Farmer Argo-Based Management System Course: UE22CS351A
: Database Management System
FINAL REPORT
Semester: 5
Section: L

Team Members:

1. Suhas PH (PES2UG22CS588) 2. Chandan N(PES2UG22CS140)

1. Introduction

1.1 Purpose

This Software Requirements Specification (SRS) document outlines the essential features, functionality, and performance requirements for the Farmer Agro-Based Management System. This system is designed to connect farmers and buyers through a web-based platform where farmers can manage their product listings and buyers can place orders for agricultural goods. The primary goal is to streamline the agricultural sales process, providing an efficient, secure, and user-friendly experience for both parties.

1.2 Scope

The Farmer Agro-Based Management System is intended for use by farmers who wish to sell their agricultural products and buyers looking to purchase these products. The system allows farmers to manage their product inventory, while buyers can browse, order, and track their purchases. The system is built to ensure secure transactions, reliable performance, and ease of use across various devices.

1.3 Definitions, Acronyms, and Abbreviations

- **Farmer**: A registered user who lists and manages agricultural products for sale.
- **Buyer**: A registered user who purchases products listed by farmers.
- **UI**: User Interface.
- **DB**: Database.

1.4 References

- IEEE Standard for Software Requirements Specifications (IEEE Std 830-1998).

1.5 Overview

This document is structured to provide a clear understanding of the system's requirements, including functional and non-functional aspects, user interfaces, and external interfaces. The intention is to serve as a comprehensive guide for developers, testers, and stakeholders involved in the project.

---

2. Overall Description

2.1 Product Perspective

The Farmer Agro-Based Management System is a web-based application that interfaces with a centralized database. It serves as an independent platform allowing farmers to manage their product offerings and buyers to make purchases directly through the system. The application is designed to ensure secure data handling, intuitive navigation, and compatibility with various devices.

2.2 Product Functions

The system is built around key functionalities that include:

- **User Registration and Authentication**: Users can securely register and log in to the system.
- **Product Management**: Farmers can add, update, and remove product listings.
- **Order Placement**: Buyers can browse products, place orders, and track their purchases.
- **Order Management**: Farmers can view and manage orders placed by buyers.
- **Sales Tracking**: Farmers can monitor their sales and view detailed reports.
- **Account Management**: Users can manage their personal information and view order histories.

2.3 User Classes and Characteristics

The primary users of this system are:

- **Farmers**: Individuals who manage product listings and fulfill orders.
- **Buyers**: Individuals who browse products and place orders.
- **Administrators**: Users who maintain the database and oversee system operations.

2.4 Operating Environment

- **Software**: The system is a web-based application, accessible through standard web browsers (e.g., Chrome, Firefox).
- **Hardware**: Users access the system via desktops, laptops, tablets, or smartphones with an internet connection.

2.5 Design and Implementation Constraints

- The system must provide secure data transmission to protect user information and transaction details.

-

- Compliance with relevant agricultural regulations is required for product listings.

  The user interface must be responsive and accessible across various devices to ensure a consistent user experience.

2.6 Assumptions and Dependencies

- The system assumes that users have access to devices with internet connectivity and a

modern web browser.

- A stable network connection is necessary for accessing the database and conducting transactions.

---

3. External Interface Requirements

3.1 User Interfaces

The system includes the following interfaces:

- **Web Interface**: The primary interface through which users interact with the system. It includes functionalities for login, registration, product management, order placement, and sales tracking.
- **Admin Interface**: Used by administrators to manage system operations, including database maintenance and user management.

3.2 Hardware Interfaces

- **Client Devices**: The system is accessible on any device with internet access, such as desktops, laptops, tablets, or smartphones.
- **Server**: The database and web application are hosted on a server that ensures data security and reliability.

3.3 Software Interfaces

- **Database Interface**: A MySQL database is used to store user data, product information, and transaction records.
- **API**: Potential integration with external services (e.g., payment gateways) for enhanced functionality.

3.4 Communication Interfaces

- **HTTPS**: Secure communication between the client and server is ensured through HTTPS, protecting data during transmission.

