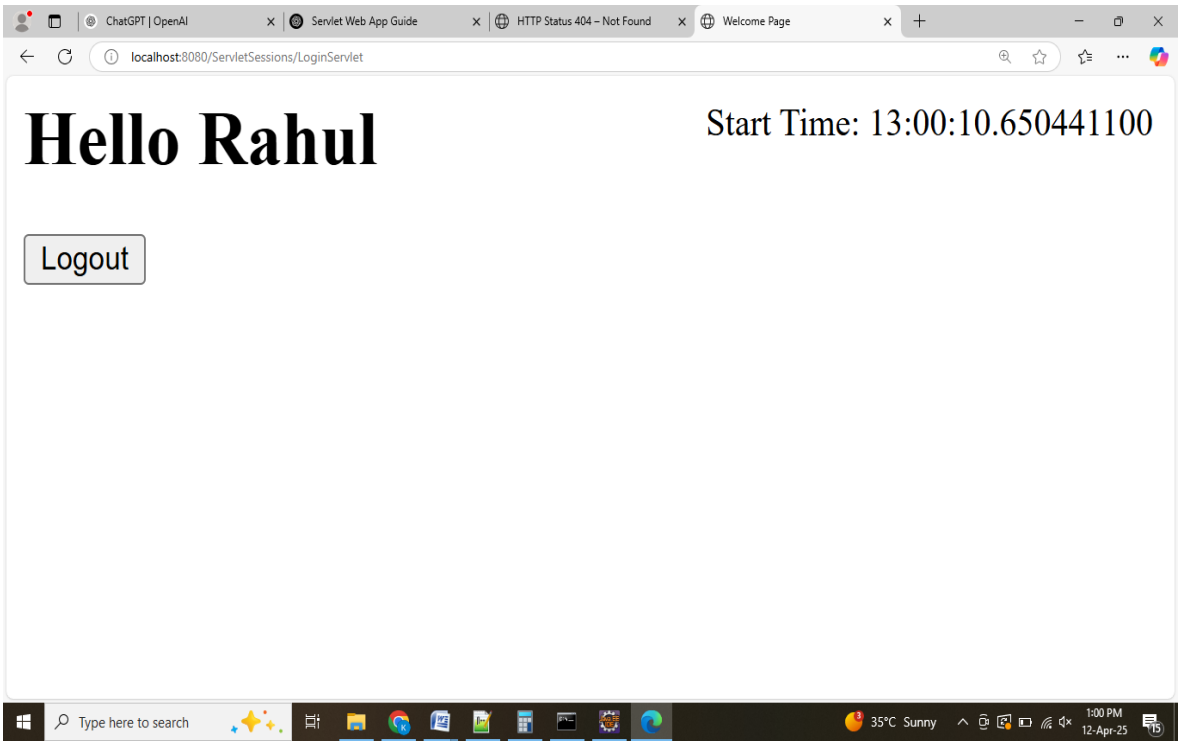
web.xml

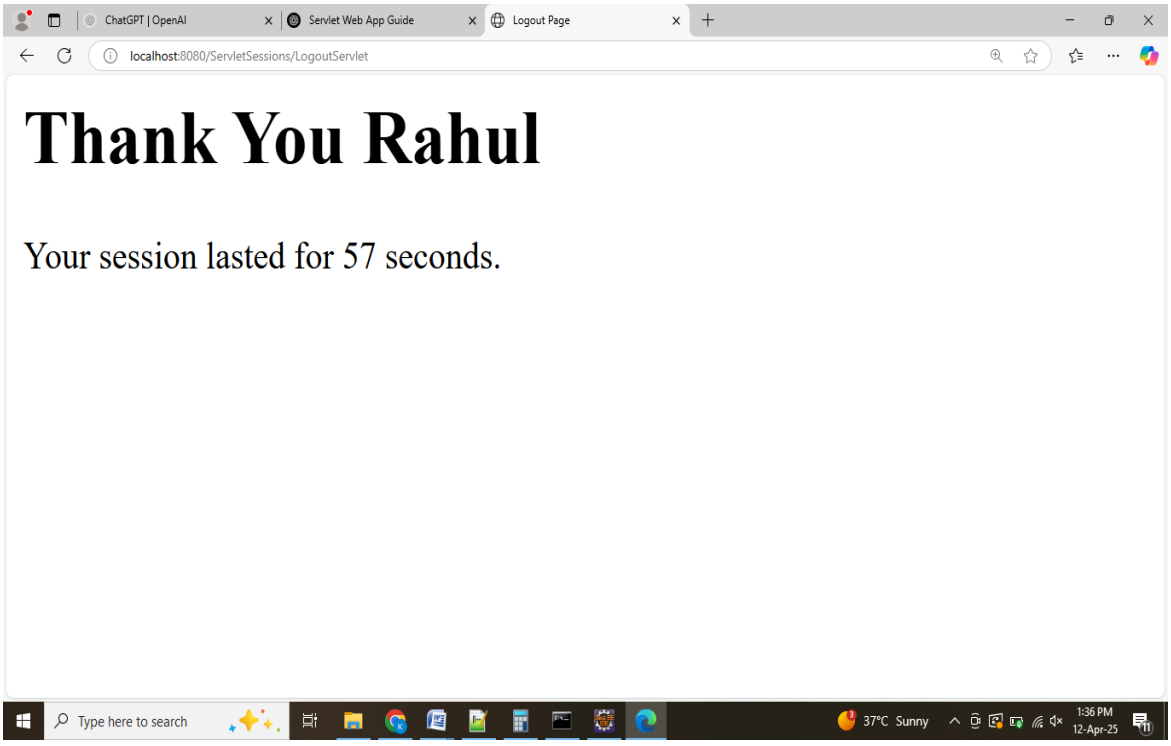
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd" id="WebApp_ID"
version="4.0">
  <display-name>ServletSessions</display-name>
  <welcome-file-list>
    <welcome-file>login.html</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.jsp</welcome-file>
    <welcome-file>default.htm</welcome-file>
  </welcome-file-list>
  <!-- Login Servlet Configuration -->
    <servlet>
        <servlet-name>LoginServlet</servlet-name>
        <servlet-class>LoginServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>LoginServlet</servlet-name>
        <url-pattern>/LoginServlet</url-pattern>
    </servlet-mapping>
    <!-- Logout Servlet Configuration -->
    <servlet>
```

