

FARM MANAGEMENT SYSTEM USING MYSQL & PYTHON FLASK



A DESIGN PROJECT REPORT

submitted by

SANJAY A (811722104128)

VISHNU KARTHIC R (811722104186)

KARTHIKEYAN S (811722104303)

in partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

K RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai, Approved by AICTE, New Delhi)

Samayapuram – 621 112

DECEMBER, 2024

K RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(AUTONOMOUS)

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report titled “**FARM MANAGEMENT SYSTEM USING MYSQL & PYTHON FLASK**” is bonafide work of **SANJAY A (811722104128), VISHNU KARTHIC R (811722104186), KARTHIKEYAN S (811722104303)** who carried out the project under my supervision. Certified further, that to the best of my knowledge the work reported here in does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr.A Delphin Carolina Rani M.E.,Ph.D.,

HEAD OF THE DEPARTMENT

PROFESSOR

Department of CSE

K Ramakrishnan College of Technology

(Autonomous)

Samayapuram – 621 112

SIGNATURE

Dr.A Delphin Carolina Rani M.E.,Ph.D.,

SUPERVISOR

Assistant Professor

Department of CSE

K Ramakrishnan College of Technology

(Autonomous)

Samayapuram – 621 112

Submitted for the viva-voice examination held on

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION

We jointly declare that the project report on “**FARM MANAGEMENT SYSTEM USING MYSQL & PYTHON FLASK**” is the result of original work done by us and best of our knowledge, similar work has not been submitted to “**ANNA UNIVERSITY CHENNAI**” for the requirement of Degree of Bachelor Of Engineering. This project report is submitted on the partial fulfilment of the requirement of the award of Degree of Bachelor Of Engineering.

Signature

SANJAY A

VISHNU KARTHIC R

KARTHIKEYAN S

Place: Samayapuram

Date:

ACKNOWLEDGEMENT

It is with great pride that we express our gratitude and in-debt to our institution “**K RAMAKRISHNAN COLLEGE OF TECHNOLOGY**”, for providing us with the opportunity to do this project.

We are glad to credit honorable chairman **Dr. K. RAMAKRISHNAN, B.E.**, for having provided for the facilities during the course of our study in college.

We would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.**, for forwarding our project and offering adequate duration to complete it.

We would like to thank **Dr. N. VASUDEVAN, M.Tech., Ph.D.**, Principal, who gave opportunity to frame the project with full satisfaction.

We whole heartily thank **Dr. A DELPHIN CAROLINA RANI, M.E., Ph.D.**, Head of the Department, **COMPUTER SCIENCE AND ENGINEERING** for providing her support to pursue this project.

We express our deep and sincere gratitude and thanks to our project guide **Dr. A. DELPHIN CAROLINA RANI, M.E., Ph.D.**, Department of **COMPUTER SCIENCE AND ENGINEERING**, for his incalculable suggestions, creativity, assistance and patience which motivated us to carry this project.

We render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course. We wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

ABSTRACT

The "Farm Management System Project" is a web-based application designed to support farmers by providing agricultural information, fostering knowledge sharing, and enhancing productivity and profitability. It enables farmers to sell their products online, purchase tools and seeds directly from sellers, and manage their profiles with options to register, edit, or delete data. The platform promotes collaboration by sharing best-practice farming methods and connecting farmers with buyers through an integrated online marketplace. Buyers can browse and purchase agricultural products and send quality-check requests via email, ensuring transparency and trust in transactions. This system eliminates intermediaries, ensures fair pricing, and streamlines procurement processes, making agriculture more efficient and profitable. By integrating technology with farming, the project aims to create a sustainable and connected agricultural ecosystem. It bridges the gap between producers and consumers while driving innovation and fostering growth in the agricultural sector.

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE NO
	ABSTRACT	v
	LIST OF FIGURES	viii
	LIST OF ABBREVIATIONS	ix
1	INTRODUCTION	1
	1.1 Background	1
	1.2 Overview	2
	1.3 Problem Statement	3
	1.4 Objective	4
	1.5 Implication	4
2	LITERATURE SURVEY	6
3	SYSTEM ANALYSIS	9
	3.1 Existing System	9
	3.2 Proposed System	10
	3.3 Block Diagram for Proposed System	11
	3.4 Flowchart	12
	3.5 Process Cycle	13
	3.6 Activity Diagram	14
4	MODULES	15
	4.1 Module Description	15
	4.1.1 User authentication	15
	4.1.2 Database Interaction Module	16
	4.1.3 Product Management Module	16
	4.1.4 Farmer Registration Module	17
	4.1.5 Farming Type Management Module	17
	4.1.6 Order and Purchase Module	18
	4.1.7 Trigger Module	19
	4.1.8 Admin Control Module	20
	4.1.9 User Interface Module	21
	4.1.10 Communication Module	22

5	SYSTEM SPECIFICATION	24
	5.1 Software Requirements	24
	5.1.1 PyCharm	24
	5.1.2 Xampp	25
	5.1.3 SQLAlchemy	26
	5.1.4 Python Flask	27
	5.2 Hardware Requirements	29
6	METHODOLOGY	30
	6.1 Requirement Gathering and Analysis	30
	6.2 System Design	30
	6.3 Frontend and Backend Development	30
	6.4 Database Design and Integration	31
	6.5 Security and Authentication	31
	6.6 Testing and Quality Assurance	32
	6.7 Deployment and Maintenance	32
	6.8 Agile Development Approach	32
7	CONCLUSION AND FUTURE ENHANCEMENT	33
	7.1 Conclusion	33
	7.2 Future Enhancement	33
	APPENDIX-1	
	APPENDIX-2	
	REFERENCES	

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
1.1	Flow of Farm management	1
3.1	Use case diagram	11
3.2	Flow chart	12
3.3	Process cycle	13
3.4	Action Sequence structure of gesture control	14
7.1	Home Page	42
7.2	Sign up Page	42
7.3	Registration Page	43
7.4	Product View	43
7.5	Trigger Record	44
7.6	Farming Type	44
7.7	Farmer Details	45
7.8	Add Product	45
7.9	Database Page	46
7.10	Trigger Records in Php myAdmin	46
7.11	Argo Products Records	47
7.12	User Records	47

LIST OF ABBREVIATIONS

ABBREVIATION	FULL FORM
API	Application Programming Interface
DB	Database
IOT	Internet of Things
UI	User Interface
HTML	Hyper Text Markup Language
CSS	Cascading Style Sheet
GPS	Global Positioning System
JS	Java Script

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

Agriculture is the backbone of many economies, contributing significantly to employment and food production. However, farmers often face challenges such as inefficient resource management, lack of access to markets, and limited knowledge of modern farming practices. To address these issues, technology-driven solutions have become increasingly vital. The Farm Management System was developed as a comprehensive web application to assist farmers in managing their agricultural activities more effectively. The system aims to bridge the gap between traditional farming methods and modern technology by providing tools to enhance productivity, profitability, and accessibility to markets.

This project focuses on empowering farmers by enabling them to sell their products online, purchase necessary tools and seeds, and access valuable agricultural information. Additionally, the system facilitates buyers in sourcing quality agricultural products, fostering a mutually beneficial ecosystem. By integrating a user-friendly interface and leveraging robust backend technologies, the project ensures seamless interaction and improved decision-making for all stakeholders in the agricultural value chain.

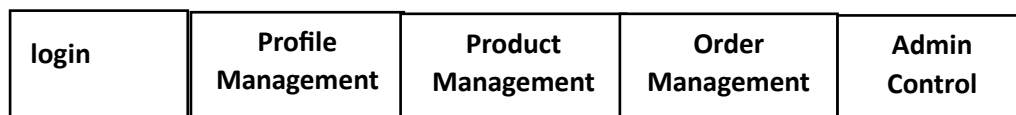


Figure 1.1: Flow of farm management

1.2 OVERVIEW

The Farm Management System is a comprehensive, web-based platform designed to modernize agricultural practices and improve the overall efficiency of farming operations. Recognizing the challenges faced by farmers—such as inadequate access to modern farming tools, difficulty in reaching potential buyers, and limited knowledge of efficient farming techniques—the system serves as a bridge between traditional agricultural methods and modern technological solutions. The platform empowers farmers by providing them with the necessary tools to streamline their operations, increase productivity, and enhance their profitability.

Farmers can register on the platform, create and manage their profiles, and list the products they wish to sell. This digital marketplace allows farmers to reach a broader audience, breaking down the barriers of traditional selling methods. The system also enables farmers to purchase farming tools, seeds, and other essential supplies from trusted vendors, ensuring that they have the resources required for successful crop cultivation. Additionally, the platform offers educational resources and best practices to help farmers adopt new techniques that increase crop yield and sustainability. By creating a space where farmers can easily interact with each other and access relevant information, the platform fosters a sense of community and collaboration within the agricultural industry.

On the other side, buyers benefit from the system by having access to a wide variety of fresh, locally-grown agricultural products. The platform allows buyers to browse through product listings, make informed purchasing decisions, and communicate directly with farmers for queries or negotiations. One unique feature of the platform is the ability for buyers to send quality-check requests via email, ensuring that they are receiving high-quality products that meet their expectations. This feature builds trust between farmers and buyers, further enhancing the market's transparency and fostering long-term relationships.

The Farm Management System is built using modern technologies to ensure a seamless user experience. The backend is powered by Python Flask, a lightweight framework known for its simplicity and scalability. This ensures that the system is not only fast and responsive but also able to handle increasing user loads as the platform grows. SQLAlchemy is used for database management, providing efficient storage and retrieval of data, from user profiles to product listings and transaction records. The frontend, designed with HTML, CSS, JavaScript, and Bootstrap, ensures that the platform is fully responsive, making it accessible on a wide

range of devices, from desktops to mobile phones. The clean, user-friendly interface provides farmers and buyers with easy navigation, ensuring that both can access the system's features without confusion.

By integrating modern technologies with traditional farming practices, the Farm Management System has the potential to transform the way agriculture is conducted. It creates a more accessible and efficient market for both farmers and buyers, leading to better resource management, increased sales opportunities for farmers, and improved access to fresh, high-quality produce for buyers. This not only boosts local economies but also contributes to the sustainability of farming by promoting responsible practices, improving crop yields, and encouraging the use of eco-friendly resources. Ultimately, the platform serves as a key tool in reshaping the future of agriculture, ensuring that both farmers and buyers benefit from a more connected and efficient agricultural ecosystem.

1.3 PROBLEM STATEMENT

The objective of the Farm Management System Project is to develop a comprehensive web application that assists farmers in managing various aspects of their farming activities, including tracking farming information, selling agricultural products online, and managing their profiles. The system should provide a seamless experience for farmers to register, edit, and delete their profiles, while buyers can also browse products, make purchases, and request quality checks.

Key functionalities of the system include:

1. **Farmer Registration and Profile Management:** Farmers can register on the platform, providing personal details such as their name, Aadhaar number, age, gender, contact details, and farming type. They can update their profile and delete it if necessary.
2. **Agricultural Product Management:** Farmers can add agricultural products for sale, including product name, description, and price. These products can be listed for potential buyers to view and purchase.
3. **Trigger Alerts:** The system should allow generating triggers for specific actions, such as when a product is purchased or when a farming activity is updated.
4. **Buyer Interaction:** Buyers can view the products listed by farmers and make online purchases. They can also request a quality check for products via email.

