WEB DESIGN PRACTICAL

3. DESIGN AND DEVELOP AN EVENT REGISTER FORM

Event\_html\_reload.txt

document.getElementById('registrationForm').addEventListener('submit', function(event) {

event.preventDefault(); // Prevent form from reloading

// Get form data

const name = document.getElementById('name').value;

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

const phone = document.getElementById('phone').value;

const eventType = document.getElementById('eventType').value;

const attendees = document.getElementById('attendees').value;

const date = document.getElementById('date').value;

const requirements = document.getElementById('requirements').value;

// Optional: Validate form

if (!name || !email || !phone || !eventType || !attendees || !date) {

alert('Please fill in all required fields.');

return; }

// Process submission here (e.g., send data to the server)

// Redirect to another page (e.g., a thank you page)

window.location.href = "thankyou.html";

});

Index.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Event Registration Form</title>

    <link rel="stylesheet" href="style.css">

</head>

<body>

    <div class="container">

        <h1>Tech Innovators Conference 2024</h1>

        <form id="registrationForm">

            <div class="form-group">

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

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

            </div>

            <div class="form-group">

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

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

            </div>

            <div class="form-group">

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

                <input type="tel" id="phone" name="phone" required>

            </div>

            <div class="form-group">

                <label for="eventType">Event Type</label>

                <select id="eventType" name="eventType" required>

                    <option value="">Select Event Type</option>

                    <option value="workshop">Workshop</option>

                    <option value="seminar">Seminar</option>

                    <option value="conference">Conference</option>

                </select>

            </div>

            <div class="form-group">

                <label for="attendees">Number of Attendees</label>

                <input type="number" id="attendees" name="attendees" min="1" required>

            </div>

            <div class="form-group">

                <label for="date">Event Date</label>

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

            </div>

            <div class="form-group">

                <label for="requirements">Special Requirements</label>

                <textarea id="requirements" name="requirements" rows="4"></textarea>

            </div>

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

        </form>

    </div>

    <script src="script.js"></script>

</body>

</html>

Script.js

document.getElementById('registrationForm').addEventListener('submit', function(event) {

    // No preventDefault(), so form will reload/submit normally

    // Get form data

    const name = document.getElementById('name').value;

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

    const phone = document.getElementById('phone').value;

    const eventType = document.getElementById('eventType').value;

    const attendees = document.getElementById('attendees').value;

    const date = document.getElementById('date').value;

    const requirements = document.getElementById('requirements').value;

    // Optional: Validate form (if needed)

    if (!name || !email || !phone || !eventType || !attendees || !date) {

        alert('Please fill in all required fields.');

        return;

    }

    // Optional: Display submission result

    alert(`Thank you for registering, ${name}!`);

});

Style.css

\* {

    box-sizing: border-box;

    margin: 0;

    padding: 0;

}

body {

    font-family: Arial, sans-serif;

    background-color: #f4f4f4;

    display: flex;

    justify-content: center;

    align-items: center;

    height: 100vh;

}

.container {

    background-color: #fff;

    padding: 20px;

    border-radius: 10px;

    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

    width: 400px;

}

h1 {

    text-align: center;

    margin-bottom: 20px;

    color: #333;

}

.form-group {

    margin-bottom: 15px;

}

label {

    display: block;

    margin-bottom: 5px;

    font-weight: bold;

}

input, select, textarea {

    width: 100%;

    padding: 10px;

    border: 1px solid #ccc;

    border-radius: 5px;

}

button {

    width: 100%;

    padding: 10px;

    background-color: #28a745;

    color: white;

    border: none;

    border-radius: 5px;

    font-size: 16px;

    cursor: pointer;

