

Lesson 01 Demo 07

Generating E-commerce Product Module Using JavaScript

Objective: To create an e-commerce product module using JavaScript, enabling users to add, store, and view product details with seamless navigation and a well-structured interface

Tools required: ChatGPT and Visual Studio Code

Prerequisites: None


Steps to be followed:

1. Set up the project structure
2. Implement the product management pages
3. Link the pages and test the application

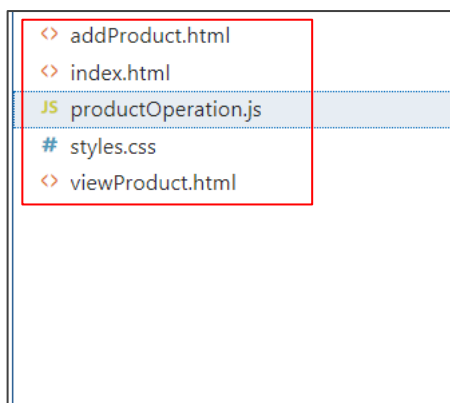
Note: The generative AI tool used in this exercise can produce varied outputs even when presented with the same prompts. Hence, you may get different outputs for the prompts provided.

Step 1: Set up the project structure

- 1.1 Create a folder named **e-commerce-product-app** on your system's local drive

 e-commerce-product-app

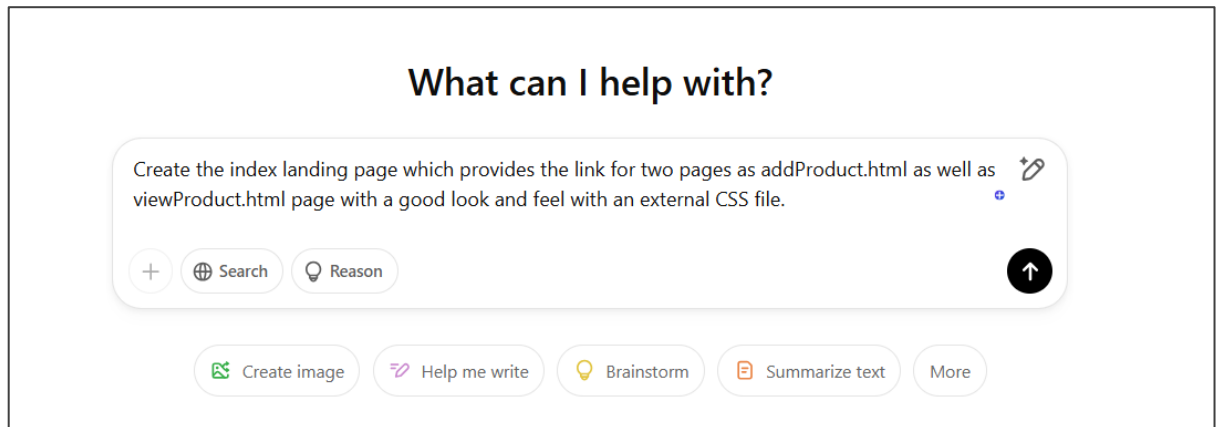
- 1.2 Create files named **addProduct.html**, **index.html**, **productOperation.js**, **styles.css**, and **viewProduct.html** inside e-commerce-app folder



Step 2: Implement the product management pages

- 2.1 Open ChatGPT and generate the HTML code for the index.html page using the following prompt:

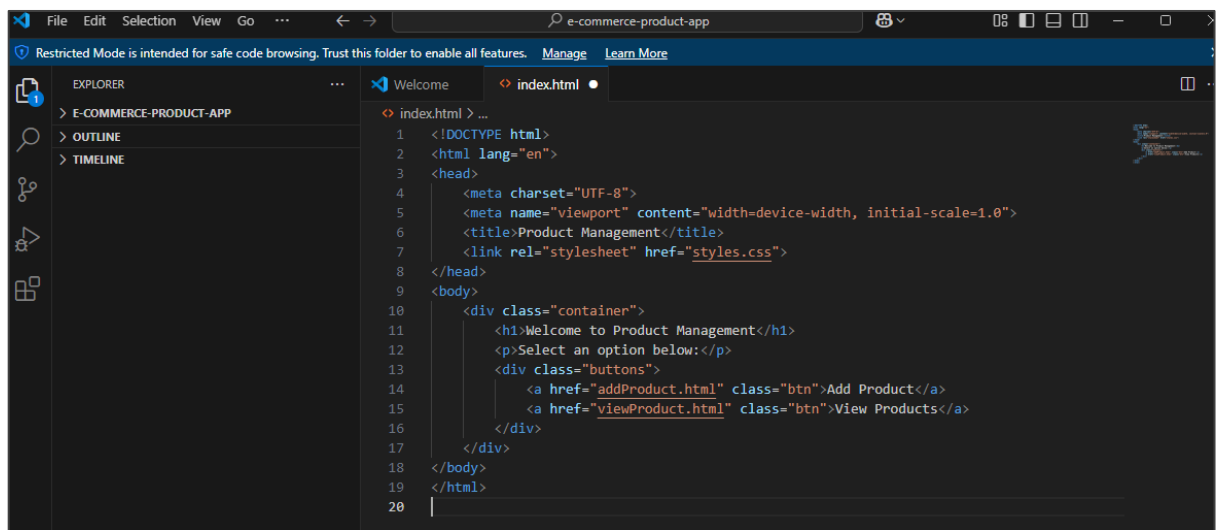
Create the index landing page which provides the link for two pages as addProduct.html as well as viewProduct.html page with a good look and feel with an external CSS file



