

Class 2: Building a Responsive Signup/Signin Page

Session Overview

By the end of this session, students will be able to:

- Understand how to create HTML forms using POST and GET methods.
- Master CSS techniques for responsive design using Media Queries and Container Queries.
- Apply these concepts to develop an E-commerce Signup/Signin Pag/otp registration and enhance Landing Page responsiveness.

Total Session Duration: 1.5 - 2 hours

Layout of the User Registration Pages:

Let's break down the implementation of a restaurant menu app User Registration pages :

1. Login page
2. Sign up page
3. OTP Verification Page

Login page:

Let's Style and structure the Login page

```
<!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>
</head>
<body>
  <header>
    <nav class="navbar">
      <div class="logo">
        <a href="index.html"> </a>
</div>
</nav>
</header>
<div class="form-container">

    <form class="login-form">
        <h2>Welcome Back!</h2>
        <div class="form-group">
            <label for="username"><b>Mobile Number : </b></label>
            <input type="number" id="mobile-number"
name="mobile-number" required placeholder="Enter your mobile number">
        </div>

        <button type="submit"
onclick="redirectToOTPRegister()">Login</button>
        <p>New to our restaurant? <a href="signup.html">Sign
up</a></p>

        <script>
            function redirectToOTPRegister() {
                window.location.href = "otpregister.html";
            }
        </script>
    </form>

</div>

</body>
</html>
```



Explanation:

- HTML Structure Setup:**
 It defines a basic HTML document with a `DOCTYPE` declaration and metadata in the `<head>` section (like character encoding and viewport settings for responsiveness).
- Header and Navigation:**
 Inside the `<header>`, there is a navigation bar (`<nav>`), which contains a logo linking to the homepage (`index.html`).
- Login Form:**
 The login form is placed inside a `<div class="form-container">`:
 - It asks for the **mobile number** as input.
 - It includes a **button** to log in, which triggers a `redirectToOTPRegister()` function, redirecting the user to an **OTP registration page** (`otpregister.html`).
- New User Prompt:**
 If the user doesn't have an account, it provides a link to a **sign-up page** (`signup.html`).
- JavaScript Function:**
 The function `redirectToOTPRegister()` redirects the user to the OTP registration page upon clicking the **Login button**.

Sign Up Page

Let's structure the sign up page

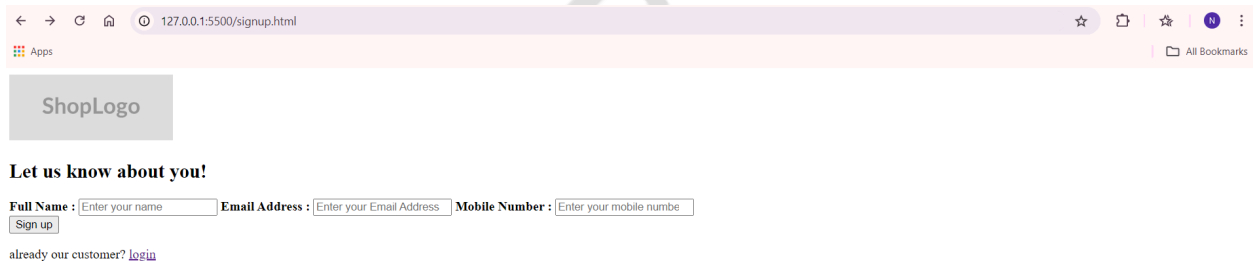
```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Login Page</title>

  <link rel="stylesheet" href="user-registration.css" >
  <link rel="stylesheet" href="styles.css"
</head>
<body>
  <header>
    <nav class="navbar">
      <div class="logo">
        <a href="index.html"> </a>
      </div>
    </nav>
  </header>
  <div class="form-container">

    <form class="login-form">
      <h2>Let us know about you!</h2>
      <div class="form-group">
        <label for="username"><b> Full Name : </b></label>
        <input type="text" id="username" name="username" required
placeholder="Enter your name">
        <label for="username"><b>Email Address : </b></label>
        <input type="email" id="email-address" name="email" required
placeholder="Enter your Email Address">
        <label for="username"><b>Mobile Number : </b></label>
        <input type="number" id="mobile-number" name="mobile-number"
required placeholder="Enter your mobile number">
      </div>
```

```
<button type="submit" onclick="redirectToOTPRegister()">Sign  
up</button>  
  
<p>already our customer? <a href="login.html">login</a></p>  
<script>  
    function redirectToOTPRegister() {  
        window.location.href = "otpregister.html";  
    }  
</script>  
</form>  
  
</div>  
  
</body>  
</html>
```



Explanation

HTML Structure and Meta Information:

- The document uses `<!DOCTYPE html>` to declare it as an HTML5 page.

