Crop Deal

A Project Report submitted in partial fulfilment of the requirements for the award of the degree of

Bachelor of Technology

in

Computer Science and Engineering

by

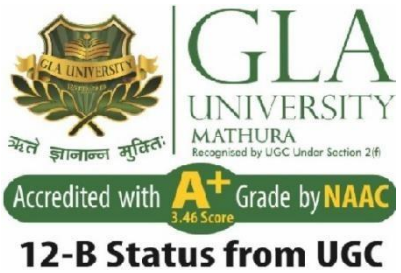
Divyanshu Rajpoot
2115000384

Under the Guidance of

Mrs. Samruddhi Keshav Zarapkar

Department of Computer Engineering & Applications

Institute of Engineering & Technology



GLA University
Mathura- 281406, INDIA

May, 2025

Department of Computer Engineering and Applications
GLA University, 17 km Stone, NH#2, Mathura-Delhi Road, P.O. Chaumuhan, Mathura-281406 (U.P.)

Declaration

I hereby declare that the work which is being presented in the B.Tech. Project **"Crop Deal"**, in partial fulfillment of the requirements for the award of the ***Bachelor of Technology*** in Computer Science and Engineering and submitted to the Department of Computer Engineering and Applications of GLA University, Mathura, is an authentic record of my own work carried under the supervision of Mrs. Samruddhi Keshav Zarapkar, Manager.

The contents of this project report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree.

Sign _____

ACKNOWLEDGEMENT

I, Divyanshu Rajpoot (2115000384), would like to express my sincere gratitude to my project manager, Mrs. Samruddhi Keshav Zarapkar, for his continuous guidance, valuable feedback, and constant encouragement throughout this project. His mentorship, insightful reviews, and problem-solving approach played a crucial role in overcoming challenges and successfully completing this work. The lessons I've learned under his supervision have greatly contributed to my personal and professional growth.

I am also thankful to Capgemini for providing me with the opportunity, resources, and supportive environment to work on this project. The access to modern tools, technologies, and a collaborative culture enabled me to enhance my technical skills and gain practical experience in a professional setting.

Finally, I appreciate the teamwork, knowledge-sharing, and encouragement from my colleagues at Capgemini. This experience has been immensely rewarding, and I look forward to applying these learnings in my future career.

Sign _____
Name of Candidate: Divyanshu Rajpoot
University Roll No: 2115000384

ABSTRACT

*The **CropDeal** project is a microservices-based web application developed using Spring Boot, aimed at streamlining the agricultural trading process between farmers, dealers, and administrators. The system is designed to facilitate secure and efficient interactions by providing tailored functionalities for each user role through independently deployable microservices.*

*The architecture comprises several microservices including **Farmer Service**, **Dealer Service**, **Admin Service**, **Eureka Discovery Server**, **API Gateway**, **Authentication Service**, **Notification Service** and **Payment Service**. Farmers can register, log in, and manage their crops by performing create, read, update, and delete (CRUD) operations. Dealers can browse available crops, add selected items to their cart, and proceed with payments through a secure transaction system. The **Payment Service** handles transactions and ensures the integrity and traceability of each order.*

*An **Admin Service** allows administrators to monitor platform activity, view a list of all registered users, and delete users when necessary to maintain platform integrity and security. All services are registered with **Eureka**, enabling service discovery and load balancing. The **API Gateway** routes client requests to the appropriate services, ensuring centralized access control and monitoring.*

*The outcome of the project demonstrates a robust, scalable solution for digital crop trading. By leveraging microservices architecture, **CropDeal** offers modularity, easier maintenance, and improved scalability, making it a practical foundation for real-world agricultural trading platforms.*

CONTENTS

CHAPTER 1 Introduction

Overview and Motivation:

Agriculture remains a cornerstone of many economies, yet farmers often face challenges in accessing fair markets, transparent pricing, and reliable buyers. At the same time, dealers struggle to find trustworthy sources for high-quality crops. Traditional systems are often inefficient, prone to middlemen exploitation, and lack digital traceability.

The CropDeal project aims to bridge this gap by providing a digital platform that connects farmers and dealers directly, enabling seamless crop trading through a user-friendly and secure application. This system leverages Spring Boot and a microservices architecture to ensure modularity, scalability, and maintainability.

Each stakeholder — farmer, dealer, and admin — has dedicated functionalities that streamline their respective interactions. Farmers can list, update, or remove their crops. Dealers can browse available crops, add them to a cart, and make secure payments. Admins monitor system activity and manage user accounts to ensure platform integrity.

Security is a core aspect of CropDeal. The application incorporates a role-based authentication and authorization system using JSON Web Tokens (JWT), ensuring that users can only access features appropriate to their role. This enhances data security and prevents unauthorized access across the system.

In addition, CropDeal integrates a notification system that sends automated email messages to users upon signup and login, helping to confirm activity and enhance account security. This feature also improves user engagement and trust.

The motivation behind building CropDeal stems from the need to empower farmers with digital tools, ensure fair market access, and enhance transparency in agricultural trade. By applying modern software engineering principles and a service-oriented architecture, CropDeal not only facilitates efficient crop trading but also lays the foundation for scalable enhancements such as analytics, pricing intelligence, and mobile access in future iterations.

This project showcases how technology can positively impact agriculture by fostering trust, reducing manual processes, and improving overall trade efficiency.