```
    <servlet-name>LogoutServlet</servlet-name>
    <servlet-class>LogoutServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>LogoutServlet</servlet-name>
    <url-pattern>/LogoutServlet</url-pattern>
  </servlet-mapping>
</web-app>
```

Output:

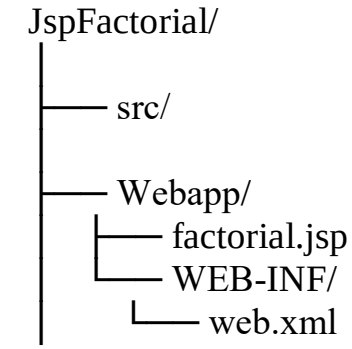



ChatGPT | OpenAI
Servlet Web App Guide
HTTP Status 404 – Not Found
Welcome Page
localhost:8080/ServletSessions/LoginServlet
Hello Rahul
Start Time: 13:00:10.650441100
Logout
Type here to search
35°C Sunny
1:00 PM
12-Apr-25


ChatGPT | OpenAI
Servlet Web App Guide
Logout Page
localhost:8080/ServletSessions/LogoutServlet
Thank You Rahul
Your session lasted for 57 seconds.
Type here to search
37°C Sunny
1:36 PM
12-Apr-25

8. a) Write a JSP program to find a factorial of the given number.

Project Structure



factorial.jsp

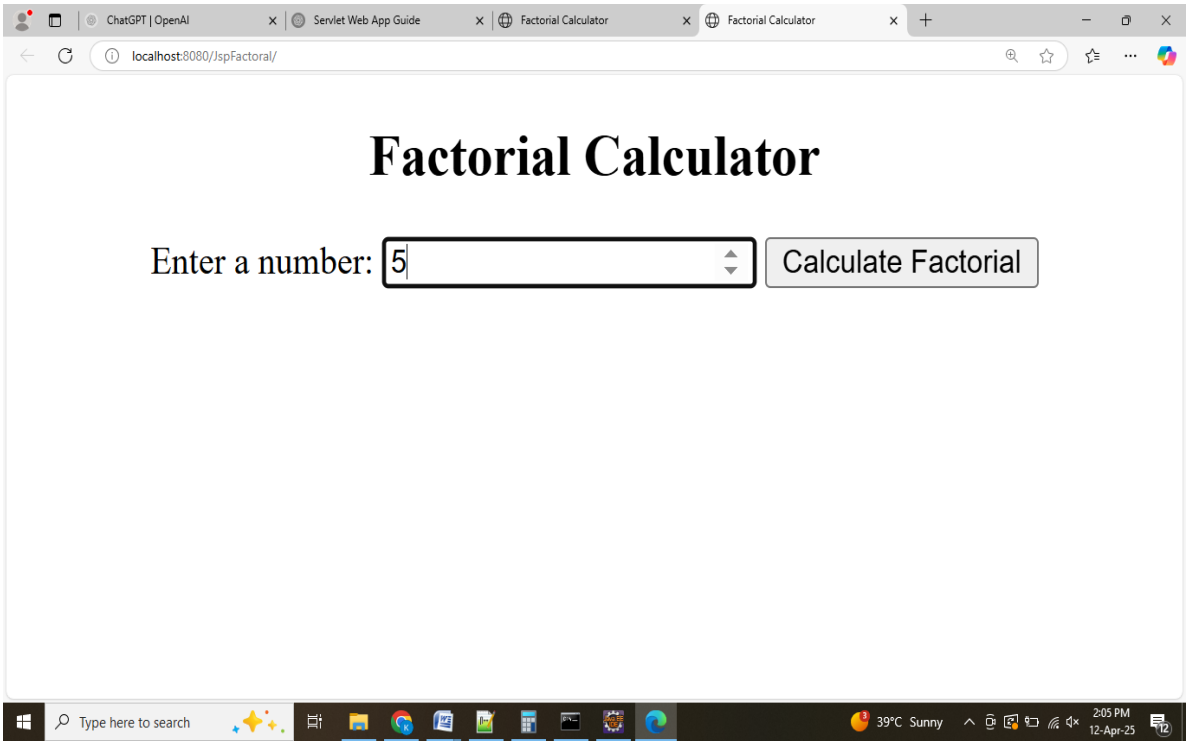
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html lang="en">

<head>

    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Factorial Calculator</title>

</head>

<body>

    <h2>Factorial Calculator</h2>

    <form action="factorial.jsp" method="POST">

        Enter a number: <input type="number" name="number" required>
        <input type="submit" value="Calculate Factorial">

  </form>
```

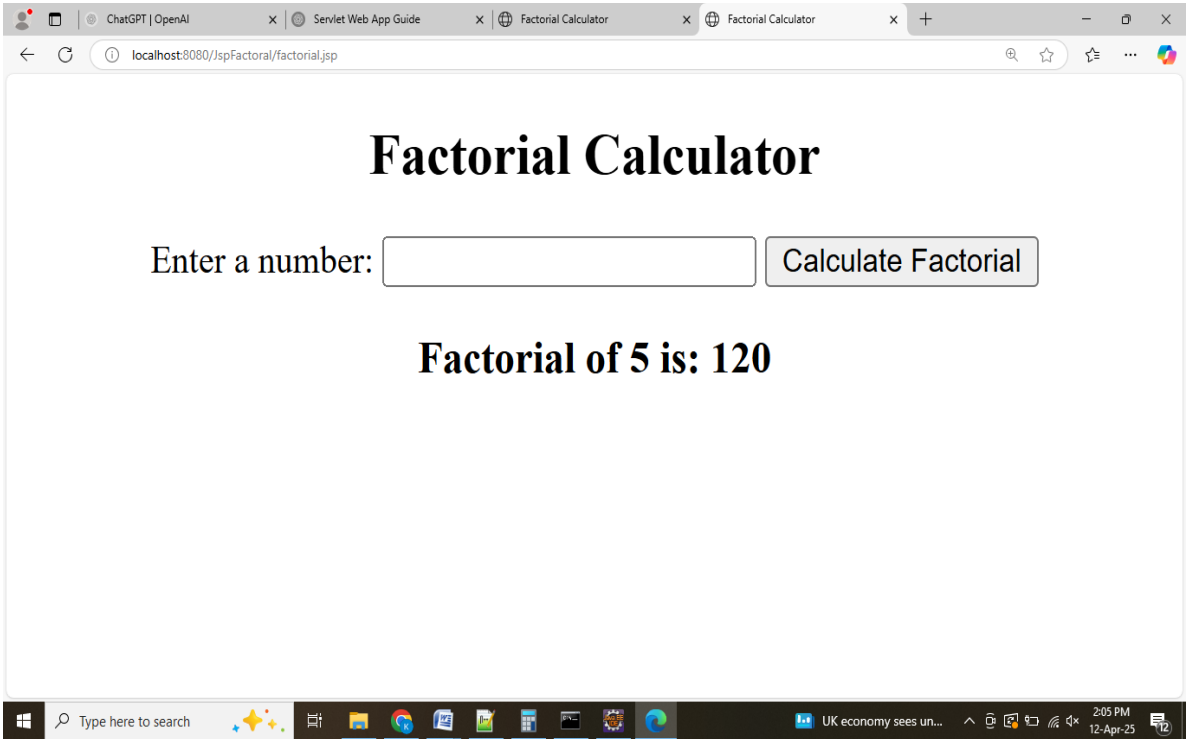
```
<%
    // Check if the request has the "number" parameter
    String numberParam = request.getParameter("number");
    if (numberParam != null && !numberParam.isEmpty())
    {
        try
        {
            int number = Integer.parseInt(numberParam);

            // Method to calculate factorial
            long factorial = 1;
            for (int i = 1; i <= number; i++)
             {
                factorial *= i;
            }
            out.println("<h3>Factorial of " + number + " is: " + factorial + "</h3>");
        }
        catch (NumberFormatException e)
        {
            out.println("<h3>Please enter a valid number.</h3>");
        }
    }
  %>
</body>
</html>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd" id="WebApp_ID"
version="4.0">
  <display-name>JspFactoral</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>factorial.jsp</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.jsp</welcome-file>
    <welcome-file>default.htm</welcome-file>
  </welcome-file-list>
</web-app>
```