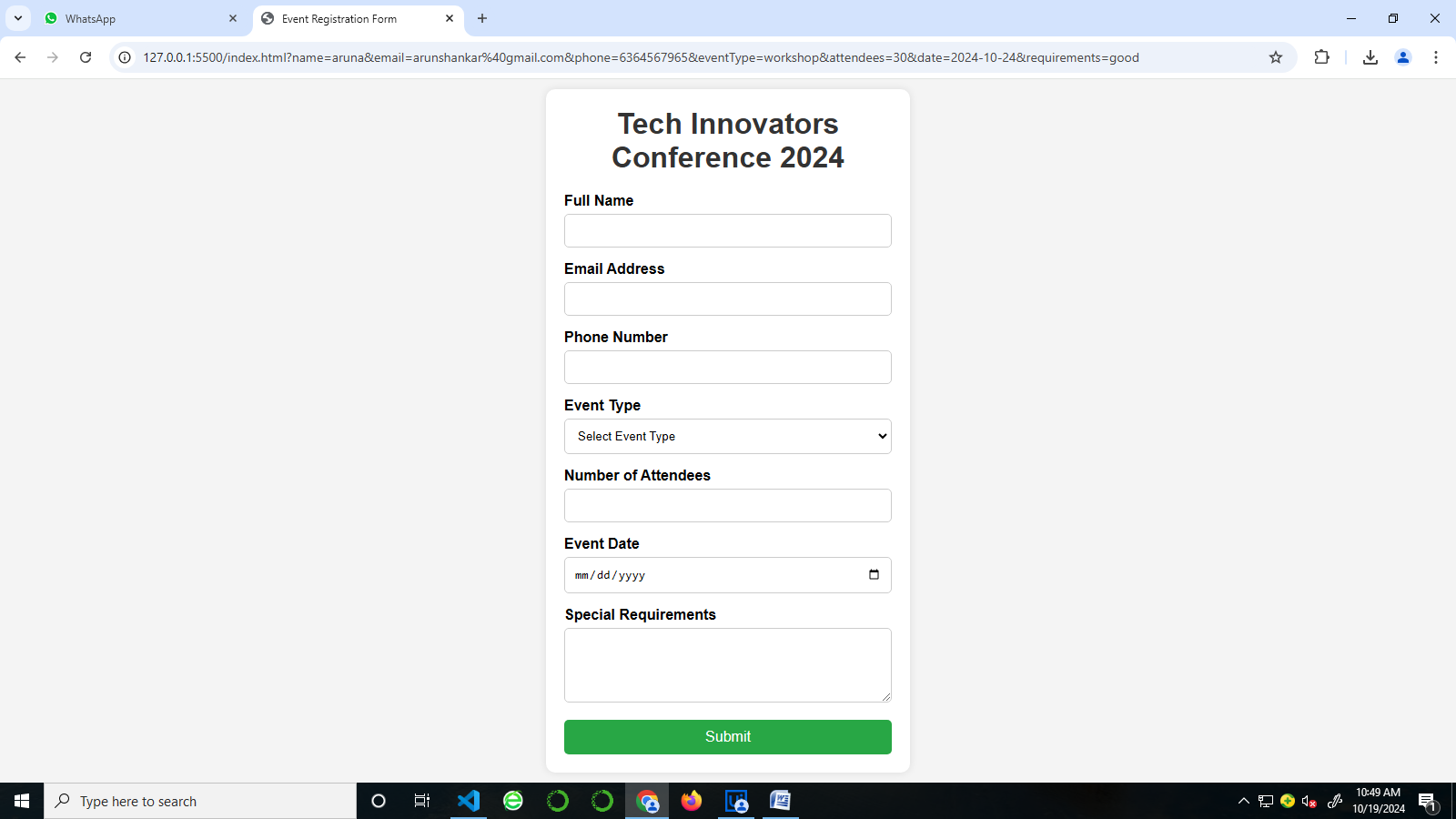
}

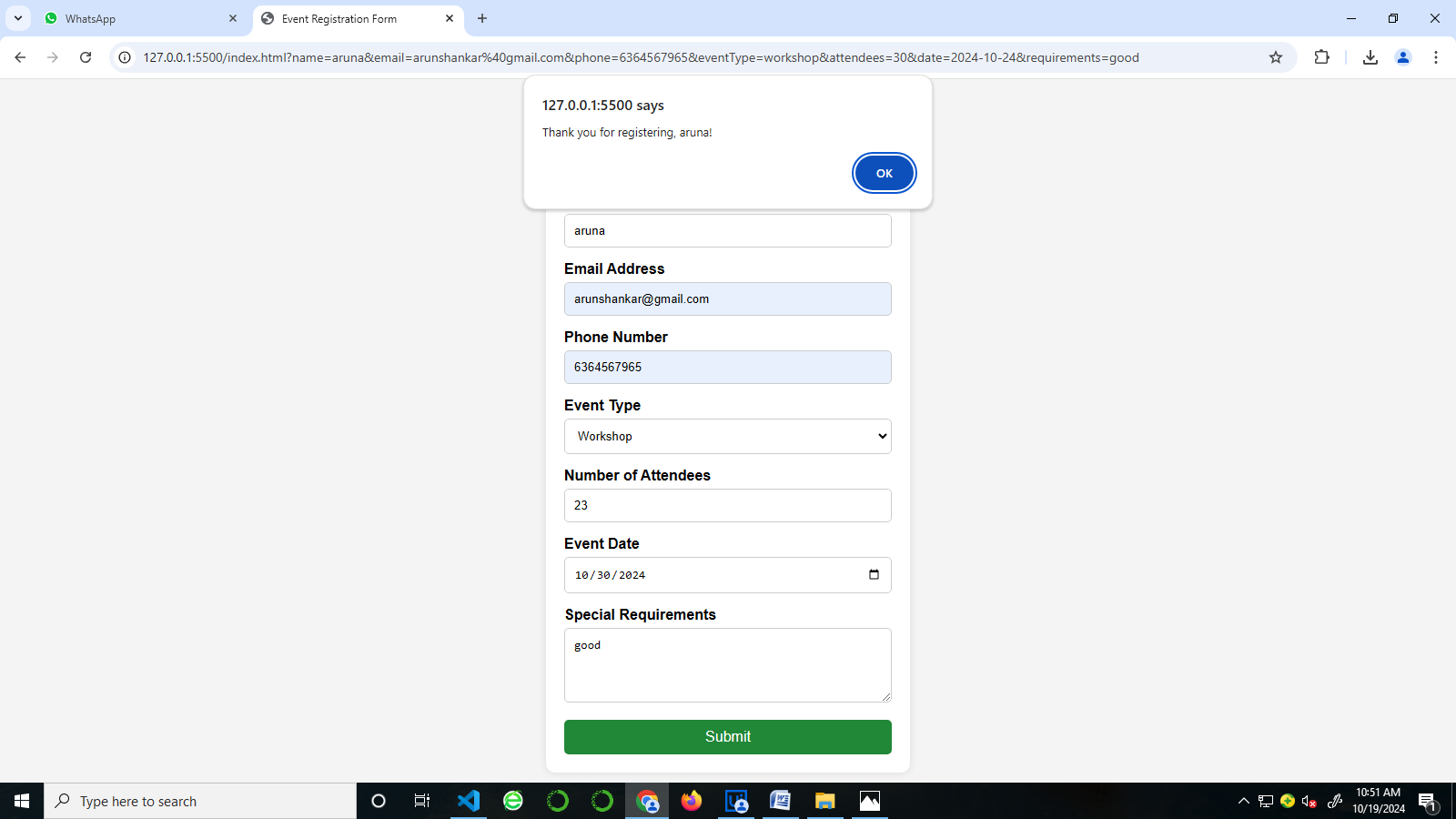
button:hover {

    background-color: #218838;

}

Output:





5.DESIGN AND DEVELOP A DEVELOPER PORFOLIC PAGE.DEVELOP THE LAYOUT USING FLEXBOX AND ENSURE THE PAGE IS RESPONSIVE

Index.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Developer Portfolio</title>

    <link rel="stylesheet" href="styles.css">

</head>

<body>

    <!-- Header Section -->

    <header>

        <nav class="navbar">

            <h1 class="logo">Mugesh M</h1>

            <ul class="nav-links">

                <li><a href="#about">About</a></li>

                <li><a href="#skills">Skills</a></li>

                <li><a href="#projects">Projects</a></li>

                <li><a href="#contact">Contact</a></li>

            </ul>

        </nav>

    </header>

    <!-- Hero Section -->

    <section class="hero">

        <div class="hero-content">

            <h2>Hello, I'm Mugesh M</h2>

            <p>Software Developer | Web Enthusiast | Tech Innovator</p>

            <a href="#projects" class="btn">View My Work</a>

        </div>

    </section>

    <!-- Skills Section -->

    <section id="skills" class="skills">

        <h2>Skills & Expertise</h2>

        <div class="skills-list">

            <div class="skill-item">HTML5</div>

            <div class="skill-item">CSS3</div>

            <div class="skill-item">JavaScript</div>

            <div class="skill-item">Python</div>

            <div class="skill-item">MySQL</div>

            <div class="skill-item">React</div>

        </div>

    </section>

    <!-- Projects Section -->

    <section id="projects" class="projects">

        <h2>Projects</h2>

        <div class="projects-list">

            <div class="project-item">

                <h3>Project 1: WhyTap Audit Tool</h3>

                <p>A web-based SEO audit tool built using Flask, MySQL, and web scraping techniques.</p>

            </div>

            <div class="project-item">

                <h3>Project 2: Diabetic Retinopathy Classification</h3>

                <p>A CNN-based application to predict the likelihood of diabetic retinopathy using retina images.</p>

            </div>

            <!-- Add more project items as needed -->

        </div>

    </section>

    <!-- Contact Section -->

    <section id="contact" class="contact">

        <h2>Contact Me</h2>

        <form class="contact-form">

            <input type="text" placeholder="Your Name" required>

            <input type="email" placeholder="Your Email" required>

            <textarea placeholder="Your Message" required></textarea>

            <button type="submit" class="btn">Send Message</button>

        </form>

    </section>

    <!-- Footer Section -->

    <footer>

        <p>&copy; 2024 Mugesh M | All Rights Reserved</p>

    </footer>

</body>

</html>

Style.css

\* {

    margin: 0;

    padding: 0;

    box-sizing: border-box;

}

body {

    font-family: 'Arial', sans-serif;

    line-height: 1.6;

}

header {

    background-color: #333;

    color: #fff;

    padding: 10px 20px;

}

.navbar {

    display: flex;

    justify-content: space-between;

    align-items: center;

}

.navbar ul {

    list-style: none;

    display: flex;

}

.navbar ul li {

    margin-left: 20px;

}

.navbar ul li a {

    color: #fff;

    text-decoration: none;

}

.hero {

    background-color: #f4f4f4;

    height: 100vh;

    display: flex;

    justify-content: center;

    align-items: center;

    text-align: center;