Objective:

The primary objective of the **CropDeal** project is to develop a scalable, secure, and efficient web-based platform that facilitates smooth interactions between farmers, dealers, and administrators for the purpose of crop trading. The system aims to digitalize and simplify the agricultural trade process by leveraging modern microservices architecture and Spring Boot framework.

The specific objectives include:

- To enable **farmers** to register, log in, and manage their crop listings (add, update, delete) through an intuitive interface.
- To allow **dealers** to browse available crops, add them to a cart, and proceed with secure payments.
- To provide **administrators** with the ability to view and manage all users, including deleting accounts when necessary.
- To design and implement a **microservices architecture** that ensures modular development, scalability, and easy maintenance.
- To integrate an **API Gateway** for centralized routing, monitoring, and access control.
- To implement **Eureka Discovery Service** for efficient service registration and communication between microservices.
- To ensure **secure and reliable payment handling** using a dedicated Payment Service.
- To demonstrate a practical application of cloud-ready architecture for real-world agricultural use cases.

By achieving these objectives, the CropDeal platform aims to create a fair, transparent, and efficient ecosystem for agricultural trade.

Summary of Similar Application:

Several platforms exist that offer agricultural marketplace services, connecting farmers, dealers, and other stakeholders in the supply chain. Analyzing these existing solutions helps identify best practices, feature gaps, and opportunities for innovation in the **CropDeal** project.

1. AgriBazaar

AgriBazaar is a digital agri-trading marketplace that connects farmers, buyers, and agri-businesses. It offers services like crop listings, price discovery, digital payments, and logistics support. However, it is a large-scale commercial platform, and its architecture is largely monolithic, which may limit flexibility in small or customized deployments.

2. eNAM (National Agriculture Market)

eNAM is a government-backed platform in India that integrates agricultural markets across the country. It allows farmers to sell produce online and provides real-time price discovery. While highly impactful, eNAM has limited user-level customization and requires integration with physical mandis, which may not suit small-scale or fully digital models like CropDeal.

3. DeHaat

DeHaat is a full-stack agri-tech platform that offers end-to-end services including crop advisory, input sales, and market linkage. It uses technology extensively but is primarily available as a mobile application. Its services are broader in scope compared to CropDeal, which focuses strictly on digital crop trading and user management.

Comparison with CropDeal

Unlike most of the above platforms, **CropDeal** is built using a **microservices architecture**, allowing for independent scaling and deployment of services like user management, crop handling, and payment processing. Its modular design is ideal for academic learning and practical deployment in localized agri markets. Moreover, CropDeal provides a clear separation of roles (farmer, dealer, admin) and offers an extensible foundation for future integrations like mobile apps, machine learning for price prediction, or blockchain for traceability.

Organization of the Project:

The **CropDeal** project is organized into multiple logical components, each addressing a specific functional and technical requirement. The system is built using a **microservices architecture**, which allows independent development, testing, deployment, and scaling of each service. This modular design enhances the system's maintainability and extensibility.

The overall organization of the project is as follows:

1. Farmer Service

- Handles farmer registration, login, and profile management.
- Allows farmers to perform CRUD operations on crop listings.
- Stores farmer-related data in a dedicated database.

2. Dealer Service

- Manages dealer registration, login, and profile functions.
- Enables dealers to browse crops, add selected crops to a cart, and initiate payments.
- Maintains a cart and order history for each dealer.

3. Admin Service

- Allows the admin to view all registered users (farmers and dealers).
- Provides functionality to delete users when necessary to maintain platform integrity.
- Ensures administrative control and system oversight.

4. Eureka Discovery Server

- Acts as the service registry.
- Allows dynamic discovery and communication between microservices.

5. API Gateway

- Serves as a single entry point for all client requests.
- Routes requests to appropriate services.
- Provides basic security, monitoring, and request logging.

6. Authentication Service

- Manages user authentication using **JWT (JSON Web Tokens)**.
- Provides role-based access control for farmers, dealers, and admins.
- Ensures secure login and token validation for protected resources.

7. Payment Service

- Sends automated **email notifications** on user **signup and login**.
- Enhances user engagement and provides an added layer of security.
- Can be extended to support alerts for order confirmations, status updates, etc.

6. Payment Service

- Simulates payment processing for dealer transactions.
- Handles transaction verification and status updates.
- Ensures secure and reliable flow of payment data.

7. Configuration and Support Files

- Each service has its own `application.properties` for environment-specific settings.

CHAPTER 2 Software Requirement Analysis

2.1 Introduction

This chapter outlines the functional and non-functional requirements essential for the development of the CropDeal platform. The goal of this requirement analysis is to identify the system's capabilities, performance expectations, and user needs. It serves as a foundation for the software design and implementation phases.

2.2 Functional Requirements

Functional requirements define the system's core behavior and interactions:

1. **User Registration and Login**
    - Farmers, dealers, and admins must be able to register and log in securely.
    - System must validate credentials and assign appropriate roles.
2. **Role-Based Access Control**
    - Access to services and features must be restricted based on user roles (Farmer, Dealer, Admin).
3. **Crop Management (Farmer)**
    - Farmers can add, update, view, and delete crop listings.
4. **Crop Browsing and Ordering (Dealer)**
    - Dealers can browse crop listings, add them to a cart, and place orders.
5. **Payment Processing**
    - Dealers can simulate payments for selected crops.
    - Payment status must be updated accordingly.
6. **User Management (Admin)**
    - Admin can view all registered users and delete accounts as needed.
7. **Authentication Service**
    - The system must issue JWT tokens on successful login for secure access.
8. **Email Notification**
    - The system should send automated email alerts on signup and login events.