Output:


ChatGPT | OpenAI
Servlet Web App Guide
Factorial Calculator
Factorial Calculator
localhost:8080/JspFactoral/
Factorial Calculator
Enter a number: 5
Calculate Factorial
Type here to search
39°C Sunny
2:05 PM
12-Apr-25


ChatGPT | OpenAI
Servlet Web App Guide
Factorial Calculator
Factorial Calculator
localhost:8080/JspFactoral/factorial.jsp
Factorial Calculator
Enter a number:
Calculate Factorial
Factorial of 5 is: 120
Type here to search
UK economy sees un...
2:05 PM
12-Apr-25

8. b) Create a user validation web application using JSP, where the user submits the login name and password to the server. The name and password are checked against the data already available in database and if the data matches, a successfull login page is returned. Otherwise show a failure message to the user.

Project Structure

```
UserLogin/
│
├── src/
│   └── com.login.servlet/
│       ├── LoginServlet.java
│
├── Webapp/
│   ├── login.jsp
│   |-----loginSuccess.jsp
│   |------logout.jsp
│   └── WEB-INF/
│       └── web.xml
```

Add MySQL Connector

- Download the **MySQL Connector JAR**.
- Add it to your project:
  - Right-click project > **Build Path** > **Configure Build Path**
  - Add External JAR > Select **mysql-connector-java-x.x.x.jar**

Create MySQL Table

- Run this SQL in your MySQL server:

```
CREATE DATABASE vjit;

USE vjit;

CREATE TABLE users (
    username VARCHAR(50) PRIMARY KEY,
    password VARCHAR(50) NOT NULL
);

INSERT INTO users VALUES ('admin', 'admin123'), ('user1', 'pass1');
```

login.jsp

```
<!DOCTYPE html>
<html>
<head>
   <title>User Login</title>
</head>
<body>
   <h2 align="center">User Login</h2>
   <form action="login" method="POST">
      <p align="center"> <label for="username">Username: </label>
      <input type="text" id="username" name="username" required><br><br>

      <label for="password">Password: </label>
      <input type="password" id="password" name="password" required><br><br>

      <button type="submit">Login</button></p>
   </form>
   <p style="color: red;"><%= request.getAttribute("errorMessage") != null ?
request.getAttribute("errorMessage") : "" %></p>
</body>
</html>
```

LoginServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;

public class LoginServlet extends HttpServlet
 {
private static final String DB_URL = "jdbc:mysql://localhost:3306/vjit";  // Database URL
    private static final String DB_USER = "root"; // MySQL username
    private static final String DB_PASSWORD = "vjit"; // MySQL password

    // Database connection method
    private Connection getConnection() throws SQLException
    {
        try
        {
            Class.forName("com.mysql.cj.jdbc.Driver");
            return DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
        }
```

```
        catch (ClassNotFoundException | SQLException e)
        {
            throw new SQLException("Database connection error", e);
         }
    }

    // Handle POST request for login
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
 {
        String username = request.getParameter("username");
        String password = request.getParameter("password");

        // Check if username and password are valid
        try (Connection conn = getConnection())
       {
            String query = "SELECT * FROM users WHERE username = ? AND
password = ?";
            try (PreparedStatement stmt = conn.prepareStatement(query))
           {
                stmt.setString(1, username);
                stmt.setString(2, password);
                ResultSet rs = stmt.executeQuery();

