**Name : Kalashri Patil**

**Employee Id : 47294**

**Architecture and Design**

**Frontend (ASP.NET Core MVC):**

MVC (Model-View-Controller) pattern used for structuring the frontend.

Views: Render HTML/CSS/JavaScript to display UI elements.

Controllers: Handle user requests, interact with the backend, and manage the application flow.

Models: Represent data entities and interact with the backend for data retrieval and manipulation.

Utilizes Razor syntax for dynamic content generation.

Implements responsive design principles for cross-device compatibility.

**Backend (ASP.NET Core Web API):**

Web API architecture to provide RESTful endpoints for communication between frontend and backend.

Controllers: Handle HTTP requests from the frontend and execute appropriate actions.

Services: Implement business logic, such as processing orders, managing products, and handling user authentication.

DTOs (Data Transfer Objects): Used for transferring data between frontend and backend, ensuring loose coupling.

Middleware: Implement cross-cutting concerns like logging, authentication, and exception handling.

Utilizes Dependency Injection for managing component dependencies and promoting testability.

**Database (SQL Server):**

Relational database management system used for storing and managing e-commerce data.

Utilizes SQL Server to create and manage databases, tables, indexes, and relationships.

Tables designed to represent entities like users, products, orders, and transactions.

Implements normalization to reduce redundancy and improve data integrity.

Utilizes SQL queries and stored procedures for data retrieval, insertion, update, and deletion operations.

Implements appropriate indexing strategies for optimizing query performance.

**Security:**

Implements authentication and authorization mechanisms to ensure secure access to resources.

Utilizes ASP.NET Core Identity for user authentication and role-based access control.

Incorporates HTTPS protocol for secure communication between client and server.

Implements measures like input validation, parameterized queries, and data encryption to prevent

security vulnerabilities like SQL injection and cross-site scripting (XSS).

**Scalability and Performance:**

Designs the application with scalability in mind to handle increasing user loads.

Implements caching mechanisms to reduce database load and improve performance.

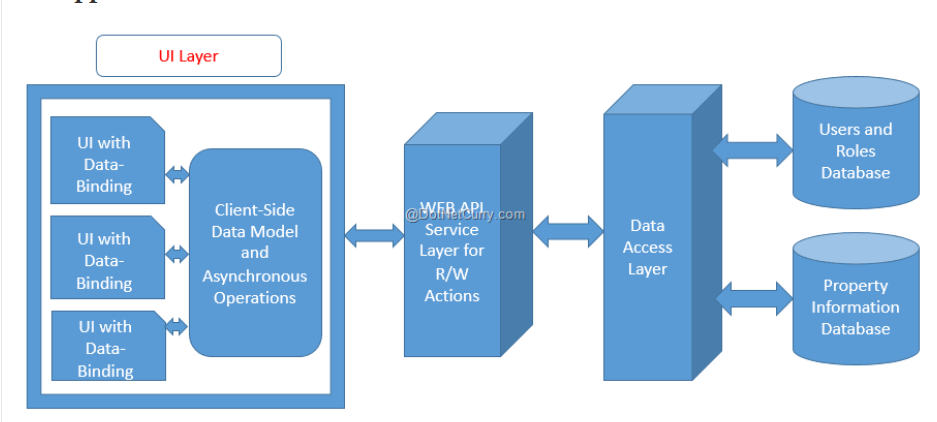
Utilizes asynchronous programming techniques to optimize resource utilization and responsiveness.

Implements load balancing and horizontal scaling strategies to distribute incoming traffic across multiple server instances.

**Testing**

Implements unit tests, integration tests, and end-to-end tests to ensure the reliability and correctness of the application.