2.3 Non-Functional Requirements

These requirements define the quality and constraints of the system:

1. **Scalability**
    - The system must support an increasing number of users and services.
2. **Security**
    - Authentication must use JWT.
    - Role-based access must be enforced for all services.
    - Sensitive data (passwords, payment info) must be securely stored.
3. **Performance**
    - System must respond to user actions within an acceptable time (e.g., <2 seconds).
4. **Availability**
    - The platform should aim for high availability and fault tolerance using service discovery.
5. **Maintainability**
    - Modular microservices should allow easy updates and debugging.
6. **Usability**
    - The UI should be clean, responsive, and easy to navigate across different devices.

2.4 Hardware Requirements

Component	Specification
Processor	Intel Core i5 or equivalent
RAM	8 GB minimum
Hard Disk	500 GB
Network	Stable internet connection

2.5 Software Requirements

Software Component	Version / Technology
Operating System	Windows / Linux / macOS
Backend Framework	Spring Boot
API Gateway	Spring Cloud Gateway
Database	MySQL
Frontend (optional)	ReactJS / Angular (if applicable)
Email Service	JavaMailSender
Authentication	JWT (JSON Web Tokens)
Service Discovery	Eureka Discovery Server
Build Tool	Maven
IDE	IntelliJ IDEA / Eclipse

2.6 Technical Feasibility:

The **technical feasibility** of the CropDeal project assesses whether the current technology stack, tools, and architecture can effectively support the development, deployment, and maintenance of the platform. After careful consideration, the project was deemed technically feasible based on the following factors:

1. Use of Microservices Architecture

- CropDeal uses a **Spring Boot-based microservices architecture**, which allows each module (Farmer, Dealer, Admin, Payment, etc.) to be developed, deployed, and scaled independently.
- This approach enhances flexibility, fault isolation, and modular development, making the system robust and maintainable.

2. Proven Technology Stack

- The backend is implemented using **Java with Spring Boot**, a mature and widely adopted framework for building enterprise-grade applications.
- **Spring Cloud** components like **Eureka** and **API Gateway** are leveraged for service discovery and routing, ensuring efficient communication between services.
- RESTful APIs ensure interoperability and scalability of the system.

3. Database and Storage

- Each microservice uses a dedicated **relational database (MySQL)**promoting data separation and service autonomy.
- ORM tools like **Spring Data JPA** simplify database interactions and reduce boilerplate code.

4. Security and Authentication

- The system supports **role-based access control**, enabling secure user authentication and authorization for Farmers, Dealers, and Admins.
- Spring Security or JWT tokens can be integrated to enhance session security and protect endpoints.

5. Deployment Readiness

- All services are designed to run independently and can be containerized using **Docker** for easier deployment.
- The architecture is suitable for cloud environments (e.g., AWS, Azure, GCP) or local server deployment.

6. Development Tools and IDE Support

- Tools like **IntelliJ IDEA**, **Postman**, and **Git** streamline the development and testing process.
- Logs, error tracking, and Spring Boot Actuator endpoints aid in monitoring and debugging.

CHAPTER 3 Software Design

Software Design

The **CropDeal** project is designed using a **microservices-based architecture** that ensures modularity, scalability, and maintainability. Each service is independently developed, deployed, and managed, enabling seamless integration and easier debugging. This section outlines the design principles, architecture, and key components of the system.

3.1 Design Principles

- **Separation of Concerns:** Each service is responsible for a specific domain (e.g., user management, crop management, payments).
- **Loose Coupling:** Services communicate over HTTP REST APIs, ensuring low dependency between modules.
- **High Cohesion:** Functionality related to a specific business logic is grouped together within a service.
- **Scalability:** Services can be independently scaled based on load.

3.2 Architecture Overview

The system comprises the following microservices:

1. **Farmer Service** – Manages farmer registration, login, and crop CRUD operations.
2. **Dealer Service** – Allows dealers to view crops, add to cart, and place orders.
3. **Admin Service** – Provides admin-level control for managing user accounts.
4. **Payment Service** – Simulates and processes dealer payments.
5. **Authentication Service** – Handles user authentication and authorization using JWT (JSON Web Tokens). It enforces role-based access control, ensuring users access only features permitted to their roles.
6. **Notification Service** – Sends automated email notifications to users on signup and login events, enhancing security and user engagement.
7. **API Gateway** – Central entry point for all external requests, routing them to respective services.
8. **Eureka Discovery Server** – Manages dynamic service registration and discovery.

3.3 Database Design

- Each service has its own **dedicated database** to ensure data encapsulation and service autonomy.
- ORM is implemented using **Spring Data JPA**, and entity classes represent tables such as:
    - Farmers
    - Dealers
    - Crops
    - Orders
    - Users
    - Payments

3.4 API Design

Each service exposes a RESTful API with endpoints like:

- POST /register, POST /login – for user authentication
- POST /crops, GET /crops, PUT /crops/{id}, DELETE /crops/{id} – for crop management
- POST /cart, GET /cart, POST /pay – for dealer shopping and payment
- GET /admin/users, DELETE /admin/user/{id} – for admin operations

3.5 Security Design

- Role-based access control ensures that each user type (Farmer, Dealer, Admin) can only access allowed endpoints.
- Basic authentication and token-based security (e.g., JWT) can be integrated for added security.