                if (rs.next())
                {
                    // If the user exists, redirect to the success page
                    response.sendRedirect("loginSuccess.jsp");
                }
                 else
                 {
                    // If the user doesn't exist, show failure message
                     request.setAttribute("errorMessage", "Invalid username or password");
                    request.getRequestDispatcher("login.jsp").forward(request, response);
                }
            }
        }
         catch (SQLException e)
          {
            e.printStackTrace();
           response.getWriter().println("Error connecting to database: " + e.getMessage());
        }
    }
}
```

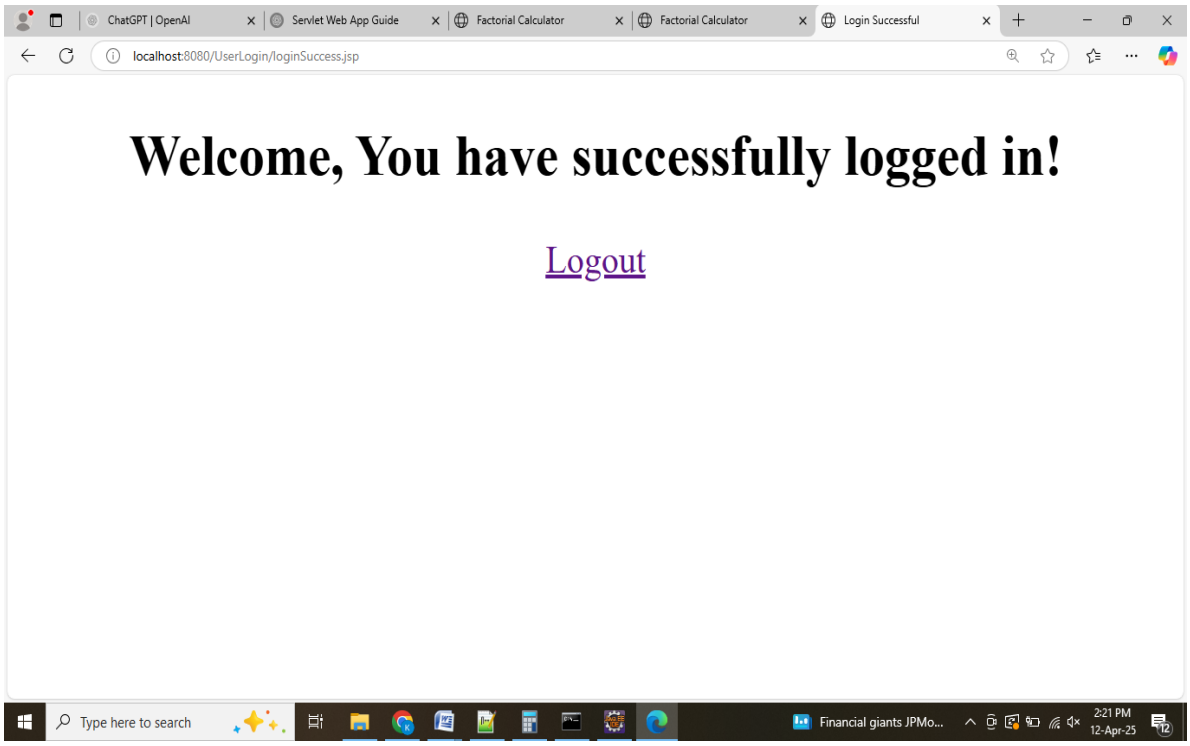
loginSuccess.jsp

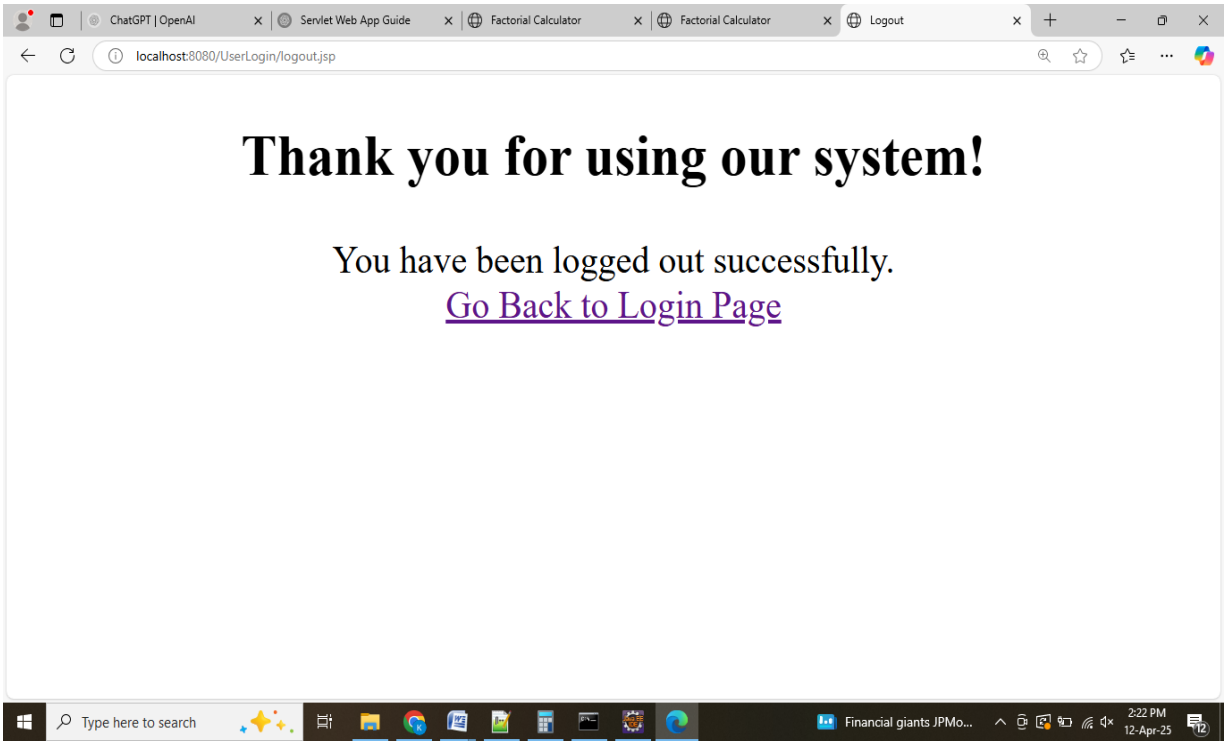
```
<!DOCTYPE html>
<html>
<head>
   <title>Login Successful</title>
</head>
<body>
    <h2 align="center">Welcome, You have successfully logged in!</h2>
  <p align="center"> <a href="logout.jsp">Logout</a></p>
</body>
</html>
```

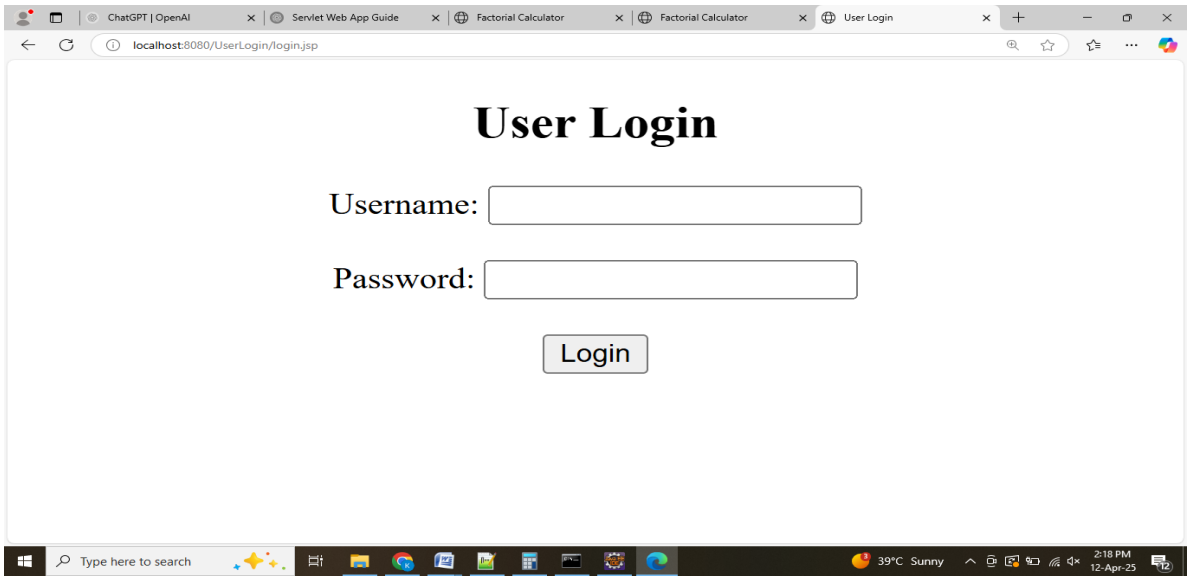
logout.jsp

```
    <!DOCTYPE html>
    <html>
    <head>
       <title>Logout</title>
    </head>
    <body>