---

4. System Features

4.1 User Registration and Authentication

Description: The system provides a secure process for users to register and authenticate.
Functional Requirements:

- Users must be able to register by providing an email, phone number, and password.

- The system must authenticate users upon login, ensuring the correct credentials are provided.

4.2 Product Management

Description: Farmers can manage their product listings directly through the system. **Functional Requirements**:

- The system must allow farmers to add new products, including details such as product name, description, and price.
- Farmers must be able to edit or remove their product listings as needed.

4.3 Order Placement and Management

Description: Buyers can place orders for products listed by farmers, and farmers can manage these orders.
Functional Requirements:

- Buyers must be able to select products, specify quantities, and place orders.
- The system must track order status and allow farmers to confirm or update orders.

4.4 Sales Tracking

Description: Farmers can track their sales within the system, providing insights into their business performance.
Functional Requirements:

- The system must calculate and display total sales for each farmer.
- A summary of sales data should be available for farmers to review.

4.5 Account Management

Description: Users can manage their account details and view their transaction history.
Functional Requirements:

- The system must allow users to update their profile information, including name and contact details.
- Users must be able to view their past orders and, for farmers, their sales records.

---

5. Non-Functional Requirements

5.1 Performance Requirements

- The system should respond to user inputs within a maximum of 3 seconds under normal conditions.
- Order processing and confirmation should be completed within 5 seconds.

•

5.2 Security Requirements

- All sensitive data, such as user passwords, must be encrypted both in storage and during transmission.

  The system must implement secure login procedures to prevent unauthorized access.

5.3 Usability Requirements

- The user interface should be intuitive and straightforward, allowing users to navigate the system with ease.
- The system should be compatible with a wide range of devices, ensuring accessibility for all users.

5.4 Reliability Requirements

- The system must maintain an uptime of at least 99.9% to ensure availability for users.
- Data integrity must be maintained during all transactions and database updates.

---

6. Other Requirements

6.1 Regulatory Requirements

- The system must adhere to all applicable agricultural and trade regulations, particularly those related to product listings and sales.

6.2 Environmental Requirements

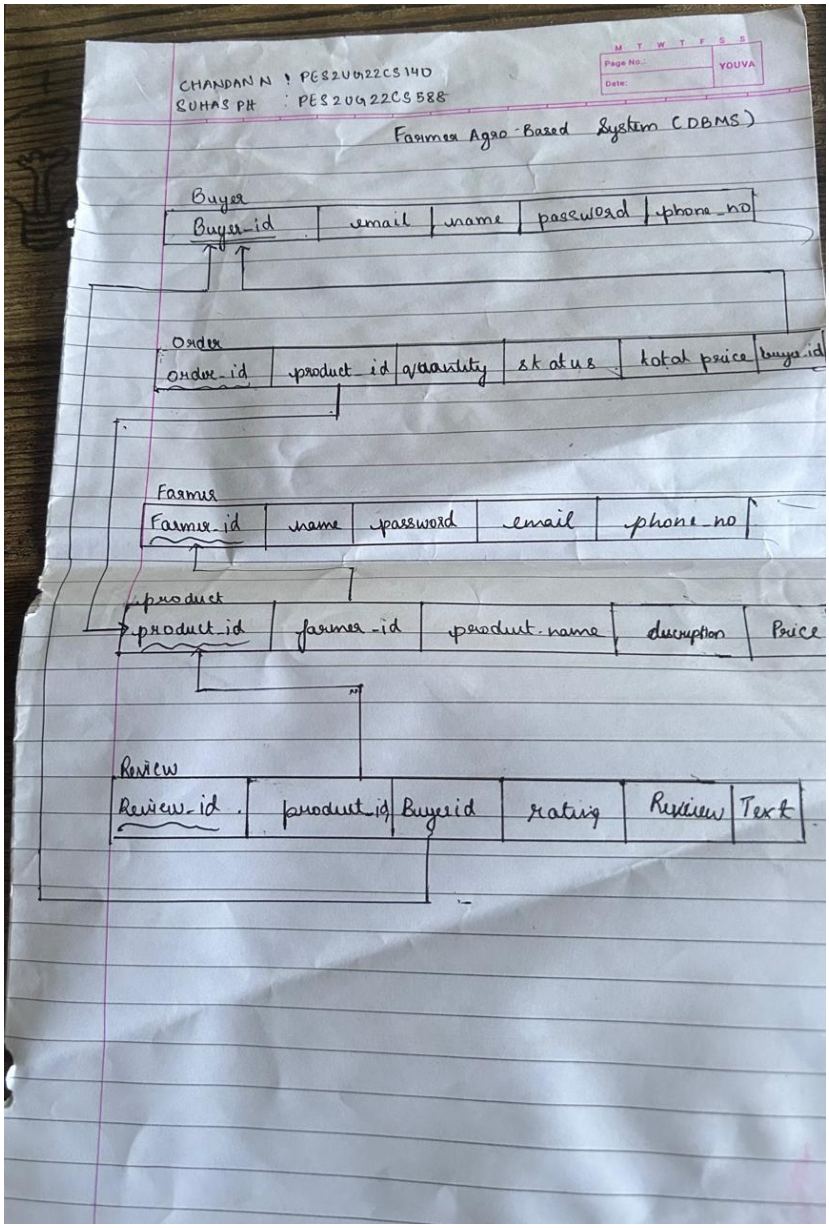
- The system should operate effectively within typical indoor environmental conditions, with no special hardware requirements.