3.6 Error Handling and Logging

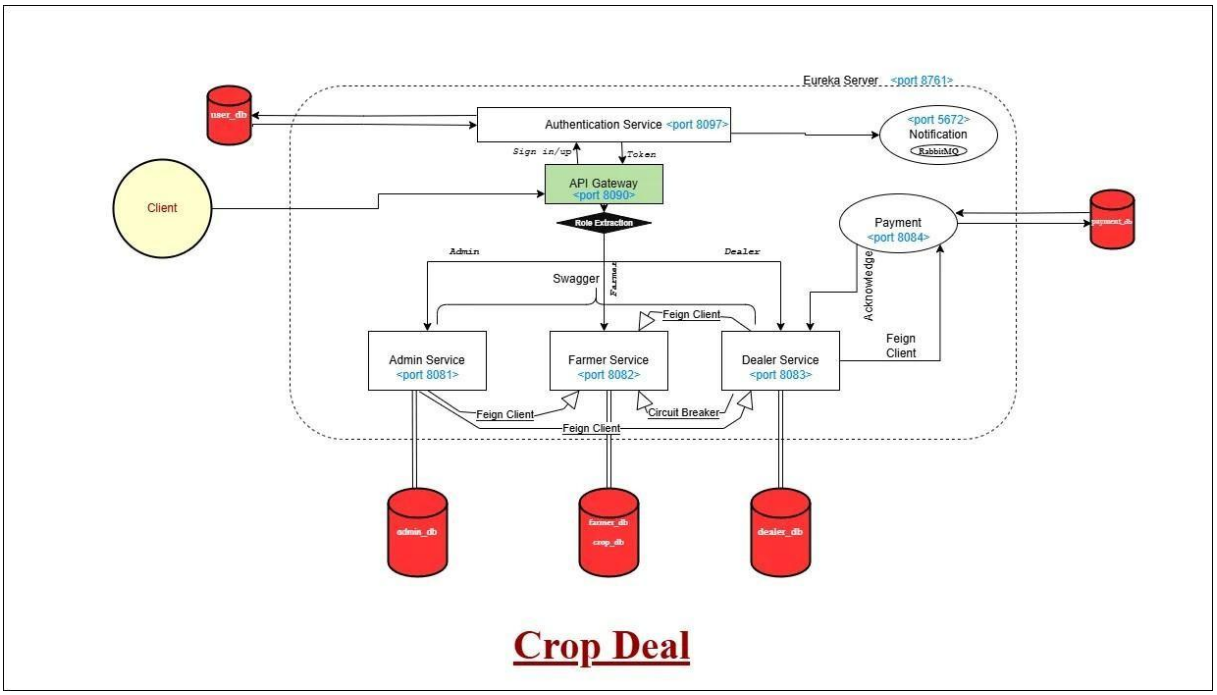
- Standardized response formats and status codes are used.
- Logs are maintained using Spring Boot's logging framework for debugging and auditing.

5.7 Architecture Diagram

The **Architecture Diagram** presents the high-level view of the system's components and how they interact. It outlines the client interaction through the **API Gateway**, which routes requests to individual microservices such as:

- **Farmer Service**
- **Dealer Service**
- **Admin Service**
- **Authentication Service**
- **Notification Service**
- **Payment Service**
- **Eureka Discovery Server**

Each service communicates via REST APIs and registers with the Eureka server for service discovery. This design ensures **loose coupling** and **independent deployment** of each module.