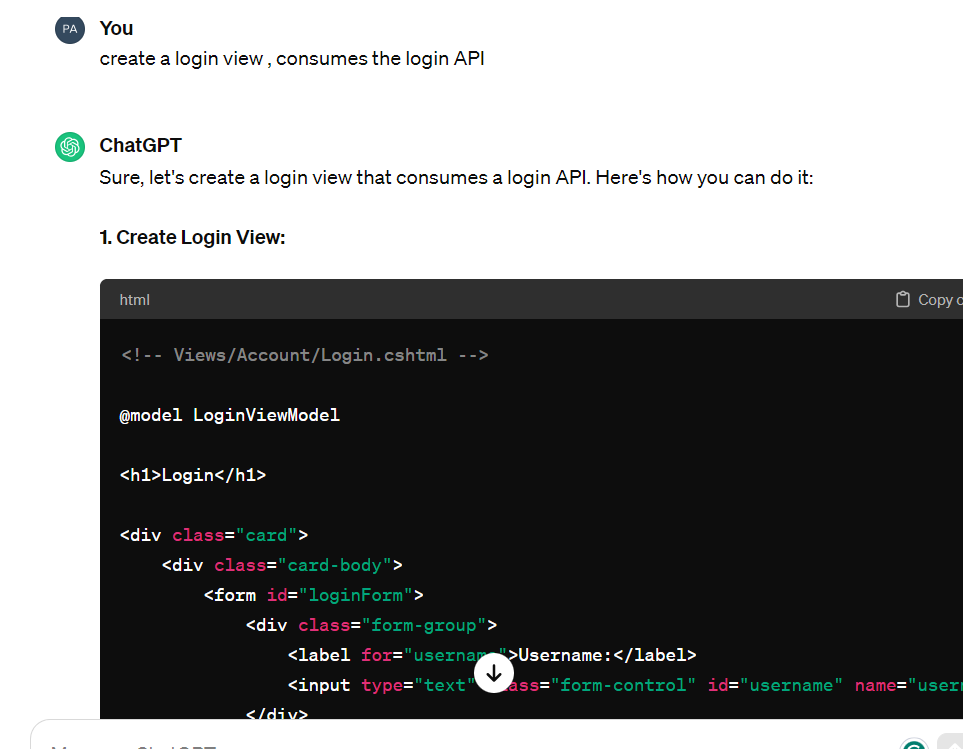
**Overall architecture:**

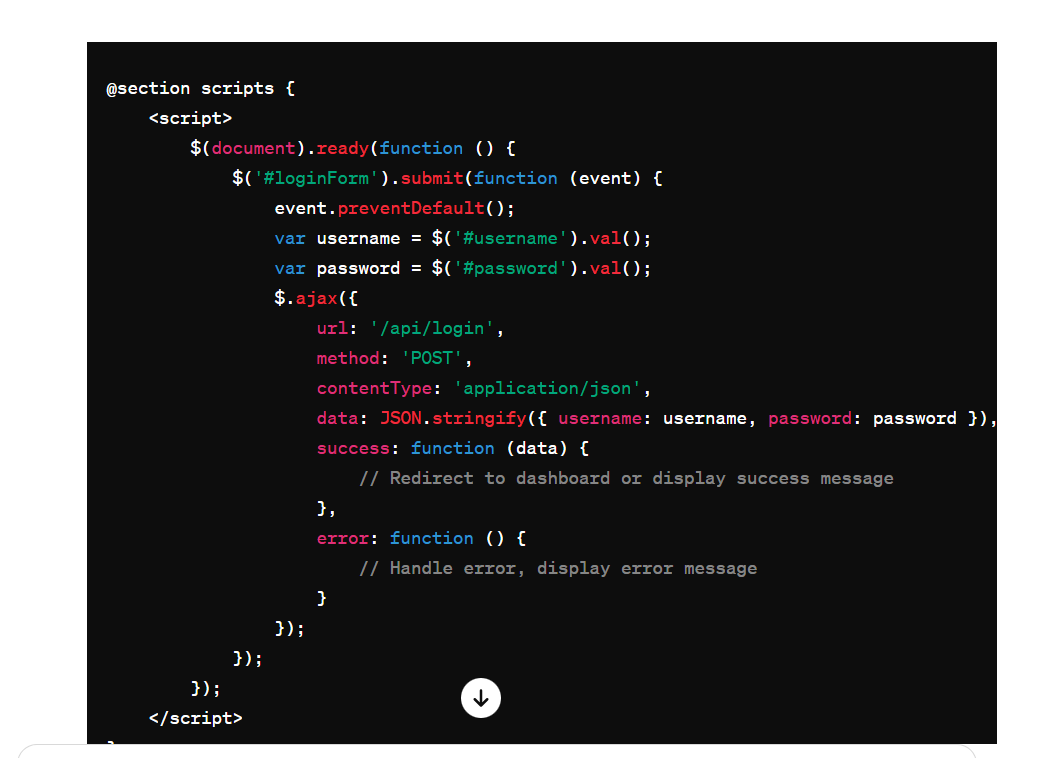


**ChatGPT prompts screenshots:**

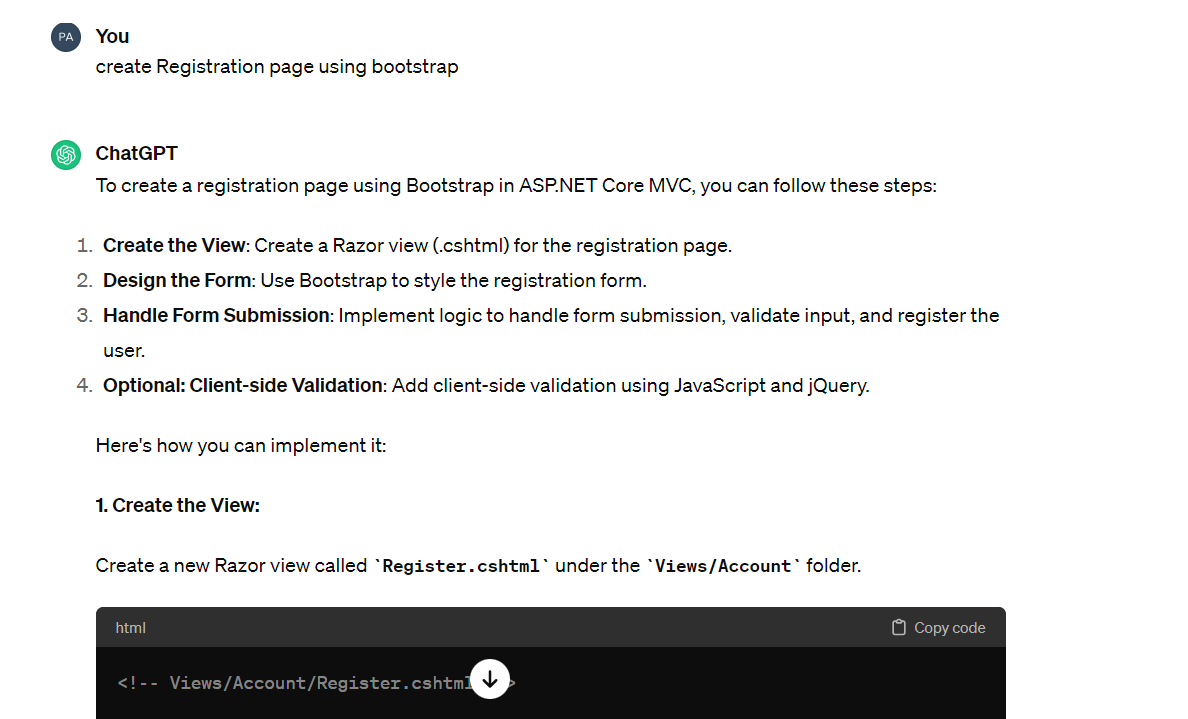
1. **For UI**

**Login view**

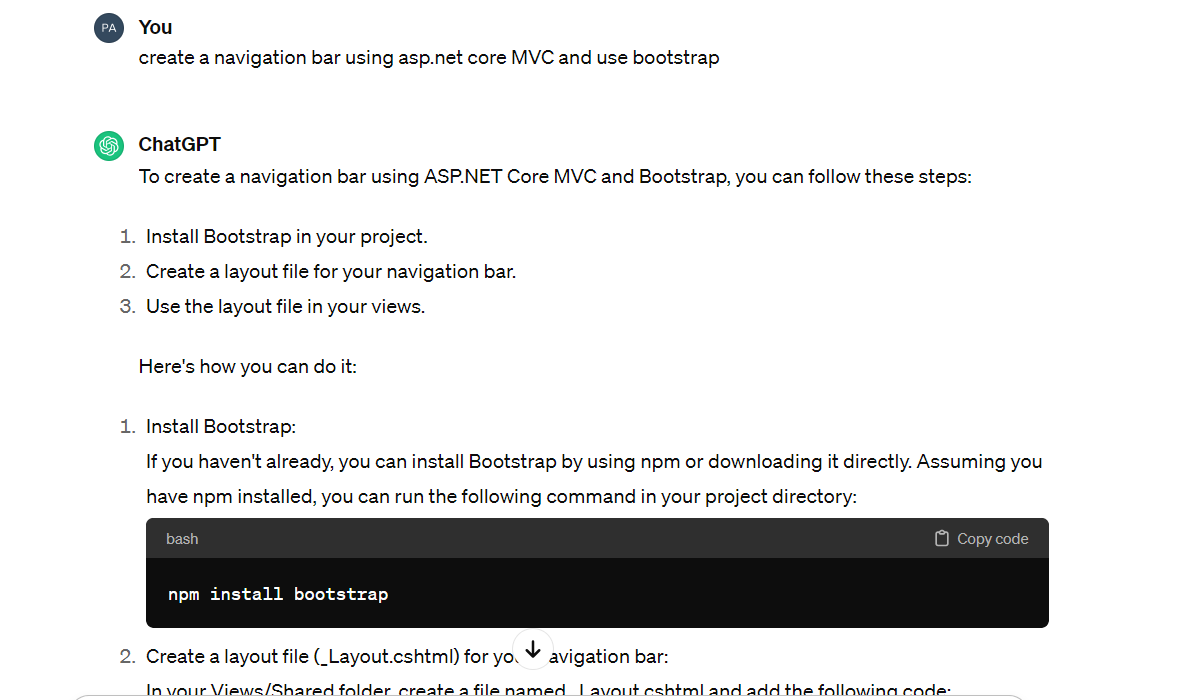
