**Full-Stack E-Commerce Web Application & Admin Portal**

**Course:** CSE 214 Advanced Application Development
**Project Title:** Full-Stack E-Commerce Web Application Development

## Objective

The objective of this project is to design and develop a full-stack e-commerce web application with an integrated admin portal. The application will be built using **Angular (Front-End)** and **Spring Boot (Back-End)**, ensuring seamless communication via **REST APIs**. The platform will provide secure user authentication, an interactive shopping experience, and an admin dashboard for managing products, users, and transactions.

## Scope & Features

### 1. Front-End (Angular)

1. **Single Page Application (SPA):** Implemented using Angular with lazy loading to optimize performance.
2. **Authentication & Authorization:** Role-based access control for **Users (Customers) and Admins**.
3. **Guards & Routing:** Route protection using **AuthGuard**, ensuring only authorized users can access certain pages.
4. **E-Commerce Features:**
    - Product browsing, searching, and filtering.
    - Product comparisons and reviews.
    - Shopping cart and checkout process.
    - Order tracking system.
5. **Admin Portal (Premium Role):**
    - Product & inventory management.

- Order & shipment tracking.
- User management (ban/unban users, view transactions).
- Issue resolution for payments and orders.

**2. Back-End (Spring Boot)**

1. **User Management:** Secure authentication and role-based authorization.
2. **Product & Order Management:** CRUD operations for managing products, orders, and payments.
3. **Payment Processing:** Integration with third-party payment gateways (e.g., Stripe, PayPal).
4. **Admin Portal Functionalities:** Data analytics, reports, and resolving user issues.
5. **Database (MySQL):** Relational database with Hibernate/JPA mappings for storing user, product, order, and payment data.

---

**Use Case Scenarios**

**1. User Purchase Flow**

**Actors:** User (Customer), System, Seller, Payment Gateway, Logistics Provider, Admin

1. User browses product categories or searches for a product.
2. User views product details, compares products, and reads reviews.
3. User adds products to the shopping cart.
4. User proceeds to checkout and makes a secure payment.
5. The system confirms the order and notifies the user.
6. The seller processes the order, and logistics ship it.
7. The user tracks the order until delivery.

8. The user receives the product and leaves a review.
9. If needed, the user requests a return or refund.

**2. Admin Portal Use Case**

**Actors:** Admin (Superuser), System, Users, Sellers, Payment Gateway, Logistics Provider

**Admin Authentication & Access Control:**

- Only **authorized admins (premium role)** can access the admin panel.
- Admins log in with credentials and are verified before access is granted.

**Admin Dashboard Functionalities:**

1. **User Management:** View user details, disable suspicious accounts, and reset passwords.
2. **Order & Payment Issue Resolution:** Resolve failed transactions, refunds, or shipping delays.
3. **Product & Inventory Management:** Add, update, or remove products.
4. **Shipment & Logistics:** Track pending and completed orders.
5. **Customer Support:** View and respond to complaints, returns, and refund requests.

---

**Technology Stack**

**Frontend (Angular)**

- **State Management:** RxJS
- **Lazy Loading:** Optimized module loading
- **Guards:** Route protection using AuthGuard
- **UI Framework:** Angular Material / Tailwind CSS

**Backend (Spring Boot)**

- **Security:** Spring Security + JWT for authentication
- **Database:** MySQL with Hibernate ORM
- **API Development:** RESTful API with Swagger Documentation
- **Payment Integration:** Stripe/PayPal

Both **Stripe** and **PayPal** provide **sandbox (test) environments** that allow students to send test payment requests without processing real transactions.

**1. Stripe Test API**

Stripe offers a **test mode** where students can make payments using test cards.

**How to Use Stripe Test API?**

1. **Sign Up**: Create a free account on *https://dashboard.stripe.com/register*
2. **Get API Keys**:
    - Go to **Developers → API keys** in the dashboard.
    - Use the **Test Secret Key** and **Test Publishable Key**.
3. **Use Test Cards**: Stripe provides **test card numbers**
4. **Make a Payment Request**: Send a POST request to:

    *https://api.stripe.com/v1/payment_intents*
5. **View Test Transactions**: All test payments appear in the **Stripe Dashboard** under "Test Mode".

## 2. PayPal Sandbox API

PayPal has a **Sandbox Mode** for developers to test transactions.

**How to Use PayPal Sandbox API?**

1. **Create a Developer Account** at PayPal Developer.
2. **Set Up Sandbox Accounts**:
    - Inside the **PayPal Developer Dashboard**, create test **buyer** and **seller** accounts.
3. **Get API Credentials**:
    - Navigate to **My Apps & Credentials** → Create a new app.
    - Get the **Client ID** and **Secret Key** for testing.
4. **Use the PayPal API Endpoint:**

    *https://api-m.sandbox.paypal.com*
5. **Make a Test Payment**:
    - Use PayPal's test accounts for transactions.