ER DIAGRAM:

Its uploaded as separate png file

•

Relation Schema:



-

DDL COMMANDS

1) BUYERS TABLE

```
mysql> SHOW CREATE TABLE buyers;
+--------+------------------------------------------------------------------
| Table  | Create Table

+--------+------------------------------------------------------------------
| buyers | CREATE TABLE `buyers` (
  `buyer_id` int NOT NULL AUTO_INCREMENT,
  `name` varchar(255) NOT NULL,
  `email` varchar(255) NOT NULL,
  `phone_no` varchar(15) NOT NULL,
  `password` varchar(255) NOT NULL,
  `created_at` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`buyer_id`),
  UNIQUE KEY `email` (`email`)
) ENGINE=InnoDB AUTO_INCREMENT=13 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci |
+--------+------------------------------------------------------------------
1 row in set (0.01 sec)

mysql> |
```

2)FARMERS TABLE

```
mysql> SHOW CREATE TABLE FARMERS;
+---------+-----------------------------------------------------------------
--+
| Table   | Create Table

  |
+---------+-----------------------------------------------------------------
--+
| FARMERS | CREATE TABLE `farmers` (
  `farmer_id` int NOT NULL AUTO_INCREMENT,
  `name` varchar(255) NOT NULL,
  `email` varchar(255) NOT NULL,
  `phone_no` varchar(15) NOT NULL,
  `password` varchar(255) NOT NULL,
  `created_at` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`farmer_id`),
  UNIQUE KEY `email` (`email`)
) ENGINE=InnoDB AUTO_INCREMENT=10 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci |
+---------+-----------------------------------------------------------------
--+
1 row in set (0.00 sec)
```

3)Products Table

```
+----------+----------------------------------------------------------------
| products | CREATE TABLE `products` (
  `product_id` int NOT NULL AUTO_INCREMENT,
  `farmer_id` int DEFAULT NULL,
  `product_name` varchar(255) NOT NULL,
  `product_description` text,
  `unit_price` decimal(10,2) NOT NULL,
  `quantity` int NOT NULL,
  `status` enum('In Stock','Out of Stock') DEFAULT 'In Stock',
  PRIMARY KEY (`product_id`),
  KEY `farmer_id` (`farmer_id`),
  KEY `idx_products_name` (`product_name`),
  CONSTRAINT `products_ibfk_1` FOREIGN KEY (`farmer_id`) REFERENCES `farmers` (`farmer_id`) ON DELETE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=10 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci |
+----------+----------------------------------------------------------------
1 row in set (0.00 sec)

mysql> |
```