****

****

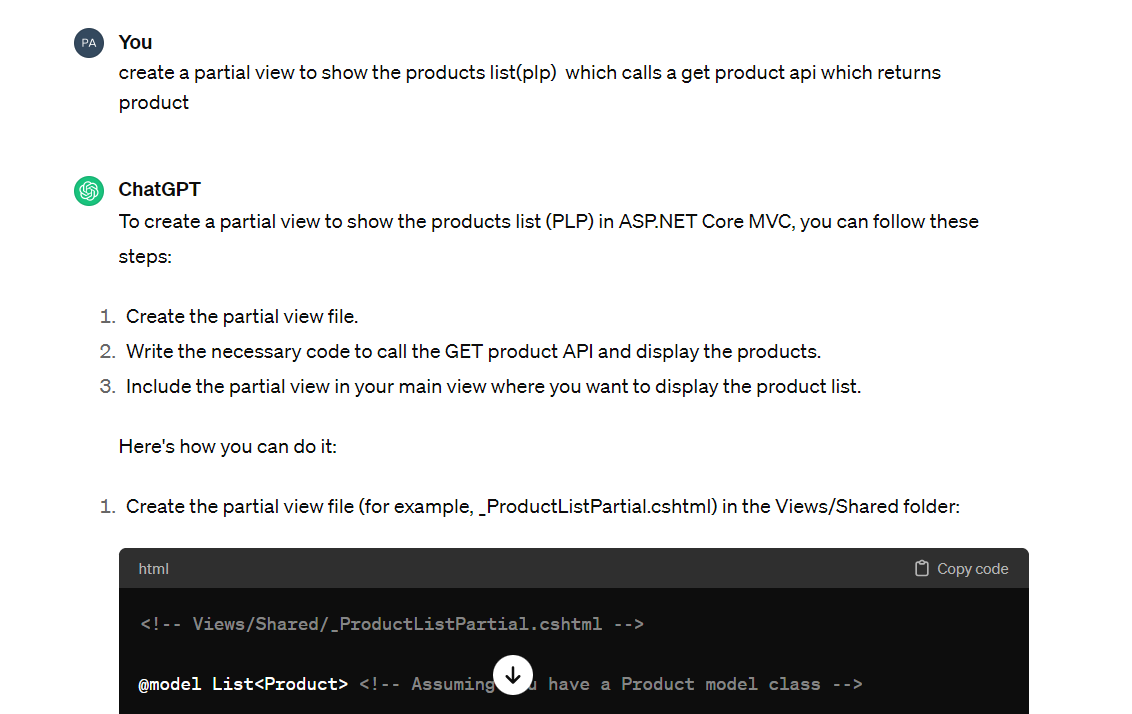
**Registration view :**

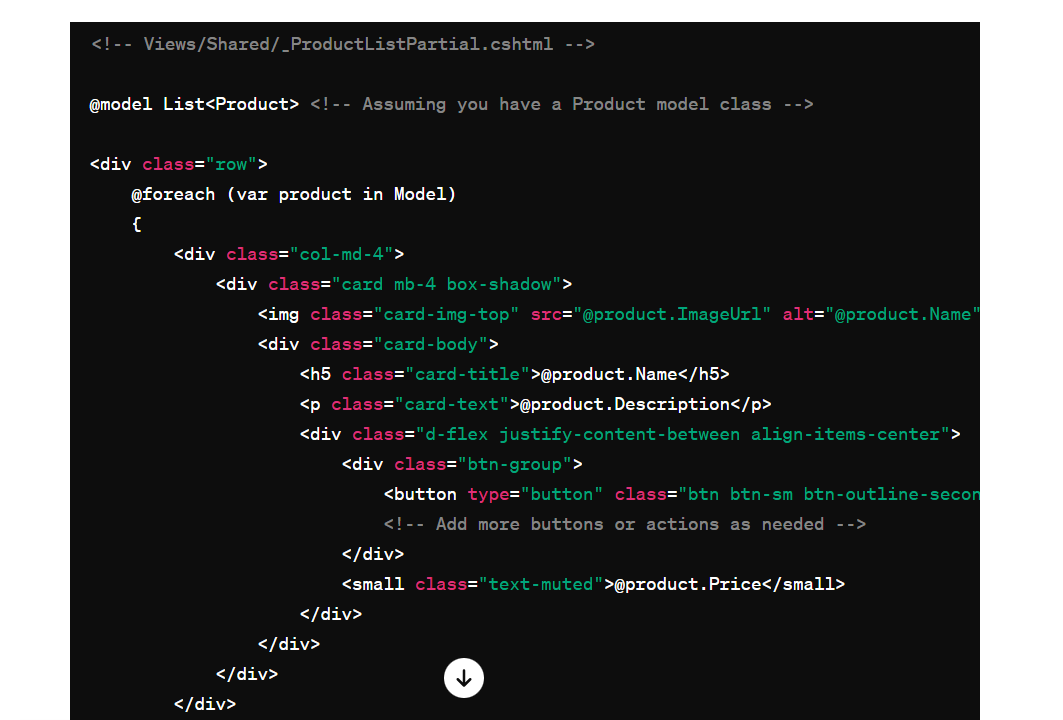
****

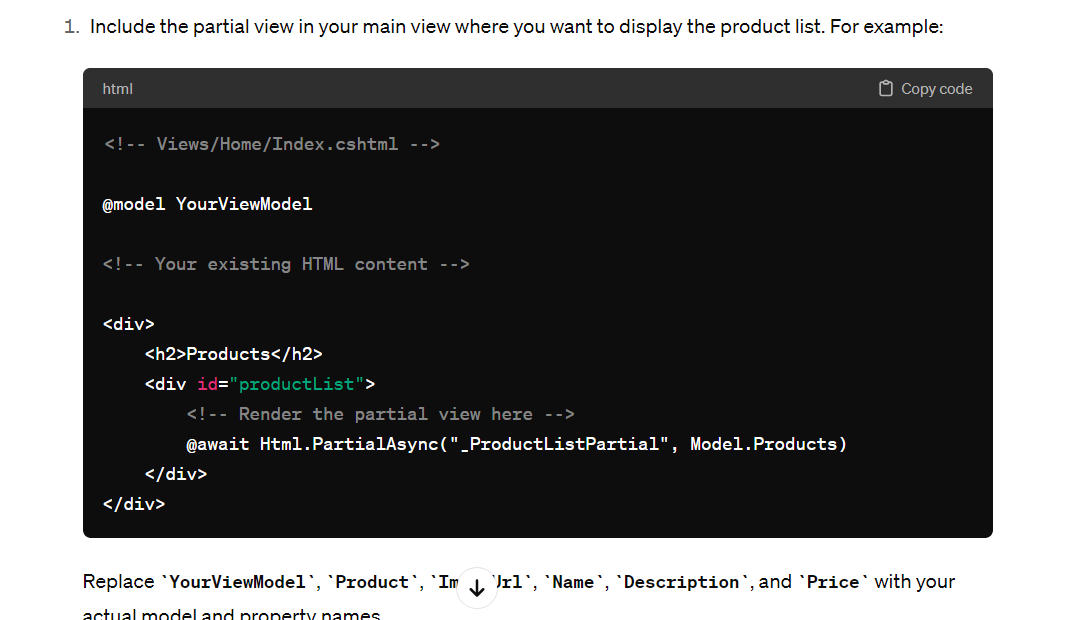
**Navbar :**

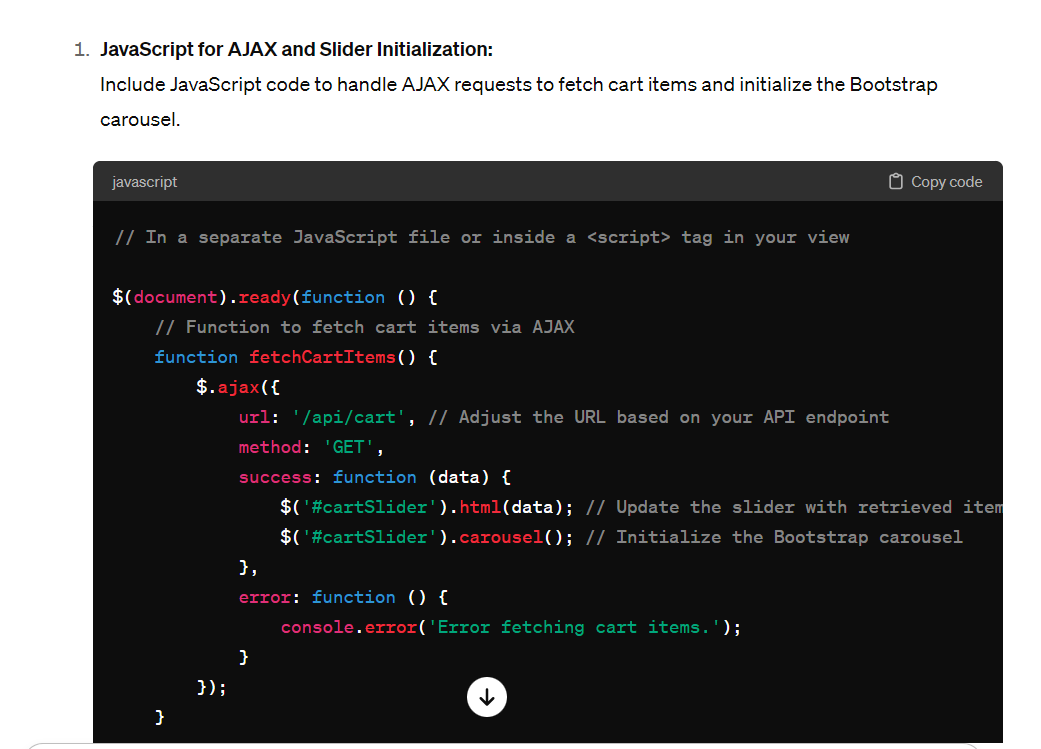
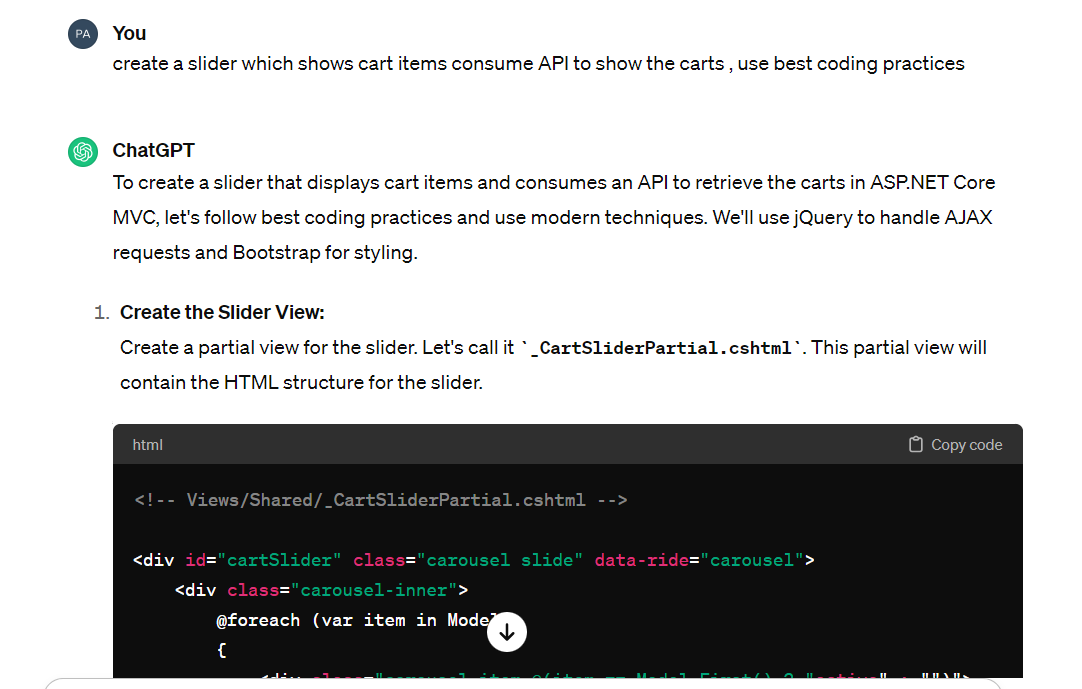
****

