EXPERIMENYT:1

1. Build a responsive web application for shopping cart with registration, login, catalog and cart pages using CSS3 features, flex and grid.

Project Structure

HTML Files: Different pages for registration, login, catalog, and cart.

CSS Stylesheets: Main CSS file for styling.

JavaScript Files: For adding interactivity like cart management and user login/registration logic.

Backend Services: To manage user data and product catalog (you can use a simple Node.js/Express backend, but let's focus on the front-end in this guide).

Step 1: Set Up the HTML Structure

Start by creating the following HTML files:

index.html: The landing page and product catalog.

login.html: The user login page.

register.html: The user registration page.

cart.html: The shopping cart page.

Step 2: Create CSS Stylesheets

Create a styles.css file to hold the CSS rules for your application. Use Flexbox and Grid to ensure responsive layouts.

Step 3: Add HTML Content

Here's a simple outline for each of the pages with some key CSS styling features.

Catalog Page (index.html)

```
<IDOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Product Catalog</title>
<link rel="stylesheet" href="styles.css">
</head>
<body>
<header>
<h1>Welcome to Our Shop</h1>
<nav>
<a href="index.html">Catalog</a>
<a href="cart.html">Cart</a>
<a href="login.html">Login</a>
</nav>
</header>
<main>
<div class="product-grid">
<div class="product">
<img src="example-product.jpg" alt="Product Image">
<h3>Product Name</h3>
<p>$19.99</p>
<button>Add to Cart</button>
</div>
<!-- Add more product divs as needed -->
</div>
</main>
<footer>
<p>&copy: 2024 Our Shop. All rights reserved.</p>
</footer>
</body>
</htm]>
```

Login Page (login.html)

```
<!DOCTYPE html>
<html lang=""en"">
<head>
<meta charset="UTF-8">
<title>Login</title>
```

<link rel="stylesheet' href="styles.css">
</head>
<body>
<header>
<h1>Login</h1>
<nav>
Catalog
Cart
</mav>
</header>
<main>
<form id="login-form'>
<label for=""username'>Username:</label>
<input type=""text" id=""username' name="username'>
<label for=""password'>Password:</label>
<input type="password" id=""password" name="password'>
<button type="submit'>Login</button>
</form>
</main>
<footer>
© 2024 Our Shop. All rights reserved.
</footer>
</body>
</html>
Registration Page (register.html)
<!DOCTYPE html>
<html lang=""en">
<head>
<meta charset="UTF-8">
<title>Register</title>
<link rel="stylesheet'" href=""styles.css'">
</head>
<body>
<header>
<h1>Register</h1>
<nav>
Catalog
Cart
</nav>
</header>
<main>
<form id=""register-form'>
<label for=""username'>Username:</label>
<input type=""text" id="username' name=""username'>
<label for="password'>Password:</label>
n
<input type=""password" id=""password" name="password'">
<button type="submit'>Register</button>
</form>
</main>
<footer>
© 2024 Our Shop. All rights reserved.
</footer>
</body>
</html>
Cart Page (cart.html)
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">

```
<title>Shopping Cart</title>
<link rel="stylesheet" href="styles.css">
</head>
<body>
<header>
<h1>Shopping Cart</h1>
<nav>
<a href="index.htm|">Catalog</a>
<a href="login.html">Login</a>
</nav>
</header>
<main>
<div class="cart-items">
<div class="cart-item">
<h3>Product Name</h3>
<p>Quantity: 1</p>
<p>Price: $19.99</p>
</div>
<!-- Add more cart items as needed -->
</div>
<div class="cart-summary">
<h3>Total: $19.99</h3>
<button>Checkout</button>
</div>
</main>
<footer>
<p>&copy; 2024 Our Shop. All rights reserved.</p>
</footer>
</body>
</html>
```

