SOFTWARE REQUIREMENTS SPECIFICATION

Project Title: Online Grocery Ordering System

Prepared by: Praneethha

Date: 30-07-2025

1.Introduction

1.1 Purpose

The purpose of this document is to outline the requirements for an Online Grocery Ordering System. It enables users to browse, search, and order groceries online with home delivery or in-store pickup options.

1.2 Scope

- The Online Grocery Ordering System (OGOS) will include the following capabilities:
- User registration and authentication
- Browsing and searching for grocery products
- Viewing product details and availability
- Managing a virtual cart
- Placing orders for home delivery or in-store pickup
- Online payment integration (debit/credit card, UPI, wallet, and cash-on-delivery)
- Order tracking and history
- Admin dashboard to manage products, orders, inventory, and users

The system will be accessible via web browsers and mobile devices and will be designed to ensure scalability, security, and ease of use.

1.3 Definitions

- **User (Customer)**: A person who registers and uses the platform to purchase groceries.
- **Admin**: A backend user with privileges to manage products, users, and order workflows.
- **Cart**: A virtual collection of items the user intends to purchase.
- **Order**: A confirmed request made by a user for purchasing products from the system.
- **COD**: Cash on Delivery
- **UI/UX**: User Interface / User Experience

2. Functional Requirements

2.1 User Features

2.1.1 User Registration and Login

- Users can create accounts using email/mobile and password.
- Users receive confirmation via email or SMS.
- Login via secure form with CAPTCHA and password encryption.
- Forgot Password functionality via OTP/email verification.

2.1.2 Product Browsing

- Products are organized by categories and subcategories.
- Displayed in a paginated, responsive layout with thumbnails, price, quantity, and “Add to Cart” button.
- Users can browse featured, trending, and recommended products.

2.1.3 Product Search and Filter

- Keyword search for products.
- Filtering by brand, category, price range, availability, and rating.
- Sorting by price, popularity, newest first, etc.

2.1.4 View Product Details

- Product detail page includes image gallery, price, stock info, full description, ingredients (if applicable), seller, and customer reviews.

2.1.5 Shopping Cart

- Add items to cart.
- Edit quantity or remove items.
- Display of subtotal, tax, delivery charges, and final total.
- Cart persistence using session or account sync.

2.1.6 Order Placement

- Address selection or input
- Delivery slot selection
- Payment method selection
- Apply coupon codes (optional)
- Confirm and place order
- Receive order confirmation via email/SMS

2.1.7 Payment Integration

- Integrated payment gateways (Razorpay, Stripe, etc.)
- COD option
- Refund initiation for cancellations

2.1.8 Order History and Tracking

- View list of previous orders
- View status: Pending, Processing, Dispatched, Delivered, Canceled
- Estimated delivery time
- Option to cancel or return based on status

2.1.9 Profile Management

- View and update personal details
- Manage addresses and contact info
- Change password
- View saved payment methods (optional)

2.2 Admin Features

2.2.1 Admin Authentication

- Admin login using secure credentials
- Two-factor authentication (optional)

2.2.2 Product Management

- Add/edit/delete products
- Manage stock levels and price updates
- Upload product images
- Manage product categories and tags

2.2.3 Order Management

- View all orders by status
- Update order status (e.g., processing, shipped, delivered)
- Generate order invoices

- Cancel or modify orders

2.2.4 User Management

- View list of registered users
- Deactivate/reactivate accounts
- View order history per user
- Send announcements or notifications

2.2.5 Reports and Analytics

- Sales reports by day/week/month
- Best-selling products
- Inventory alerts
- User activity and engagement metrics

3. Non-Functional Requirements

3.1 Security

- Passwords hashed and salted.
- Secure access control for admin panel.
- OWASP Top 10 compliance for web security.

3.2 Performance

- Pages load within seconds.
- Concurrent support for users.
- Background job queues for order processing and email.

3.3 Scalability

- Microservices or modular backend that supports horizontal scaling.

- Cloud-based storage and content delivery (CDN) for product media.

3.4 Availability

- Automated backups.
- Failover servers and load balancing enabled.