**Products View page**

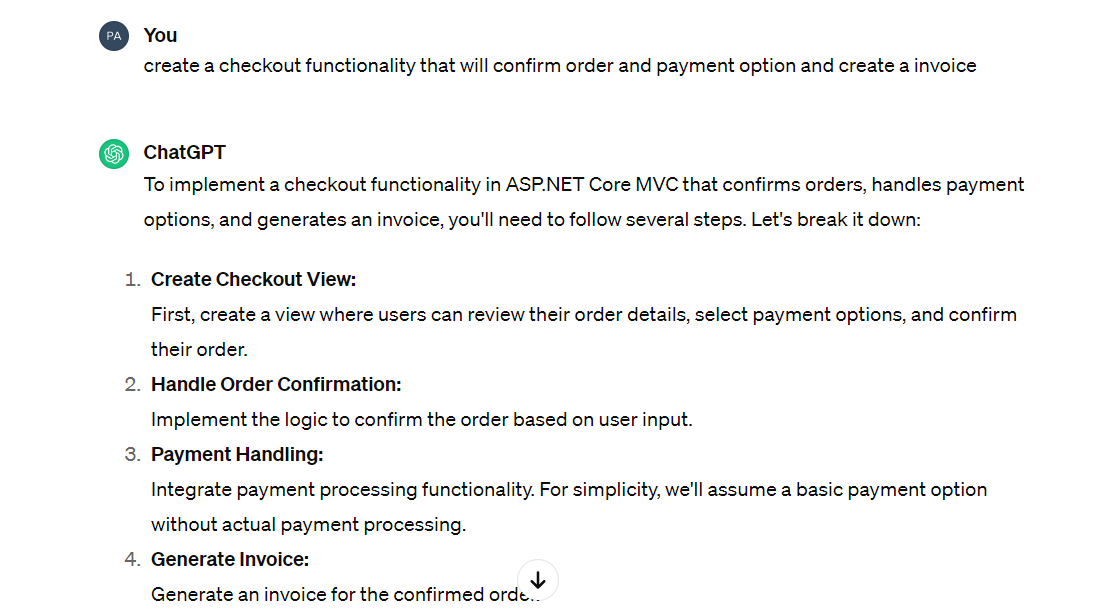
****

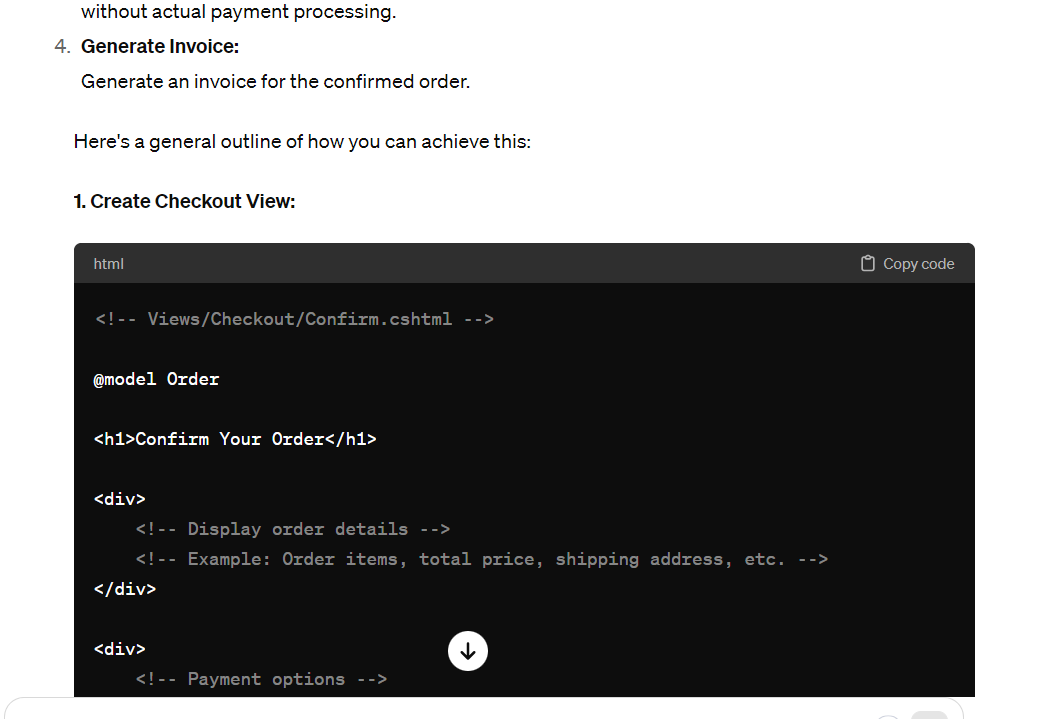


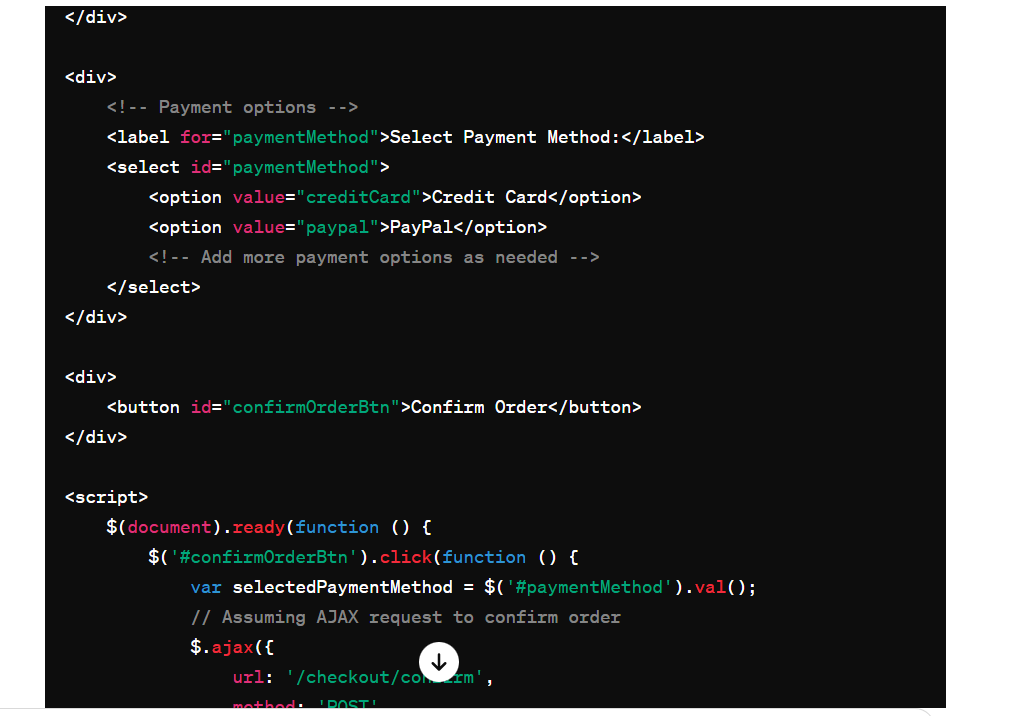


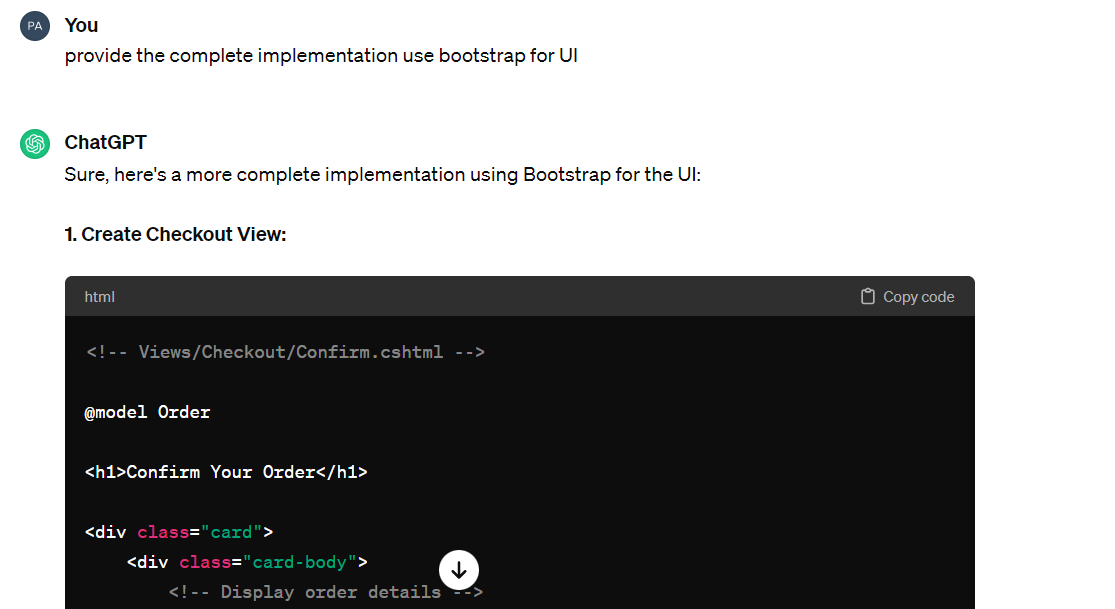
****

**Checkout view**

****

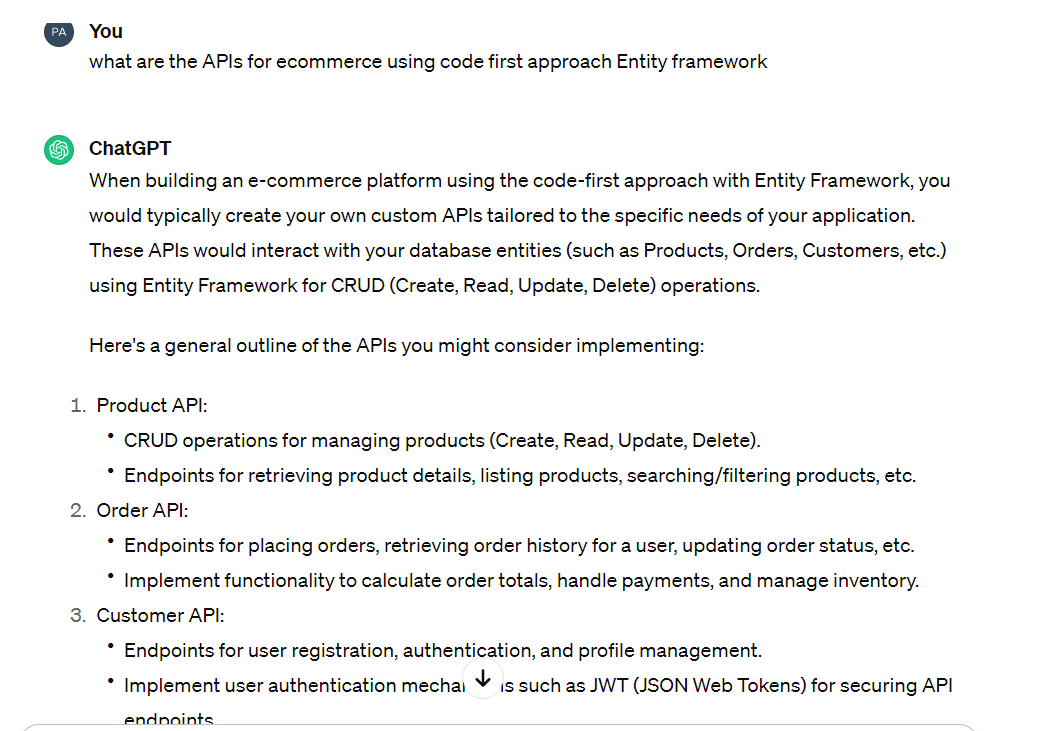


