UE22CS352B - Object Oriented Analysis & Design

Mini Project Report

Title: Ecommerce Application

Submitted by:

NITHIN : PES1UG22CS399

Prakul : PES1UG22CS428

Piyush Jain: PES1UG22CS414

Naman Jain: PES1UG22CS371

Semester:6 Section: G

Faculty Name: DR. BHARGAVI MOKASHI

January - May 2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

Problem Statement:

Key Features:

Problem Statement:

This project implements a comprehensive ecommerce platform that allows users to browse products, manage shopping carts, place orders, and track order status. The application includes both customer-facing features and administrative capabilities for managing products, users, and orders. The architecture follows best practices with clear separation between frontend and backend components, secure authentication, and implementation of various design patterns to ensure maintainability and extensibility. **Key features:**

Customer Features

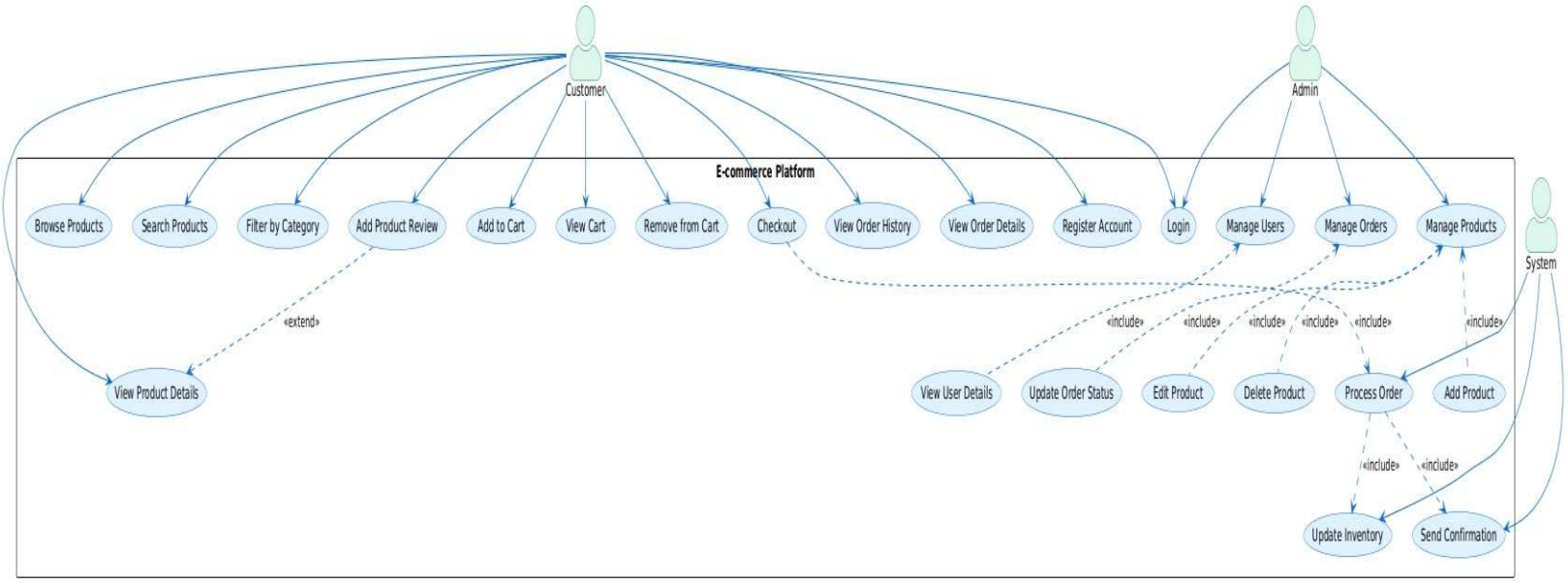
1. Product Management ○ Browse product listings with search and filtering ○ View detailed product information with images ○ Submit and read product reviews with star ratings
2. Shopping Cart ○ Add products to cart with quantity selection ○ View cart contents and update quantities ○ Remove items from cart
   - Calculate subtotal, taxes, and total cost
3. Checkout Process ○ Provide shipping address information ○ Select delivery options ○ Place orders with confirmation
4. Order Management ○ View order history
   - Track order status (PENDING, DELIVERED, CLOSED)
   - View detailed order information including line items and delivery dates
5. User Authentication ○ User registration and login
   - Secure password storage with PBKDF2 hashing