- The `<meta>` tags ensure proper character encoding (UTF-8) and make the layout responsive on mobile devices (viewport settings).
- Two external CSS stylesheets (user-registration.css and styles.css) are linked to apply styling.

Header with Navigation Bar:

- A `<header>` contains a navigation bar with a logo that links back to the homepage (index.html).

Sign-up Form:

- Inside the `<div class="form-container">`, there's a form prompting users to provide:
 - Full Name
 - Email Address
 - Mobile Number
- These are essential details required for registration.

Sign-up Button and Redirection:

- The Sign-up button triggers a JavaScript function `redirectToOTPRegister()`.
- This function redirects the user to the OTP verification page (otpreregister.html) when clicked.

Link to Login Page:

- A message provides a link to the login page (login.html) if the user already has an account.

OTP Verification page

```
<!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>

  <!-- <link rel="stylesheet" href="user-registration.css" >
  <link rel="stylesheet" href="styles.css" -->
</head>
<body>
```

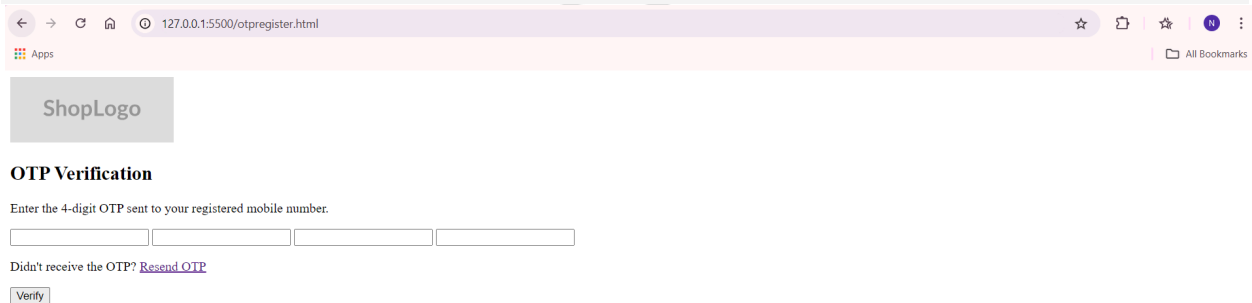
```
<header>
  <nav class="navbar">
    <div class="logo">
      <a href="index.html"> </a>
    </div>
  </nav>
</header>
<div class="form-container">
  <form class="otp-form">
    <h2>OTP Verification</h2>
    <p class="otp-instruction">Enter the 4-digit OTP sent to your registered
mobile number.</p>

    <div class="otp-inputs">
      <input type="text" class="otp-input" pattern="[0-9]*" inputmode="numeric"
maxlength="1" required>
      <input type="text" class="otp-input" pattern="[0-9]*" inputmode="numeric"
maxlength="1" required>
      <input type="text" class="otp-input" pattern="[0-9]*" inputmode="numeric"
maxlength="1" required>
      <input type="text" class="otp-input" pattern="[0-9]*" inputmode="numeric"
maxlength="1" required>
    </div>

    <p class="resend-info">
      Didn't receive the OTP?
      <a href="#" class="resend-link">Resend OTP</a>
    </p>

    <button type="submit">Verify</button>
  </form>
</div>
<script>
  const otpInputs = document.querySelectorAll('.otp-input');
```

```
otpInputs.forEach((input, index) => {  
  input.addEventListener('input', () => {  
    if (input.value && index < otpInputs.length - 1) {  
      otpInputs[index + 1].focus(); // Move to the next input  
    }  
  });  
  
  input.addEventListener('keydown', (e) => {  
    if (e.key === 'Backspace' && !input.value && index > 0) {  
      otpInputs[index - 1].focus(); // Move to the previous input  
    }  
  });  
});  
  
</script>  
  
</body>  
</html>
```



ShopLogo

OTP Verification

Enter the 4-digit OTP sent to your registered mobile number.

Didn't receive the OTP? [Resend OTP](#)

Explanation:

HTML Structure and Meta Tags:

- Declares the page as an HTML5 document and sets the character encoding to UTF-8.
- Makes the layout responsive with the viewport meta tag.