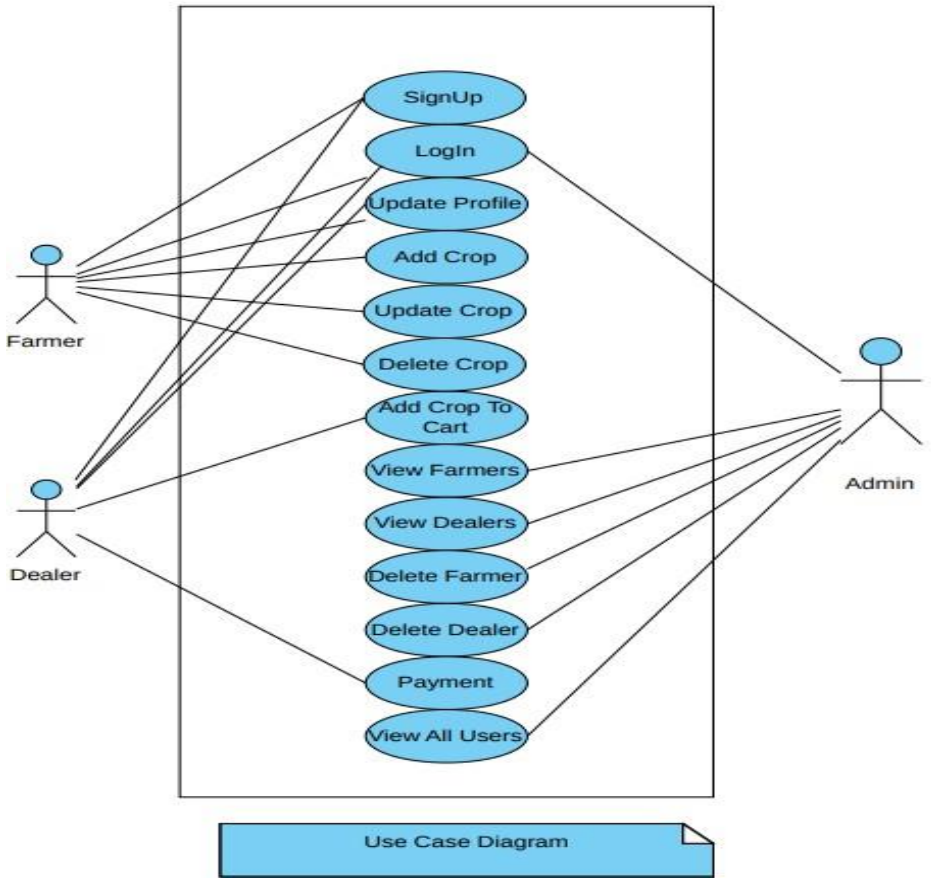


5.8 Use Case Diagram

The **Use Case Diagram** defines the functional requirements of the system from the perspective of each user type:

- **Farmer**: Can register, login, add/update/delete crops, and view listed crops.
- **Dealer**: Can register, login, browse crops, add to cart, and proceed to payment.
- **Admin**: Can view all users, monitor system activity, and delete users when necessary.

This diagram helps to clearly identify and define all user interactions and system responsibilities.

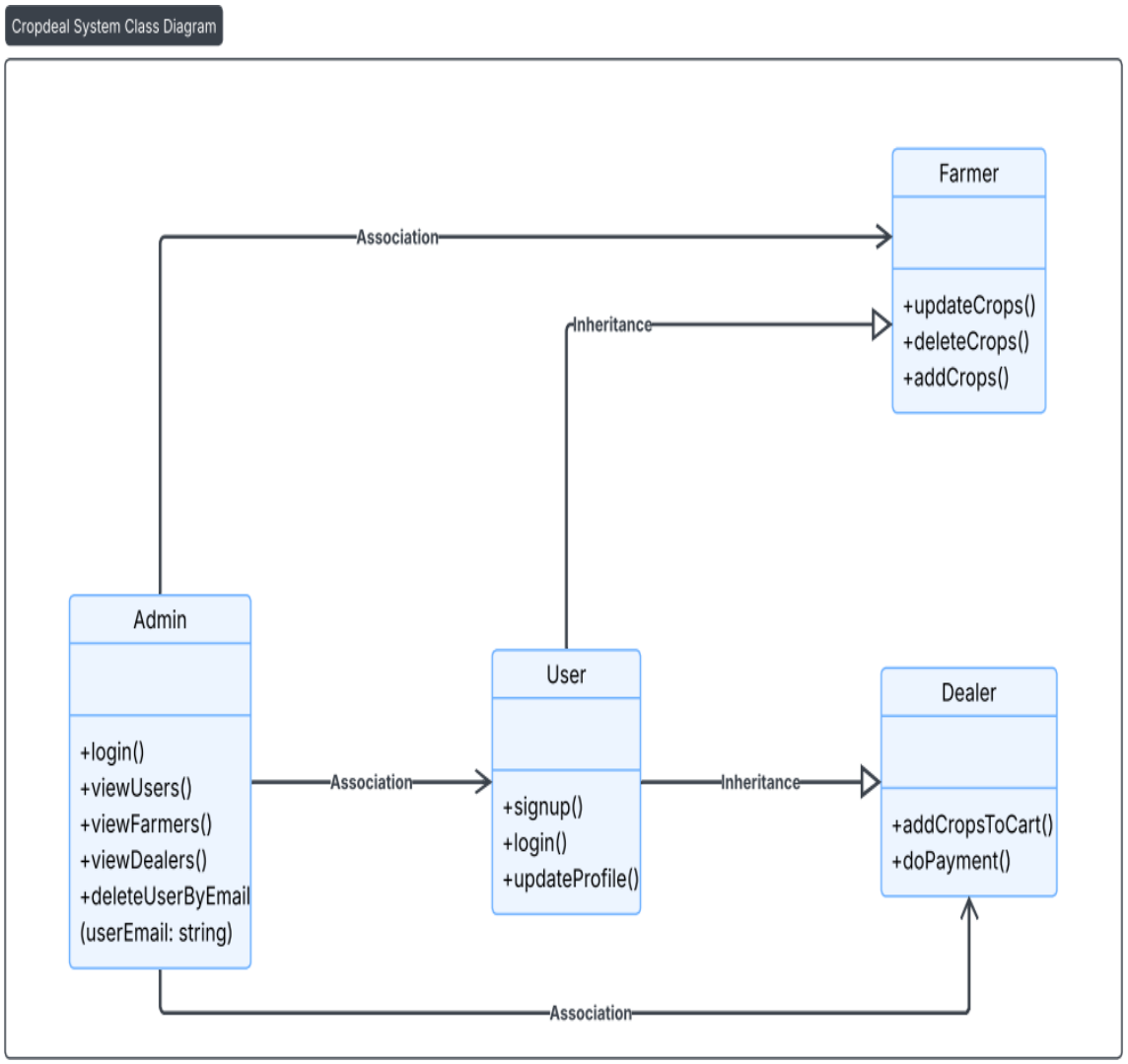


5.9 Class Diagram

The **Class Diagram** illustrates the object-oriented structure of the system. It defines the main classes, their attributes, and the relationships among them. Some key classes include:

- Farmer: Contains fields like id, name, email, password, and has a relationship with Crop.
- Crop: Includes properties such as cropId, cropName, price, quantity, and is associated with a Farmer.
- Dealer: Contains details like dealerId, name, email, and is linked to Cart and Order.
- Admin: Manages user data and system operations.
- Payment: Connected to Dealer and Order for processing transactions.