       <h2 align="center">Thank you for using our system!</h2>
       <p align="center">You have been logged out successfully.<br>
       <a href="login.jsp">Go Back to Login Page</a></p>
    </body>
    </html>
```

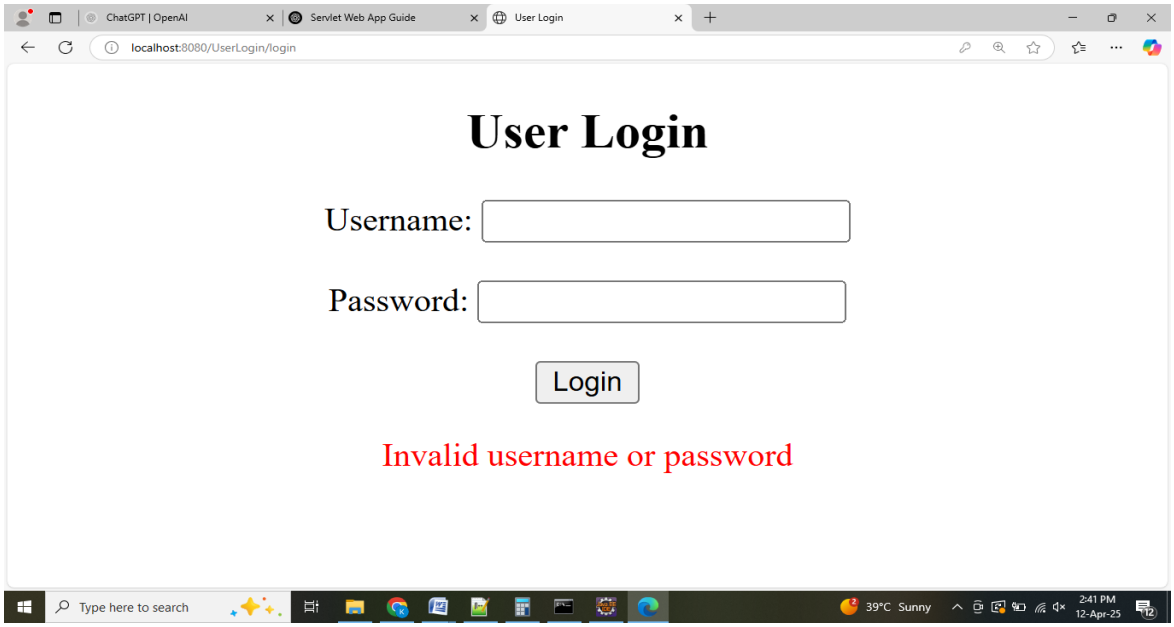
Output:


ChatGPT | OpenAI
Servlet Web App Guide
Factorial Calculator
Factorial Calculator
Login Successful
localhost:8080/UserLogin/loginSuccess.jsp
Welcome, You have successfully logged in!
Logout
Type here to search
Financial giants JPMo...
2:21 PM
12-Apr-25


ChatGPT | OpenAI
Servlet Web App Guide
Factorial Calculator
Factorial Calculator
Logout
localhost:8080/UserLogin/logout.jsp
Thank you for using our system!
You have been logged out successfully.
Go Back to Login Page
Type here to search
Financial giants JPMo...
2:22 PM
12-Apr-25


ChatGPT | OpenAI
Servlet Web App Guide
Factorial Calculator
Factorial Calculator
User Login
localhost:8080/UserLogin/login.jsp
User Login
Username:
Password:
Login
Type here to search
39°C Sunny
2:18 PM
12-Apr-25


ChatGPT | OpenAI
Servlet Web App Guide
User Login
localhost:8080/UserLogin/login.jsp
User Login
Username: vjit
Password: vjit123
Login
Type here to search
39°C Sunny
2:39 PM
12-Apr-25


ChatGPT | OpenAI
Servlet Web App Guide
User Login
localhost:8080/UserLogin/login
User Login
Username:
Password:
Login
Invalid username or password
Type here to search
39°C Sunny
2:41 PM
12-Apr-25

9. a) Demonstrate a simple example of Spring web MVC framework.

Project Structure

```
SpringMVCDemo/
├
├── src/
│   └── com/example/controller/HelloController.java
├── Webapp/
│   ├── index.jsp
│   ├── welcome.jsp
│   └── WEB-INF/
│       ├── web.xml
│       ├── dispatcher-servlet.xml
│       └── lib/ (Spring JARs)
```

Step 1: Create a Dynamic Web Project

1. Open Eclipse
2. Go to **File → New → Dynamic Web Project**
3. Name the project: **SpringMVCDemo**
4. Select **Target runtime**: **Apache Tomcat v9.0**
5. Click **Finish**

Step 2: Add Required Spring JARs

- Copy these into:
  Webapp/WEB-INF/lib/

Required JARs:

- **spring-webmvc.jar**
- **spring-core.jar**
- **spring-context.jar**
- **spring-beans.jar**
- **commons-logging.jar**

✅Add to build path:
Right-click project → **Build Path** → **Configure Build Path** → Add JARs

---

Step 3: Configure web.xml

File path: **Webapp/WEB-INF/web.xml**

```
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
        http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
        version="3.1">

   <display-name>Spring MVC Example</display-name>

   <!-- Spring Front Controller -->
   <servlet>
      <servlet-name>dispatcher</servlet-name>
      <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-
class>
      <load-on-startup>1</load-on-startup>
   </servlet>

   <servlet-mapping>
      <servlet-name>dispatcher</servlet-name>
      <url-pattern>/</url-pattern>
   </servlet-mapping>

</web-app>
```

Step 4: Create dispatcher-servlet.xml

File path: **Webapp/WEB-INF/dispatcher-servlet.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:context="http://www.springframework.org/schema/context"
      xmlns:mvc="http://www.springframework.org/schema/mvc"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="
         http://www.springframework.org/schema/beans
         http://www.springframework.org/schema/beans/spring-beans.xsd
         http://www.springframework.org/schema/context
         http://www.springframework.org/schema/context/spring-context.xsd
         http://www.springframework.org/schema/mvc
         http://www.springframework.org/schema/mvc/spring-mvc.xsd">
```

```
    <!-- Enable @Controller annotation support -->
    <mvc:annotation-driven />

    <!-- Scan controller package -->
    <context:component-scan base-package="com.example.controller" />

    <!-- View resolver -->
    <bean
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/" />
        <property name="suffix" value=".jsp" />
    </bean>

</beans>
```

Step 5: Create Controller

Package: **src/com/example/controller**
File: **HelloController.java**

```
package com.example.controller;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;

@Controller
public class HelloController
 {

    @RequestMapping(value = "/hello", method = RequestMethod.POST)
    public String sayHello(@RequestParam("username") String name, Model model)
{
        model.addAttribute("message", "Hello, " + name + "!");
        return "welcome";
    }
}
```

Step 6: Create JSP Files

File: **Webapp/index.jsp**

```
<!DOCTYPE html>
<html>
<head>
   <title>Spring MVC Example</title>
</head>
<body>
   <h2>Enter Your Name</h2>
   <form action="hello" method="post">
      Name: <input type="text" name="username" />
      <input type="submit" value="Submit" />
   </form>
</body>
</html>
```

File: **Webapp/welcome.jsp**

```
<!DOCTYPE html>
<html>
<head>
   <title>Welcome</title>
</head>
<body>
   <h2>${message}</h2>
</body>
</html>
```

Step 7: Run Your App

1. Right-click project → **Run As → Run on Server**
2. Open browser:
   http://localhost:8080/SpringMVCDemo/

9. b) Illustrate how database is connected in Spring Framework by using simple CRUD application.