1.4 OBJECTIVE

The goal of this project is to build an intuitive, secure, and scalable web application for farmers and buyers, enabling efficient management of farming activities and online product sales. The system should provide real-time notifications, allow easy access to farming-related information, and facilitate seamless transactions.

Key objectives include:

1. Seamless User Authentication
2. Efficient Product Management
3. Improved Buyer Interaction
4. Timely Notifications
5. User-Friendly Interface

1.5 IMPLICATION

The implementation of the Farm Management System includes the following key components:

1. User Authentication and Authorization:
 - Flask-Login is used to handle user sessions and authentication for both farmers and administrators.
 - A secure sign-up and login system is implemented to ensure that only authorized users can access the platform.
2. Database Design:
 - The system is backed by a MySQL database, with tables for user information (farmers and buyers), agricultural products, farming details, and triggers.
 - SQLAlchemy ORM is used for easy interaction with the database, including adding, editing, and deleting records.
3. Farmer Profile Management:
 - Farmers can add and edit their personal details and farming information.
 - They can also add agricultural products, with details like product name, description, and price.
4. Product Listings and Purchases:
 - Farmers can list their products for sale, which can then be viewed and purchased by buyers.

- Each product can be associated with a price and description to provide more information to potential buyers.

5. Triggers and Notifications:

- The system generates triggers based on specific actions (e.g., product purchase) and sends notifications to the relevant parties.

6. Web Interface:

- The front end is built using HTML, CSS, JavaScript, and Bootstrap to ensure a responsive and user-friendly interface.
- Flask is used to handle routing and serve the pages, while templates are rendered dynamically based on the data from the backend.

CHAPTER 2

LITERATURE SURVEY

TITLE : A Web-Based Farm Management System for Enhancing Agricultural Productivity

AUTHORS: R. K. Yadav, R. S. Saini, R. K. Choudhary

YEAR: 2021

This paper discusses a web-based farm management system designed to improve agricultural productivity. The system integrates features for managing crops, fertilizers, irrigation, and weather conditions, thereby assisting farmers in decision-making. The study highlights how technology can be applied to simplify farm management tasks and improve efficiency. By using cloud-based tools and a user-friendly interface, farmers can easily track field activities and crop conditions. This paper serves as a foundation for the development of web-based applications aimed at optimizing agricultural processes.

TITLE: IoT-Based Smart Agriculture System for Farm Management

AUTHORS: M. K. Garg, R. S. A. S. Vora

YEAR: 2019

The research focuses on the application of the Internet of Things (IoT) in agricultural management. It discusses how IoT devices like sensors, actuators, and cloud computing platforms can be integrated into farm management systems to automate tasks such as irrigation, monitoring soil quality, and crop management. This paper provides insights into the hardware and software components needed to implement a smart farming solution, allowing farmers to efficiently manage resources and improve crop yield. The proposed system reduces manual intervention and helps in making data-driven decisions.

TITLE: Web-Based Farm Management System: A Review

AUTHOR: M. F. Raza, S. M. Iqbal, H. Shah

YEAR: 2018

This paper reviews the use of web-based farm management systems in modern agriculture. It outlines the key features such systems should have, including crop management, resource scheduling, and tracking farm performance. The paper also reviews several existing web-based solutions for farm management, evaluating their usability and efficiency in different farming environments. It further highlights the advantages of web-based systems over traditional paper-based farm management methods, especially in terms of accessibility, real-time data tracking, and decision-making support.

TITLE: A Cloud-Based Farm Management System for Agricultural Data Management and Decision Support

AUTHOR: S. A. Hassan, M. S. B. Zubair, A. N. Malik

YEAR: 2020

This paper presents a cloud-based farm management system that helps farmers manage data related to crop planting, irrigation schedules, and harvesting. The system integrates weather data, market trends, and other relevant agricultural information, providing farmers with actionable insights to improve farm productivity. The system's cloud architecture ensures that farmers can access critical data remotely from any device, empowering them with real-time information for better decision-making. The research emphasizes the importance of cloud technology in supporting sustainable and efficient farming practices.

TITLE: A Web-Based Agricultural Management System for Farmers

AUTHOR: A. R. P. Borkar, P. P. Rathi, V. M. Kher

YEAR: 2017

This study explores the development and implementation of a web-based agricultural management system designed to help farmers in various aspects of farm management, such as crop planning, sales tracking, and inventory management. It discusses the architecture of the

system, the technologies used, and the benefits for farmers. The system offers a central hub for managing farm resources and facilitating communication between farmers, suppliers, and buyers, contributing to increased productivity and market reach. The paper concludes that the adoption of such systems can significantly enhance the efficiency of farming operations.

-TITLE: Smart Farm Management System Using Machine Learning and IoT

AUTHOR: A. S. Verma, S. S. Kapoor, R. K. Meena

YEAR: 2021

This paper discusses the integration of machine learning (ML) and IoT technologies in smart farm management systems. The system leverages IoT-based sensors to collect real-time data about environmental factors such as temperature, humidity, and soil moisture. Machine learning models are then used to predict crop growth and yield based on these data inputs, helping farmers make informed decisions about crop rotation, irrigation, and pest control. The research demonstrates how advanced technologies can assist in optimizing farm operations and increasing agricultural productivity.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

The existing system, developed by Source Trace in collaboration with the Small Farmers Agri-business Consortium (SFACH) and the Karnataka Horticulture Department, aims to support the horticulture farmers of India, particularly those in the Karnataka region. The core objective of the system is to create comprehensive farmer profiles, focusing on both the welfare of individual farmers and the management of Farmer Producer Organizations (FPOs). This initiative is aligned with the growing recognition of the potential of Information Technologies (IT) to enhance agricultural practices and improve the overall livelihood of farmers in rural India.

The system is built using a combination of Python Flask for the backend, along with HTML, CSS, JavaScript, and MySQL for the user interface and database management. These technologies work seamlessly together to create a platform that allows farmers to register and maintain their profiles, which includes details about their farming practices, crops, and personal information. By offering an easy-to-use interface, the system ensures that even farmers with limited technical expertise can effectively manage their profiles and access relevant resources. Additionally, the system is designed to handle large datasets, aiming to support 100,000 farmer profiles as part of its initial deployment phase.

One of the key strengths of the system is its efficiency. It is built to be error-free, ensuring that farmers can rely on accurate data for decision-making. The system is optimized for speed, minimizing the time required to perform tasks such as data entry, updating profiles, and generating reports. This ensures that farmers can focus more on their work rather than dealing with technical issues. Moreover, the system is designed to be robust, able to withstand heavy usage and scale as more users and features are added over time. This is especially important as the system will eventually expand to support not just individual farmers but also Farmer Producer Organizations (FPOs), which are essential for collective farming efforts and distribution.

The system is developed using the full software development lifecycle, ensuring it is thoroughly tested, maintained, and updated regularly. As a result, it is highly reliable and well-suited to the dynamic needs of Indian farmers. Furthermore, the system is designed with future growth in mind. There are provisions for incorporating new features and technologies as they emerge. For example, there is potential to integrate advanced tools like real-time data analytics and machine learning algorithms to provide farmers with deeper insights into weather patterns, crop yields, and market trends. These innovations would help farmers make more informed decisions, ultimately improving productivity and profitability.

In summary, the existing system is a comprehensive digital solution designed to improve the lives of horticulture farmers in India. By leveraging modern web technologies, it provides a user-friendly platform for managing farmer profiles, enhancing productivity, and facilitating better management of Farmer Producer Organizations. Its efficiency, robustness, and scalability make it a powerful tool for the future of agriculture in India, and the provisions for future development ensure that the system can continue to evolve in response to changing agricultural needs.

3.2 PROPOSED SYSTEM

The Farm Management System is designed to connect farmers directly with customers, allowing them to sell agricultural products online while ensuring transparency and quality. The system serves two main types of users: Farmers and Customers. Farmers can list their agricultural products, such as fruits, vegetables, and grains, and provide details like product descriptions, pricing, and availability. They can also share articles about farming practices, tips for improving productivity, and updates on their operations, which help build credibility and increase sales. On the other hand, Customers browse through the available products, and before purchasing, they can send purchase requests to check the quality of the products or ask for additional details. This feature ensures customers are confident in their purchase before committing. Once the quality is verified, customers can finalize their purchase, completing a secure transaction. The system requires users to log in using a username and password to ensure secure access and protect personal and transaction data.