•

4) ORDERS TABLE

```
---------------------------------------------------------+
| Table  | Create Table
                                                          |
+--------+-------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------+
| orders | CREATE TABLE `orders` (
  `order_id` int NOT NULL AUTO_INCREMENT,
  `buyer_id` int DEFAULT NULL,
  `order_date` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  `total_amount` decimal(10,2) DEFAULT '0.00',
  PRIMARY KEY (`order_id`),
  KEY `idx_orders_buyer_id` (`buyer_id`),
  CONSTRAINT `orders_ibfk_1` FOREIGN KEY (`buyer_id`) REFERENCES `buyers` (`buyer_id`) ON DELETE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=14 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci |
+--------+-------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------+
1 row in set (0.00 sec)
```

5) ORDER_ITEMS TABLE

```
-------------------------------------------------------------+
| order_items | CREATE TABLE `order_items` (
  `order_item_id` int NOT NULL AUTO_INCREMENT,
  `order_id` int DEFAULT NULL,
  `product_id` int DEFAULT NULL,
  `quantity` int DEFAULT NULL,
  `price_at_time` decimal(10,2) DEFAULT NULL,
  `item_total` decimal(10,2) DEFAULT NULL,
  PRIMARY KEY (`order_item_id`),
  KEY `order_id` (`order_id`),
  KEY `product_id` (`product_id`),
  CONSTRAINT `order_items_ibfk_1` FOREIGN KEY (`order_id`) REFERENCES `orders` (`order_id`) ON DELETE CASCADE,
  CONSTRAINT `order_items_ibfk_2` FOREIGN KEY (`product_id`) REFERENCES `products` (`product_id`) ON DELETE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=19 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci |
+-------------+--------------------------------------------------------------------------------------------------------------
```

•

6) REVIEWS TABLE

```
| REVIEWS | CREATE TABLE `reviews` (
  `review_id` int NOT NULL AUTO_INCREMENT,
  `buyer_id` int DEFAULT NULL,
  `product_id` int DEFAULT NULL,
  `rating` int DEFAULT NULL,
  `review_text` text,
  `review_date` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`review_id`),
  KEY `buyer_id` (`buyer_id`),
  KEY `product_id` (`product_id`),
  CONSTRAINT `reviews_ibfk_1` FOREIGN KEY (`buyer_id`) REFERENCES `buyers` (`buyer_id`) ON DELETE CASCADE,
  CONSTRAINT `reviews_ibfk_2` FOREIGN KEY (`product_id`) REFERENCES `products` (`product_id`) ON DELETE CASCADE,
  CONSTRAINT `reviews_chk_1` CHECK (((`rating` >= 1) and (`rating` <= 5)))
) ENGINE=InnoDB AUTO_INCREMENT=16 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci |
```

QUERIES ACORDING TO RUBRICS(Nested, Aggregated , Join) ;

1)

```python
def get_products():
    conn = get_database_connection()
    cursor = conn.cursor(dictionary=True)

    query = """
    SELECT p.*,
           f.name AS farmer_name,
           CASE
               WHEN p.quantity > 50 THEN 'High Stock'
               WHEN p.quantity BETWEEN 10 AND 50 THEN 'Medium Stock'
               ELSE 'Low Stock'
           END AS stock_status,
           (SELECT AVG(rating) FROM reviews WHERE reviews.product_id = p.product_id) AS avg_rating
    FROM products p
    JOIN farmers f ON p.farmer_id = f.farmer_id
    WHERE p.quantity > 0
    ORDER BY avg_rating DESC, p.unit_price ASC;
    """

    cursor.execute(query)
    products = cursor.fetchall()

    cursor.close()
    conn.close()
    return products
```