}

.hero-content h2 {

    font-size: 3rem;

}

.hero-content p {

    font-size: 1.2rem;

    margin: 10px 0;

}

.btn {

    display: inline-block;

    padding: 10px 20px;

    background-color: #333;

    color: #fff;

    text-decoration: none;

    margin-top: 20px;

    border-radius: 5px;

}

.skills {

    padding: 50px 20px;

    background-color: #eaeaea;

    text-align: center;

}

.skills h2 {

    margin-bottom: 20px;

}

.skills-list {

    display: flex;

    justify-content: center;

    flex-wrap: wrap;

}

.skill-item {

    background-color: #333;

    color: #fff;

    padding: 10px 20px;

    margin: 10px;

    border-radius: 5px;

}

.projects {

    padding: 50px 20px;

}

.projects h2 {

    text-align: center;

    margin-bottom: 20px;

}

.projects-list {

    display: flex;

    justify-content: space-around;

    flex-wrap: wrap;

}

.project-item {

    background-color: #f4f4f4;

    padding: 20px;

    margin: 10px;

    border-radius: 5px;

    width: 300px;

}

.contact {

    padding: 50px 20px;

    background-color: #eaeaea;

    text-align: center;

}

.contact h2 {

    margin-bottom: 20px;

}

.contact-form {

    display: flex;

    flex-direction: column;

    align-items: center;

}

.contact-form input,

.contact-form textarea {

    width: 100%;

    max-width: 400px;

    padding: 10px;

    margin: 10px 0;

    border: 1px solid #ccc;

    border-radius: 5px;

}

.contact-form button {

    background-color: #333;

    color: #fff;

    border: none;

    padding: 10px 20px;

    border-radius: 5px;

}

footer {

    text-align: center;

    background-color: #333;

    color: #fff;

    padding: 20px 0;

}

@media (max-width: 768px) {

    .navbar ul {

        flex-direction: column;

        align-items: flex-start;

    }

    .projects-list {

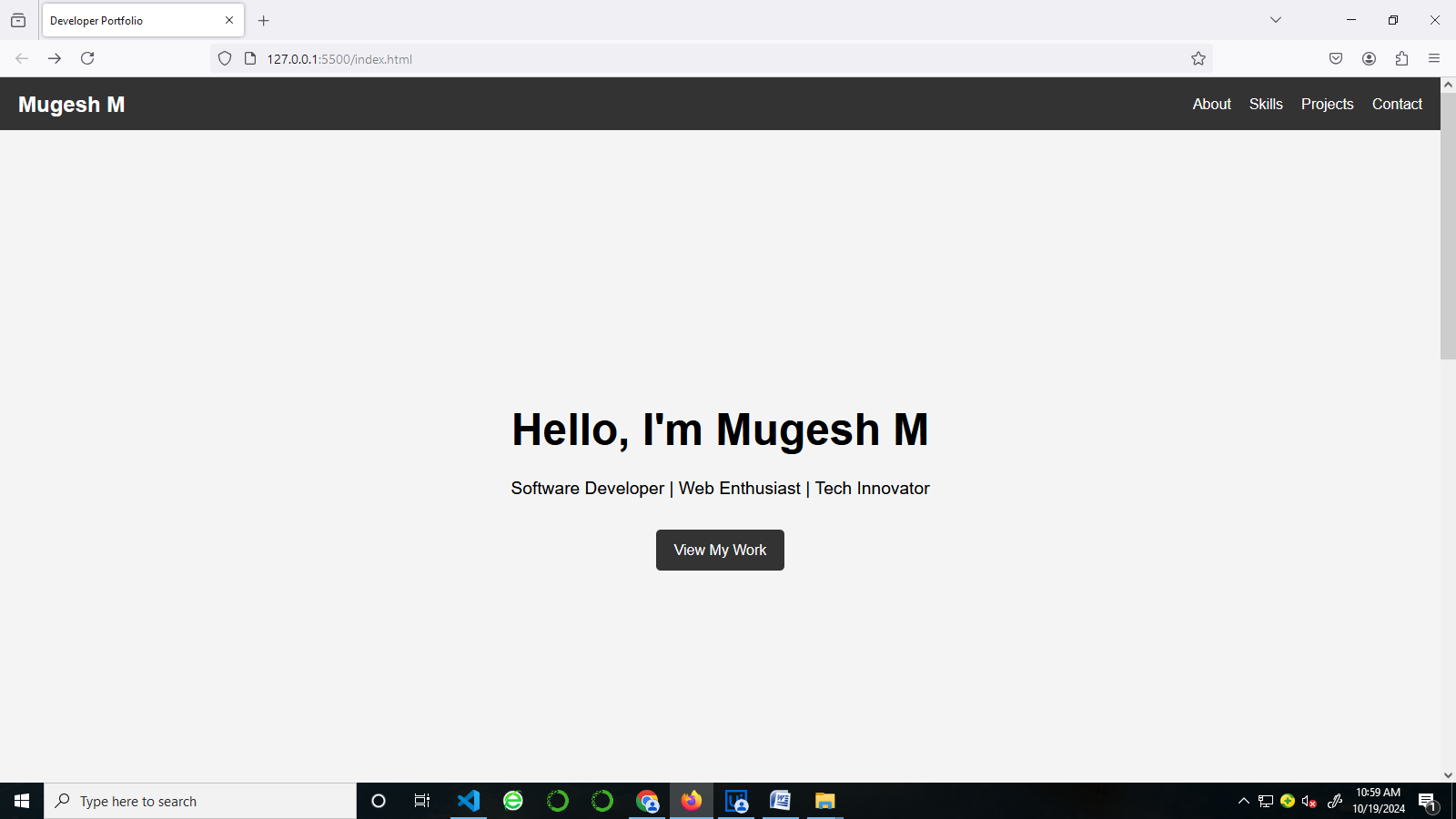
        flex-direction: column;