Admin Features

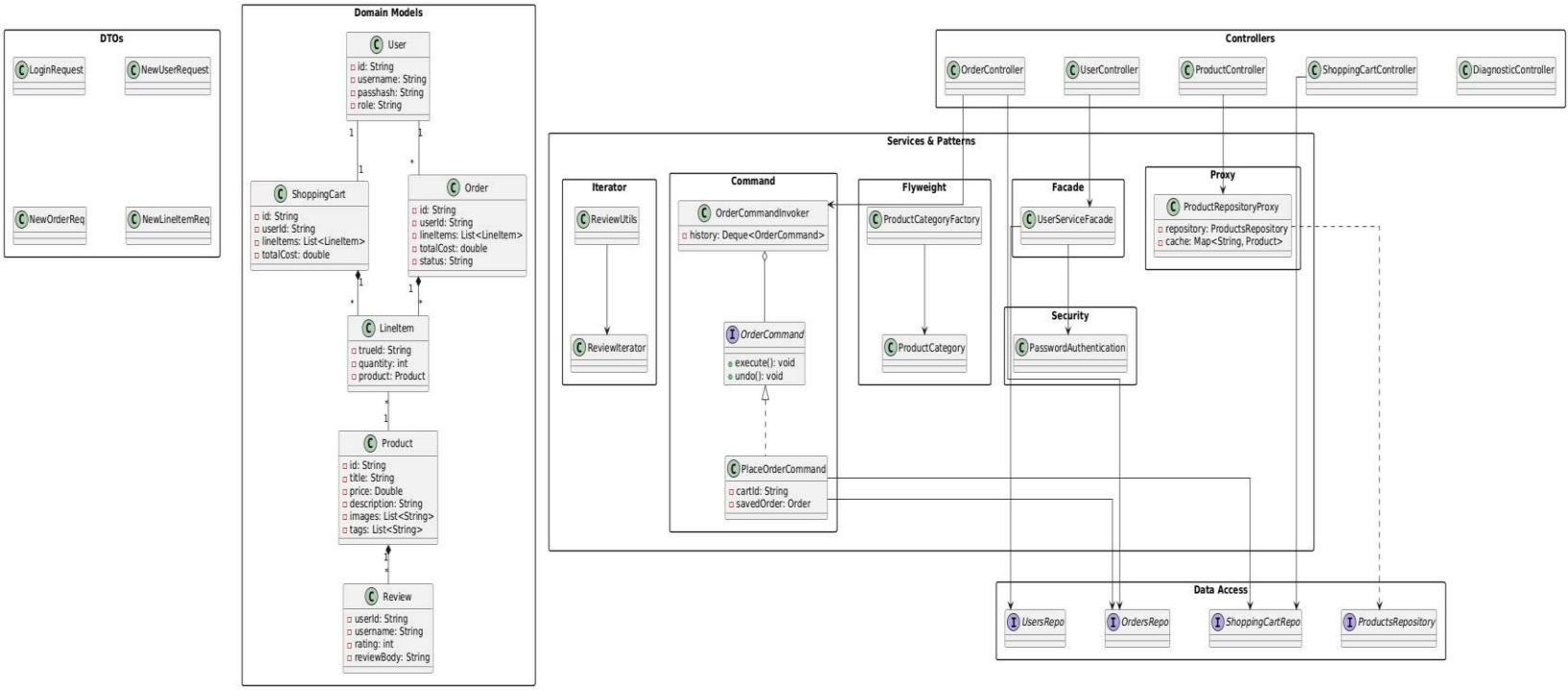
1. User Management
   - View all users
   - Access user details and order history
2. Product Administration ○ Add new products with details and images ○ Edit existing product information ○ Delete products from the catalog
3. Order Processing
   - View all orders in the system
   - Update order status (mark as delivered or closed) ○ Access detailed order information

Models:

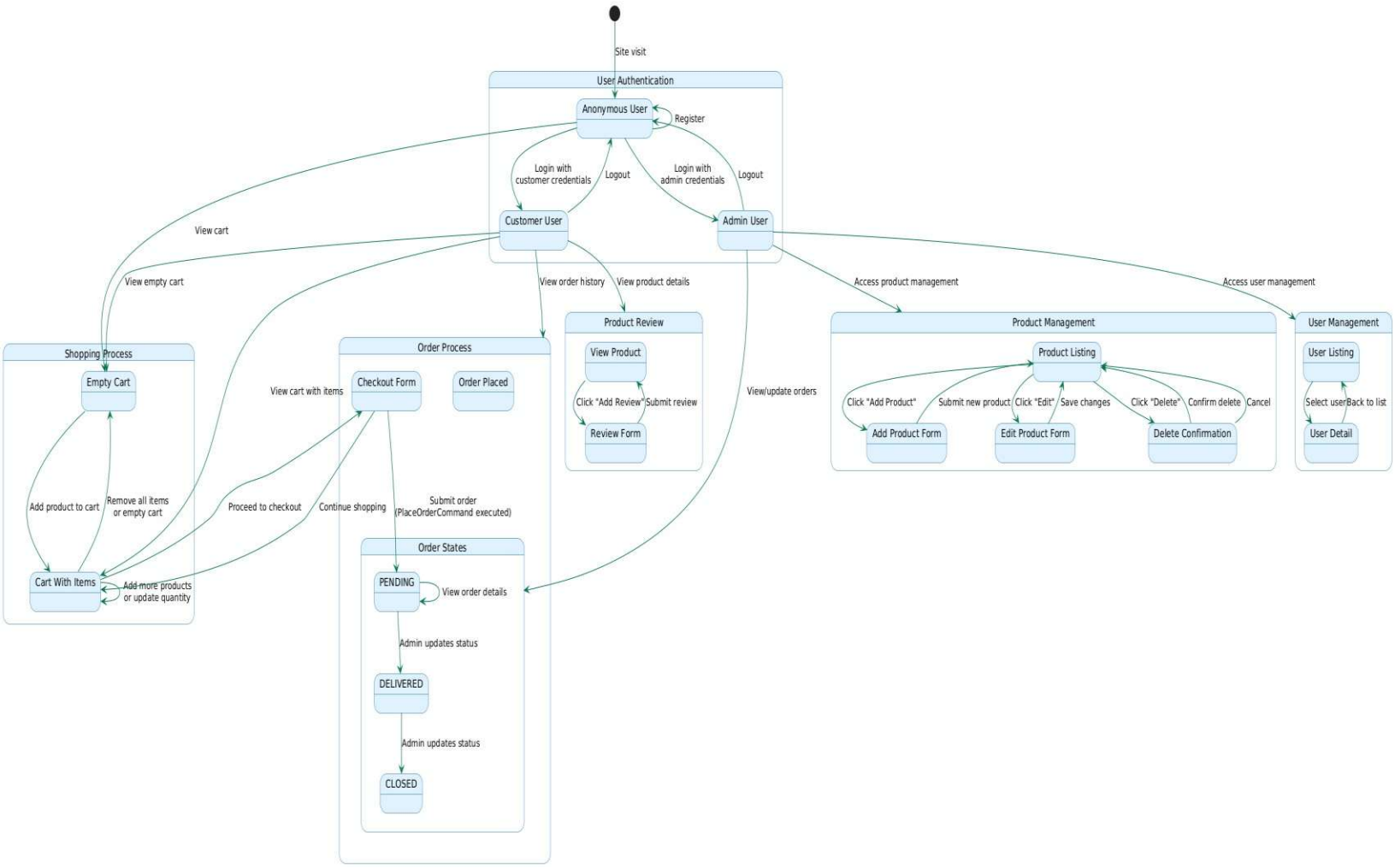
Use Case Diagram:



Class Diagram:

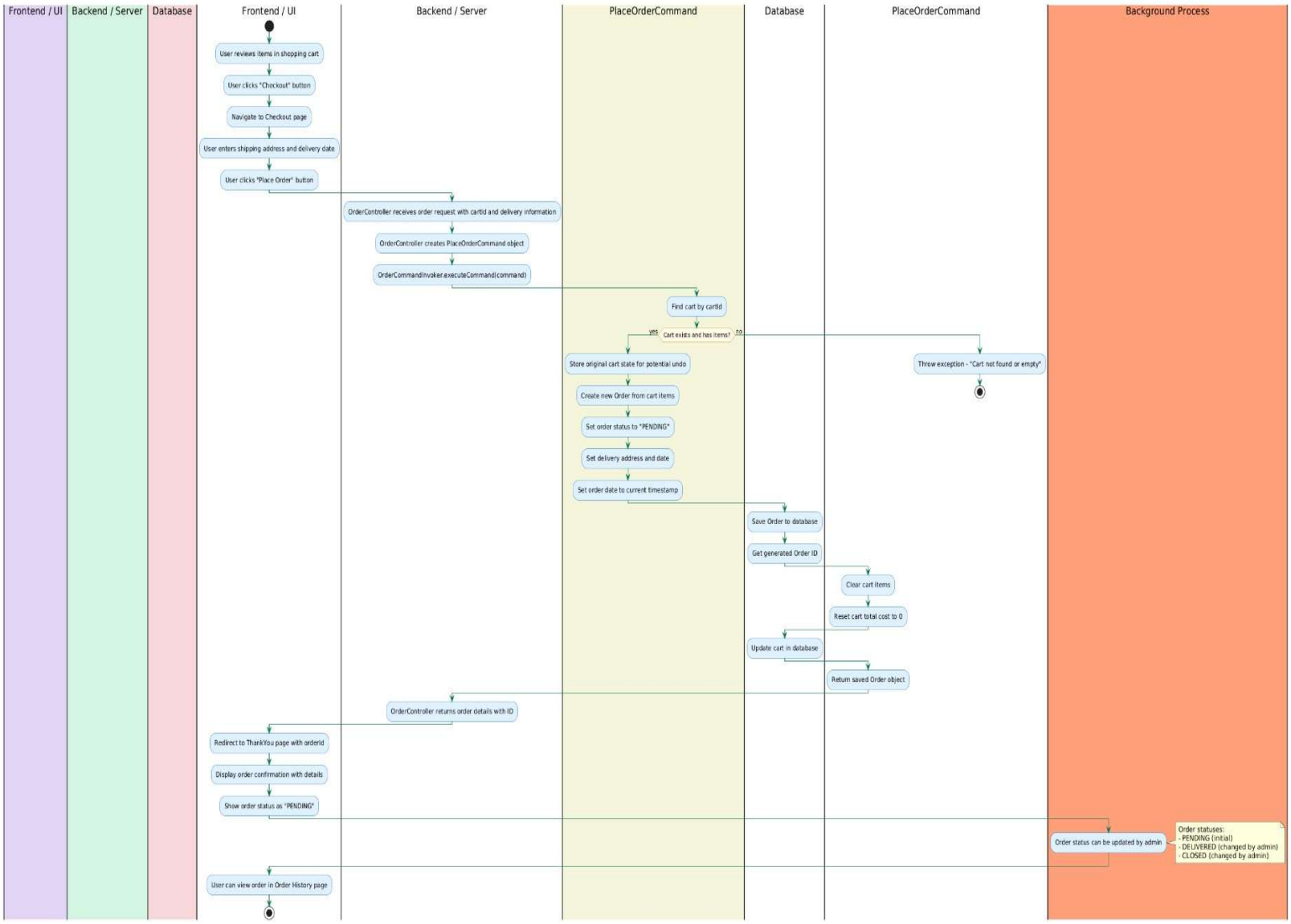


State Diagram:

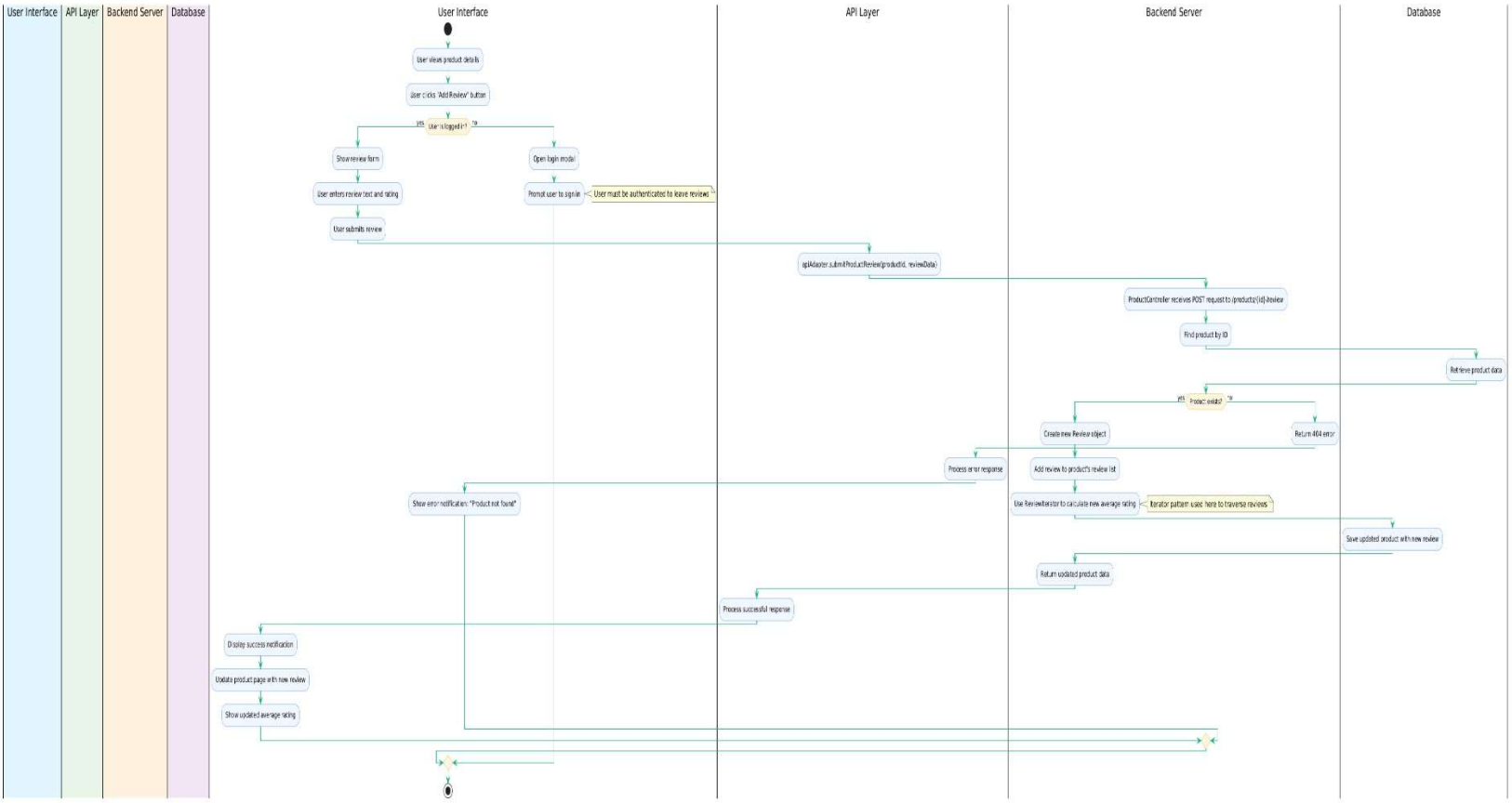


Activity Diagrams:

1. Major Usecase: checkout for payment



2.Minor Use case: product review submission



Architecture Patterns, Design Principles, and Design Patterns:

Architecture Patterns

Model – View – Controller Pattern (MVC)

Model Components
The Model layer represents the application data and business logic:
- Product.java
- Order.java
- User.java
- ShoppingCart.java
- LineItem.java
- Review.java

Data Access / Repositories
- ProductsRepository.java
- OrdersRepo.java
- UsersRepo.java
- ShoppingCartRepo.java
- ProductRepositoryProxy.java (implements Proxy pattern for caching)

Business Logic
- UserServiceFacade.java (implements Facade pattern)
- OrderCommandInvoker.java (implements Command pattern)
- PlaceOrderCommand.java (implements Command pattern)
- ReviewUtils.java (implements Iterator pattern)
- AuthorizationService.java

View Components
The View layer is primarily implemented in React components:
User Interface Components
- Products.jsx (product listing view)
- Product.jsx (product detail view)
- Cart.jsx (shopping cart view)
- Checkout.jsx (checkout process view)
- OrdersPage.jsx (order history view)
- AdminProductManagement.jsx (admin product management view)
- NavBar.jsx (navigation component)
- LoginModal.jsx/RegisterModal.jsx (authentication views)

- ThankYou.jsx (order confirmation view)

Controller Components

The Controller layer mediates between Model and View:

Backend Controllers

- ProductController.java (handles product-related HTTP requests)
- OrderController.java (handles order-related HTTP requests)
- ShoppingCartController.java (handles cart-related HTTP requests)
- UserController.java (handles user-related HTTP requests)
- DiagnosticController.java (handles diagnostic HTTP requests)

MVC Interaction Flow Example

View: The user interacts with Product.jsx and submits a review form
Controller:
Frontend: ApiAdapter.js sends a POST request to /products/{id}/review Backend: ProductController.java receives the request Model:
ProductController finds the product via ProductRepositoryProxy
It creates a new Review object and adds it to the product
ReviewUtils uses ReviewIterator to calculate the new rating
The updated Product is saved to the database
Controller: Returns the updated product data
View: Product.jsx updates to display the new review and rating

Design Principles

1. Information Expert

Definition: Assign responsibility to the class that has the necessary information.

Examples in code:

- ProductController handles product operations because it has access to product data via ProductRepositoryProxy:
- ShoppingCartController manages cart operations because it has cart information:

2. Creator

Definition: Assign responsibility for creating objects to the class that has the necessary information.

