INSTITUTE FOR ADVANCED COMPUTING AND SOFTWARE DEVELOPMENT, AKURDI, PUNE

"Farmers Market Place"

PG-DAC August 2025

Submitted By:

Group No: 36

Roll No.	**Name of Student**
258043	Shrikant S. Londhe
258105	Vishwajit N. Nikam

Mr. Prithviraj Shinde
Project Guide

Mr. Anil Sharma
Centre Coordinator

ABSTRACT

The Business-to-Consumer (B2C) e-commerce model has evolved significantly since its inception and has expanded across multiple product categories. However, the online sale of **fresh agricultural produce** such as fruits and vegetables remains relatively underdeveloped. With increasing consumer awareness regarding health, nutrition, and organic food consumption, the demand for fresh and locally sourced products has grown substantially. This creates an opportunity to bridge the gap between **farmers and consumers** through a digital marketplace.

This project focuses on the development of an **Online Fresh Food Marketplace**, which enables farmers to sell fresh fruits and vegetables directly to customers through an e-commerce platform. The system provides customers with access to a list of registered farmers, detailed product listings, a shopping cart for managing selections, secure order placement, and order history tracking. Farmers can manage their products and inventory, while administrators oversee user and system management.

The application is built using **Java (Spring Boot)** for backend development and **React** for frontend user interfaces. React enables client-side rendering and delivers a smooth single-page application experience, while Spring Boot handles secure RESTful APIs, business logic, and database interactions. **JWT-based authentication and role-based authorization** are implemented to ensure system security. **Razorpay** is integrated for secure online payment processing, and **MySQL** is used as the relational database for storing users, products, orders, and payment data.

The system follows a **multilevel architecture** with clear separation of concerns, ensuring high cohesion and low coupling. This architectural approach improves maintainability, scalability, and performance. The proposed solution provides a reliable digital platform that benefits both farmers and consumers by promoting transparency, convenience, and access to fresh agricultural products.

ACKNOWLEDGEMENT

I take this opportunity to express my sincere gratitude to **Almighty God** for bestowing His blessings and granting me the strength and perseverance to successfully complete this project.

I would like to express my heartfelt thanks to my esteemed project guide, **Mr. Prithviraj Shinde**, for his invaluable guidance, continuous encouragement, and constructive feedback throughout the development of this project. His expertise and timely suggestions played a crucial role in shaping this work and steering it in the right direction.

I am also grateful to our respected **Centre Coordinator, Mr. Anil Sharma**, for providing the necessary facilities and support that enabled the smooth execution of this project. I extend my sincere thanks to all the faculty members for their guidance, cooperation, and support during the course of this project.

Finally, I would like to thank my **family and friends** for their constant motivation, understanding, and moral support, which helped me stay focused and complete this work successfully.

Shrikant S. Londe (250841220089)

Vishwajit N. Nikam (250841220209)

Table of Contents

Sr. No	Description	Page No
1	Introduction	1
2	SRS	6
3	Diagrams	12
3.1	ER Diagram	12
3.2	Use Case Diagram	13
3.3	Data Flow Diagram	14
3.4	Activity Diagram	16
3.5	Class Diagram	19
3.6	Sequence Diagram	20
4	Database Design	21
5	Snapshots	25
6	Conclusion	30
7	References	31

1. INTRODUCTION

The **Online Marketplace Application** is a comprehensive Business-to-Consumer (B2C) e-commerce platform designed specifically for the online sale of **fresh fruits and vegetables directly from farmers to customers**. With the growing awareness of health, organic produce, and farm-fresh consumption, this platform aims to bridge the gap between farmers and end consumers by providing a secure, efficient, and user-friendly digital marketplace.

The system enables farmers to showcase their fresh produce, customers to browse and purchase products conveniently, and administrators to manage the overall platform operations. The application is built using **Spring Boot** for the backend and **React** for the frontend, ensuring a scalable, secure, and responsive system architecture. **MySQL** is used for data persistence, while **Razorpay** is integrated to provide secure and reliable online payment processing.

Security is a core focus of the platform. Authentication and authorization are implemented using **Spring Security with JWT (JSON Web Tokens)**, ensuring role-based access control and secure RESTful APIs. The system follows a **multi-tier architecture**, promoting loose coupling and high cohesion, which enhances maintainability, scalability, and performance.

1.1 System Roles Overview

The Online Marketplace Application supports three primary roles:

1. Admin

The Admin role has full control over the platform and is responsible for managing users, monitoring system activities, and ensuring smooth operation of the marketplace.

2. Farmer

Farmers are registered sellers who can list fresh fruits and vegetables, manage product inventory, and track customer orders.

3. Customer

Customers can browse available products, place orders, make secure payments, and view their order history.

This role-based structure ensures clear separation of responsibilities and secure access to system functionalities.

1.2 Admin Role

The **Admin** is the highest authority in the system, responsible for supervising platform operations and maintaining system integrity.

Responsibilities:-

- **User Management**
  - Create, update, and deactivate user accounts.
  - Assign roles such as Admin, Farmer, or Customer.
- **Product Oversight**
  - Monitor products listed by farmers.
  - Ensure quality and compliance with platform policies.
- **Order Monitoring**
  - View and track all orders placed on the platform.
  - Resolve disputes and oversee order statuses.
- **Payment Supervision**
  - Monitor transactions processed through Razorpay.
  - Handle refunds and payment issues.
- **System Configuration**
  - Manage platform settings and system rules.
- **Security Management**
  - Enforce authentication and authorization policies using Spring Security.
  - Monitor logs and prevent unauthorized access.
- **Reporting & Analytics**
  - Generate reports related to users, orders, payments, and platform usage.

Access Level

- Full access to all system features and data.
- Permission to override system settings and user actions.

1.3 Farmer Role

Farmers act as sellers who provide fresh produce directly to customers through the platform.

Responsibilities:-

- **Product Management**
  - Add, update, and remove product listings.
  - Manage product prices, availability, and descriptions.
- **Order Management**
  - View orders placed for their products.
  - Update order status (e.g., accepted, dispatched).
- **Inventory Control**
  - Maintain stock availability for products.
- **Profile Management**
  - Update personal and business details.
  -

Access Level

- Limited to managing own products and orders.
- No access to system-wide settings or other users' data.

1.4 Customer Role

Customers are end users who purchase fresh produce from farmers.

Responsibilities:-

- **Product Browsing**
  - View farmers and available products.

    - Search and filter products.
- **Order Placement**
    - Add products to cart and place orders.
    - View order history and order status.
- **Payment Processing**
    - Make secure online payments using Razorpay.
    - View payment history and receipts.
- **Profile Management**
    - Update personal details such as address and contact information.

Access Level

- Restricted to personal data and shopping activities.
- No access to administrative or seller functionalities.

1.5 Role-Based Access Control (RBAC)

The system implements **Role-Based Access Control (RBAC)** using **Spring Security and JWT**, ensuring that users can only access resources permitted by their roles.

- **Admins** have unrestricted access.
- **Farmers** can manage only their products and orders.
- **Customers** can browse, order, and manage their accounts.

This mechanism enhances system security and prevents unauthorized access.

1.6 Purpose

The purpose of the Online Marketplace Application is to provide a **secure, scalable, and efficient platform** that enables direct trade between farmers and consumers. The system simplifies product discovery, order processing, and payment handling while promoting transparency and fresh food accessibility.

1.7 Scope

The scope of the project includes:

- Design and development of a full-stack web application
- Secure user authentication and authorization
- Product, order, and payment management
- Integration with third-party payment gateway (Razorpay)
- Deployment-ready REST APIs
- Support for future scalability and feature expansion

1.8 Objectives of the Online Marketplace Application

1. Enable direct farmer-to-customer transactions
2. Provide efficient product and order management
3. Ensure secure authentication and payments
4. Improve customer experience through a responsive UI
5. Maintain data integrity and system security
6. Support scalability and future enhancements
7. Reduce intermediaries and promote fair pricing

1.9 Key Functionalities

- User Registration and Login (JWT-based)
- Role-based Authorization
- Product Management by Farmers
- Order Placement and Tracking
- Secure Online Payments via Razorpay
- Admin Monitoring and Control
- Reporting and System Logs
- RESTful API Integration

2. SOFTWARE REQUIREMENT SPECIFICATION

The Software Requirement Specification (SRS) defines the functional and non-functional requirements of the **Online Marketplace Application for Fresh Food Products**. These requirements describe the expected behavior, constraints, and performance characteristics of the system. The SRS serves as a foundation for system design, development, testing, and validation, ensuring that the final product meets user needs and business objectives.

2.1 Functional Requirements for Online Marketplace Application

1. User Management

1.1 User Registration

- The system shall allow new users to register by providing details such as name, email, password, and role (Farmer or Customer).
- The system shall validate user inputs and prevent duplicate email registrations.

1.2 User Authentication

- The system shall authenticate users using email and password.
- The system shall generate a JWT token upon successful login.
- The system shall allow users to securely log out.

1.3 Role-Based Access Control

- The system shall support role-based access for Admin, Farmer, and Customer.
- The system shall restrict access to APIs and functionalities based on assigned roles.

1.4 Profile Management

- Users shall be able to view and update their profile details.
- Users shall be able to change their passwords securely.