3.3 BLOCK DIAGRAM OF PROPOSED SYSTEM

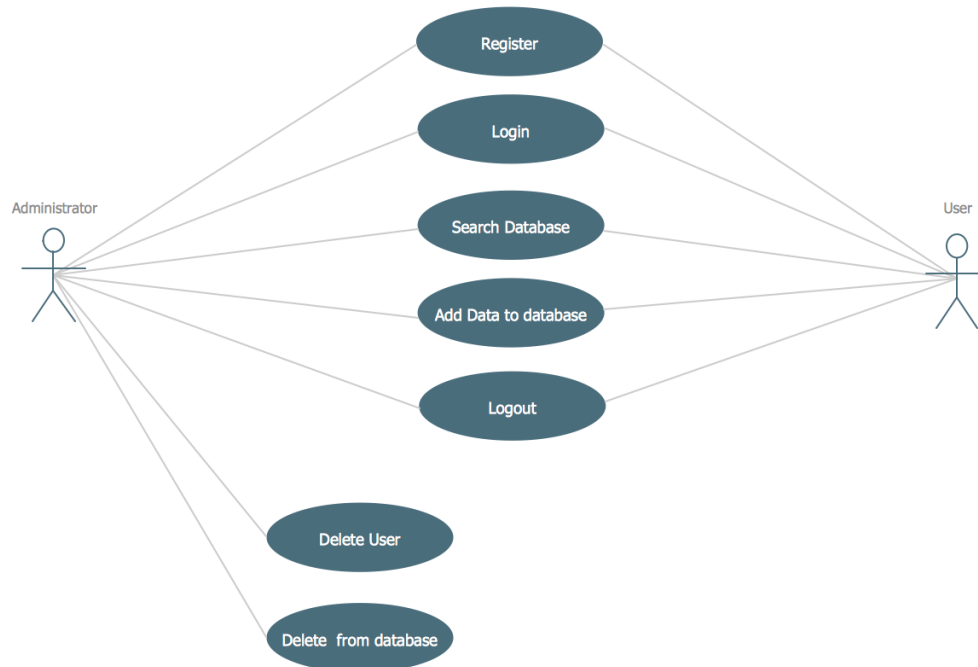


Figure 3.1: Usecase Diagram

3.4 FLOWCHART

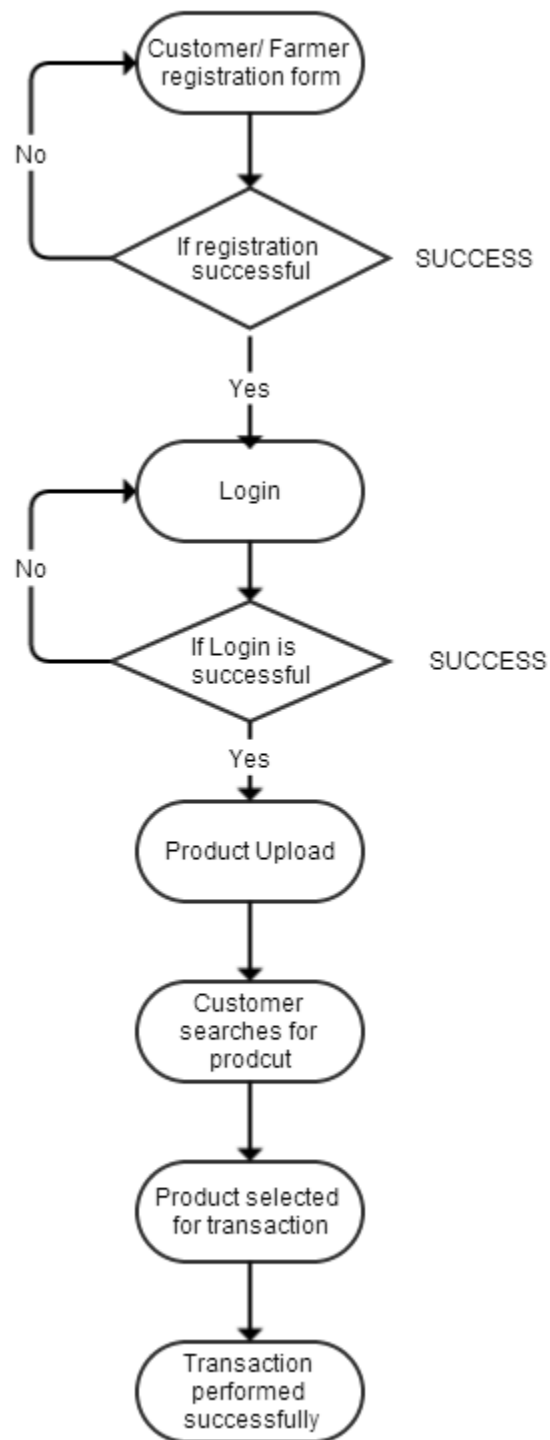


Figure 3.2: Flow of Control

3.5 PROCESS CYCLE

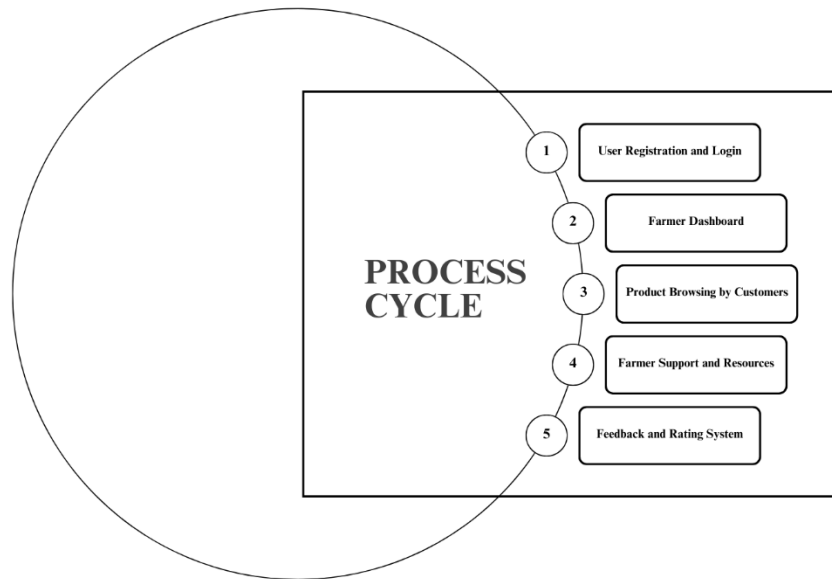


Figure 3.3: Life Cycle of the Process

3.6 ACTIVITY DIAGRAM

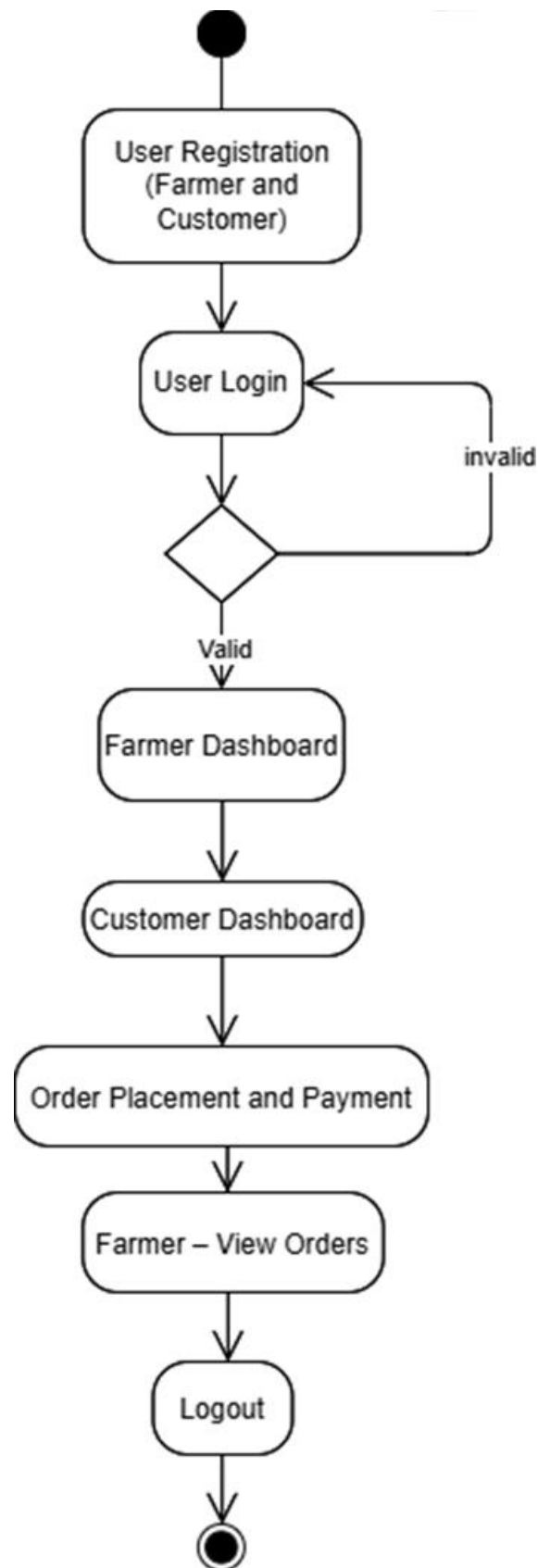


Figure 3.4: Action Sequence Structure of farm management system

CHAPTER 4

MODULES

4.1 MODULE DESCRIPTION

- User Authentication Module
- Database Interaction Module
- Product Management Module
- Farmer Registration and Management Module
- Farming Type Management Module
- Order Management and Purchase Request Module
- Trigger and Logging Module
- Admin Control and Management Module
- User Interface (UI) Module
- Communication Module

4.1.1 USER AUTHENTICATION MODULE

The User Authentication Module in the Farm Management System ensures secure access and user management. It enables users to register, log in, and maintain sessions within the platform. During registration, users provide essential details like username, email, and password, with the system verifying the email's uniqueness and securely storing the password using hashing. The login process allows registered users to authenticate by comparing the entered credentials with the stored data. Once logged in, user sessions are managed using Flask-Login, which ensures that only authorized users can access restricted pages. Additionally, the module controls user access based on roles, such as farmers, customers, and admins, granting appropriate permissions for each. Flash messages are used to provide feedback during actions like successful logins or errors, while password hashing ensures that user data remains secure. This module plays a crucial role in maintaining the integrity and security of the application, allowing users to interact with the system safely and efficiently.

4.2 DATABASE INTERACTION MODULE

The Database Interaction Module in the Farm Management System is responsible for facilitating the interaction between the application and the underlying database, ensuring smooth data retrieval, storage, and management. It uses SQLAlchemy, a powerful Object-Relational Mapping (ORM) tool, to handle all database operations. This module includes various models that represent the key entities of the system, such as users, farmers, agro-products, farming types, and triggers.

When users perform actions like registering, adding products, or updating records, the module interfaces with the database to store, modify, or retrieve data. For instance, when a farmer registers or updates their profile, the data is saved in the "Register" table, and when a new agro-product is added, it is stored in the "Addagroproducts" table. The system supports querying, inserting, updating, and deleting records, which is essential for maintaining the dynamic nature of the platform. Through structured queries, the module enables the display of relevant data on various pages, such as listing all registered farmers or showing available agro-products. Moreover, the module ensures data integrity by using SQL Alchemy's session management, which allows for safe commits and rollbacks. This interaction with the database ensures that all user interactions with the system are reflected in real-time and that the data remains consistent, accurate, and secure across the application.

4.3 PRODUCT MANAGEMENT MODULE

The Product Management Module in the Farm Management System is designed to handle all aspects of managing agricultural products on the platform. This module allows farmers to list their products, update details, and manage product availability, pricing, and descriptions. It supports farmers in adding new agro-products such as fruits, vegetables, seeds, and tools, by capturing essential details such as product name, description, price, and the farmer's information.

The system uses a dedicated database table, Add agro products, to store product information, which is then made available for potential buyers to browse through. When farmers add a new product, the module ensures that all necessary data is validated and stored correctly, allowing the products to be displayed on the platform. The Product Management Module also provides features for updating and deleting products, ensuring that farmers can

keep their product listings up to date. By leveraging the module, farmers can effectively manage their inventory and offer a variety of products to consumers, while buyers can view product details, making informed purchasing decisions.

Additionally, the module integrates with other parts of the system, such as the Database Interaction Module, to ensure that the product data is properly synchronized and that any updates or deletions are reflected across the platform. Overall, the Product Management Module enhances the functionality of the Farm Management System by streamlining the process of product listing and ensuring that both farmers and buyers have access to accurate, real-time product information.

4.4 FARMER REGISTRATION AND MANAGEMENT MODULE

The Farmer Registration and Management Module is a critical component of the system that allows farmers to register and manage their profiles on the platform. This module enables farmers to provide essential details such as their name, contact information, farming practices, and location. It includes a registration form where farmers can input their personal and farming information, which is then stored securely in the system's database. Once registered, farmers can access the system, update their details, and view or modify their profile as needed. The module also allows administrators to manage farmer records, ensuring that all data is accurate and up to date. By providing an easy-to-use interface, this module helps farmers stay connected with the platform, ensuring that they can sell products, manage their inventory, and receive support for their agricultural activities. It forms the foundation of the platform's user base, ensuring that farmers are adequately represented and able to participate in all services the system offers.