3.5 Usability

- Clean, intuitive UI for both users and admins
- Multilingual support (optional)

3.6 Maintainability

- Well-documented codebase
- Modular MVC or REST architecture
- Continuous integration and deployment pipeline (CI/CD)

4. System Requirements

4.1 Frontend

- Technologies: React Native

Responsive design for mobile, tablet, and desktop

4.2 Backend

- Technologies: Spring Boot (Java)
- RESTful APIs for client-server communication

4.3 Database

- MySQL / PostgreSQL for relational data

- MongoDB for document-oriented flexibility
- Redis (optional) for caching sessions or cart data

4.4 Hosting & Infrastructure

- AWS EC2 / Google Cloud / Microsoft Azure
- S3 or equivalent for media file storage
- CloudFront or CDN for fast media delivery
- Docker + Kubernetes for containerization and orchestration

4.5 External Integrations

- Payment Gateways: Razorpay, Stripe, PayPal
- Email/SMS Notification APIs: SendGrid, Twilio
- Google Maps API for address autocomplete and delivery mapping

SOFTWARE DESIGN DOCUMENT

Project: Online Grocery Ordering System

1. Purpose

This Software Design Document (SDD) outlines **how** the Online Grocery Ordering System will be implemented. While the SRS defines **what** the system should do, this document focuses on the system's architecture, design patterns, interface specifications, data structures, and interaction flows to guide developers, testers, and stakeholders during the development phase.

2. Functional Requirements

Main features implemented:

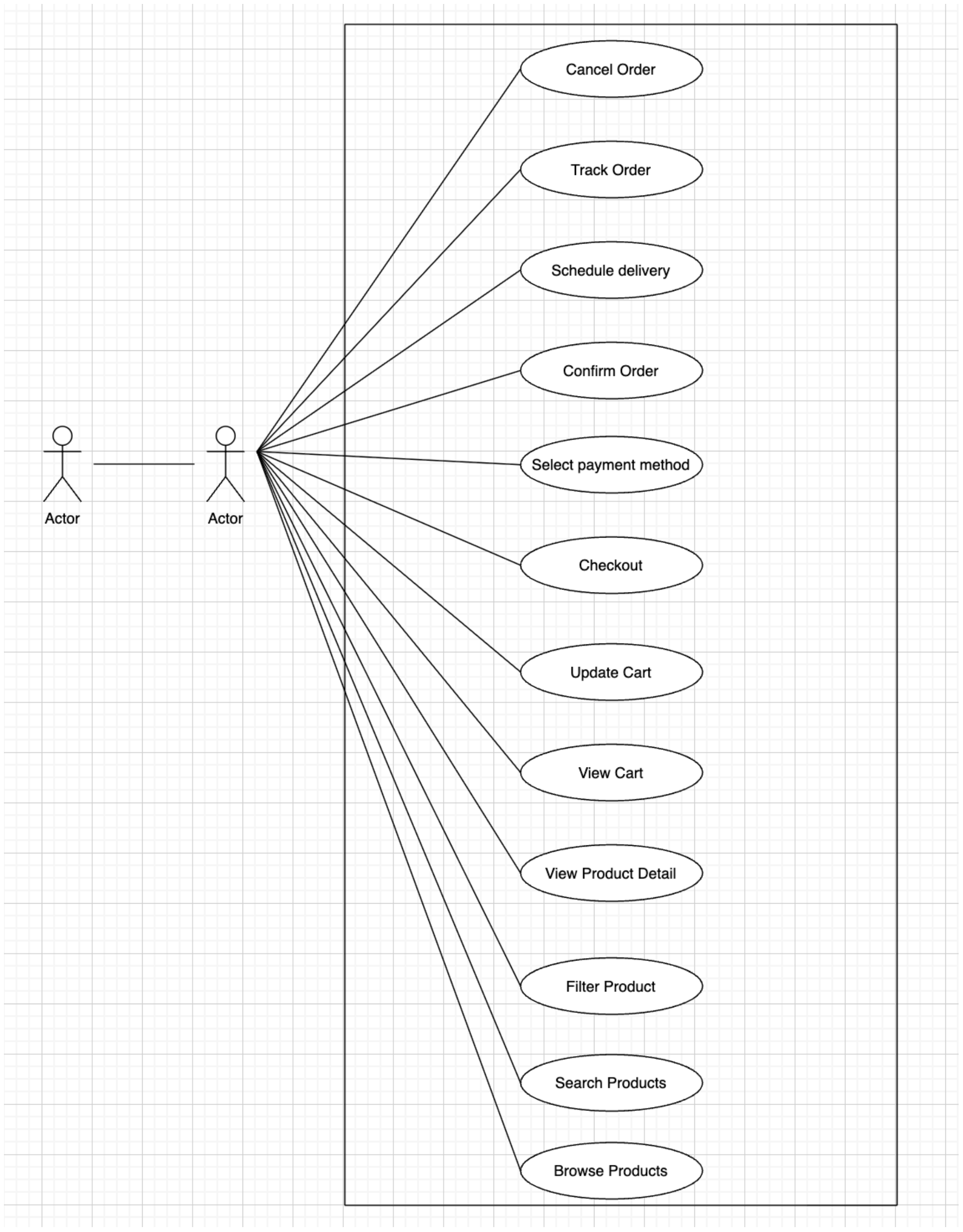
- User registration and login
- Product catalog browsing
- Cart management
- Order placement with delivery/pickup
- Online/Cash payment
- Admin inventory/order/user management