2. Product Management (Farmer)

2.1 Product Listing

- The system shall allow farmers to add fresh food products such as fruits and vegetables.
- The system shall store product details including name, description, price, quantity, category, and availability.

2.2 Product Update and Removal

- Farmers shall be able to update product details.
- Farmers shall be able to remove products from the marketplace.

2.3 Product Visibility

- The system shall display only active and available products to customers.

3. Order Management

3.1 Order Placement

- Customers shall be able to add products to a cart and place orders.
- The system shall validate product availability before order placement.

3.2 Order Processing

- The system shall create an order record after successful order placement.
- The system shall update order status (CREATED, CONFIRMED, SHIPPED, DELIVERED).

3.3 Order Tracking

- Customers shall be able to view and track their order status.
- Farmers shall be able to view orders related to their products.

3.4 Order History

- The system shall maintain order history for customers and farmers.

4. Payment Management

4.1 Payment Processing

- The system shall integrate with Razorpay to process online payments securely.

- The system shall support payment methods such as UPI, credit cards, debit cards, and wallets.

4.2 Payment Verification

- The system shall verify payment authenticity using Razorpay signature verification.
- The system shall update payment status as CREATED, SUCCESS, or FAILED.

4.3 Payment Records

- The system shall store payment details for future reference and auditing.

5. Admin Management

5.1 User Monitoring

- Admins shall be able to view and manage all users.
- Admins shall be able to activate or deactivate user accounts.

5.2 System Oversight

- Admins shall be able to view all products, orders, and payments.
- Admins shall be able to monitor system activities.

6. Reporting and Analytics

6.1 Sales Reports

- The system shall generate reports on total sales and order counts.

6.2 User Reports

- The system shall generate reports related to registered users and role distribution.

7. Security Requirements

7.1 Data Protection

- The system shall encrypt sensitive user data such as passwords.
- The system shall never store plain-text passwords.

7.2 Authentication & Authorization

- The system shall enforce authentication for protected APIs.
- The system shall authorize users based on roles using Spring Security and JWT.

2.2 Non-Functional Requirements

1. Performance Requirements

1.1 Response Time

- The system shall respond to user requests within **2 seconds** under normal conditions.

1.2 Scalability

- The system shall support growth in users and transactions without performance degradation.
- The system shall support at least **5,000 concurrent users**.

1.3 Throughput

- The system shall process multiple orders and payments efficiently during peak usage.

2. Reliability Requirements

2.1 Availability

- The system shall be available **24×7** with minimal downtime.

2.2 Fault Tolerance

- The system shall handle failures gracefully without data loss.

2.3 Error Handling

- The system shall provide meaningful error messages for failures and exceptions.

3. Usability Requirements

- The system shall provide a clean, intuitive, and responsive user interface.
- The system shall support desktop and mobile browsers.

4. Maintainability Requirements

4.1 Modularity

- The system shall follow a layered architecture (Controller, Service, Repository).
- Components shall be loosely coupled.

4.2 Code Quality

- The system shall follow coding standards and best practices.
- The code shall be readable, maintainable, and well-documented.

4.3 Testing

- The system shall undergo unit testing, integration testing, and system testing.

2.3 Other Requirements

Hardware and Network Interfaces

Backend Server Configuration

- Processor: Intel / AMD Multi-core Processor
- RAM: Minimum 8 GB
- Storage: Minimum 20 GB

Frontend Client Configuration

- Processor: Intel / AMD Processor
- RAM: Minimum 4 GB
- Browser: Chrome / Firefox / Edge

Software Interfaces

Backend Software Configuration

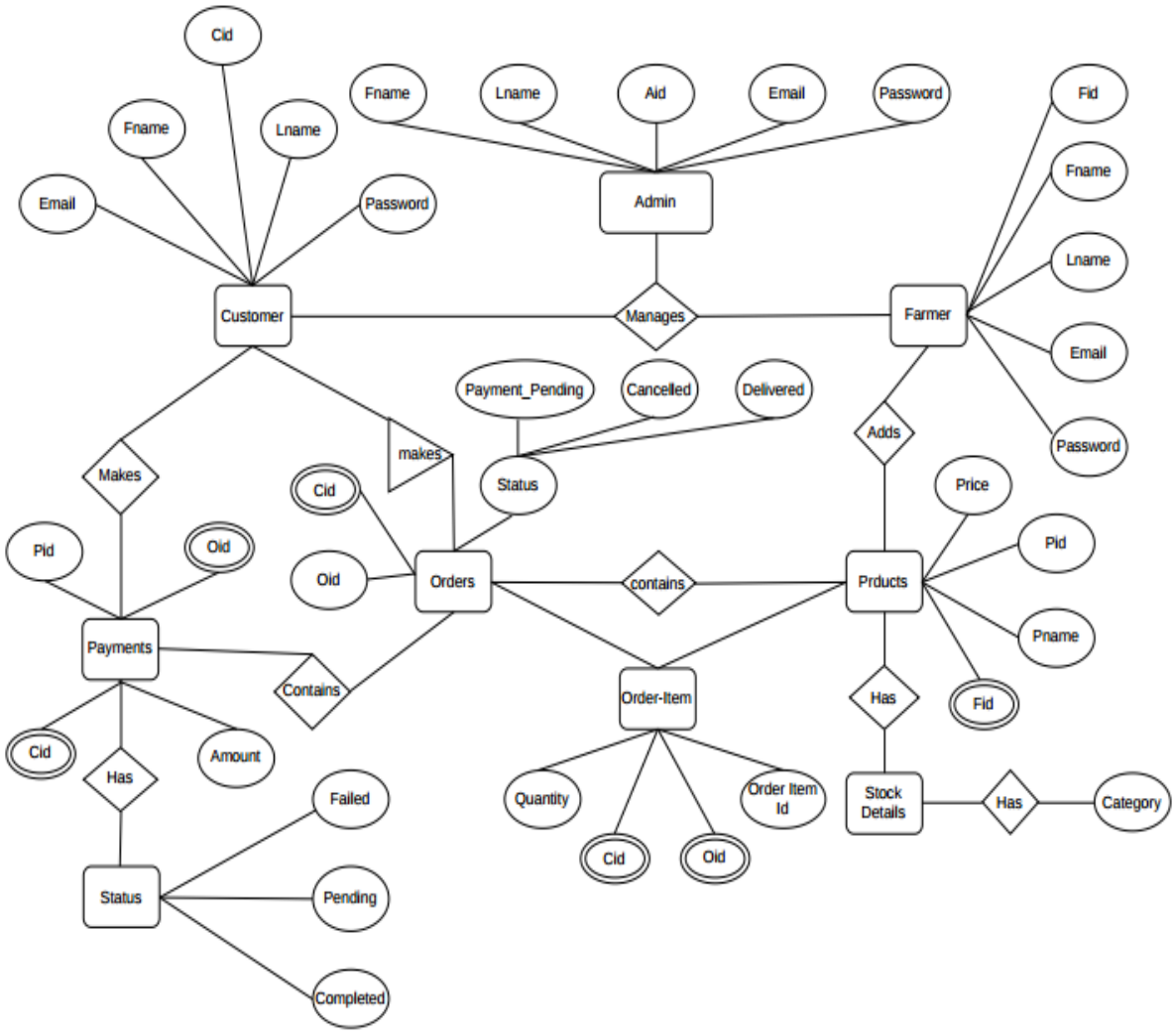
- Java (JDK 17 or above)
- Spring Boot v 4.0.2
- Spring Security with JWT
- JPA / Hibernate v7.2.3
- Razorpay Payment Gateway v3
- MySQL Database v8.045
- IntelliJ IDEA / STS