4.5 FARMING TYPE MANAGEMENT MODULE

The Farming Type Management Module not only serves as a classification system for different farming practices but also plays a pivotal role in enhancing the user experience and streamlining the system's operations. By categorizing farming types, this module allows the platform to offer tailored services, tools, and resources to farmers based on their specific farming methods. For example, farmers engaged in organic farming can receive specialized advice on crop rotation and pest management, while those involved in livestock farming can

access information about animal husbandry and health. This customization ensures that the farmers' unique needs are met, making the platform more valuable and relevant to their day-to-day operations.

Moreover, this module enables administrators to add new farming types as agriculture evolves, ensuring that the system stays current with emerging trends and practices. It also facilitates updates and corrections to existing farming types, allowing for a dynamic and responsive system that adapts to changing needs in the agricultural sector.

For farmers, the module provides the ability to select or modify their farming type through their profile, ensuring that the platform's recommendations and content align with their specific field of expertise. Additionally, the farming type data can be used for segmentation, allowing the system to organize farmers into different groups or networks based on the type of farming they practice, encouraging knowledge-sharing and collaboration.

By maintaining accurate and up-to-date farming type classifications, this module also plays a critical role in inventory management, product sales, and agricultural services, linking the right products and services to the right farming practices. In essence, the Farming Type Management Module enhances the platform's effectiveness by offering farmers personalized support while ensuring the system can adapt to the changing landscape of agriculture.

4.6 ORDER MANAGEMENT AND PURCHASE REQUEST MODULE

The Order Management and Purchase Request Module is a critical component of the Farm Management System, designed to streamline and automate the entire order and purchase process for both farmers and buyers. This module allows farmers to efficiently manage the sale of their products by creating detailed product listings, which are visible to potential buyers. It ensures that farmers can track orders, monitor stock levels, and manage product availability in real-time, thereby reducing the risk of stockouts or overstocking. The system also allows farmers to update product information, including pricing and descriptions, to reflect market changes.

For buyers, the module facilitates a seamless purchasing experience, enabling them to browse through various agricultural products and place orders directly through the platform. Buyers can easily search for specific products, check prices, and review detailed descriptions before making a purchase decision. The module includes features for adding items to a shopping cart, providing payment options, and confirming orders.

One of the key features of the module is the purchase request functionality, where buyers can submit requests for specific products or tools that are not available on the platform at the moment. This feature encourages interaction between buyers and farmers, creating opportunities for custom orders or sourcing hard-to-find agricultural products. Once a request is received, farmers can respond with quotes, availability, and estimated delivery times, allowing for negotiation and personalized service.

Additionally, the module supports order tracking, providing both farmers and buyers with real-time updates on the status of orders, from confirmation to shipping and delivery. It also allows for the management of returns, refunds, or exchanges, ensuring that both parties are satisfied with the transaction.

This comprehensive Order Management and Purchase Request Module not only improves the efficiency and transparency of the buying and selling process but also fosters trust and long-term relationships between farmers and buyers, making it a key feature of the platform's success.

4.7 TRIGGER AND LOGGING MODULE

The Trigger and Logging Module is an essential component of the Farm Management System, designed to monitor and track all significant events, actions, and processes within the platform. This module plays a critical role in ensuring transparency, accountability, and system reliability. It captures various triggers or actions initiated by users, such as product listings, orders, registrations, and updates, and logs them for future reference. These logs can be vital for auditing purposes, troubleshooting issues, and maintaining a secure environment for all users.

The trigger functionality within the module is responsible for responding to specific actions or changes in the system. For instance, when a new product is added by a farmer, a trigger may be set off to notify potential buyers or update the stock levels. Similarly, if a buyer places an order or submits a purchase request, the system triggers automated responses to confirm the order, update the inventory, and notify relevant parties. This automation streamlines the operational flow and ensures that no action goes unnoticed or unaddressed. The logging feature, on the other hand, records every event and action, from user logins and registrations to order processing and farming updates. These logs are stored securely and can be accessed by administrators or support teams to investigate issues, analyze system performance, or identify security threats. The logs also include timestamps, user information, and detailed descriptions of the actions performed, ensuring that every interaction within the system is documented for accountability.

In addition to operational tracking, the Trigger and Logging Module also facilitates reporting and analytics by providing insights into system usage, user behavior, and product trends. This data is invaluable for improving the platform's functionality, optimizing performance, and making data-driven decisions that enhance the user experience. Overall, the Trigger and Logging Module ensures that all critical actions within the Farm Management System are efficiently managed and thoroughly recorded, contributing to the system's overall reliability, transparency, and effectiveness.

4.8 ADMIN CONTROL AND MANAGEMENT MODULE

The Admin Control and Management Module is a pivotal component of the Farm Management System, providing administrators with full control and oversight over the platform's operations. This module allows administrators to manage user accounts, including farmers, buyers, and system users, ensuring that all interactions within the system are regulated and secure. With this module, admins can perform essential tasks such as adding or removing users, modifying user details, and managing permissions to control access to various features and functionalities of the system.

Furthermore, the Admin Control and Management Module facilitates the management of farming types, agro products, and other critical data entries. Administrators can add, update, or delete farming types, ensuring that the list of farming practices is always relevant and

accurate. Similarly, they can manage the products listed by farmers, reviewing product information, pricing, and availability to maintain quality and consistency in the marketplace. This module also plays a vital role in monitoring and controlling the flow of orders and purchase requests. Admins can view and process customer orders, manage inventory, and ensure timely deliveries. Additionally, it allows for the resolution of any issues related to transactions, product quality, or disputes, providing a central point of control for customer service operations.

The Admin Control and Management Module also incorporates advanced features such as data analytics, reporting, and system performance tracking. Admins can generate reports on user activities, product sales, market trends, and system usage, providing valuable insights that help in decision-making and improving the platform's overall performance. The module also includes security controls to monitor and manage suspicious activities, helping to prevent fraud and maintain a secure environment for all users.

In essence, the Admin Control and Management Module ensures that the administrators have all the necessary tools to efficiently manage the farm management system, maintain smooth operations, enforce rules and regulations, and support the overall growth and success of the platform.

4.9 USER INTERFACE (UI) MODULE

The User Interface (UI) Module is a critical component of the Farm Management System, designed to offer an intuitive and seamless experience for users, including farmers, buyers, and administrators. This module is responsible for presenting the system's functionality in a clear, user-friendly manner, ensuring that users can navigate through the platform with ease. The UI module integrates all the features of the system, such as product management, order processing, and farmer registration, into a visually appealing and accessible interface.

The UI module prioritizes simplicity and usability, ensuring that users, regardless of their technical proficiency, can interact with the system without confusion. For example, farmers can easily manage their product listings, update their profiles, and track orders through straightforward forms and dashboards.

Buyers benefit from easy access to product listings, order tracking, and secure payment options, all presented in a clean and responsive layout. Administrators are provided with dedicated dashboards for overseeing system performance, managing users, and generating reports.

In addition to being visually appealing, the UI Module is designed to be responsive, adapting to different devices and screen sizes. Whether accessed on a desktop, tablet, or smartphone, the platform ensures a consistent experience across all devices, making it convenient for users to access the system on the go. The UI also incorporates modern web design principles, such as the use of interactive elements, dynamic content, and easy-to-understand icons, improving the overall user engagement.

The UI Module also works in conjunction with the backend to provide real-time data updates and feedback to users. For instance, when a farmer adds a new product, the interface immediately reflects the change, providing instant confirmation of the action. Similarly, buyers receive real-time notifications regarding their orders, ensuring that they are always up to date on the status of their purchases.

Overall, the User Interface Module enhances the overall user experience by ensuring the platform is accessible, easy to navigate, and visually engaging, making it a vital aspect of the Farm Management System that contributes to user satisfaction and effective platform utilization.

4.10 COMMUNICATION MODULE

The Communication Module is an essential component of the Farm Management System, enabling seamless interaction between different users—farmers, buyers, and administrators—while ensuring the effective exchange of information within the platform. This module facilitates real-time communication through various channels, such as email notifications, system alerts, and messaging features, enhancing the overall user experience and operational efficiency.

One of the primary functions of the Communication Module is to notify users about important actions or events. For instance, when a farmer successfully registers a new product or updates their profile, the system can send automated email notifications to confirm the

action. Similarly, when a buyer places an order or submits a purchase request, the system sends instant notifications to both the farmer and the buyer, ensuring they are both informed of the transaction details.

The Communication Module also plays a critical role in handling inquiries and requests. Buyers can communicate directly with farmers through messaging features, which allow them to ask questions about products, request quality checks, or negotiate prices. Similarly, farmers can reach out to buyers with updates or follow-up messages. This ensures a transparent and open line of communication, fostering trust and collaboration between the two parties.

For administrators, the Communication Module offers tools to monitor and manage system-wide notifications, such as account activity, order status updates, and system alerts. The module may also include an automated support feature that helps resolve common issues or FAQs, improving the efficiency of customer support.

In addition, the Communication Module is designed with scalability in mind, allowing it to integrate with third-party communication platforms, such as SMS services or external email providers, to expand its functionality as the system grows. This ensures that users, regardless of their location or preferred communication method, can stay connected and informed in a timely and efficient manner.

Overall, the Communication Module is a pivotal feature that ensures smooth, real-time communication between users, enhancing the operational flow and providing an interactive platform for both farmers and buyers.

CHAPTER 5

SYSTEM SPECIFICATION

5.1 SOFTWARE REQUIREMENTS

Frontend- HTML, CSS, Java Script, Bootstrap

Backend-Python flask (Python 3.7) , SQLAlchemy,

- Operating System: Windows 10
- Google Chrome/Internet Explorer
- XAMPP (Version-3.7)
- Python main editor (user interface): PyCharm Community
- workspace editor: Sublime text 3

5.1.1 PYCHARM

PyCharm is a robust integrated development environment (IDE) designed specifically for Python programming, developed by JetBrains. It offers a wide array of features that enhance productivity and streamline the development process for both beginners and experienced developers. Key features include intelligent code completion, on-the-fly error checking, and quick-fixes that significantly speed up coding tasks. PyCharm supports web development frameworks such as Django and Flask, enabling developers to build complex web applications efficiently. Its powerful debugging tools, including a visual debugger and integrated testing capabilities, help in quickly identifying and resolving issues. PyCharm also provides seamless integration with version control systems like Git, facilitating collaborative development and code management. The IDE's customizable interface and extensive plugin ecosystem allow developers to tailor their environment to suit specific needs. Additionally, PyCharm supports scientific computing with integrations for Jupyter notebooks, Anaconda, and scientific libraries like NumPy and pandas. With its comprehensive toolset, PyCharm is a preferred choice for Python developers seeking a versatile and efficient development environment.

5.1.2 XAMPP

XAMPP is a versatile, open-source software package that provides an integrated local server environment for developing, testing, and deploying web applications. The acronym XAMPP stands for Cross-Platform (X), Apache (A), MariaDB/MySQL (M), PHP (P), and Perl (P), representing the essential components that power dynamic web applications. By bundling these technologies into a single package, XAMPP simplifies the development process and offers a streamlined solution for developers working on diverse platforms, including Windows, macOS, and Linux.