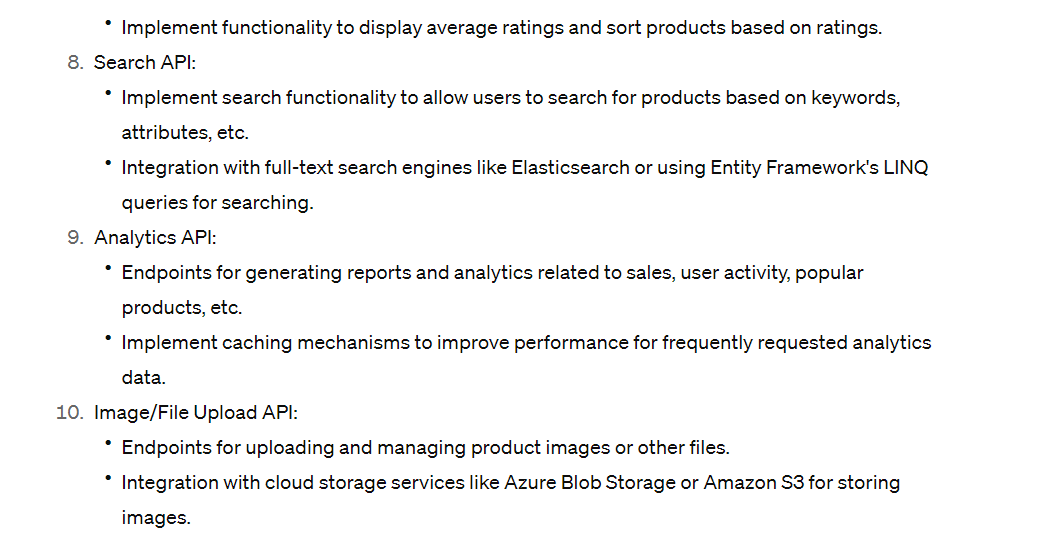


****

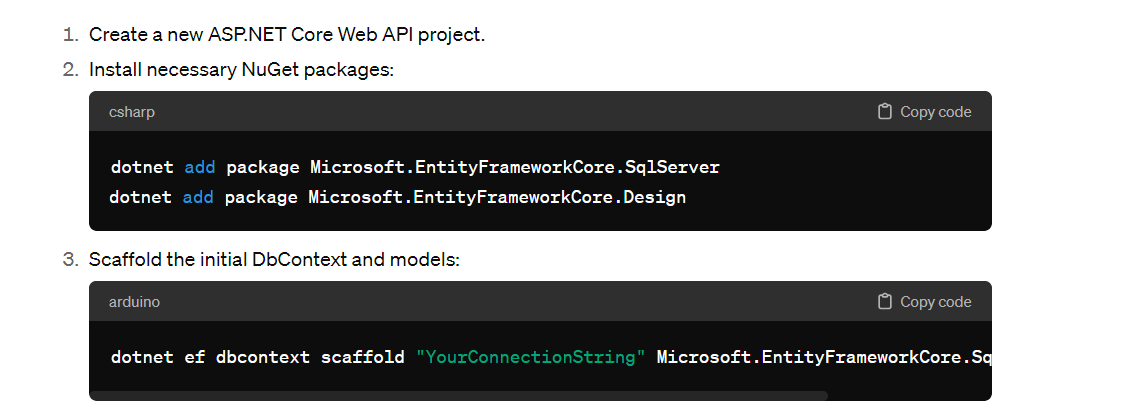
1. **For APIs**

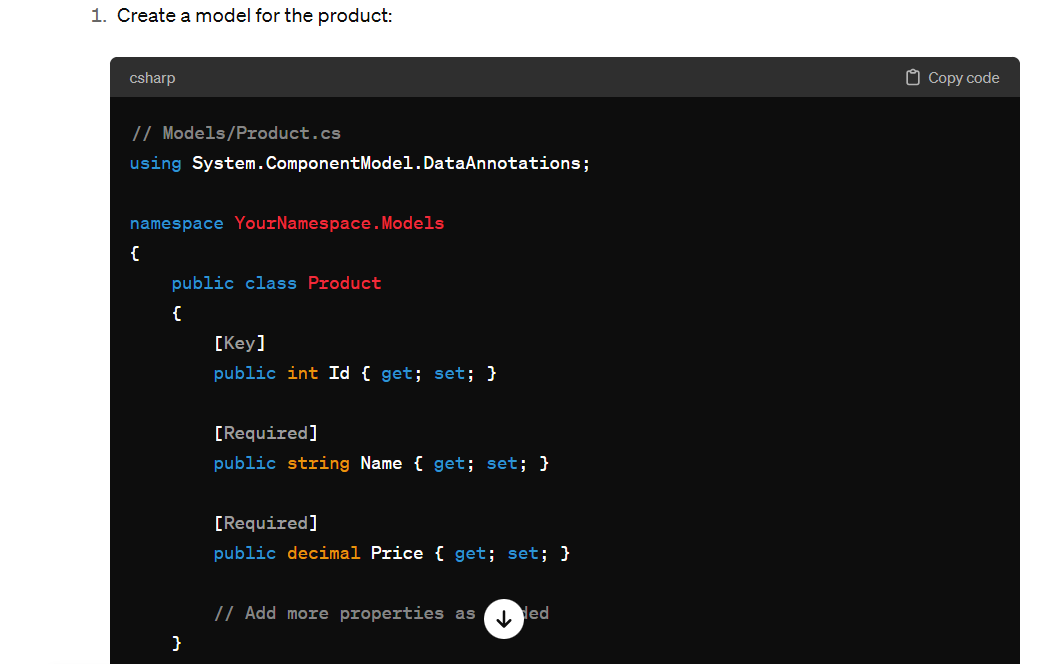
**API suggestion by chatGPT**

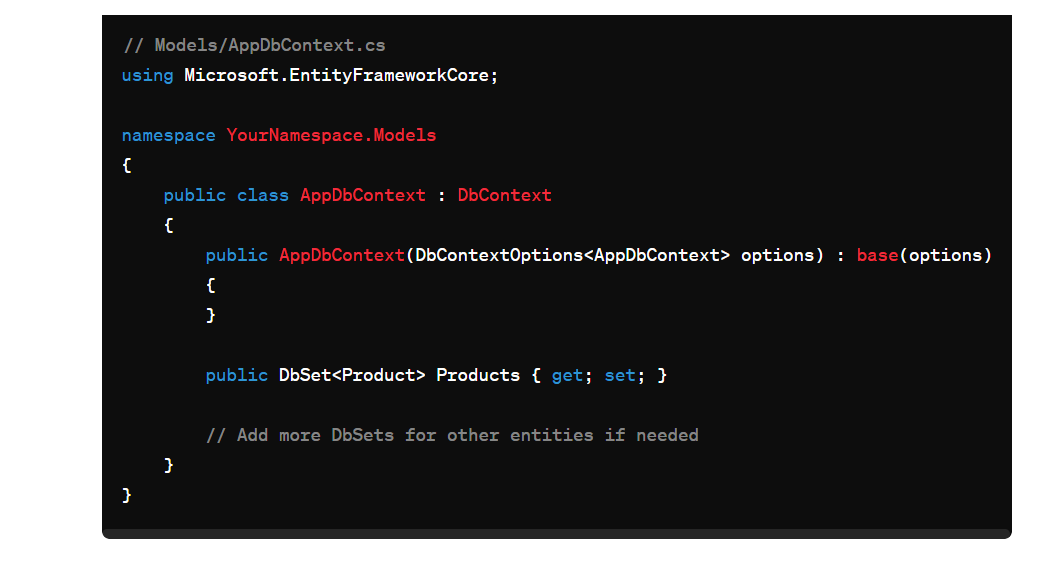