Frontend Software Configuration

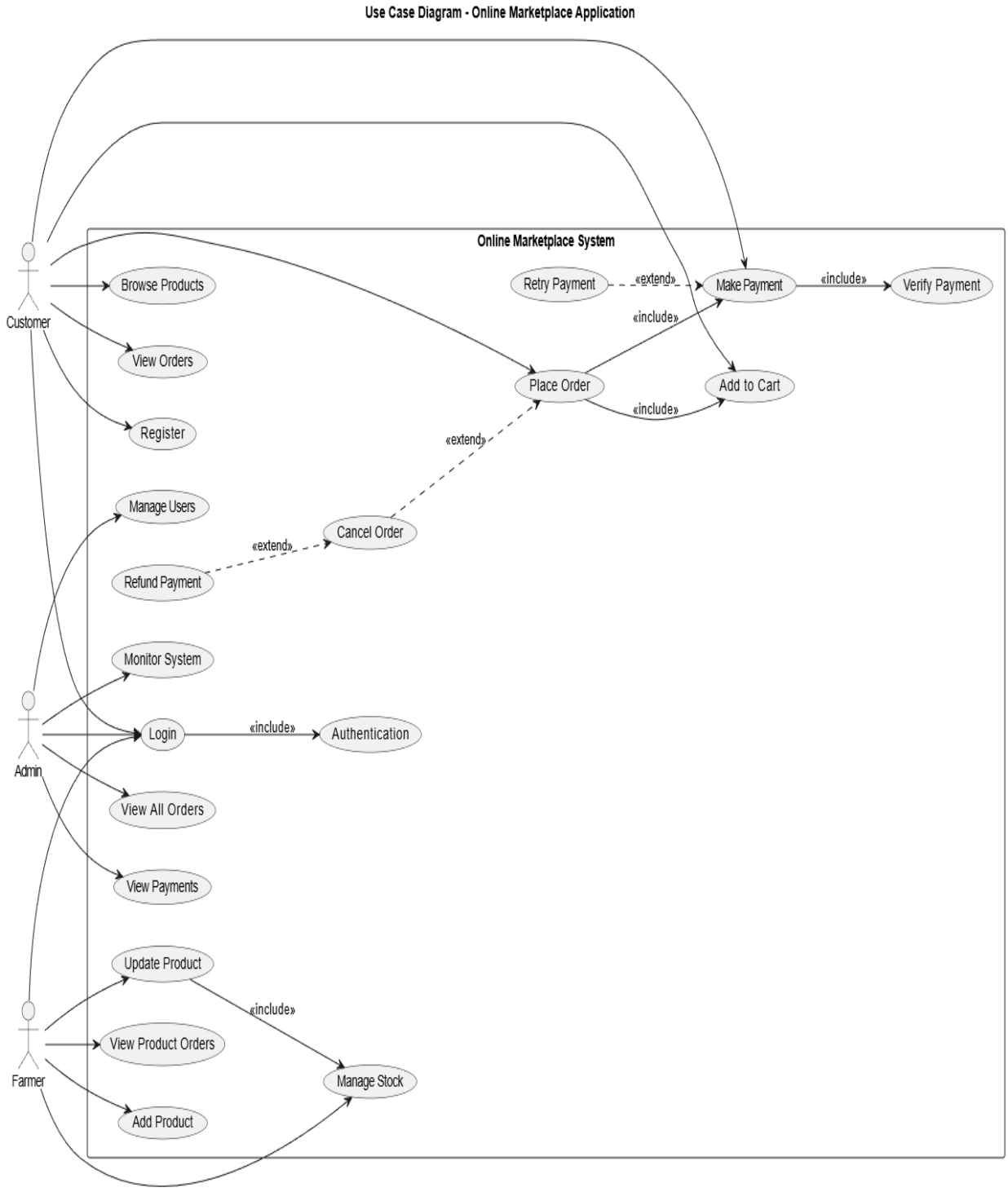
- ReactJS v18
- HTML5, CSS3, JavaScript
- Bootstrap v5/ Tailwind CSS v3.1
- VS Code