Step 4: Style with CSS

Here's an example of the CSS styling with Flexbox and Grid for a responsive layout (styles.css):

```
/* Basic Styles */
body {
font-family: Arial, sans-serif,
margin: 0;
padding: 0;
header {
background-color: #333;
color: white;
text-align: center;
padding: lem;
nav {
display: flex;
justify-content: center;
nava {
color: white;
margin: 0 10px;
text-decoration: none;
main {
padding: 20px;
/* Product Grid */
.product-grid {
display: grid;
grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));
gap: 20px;
.product {
border: 1px solid #ddd;
padding: 10px;
```

```
text-align: center;
.product img {
max-width: 100%;
.product button {
background-color: #333;
color: white;
border: none;
padding: 10px;
cursor: pointer;
.product button:hover {
background-color: #555;
/* Cart */
.cart-items {
display: flex;
flex-direction: column;
.cart-item {
border-bottom: 1px solid #ddd:
padding: 10px;
.cart-summary {
text-align: right;
padding: 10px;
.cart-summary button {
background-color: #333;
color: white;
border: none;
padding: 10px;
cursor: pointer;
.cart-summary button:hover {
background-color: #555;
/* Forms */
10
form {
max-width: 400px;
margin: 0 auto;
label {
display: block;
margin-bottom: 10px;
input {
width: 100%;
padding: 10px;
border: 1px solid #ddd;
margin-bottom: 20px:
button {
display: block;
width: 100%;
background-color: #333;
color: white;
padding: 10px;
border: none;
cursor: pointer;
11
button:hover {
background-color: #555;
footer {
text-align: center;
background-color: #333;
color: white;
padding: 10px;
```

Step 5: Add JavaScript Logic

For simplicity, here's an example of JavaScript logic for adding items to the cart and

handling login/registration (app.js):

```
document.addEventListener("DOMContentLoaded", () => {
// Add event listeners for adding to cart
const addCartButtons = document.querySelectorAll(".product button");
addCartButtons.forEach((button) => {
button.addEventListener("click", () => {
// Logic to add to cart
console.log("Item added to cart");
12
// Event listener for login form
const loginForm = document.getElementBylId("login-form");
if (loginForm) {
loginForm.addEventListener("submit", (¢) => {
e.preventDefault();
/I Login logic
console.log("User logged in");
D:
// Event listener for registration form
const registerForm = document.getElementById("register-form");
if (registerForm) {
registerForm.addEventListener("submit", (e) => {
e.preventDefault();
// Registration logic
console.log("User registered"):
s
13
```

EXPERIMENYT: 2

2. Make the above web application responsive web application using Bootstrap framework.

Project Structure

HTML Files: Registration, login, catalog, and cart pages.

CSS Stylesheets: Main CSS file for additional custom styles.

JavaScript Files: Logic for interactivity like managing the cart and user authentication.

Bootstrap Assets: Include Bootstrap's CSS and JavaScript for styling and functionality.

Step 1: Include Bootstrap Framework

In your HTML files, include the Bootstrap CSS and JavaScript files. You can use a CDN for convenience:

```
<!-- Include Bootstrap CSS -->
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
<!-- Include jQuery, Popper.js, and Bootstrap JavaScript -->
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
<script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@?2.11.6/dist□'umd/popper.min.js"></scr
ipt>
<script
sre="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
```

Step 2: Create HTML Pages with Bootstrap

Now let's create the pages with Bootstrap. We'll use the grid system, components like cards, forms, and utilities to build a responsive design.

Catalog Page (index.html)

```
<IDOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Product Catalog</title>
14
<link rel="stylesheet" href="styles.css">
<!-- Include Bootstrap -->
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
```

```
</head>
<body>
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
<a class="navbar-brand" href="#">Our Shop</a>
<button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-
label="Toggle navigation">
<span class=" navbar-toggler-icon"></span>
</button>
<div class="collapse navbar-collapse" id="navbarNav'">
<ul class="navbar-nav ml-auto">
<li class="nav-item">
<a class="nav-link" href="index.html">Catalog</a>
</li>
<li class="nav-item">
<a class="nav-link" href="cart.html">Cart</a>
</i>
<li class="nav-item">
<a class="nav-link" href="login.html">Login</a>
</li>
<ul>
</div>
</nav>
<div class="container mt-4">
<div class="row">
<!-- Product Card -->
<div class="col-md-4">
<div class="card">
<img src="example-product.jpg" class="card-img-top" alt="Product
Image">
<div class="card-body">
<h5 class="card-title">Product Name</h5>
<p class="card-text">$19.99</p>
<button class="btn btn-primary">Add to Cart</button>
</div>
15
</div>
</div>
<!-- Add more product cards as needed -->
</div>
</div>
<footer class="bg-dark text-white text-center py-3 mt-4">
<p>&copy; 2024 Our Shop. All rights reserved.</p>
</footer>
<!-- Include Bootstrap JavaScript -->
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
<script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.6/dist□'umd/popper.min.js"></scr
ipt>
<script
sre="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</body>
</htm]>
```