****

****

**Configuring asp.net core web api application**

****

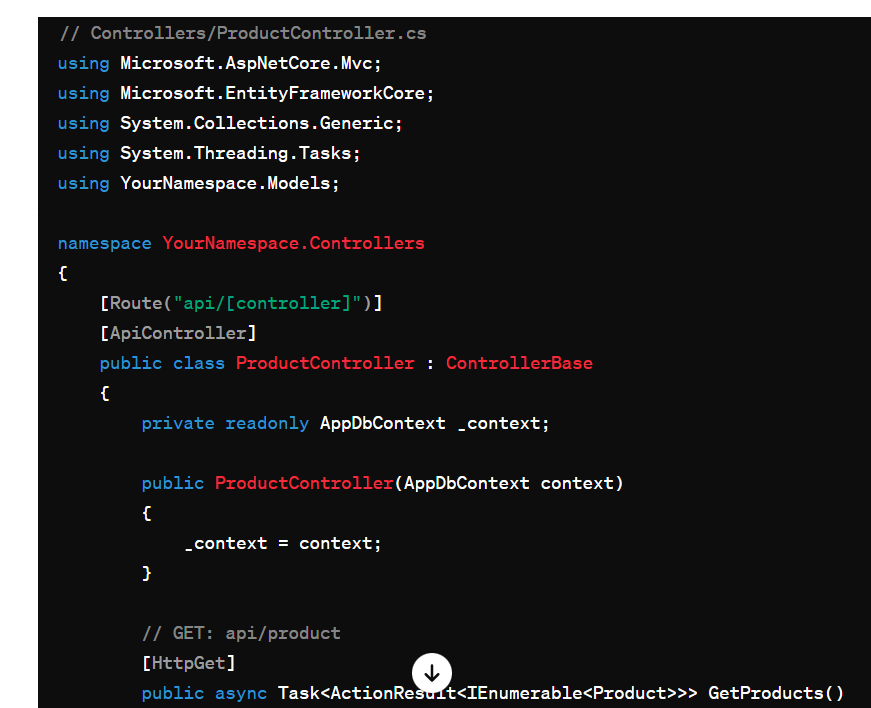
****

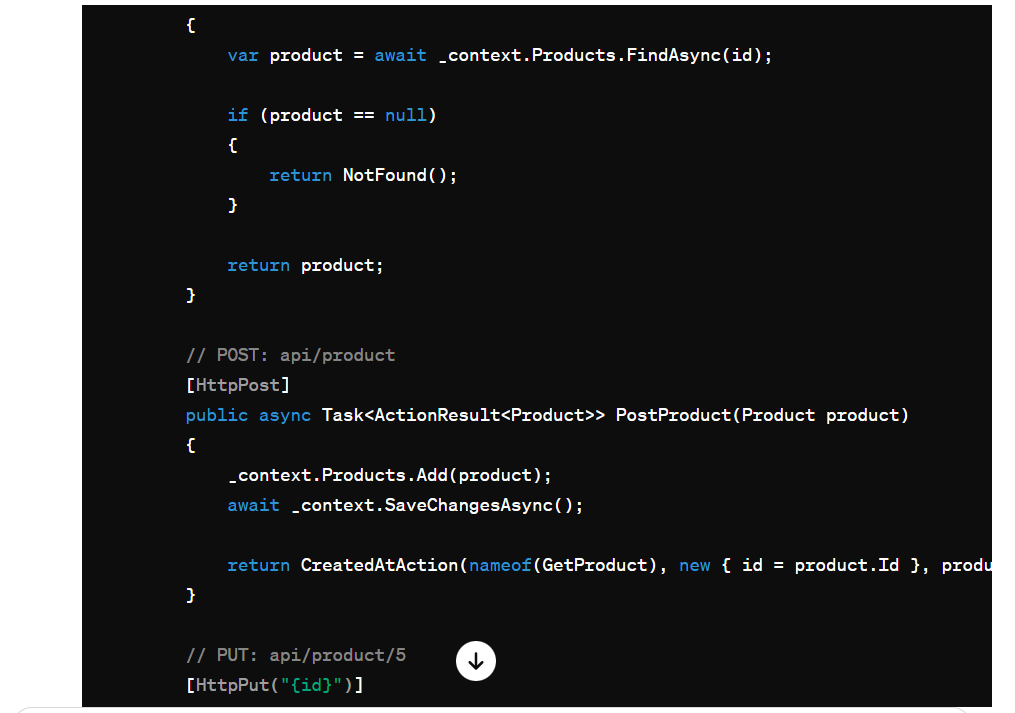
****

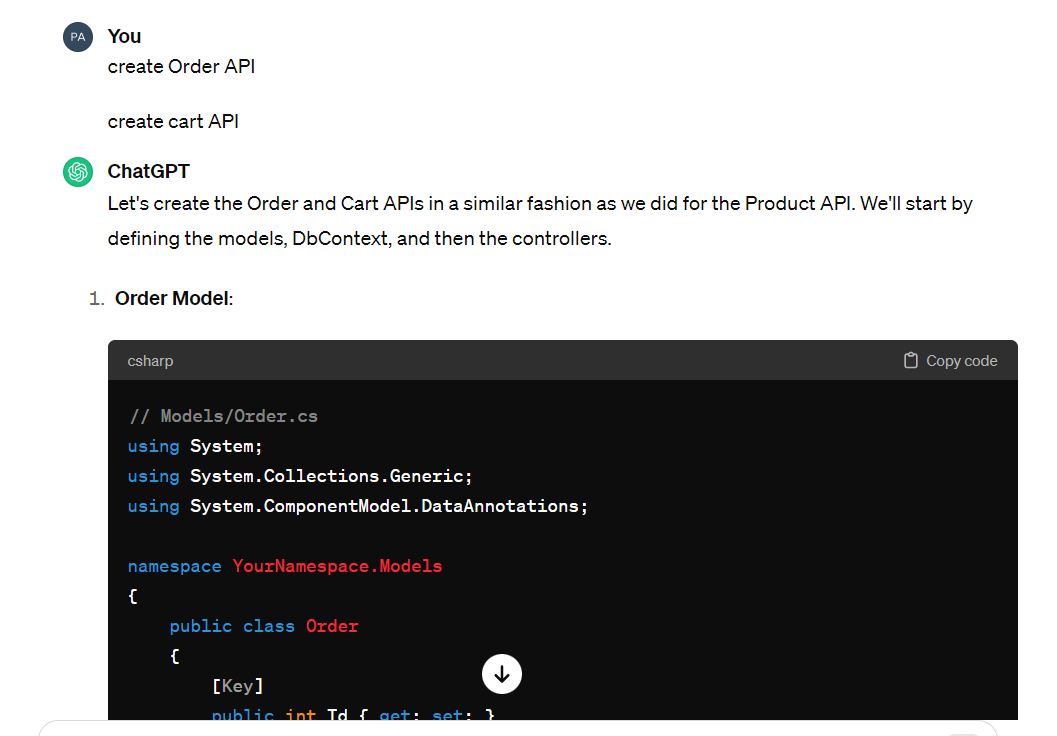
****

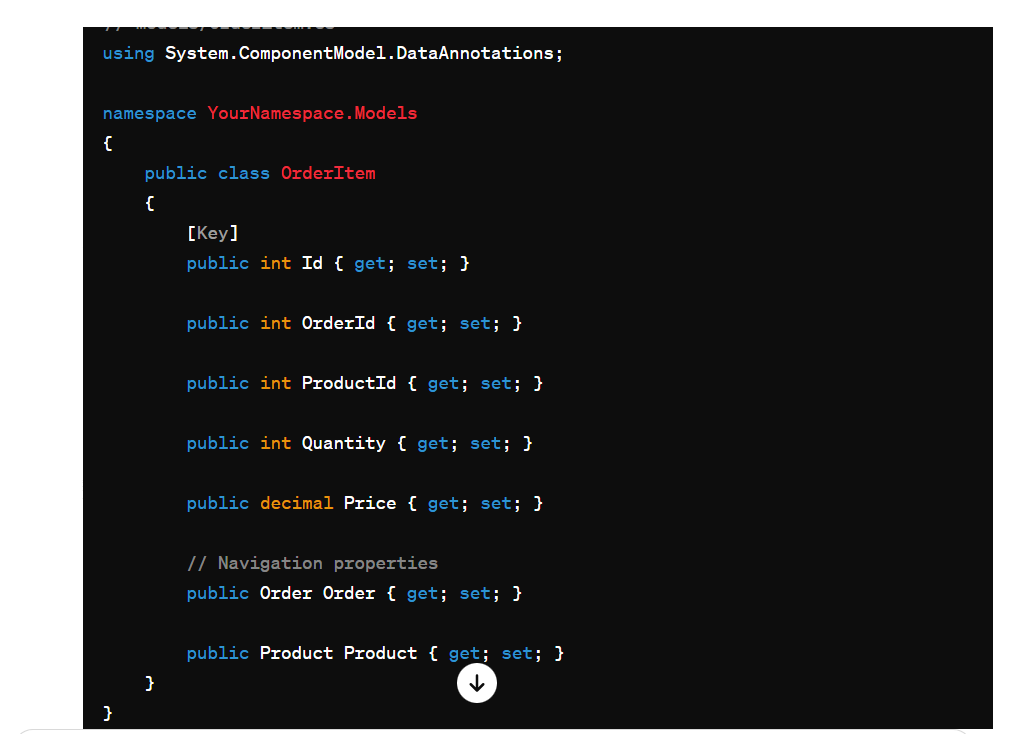
**Prompts and response for creating each API**