2)

```
# Fetching orders with aggregated totals and product details
def get_orders(buyer_id):
    conn = get_database_connection()
    cursor = conn.cursor(dictionary=True)

    query = """
   SELECT o.order_id,
       o.order_date,
       oi.quantity,
       SUM(oi.item_total) AS total_order_amount,
       COUNT(DISTINCT oi.product_id) AS total_products,
       GROUP_CONCAT(p.product_name SEPARATOR ', ') AS product_name
FROM orders o
JOIN order_items oi ON o.order_id = oi.order_id
JOIN products p ON oi.product_id = p.product_id
WHERE o.buyer_id = %s
GROUP BY o.order_id, o.order_date, oi.quantity
ORDER BY o.order_date DESC;

    """

    cursor.execute(query, (buyer_id,))
    orders = cursor.fetchall()

    cursor.close()
    conn.close()
    return orders
```

3)

```
# Fetching sales data for a farmer with aggregate totals and date range filtering
def get_sales_data(farmer_id, start_date=None, end_date=None):
    conn = get_database_connection()
    cursor = conn.cursor(dictionary=True)

    query = """
    SELECT o.order_id,
           p.product_name,
           SUM(oi.quantity) AS total_quantity_sold,
           SUM(oi.item_total) AS total_sales,
           o.order_date
    FROM orders o
    JOIN order_items oi ON o.order_id = oi.order_id
    JOIN products p ON oi.product_id = p.product_id
    WHERE p.farmer_id = %s
        AND o.order_date BETWEEN %s AND %s
    GROUP BY o.order_id, p.product_id
    ORDER BY total_sales DESC, total_quantity_sold DESC;
    """

    # Defaulting to entire history if dates are not provided
    if not start_date:
        start_date = "1900-01-01"
    if not end_date:
        end_date = "2100-12-31"

    cursor.execute(query, (farmer_id, start_date, end_date))
    sales_data = cursor.fetchall()

    cursor.close()
    conn.close()
    return sales_data
```

•

4)

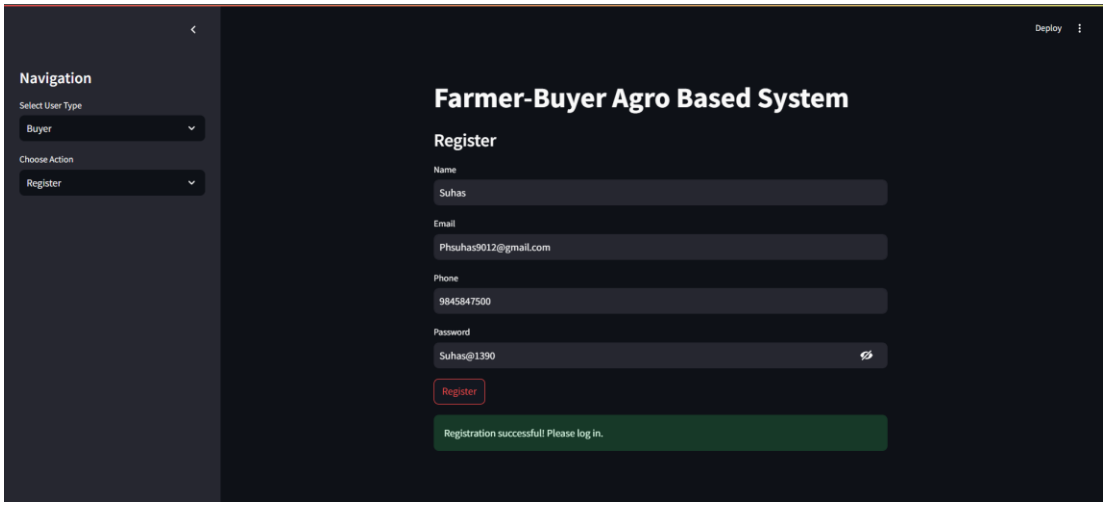
```
def get_top_products():
    conn = get_database_connection()
    cursor = conn.cursor(dictionary=True)

    query = """
    SELECT p.product_name,
           p.unit_price,
           AVG(r.rating) AS avg_rating,
           f.name AS farmer_name
    FROM products p
    LEFT JOIN reviews r ON p.product_id = r.product_id
    LEFT JOIN farmers f ON p.farmer_id = f.farmer_id
    WHERE p.quantity > 0
    GROUP BY p.product_id
    HAVING avg_rating IS NOT NULL
    ORDER BY avg_rating DESC
    LIMIT 10;
    """

    cursor.execute(query)
    top_products = cursor.fetchall()

    cursor.close()
    conn.close()
    return top_products
```