3. DIAGRAMS

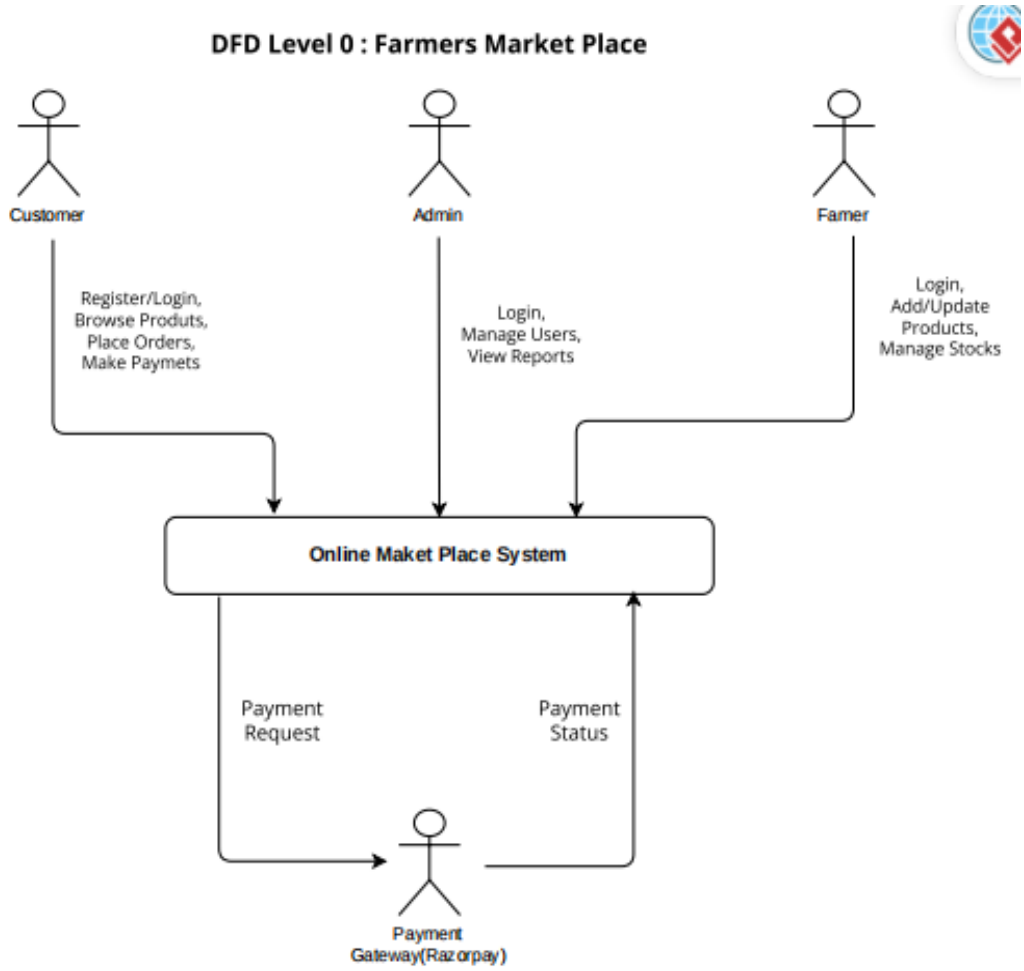
3.1 Entity Relationship Diagram



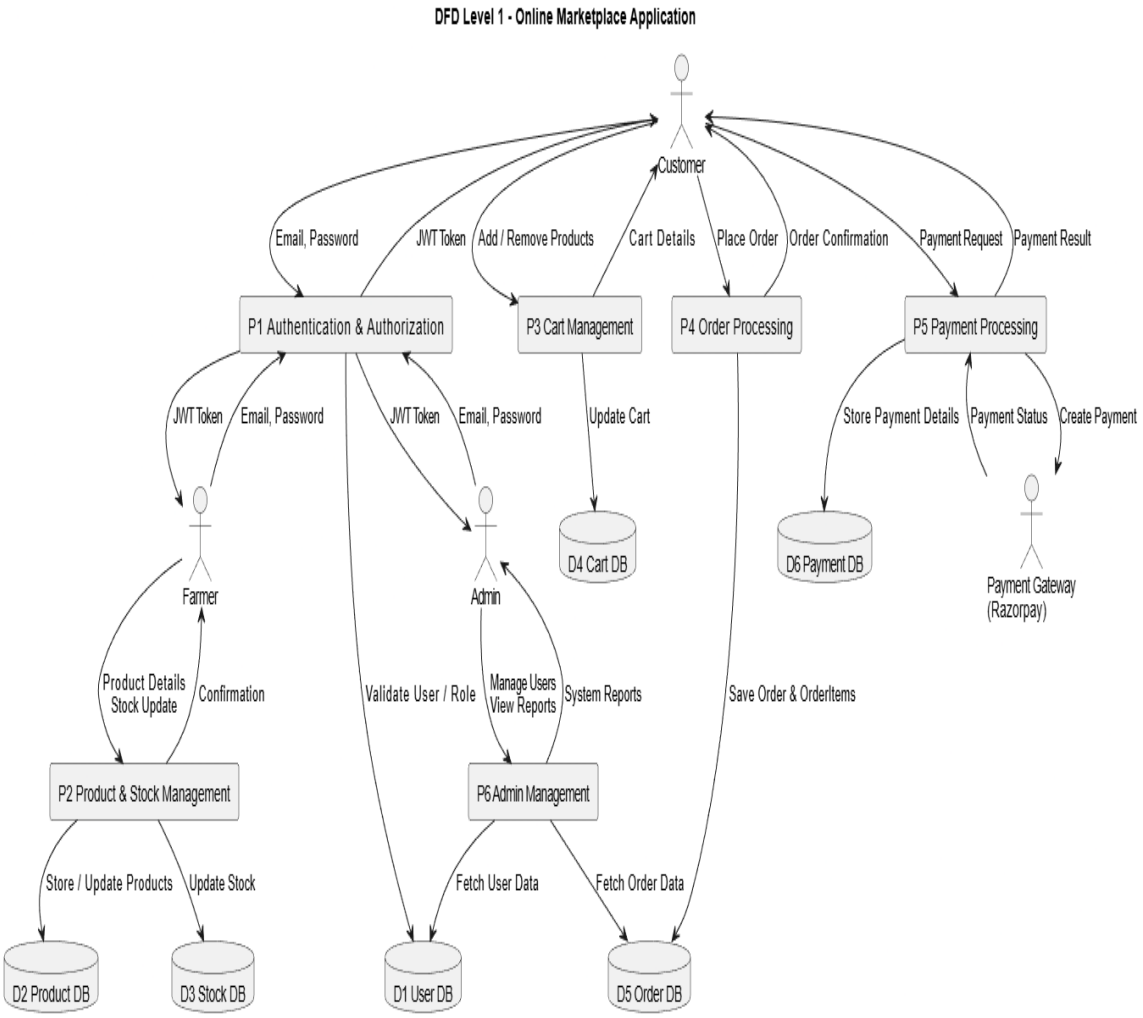
3.2Use case Diagram



3.3 Data Flow Diagram

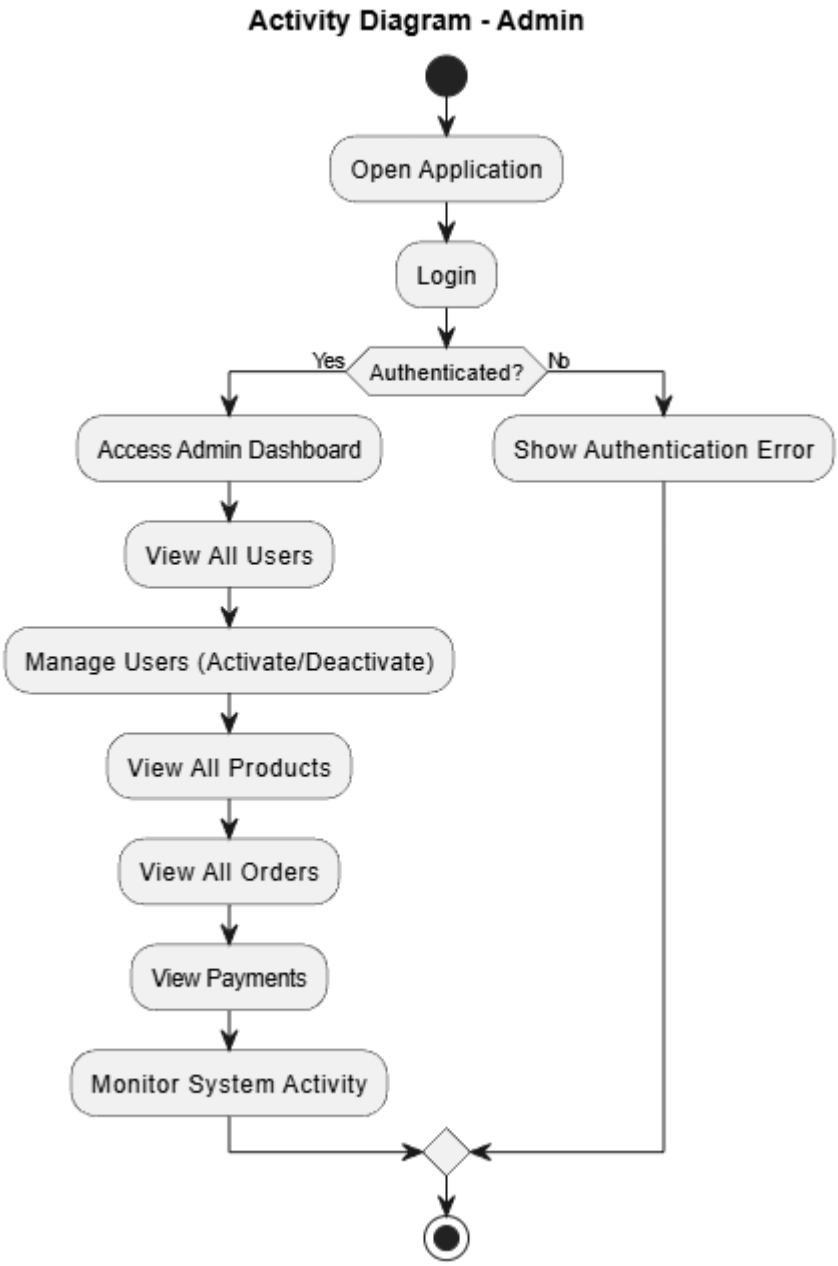


Level 0 : Data Flow Diagram

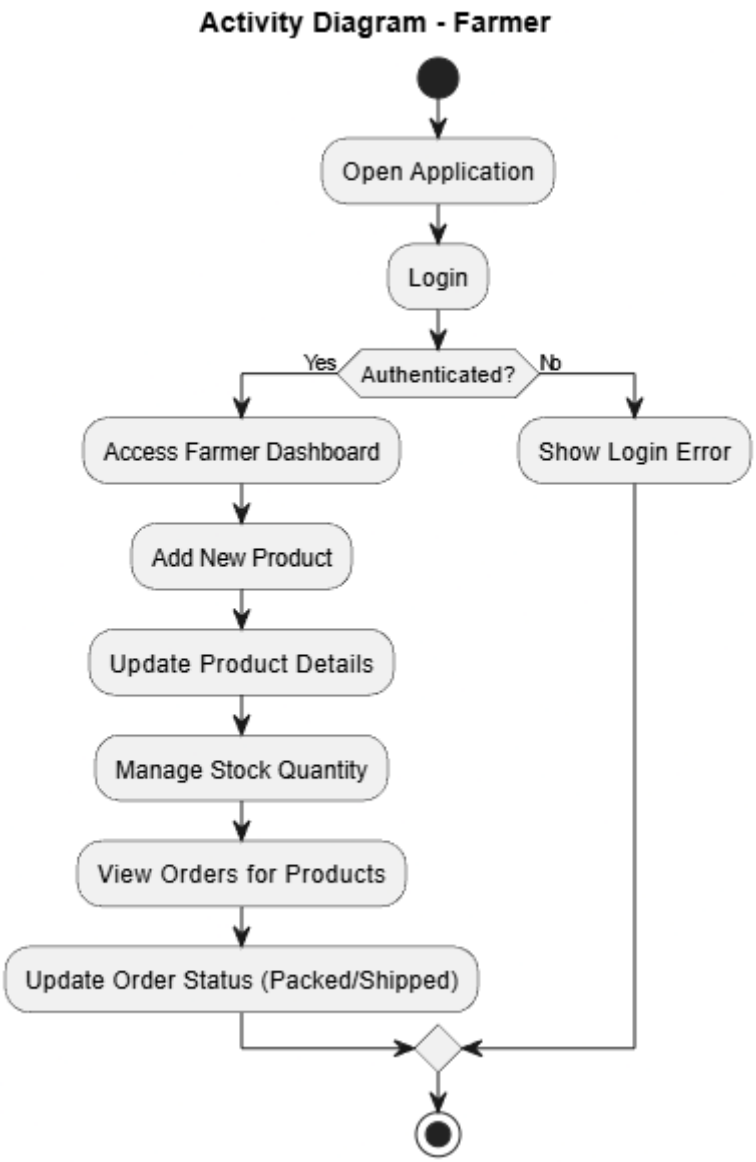


Level 1: Data Flow Diagram

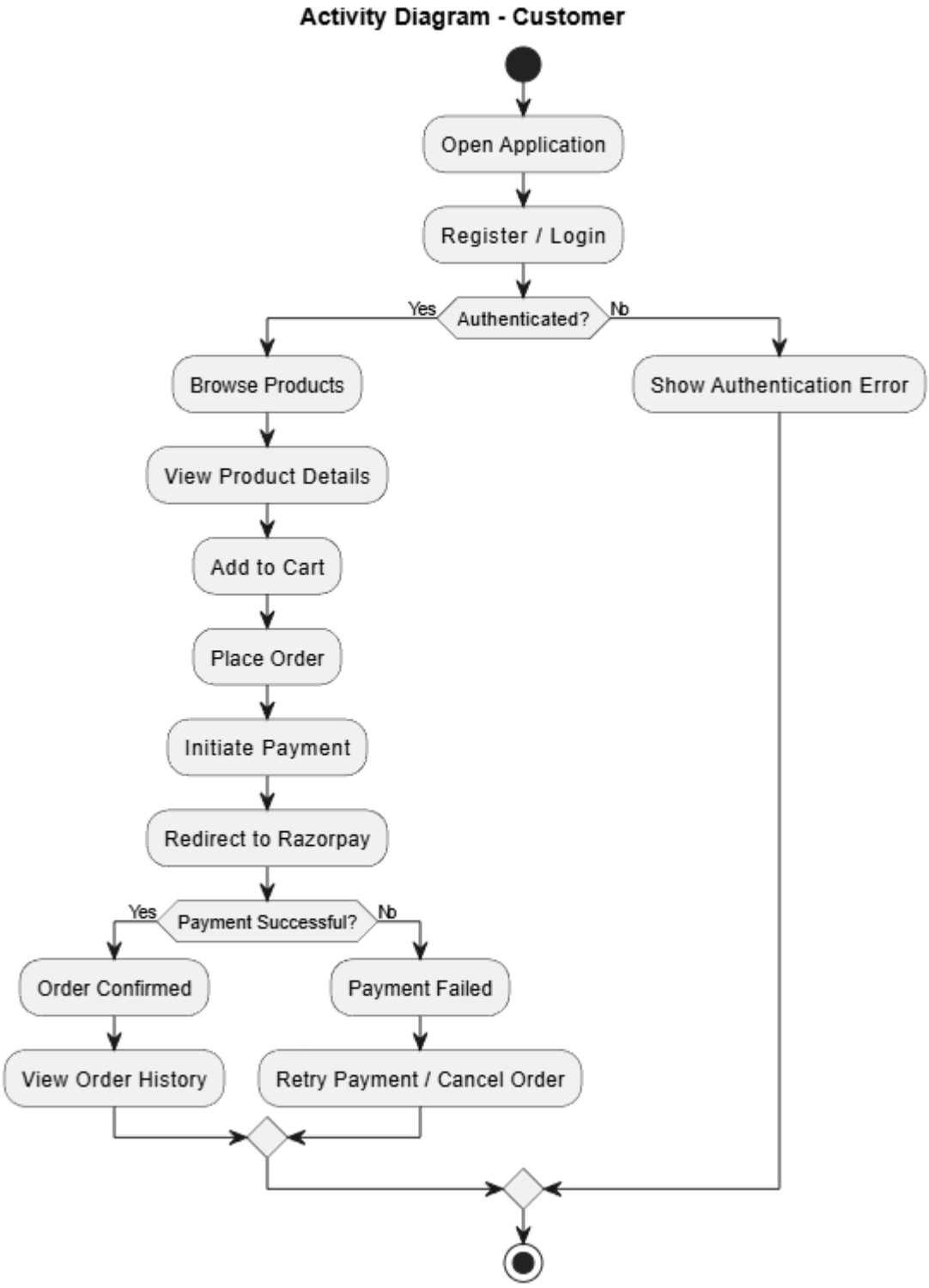
3.4 Activity Diagrams:



Activity Diagram For Admins

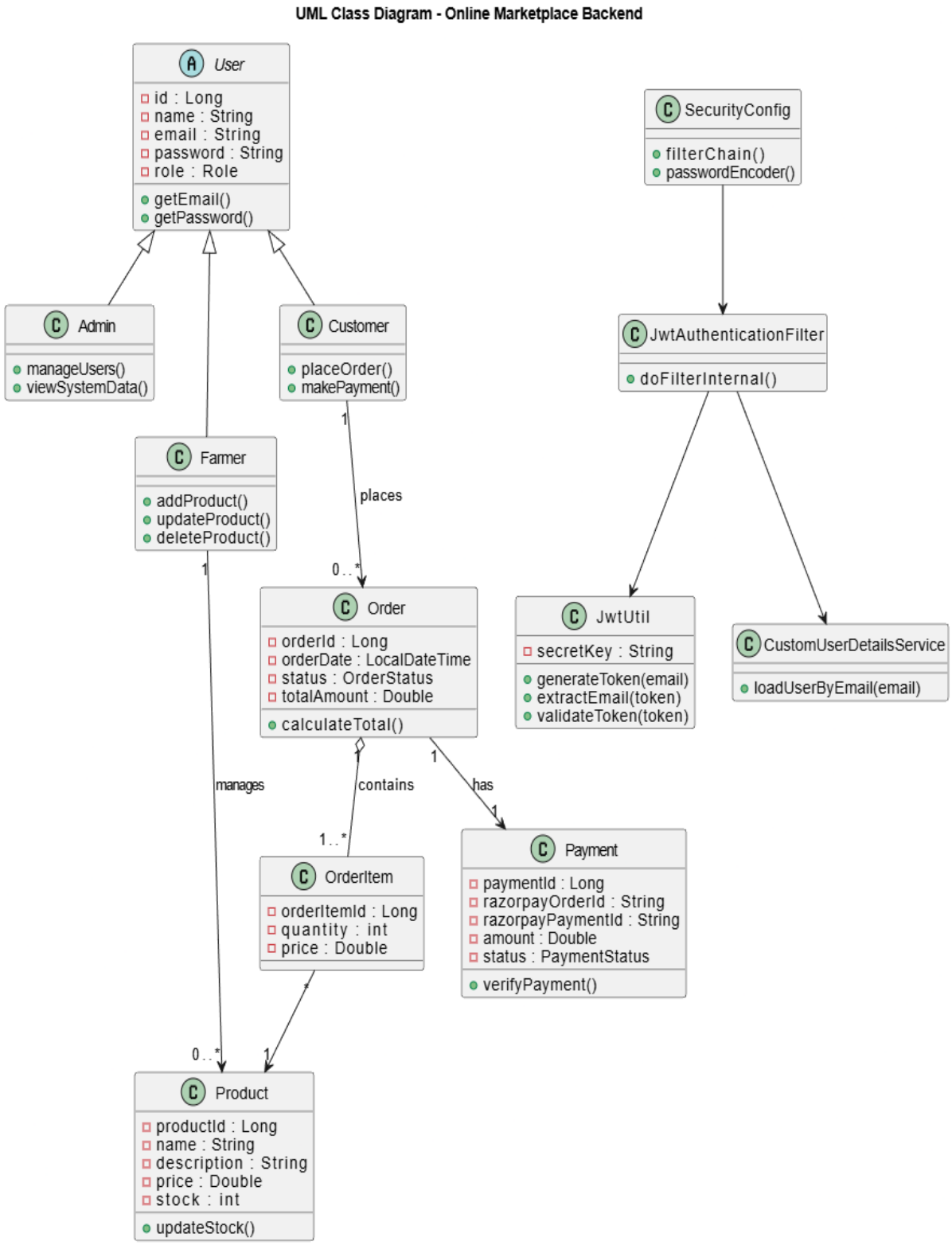


Activity Diagram for Farmer

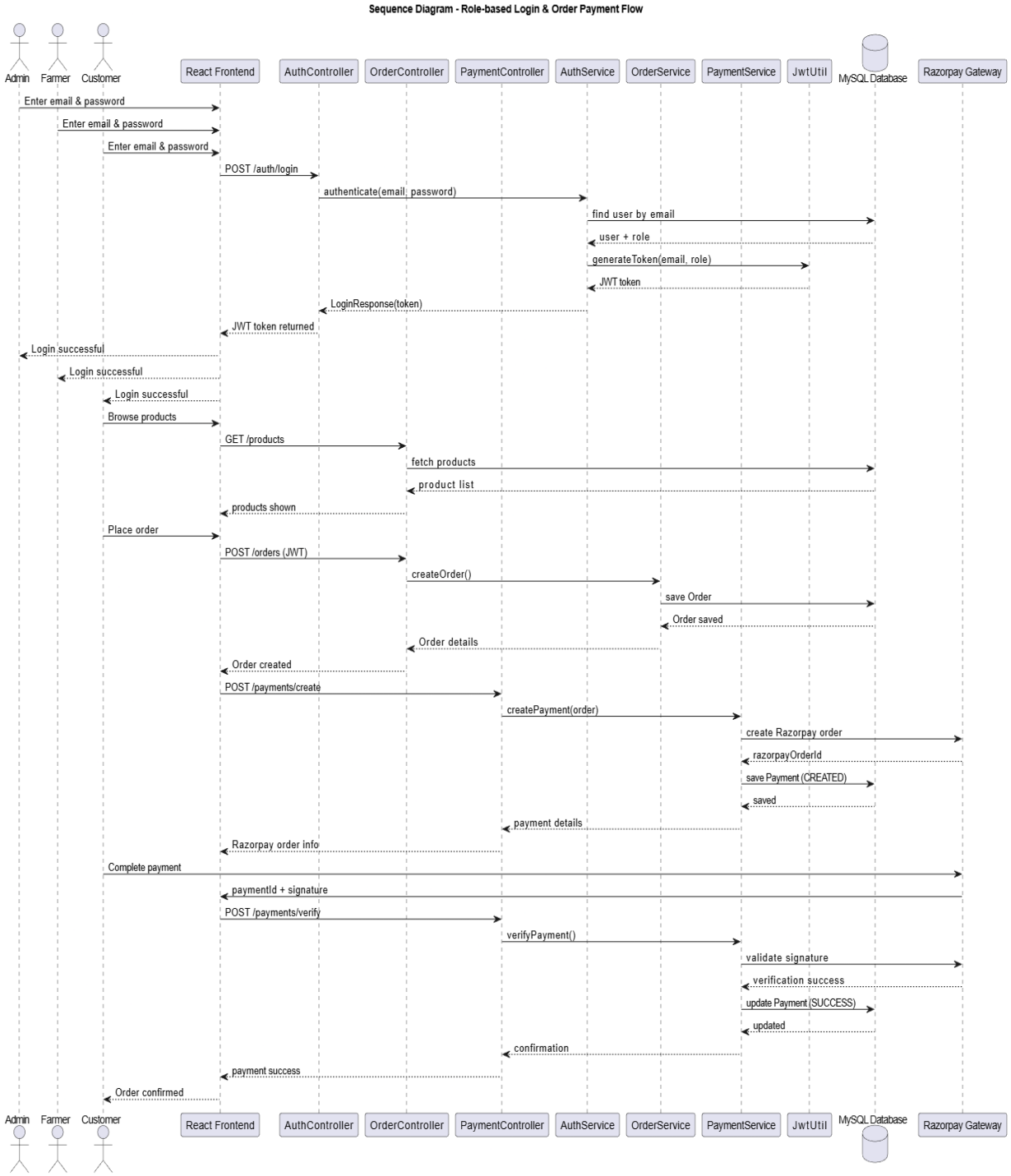


Activity Diagram For Customers

3.5 Class Diagram

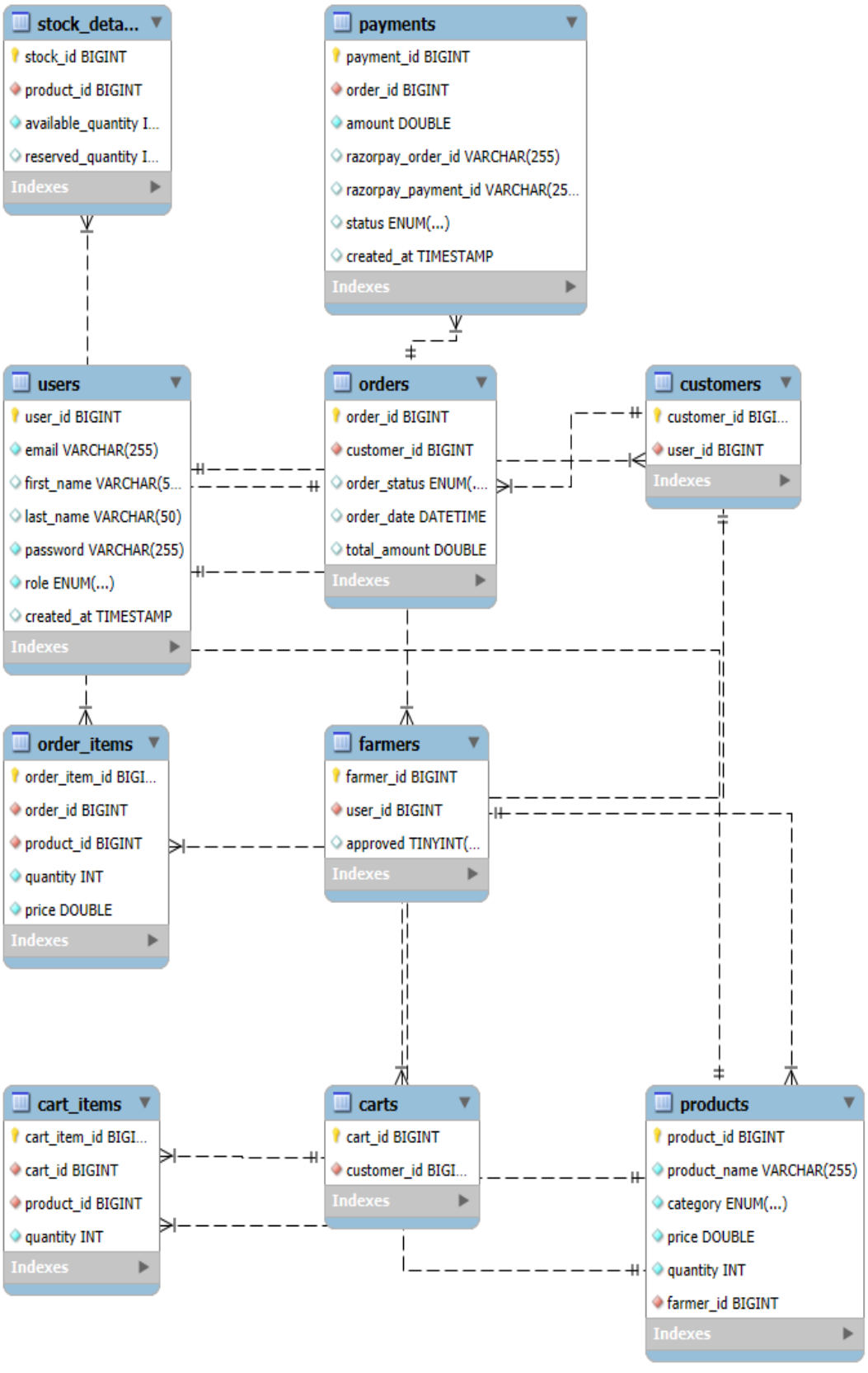


3.6 Sequence Diagram



4. Database Design

4.1 Database Schema:



4.2 Tables

The following Structure depict the Database Design.

```
mysql> use farmers_market_place;
Database changed
mysql> show tables;
+-----------------------------------+
| Tables_in_farmers_market_place    |
+-----------------------------------+
| cart_items                        |
| carts                             |
| customers                         |
| farmers                           |
| order_items                       |
| orders                            |
| payments                          |
| products                          |
| stock_details                     |
| users                             |
+-----------------------------------+
10 rows in set (0.15 sec)
```

Table 1: Users

```
mysql> desc users;
+------------+----------------------------------+------+-----+-------------------+-------------------+
| Field      | Type                             | Null | Key | Default           | Extra             |
+------------+----------------------------------+------+-----+-------------------+-------------------+
| user_id    | bigint                           | NO   | PRI | NULL              | auto_increment    |
| email      | varchar(255)                     | NO   | UNI | NULL              |                   |
| first_name | varchar(50)                      | YES  |     | NULL              |                   |
| last_name  | varchar(50)                      | YES  |     | NULL              |                   |
| password   | varchar(255)                     | NO   |     | NULL              |                   |
| role       | enum('ADMIN','FARMER','CUSTOMER') | NO   |     | NULL              |                   |
| created_at | timestamp                        | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
+------------+----------------------------------+------+-----+-------------------+-------------------+
7 rows in set (0.07 sec)
```

Table 2: Customers

```
mysql> desc Customers;
+-------------+--------+------+-----+---------+----------------+
| Field       | Type   | Null | Key | Default | Extra          |
+-------------+--------+------+-----+---------+----------------+
| customer_id | bigint | NO   | PRI | NULL    | auto_increment |
| user_id     | bigint | NO   | UNI | NULL    |                |
+-------------+--------+------+-----+---------+----------------+
2 rows in set (0.01 sec)
```

Table 3: Farmers

```
mysql> desc Farmers;
+-----------+------------+------+-----+---------+----------------+