1. Project Structure

```
StudentCRUD/
├── src/
│   ├── com.example.controller.StudentController.java
│   ├── com.example.dao.StudentDAO.java
│   ├── com.example.dao.StudentDAOImpl.java
│   ├── com.example.model.Student.java
│   └── com.example.config.DBConfig.java
├── Webapp/
│   ├── index.jsp
│   ├── list.jsp
│   ├── add.jsp
│   └── WEB-INF/
│       ├── web.xml
│       └── dispatcher-servlet.xml
│       └── lib/ (Spring JARs + MySQL JDBC driver)
```

---

2. Add Required JARs to WEB-INF/lib

Include the following:

- **spring-core.jar**
- **spring-context.jar**
- **spring-jdbc.jar**
- **spring-beans.jar**
- **spring-webmvc.jar**
- **commons-logging.jar**
- **mysql-connector-java-x.x.xx.jar**

---

3. Database Setup (MySQL)

```
CREATE DATABASE studentdb;

USE studentdb;

CREATE TABLE student (
   id INT AUTO_INCREMENT PRIMARY KEY,
   name VARCHAR(50),
   email VARCHAR(50)
);
```

4. web.xml (Front Controller Setup)

```
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
        http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
        version="3.1">

   <display-name>Student CRUD App</display-name>

   <servlet>
      <servlet-name>dispatcher</servlet-name>
      <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-
class>
      <load-on-startup>1</load-on-startup>
   </servlet>

   <servlet-mapping>
      <servlet-name>dispatcher</servlet-name>
      <url-pattern>/</url-pattern>
   </servlet-mapping>

</web-app>
```

5. dispatcher-servlet.xml (Spring Config)

```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="
          http://www.springframework.org/schema/beans
          http://www.springframework.org/schema/beans/spring-beans.xsd
          http://www.springframework.org/schema/context
          http://www.springframework.org/schema/context/spring-context.xsd
          http://www.springframework.org/schema/mvc
          http://www.springframework.org/schema/mvc/spring-mvc.xsd">

    <context:component-scan base-package="com.example" />
    <mvc:annotation-driven />

    <bean
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/" />
        <property name="suffix" value=".jsp" />
    </bean>

    <import resource="db-config.xml" />
</beans>
```

---

6. db-config.xml (Database Connection)

```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd">

    <!-- DataSource -->
    <bean id="dataSource"
class="org.springframework.jdbc.datasource.DriverManagerDataSource">
        <property name="driverClassName" value="com.mysql.cj.jdbc.Driver" />
        <property name="url" value="jdbc:mysql://localhost:3306/studentdb" />
        <property name="username" value="root" />
        <property name="password" value="your_password" />
    </bean>
```

```
    <!-- JDBC Template -->
    <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
        <property name="dataSource" ref="dataSource" />
    </bean>
</beans>
```

---

7. Student.java (Model)

```
package com.example.model;

public class Student
 {
    private int id;
    private String name;
    private String email;

    // Getters & Setters
}
```

---

8. DAO Interface & Implementation

StudentDAO.java

```
package com.example.dao;

import java.util.List;
import com.example.model.Student;

public interface StudentDAO
{
    void save(Student s);
    List<Student> getAll();
    void delete(int id);
    void update(Student s);
    Student getById(int id);
}
```

StudentDAOImpl.java

```
package com.example.dao;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.List;

import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.RowMapper;

import com.example.model.Student;

public class StudentDAOImpl implements StudentDAO
{

   private JdbcTemplate jdbcTemplate;

   public void setJdbcTemplate(JdbcTemplate jdbcTemplate)
{
      this.jdbcTemplate = jdbcTemplate;
   }

   public void save(Student s)
{
      String sql = "INSERT INTO student (name, email) VALUES (?, ?)";
      jdbcTemplate.update(sql, s.getName(), s.getEmail());
   }

   public List<Student> getAll()
{
      return jdbcTemplate.query("SELECT * FROM student", new
RowMapper<Student>()
{
         public Student mapRow(ResultSet rs, int rowNum) throws SQLException
{
            Student s = new Student();
            s.setId(rs.getInt("id"));
            s.setName(rs.getString("name"));
            s.setEmail(rs.getString("email"));
            return s;
         }
      });
   }
```

```
    public void delete(int id)
 {
        jdbcTemplate.update("DELETE FROM student WHERE id = ?", id);
    }

    public void update(Student s)
{
        String sql = "UPDATE student SET name = ?, email = ? WHERE id = ?";
        jdbcTemplate.update(sql, s.getName(), s.getEmail(), s.getId());
    }

    public Student getById(int id)
{
        String sql = "SELECT * FROM student WHERE id = ?";
        return jdbcTemplate.queryForObject(sql, new Object[]{id}, new
RowMapper<Student>()
{
            public Student mapRow(ResultSet rs, int rowNum) throws SQLException
 {
                Student s = new Student();
                s.setId(rs.getInt("id"));
                s.setName(rs.getString("name"));
                s.setEmail(rs.getString("email"));
                return s;
            }
        });
    }
}
```

9. Controller: StudentController.java

```
package com.example.controller;

import com.example.dao.StudentDAO;
import com.example.model.Student;
import java.util.List;
import javax.annotation.Resource;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
```

```
@Controller
public class StudentController
{

    @Resource
    private StudentDAO studentDAO;

    @GetMapping("/")
    public String viewHome(Model model)
{
        List<Student> list = studentDAO.getAll();
        model.addAttribute("students", list);
        return "list";
    }

    @GetMapping("/add")
    public String showAddForm(Model model)
 {
        model.addAttribute("student", new Student());
        return "add";
    }

    @PostMapping("/save")
    public String save(@ModelAttribute("student") Student student)
{
        studentDAO.save(student);
        return "redirect:/";
    }

    @GetMapping("/delete/{id}")
    public String delete(@PathVariable int id)
{
        studentDAO.delete(id);
        return "redirect:/";
    }

    @GetMapping("/edit/{id}")
    public String edit(@PathVariable int id, Model model)
{
        model.addAttribute("student", studentDAO.getById(id));
        return "add";
    }
```

```
 @PostMapping("/update")
   public String update(@ModelAttribute("student") Student student)
{
      studentDAO.update(student);
      return "redirect:/";
   }
}
```