3. Non-Functional Requirements

Only considered here **if they impact design**. Relevant ones include:

- **Security**: Impacts authentication, data encryption, and access control modules.
- **Performance**: Influences caching strategies, database indexing, and API efficiency.
- **Scalability**: Impacts architecture choice (modular/MVC/microservices).
- **Availability**: Guides deployment on cloud infrastructure with redundancy.

4. Use Case Diagrams



5. System Overview

The system is a **web-based e-commerce platform** for grocery shopping. It is structured using a **layered architecture** with:

- A frontend interface for customer and admin users

- A backend server managing business logic and data
- A secure database for persistent storage

Deployment targets high availability and modular expansion, supporting responsive design for mobile and desktop access.

6. Architecture Design

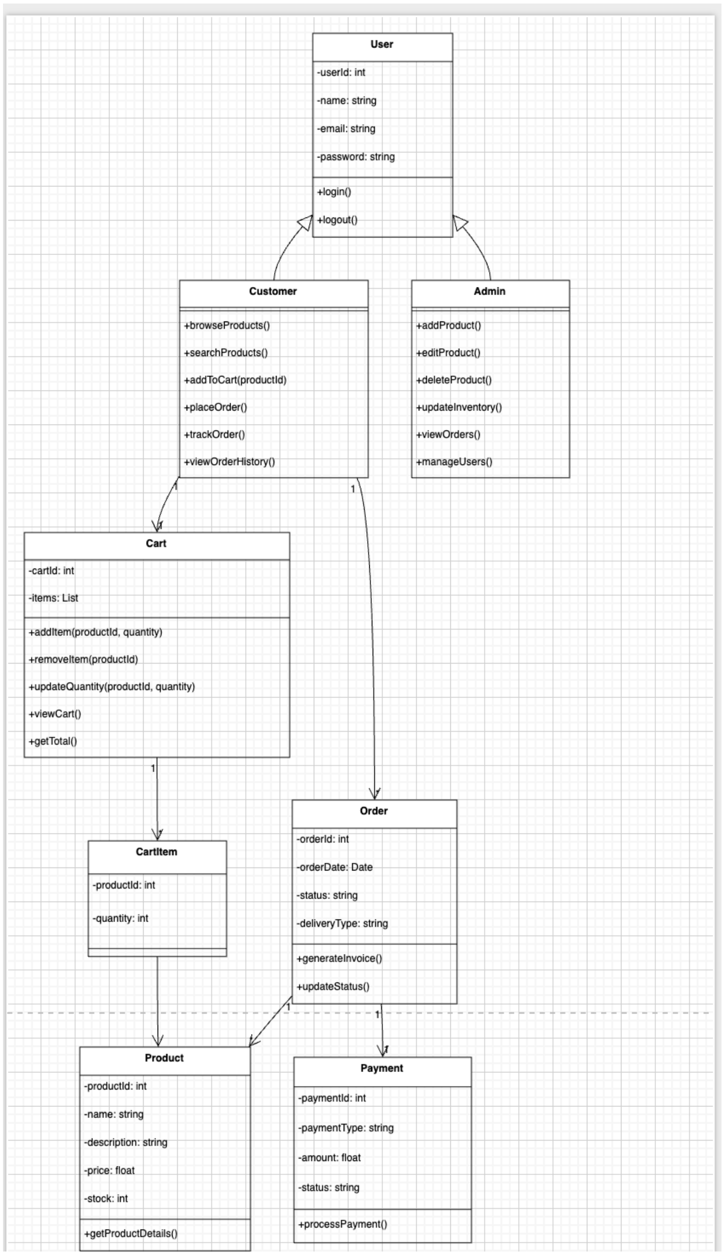
The system follows an **MVC architecture** with RESTful services and modular components.