At its core, XAMPP includes the Apache HTTP Server, which enables developers to simulate a live server environment locally. This allows applications to be tested and debugged without requiring internet connectivity or access to a remote server. Apache handles requests and delivers web pages, ensuring that developers can evaluate how their applications will behave in a production-like setting. This feature is invaluable for web development projects, where iterative testing and debugging are crucial for delivering high-quality applications.

The inclusion of MariaDB/MySQL in XAMPP facilitates robust database management. This component is critical for projects that require structured storage and retrieval of data, such as user authentication systems, inventory management, and transaction processing. XAMPP simplifies database management through phpMyAdmin, a web-based interface that enables developers to create, modify, and query databases with ease. With its intuitive user interface, phpMyAdmin makes it accessible for both novice and experienced developers to work with relational databases effectively.

XAMPP also integrates support for PHP and Perl, two powerful programming languages commonly used in web development. PHP powers the backend logic, enabling features like form processing, user authentication, and dynamic content generation. Perl, on the other hand, is used for tasks such as server-side scripting and network programming. The seamless integration of these languages within the XAMPP environment ensures that developers have all the tools they need to create feature-rich web applications.

One of the most significant advantages of XAMPP is its cross-platform compatibility, which allows developers to set up and work on their projects across multiple operating systems without worrying about compatibility issues. This flexibility makes XAMPP a preferred choice for collaborative projects where team members might use different platforms.

For your project, XAMPP plays a vital role in managing backend operations and database interactions. The MySQL/MariaDB database provided by XAMPP can store crucial information such as farmer profiles, product details, user credentials, and purchase transactions. With its local server setup, XAMPP enables you to test all database queries, user actions, and backend processes in a secure, offline environment. Additionally, features like email integration and purchase requests can be tested using XAMPP's Apache server capabilities.

Overall, XAMPP is an indispensable tool for developers, providing a complete, easy-to-install package for setting up a development environment. Its ability to simulate a live server, manage databases, and support multiple programming languages makes it a cornerstone of modern web development workflows.

5.1.3 SQLAlchemy

SQLAlchemy is a comprehensive Python library that simplifies database management and interaction. It acts as both an Object Relational Mapper (ORM) and a Core SQL toolkit, providing developers with powerful tools to work with relational databases. SQLAlchemy bridges the gap between Python objects and database tables, enabling seamless integration and simplifying the process of querying, updating, and managing data. Its dual approach—abstracting database interactions while allowing raw SQL capabilities—makes it an ideal choice for developers seeking flexibility and efficiency.

One of the standout features of SQLAlchemy is its Object Relational Mapper (ORM), which allows developers to represent database tables as Python classes. Rows in the database correspond to instances of these classes, enabling intuitive and object-oriented manipulation of data. Instead of writing SQL queries for common operations, developers can use Python syntax to perform actions like filtering, updating, or deleting records. For instance, retrieving a user with a specific ID can be done using ``session.query(User).filter_by(id=1).first()```—a clear and concise alternative to traditional SQL.

SQLAlchemy also excels in providing a powerful Core SQL Expression Language for scenarios where direct SQL usage is necessary. This feature enables the construction of complex SQL queries using Python objects, offering the flexibility to work with raw SQL while benefiting from SQLAlchemy's abstractions. This dual capability ensures that developers can

seamlessly transition between high-level abstractions and low-level SQL as per their requirements.

Another significant advantage of SQLAlchemy is its database-agnostic design. It supports a wide range of relational databases, including MySQL, PostgreSQL, SQLite, Oracle, and Microsoft SQL Server. This cross-compatibility allows developers to build applications that can easily switch between different databases with minimal code changes. Such flexibility is invaluable in projects where database technology might evolve over time.

SQLAlchemy introduces the concept of a Session, which acts as a transactional workspace for interacting with the database. Sessions provide a layer of abstraction for managing database connections and ensure transactional integrity. This means that any changes made during a session can be committed to the database or rolled back in case of errors. This transactional scope is essential for maintaining data consistency and reliability in complex applications.

For managing relationships between tables, SQLAlchemy simplifies the process with its ORM. It supports defining relationships such as one-to-one, one-to-many, and many-to-many directly in Python classes. This feature streamlines querying and joining tables, even in scenarios involving intricate relationships. For instance, retrieving all products added by a specific farmer becomes a straightforward task when relationships are defined properly.

In the context of your project, SQLAlchemy plays a critical role in managing the core functionalities of the application. Tables like `User`, `Register`, `Addagroproducts`, and `Farming` are represented as Python classes, making it easy to perform CRUD operations programmatically. With SQLAlchemy's ORM, adding a new farmer, updating product details, or fetching all orders becomes a clean and efficient process. Furthermore, its session management ensures transactional integrity, and its support for relationships simplifies querying interconnected data, such as mapping products to their respective farmers.

By leveraging SQLAlchemy, your project benefits from a structured, scalable, and maintainable approach to database interaction.

5.1.4 PYTHON FLASK

Flask is a lightweight and versatile web framework for Python, designed to make web application development simple and intuitive. It is often described as a "micro-framework" because it keeps the core framework minimal while allowing developers to extend its

functionality with additional libraries and modules as needed. Despite its simplicity, Flask is robust enough to power complex web applications and APIs, making it a popular choice for both beginners and experienced developers.

One of Flask's defining characteristics is its minimalist nature. Unlike monolithic frameworks, Flask doesn't impose strict rules or conventions, giving developers the freedom to structure their applications according to their preferences. This flexibility makes it ideal for building small projects, prototypes, and even larger applications where custom architecture is preferred. The framework provides only the essential tools, such as routing, request handling, and templating, while leaving advanced features like database integration, authentication, or form validation to third-party extensions.

The routing system in Flask is simple yet powerful. It allows developers to define URL endpoints and associate them with Python functions. Each function acts as a "view," returning responses to incoming HTTP requests. This clean mapping between URLs and functions makes Flask highly readable and easy to manage. For instance, a simple route to display a homepage can be created with just a few lines of code.

Flask also excels in handling HTML templates through its integration with the Jinja2 templating engine. This allows developers to create dynamic web pages by embedding Python code within HTML files. Templating makes it easy to pass data from the backend to the frontend and render it dynamically. Features like template inheritance further streamline the process of building complex web layouts by allowing developers to reuse common elements like headers, footers, and navigation bars.

Another strength of Flask is its extensive ecosystem of extensions. While Flask itself is lightweight, developers can enhance its capabilities by integrating third-party libraries. For instance, Flask-SQLAlchemy provides ORM support for database interactions, Flask-WTF simplifies form handling and validation, and Flask-Login adds authentication functionality. This modularity ensures that developers only include the features they need, keeping the application efficient and uncluttered.

Flask also supports RESTful API development, making it an excellent choice for building APIs. Its simplicity in handling HTTP methods like GET, POST, PUT, and DELETE enables developers to create APIs for web and mobile applications with ease. The ability to seamlessly integrate with JSON ensures smooth data exchange between the server and client, a crucial feature in modern web development.

One of the reasons Flask is favored in the development community is its scalability. Although it is lightweight, Flask can handle the demands of growing applications by incorporating extensions or integrating with production-ready tools. For example, integrating Flask with WSGI servers like Gunicorn or deploying it on platforms like Docker and Kubernetes makes it capable of managing high traffic and complex deployments.

In your Farm Management System Project, Flask serves as the backbone for the backend operations. It handles the routing and request processing, allowing users to interact with features such as farmer registration, product management, and online transactions. Through its integration with SQLAlchemy, Flask efficiently manages database operations, ensuring data consistency and smooth functionality. Its simplicity in managing routes and templates allows the creation of a user-friendly interface, while its extensibility ensures that advanced features like authentication, email communication, and logging are seamlessly implemented.

Overall, Flask combines simplicity, flexibility, and power, making it a reliable choice for developers building web applications or APIs. Its ease of use and vibrant community support ensure that Flask remains a go-to framework for Python developers worldwide.

5.2 HARDWARE REQUIREMENTS

- Computer with a 1.1 GHz or faster processor
- Minimum 2GB of RAM or more
- 2.5 GB of available hard-disk space
- 5400 RPM hard drive
- 1366 × 768 or higher-resolution display
- DVD-ROM drive

CHAPTER 6

METHODOLOGY

The methodology for the Farm Management System Project revolves around utilizing modern web development techniques, agile development practices, and appropriate technologies to create a robust, scalable, and user-friendly application for farmers and buyers. The project follows a structured approach that encompasses design, development, testing, and deployment phases, ensuring the creation of a fully functional farm management system.

6.1 Requirement Gathering and Analysis

The first step of the project is the collection and analysis of the requirements. A clear understanding of user needs is crucial to developing a system that meets the expectations of both farmers and buyers. The system is designed to help farmers register, manage their products, and sell them online, while buyers can browse agricultural products, make purchases, and submit quality-check requests. Based on the requirements, the system will also handle farmer profiles, farming types, and product management. Functional requirements such as product catalog, order management, and user authentication are carefully outlined, and technical requirements for backend, frontend, and database interactions are established.

6.2 System Design

Once the requirements are defined, the design phase begins, focusing on the architecture of the system. The application uses a client-server model, where the frontend interacts with the backend through HTTP requests. The backend is built using Python Flask with SQLAlchemy for database management, while the frontend is developed using HTML, CSS, JavaScript, and Bootstrap for responsiveness. The design emphasizes user-centric layouts, ensuring that both farmers and buyers can easily navigate the platform. The database schema is designed to handle multiple entities like user profiles, products, orders, and farming types efficiently.

6.3 Frontend and Backend Development

The next step in the methodology is the development of the frontend and backend components in parallel. On the frontend, pages are created for farmer registration, product

catalog, order management, and user login. The interface is designed to be intuitive and mobile-responsive, allowing farmers to manage their profiles, products, and orders seamlessly. HTML forms are used for submitting data like product information, while JavaScript is used for dynamic interactions, such as displaying error messages or updating the user interface based on backend responses.

On the backend, Python Flask is used to handle routing, request processing, and user authentication. The system uses Flask-SQLAlchemy for database interactions, which helps store and retrieve information such as farmer details, product listings, and orders. CRUD (Create, Read, Update, Delete) operations are implemented to manage farming details, product information, and buyer requests. Flask's routing system allows clear mapping of URLs to functions, ensuring that users are directed to the right page or feature based on their actions.

6.4 Database Design and Integration

A significant part of the methodology is database design and integration. The system uses SQLAlchemy, an Object Relational Mapping (ORM) library, to manage interactions with the MySQL database. SQLAlchemy abstracts database queries and simplifies operations like inserting, updating, and deleting records. Entities like User, Farmer, Product, Order, and Farming Type are modeled as classes, allowing seamless data retrieval and storage. Data consistency and integrity are ensured through relational tables and proper foreign key relationships between entities, ensuring that users, products, and orders are correctly associated with each other.

6.5 Security and Authentication

Security is a critical aspect of the system, especially considering that it handles sensitive user information such as farmer profiles and financial transactions. Flask-Login is used to implement user authentication and session management. Secure login and registration features are developed to ensure only authenticated users can access certain functionalities, such as adding products or viewing detailed order histories. Passwords are hashed using the Werkzeug security library to ensure that user credentials are stored securely in the database. Additionally, input validation techniques are used to prevent malicious attacks like SQL injection or cross-site scripting (XSS).