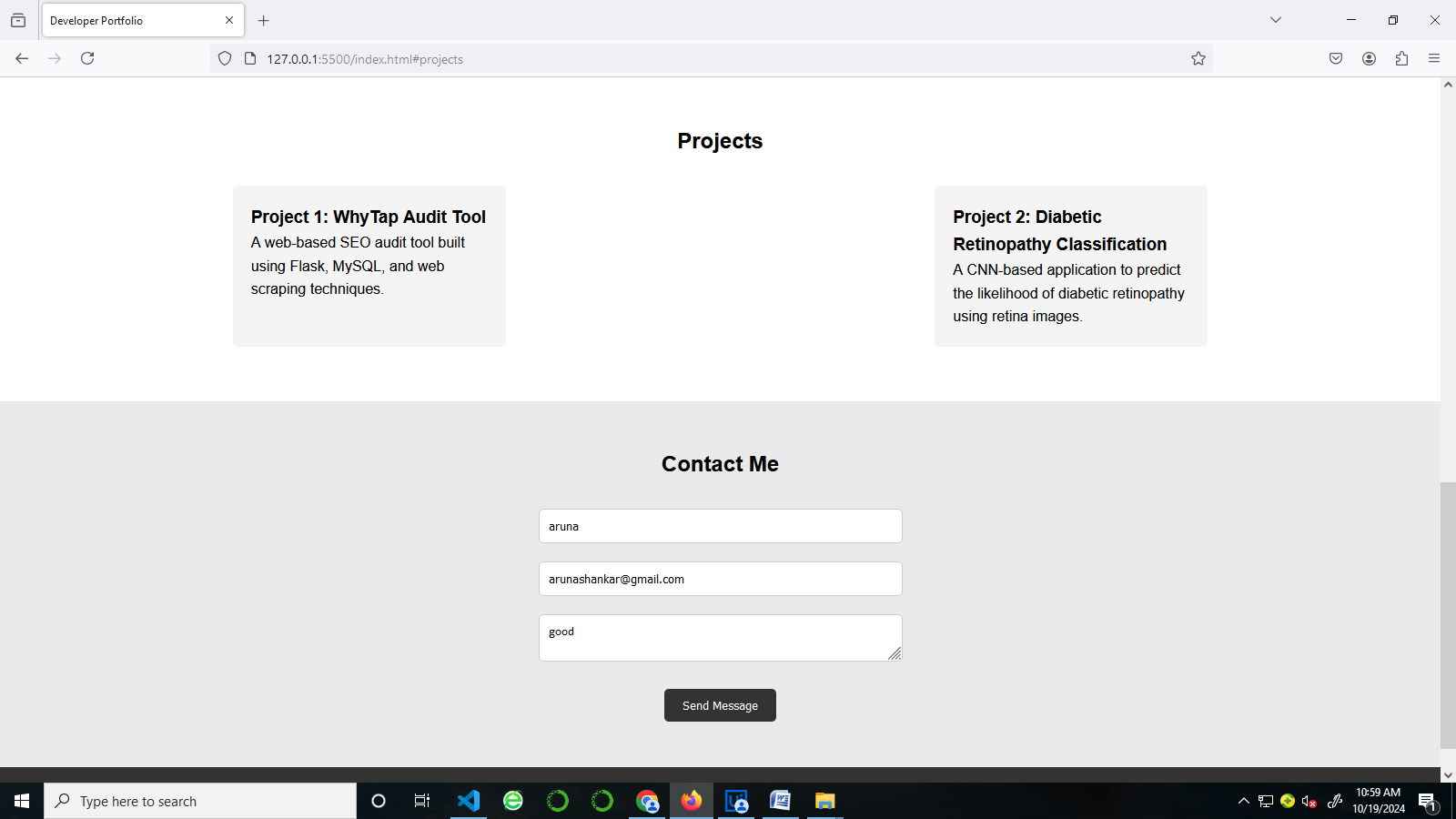
        align-items: center;

    }

}

Output:





6.DESIGN AND DEVELOP PRICING CARD LIST WHICH ARE RESPONSIVE PLAIN CSS AND FLEXBOX

Index.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Pricing Cards</title>

    <link rel="stylesheet" href="styles.css">

</head>

<body>

    <!-- Pricing Section -->

    <section class="pricing">

        <h2>Our Pricing Plans</h2>

        <div class="card-list">

            <!-- Pricing Card 1: Basic Plan -->

            <div class="card">

                <h3 class="card-title">Basic</h3>

                <p class="card-price">$19/month</p>

                <ul class="card-features">

                    <li>10 Projects</li>

                    <li>5 GB Storage</li>

                    <li>Email Support</li>

                </ul>

                <button class="btn">Get Started</button>

            </div>

            <!-- Pricing Card 2: Pro Plan -->

            <div class="card">

                <h3 class="card-title">Pro</h3>

                <p class="card-price">$49/month</p>

                <ul class="card-features">

                    <li>50 Projects</li>

                    <li>50 GB Storage</li>

                    <li>Priority Support</li>

                </ul>

                <button class="btn">Get Started</button>

            </div>

            <!-- Pricing Card 3: Enterprise Plan -->

            <div class="card">

                <h3 class="card-title">Enterprise</h3>

                <p class="card-price">$99/month</p>

                <ul class="card-features">

                    <li>Unlimited Projects</li>

                    <li>200 GB Storage</li>

                    <li>24/7 Support</li>

                </ul>

                <button class="btn">Get Started</button>

            </div>

        </div>

    </section>

</body>

</html>

Style.css

\* {

    margin: 0;

    padding: 0;

    box-sizing: border-box;

}

body {

    font-family: 'Arial', sans-serif;

    line-height: 1.6;

    background-color: #f4f4f4;

}

h2 {

    text-align: center;

    margin: 20px 0;

    font-size: 2rem;

}

/\* Pricing Section \*/

.pricing {

    padding: 40px 20px;

}

.card-list {

    display: flex;

    justify-content: space-around;

    flex-wrap: wrap;

}

.card {

    background-color: #fff;

    border-radius: 10px;

    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);

    padding: 20px;

    width: 300px;

    margin: 20px;

    text-align: center;

}

.card-title {

    font-size: 1.5rem;

    color: #333;

}

.card-price {

    font-size: 2rem;

    color: #333;

    margin: 15px 0;

}

.card-features {

    list-style: none;

    margin: 15px 0;

    padding: 0;

}

.card-features li {

    padding: 10px 0;

    border-bottom: 1px solid #eee;

}

.card-features li:last-child {

    border-bottom: none;

}

.btn {

    display: inline-block;

    padding: 10px 20px;

    background-color: #007bff;

    color: #fff;

    border: none;

    border-radius: 5px;

    cursor: pointer;

    margin-top: 15px;

}

.btn:hover {

    background-color: #0056b3;

}

/\* Responsive Design \*/

@media (max-width: 768px) {

    .card-list {

        flex-direction: column;

        align-items: center;