5)

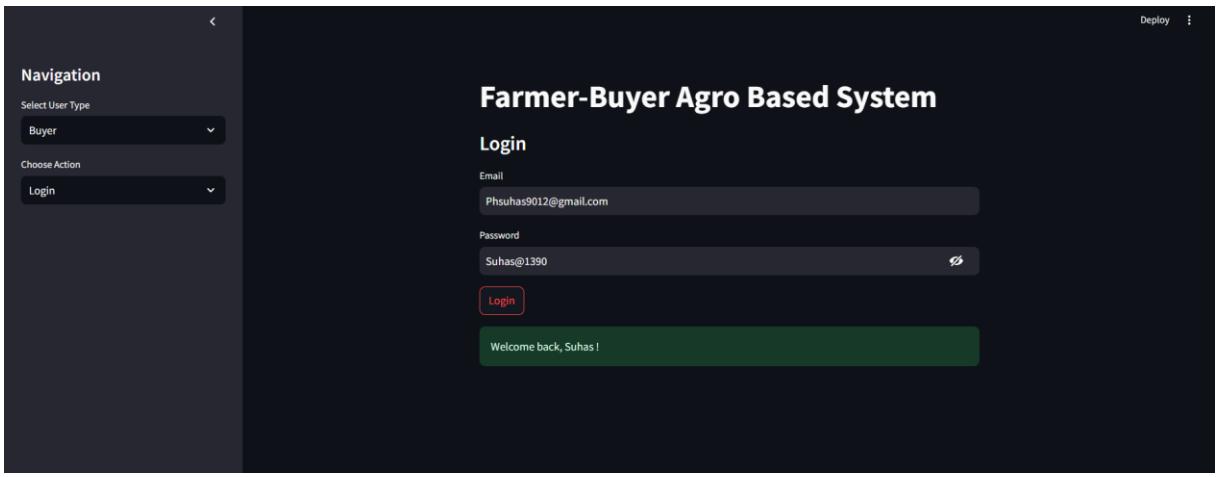
```python
def get_farmer_performance(farmer_id):
    conn = get_database_connection()
    cursor = conn.cursor(dictionary=True)

    query = """
    SELECT f.name AS farmer_name,
           COUNT(DISTINCT p.product_id) AS total_products,
           (SELECT COUNT(*)
            FROM orders o
            JOIN order_items oi ON o.order_id = oi.order_id
            WHERE oi.product_id IN (
                SELECT product_id
                FROM products
                WHERE farmer_id = f.farmer_id
            )) AS total_orders,
           COALESCE(SUM(oi.item_total), 0) AS total_earnings,
           (SELECT AVG(r.rating)
            FROM reviews r
            WHERE r.product_id IN (
                SELECT product_id
                FROM products
                WHERE farmer_id = f.farmer_id
            )) AS avg_rating
    FROM farmers f
    LEFT JOIN products p ON f.farmer_id = p.farmer_id
    LEFT JOIN order_items oi ON oi.product_id = p.product_id
    WHERE f.farmer_id = %s
    GROUP BY f.farmer_id;
    """

    cursor.execute(query, (farmer_id,))
    performance = cursor.fetchone()

    cursor.close()
    conn.close()
```

UI

1)Register((Similar for Farmer)

