---

1. Problem Statement

In today's digital era, e-commerce platforms have become essential for businesses to sell products online efficiently. Many small and medium-sized businesses struggle with setting up a secure, scalable, and user-friendly platform that supports authentication, product management, shopping cart functionalities, and seamless payment integration. This project aims to develop a full-stack e-commerce platform leveraging React for frontend and .NET Core API for backend, ensuring a microservices-based scalable solution.

---

2. User Story-Based Requirements

2.1 User Roles & Permissions

1. **Guest User**
    - Can browse products and search for items.
    - Cannot add products to the cart or checkout without registration.
    - Can view product details and reviews.
2. **Registered User**
    - Can add products to the shopping cart.
    - Can proceed to checkout and complete payment.
    - Can track past orders in the order history.
    - Can leave reviews on purchased products.
3. **Admin**
    - Can manage product listings (add, update, delete, and categorize products).
    - Can process and manage orders.
    - Can view reports and sales analytics.

2.2 Functional Features

Authentication & Authorization (JWT-Based)

- Users can register and log in securely.
- Role-based access control to restrict admin functionalities.
- JWT token-based authentication for API requests.

Product Listing & Search

- Products are listed with filtering and sorting options.
- Users can search for products by name, category, and price range.
- Detailed product pages with descriptions, images, and customer reviews.

Shopping Cart & Checkout

- Users can add/remove items from the cart.
- Order summary with total cost calculation.
- Secure payment processing via Stripe/PayPal.
- Address and contact information collection during checkout.

Order Management

- Users can track their past orders.
- Admins can process orders and update statuses (pending, shipped, delivered).

Microservices Architecture

- Separate services for authentication, product management, order processing, and payment handling.
- Communication between services using RESTful APIs.

---

3. Standard Submission Guidelines

3.1 Project Submission Requirements

- Source code should be hosted on GitHub with a detailed README.
- README should include setup instructions, project architecture, API endpoints, and usage guidelines.
- Well-structured commit history following best practices.
- Deployment link on Azure/AWS (if applicable).

3.2 Documentation Requirements

- API Documentation using Swagger.
- High-level architecture diagram (microservices overview, database schema, and data flow diagram).
- Functional and non-functional requirements list.

3.3 Code Quality & Best Practices

- Follow SOLID principles and modular code design.
- Use proper naming conventions and comments where necessary.
- Implement error handling and input validation.

---

4. API Testing Requirements

4.1 Authentication API Tests

- User registration with valid and invalid data.
- Login and JWT token validation.
- Unauthorized access to protected routes.

4.2 Product API Tests

- Adding new products (Admin only).
- Searching and filtering products.
- Fetching product details.

4.3 Cart & Checkout API Tests

- Adding/removing items from cart.
- Processing orders and verifying payment integration.
- Retrieving order history.

4.4 Order Management API Tests

- Order placement and status updates.
- Fetching user orders and tracking.
- Admin order management actions.

---

5. Project Directory Structure

```
Unset

Ecommerce-Platform/

|—— backend/ (ASP.NET Core API)
```

```
|   ├── AuthenticationService/ (Handles user authentication)
|   ├── ProductService/ (Manages product listings & search)
|   ├── CartService/ (Handles cart operations & checkout)
|   ├── OrderService/ (Processes orders & order history)
|   ├── PaymentService/ (Integrates with Stripe/PayPal)
|   ├── Common/ (Shared utilities & middleware)
|   ├── Database/ (Entity Framework Models & Migrations)
|   ├── API/Controllers/ (API endpoints for each service)
|   └── API/appsettings.json (Configuration settings)
|
|── frontend/ (React App)
|   ├── src/
|   |   ├── components/ (Reusable UI components)
|   |   ├── pages/ (Authentication, Home, Product, Cart,
Checkout, Orders)
|   |   ├── services/ (API calls using Axios)
|   |   ├── redux/ (State management)
|   |   ├── App.js (Main App component)
|   |   ├── index.js (ReactDOM render)
|   └── package.json (Dependencies & scripts)
|
|── docs/ (Project Documentation)
```

```
|   ├── API_Documentation.md
|   ├── Architecture_Diagram.png
|   ├── User_Manual.pdf
|   ├── README.md
|
|── tests/ (API & UI Tests)
|   ├── backend_tests/ (Postman, NUnit tests)
|   ├── frontend_tests/ (Jest, React Testing Library)
|
|── deployment/ (Deployment Scripts & Configurations)
|   ├── docker-compose.yml
|   ├── azure-pipelines.yml
|   ├── aws-deployment-scripts/
```