    }

    .card {

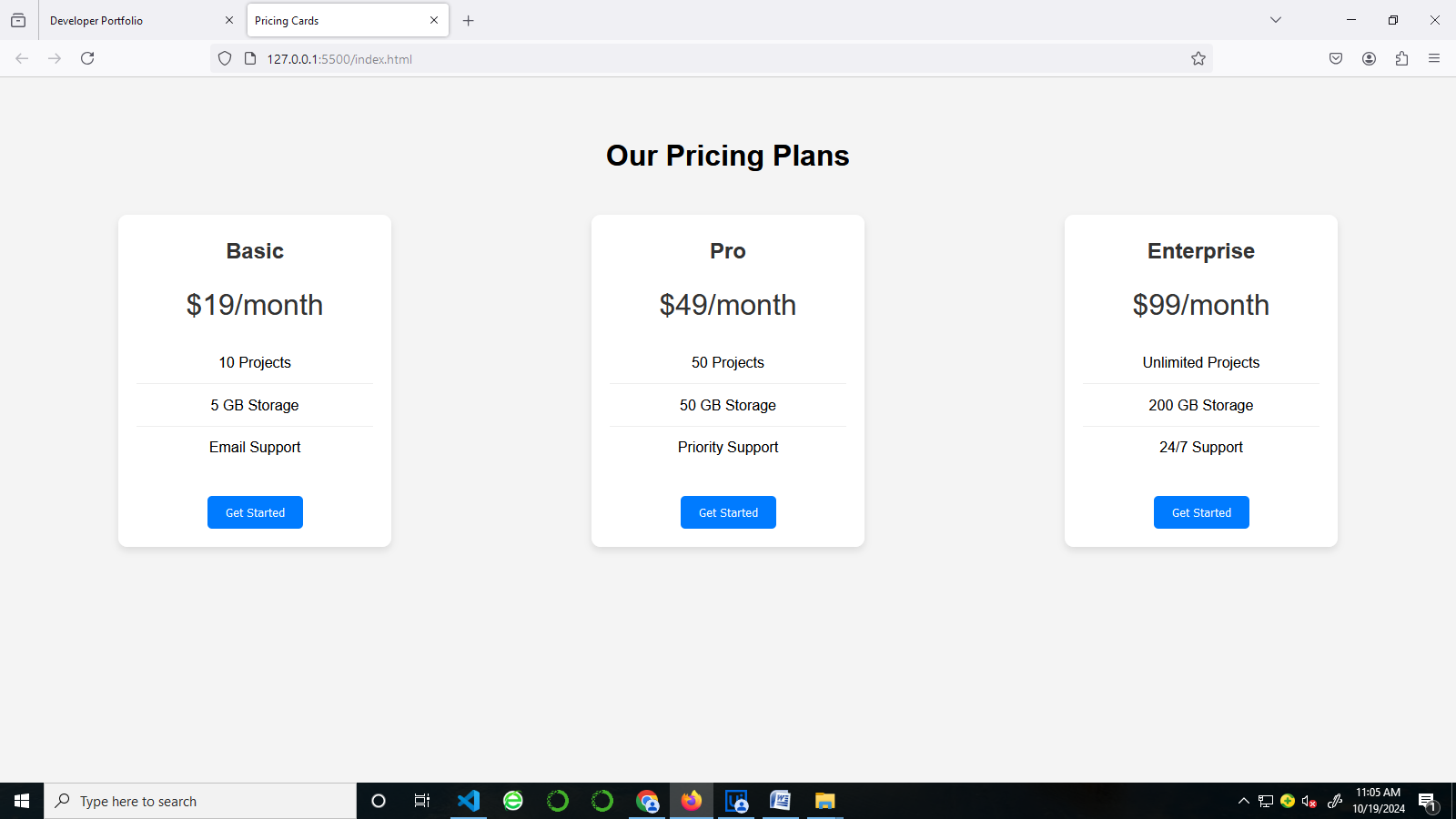
        width: 100%;

        max-width: 400px;

    }

}

Output:



7.DEVELOP A REGISTER FORM AND VALIDATE IT USING JAVASCRIPT DESIGN THE FORMS USING CSS AND DISPLAY ERROR MESSAGE HTML 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>Registration Form</title>

    <link rel="stylesheet" href="styles.css">

</head>

<body>

    <div class="container">

        <h2>Register</h2>

        <form id="registerForm">

            <div class="form-group">

                <label for="username">Username</label>

                <input type="text" id="username" name="username" placeholder="Enter your username">

                <small class="error-message" id="usernameError"></small>

            </div>

            <div class="form-group">

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

                <input type="email" id="email" name="email" placeholder="Enter your email">

                <small class="error-message" id="emailError"></small>

            </div>

            <div class="form-group">

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

                <input type="password" id="password" name="password" placeholder="Enter your password">

                <small class="error-message" id="passwordError"></small>

            </div>

            <div class="form-group">

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

                <input type="password" id="confirmPassword" name="confirmPassword" placeholder="Confirm your password">

                <small class="error-message" id="confirmPasswordError"></small>

            </div>

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

        </form>

    </div>

    <script src="script.js"></script>

</body>

</html>

Script.js

document.getElementById('registerForm').addEventListener('submit', function(e) {

    e.preventDefault(); // Prevent the form from submitting

    // Clear previous error messages

    clearErrorMessages();

    // Form fields

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

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

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

    const confirmPassword = document.getElementById('confirmPassword').value.trim();

    // Validation flags

    let isValid = true;

    // Username validation

    if (username === "") {

        showError('usernameError', 'Username is required');

        isValid = false;

    }

    // Email validation

    if (email === "") {

        showError('emailError', 'Email is required');

        isValid = false;

    } else if (!validateEmail(email)) {

        showError('emailError', 'Please enter a valid email address');

        isValid = false;

    }

    // Password validation

    if (password === "") {

        showError('passwordError', 'Password is required');

        isValid = false;

    } else if (password.length < 6) {

        showError('passwordError', 'Password must be at least 6 characters');

        isValid = false;

    }

    // Confirm Password validation

    if (confirmPassword === "") {

        showError('confirmPasswordError', 'Please confirm your password');

        isValid = false;

    } else if (confirmPassword !== password) {

        showError('confirmPasswordError', 'Passwords do not match');

        isValid = false;

    }

    // If form is valid, submit or display success

    if (isValid) {

        alert('Form submitted successfully!');

        // Optionally, you can submit the form or perform any further actions here

    }

});

// Function to show error messages

function showError(elementId, message) {

    const errorElement = document.getElementById(elementId);

    errorElement.textContent = message;

    errorElement.style.display = 'block';

}

// Function to clear all error messages

function clearErrorMessages() {

    document.querySelectorAll('.error-message').forEach(function(error) {

        error.style.display = 'none';

    });

}

// Function to validate email format

function validateEmail(email) {

    const re = /^[a-zA-Z0-9.\_-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,6}$/;

    return re.test(String(email).toLowerCase());

}

Style.css

\* {

    margin: 0;

    padding: 0;

    box-sizing: border-box;

}

body {

    font-family: Arial, sans-serif;

    background-color: #f4f4f4;

    display: flex;

    justify-content: center;

    align-items: center;

    height: 100vh;

}

.container {

    background-color: #fff;

    padding: 30px;

    border-radius: 10px;

    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);

    width: 350px;

}

h2 {

    text-align: center;

    margin-bottom: 20px;

}

.form-group {

    margin-bottom: 15px;

}

.form-group label {

    display: block;

    margin-bottom: 5px;

}

.form-group input {

    width: 100%;

    padding: 10px;

    border: 1px solid #ccc;

    border-radius: 5px;

}

.btn {

    width: 100%;

    padding: 10px;

    background-color: #007bff;

    color: #fff;

    border: none;

    border-radius: 5px;

    cursor: pointer;

    margin-top: 10px;

}

.btn:hover {

    background-color: #0056b3;

}

.error-message {

    color: red;