Examples:

- PlaceOrderCommand creates Order objects because it has all the necessary information:

3. Controller

Definition: Assign responsibility for handling system events to a class representing the overall system or a use case scenario.

Examples:

- The OrderController handles external order requests and coordinates the response:

4. Low Coupling

Definition: Reduce dependencies between classes to decrease impact of changes.

Examples:

- ApiAdapter decouples frontend components from direct API interaction
- ProductRepositoryProxy decouples the controller from direct repository access

The project also follows high cohesion, Indirection,SOLID principles

Design Patterns

1. Command Pattern

The Command pattern encapsulates a request as an object, allowing you to parameterize clients with different requests, queue or log requests, and support undoable operations.
Implementation:

- OrderCommand interface defines the contract with execute() and undo() methods
- PlaceOrderCommand class implements this interface, encapsulating order placement logic
- The command stores its state (original cart) to enable the undo operation

2.Iterator Pattern

The Iterator pattern provides sequential access to elements of an aggregate object without exposing its underlying representation. Implementation:

- ReviewIterator implements java.util.Iterator<Review>
- Provides hasNext() and next() methods to traverse reviews

3. Proxy Pattern

The Proxy pattern provides a surrogate or placeholder for another object to control access to it.
Implementation:

- ProductRepositoryProxy acts as a proxy for ProductsRepository
- Adds caching functionality to improve performance for frequently accessed products

2. Facade Pattern

The Facade pattern provides a unified interface to a set of interfaces in a subsystem, making the subsystem easier to use. Implementation:

- UserServiceFacade simplifies the interface for user-related operations
- Encapsulates complex user management functionality

3. Flyweight Pattern

The Flyweight pattern uses sharing to support large numbers of fine-grained objects efficiently.
Implementation:

- ProductCategory represents shared immutable category information
- ProductCategoryFactory manages a pool of shared category instances

4. Adapter Pattern

The Adapter pattern converts the interface of a class into another interface clients expect.
Implementation:

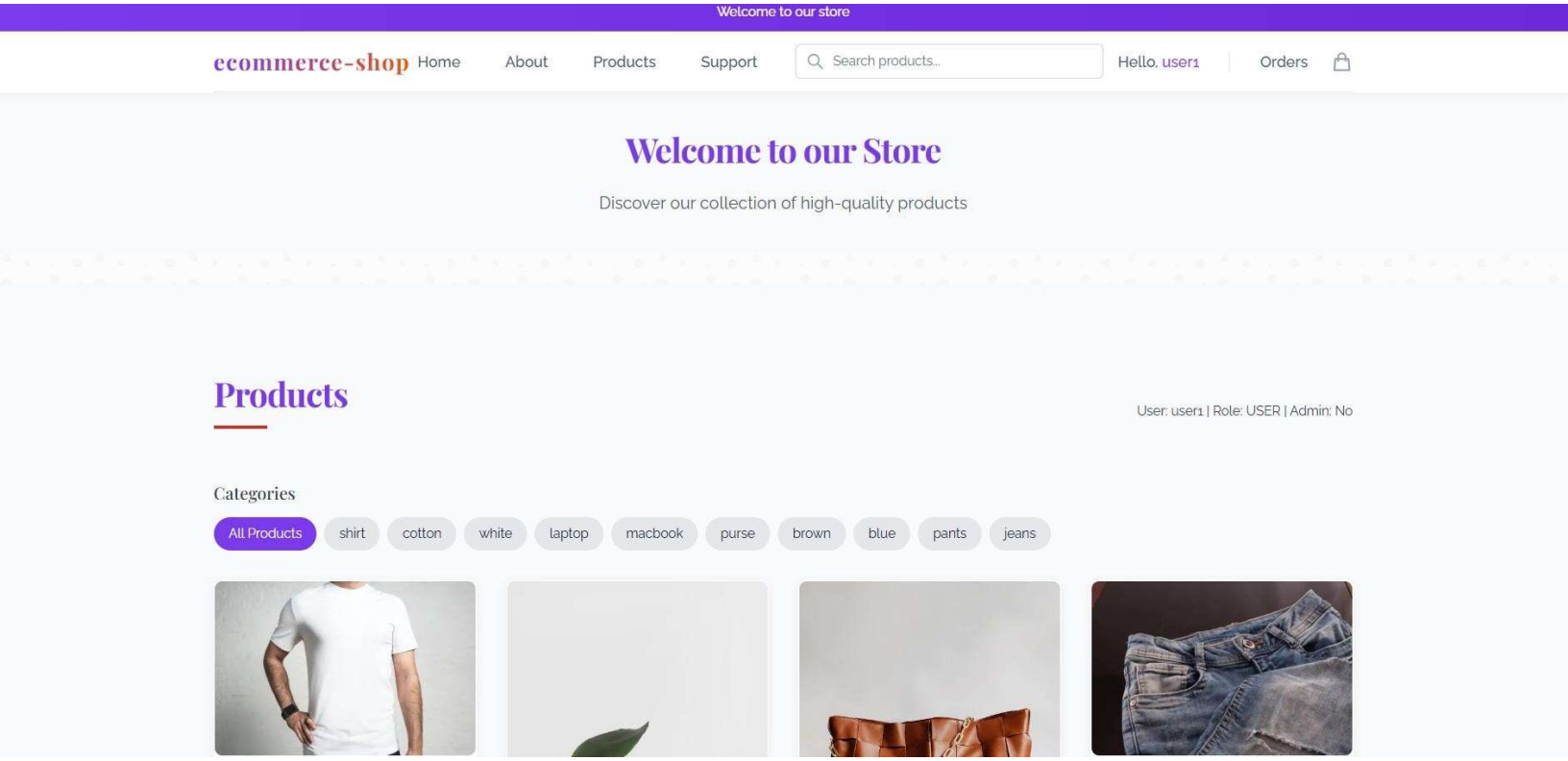
- ApiAdapter in the frontend adapts between React components and backend API
- Provides a consistent interface for frontend components to interact with backend services