- 2.2 Copy the generated code



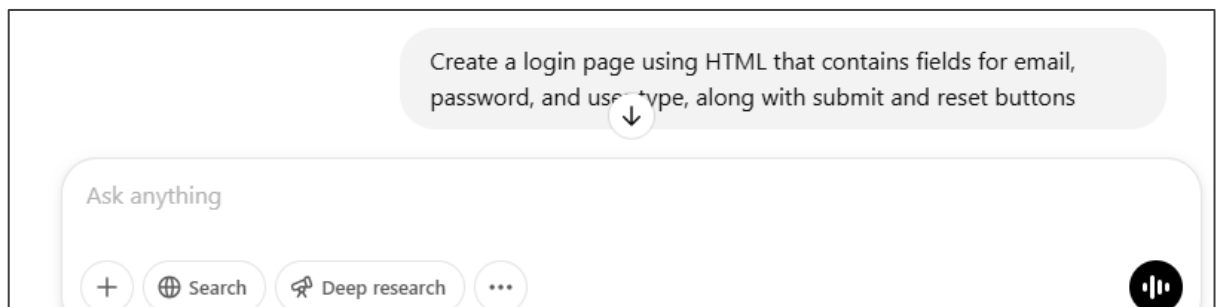
2.3 Open VS Code and paste the copied code into the index.html file