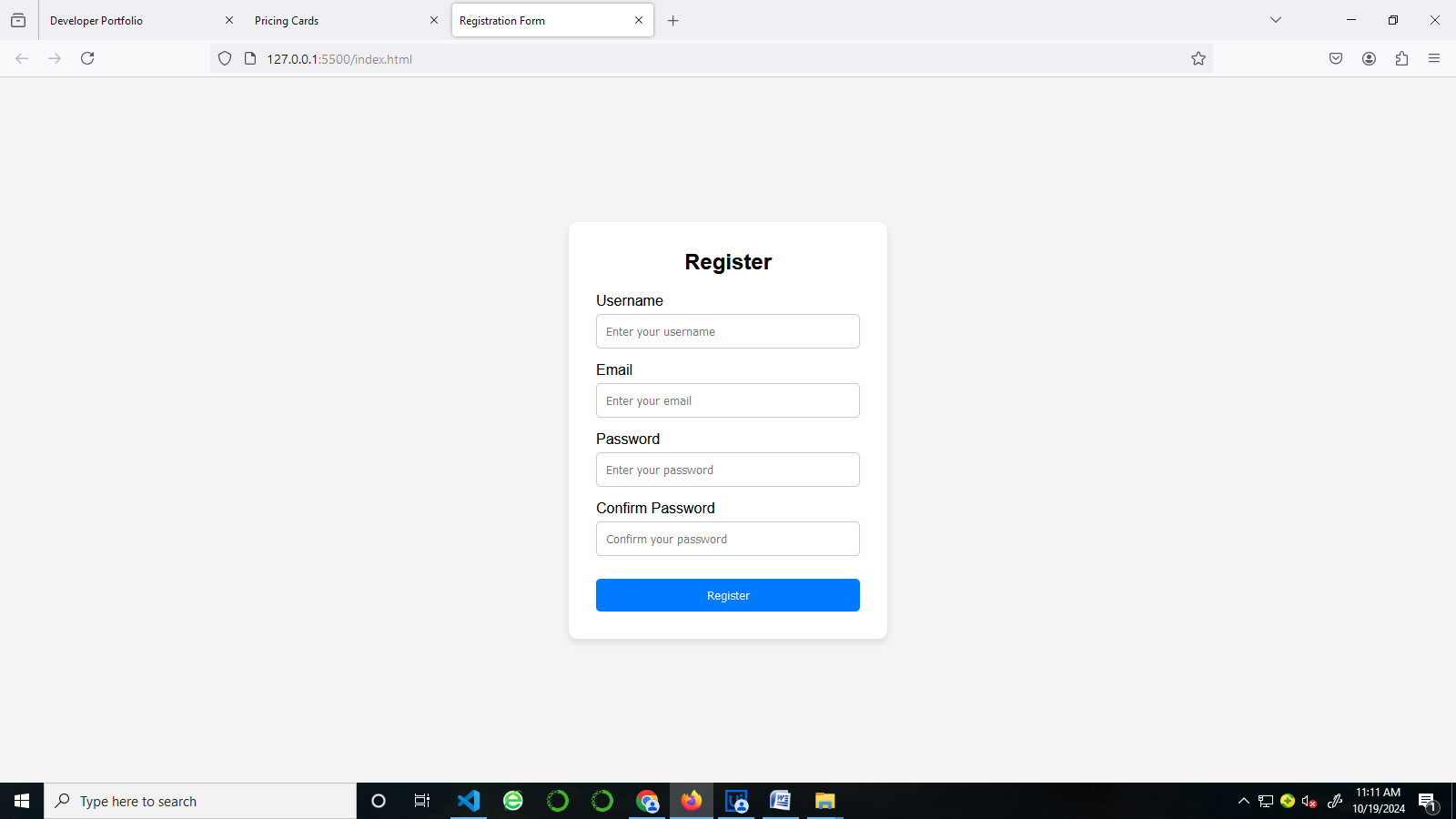
    font-size: 0.875rem;

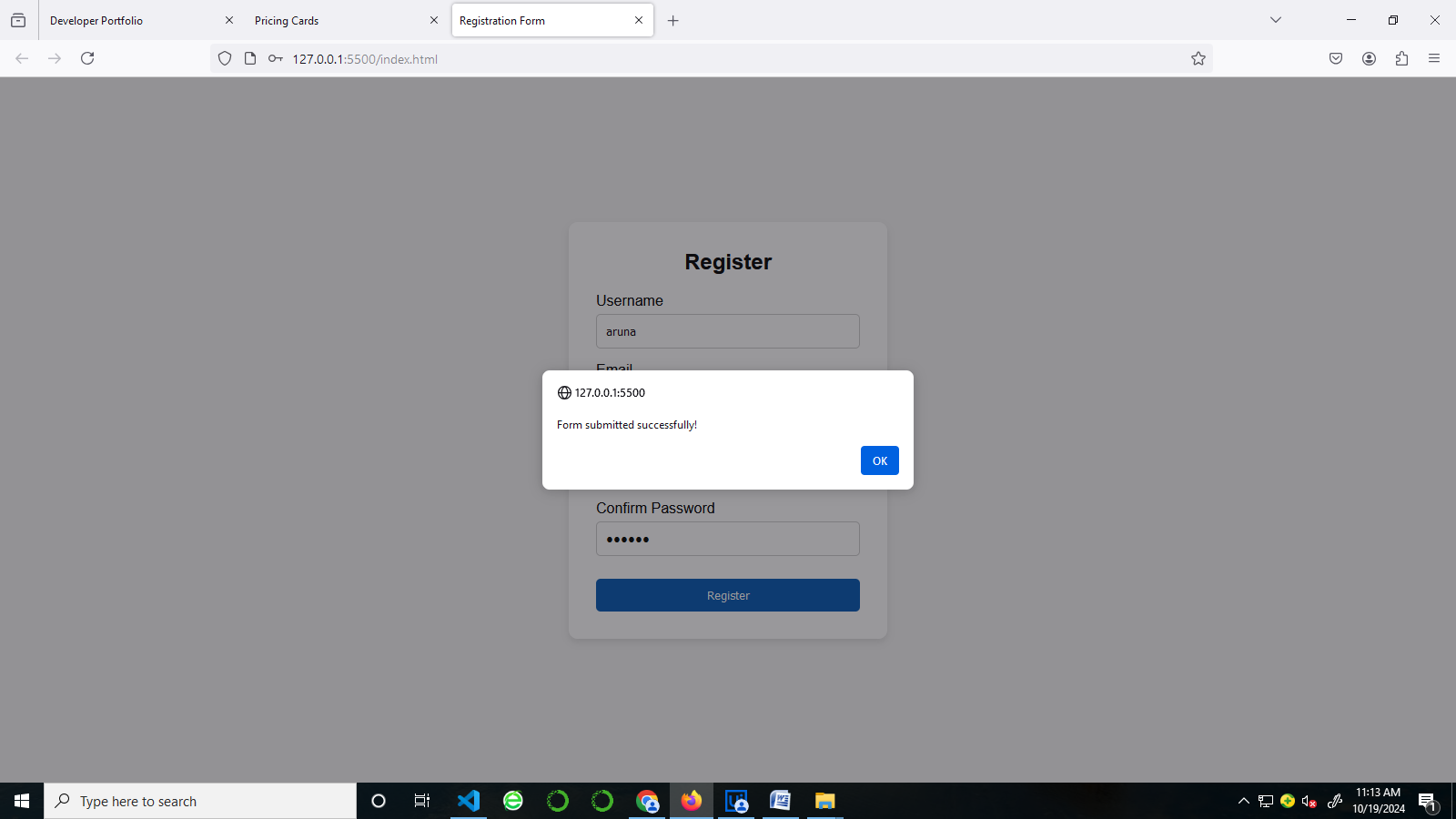
    margin-top: 5px;

    display: none;

}

Output:





8.DEVELOP A WEBSITE THAT USES THE JSON PLACEHOLDERS API TO GET POSTS DATA AND DISPLAY THEM IN THE FORM OF A CARD.USE FLEXBOX TO STYLE THE CARDS.