This design ensures encapsulation and supports code reusability across services.

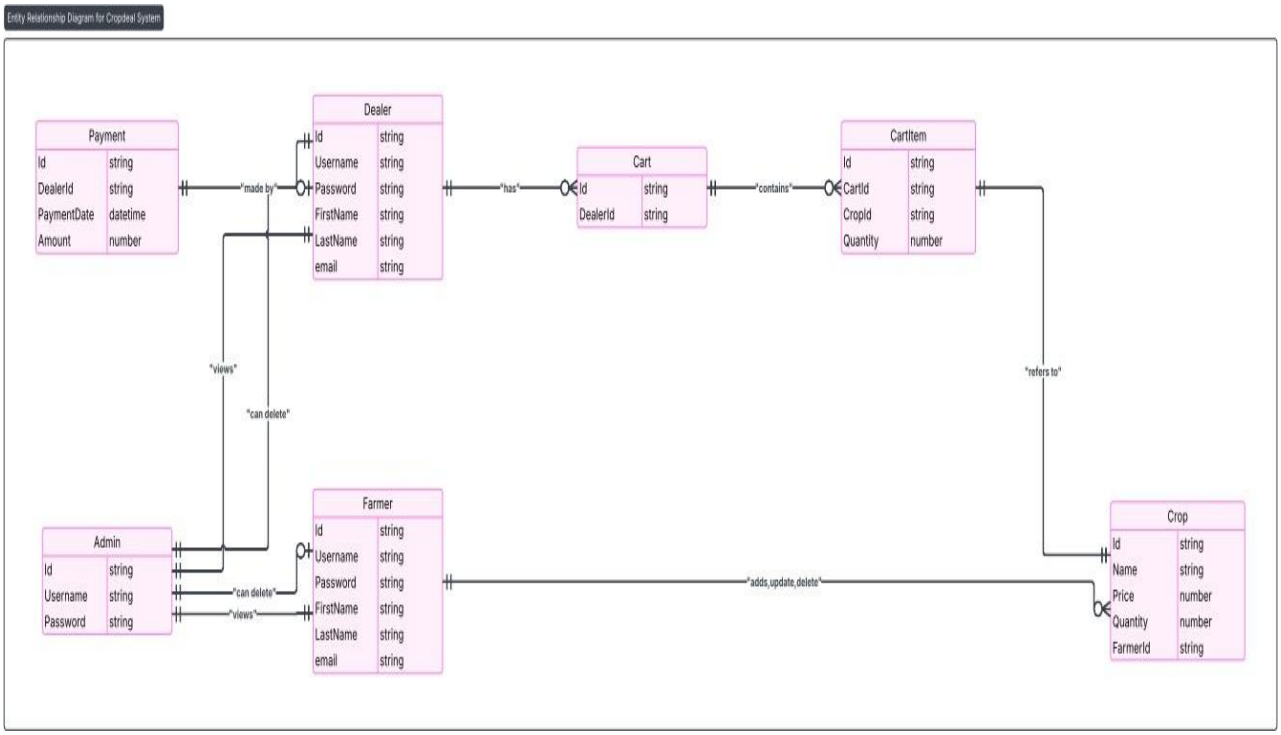


5.10 Entity-Relationship (ER) Diagram

The **ER Diagram** models the database schema and defines the relationships between various entities. Key entities include:

- **Farmer** and **Crop**: One-to-many relationship (a farmer can list multiple crops).
- **Dealer**, **Cart**, and **Order**: Dealers can maintain a cart and place orders, representing many-to-many or one-to-many relationships.
- **Payment**: Linked to orders and dealers, ensuring transactional data integrity.

Each microservice manages its own database, aligning with the **database-per-service** pattern, which improves scalability and independence.



5.11 Summary

This design approach, backed by clear and structured diagrams, ensures that each component is well-defined and contributes to a highly modular and scalable system. The use of microservices and layered design principles makes it easier to maintain and extend the application in future developments.

CHAPTER 4 Implementation and User Interface

4.1 Implementation Overview

The implementation of the CropDeal project follows a microservices-based architecture to ensure modularity, scalability, and maintainability. Each core functionality is encapsulated in its own independent service, developed using Spring Boot. The services communicate through RESTful APIs and are registered with Eureka Discovery Server for dynamic service discovery.

Security is implemented using JWT-based authentication. The Authentication Service issues secure tokens upon user login or signup, enabling role-based access control across the system. Only authorized users can access features based on their assigned roles (Farmer, Dealer, Admin).

An API Gateway acts as the single entry point for all incoming requests. It handles routing, basic security filtering, and logs incoming requests before forwarding them to the appropriate microservices.