Github link to the Codebase:

Nithin-ganiga/ecommerce

Screenshots

UI:



White T shirt

$10.00

★★★★★ 1 review

Add to bag

Shopping cart

White T shirt $10.00

Qty: 1 Remove

Purse $40.00

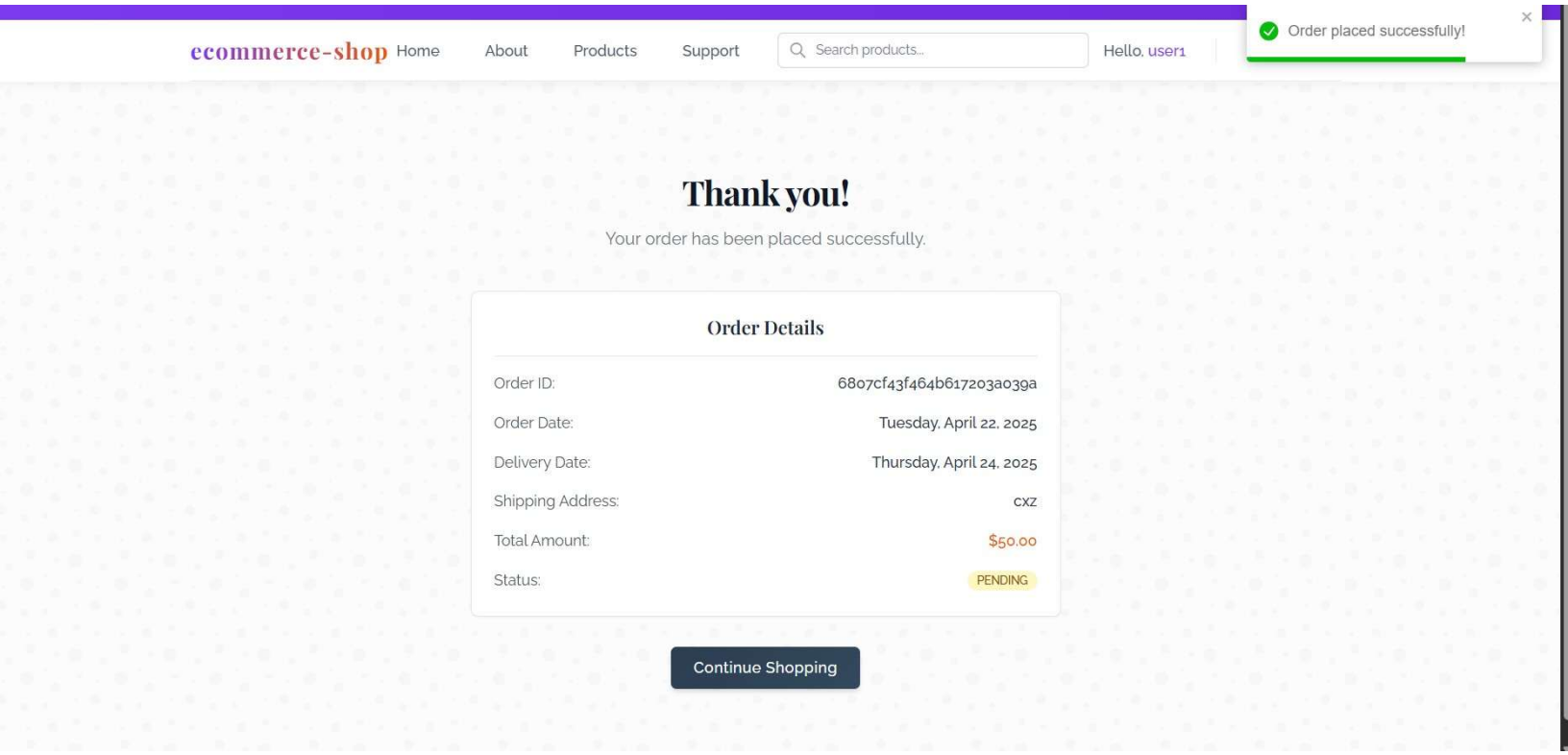
Qty: 1 Remove

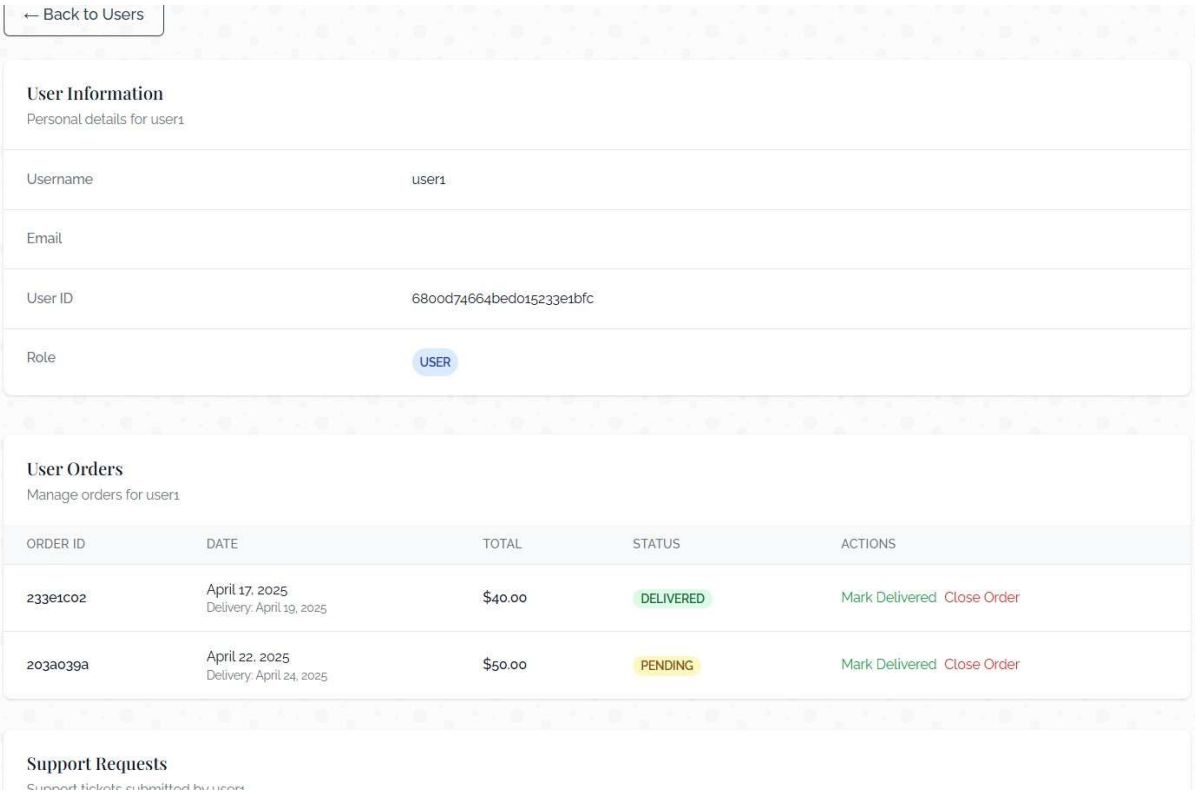
Subtotal $50.00

Shipping and taxes calculated at checkout.

Checkout

or Continue Shopping →





Individual contributions of the team members:

Name	Module worked on
NITHIN	Signup,Login,Authentication,Admin privileges
Prakul	Home, Products,Reviews,Product Page
Piyush Jain	Cart,Customer support, User details,Order details
Naman Jain	Delivery,Thank you page, Search, Categories, Adding and removal of products