Layers:

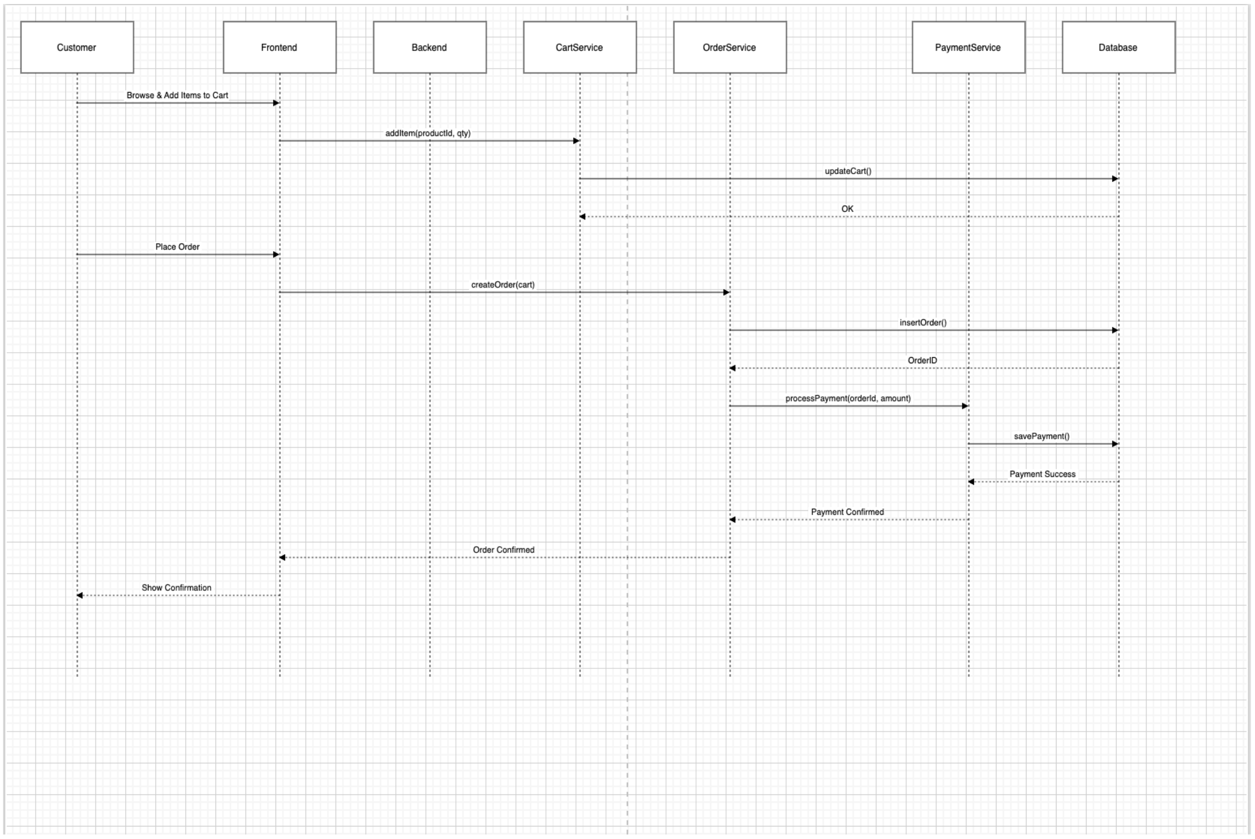
- **Presentation Layer** (UI): React.js / HTML5 / CSS3
- **Controller Layer**: Routes API calls, manages sessions, enforces authorization
- **Service Layer**: Encapsulates business logic (e.g., cart logic, order workflow)
- **Data Access Layer**: Handles interactions with relational (MySQL) or NoSQL (MongoDB) database

Optional enhancement: Use of **microservices** for scaling modules like order handling or payments independently.