| Field     | Type       | Null | Key | Default | Extra          |
+-----------+------------+------+-----+---------+----------------+
| farmer_id | bigint     | NO   | PRI | NULL    | auto_increment |
| user_id   | bigint     | NO   | UNI | NULL    |                |
| approved  | tinyint(1) | YES  |     | 0       |                |
+-----------+------------+------+-----+---------+----------------+
3 rows in set (0.03 sec)
```

Table 4: Products

```
mysql> desc Products;
+--------------+-----------------------------------------+------+-----+---------+----------------+
| Field        | Type                                    | Null | Key | Default | Extra          |
+--------------+-----------------------------------------+------+-----+---------+----------------+
| product_id   | bigint                                  | NO   | PRI | NULL    | auto_increment |
| product_name | varchar(255)                            | NO   |     | NULL    |                |
| category     | enum('VEGETABLE','FRUIT','GRAIN','DAIRY') | NO   |     | NULL    |                |
| price        | double                                  | NO   |     | NULL    |                |
| quantity     | int                                     | NO   |     | NULL    |                |
| farmer_id    | bigint                                  | NO   | MUL | NULL    |                |
+--------------+-----------------------------------------+------+-----+---------+----------------+
6 rows in set (0.01 sec)
```

Table 5: Orders

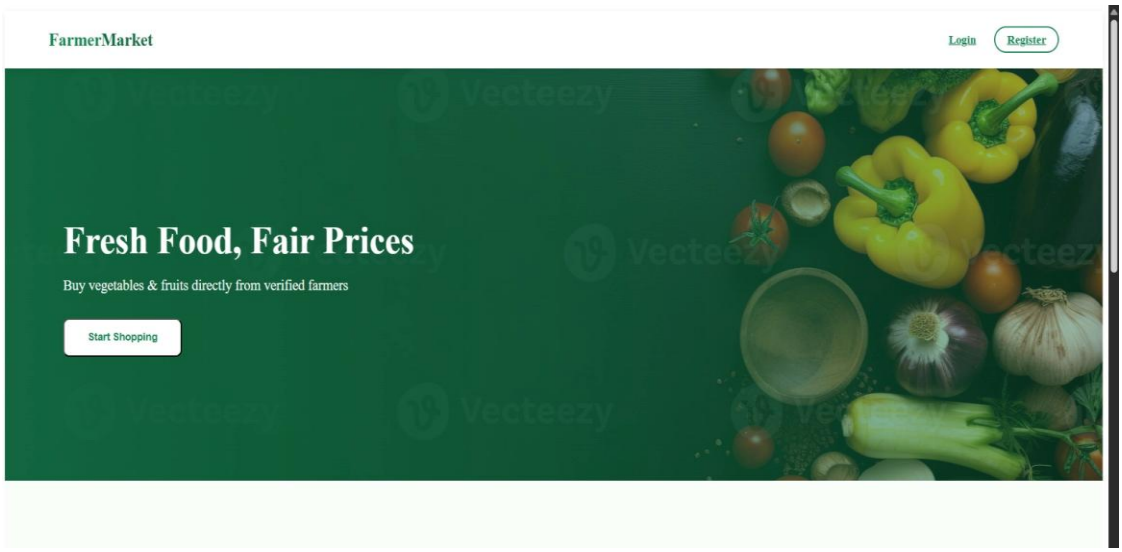
```
mysql> desc orders;
+--------------+-------------------------------------------------------------+------+-----+-------------------+-------------------+
| Field        | Type                                                        | Null | Key | Default           | Extra             |
+--------------+-------------------------------------------------------------+------+-----+-------------------+-------------------+
| order_id     | bigint                                                      | NO   | PRI | NULL              | auto_increment    |
| customer_id  | bigint                                                      | NO   | MUL | NULL              |                   |
| order_status | enum('CREATED','PAID','SHIPPED','DELIVERED','CANCELLED')    | YES  |     | NULL              |                   |
| order_date   | datetime                                                    | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
| total_amount | double                                                      | YES  |     | NULL              |                   |
+--------------+-------------------------------------------------------------+------+-----+-------------------+-------------------+
5 rows in set (0.01 sec)
```