10. JSP Pages

list.jsp

```
<h2>Students List</h2>
<a href="add">Add New</a>
<table border="1">
<tr><th>ID</th><th>Name</th><th>Email</th><th>Actions</th></tr>
<c:forEach items="${students}" var="s">
<tr>
<td>${s.id}</td>
<td>${s.name}</td>
<td>${s.email}</td>
<td>
  <a href="edit/${s.id}">Edit</a>
  <a href="delete/${s.id}">Delete</a>
</td>
</tr>
</c:forEach>
</table>
```

add.jsp

```
<h2>Add / Edit Student</h2>
<form action="${student.id == 0 ? 'save' : 'update'}" method="post">
  <input type="hidden" name="id" value="${student.id}"/>
  Name: <input type="text" name="name" value="${student.name}"/><br/>
  Email: <input type="text" name="email" value="${student.email}"/><br/>
  <input type="submit" value="Save"/>
</form>
```

10.a) Create a simple example of hibernate application using eclipse IDE

Project Structure

```
HibernateWebApp/
├── src/
│   └── com.example.model/
│       └── User.java
│   └── com.example.util/
│       └── HibernateUtil.java
│   └── com.example.servlet/
│       └── RegisterServlet.java
├── Webapp/
│   ├── register.jsp
│   ├── success.jsp
│   └── WEB-INF/
│       ├── web.xml
│       ├── lib/ (JAR files)
│       └── hibernate.cfg.xml
```

---

1. Create a Dynamic Web Project

1. Open Eclipse
2. Go to **File → New → Dynamic Web Project**
3. Project Name: **HibernateWebApp**
4. Select Apache Tomcat as Target Runtime
5. Click Finish

---

2. Add Hibernate + MySQL JAR Files

Place the following JARs in **Webapp/WEB-INF/lib**:

Required JARs:

- **hibernate-core-x.x.x.Final.jar**
- **hibernate-commons-annotations.jar**
- **hibernate-jpa-x.x-api.jar**
- **antlr.jar**, **dom4j.jar**, **jboss-logging.jar**
- **mysql-connector-java-x.x.xx.jar** (MySQL Driver)

Right-click → Build Path → Configure Build Path → Add JARs

3. MySQL Database Setup

CREATE **DATABASE hibernate_webapp;**

USE hibernate_webapp;

-- Table will be auto-created by Hibernate

4. Hibernate Configuration File

hibernate.cfg.xml in **WEB-INF**:

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
 "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
 "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
   <session-factory>
      <property
name="hibernate.connection.driver_class">com.mysql.cj.jdbc.Driver</property>
      <property
name="hibernate.connection.url">jdbc:mysql://localhost:3306/hibernate_webapp</
property>
      <property name="hibernate.connection.username">root</property>
      <property name="hibernate.connection.password">your_password</property>

      <property
name="hibernate.dialect">org.hibernate.dialect.MySQL8Dialect</property>
      <property name="hibernate.hbm2ddl.auto">update</property>
      <property name="hibernate.show_sql">true</property>

      <mapping class="com.example.model.User"/>
   </session-factory>
</hibernate-configuration>
```

5. Create User Entity Class

src/com/example/model/User.java

```
package com.example.model;

import javax.persistence.*;

@Entity
@Table(name = "users")
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    @Column(name="name")
    private String name;

    @Column(name="email")
    private String email;

    // Getters and Setters
    public int getId() { return id; }
    public void setId(int id) { this.id = id; }

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }

    public String getEmail() { return email; }
    public void setEmail(String email) { this.email = email; }
}
```

---

6. Hibernate Utility Class

src/com/example/util/HibernateUtil.java

```
package com.example.util;

import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class HibernateUtil {
    private static SessionFactory sessionFactory;
```

```
    static {
        try {
            sessionFactory = new
Configuration().configure("hibernate.cfg.xml").buildSessionFactory();
        } catch (Throwable ex) {
            System.err.println("SessionFactory creation failed: " + ex);
            throw new ExceptionInInitializerError(ex);
        }
    }

    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }
}
```

---

7. Create JSP Form (View)

Webapp/register.jsp

```
<%@ page contentType="text/html;charset=UTF-8" %>
<html>
<head><title>User Registration</title></head>
<body>
    <h2>Register User</h2>
    <form action="register" method="post">
        Name: <input type="text" name="name"/><br/>
        Email: <input type="text" name="email"/><br/>
        <input type="submit" value="Register"/>
    </form>
</body>
</html>
```

---

8. Servlet to Handle Form (Controller)

src/com/example/servlet/RegisterServlet.java

```
package com.example.servlet;

import com.example.model.User;
import com.example.util.HibernateUtil;
import org.hibernate.Session;
import org.hibernate.Transaction;
```

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.IOException;

public class RegisterServlet extends HttpServlet {
   protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {

        String name = request.getParameter("name");
        String email = request.getParameter("email");

        User user = new User();
        user.setName(name);
        user.setEmail(email);

        Session session = HibernateUtil.getSessionFactory().openSession();
        Transaction tx = session.beginTransaction();

        session.save(user);
        tx.commit();
        session.close();

        request.setAttribute("user", user);
        RequestDispatcher dispatcher = request.getRequestDispatcher("success.jsp");
        dispatcher.forward(request, response);
    }
}
```

---

9. JSP Confirmation Page

Webapp/success.jsp

```
<%@ page contentType="text/html;charset=UTF-8" %>
<html>
<head><title>Success</title></head>
<body>
   <h2>Registration Successful</h2>
   <p>Name: ${user.name}</p>
   <p>Email: ${user.email}</p>
</body>
</html>
```

---

10. Configure web.xml

Webapp/WEB-INF/web.xml

```
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
        version="3.0">
   <servlet>
      <servlet-name>RegisterServlet</servlet-name>
      <servlet-class>com.example.servlet.RegisterServlet</servlet-class>
   </servlet>

   <servlet-mapping>
      <servlet-name>RegisterServlet</servlet-name>
      <url-pattern>/register</url-pattern>
   </servlet-mapping>

   <welcome-file-list>
      <welcome-file>register.jsp</welcome-file>
   </welcome-file-list>
</web-app>
```

Run Your Application

1. Right-click the project → **Run As → Run on Server**
2. Fill the registration form
3. Submit and see the data saved in the DB

✅Hibernate will automatically create the **users** table and insert the record.