Header with Logo:

- Includes a navigation bar (`<nav>`) containing a logo that links to the homepage ([index.html](#)).

OTP Verification Form:

- The form contains:
 - A heading ("OTP Verification").
 - An instruction message explaining that the user should enter the 4-digit OTP sent to their registered mobile number.
 - Four input fields, each allowing only a single digit (using `maxlength="1"`, `pattern="[0-9]*"`, and `inputmode="numeric"` to restrict input to numbers).
 - A resend OTP link for users to request a new OTP.
 - A Verify button to submit the form.

JavaScript for OTP Input Behavior:

- Automatically focuses on the next input field when the user types a digit.
- Moves focus back to the previous input if the user presses Backspace on an empty field.

```
<script>
  const otpInputs = document.querySelectorAll('.otp-input');

  otpInputs.forEach((input, index) => {
    input.addEventListener('input', () => {
      if (input.value && index < otpInputs.length - 1) {
        otpInputs[index + 1].focus(); // Move to the next input
      }
    });

    input.addEventListener('keydown', (e) => {
      if (e.key === 'Backspace' && !input.value && index > 0) {
        otpInputs[index - 1].focus(); // Move to the previous input
      }
    });
  });
</script>
```

User Experience Enhancements:

- Smooth input switching ensures users can easily type the OTP without needing to manually click on each field.
- The resend link provides an option to handle cases where the OTP isn't received.

Create a new CSS file named **user-registration.css** to style all registration pages, including the sign-in, sign-up, and OTP registration files, in one single file.

User-registration.css

```
*{
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  font-family: 'Arial', sans-serif;
}

/* Full height container to center content */
body {
  height: 100vh;
  display: flex;
  flex-direction: column;
}

.form-container {
  height: calc(100vh - 80px); /* Subtract the height of navbar */
  display: flex;
  justify-content: center;
  align-items: center;
}

/* Login form styling */
```

```
.login-form {  
  background-color: #ffffff;  
  width: 100%;  
  max-width: 400px;  
  padding: 40px;  
  border-radius: 12px;  
  box-shadow: 0 8px 15px rgba(0, 0, 0, 0.1);  
  text-align: center;  
}
```

```
h2 {  
  margin-bottom: 20px;  
  color: #2d3436;  
  font-size: 28px;  
  font-weight: 600;  
}
```

```
.form-group {  
  margin-bottom: 15px;  
  text-align: left;  
}
```

```
label {  
  display: block;  
  margin-bottom: 5px;  
  font-weight: 500;  
  color: #636e72;  
}
```

```
input {  
  width: 100%;  
  padding: 10px;  
  border: 1px solid #dfe6e9;  
  border-radius: 8px;  
  outline: none;  
  font-size: 16px;  
  margin-bottom: 8px;  
  transition: 0.3s ease-in-out;  
}
```

```
.otp-form {
  background-color: #fff;
  padding: 20px;
  border-radius: 8px;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
  text-align: center;
}

.otp-inputs {
  display: flex;
  gap: 10px;
  justify-content: center;
  margin: 15px 0;
}

.otp-input {
  width: 50px;
  height: 50px;
  text-align: center;
  font-size: 1.5rem;
  border: 1px solid #ccc;
  border-radius: 8px;
  outline: none;
}

.otp-input:focus {
  border-color: #007bff;
  box-shadow: 0 0 5px rgba(0, 123, 255, 0.5);
}

.otp-form h2 {
  margin-bottom: 10px;
  color: #333;
}

.otp-instruction {
  margin: 10px 0 20px;
  font-size: 0.9rem;
  color: #666;
}
```

```
/* Remove spinner arrows on number input */
input[type="number"]::-webkit-inner-spin-button,
input[type="number"]::-webkit-outer-spin-button {
  -webkit-appearance: none;
  margin: 0;
}

input[type="number"]:focus {
  border-color: black;
  /* box-shadow: 0 0 5px #dd4955; */
}

button {
  width: 100%;
  padding: 12px;
  background-color: #e63946;
  color: #ffffff;
  border: none;
  border-radius: 8px;
  font-size: 18px;
  cursor: pointer;
  transition: 0.3s ease;
  margin-top: 10px;
}

button:hover {
  background-color: #c12330;
}

p {
  margin-top: 20px;
  color: #2d3436;
}

a {
  color: #0984e3;
  text-decoration: none;
  font-weight: 500;
}
```

```
}

a:hover {
  text-decoration: underline;
}

/* Responsive design for smaller screens */
@media (max-width: 480px) {
  .login-form {
    padding: 30px;
  }
  h2 {
    font-size: 24px;
  }
}
```

Explanation:

Global Reset (*)

- Ensures consistent margins, padding, and fonts across elements.