6.6 Testing and Quality Assurance

Testing is an integral part of the methodology to ensure the system works as expected and meets the requirements. Unit testing, integration testing, and functional testing are conducted throughout the development process. The backend is thoroughly tested for data integrity and correct API responses, while the frontend is tested for usability, performance, and responsiveness. Testing is done using PyTest for Python code and manual testing for frontend interactions. After fixing bugs and resolving issues, the system undergoes user acceptance testing (UAT), where real users test the application in a simulated environment to ensure it aligns with their needs and expectations.

6.7 Deployment and Maintenance

After rigorous testing, the system is deployed to a live environment using cloud platforms like Heroku, AWS, or Google Cloud, where it can handle live traffic and user requests. Deployment involves setting up the application on the server, ensuring proper configurations for the database and backend, and configuring domain names for the website. The system is monitored post-deployment to ensure smooth functioning, and regular updates and maintenance are performed to address new features or bugs.

6.8 Agile Development Approach

The project follows an agile development approach, which is characterized by iterative development cycles (sprints). At the start of each sprint, specific goals are set, and at the end of each sprint, the project's progress is evaluated. This allows flexibility in adapting to changes, quickly addressing issues, and continuously improving the product based on user feedback. The agile methodology ensures that the Farm Management System evolves over time, incorporating user suggestions and improving performance.

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

7.1 CONCLUSION

FARM MANAGEMENT SYSTEM successfully implemented based on online selling which helps us in administrating the agro products user for managing the tasks performed in farmers. The project successfully used various functionalities of Xampp and python flask and also create the fully functional database management system for online portals.

Using MySQL as the database is highly beneficial as it is free to download, popular and can be easily customized. The data stored in the MySQL database can easily be retrieved and manipulated according to the requirements with basic knowledge of SQL.

With the theoretical inclination of our syllabus, it becomes very essential to take the at most advantage of any opportunity of gaining practical experience that comes along. The building blocks of this Major Project “Farm Management System” was one of these opportunities. It gave us the requisite practical knowledge to supplement the already taught theoretical concepts thus making us more competent as a computer engineer. The project from a personal point of view also helped us in understanding the following aspects of project development:

- The planning that goes into implementing a project.
- The importance of proper planning and an organized methodology.
- The key element of team spirit and co-ordination in a successful project.

7.2 FUTURE ENHANCEMENT

The Farm Management System Project has significant potential for future enhancement. One major improvement could be the development of a mobile application, providing farmers and buyers with an easier, more accessible way to interact with the system. This app could include features like notifications for new orders, GPS integration for deliveries, and a user-friendly interface for both Android and iOS. Another promising addition would be the integration of Internet of Things (IoT) sensors on farms, allowing for real-time data on environmental factors like soil moisture and temperature. This would help farmers make data-driven decisions for better crop management.

Further, incorporating machine learning for crop prediction and yield estimation would provide farmers with valuable insights into the best crops to grow and help predict yields based on historical and environmental data. Another possible enhancement is automated financial and inventory management, which could allow farmers to track expenses, revenue, and product inventory seamlessly. Additionally, the system could be upgraded to offer advanced analytics and reporting, helping farmers and administrators to generate detailed reports on sales trends, crop performance, and market demands.

User personalization could also be enhanced, where the system suggests relevant farming products or services based on individual preferences and behaviors. To expand its reach, the platform could offer multilingual support, catering to farmers in different regions and making the system accessible in various languages. Introducing blockchain technology could also improve the transparency and security of transactions, ensuring that every transaction is recorded in a decentralized ledger, reducing fraud.

Moreover, the system could promote sustainable farming practices, offering eco-friendly solutions and organic product listings to attract environmentally conscious buyers. Introducing subscription-based models would provide premium features to farmers, enhancing user engagement while creating a new revenue stream. Lastly, integrating the platform with popular e-commerce platforms like Amazon or eBay could help farmers reach a broader audience and increase sales. These enhancements will elevate the platform, making it more robust, accessible, and aligned with modern farming needs.

APPENDIX – 1

SOURCE CODE

main.py

```
from flask import Flask,render_template,request,session,redirect,url_for,flash
from flask_sqlalchemy import SQLAlchemy
from flask_login import UserMixin
from werkzeug.security import generate_password_hash,check_password_hash
from flask_login import login_user,logout_user,login_manager,LoginManager
from flask_login import login_required,current_user

# MY db connection
local_server= True
app = Flask(__name__)
app.secret_key='harshithbhaskar'
# this is for getting unique user access
login_manager=LoginManager(app)
login_manager.login_view='login'
@login_manager.user_loader
def load_user(user_id):
    return User.query.get(int(user_id))
#
app.config['SQLALCHEMY_DATABASE_URL']='mysql://username:password@localhost/
database_name'
app.config['SQLALCHEMY_DATABASE_URI']='mysql://root:@localhost/farmers'
db=SQLAlchemy(app)

# here we will create db models that is tables
class Test(db.Model):
    id=db.Column(db.Integer,primary_key=True)
    name=db.Column(db.String(100))

class Farming(db.Model):
    fid=db.Column(db.Integer,primary_key=True)
    farmingtype=db.Column(db.String(100))
```

```

class Addagroproducts(db.Model):
    username=db.Column(db.String(50))
    email=db.Column(db.String(50))
    pid=db.Column(db.Integer,primary_key=True)
    productname=db.Column(db.String(100))
    productdesc=db.Column(db.String(300))
    price=db.Column(db.Integer)

```

```

class Trig(db.Model):
    id=db.Column(db.Integer,primary_key=True)
    fid=db.Column(db.String(100))
    action=db.Column(db.String(100))
    timestamp=db.Column(db.String(100))

```

```

class User(UserMixin,db.Model):
    id=db.Column(db.Integer,primary_key=True)
    username=db.Column(db.String(50))
    email=db.Column(db.String(50),unique=True)
    password=db.Column(db.String(1000))

```

```

class Register(db.Model):
    rid=db.Column(db.Integer,primary_key=True)
    farmername=db.Column(db.String(50))
    adharnumber=db.Column(db.String(50))
    age=db.Column(db.Integer)
    gender=db.Column(db.String(50))
    phonenumber=db.Column(db.String(50))
    address=db.Column(db.String(50))
    farming=db.Column(db.String(50))

```

```

@app.route('/')
def index():
    return render_template('index.html')

```

```

@app.route('/farmerdetails')
@login_required
def farmerdetails():
    # query=db.engine.execute(f"SELECT FROM `register`")
    query=Register.query.all()
    return render_template('farmerdetails.html',query=query)

@app.route('/agroproducts')
def agroproducts():
    # query=db.engine.execute(f"SELECT FROM `addagroproducts`")
    query=Addagroproducts.query.all()
    return render_template('agroproducts.html',query=query)
@app.route('/addagroproduct',methods=['POST','GET'])
@login_required
def addagroproduct():
    if request.method=="POST":
        username=request.form.get('username')
        email=request.form.get('email')
        productname=request.form.get('productname')
        productdesc=request.form.get('productdesc')
        price=request.form.get('price')

products=Addagroproducts(username=username,email=email,productname=productname,productdesc=productdesc,price=price)
        db.session.add(products)
        db.session.commit()
        flash("Product Added","info")
        return redirect('/agroproducts')

    return render_template('addagroproducts.html')

@app.route('/triggers')
@login_required
def triggers():

```

```

# query=db.engine.execute(f"SELECT FROM `trig`")
query=Trig.query.all()
return render_template('triggers.html',query=query)

@app.route('/addfarming',methods=['POST','GET'])
@login_required
def addfarming():
    if request.method=="POST":
        farmingtype=request.form.get('farming')
        query=Farming.query.filter_by(farmingtype=farmingtype).first()
        if query:
            flash("Farming Type Already Exist","warning")
            return redirect('/addfarming')
        dep=Farming(farmingtype=farmingtype)
        db.session.add(dep)
        db.session.commit()
        flash("Farming Addes","success")
        return render_template('farming.html')
@app.route("/delete/<string:rid>",methods=['POST','GET'])
@login_required
def delete(rid):
    # db.engine.execute(f"DELETE FROM `register` WHERE `register`.`rid`={rid}")
    post=Register.query.filter_by(rid=rid).first()
    db.session.delete(post)
    db.session.commit()
    flash("Slot Deleted Successful","warning")
    return redirect('/farmerdetails')
@app.route("/edit/<string:rid>",methods=['POST','GET'])
@login_required
def edit(rid):
    # farming=db.engine.execute("SELECT FROM `farming`")
    if request.method=="POST":
        farmername=request.form.get('farmername')
        adharnumber=request.form.get('adharnumber')

```

```

age=request.form.get('age')
gender=request.form.get('gender')
phonenummer=request.form.get('phonenummer')
address=request.form.get('address')
farmingtype=request.form.get('farmingtype')
#          query=db.engine.execute(f"UPDATE          `register`          SET
`farmername`='{farmername}',`adharnumber`='{adharnumber}',`age`='{age}',`gender`='{gender}',`phonenummer`='{phonenummer}',`address`='{address}',`farming`='{farmingtype}'")
post=Register.query.filter_by(rid=rid).first()
print(post.farmername)
post.farmername=farmername
post.adharnumber=adharnumber
post.age=age
post.gender=gender
post.phonenummer=phonenummer
post.address=address
post.farming=farmingtype
db.session.commit()
flash("Slot is Updates","success")
return redirect('/farmerdetails')
posts=Register.query.filter_by(rid=rid).first()
farming=Farming.query.all()
return render_template('edit.html',posts=posts,farming=farming)

```

```

@app.route('/signup',methods=['POST','GET'])
def signup():
    if request.method == "POST":
        username=request.form.get('username')
        email=request.form.get('email')
        password=request.form.get('password')
        print(username,email,password)
        user=User.query.filter_by(email=email).first()
        if user:
            flash("Email Already Exist","warning")

```



```

        return render_template('/signup.html')
    # encpassword=generate_password_hash(password)

    # new_user=db.engine.execute(f"INSERT INTO `user` (`username`,`email`,`password`)
VALUES ({username}},{email},{encpassword})")

    # this is method 2 to save data in db
    newuser=User(username=username,email=email,password=password)
    db.session.add(newuser)
    db.session.commit()
    flash("Signup Succes Please Login","success")
    return render_template('login.html')
    return render_template('signup.html')

@app.route('/login',methods=['POST','GET'])
def login():
    if request.method == "POST":
        email=request.form.get('email')
        password=request.form.get('password')
        user=User.query.filter_by(email=email).first()
        if user and user.password == password:
            login_user(user)
            flash("Login Success","primary")
            return redirect(url_for('index'))
        else:
            flash("invalid credentials","warning")
            return render_template('login.html')

    return render_template('login.html')

@app.route('/logout')
@login_required
def logout():
    logout_user()