A Notification Service is integrated to send email alerts to users upon successful signup and login, improving both security and user engagement. The Payment Service simulates transaction handling for crop purchases and confirms the transaction status to the Dealer Service.

Each microservice has been implemented with a focus on separation of concerns and high cohesion, ensuring ease of testing, debugging, and future extension.

4.2 Technology Stack

- Backend Framework: Spring Boot (Java)
- Architecture: Microservices
- Authentication: JWT (JSON Web Token)
- Service Discovery: Netflix Eureka
- API Gateway: Spring Cloud Gateway
- Database: MySQL
- Email Notifications: JavaMailSender
- Build Tool: Maven

- IDE: IntelliJ IDEA / Eclipse

4.3 User Interface

The front-end of the CropDeal application is designed with a focus on simplicity and usability. The UI components are organized role-wise:

- **Farmer Dashboard:**
  - Register/Login
  - Add/Update/Delete Crops
  - View Listed Crops
- **Dealer Dashboard:**
  - Register/Login
  - Browse Crop Listings
  - Add to Cart
  - Place Orders and View Order History
- **Admin Dashboard:**
  - Login
  - View All Users
  - Delete User Accounts

The user interface is responsive and provides immediate feedback on actions such as login, crop upload, and order placement. It communicates with backend services via REST APIs through the API Gateway.

CHAPTER 5 Software Testing

Software Testing

Testing is a critical phase in the development of the **CropDeal** project to ensure the system is reliable, secure, and performs as expected. Various testing strategies were employed to validate individual components and the integrated system.

5.1 Types of Testing Conducted

- **Unit Testing:**
  Each microservice's core functions were tested using **JUnit** and **Mockito** to verify the correctness of individual methods, such as CRUD operations in Farmer Service and payment processing in Payment Service.
- **Integration Testing:**
  Integration tests were conducted to verify communication between services, especially for workflows involving multiple microservices like user authentication, crop ordering, and payment confirmation. Spring Boot's testing support was used for these tests.
- **API Testing:**
  All RESTful APIs were tested using **Postman** to ensure endpoints respond correctly with valid inputs and handle invalid requests gracefully. Authentication tokens were included to test role-based access restrictions.
- **Functional Testing:**
  Verified that the application meets functional requirements, including user registration, login, crop management, cart functionality, payment processing, and admin controls.
- **Security Testing:**
  Focused on authentication, authorization, and data protection. Tests were performed to ensure JWT token validation, role-based access control, and prevention of unauthorized access.
- **Performance Testing (Basic):**
  Basic load testing was performed by simulating multiple user requests to check system responsiveness and scalability under moderate load.

5.2 Testing Tools

- **JUnit & Mockito:** For unit and integration testing of service logic.
- **Postman:** To perform manual API testing, automate test suites, and validate responses.
- **Spring Boot Test:** For context loading and integration tests within Spring environment.
- **Logs & Debugging:** Monitored application logs to trace errors and monitor system health during tests.

5.3 Test Coverage

- Most critical service functionalities, such as authentication, crop CRUD operations, cart management, and payment transactions, have corresponding unit and integration tests.
- Error handling and boundary cases, such as invalid inputs or unauthorized access attempts, were also tested.
- Future work includes increasing automated test coverage and adding end-to-end testing with tools like Selenium.
- Basic checks were conducted to validate user roles and protect endpoints from unauthorized access.

5.4 Test Results

Test Case	Result	Status
User Signup/Login	Valid tokens generated	Passed
Crop CRUD Operations	All operations functional	Passed
Dealer Cart and Order Flow	Working as expected	Passed
Admin Delete User	Restricted to admin only	Passed
JWT Role Enforcement	Access controlled	Passed
Email Notification on Signup	Email received	Passed

4.4 Challenges in Testing

- Coordinating tests across distributed microservices required proper setup of service dependencies and mocks.
- Testing inter-service communication and handling transient failures was complex.
- Ensuring security mechanisms (JWT tokens, role-based access) were properly enforced during API tests.
- The **CropDeal** project successfully demonstrates a scalable and modular microservices-based platform for connecting farmers and dealers in the agricultural marketplace. By leveraging Spring Boot and Spring Cloud technologies, the system efficiently manages user authentication, crop management, dealer transactions, and administrative controls through independently deployable services.
- The implementation of dedicated microservices such as Authentication, Farmer, Dealer, Admin, Payment, Eureka Discovery Server, and API Gateway ensures high

cohesion and loose coupling, facilitating easier maintenance and future enhancements. The use of RESTful APIs and service discovery enables seamless communication between components, while role-based access control secures the platform for different user types.

- Testing across unit, integration, functional, and security dimensions validated the reliability and robustness of the system. Although challenges such as inter-service communication and security enforcement were encountered, they were effectively addressed during development.
- In conclusion, CropDeal offers a practical and extensible solution to modernize agricultural trade by providing farmers with a digital platform to market their crops and dealers with a streamlined purchasing experience. Future work can focus on adding advanced features such as real-time notifications, analytics, and mobile app support to further enhance user engagement and system capabilities.

CHAPTER 6 Conclusion

The CropDeal project was developed with the objective of bridging the gap between farmers and dealers through a secure, scalable, and efficient digital platform. By leveraging modern software engineering practices such as microservices architecture, JWT-based authentication, and role-based access control, the system successfully addresses many challenges faced in traditional agricultural trade, including market inefficiencies, lack of transparency, and the dominance of intermediaries.