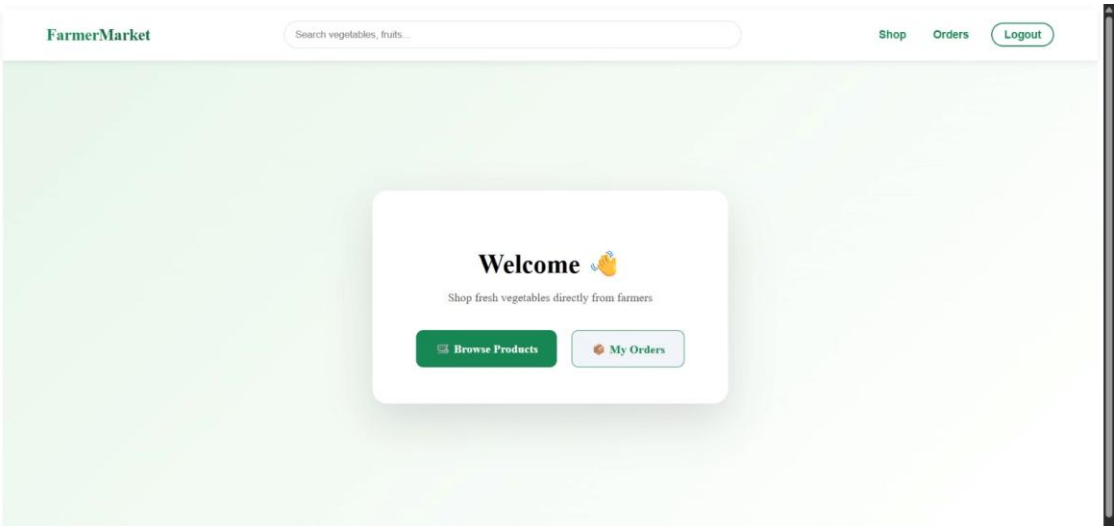
Table 6: Payments

```
mysql> desc payments;
+--------------------+-----------------------------------+------+-----+-------------------+-------------------+
| Field              | Type                              | Null | Key | Default           | Extra             |
+--------------------+-----------------------------------+------+-----+-------------------+-------------------+
| payment_id         | bigint                            | NO   | PRI | NULL              | auto_increment    |
| order_id           | bigint                            | NO   | UNI | NULL              |                   |
| amount             | double                            | NO   |     | NULL              |                   |
| razorpay_order_id  | varchar(255)                      | YES  |     | NULL              |                   |
| razorpay_payment_id | varchar(255)                     | YES  |     | NULL              |                   |
| status             | enum('CREATED','SUCCESS','FAILED') | YES  |     | NULL              |                   |
| created_at         | timestamp                         | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
+--------------------+-----------------------------------+------+-----+-------------------+-------------------+
7 rows in set (0.01 sec)
```

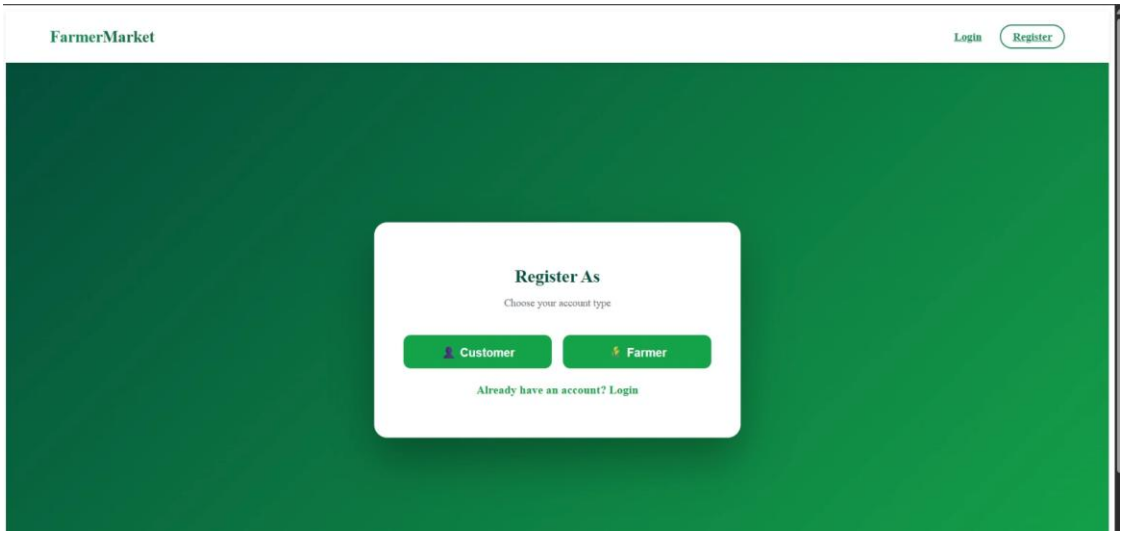
5. Snapshots

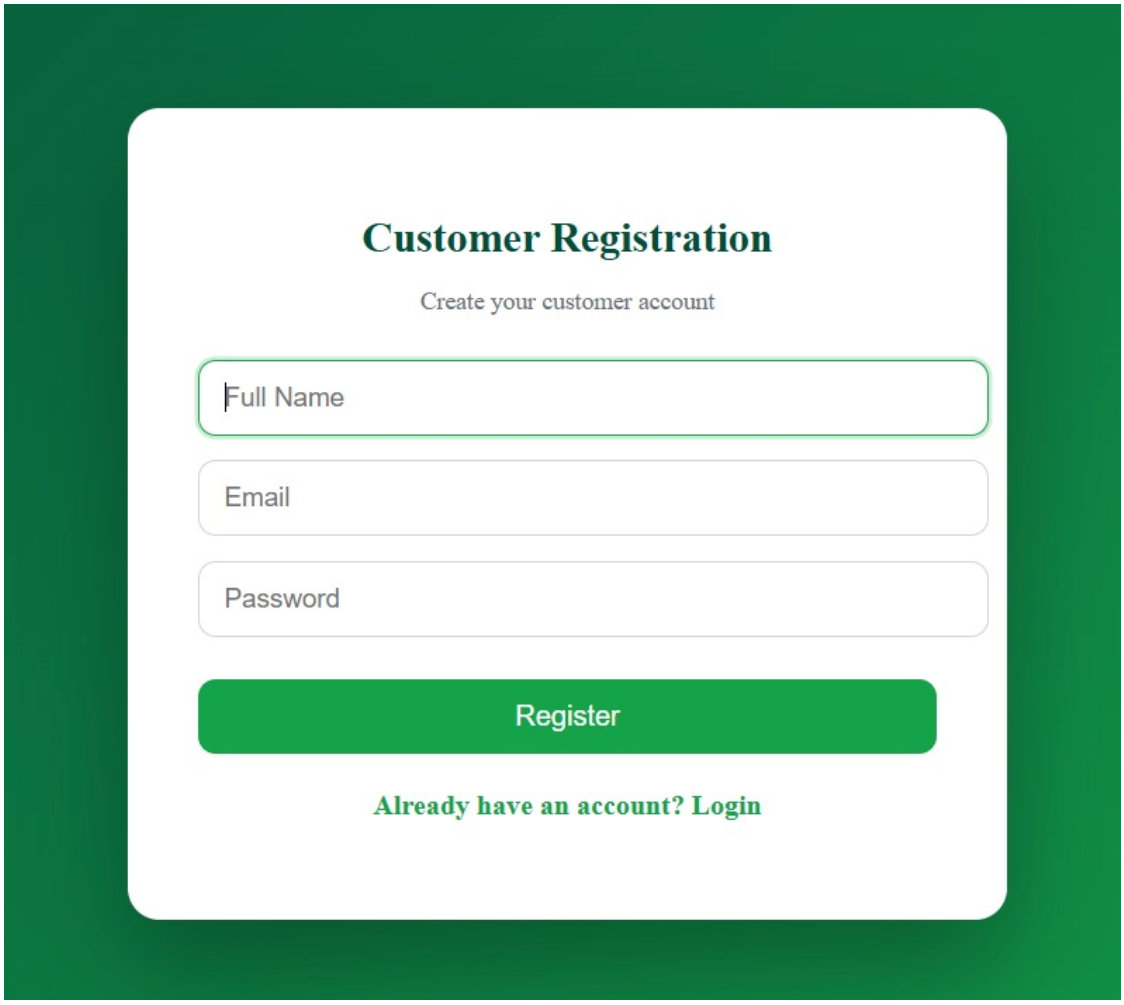
Home Page:



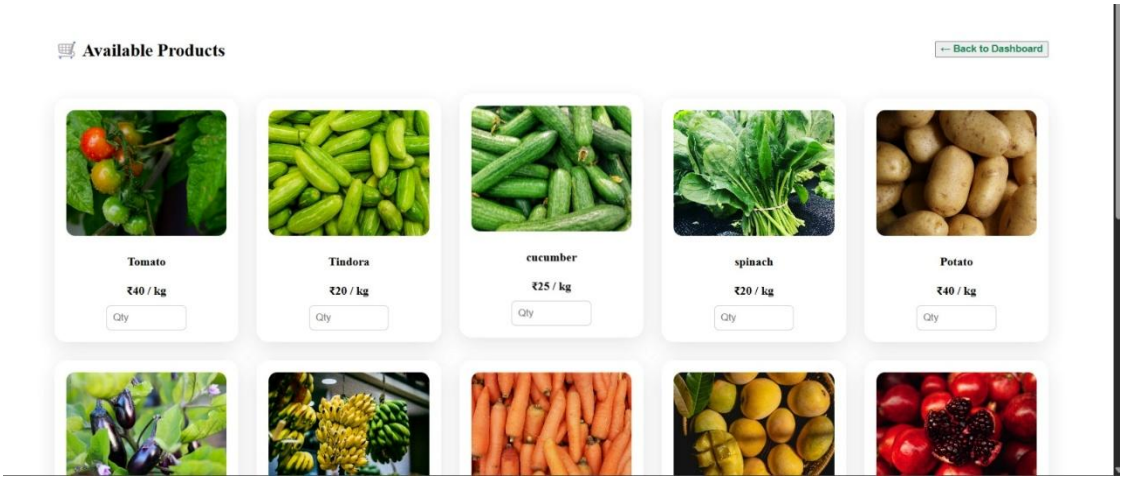


Register Page:

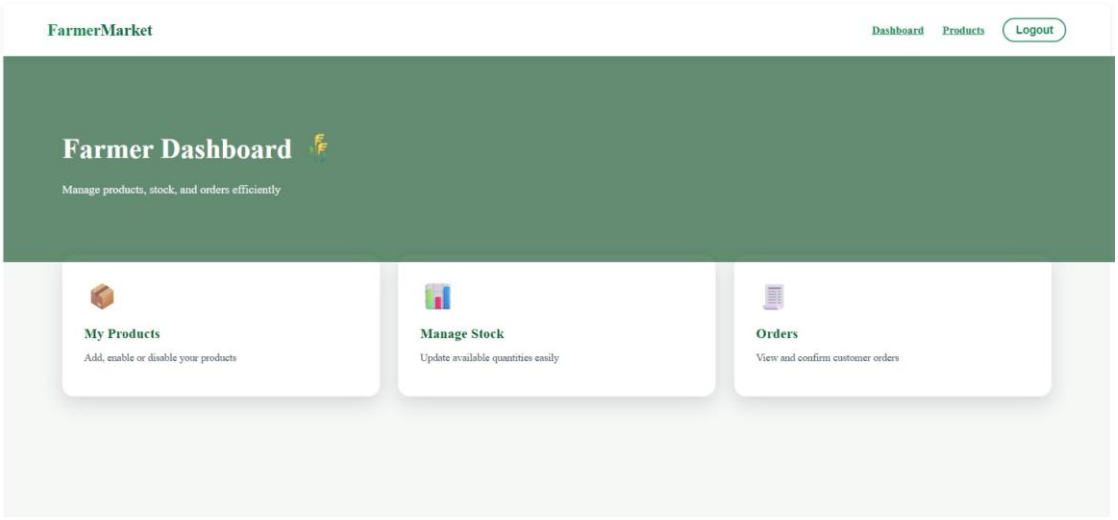




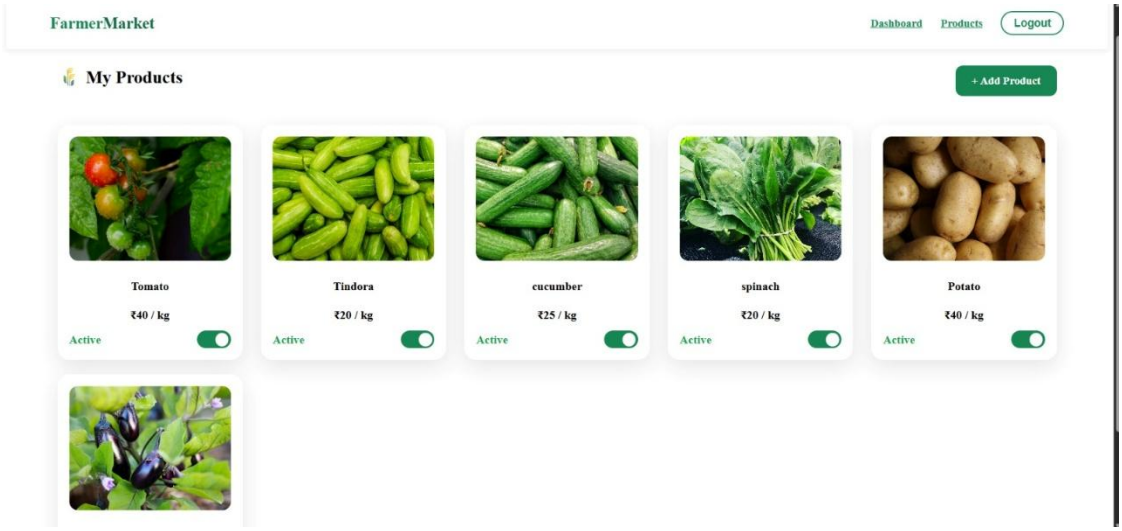
Available Products:



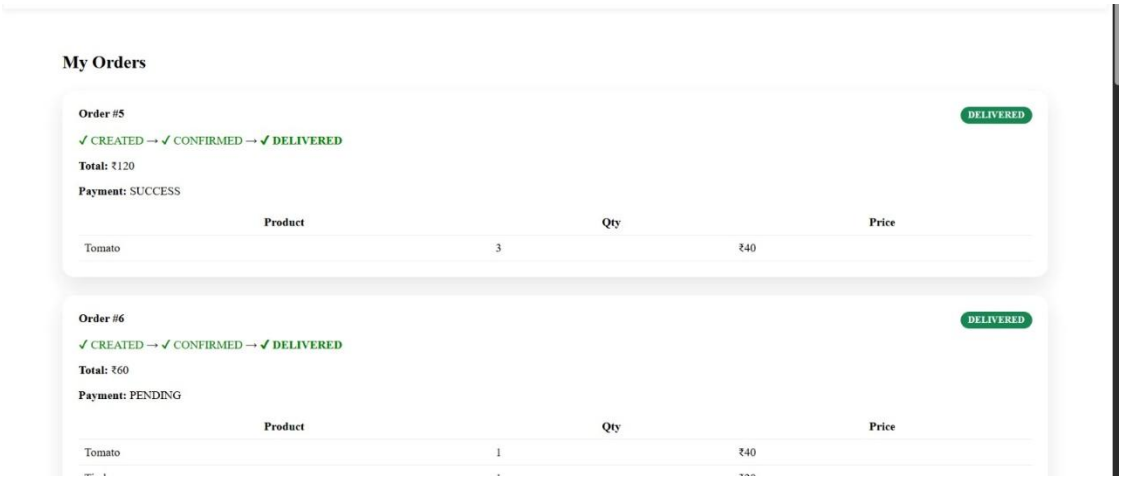
Farmer Dashboard:



Farmers Products:



Customers Orders:



6. Conclusions

In conclusion, the *Farmers Market Place* project provides an effective digital platform that bridges the gap between farmers and customers by enabling direct buying and selling of agricultural products. The system simplifies product listing, order management, and payment processing, thereby reducing dependency on intermediaries and promoting transparency in agricultural trade.

The application is designed using a modular and scalable architecture that ensures smooth interaction among farmers, customers, and administrators. Farmers can efficiently manage their products and track orders, customers can browse and purchase fresh produce with ease, and administrators can monitor and regulate system operations. The use of modern technologies such as Spring Boot, RESTful services, and MySQL ensures reliability, maintainability, and efficient data management.

Security and data integrity have been given significant importance through authentication mechanisms and controlled access to system functionalities. This ensures safe transactions and protects sensitive user information. The project not only fulfils current functional requirements but also provides flexibility for future enhancements such as advanced analytics, mobile application integration, and support for additional payment gateways.

Overall, the *Farmers Market Place* project demonstrates a practical and sustainable solution for modernizing agricultural commerce. Its user-centric design, robust backend architecture, and adaptability to future technological advancements make it a valuable contribution toward building efficient and transparent digital marketplaces for the agricultural sector.

7. References

1. Spring Boot Official Documentation
   Spring Boot Reference Guide,
   https://spring.io/projects/spring-boot

2. Spring Framework Documentation
   Spring Framework Core Documentation,
   https://docs.spring.io/spring-framework/docs/current/reference/html/

3. Hibernate ORM Documentation
   Hibernate ORM User Guide,
   https://hibernate.org/orm/documentation/

4. Java Platform, Standard Edition Documentation
   Oracle Java SE Documentation,
   https://docs.oracle.com/en/java/javase/

5. MySQL Official Documentation
   MySQL 8.0 Reference Manual,
   https://dev.mysql.com/doc/

6. RESTful Web Services Architecture
   Fielding, R. T., *Architectural Styles and the Design of Network-based Software Architectures*,
   Doctoral Dissertation, University of California, Irvine, 2000

7. Razorpay Payment Gateway Documentation
   Razorpay API Reference,
   https://razorpay.com/docs/

8. UML and Software Design
   Booch, G., Rumbaugh, J., Jacobson, I.,
   The Unified Modelling Language User Guide, Addison-Wesley

9. IntelliJ IDEA Documentation
JetBrains IntelliJ IDEA Documentation,
https://www.jetbrains.com/idea/documentation/

10. Postman API Testing Tool
Postman Learning Centre,
https://learning.postman.com/