The screenshot shows the Visual Studio Code editor with the 'index.html' file open. The code is as follows:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Product Management</title>
7   <link rel="stylesheet" href="styles.css">
8 </head>
9 <body>
10   <div class="container">
11     <h1>Welcome to Product Management</h1>
12     <p>Select an option below:</p>
13     <div class="buttons">
14       <a href="addProduct.html" class="btn">Add Product</a>
15       <a href="viewProduct.html" class="btn">View Products</a>
16     </div>
17   </div>
18 </body>
19 </html>
20
```

2.4 Open ChatGPT and enter the following prompt to generate the code for the login page: **Create a login page using HTML that contains fields for email, password, and user type, along with submit and reset buttons**



2.5 Copy the generated code for the login.html and styles.css files separately



2. styles.css (External CSS)

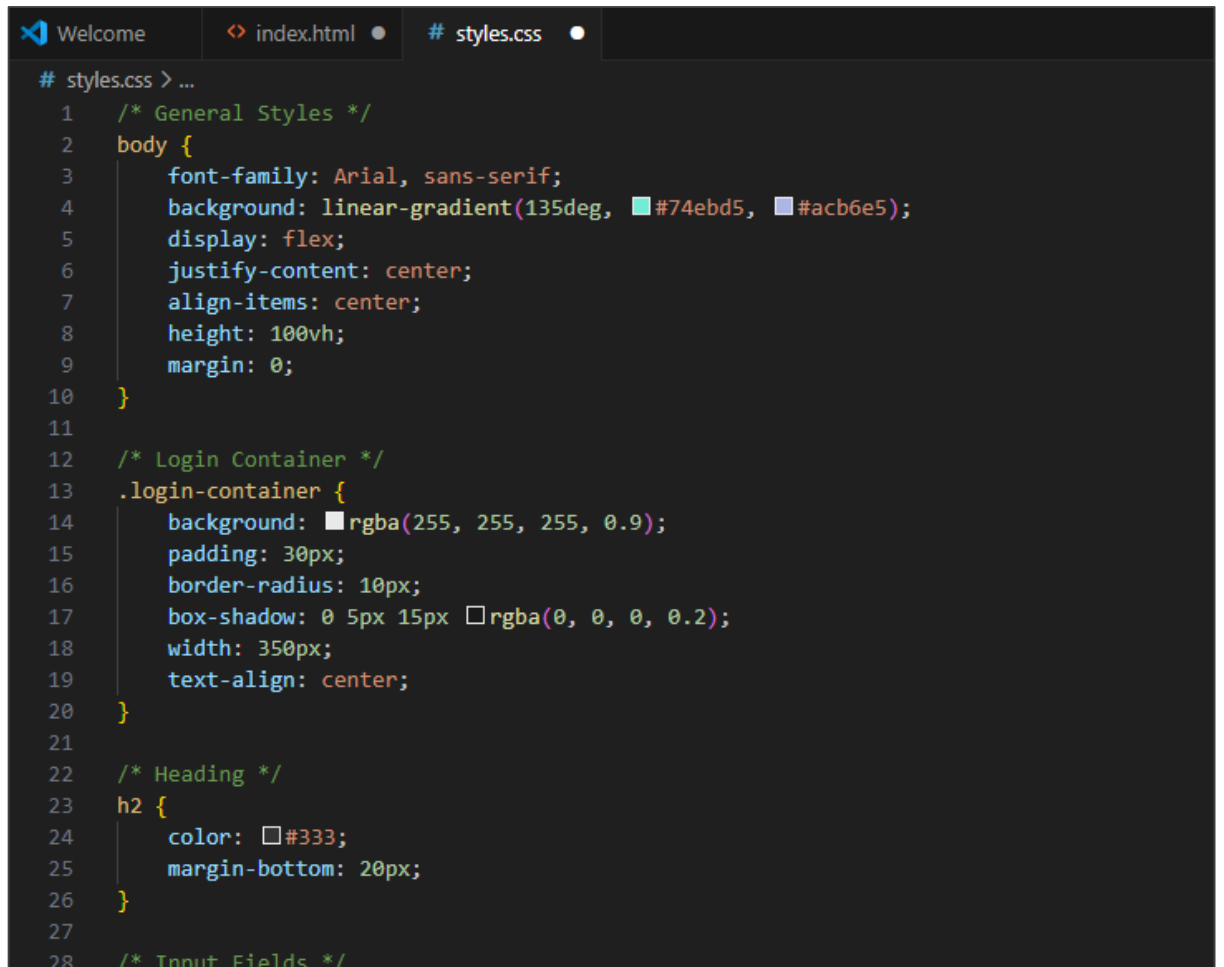
```
css Copy Edit

/* General Styles */
body {
  font-family: Arial, sans-serif;
  background: linear-gradient(135deg, #74ebd5, #acb6e5);
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
  margin: 0;
}