Login Page (login.html)

```
<IDOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Login</title>
<link rel="stylesheet" href="styles.css">
```

```
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
</head>
<body>
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
<a class="navbar-brand" href="#">Our Shop</a>
<div class="collapse navbar-collapse">
<ul class="navbar-nav ml-auto">
<li class="nav-item">
<a class="nav-link" href="index.htm!">Catalog</a>
</li>
<li class="nav-item">
<a class="nav-link" href="cart.html">Cart</a>
</li>
16
</ul>
</div>
</nav>
<div class="container mt-4">
<h2>Login</h2>
<form id="login-form">
<div class="form-group">
<label for="username">Username:</label>
<input type="text" class="form-control" id="username" name="username">
</div>
<div class="form-group">
<label for="password">Password:</label>
<input type="password" class="form-control" id="password"
name="password">
</div>
<button type="submit" class="btn btn-primary">Login</button>
</form>
</div>
<footer class="bg-dark text-white text-center py-3 mt-4">
<p>&copy; 2024 Our Shop. All rights reserved.</p>
</footer>
<!-- Include Bootstrap JavaScript -->
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
<script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.6/dist/umd/popper.min.js"></scr
ipt>
<script
sre="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</body>
</html>
```

Registration Page (register.html)

```
<IDOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Register</title>
<link rel="stylesheet" href="styles.css">
17
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
</head>
<body>
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
<a class="navbar-brand" href="#">Our Shop</a>
<div class="collapse navbar-collapse">
```

```
_n
<ul class="navbar-nav ml-auto">
<li class="nav-item">
<a class="nav-link" href="index.html">Catalog</a>
</li>
<li class="nav-item">
<a class="nav-link" href="cart.html">Cart</a>
</li>
<>
</div>
</nav>
<div class="container mt-4">
<h2>Register</h2>
<form id="register-form">
<div class="form-group">
<label for="username">Username:</label>
<input type="text" class="form-control" id="username" name="username">
</div>
<div class="form-group">
<label for="password">Password:</label>
<input type="password" class="form-control" id="password"
name="password">
</div>
<button type="submit" class="btn btn-primary">Register</button>
</form>
</div>
<footer class="bg-dark text-white text-center py-3 mt-4">
<p>&copy; 2024 Our Shop. All rights reserved.</p>
</footer>
<!-- Include Bootstrap JavaScript -->
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
18
<script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@?2.11.6/dist□'umd/popper.min.js"></scr
ipt>
<script
sre="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</body>
</html>
```

Cart Page (cart.html)

```
<IDOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Shopping Cart</title>
<link rel="stylesheet" href="styles.css">
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
</head>
<body>
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
<a class="navbar-brand" href="#">Our Shop</a>
<div class="collapse navbar-collapse">
<ul class="navbar-nav ml-auto">
<li class="nav-item">
<a class="nav-link" href="index.html">Catalog</a>
</li>
<li class="nav-item">
<a class="nav-link" href="login.html">Login</a>
</
```