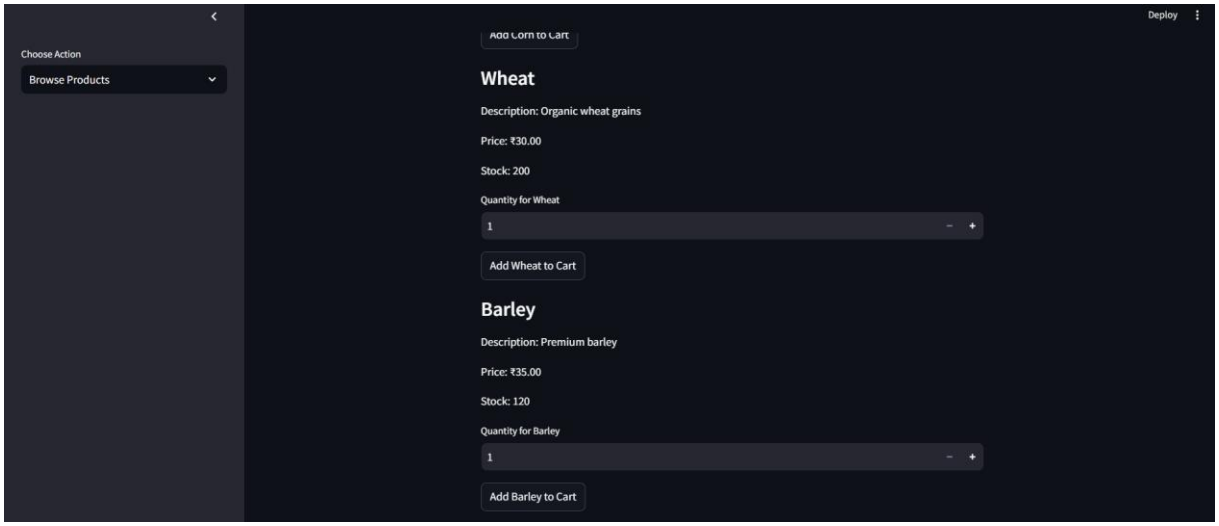


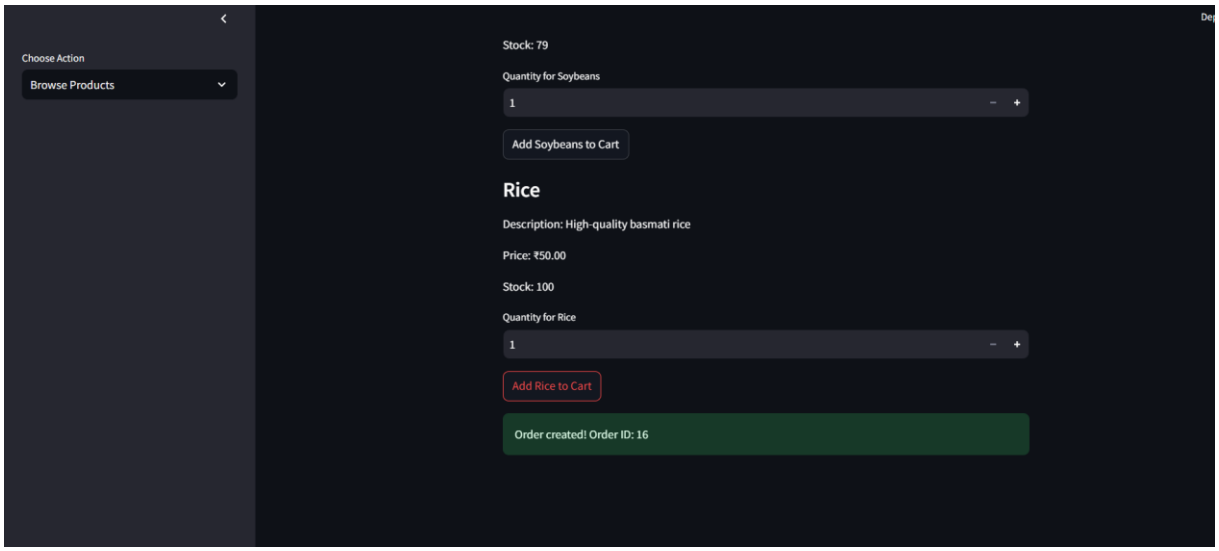
2)Login(Similar for Farmer)