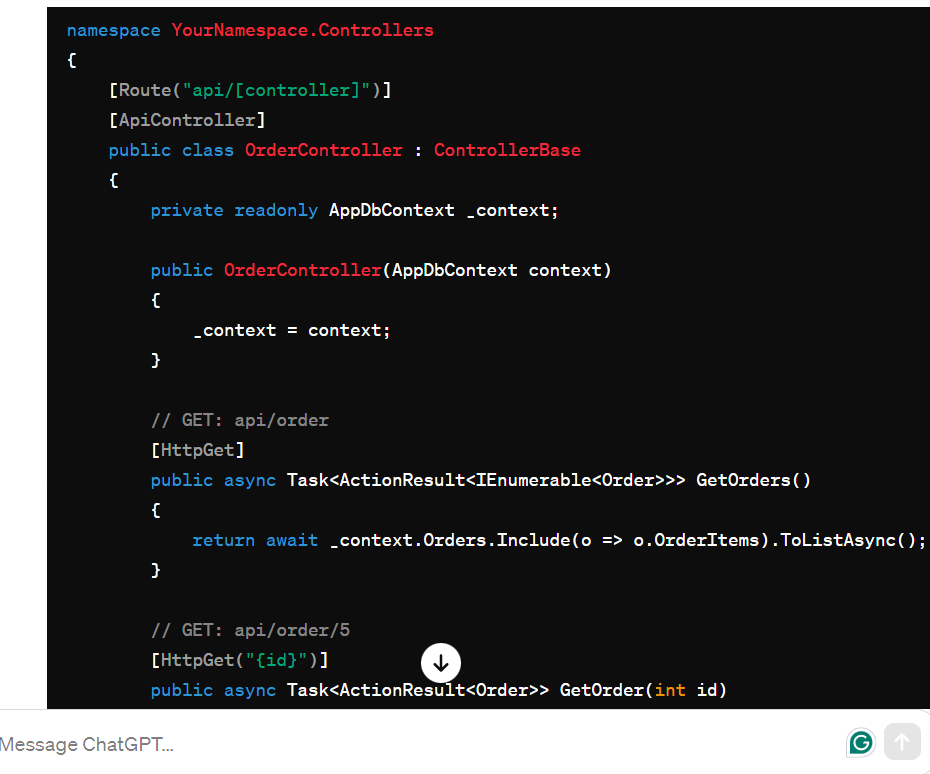
**Product API**



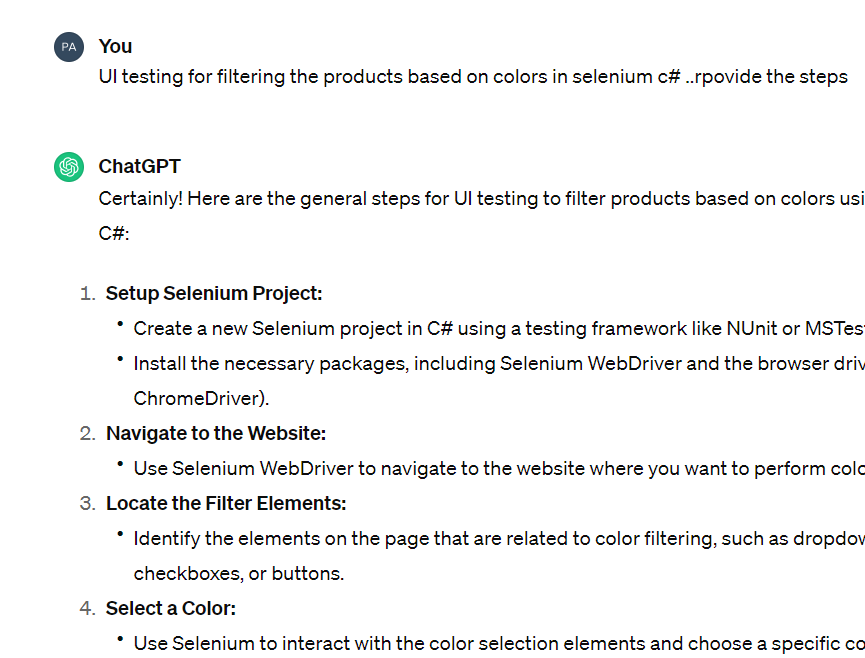


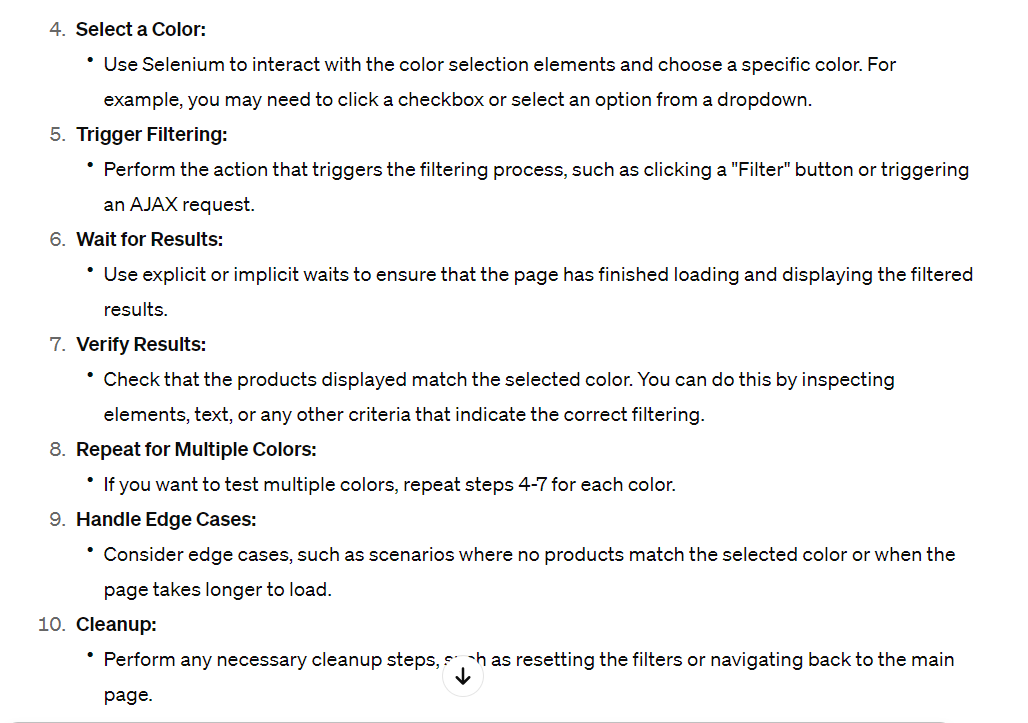


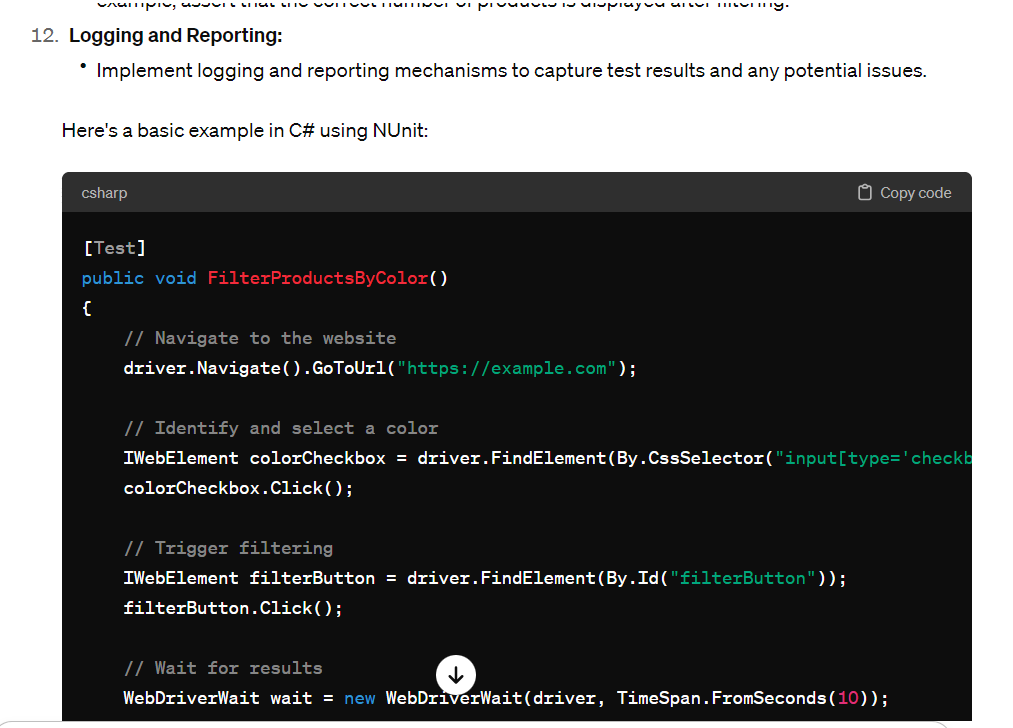




**UI Testing using selenium:**



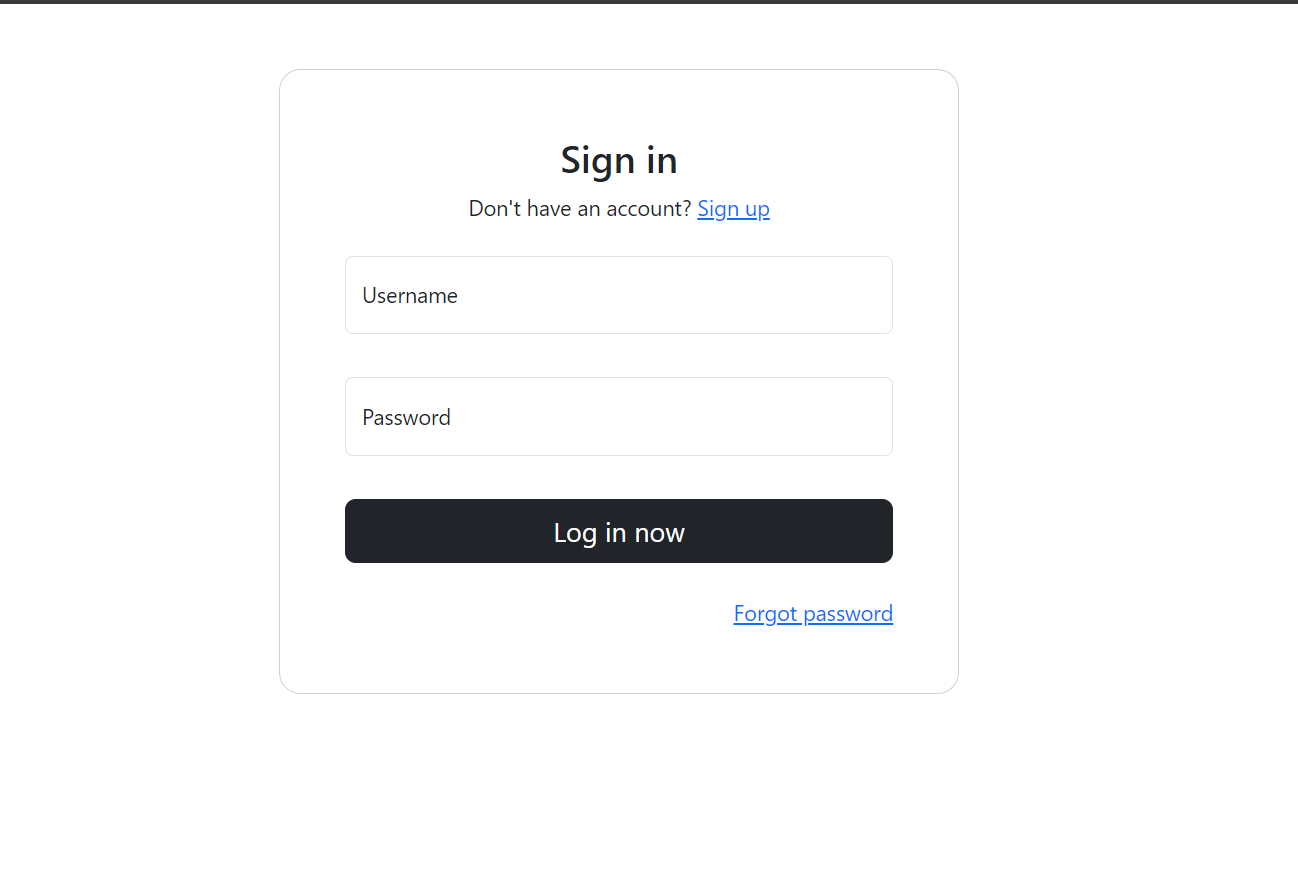


****

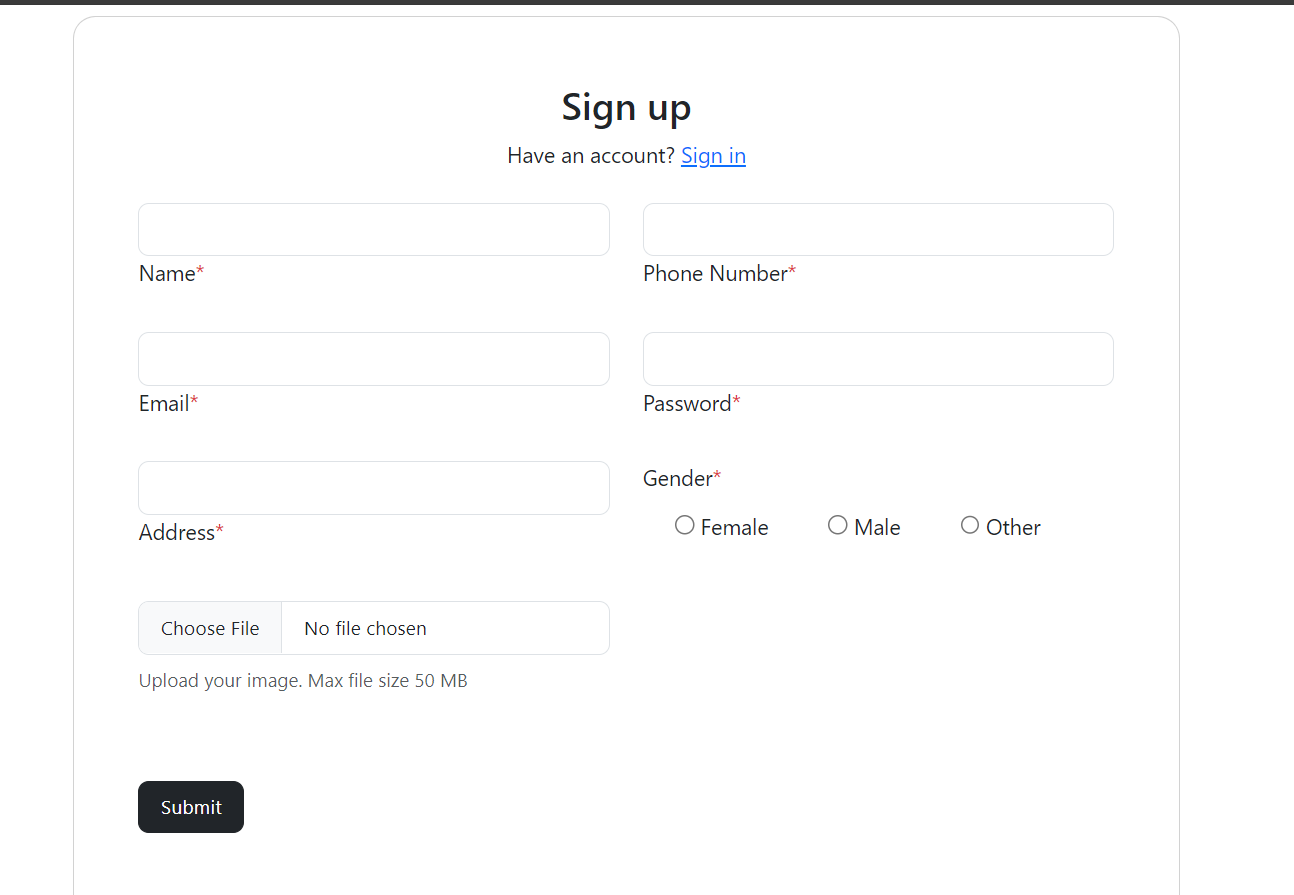