EXPERIMENT:3
3. Use JavaScript for doing client side validation of the pages implemented in experiment 1 and experiment 2.
client-side validation helps ensure that user input meets certain criteria before it's sent to the server. This reduces the number of server-side validation checks and can provide a better user experience by alerting users to errors early.
Here's an outline to implement JavaScript-based client-side validation for a shopping cart application with registration, login, catalog, and cart pages. We'll focus on validation for the registration and login forms.
Validation Checks
Common validation checks include:
Required fields: Ensuring certain fields are not left blank.
Minimum/maximum length: For example, a username should have a minimum and maximum length.
Email format: Validating if an email address is in the correct format.
Password complexity: Checking if the password meets complexity requirements (e.g., contains a mix of characters, numbers, symbols, etc.).
Let's use the registration and login pages from our previous response and add JavaScript-based validation logic.
Registration Page (register.html)
```
<IDOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Register</title>
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
</head>
<body>
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
<a class="navbar-brand" href="#">Our Shop</a>
<div class="collapse navbar-collapse">
<ul class="navbar-nav ml-auto">
<li class="nav-item">
```

```
<a class="nav-link" href="index.html">Catalog</a>
</li>
<li class="nav-item">
<a class="nav-link" href="cart.html">Cart</a>
</i>
<ful>
</div>
</nav>
<div class="container mt-4">
<h2>Register</h2>
<form id="register-form">
<div class="form-group">
<label for="username">Username:</label>
<input type="text" class="form-control" id="username" name="username"
required>
<small class="form-text text-muted">Username must be between 3 and 20
characters long.</small>
</div>
<div class="form-group">
<label for="email">Email:</label>
<input type="email" class="form-control" id="email" name="email" required>
</div>
<div class="form-group">
<label for="password">Password:</label>
```

```
<input type="password" class="form-control" id="password"
name="password" required>
<small class="form-text text-muted">Password must contain at least 8
characters, including at least one uppercase letter, one lowercase letter, and one
number.</small>
</div>
<button type="submit" class="btn btn-primary">Register</button>
</form>
</div>
<footer class="bg-dark text-white text-center py-3 mt-4">
<p>&copy; 2024 Our Shop. All rights reserved.</p>
</footer>
<!-- JavaScript for client-side validation -->
21
<script>
document.getElementByIId("register-form").addEventListener("submit",
function(event) {
const username = document.getElementById("username").value;
const email = document.getElementById("email").value;
const password = document.getElementById("password").value;
 .
let errorMessage 3
// Username validation
if (username.length <3 || username.length > 20) {
errorMessage += "Username must be between 3 and 20 characters long. ";
}
// Email validation
const emailPattern = /[ N\s@+@["\s@]+\.[\s@]+$/:
if (lemailPattern.test(email)) {
errorMessage += "Invalid email format. ";
1
s
// Password validation
const passwordPattern = /(?=*[a-z])(?=*[A-Z])(?=*\d)[A-Za-2\d] {8.} $/;
if (!passwordPattern.test(password)) {
errorMessage += "Password must contain at least one uppercase letter, one
lowercase letter, and one number, and be at least 8 characters long. ";
}
if (errorMessage) {
alert(errorMessage);
event.preventDefault(); // Prevent form submission if validation fails
)
1
</script>
<!-- Include Bootstrap JavaScript -->
<script sre="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
<script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.6/dist'umd/popper.min.js"></scr
ipt>
22
<script
sre="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</body>
</html>
```

This script validates the username, email, and password fields according to the specified criteria. If validation fails, it prevents the form from being submitted and displays an alert with the validation errors.

Login Page (login.html):

```
<IDOCTYPE html>
<html lang="en">
```

```
<head>
<meta charset="UTF-8">
<title>Login</title>
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
</head>
<body>
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
<a class="navbar-brand" href="#">Our Shop</a>
<div class="collapse navbar-collapse">
_n
<ul class="navbar-nav ml-auto">
<li class="nav-item">
<a class="nav-link" href="index.html">Catalog</a>
</li>
<li class="nav-item">
<a class="nav-link" href="cart.html">Cart</a>
</li>
</l>
</div>
</nav>
<div class="container mt-4">
<h2>Login</h2>
<form id="login-form">
<div class="form-group">
<label for="username">Username:</label>
<input type="text" class="form-control" id="username" name="username"
required>
</div>
23
<div class="form-group">
<label for="password">Password:</label>
<input type="password" class="form-control" id="password"
name="password" required>
</div>
<button type="submit" class="btn btn-primary">Login</button>
</form>
</div>
<footer class="bg-dark text-white text-center py-3 mt-4">
<p>&copy; 2024 Our Shop. All rights reserved.</p>
</footer>
<!-- JavaScript for client-side validation -->
<script>
document.getElementById("login-form").addEventListener("submit",
function(event) {
const username = document.getElementById("username").value;
const password = document.getElementById("password").value;
let errorMessage = "";
// Check if username and password are not empty
if (username.trim() ===="") {
errorMessage += "Username is required. ";
}
if (password.trim() === "") {
errorMessage += "Password is required. ";
1
s
if (errorMessage) {
alert(errorMessage);
event.preventDefault(); // Prevent form submission if validation fails
}
```