Index.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Posts</title>

    <link rel="stylesheet" href="styles.css">

</head>

<body>

    <div class="container">

        <h1>Posts</h1>

        <div class="posts" id="posts">

            <!-- Cards will be inserted here dynamically using JavaScript -->

        </div>

    </div>

    <script src="script.js"></script>

</body>

</html>

Script.js

// Fetch posts data from JSONPlaceholder API and display in cards

document.addEventListener("DOMContentLoaded", function () {

    const postsContainer = document.getElementById('posts');

    // Fetch data from the API

    fetch('https://jsonplaceholder.typicode.com/posts')

        .then(response => response.json())

        .then(posts => {

            displayPosts(posts);

        })

        .catch(error => console.error('Error fetching posts:', error));

    // Function to display posts in cards

    function displayPosts(posts) {

        posts.forEach(post => {

            // Create a card for each post

            const postCard = document.createElement('div');

            postCard.classList.add('post-card');

            // Post title

            const postTitle = document.createElement('h2');

            postTitle.textContent = post.title;

            postCard.appendChild(postTitle);

            // Post body

            const postBody = document.createElement('p');

            postBody.textContent = post.body;

            postCard.appendChild(postBody);

            // Append the card to the posts container

            postsContainer.appendChild(postCard);

        });

    }

});

Style.css

\* {

    margin: 0;

    padding: 0;

    box-sizing: border-box;

}

body {

    font-family: Arial, sans-serif;

    background-color: #f8f9fa;

    display: flex;

    justify-content: center;

    align-items: center;

    height: 100vh;

}

.container {

    width: 80%;

    margin: 0 auto;

    text-align: center;

}

h1 {

    margin-bottom: 20px;

    color: #333;

}

/\* Flexbox layout for posts \*/

.posts {

    display: flex;

    flex-wrap: wrap;

    justify-content: space-between;

    gap: 20px;

}

/\* Card styling \*/

.post-card {

    background-color: white;

    border-radius: 8px;

    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);

    width: calc(33% - 20px);

    padding: 20px;

    display: flex;

    flex-direction: column;

}

.post-card h2 {

    font-size: 1.25rem;

    color: #007bff;

    margin-bottom: 10px;

}

.post-card p {

    font-size: 1rem;

    color: #555;

    margin-bottom: 15px;

}

@media (max-width: 768px) {

    .post-card {

        width: calc(50% - 20px);

    }

}

@media (max-width: 500px) {

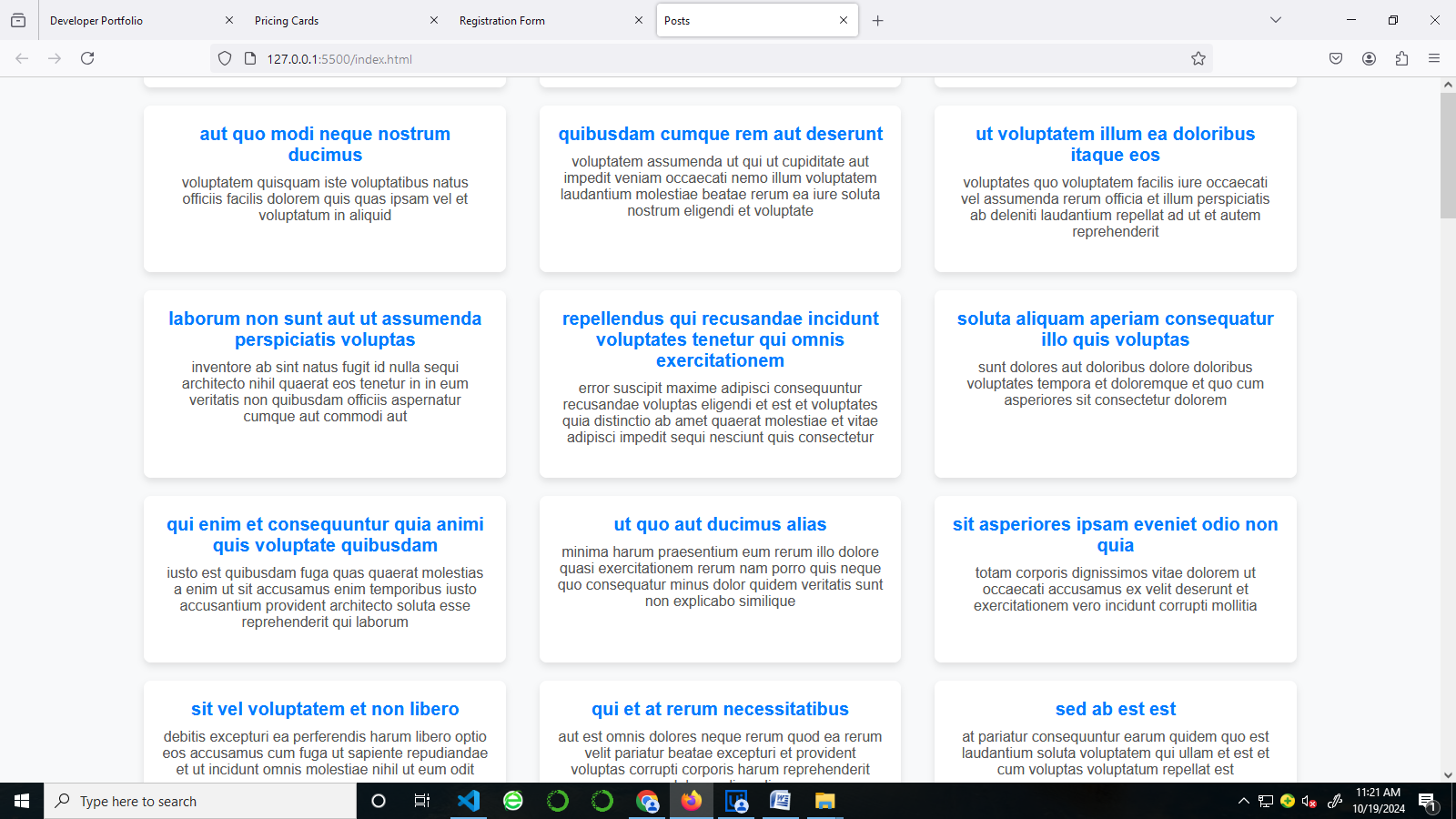
    .post-card {

        width: 100%;

    }

}

Output:



10.DEVELOP A PHP SERVER THAT REGISTER AND AUTHENTICATE USER SESSION AND STORES USER DATA IN MYSQL DATABASE.

TXT:

Database Setup

First, create a database and a table for storing user data.

1.SQL to Create Database and Table

CREATE DATABASE user\_auth;

USE user\_auth;

CREATE TABLE users (

id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(50) NOT NULL UNIQUE,

email VARCHAR(100) NOT NULL UNIQUE,

password VARCHAR(255) NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

2. PHP Server Setup

File Structure:

/user-auth-app

/config

db.php

/auth

register.php

login.php

logout.php

dashboard.php

index.php

Step 1: Database Connection

Create a db.php file in the config folder to connect to the MySQL database.

config/db.php (Database Connection)

<?php

$host = 'localhost';

$dbname = 'user\_auth';

$username = 'root'; // Replace with your MySQL username

$password = ''; // Replace with your MySQL password

try {

$pdo = new PDO("mysql:host=$host;dbname=$dbname", $username, $password);

$pdo->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

} catch (PDOException $e) {

die("Connection failed: " . $e->getMessage());

}

?>

Step 2: User Registration

Create a register.php file in the auth folder to handle user registration. We’ll use password hashing to store secure passwords.

auth/register.php

<?php

session\_start();

include\_once '../config/db.php';

// Check if form is submitted

if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {

$username = $\_POST['username'];

$email = $\_POST['email'];

$password = $\_POST['password'];