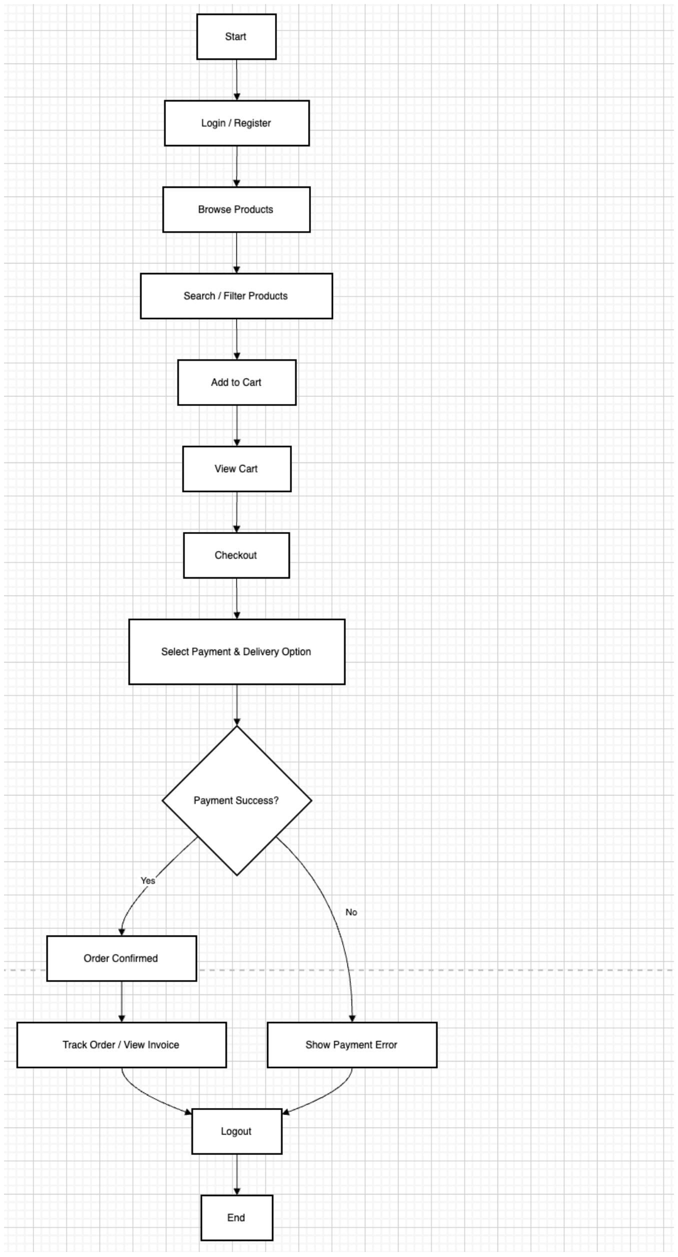
7. Class Diagrams


User
-userId: int
-name: string
-email: string
-password: string
+login()
+logout()
Customer
+browseProducts()
+searchProducts()
+addToCart(productId)
+placeOrder()
+trackOrder()
+viewOrderHistory()
Admin
+addProduct()
+editProduct()
+deleteProduct()
+updateInventory()
+viewOrders()
+manageUsers()
1
1
1
Cart
-cartId: int
-items: List
+addItem(productId, quantity)
+removeItem(productId)
+updateQuantity(productId, quantity)
+viewCart()
+getTotal()
1
CartItem
-productId: int
-quantity: int
Order
-orderId: int
-orderDate: Date
-status: string
-deliveryType: string
+generateInvoice()
+updateStatus()
1
1
1
Product
-productId: int
-name: string
-description: string
-price: float
-stock: int
+getProductDetails()
Payment
-paymentId: int
-paymentType: string
-amount: float
-status: string
+processPayment()