/* Login Container */
.login-container {
  Ask anything
```

2.6 Paste the copied code into the respective files as shown in the screenshots below:

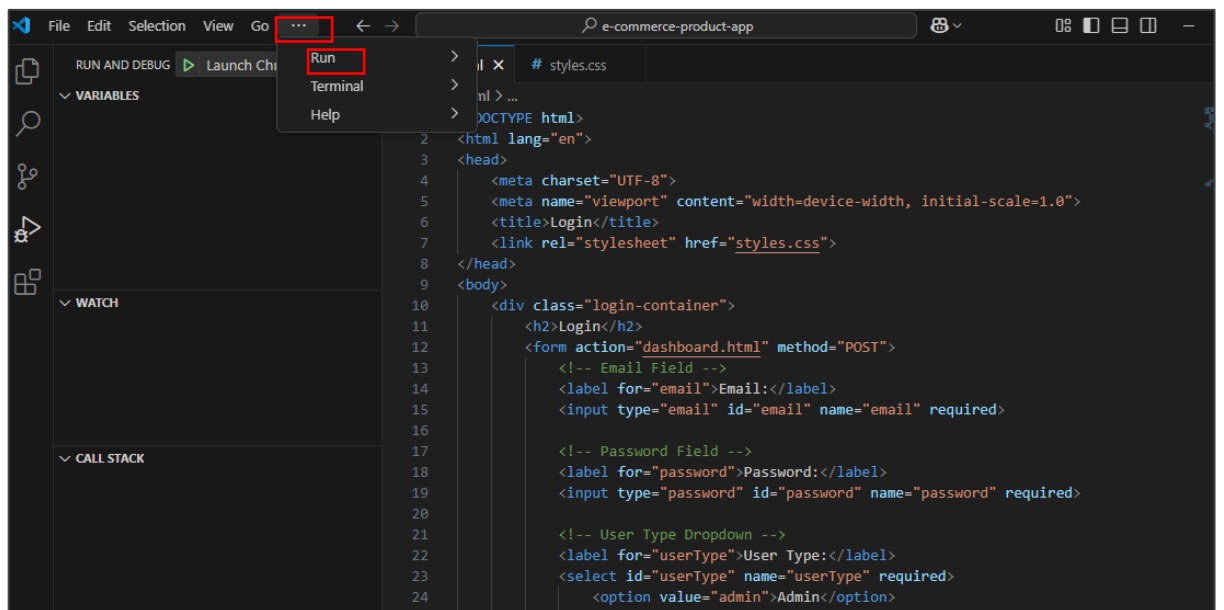
```
Welcome index.html
index.html > ...
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Login</title>
7   <link rel="stylesheet" href="styles.css">
8 </head>
9 <body>
10   <div class="login-container">
11     <h2>Login</h2>
12     <form action="dashboard.html" method="POST">
13       <!-- Email Field -->
14       <label for="email">Email:</label>
15       <input type="email" id="email" name="email" required>
16
17       <!-- Password Field -->
18       <label for="password">Password:</label>
19       <input type="password" id="password" name="password" required>
20
21       <!-- User Type Dropdown -->
22       <label for="userType">User Type:</label>
23       <select id="userType" name="userType" required>
24         <option value="admin">Admin</option>
25         <option value="user">User</option>
26       </select>
27
28       <!-- Buttons -->
29       <div class="button-container">
```



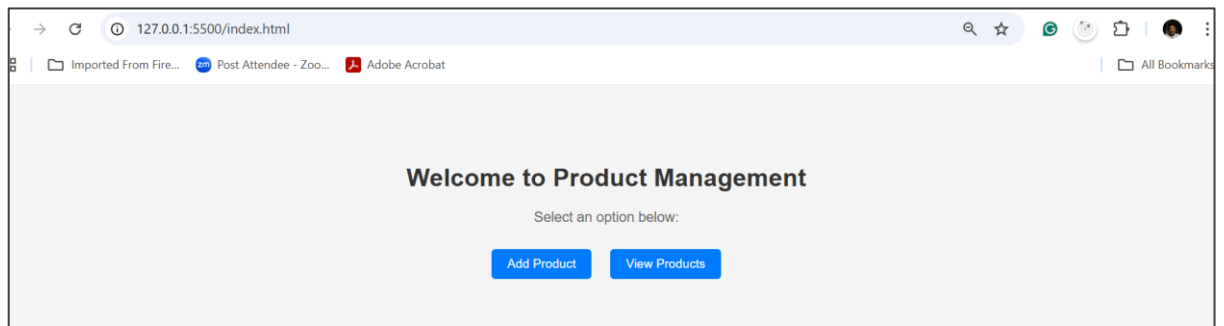
The image shows a Visual Studio Code editor window with the file explorer on the left and the editor area on the right. The editor is open to a file named `styles.css`. The code in the file is as follows:

```
# styles.css > ...
1  /* General Styles */
2  body {
3      font-family: Arial, sans-serif;
4      background: linear-gradient(135deg, #74ebd5, #acb6e5);
5      display: flex;
6      justify-content: center;
7      align-items: center;
8      height: 100vh;
9      margin: 0;
10 }
11
12 /* Login Container */
13 .login-container {
14     background: rgba(255, 255, 255, 0.9);
15     padding: 30px;
16     border-radius: 10px;
17     box-shadow: 0 5px 15px rgba(0, 0, 0, 0.2);
18     width: 350px;
19     text-align: center;
20 }
21
22 /* Heading */
23 h2 {
24     color: #333;
25     margin-bottom: 20px;
26 }
27
28 /* Input Fields */
```

2.7 Click on ... and then on **Run** to execute the index.html file

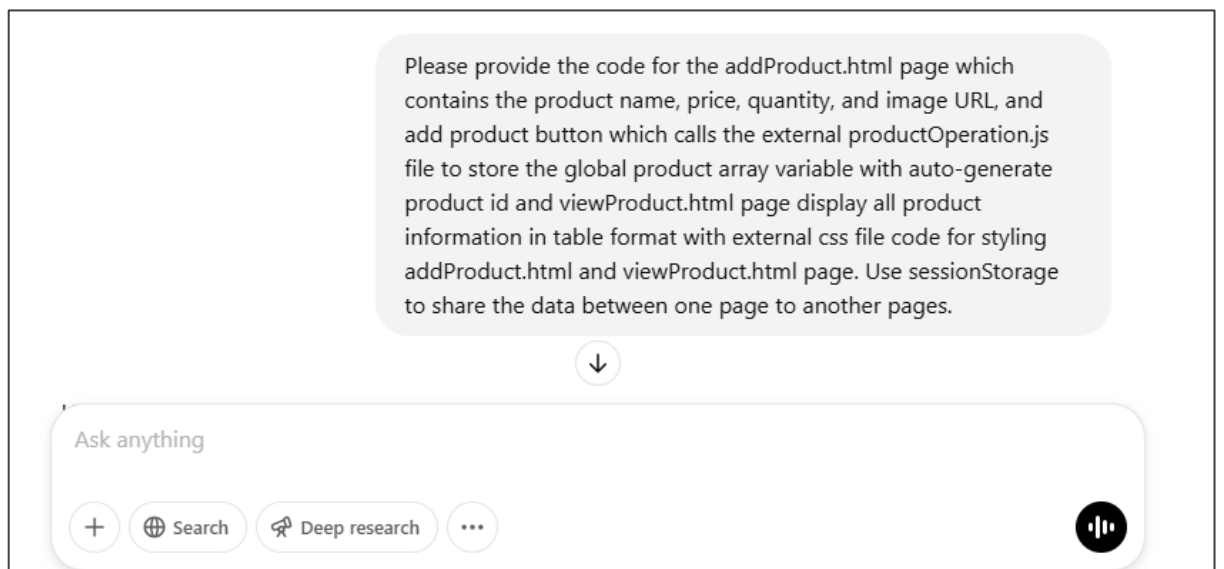


You will see the generated page in a browser.



- 2.8 Create the code for the addProduct.html page, which includes fields for the product name, price, quantity, and image URL, references an external CSS file, and stores the product details in a variable array:

Please provide the code for the addProduct.html page which contains the product name, price, quantity, and image URL, and add product button which calls the external productOperation.js file to store the global product array variable with auto-generate product id and viewProduct.html page display all product information in table format with external css file code for styling addProduct.html and viewProduct.html page. Use sessionStorage to share the data between one page to another pages.



2.9 Copy the generated code for the **addProduct.html**, **viewProduct.html**, and **productOperation.js** files separately

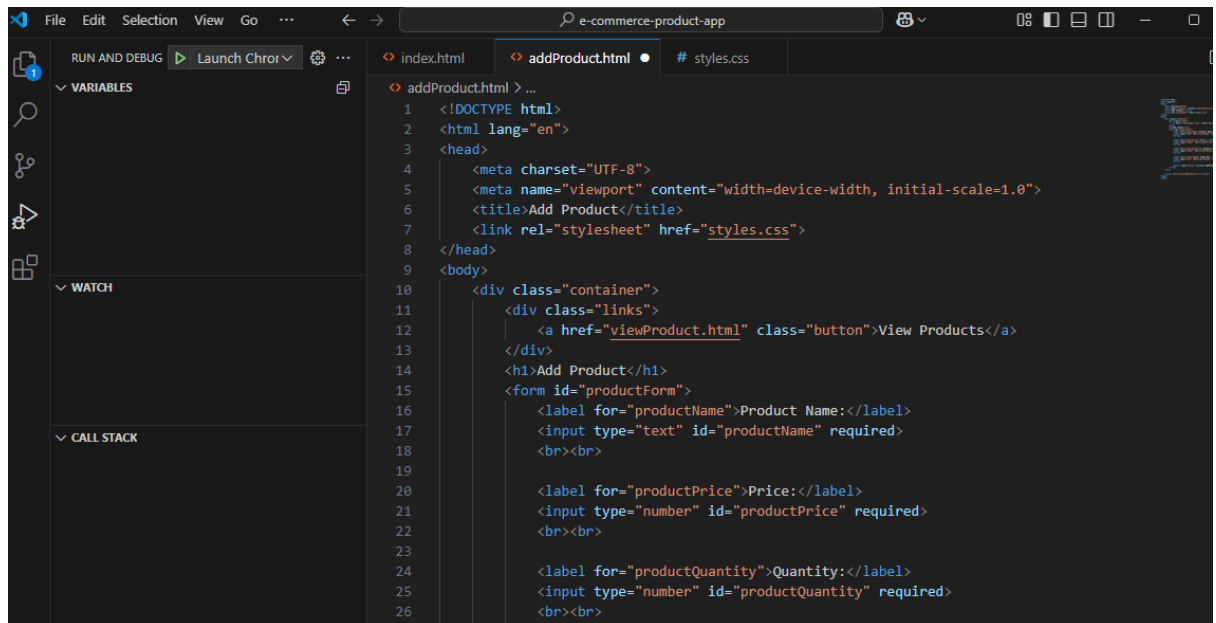
Here is the code for the `addProduct.html` and `viewProduct.html` pages, along with the necessary `productOperation.js` and CSS styling files:

1. `addProduct.html`