```

```

flash("Logout SuccessFul","warning")
return redirect(url_for('login'))

@app.route('/register',methods=['POST','GET'])
@login_required
def register():
    farming=Farming.query.all()
    if request.method=="POST":
        farmername=request.form.get('farmername')
        adharnumber=request.form.get('adharnumber')
        age=request.form.get('age')
        gender=request.form.get('gender')
        phonenumber=request.form.get('phonenumber')
        address=request.form.get('address')
        farmingtype=request.form.get('farmingtype')
    query=Register(farmername=farmername,adharnumber=adharnumber,age=age,gender=gender,phonenumber=phonenumber,address=address,farming=farmingtype)
    db.session.add(query)
    db.session.commit()
    # query=db.engine.execute(f"INSERT INTO `register`
    (`farmername`,`adharnumber`,`age`,`gender`,`phonenumber`,`address`,`farming`) VALUES
    ('{farmername}','{adharnumber}','{age}','{gender}','{phonenumber}','{address}','{farmingtype}')")
    # flash("Your Record Has Been Saved","success")
    return redirect('/farmerdetails')
    return render_template('farmer.html',farming=farming)
@app.route('/test')
def test():
    try:
        Test.query.all()
        return 'My database is Connected'
    except:
        return 'My db is not Connected'
app.run(debug=True)

```

APPENDIX – 2

SCREENSHOTS

Sample Output

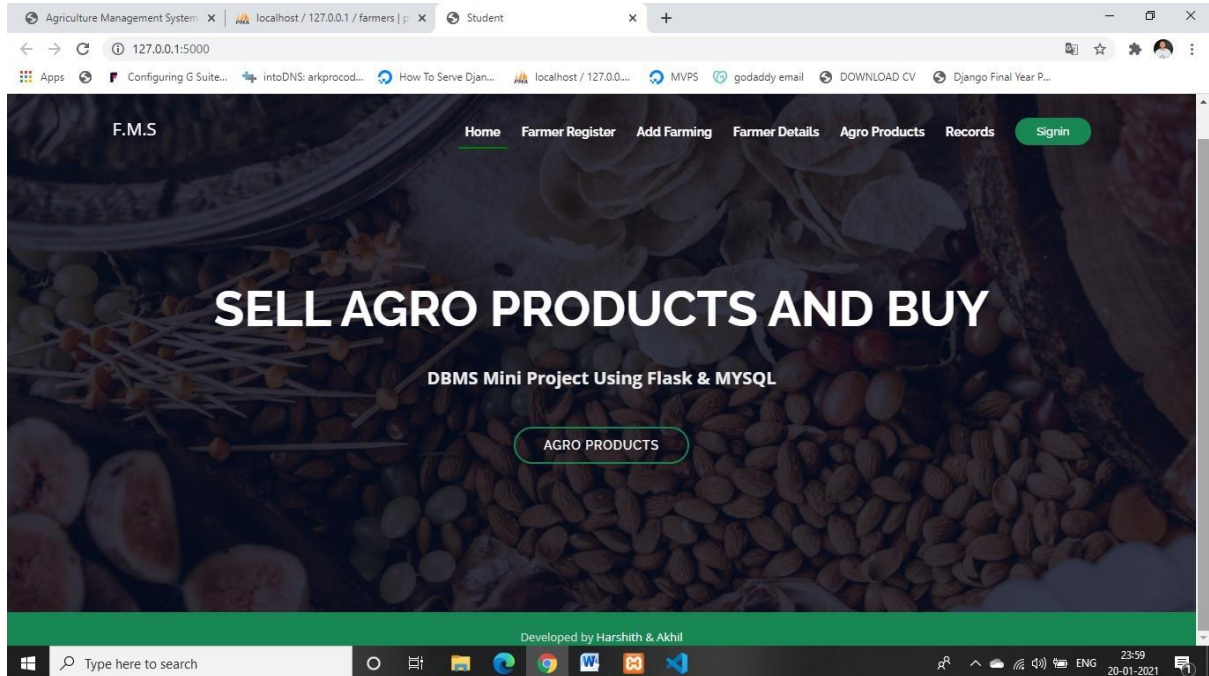


Figure 7.1: Home page

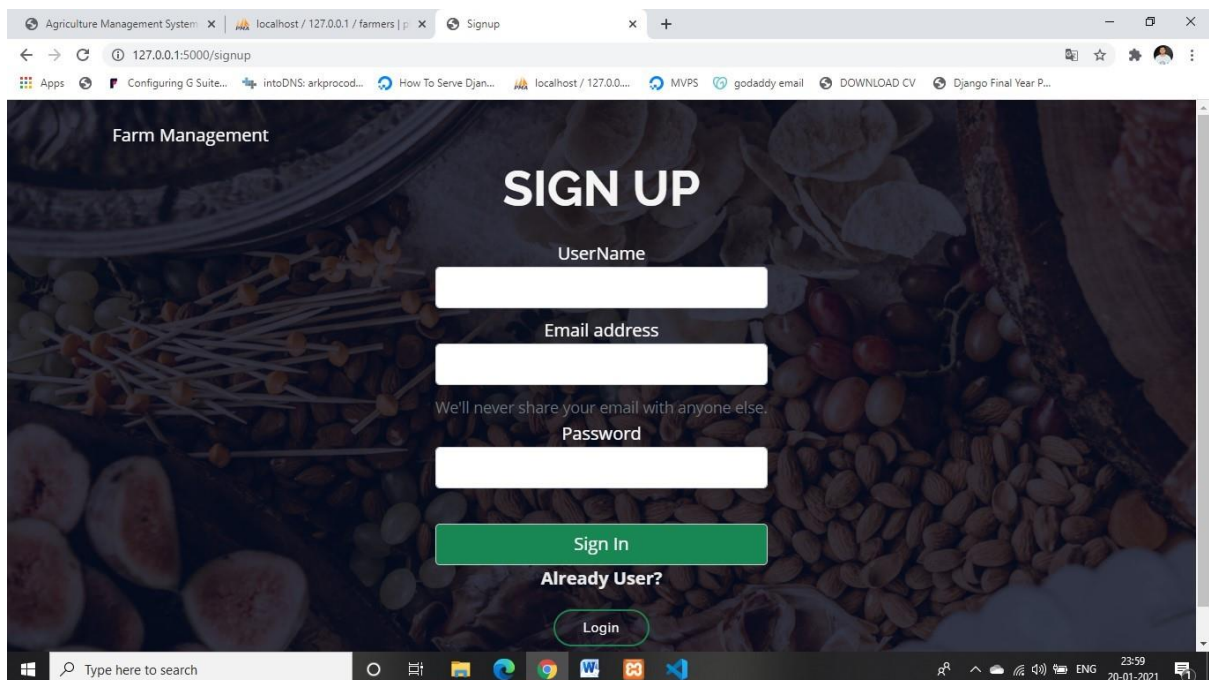


Figure 7.2: Sign up page

Agriculture Management System | localhost / 127.0.0.1 / farmers | Register Farmers Details

127.0.0.1:5000/register

F.M.S Home Farmer Register Add Farming Farmer Details Agro Products Records Welcome test Logout

Register Farmers Details

Farmer Name

Adhar Number

Age

Select Gender

Phone Number

Figure 7.3: Registration page

Agriculture Management System | localhost / 127.0.0.1 / farmers | Agro Products

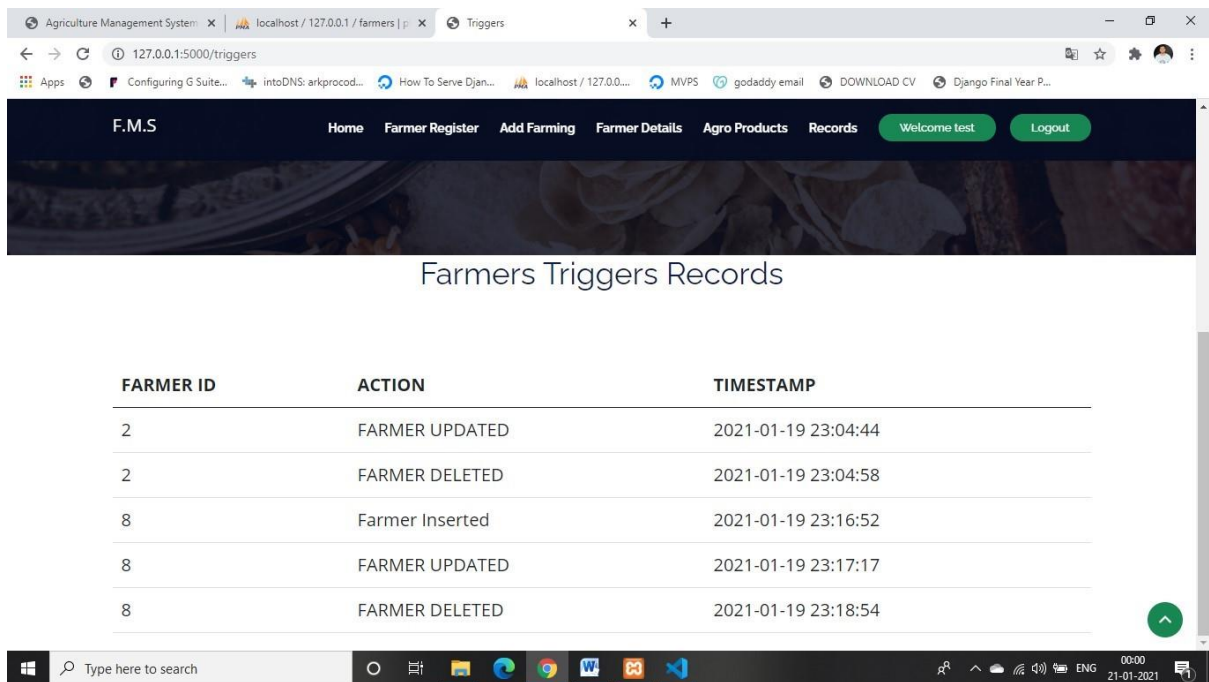
127.0.0.1:5000/agroproducts

F.M.S Home Farmer Register Add Farming Farmer Details Agro Products Records Welcome test Logout