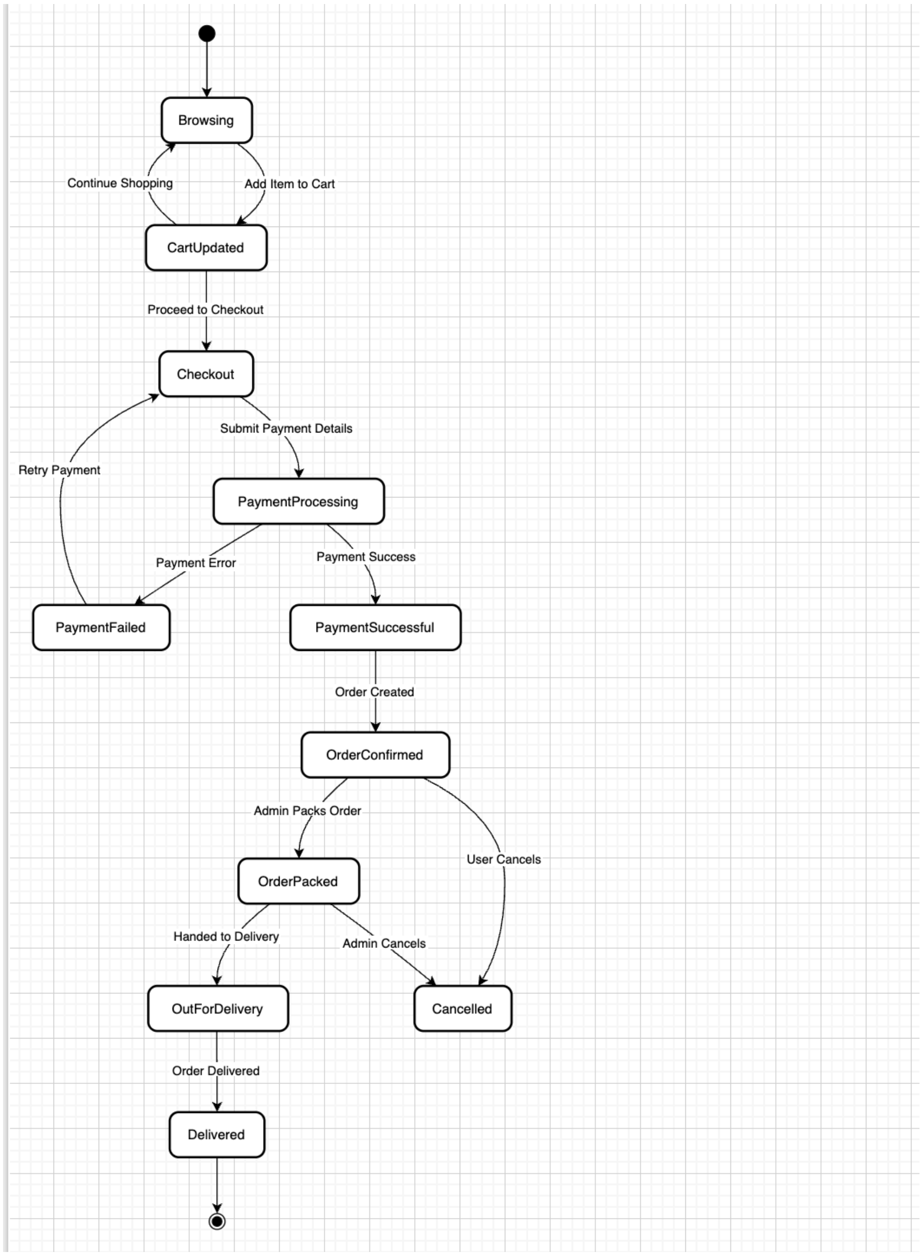
8. Sequence Diagrams



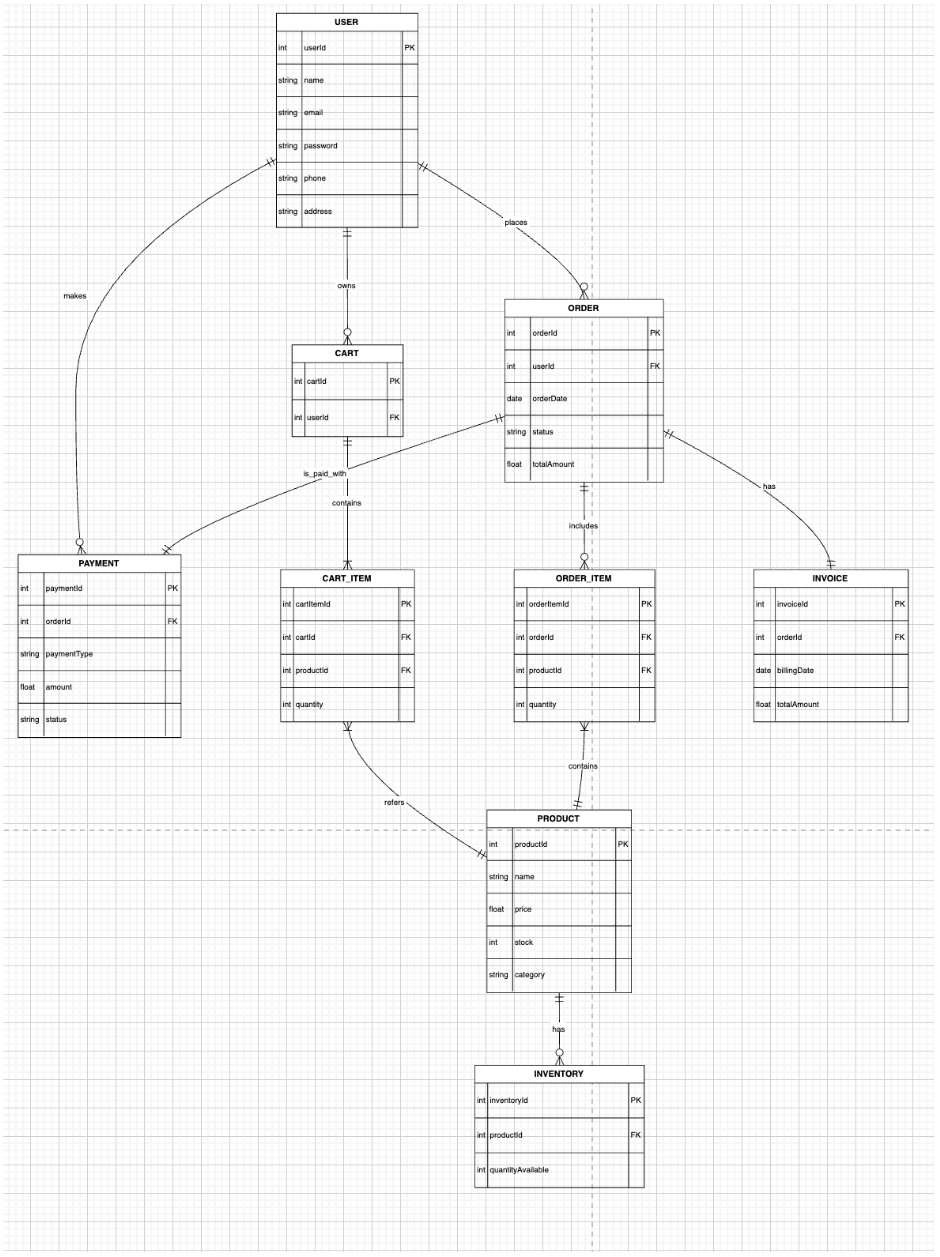
9. Activity Diagrams



10. State Diagrams



11. ER Diagrams



12. Interface Design (API, UI)

Frontend Interface Specifications:

- Product grid view with filters
- Cart summary sidebar
- Checkout flow with address and payment options
- Admin dashboard with editable tables for stock/orders

API Interface:

- POST /api/register
- POST /api/login
- GET /api/products?category=xyz
- POST /api/cart/add
- POST /api/order/confirm
- GET /api/admin/orders

13. Technology Stack

Layer	Technology
Frontend	React Native
Backend	Spring Boot
Database	MySQL / MongoDB
API Protocol	REST, secured with HTTPS

Authentication	JWT (JSON Web Token)
Hosting/Deployment	AWS EC2, Docker, Nginx
CI/CD	GitHub Actions or Jenkins

14. Security Requirements (Technical Implementation)

- **Password encryption**: Bcrypt with salting
- **Secure data transmission**: HTTPS with SSL/TLS
- **Session management**: Token-based with expiry
- **Input validation**: Server-side + frontend validation
- **Access control**: RBAC for user/admin separation
- **Database protection**: SQL injection prevention, backups, access control