```
html

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Add Product</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <div class="container">
    <h1>Add Product</h1>
    <form id="productForm">
      <label for="productName">Product Name:</label>
      <input type="text" id="productName" required>
```

2.10 Paste the generated code into the respective files



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>View Products</title>
7   <link rel="stylesheet" href="styles.css">
8 </head>
9 <body>
10   <div class="container">
11     <div class="links">
12       <a href="addProduct.html" class="button">Add Product</a>
13     </div>
14     <h1>Product List</h1>
15     <table id="productTable">
16       <thead>
17         <tr>
18           <th>Product ID</th>
19           <th>Product Name</th>
20           <th>Price</th>
21           <th>Quantity</th>
22           <th>Image</th>
23         </tr>
24       </thead>
25       <tbody>
26         <!-- Product rows will be populated by JavaScript -->
27       </tbody>
28     </table>
29   </div>
```

```
1 // Load products from sessionStorage or initialize an empty array
2 let products = JSON.parse(sessionStorage.getItem('products')) || [];
3
4 // Function to add product
5 function addProduct() {
6   const productName = document.getElementById('productName').value;
7   const productPrice = parseFloat(document.getElementById('productPrice').value);
8   const productQuantity = parseInt(document.getElementById('productQuantity').value);
9   const productImage = document.getElementById('productImage').value;
10
11   // Generate product ID automatically (simple increment)
12   const productId = products.length + 1;
13
14   // Create new product object
15   const newProduct = {
16     id: productId,
17     name: productName,
18     price: productPrice,
19     quantity: productQuantity,
20     image: productImage
21   };
22
23   // Add product to global array
24   products.push(newProduct);
25
26   // Save products to local storage
27   sessionStorage.setItem('products', JSON.stringify(products));
28 }
```