•

3)Browse Products(for Buyers)

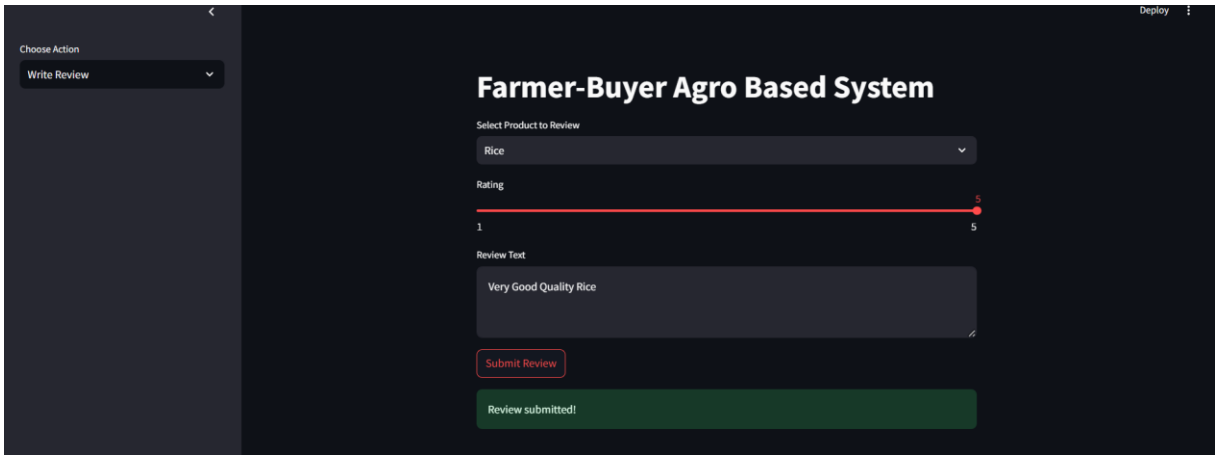




•

4)My Order(for Buyers)

5)Review(for Buyer)



•

6) View sales

7)Farmer Performance

Proecdure and Trigger

1)

•

2)

```
| Trigger               | Event  | Table        | Statement
                               | Timing | Created                | sql_mode                                                                                                          | Definer        | chara
cter_set_client | collation_connection | Database Collation |
| calculate_order_total | INSERT | order_items | BEGIN
    UPDATE orders
    SET total_amount = (
        SELECT SUM(item_total)
        FROM order_items
        WHERE order_id = NEW.order_id
    )
    WHERE order_id = NEW.order_id;
END | AFTER  | 2024-11-14 23:08:59.98 | ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION | root@localhost | cp850                | cp850_ge
neral_ci     | utf8mb4_0900_ai_ci |
| before_update_status  | UPDATE | products    | BEGIN
    IF NEW.quantity = 0 THEN
        SET NEW.status = 'Out of Stock';
    ELSE
        SET NEW.status = 'In Stock';
    END IF;
END                                                         | BEFORE | 2024-11-19 14:58:01.60 | ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION
 | root@localhost | cp850                | cp850_general_ci     | utf8mb4_0900_ai_ci |
2 rows in set (0.01 sec)
```

8)Top Products (both For Farmers and Buyers)

•

7. Conclusion

The **Farmer Agro-Based Management System** is designed to streamline the agricultural sales process by connecting farmers and buyers through an efficient and secure web-based platform. It offers essential features like user registration, product management, order placement, and sales tracking, ensuring ease of use and a seamless experience for both parties. The system's architecture is built on a MySQL database, with strong performance, security, and usability requirements. It is compliant with relevant agricultural regulations and offers a responsive interface across various devices.

The report provides comprehensive documentation of the system's functional and non-functional requirements, ensuring that developers, testers, and stakeholders are aligned in understanding and executing the project. With a focus on reliability, security, and performance, the system aims to enhance the agricultural market by providing an efficient solution for both farmers and buyers.