1
</script>
<!-- Include Bootstrap JavaScript -->
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
24
<script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.6/dist'umd/popper.min.js"></scr
ipt>
<script
sre="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</body>
</html>
In this script, the validation checks if the username and password fields are not empty before submitting the form. If the validation fails, an alert message is shown, and the form submission is prevented.
Conclusion
With the above scripts, you've added basic client-side validation to your registration and login pages. These validations ensure that users provide the correct information before submitting their forms. This doesn't replace server-side validation but complements it to improve user experience and reduce unnecessary server load.
25
EXPERIMENT: 4
4. Explore the features of ES6 like arrow functions, callbacks, promises, async/await. Implement an application for reading the weather information from openweathermap.org and display the information in the form of a graph on the web page.
To create a weather information application that retrieves weather data from OpenWeatherMap and displays it in the form of a graph, you can leverage ES6 features like arrow functions, callbacks, promises, and async/await. While I don't have internet access to connect to OpenWeatherMap or retrieve live weather data, I can guide you through the steps to implement this application in a realistic setting.
The implementation involves the following key steps:
Set up OpenWeatherMap: Sign up for an API key at OpenWeatherMap.
Fetch Weather Data: Use Fetch API with promises and async/await to get weather data.
Plot the Graph: Use a JavaScript library like Chart.js or D3.js to create the graph.
Use ES6 Features: Apply arrow functions, callbacks, promises, and async/await.
Let's break down the implementation:
Step 1: Set Up OpenWeatherMap
Sign up at OpenWeatherMap and obtain your API key.
Determine which weather data you want to retrieve (e.g., current weather, forecast).
Step 2: Fetch Weather Data
Use the Fetch API with promises and async/await to get weather data from OpenWeatherMap. Here's an example using promises:
const apiKey = "YOUR_API_KEY"; // Replace with your OpenWeatherMap API
key
const city = "London"'; / Example city
const url =
https://api.openweathermap.org/data/2.5/weather?q=${city} &appid=${apiKey}";
// Fetch weather data using Promises
fetch(url)
.then(response => response.json())
.then(data => {
console.log(""Weather Data:", data);
/I Process and use the weather data
D
26
.catch(error => {
console.error("Error fetching weather data:", error);
bH
Step 3: Fetch Data with Async/Await

The async/await syntax is more straightforward and helps with readability when dealing with promises. Here's the same example using async/await:

```
const apiKey = "YOUR_API_KEY";
const city = "London";
const url =
https://api.openweathermap.org/data/2.5/weather?q=${city} &appid=${apiKey}";
async function fetchWeatherData() {
try {
const response = await fetch(url);
const data = await response.json();
console.log(""Weather Data:", data);
// Process and use the weather data
} catch (error) {
console.error("Error fetching weather data:", error);
}
}
/I Call the function to fetch the weather data
fetchWeatherData();
```

Step 4: Display Data on a Web Page

To display weather information as a graph, use a charting library like Chart.js. Here's an example that fetches weather data and plots a simple graph with temperature data:

Html

```
<!DOCTYPE html>
<html lang="en"">
<head>
<meta charset="UTF-8">
<title>Weather Information</title>
<script sre=""https://cdn.jsdelivr.net/npm/chart.js"></script>
</head>
27
<body>
<h2>Weather Information</h2>
<canvas id="weather-chart" width="400" height="200"></canvas>
<script>
const apiKey = "YOUR_API_KEY";
const city = "London";
const url =
https://api.openweathermap.org/data/2.5/weather?q=${city} &appid=${apiKey}";
async function fetchWeatherData() {
try {
const response = await fetch(url);
const data = await response.json();
const tempK = data.main.temp; / Temperature in Kelvin
const tempC = tempK - 273.15; // Convert to Celsius
const ctx = document.getElementByld("weather-
chart").getContext(''2d");
const weatherChart = new Chart(ctx, {
type: "bar",
data: {
labels: ["Temperature'],
datasets: [{
label: "Temperature in ${city} (°C),
data: [tempC],
backgroundColor: ""rgba(75, 192, 192, 0.6)",
borderColor: "rgba(75, 192, 192, 1)",
borderWidth: 1
il
h
options: {
```

```
scales: {
YAxes: [{
ticks: {
beginAtZero: true
}
1
}
28
}
bH
} catch (error) {
console.error("Error fetching weather data:", error);
}
}
/I Call the function to fetch the weather data
fetchWeatherData();
</script>
</body>
</html>
```

This code snippet fetches weather data from OpenWeatherMap and plots a simple bar chart showing the current temperature in Celsius. You can use this as a starting point and expand it to display other weather parameters or use additional Chart.js features for more complex graphs.

Conclusion

This example demonstrates how to implement an application to fetch weather information and plot it on a graph using ES6 features like arrow functions, callbacks, promises, and async/await. By following this guide, you can build a functional weather information application with client-side visualization.