Step 3: Link the pages and test the application

3.1 Open the index.html file and update the code with the following links:

`Add Product`

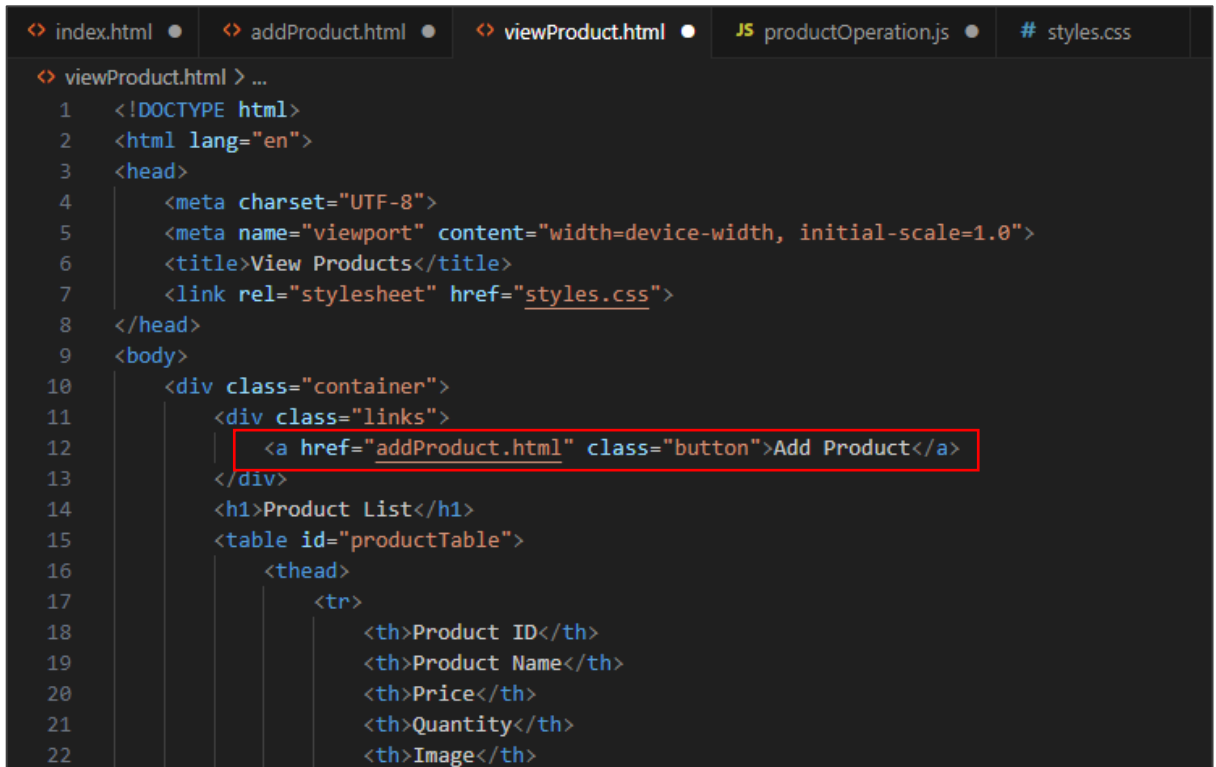
`View Product`

3.2 Open the addProduct.html file and update the code with the following link:

`View Products`

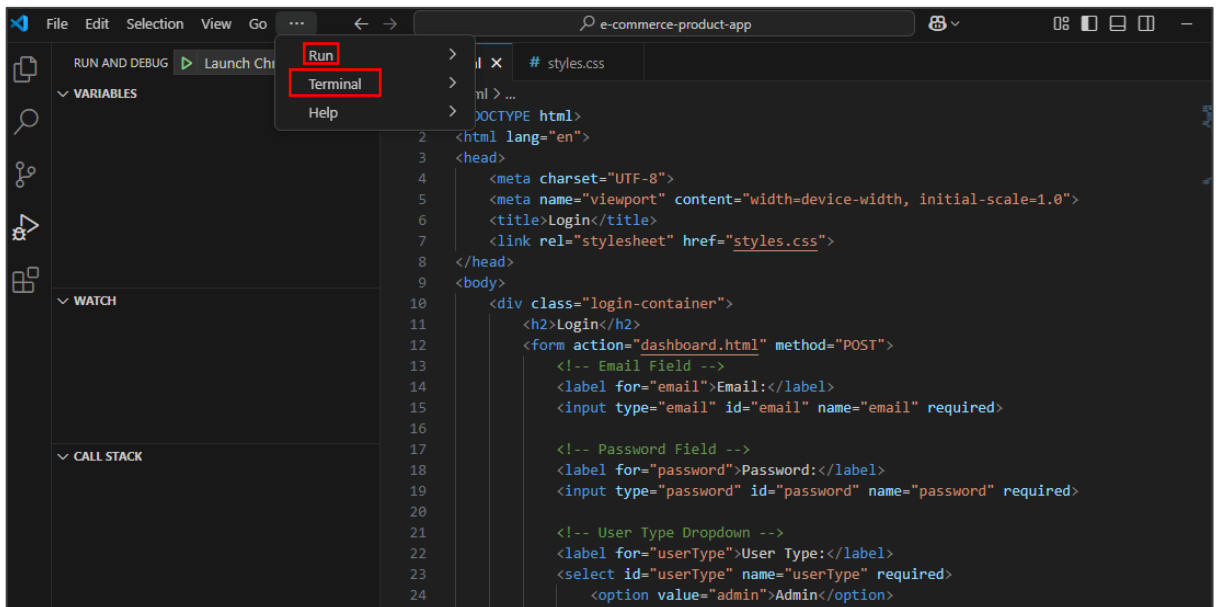
3.3 Open the viewProduct.html file and update the code with the following link:

`Add Products`



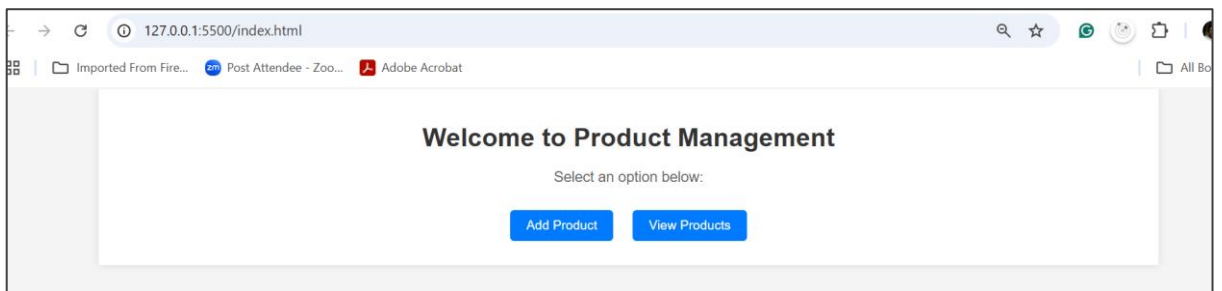
```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>View Products</title>
    <link rel="stylesheet" href="styles.css">
  </head>
  <body>
    <div class="container">
      <div class="links">
        <a href="addProduct.html" class="button">Add Product</a>
      </div>
      <h1>Product List</h1>
      <table id="productTable">
        <thead>
          <tr>
            <th>Product ID</th>
            <th>Product Name</th>
            <th>Price</th>
            <th>Quantity</th>
            <th>Image</th>
          </tr>
        </thead>
      </table>
    </div>
  </body>
</html>
```