$confirm\_password = $\_POST['confirm\_password'];

// Validate form data

if (empty($username) || empty($email) || empty($password)) {

echo 'All fields are required.';

} elseif ($password !== $confirm\_password) {

echo 'Passwords do not match.';

} else {

// Hash the password

$hashed\_password = password\_hash($password, PASSWORD\_DEFAULT);

// Insert user into the database

try {

$stmt = $pdo->prepare("INSERT INTO users (username, email, password) VALUES (?, ?, ?)");

$stmt->execute([$username, $email, $hashed\_password]);

$\_SESSION['username'] = $username; // Start a session for the user

header('Location: ../dashboard.php'); // Redirect to dashboard after registration

} catch (PDOException $e) {

echo 'Registration failed: ' . $e->getMessage();

}

}

}

?>

<!-- Registration Form -->

<form action="register.php" method="POST">

<label>Username:</label>

<input type="text" name="username" required><br>

<label>Email:</label>

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

<label>Password:</label>

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

<label>Confirm Password:</label>

<input type="password" name="confirm\_password" required><br>

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

</form>

Step 3: User Login and Session Creation

Create a login.php file in the auth folder to authenticate the user and start a session.

auth/login.php

<?php

session\_start();

include\_once '../config/db.php';

// Check if form is submitted

if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {

$email = $\_POST['email'];

$password = $\_POST['password'];

// Validate form data

if (empty($email) || empty($password)) {

echo 'Both fields are required.';

} else {

// Check if user exists in the database

$stmt = $pdo->prepare("SELECT \* FROM users WHERE email = ?");

$stmt->execute([$email]);

$user = $stmt->fetch(PDO::FETCH\_ASSOC);

if ($user && password\_verify($password, $user['password'])) {

// Start session and redirect to dashboard

$\_SESSION['username'] = $user['username'];

header('Location: ../dashboard.php');

} else {

echo 'Invalid email or password.';

}

}

}

?>

<!-- Login Form -->

<form action="login.php" method="POST">

<label>Email:</label>

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

<label>Password:</label>

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

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

</form>

Step 4: User Logout

Create a logout.php file in the auth folder to destroy the session and log the user out.

auth/logout.php

<?php

session\_start();

session\_destroy();

header('Location: ../index.php'); // Redirect to home page after logout

exit();

Step 5: User Dashboard (Protected Page)

Create a dashboard.php file to display content only to logged-in users.

dashboard.php

<?php

session\_start();

// Check if the user is logged in

if (!isset($\_SESSION['username'])) {

header('Location: index.php'); // Redirect to home page if not logged in

exit();

}

?>

<h1>Welcome, <?php echo $\_SESSION['username']; ?>!</h1>

<a href="auth/logout.php">Logout</a>

Step 6: Index (Home Page)

Create an index.php file to provide links to the registration and login pages.

index.php

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>User Authentication System</title>

</head>

<body>

<h1>Welcome to the User Authentication System</h1>

<a href="auth/register.php">Register</a> |

<a href="auth/login.php">Login</a>

</body>

</html>

Step 7: Running the Application

Start Apache and MySQL:

Use XAMPP or MAMP to start your Apache and MySQL servers.

Place your project in the server's root directory:

For XAMPP: C:/xampp/htdocs/user-auth-app/

For MAMP: /Applications/MAMP/htdocs/user-auth-app/

Test the application:

Open your browser and go to http://localhost/user-auth-app/index.php.

Register a user, log in, and navigate to the dashboard.

Test the logout functionality.

9.STORE BOOK INFORMATION IN A DATABASE USING PHP

TXT:

1. Database Setup

First, create a MySQL database and table for the Todos.

SQL for Database and Table Creation

CREATE DATABASE todo\_app;

USE todo\_app;

CREATE TABLE todos (

id INT AUTO\_INCREMENT PRIMARY KEY,

title VARCHAR(255) NOT NULL,

description TEXT NOT NULL,

status ENUM('pending', 'completed') DEFAULT 'pending',

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

2. PHP Server Setup

We'll create the PHP API endpoints to handle CRUD operations for the Todo items.

File Structure:

/todo-app

/config

db.php

index.php

Step 1: Database Connection

Create a db.php file to establish a connection to the MySQL database.

db.php (Database Connection)

<?php

$host = 'localhost';

$dbname = 'todo\_app';

$username = 'root'; // Replace with your MySQL username

$password = ''; // Replace with your MySQL password

try {

$pdo = new PDO("mysql:host=$host;dbname=$dbname", $username, $password);

$pdo->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

} catch (PDOException $e) {

die("Connection failed: " . $e->getMessage());

}

?>

Step 2: Create CRUD Operations

In the index.php, handle the Create, Read, Update, and Delete operations.

index.php

<?php

// Include the database connection

include\_once 'config/db.php';

// Set the content type to JSON

header('Content-Type: application/json');

// Get the HTTP method

$method = $\_SERVER['REQUEST\_METHOD'];

// Get input data if any

$input = json\_decode(file\_get\_contents('php://input'), true);

// Handle CRUD operations based on the HTTP method

switch ($method) {

case 'GET':

// Fetch Todos

if (isset($\_GET['id'])) {

// Fetch a single todo by ID

$stmt = $pdo->prepare("SELECT \* FROM todos WHERE id = ?");

$stmt->execute([$\_GET['id']]);

$todo = $stmt->fetch(PDO::FETCH\_ASSOC);

if ($todo) {

echo json\_encode($todo);

} else {

echo json\_encode(['message' => 'Todo not found']);

}

} else {

// Fetch all todos

$stmt = $pdo->prepare("SELECT \* FROM todos");

$stmt->execute();

$todos = $stmt->fetchAll(PDO::FETCH\_ASSOC);

echo json\_encode($todos);

}

break;

case 'POST':

// Create a new Todo

if (isset($input['title']) && isset($input['description'])) {

$stmt = $pdo->prepare("INSERT INTO todos (title, description) VALUES (?, ?)");

$stmt->execute([$input['title'], $input['description']]);

$id = $pdo->lastInsertId();

echo json\_encode(['message' => 'Todo created', 'id' => $id]);

} else {

echo json\_encode(['message' => 'Invalid input']);

}

break;

case 'PUT':

// Update a Todo

if (isset($\_GET['id']) && isset($input['title']) && isset($input['description'])) {

$stmt = $pdo->prepare("UPDATE todos SET title = ?, description = ?, status = ? WHERE id = ?");

$stmt->execute([$input['title'], $input['description'], $input['status'], $\_GET['id']]);

if ($stmt->rowCount()) {

echo json\_encode(['message' => 'Todo updated']);

} else {

echo json\_encode(['message' => 'Todo not found or no changes made']);

}

} else {

echo json\_encode(['message' => 'Invalid input or Todo ID missing']);

}

break;

case 'DELETE':

// Delete a Todo

if (isset($\_GET['id'])) {

$stmt = $pdo->prepare("DELETE FROM todos WHERE id = ?");

$stmt->execute([$\_GET['id']]);

if ($stmt->rowCount()) {

echo json\_encode(['message' => 'Todo deleted']);

} else {

echo json\_encode(['message' => 'Todo not found']);

}

} else {

echo json\_encode(['message' => 'Todo ID missing']);

}

break;

default:

echo json\_encode(['message' => 'Method not supported']);

break;

}

?>