10. b) Create an application to demonstrate Hibernate Query Language.

Project Structure

```
HibernateHQLWebApp/
├── src/
│   ├── com.example.model/          --> User.java
│   ├── com.example.util/           --> HibernateUtil.java
│   └── com.example.servlet/
│       ├── RegisterServlet.java
│       └── ViewUsersServlet.java
├── WebContent/
│   ├── register.jsp
│   ├── viewUsers.jsp
│   └── WEB-INF/
│       ├── web.xml
│       ├── lib/ (Hibernate & MySQL JARs)
│       └── hibernate.cfg.xml
```

---

1. Create a Dynamic Web Project in Eclipse

1. Go to: **File → New → Dynamic Web Project**
2. Name: **HibernateHQLWebApp**
3. Choose Tomcat server runtime
4. Finish

---

2. Add Required JARs to WEB-INF/lib

Add the following to **Webapp/WEB-INF/lib**:

- **hibernate-core-x.x.x.jar**
- **hibernate-common-annotations.jar**
- **hibernate-jpa-x.x-api.jar**
- **dom4j.jar**, **antlr.jar**, **jboss-logging.jar**
- **mysql-connector-java-x.x.xx.jar**

Right-click **lib** → Build Path → Add to Build Path

---

3. MySQL Setup

CREATE **DATABASE hql_demo;**

USE hql_demo;

No need to manually create the table — Hibernate will do it.

---

4. hibernate.cfg.xml (Hibernate Configuration)

Place in **WEB-INF/**:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
 "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
 "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
   <session-factory>
      <property
name="hibernate.connection.driver_class">com.mysql.cj.jdbc.Driver</property>
      <property
name="hibernate.connection.url">jdbc:mysql://localhost:3306/hql_demo</property>
      <property name="hibernate.connection.username">root</property>
      <property name="hibernate.connection.password">your_password</property>

      <property
name="hibernate.dialect">org.hibernate.dialect.MySQL8Dialect</property>
      <property name="hibernate.show_sql">true</property>
      <property name="hibernate.hbm2ddl.auto">update</property>

      <mapping class="com.example.model.User"/>
   </session-factory>
</hibernate-configuration>
```

---

5. User.java (Model Class)

src/com/example/model/User.java

```
package com.example.model;

import javax.persistence.*;

@Entity
```

```
@Table(name="users")
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    @Column
    private String name;

    @Column
    private String email;

    // Getters & Setters
    public int getId() { return id; }
    public void setId(int id) { this.id = id; }

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }

    public String getEmail() { return email; }
    public void setEmail(String email) { this.email = email; }
}
```

---

6. HibernateUtil.java

src/com/example/util/HibernateUtil.java

```
package com.example.util;

import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class HibernateUtil {
    private static SessionFactory factory;

    static {
        try {
            factory = new
Configuration().configure("hibernate.cfg.xml").buildSessionFactory();
        } catch (Throwable ex) {
            System.err.println("SessionFactory failed: " + ex);
            throw new ExceptionInInitializerError(ex);
        }
    }
```

```
    public static SessionFactory getSessionFactory() {
        return factory;
    }
}
```

---

7. register.jsp (UI to Register User)

Webapp/register.jsp

```
<html>
<head><title>Register</title></head>
<body>
    <h2>User Registration</h2>
    <form action="register" method="post">
        Name: <input type="text" name="name" /><br/>
        Email: <input type="text" name="email" /><br/>
        <input type="submit" value="Register"/>
    </form>
    <br/>
    <a href="view">View All Users</a>
</body>
</html>
```

---

8. RegisterServlet.java (Insert into DB)

src/com/example/servlet/RegisterServlet.java

```
package com.example.servlet;

import com.example.model.User;
import com.example.util.HibernateUtil;
import org.hibernate.*;

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.IOException;

public class RegisterServlet extends HttpServlet {
    protected void doPost(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException {
        String name = req.getParameter("name");
        String email = req.getParameter("email");
```

```
        User user = new User();
        user.setName(name);
        user.setEmail(email);

        Session session = HibernateUtil.getSessionFactory().openSession();
        Transaction tx = session.beginTransaction();

        session.save(user);
        tx.commit();
        session.close();

        res.sendRedirect("register.jsp");
    }
}
```

---

9. ViewUsersServlet.java (HQL Demo)

src/com/example/servlet/ViewUsersServlet.java

```
package com.example.servlet;

import com.example.model.User;
import com.example.util.HibernateUtil;
import org.hibernate.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.IOException;
import java.util.List;

public class ViewUsersServlet extends HttpServlet {
    protected void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException {
        Session session = HibernateUtil.getSessionFactory().openSession();
        // HQL Query
        String hql = "FROM User";
        Query query = session.createQuery(hql);
        List<User> userList = query.list();
        req.setAttribute("users", userList);
        RequestDispatcher dispatcher = req.getRequestDispatcher("viewUsers.jsp");
        dispatcher.forward(req, res);
        session.close();
    }
}
```

10. viewUsers.jsp (Show Results)

Webapp/viewUsers.jsp

```
<%@ page import="java.util.List" %>
<%@ page import="com.example.model.User" %>
<%
   List<User> users = (List<User>) request.getAttribute("users");
%>
<html>
<head><title>User List</title></head>
<body>
   <h2>All Registered Users</h2>
   <table border="1">
      <tr><th>ID</th><th>Name</th><th>Email</th></tr>
      <%
         for (User user : users) {
      %>
      <tr>
         <td><%= user.getId() %></td>
         <td><%= user.getName() %></td>
         <td><%= user.getEmail() %></td>
      </tr>
      <%
         }
      %>
   </table>
   <br/><a href="register.jsp">Back to Registration</a>
</body>
</html>
```

---

11. web.xml (Servlet Mapping)

Webapp/WEB-INF/web.xml

```
<web-app xmlns="http://java.sun.com/xml/ns/javaee" version="3.0">
   <servlet>
      <servlet-name>RegisterServlet</servlet-name>
      <servlet-class>com.example.servlet.RegisterServlet</servlet-class>
   </servlet>
   <servlet-mapping>
      <servlet-name>RegisterServlet</servlet-name>
      <url-pattern>/register</url-pattern>
   </servlet-mapping>
```

```
    <servlet>
        <servlet-name>ViewUsersServlet</servlet-name>
        <servlet-class>com.example.servlet.ViewUsersServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>ViewUsersServlet</servlet-name>
        <url-pattern>/view</url-pattern>
    </servlet-mapping>

    <welcome-file-list>
        <welcome-file>register.jsp</welcome-file>
    </welcome-file-list>
</web-app>
```

Running the App

1. Run the project on Apache Tomcat (**Run As → Run on Server**)
2. Register a few users
3. Click "View All Users" to see HQL in action (**FROM User** query)