3.4 Run the index.html page to verify if the navigation and functionality work as intended



```
File Edit Selection View Go ...
e-commerce-product-app
# styles.css
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login</title>
    <link rel="stylesheet" href="styles.css">
  </head>
  <body>
    <div class="login-container">
      <h2>Login</h2>
      <form action="dashboard.html" method="POST">
        <!-- Email Field -->
        <label for="email">Email:</label>
        <input type="email" id="email" name="email" required>
        <!-- Password Field -->
        <label for="password">Password:</label>
        <input type="password" id="password" name="password" required>
        <!-- User Type Dropdown -->
        <label for="userType">User Type:</label>
        <select id="userType" name="userType" required>
          <option value="admin">Admin</option>
          <option value="user">User</option>
        </select>
      </form>
    </div>
  </body>
</html>
```

You can see the generated page in a browser and test it.



[View Products](#)

Add Product

Product Name:

Laptop

Price:

87000

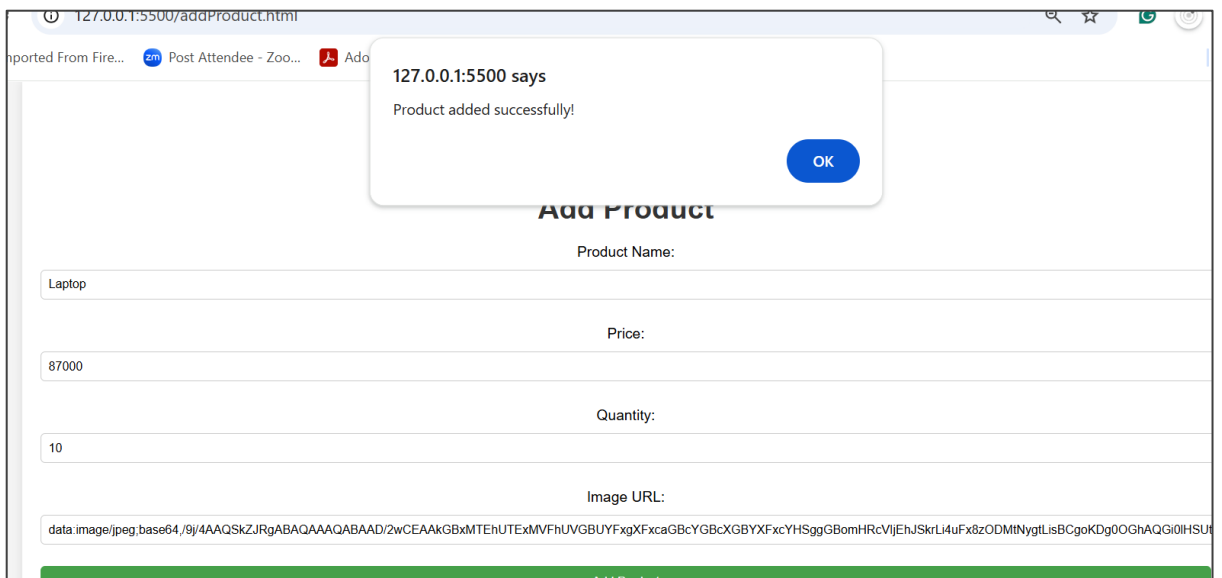
Quantity:

10

Image URL:

data:image/jpeg;base64,/9j/4AAQSkZJRgABAQAAQABAAQ/2wCEAAkGBxMTEhUTEhMVhUUVGBUUFxgXFcaGBcYGBcXGBYXFcYHSggGBomHRCvIjEhJSkrLi4uFx8zODMlNygtLisBCgoKDg0OGhAQGI0HSU



[Add Product](#)



You can also view the product details.

Add Product

Product List

Product ID	Product Name	Price	Quantity	Image
1	Laptop	\$87000.00	10	
2	pen drive	\$2600.00	50	

By following these steps, you have successfully created an e-commerce product module using JavaScript, enabling users to add, store, and view product details with seamless navigation and a well-structured interface.