Body Styling (body)

- Uses flexbox to center content vertically and horizontally, with the height set to 100vh.

Form Container (.form-container)

- Holds the form, ensuring it's centered on the screen with space adjusted for the navbar.

Login Form (.login-form)

- Styled with padding, shadows, rounded corners, and responsive adjustments.
- Headings (h2) are highlighted with color and spacing.

OTP Form (.otp-form)

- Displays OTP input fields in a row using .otp-inputs.
- Individual OTP inputs (.otp-input) have focus styles and transitions for a better UX.

Form Group (.form-group)

- Handles layout for labels and input fields, aligned vertically with proper spacing.

Button Styling (`button`)

- Styled with a background color, rounded corners, and hover effects to change color.

Link Styling (`a`)

- Links are given default colors with hover underline effects.

Responsive Design (`Media Query @media`)

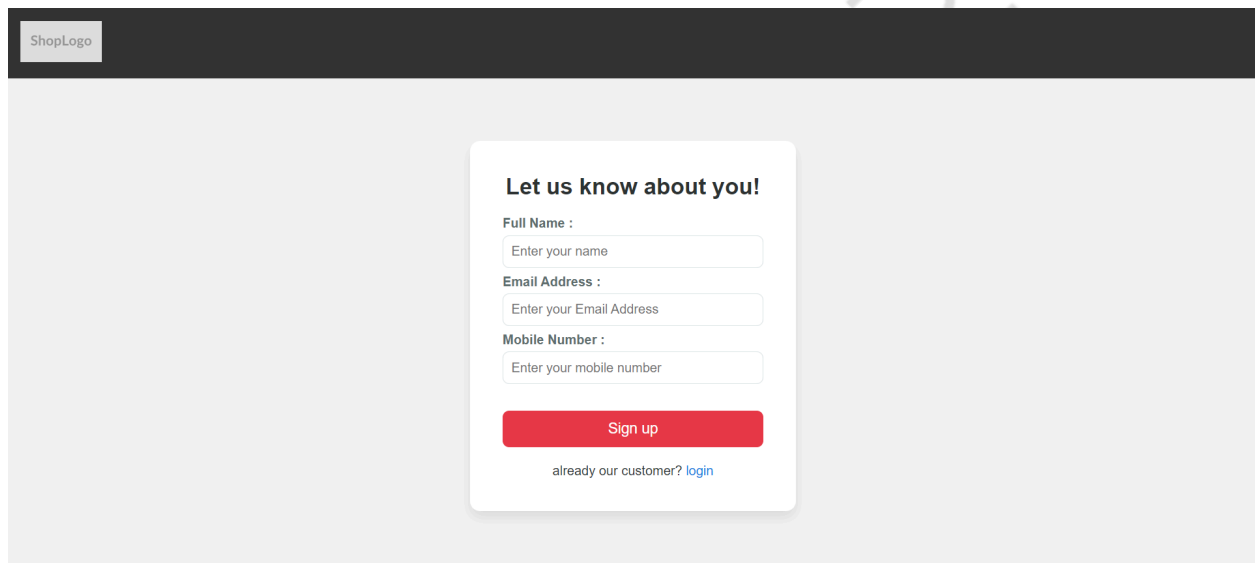
- Adjusts padding and font size in `.login-form` for screens smaller than 480px.

In the `<head>` section of all HTML files, including Sign-in, Sign-up, and OTP registration, include the following links to apply styles using external CSS files:

```
<head>
  <link rel="stylesheet" href="styles.css" />
  <link rel="stylesheet" href="user-registration.css" />
</head>
```

Outputs: After integrating the `user-registration.css` with all the html file

Sign up page



The image shows a sign-up page layout. At the top, there is a dark header bar with a 'ShopLogo' placeholder on the left. The main content area has a light gray background. In the center, there is a white card with a shadow. The card has a title 'Let us know about you!' followed by three form fields: 'Full Name :', 'Email Address :', and 'Mobile Number :'. Each field has a placeholder text 'Enter your name', 'Enter your Email Address', and 'Enter your mobile number' respectively. Below these fields is a red 'Sign up' button. At the bottom of the card, there is a link 'already our customer? login'.