****

**Screenshots of website**

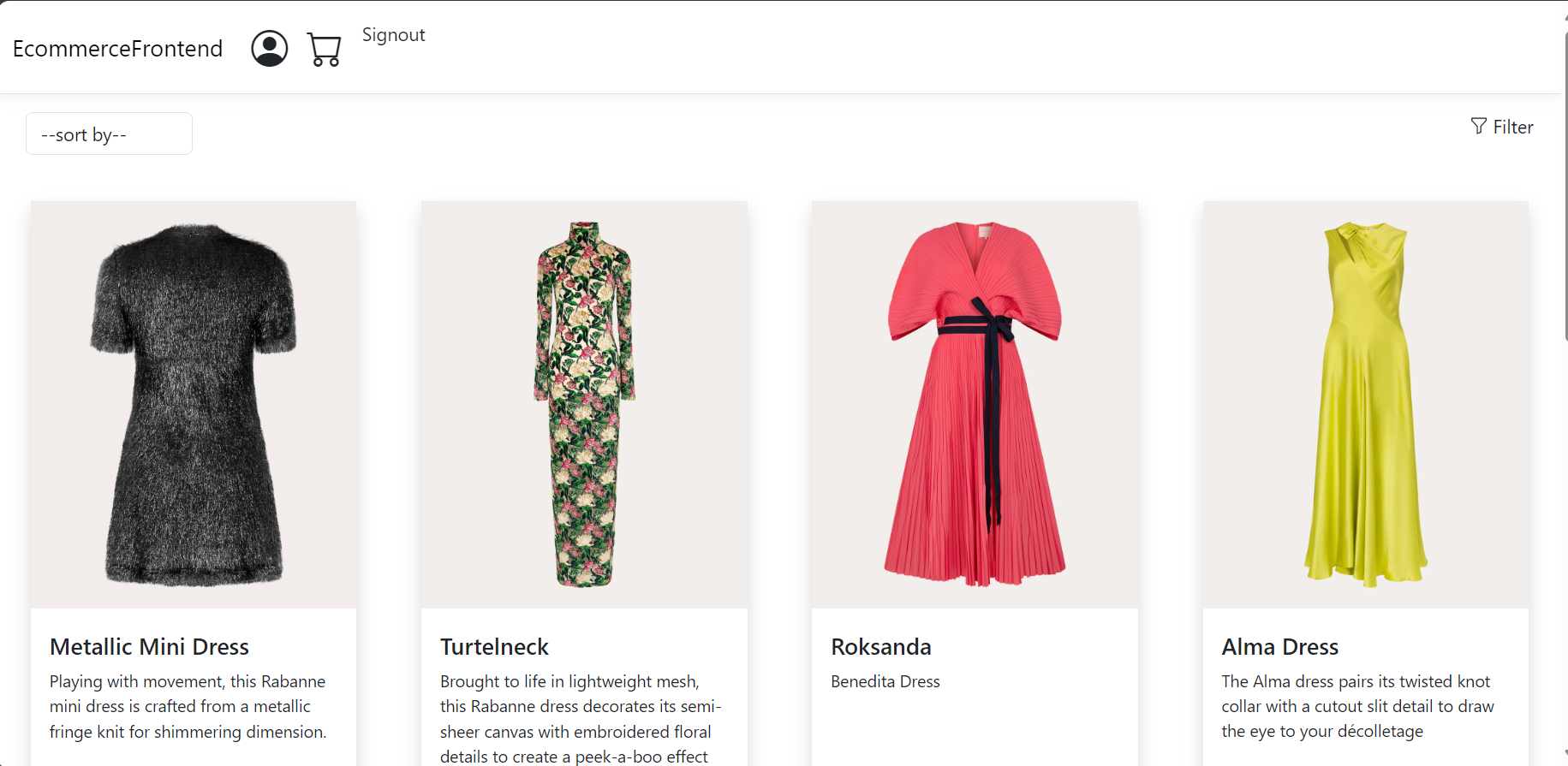
Login page



Registration page



Products page



Cart sidebar

A screenshot of a dress

Description automatically generated

Filtering the products by colors

A screenshot of a computer

Description automatically generated\

Checkout page

