Project files

Coding for E commerce shoe store :

1. \*\*Front-end (HTML, CSS, JavaScript):\*\*

You can use popular front-end frameworks like React, Angular, or Vue.js. Here's a simplified example using HTML, CSS, and JavaScript:

\*\*index.html:\*\*

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Shoe Store</title>

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

</head>

<body>

<header>

<h1>Shoe Store</h1>

</header>

<main id="app">

<!-- Product listing and details will be dynamically generated here -->

</main>

<script src="app.js"></script>

</body>

</html>

```

\*\*styles.css:\*\*

```css

/\* Your CSS styling here \*/

```

\*\*app.js:\*\*

```javascript

const app = document.getElementById('app');

// Fetch and display products

function fetchProducts() {

// Fetch products from the server (AJAX/fetch)

const products = [

{ id: 1, name: 'Running Shoe', price: 99.99, imageUrl: 'shoe1.jpg' },

// ...more products

];

// Display products

const productsHtml = products.map(product => `

<div class="product">

<img src="${product.imageUrl}" alt="${product.name}">

<h2>${product.name}</h2>

<p>$${product.price}</p>

<button onclick="addToCart(${product.id})">Add to Cart</button>

</div>

`).join('');

app.innerHTML = productsHtml;

}

// Add product to cart

function addToCart(productId) {

// Implement cart functionality (add to cart array, update UI, etc.)

}

// Load products when the page loads

fetchProducts();

```

2. \*\*Back-end (Server-side logic and API):\*\*

You'd need a back-end to handle product data, user authentication, shopping cart, and more. Node.js with Express is a common choice. Here's a simplified example:

```javascript

const express = require('express');

const app = express();

const PORT = 3000;

// Mock product data

const products = [

{ id: 1, name: 'Running Shoe', price: 99.99, imageUrl: 'shoe1.jpg' },

// ...more products

];

// API endpoints

app.get('/api/products', (req, res) => {

res.json(products);

});

app.listen(PORT, () => {

console.log(`Server is running on port ${PORT}`);

});

```

3. \*\*Database:\*\*

For a full-fledged ecommerce store, you'd need a database to store product information, user data, orders, etc. Popular databases include MySQL, PostgreSQL, or MongoDB.

4. \*\*User Authentication and Authorization:\*\*

Implement user registration, login, and session management to secure your application.

5. \*\*Shopping Cart and Checkout:\*\*

Create logic to manage the user's shopping cart and process orders during checkout.

6. \*\*Payment Gateway Integration:\*\*

Integrate a payment gateway (e.g., Stripe, PayPal) to handle payments securely.