OTP Verification page

ShopLogo

OTP Verification

Enter the 4-digit OTP sent to your registered mobile number.

Didn't receive the OTP? [Resend OTP](#)

Verify

Login page

ShopLogo

Welcome Back!

Mobile Number :

Enter your mobile number

Login

New to our restaurant? [Sign up](#)

Final HTML and CSS

[Login.html](#)

[Signup.html](#)

[Otpregistration.html](#)

Industry Insights and Best Practices:

For the **Registration module** of web applications, this guide focuses on practical concepts, common interview questions, and best practices for form handling, validation, security, and backend integration.

Key Interview Topics for Registration Forms

These topics are frequently discussed in technical interviews, including FAANG and other major companies:

1. Form Handling Basics

- Understanding `<form>` elements and how form submissions work using **GET** and **POST** methods.
- **Action and Method Attributes:** Proper configuration ensures correct data submission paths.

2. Client-side vs. Server-side Validation

- **Client-side Validation:** Uses HTML attributes or JavaScript for instant feedback (e.g., `required`, `pattern`).
- **Server-side Validation:** Ensures data integrity after submission by performing checks on the backend.

3. Security Concerns

- Addressing threats like **Cross-Site Request Forgery (CSRF)** and **SQL injection** with input sanitization and secure headers.

4. Integrating Backends

- Connecting forms to **RESTful APIs** and understanding **HTTP status codes**:
 - `200` (Success), `400` (Bad Request), and `422` (Unprocessable Entity).

Concept Brush-ups for Registration Forms

HTML Form Basics

Q: What is the purpose of the `<form>` tag, and what attributes are commonly used?

A: The `<form>` tag collects user input. Common attributes:

- **action:** URL where the form sends the data.
- **method:** HTTP method to use (GET or POST).
- **autocomplete:** Controls browser autofill behavior.

Example:

```
<form action="/register" method="POST" autocomplete="on">
  <input type="email" name="userEmail" required />
</form>
```

Q: How can you make input fields mandatory?

A: Use the `required` attribute:

```
<input type="text" name="username" required />
```

Validation Techniques

Q: What is the difference between client-side and server-side validation?

A:

- **Client-side validation:** Runs in the browser for instant feedback (e.g., `required`, `pattern` attributes).
- **Server-side validation:** Ensures data integrity after submission, preventing malicious inputs from bypassing client-side checks.

Q: How do you implement regex validation for an email field?

A: Example:

```
<input type="email" name="userEmail"
pattern="[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,}$" required />
```

This pattern ensures the input follows the standard email format.

Advanced Interview Questions: Security in Forms

Q: What are CSRF attacks, and how can they be prevented?

A:

- **CSRF (Cross-Site Request Forgery)** tricks users into performing unintended actions.
- **Prevention Strategies:**
 - Use **CSRF tokens** in forms.
 - Validate **Origin** and **Referrer** headers in requests.

Q: How do you sanitize inputs to prevent SQL injection?

A:

- Use **parameterized queries** or ORMs like **Mongoose** in Node.js to prevent malicious SQL commands.
 - Avoid directly concatenating user inputs into SQL queries.
-

CSS Basics for Forms

Responsive Form Design

Q: How can you ensure forms are responsive across devices?

A: Use **media queries** or frameworks like [Bootstrap](#).

Example:

```
@media (max-width: 768px) {

  form {

    width: 100%; // Apply any properties of the CSS to make the form
    responsive

  }
}
```

}

Enhancing User Experience with Pseudo-Classes

Q: How do pseudo-classes like `:focus` and `:hover` improve user experience?

A:

- `:focus` helps highlight the active input field for better UX.

Example:

```
input:focus {  
    border-color: #4caf50;  
}
```

- `:hover` can improve visual feedback on interactive elements like buttons.
-

Best Practices and Advanced Tips

1. Master Client-side and Server-side Validation

- Use **regex patterns** to validate emails and phone numbers.
- Implement **server-side checks** even if client-side validation is present for security.

2. Use CSS Frameworks and DevTools for Form Styling

- Use **CSS Grid** or **Flexbox** to align form elements.
 - Use **Browser DevTools** to inspect and modify CSS on the fly for quick styling fixes.
-

Stay Current with Industry Trends

- Follow resources like [CSS-Tricks](#), [MDN](#), and [Dev.to](#) to stay updated on modern form-handling techniques.
- Keep an eye on **HTML5** and **CSS3** features such as new input types and CSS variables.