Through dedicated services for each core functionality—farmer crop management, dealer order processing, admin oversight, secure payments, and email notifications—the platform ensures modularity, maintainability, and ease of scaling. The Authentication Service secures the platform by verifying users and assigning role-specific access, while the Notification Service improves engagement and security by keeping users informed of account activities.

The implementation of an API Gateway and Eureka Discovery Server supports dynamic service registration, fault tolerance, and simplified communication between microservices, resulting in a robust and resilient architecture.

The user interface was designed to be intuitive and role-specific, ensuring a smooth and productive experience for all stakeholders—farmers, dealers, and admins. The system's design and implementation lay a strong foundation for future enhancements such as analytics dashboards, real-time pricing intelligence, mobile app integration, and integration with real payment gateways.

In conclusion, CropDeal demonstrates how digital technology can be effectively utilized to transform the agricultural sector by promoting fair trade, transparency, and efficiency. It not only empowers farmers but also creates a trustworthy marketplace for dealers. This project has the potential to evolve into a full-scale commercial application that contributes positively to the agricultural ecosystem.

CHAPTER 7 Summary

The **CropDeal** project is a microservices-based application designed to facilitate seamless interaction between farmers and dealers in the agricultural marketplace. Developed using Spring Boot and Spring Cloud technologies, the platform consists of multiple independent services including Farmer, Dealer, Admin, Authentication, Payment, Eureka Discovery Server, and API Gateway.

Farmers can register, manage their crop listings, and update inventory, while dealers can browse available crops, add items to their cart, and complete payments securely. The admin service oversees user management and platform integrity by monitoring and controlling user access.

The implementation leverages RESTful APIs, service discovery, and role-based access control to create a scalable and secure environment. Extensive testing ensured the application's functionality, reliability, and security across various use cases.

Overall, CropDeal addresses the need for digitizing agricultural trade, providing an efficient and user-friendly solution that benefits both farmers and dealers. The modular design facilitates future enhancements and scalability to meet evolving market demands.

APPENDICES

Appendix 1: Example of Description Page

CropDeal - Crop Management Description

Module: Farmer Service

Description:
This module allows farmers to register, log in, and manage their crop listings on the CropDeal platform. Farmers can add new crops, update details of existing crops, or delete crops that are no longer available for sale.

Features:

- **Registration and Login:**
  Farmers can create accounts and securely log in to the system via the Authentication Service.
- **Add Crop:**
  Farmers can input crop details such as crop name, quantity, price, and expected harvest date.
- **Update Crop:**
  Modify the details of existing crops to reflect changes in quantity, price, or availability.
- **Delete Crop:**
  Remove crops from the listing once they are sold out or no longer available.
- **View Crops:**
  Farmers can view all their listed crops with complete details.

User Interface:

- Simple forms for crop details entry.
- Validation checks for required fields (e.g., price must be a positive number).
- Confirmation prompts before deletion.

API Endpoints:

HTTP Method	Endpoint	Description
POST	/farmer/register	Register a new farmer
POST	/farmer/login	Authenticate farmer
POST	/crops	Add a new crop
GET	/crops	Retrieve farmer’s crops
PUT	/crops/{id}	Update crop details
DELETE	/crops/{id}	Delete a crop

Sample JSON Request for Adding a Crop:

json

CopyEdit

```
{
  "cropName": "Wheat",
  "quantity": 100,
  "price": 1500,
  "harvestDate": "2025-10-01"
}
```

Business Logic:

- Crops must have a positive quantity and price.
- Only authenticated farmers can perform crop management actions.
- Crop updates trigger notifications to interested dealers (future feature).

Dependencies:

- Authentication Service for user validation.
- Database service for storing crop information.

Known Issues / Limitations:

- Crop images upload not implemented.

- No support for bulk crop uploads (planned for future releases).

This description page serves as a detailed overview for developers and stakeholders to understand the functionality and technical details of the Farmer Service module.

 Pivotal Software, Inc. *Spring Boot Documentation*. https://spring.io/projects/spring-boot (Accessed: 2025)

 Netflix OSS. *Eureka: Service Discovery*. https://netflix.github.io/eureka/ (Accessed: 2025)

 Richardson, C. *Microservices Patterns: With examples in Java*. Manning Publications, 2018.

 Fowler, M. *Microservices: a definition of this new architectural term*. https://martinfowler.com/articles/microservices.html (Accessed: 2025)

 Fielding, R. T. *Architectural Styles and the Design of Network-based Software Architectures* (PhD Thesis), University of California, Irvine, 2000.

 Oracle Corporation. *Java SE Documentation*. https://docs.oracle.com/javase/ (Accessed: 2025)

 Baeldung. *A Guide to Spring Cloud Gateway*. https://www.baeldung.com/spring-cloud-gateway (Accessed: 2025)

 Postman, Inc. *Postman Documentation*. https://learning.postman.com/docs/getting-started/introduction/ (Accessed: 2025)

 Venkatesh, K., & Sharma, S. *Design and Implementation of Secure Microservices Architecture*, International Journal of Computer Applications, 2024.

 MySQL Documentation. *MySQL 8.0 Reference Manual*. https://dev.mysql.com/doc/ (Accessed: 2025)