Agro Products

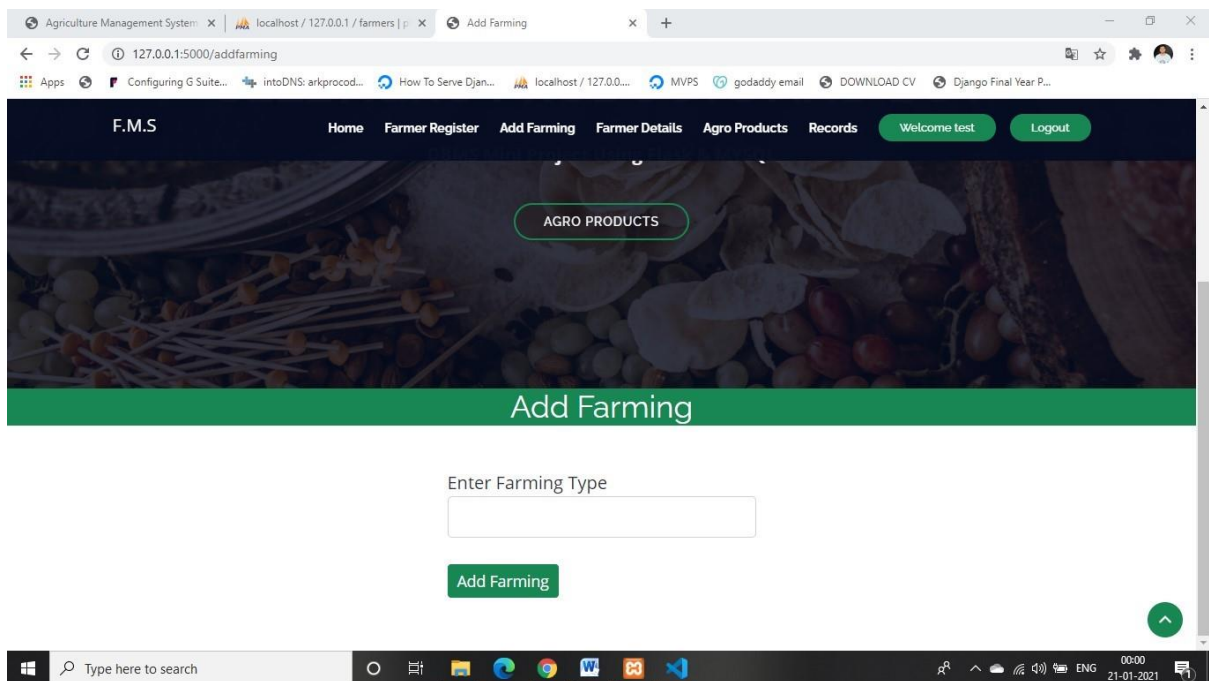
<p>GIRIJA CAULIFLOWER</p> <p>Price : 520</p> <p>Tips for Growing Cauliflower. Well drained medium loam and or sandy loam soils are suitable.</p> <p>Owner : test</p> <p>Email : test@gmail.com</p> <p>Purchase</p>	<p>COTTON</p> <p>Price : 563</p> <p>Cotton is a soft, fluffy staple fiber that grows in a boll, around the seeds of the cotton</p> <p>Owner : test</p> <p>Email : test@gmail.com</p> <p>Purchase</p>	<p>silk</p> <p>Price : 582</p> <p>silk is best business developed from cocoon for saries preparation and so on</p> <p>Owner : arkpro</p> <p>Email : arkpro@gmail.com</p> <p>Purchase</p>
---	---	---

Figure 7.4: Product view



FARMER ID	ACTION	TIMESTAMP
2	FARMER UPDATED	2021-01-19 23:04:44
2	FARMER DELETED	2021-01-19 23:04:58
8	Farmer Inserted	2021-01-19 23:16:52
8	FARMER UPDATED	2021-01-19 23:17:17
8	FARMER DELETED	2021-01-19 23:18:54

Figure 7.5: Trigger records



Enter Farming Type

Add Farming

Figure 7.6: Farming type

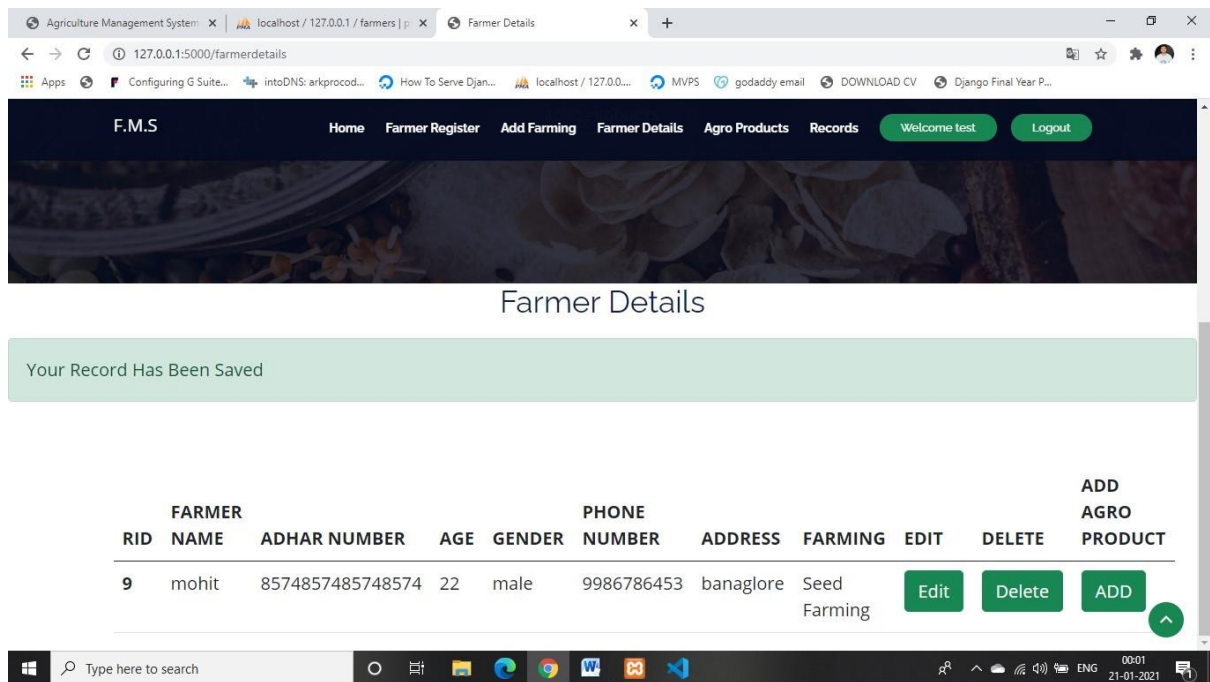


Figure 7.7: Farmer details

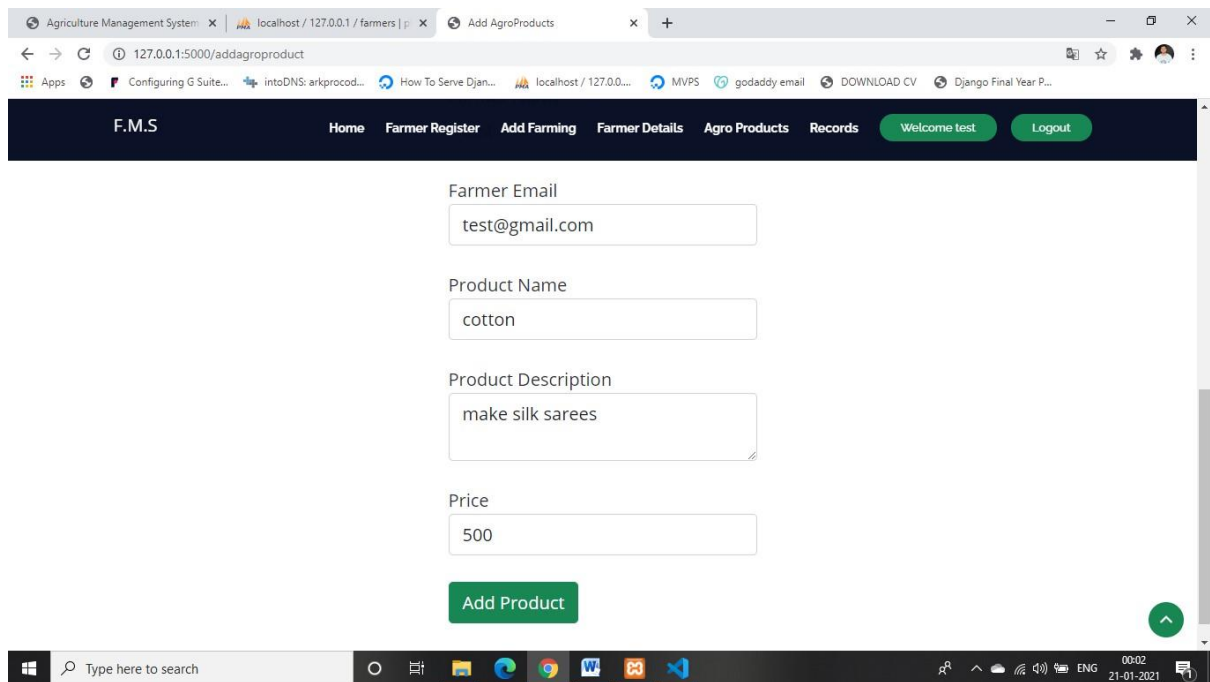


Figure 7.8: Add product

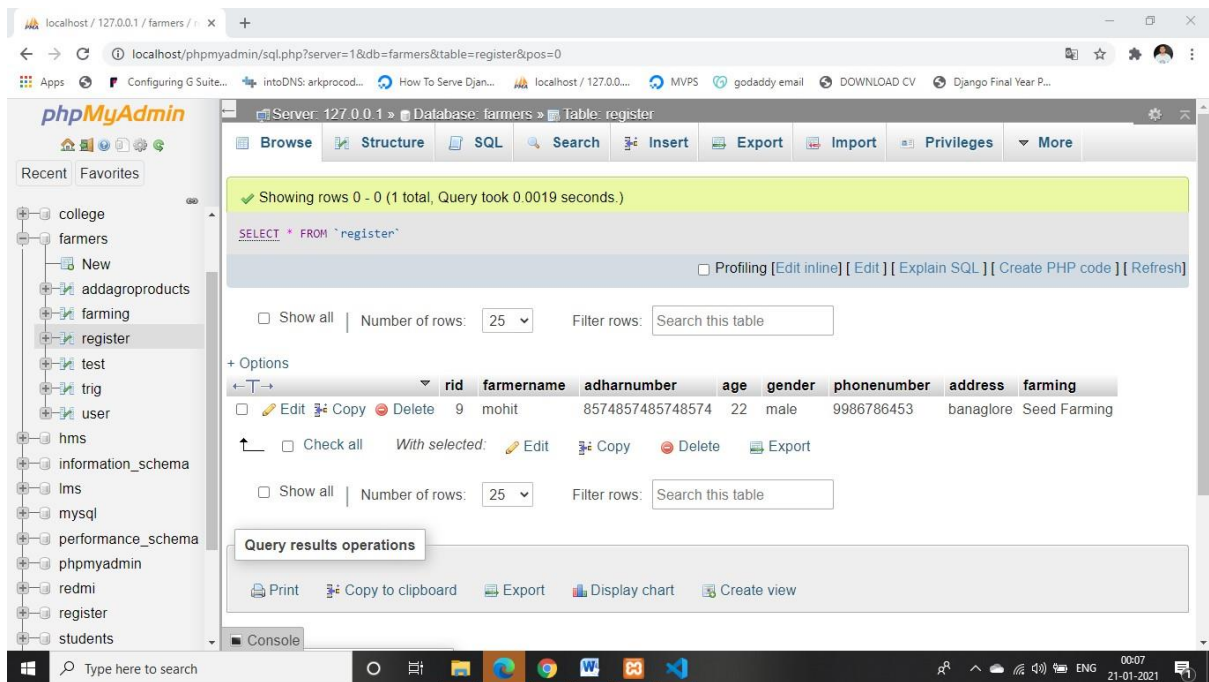


Figure 7.9: Database page

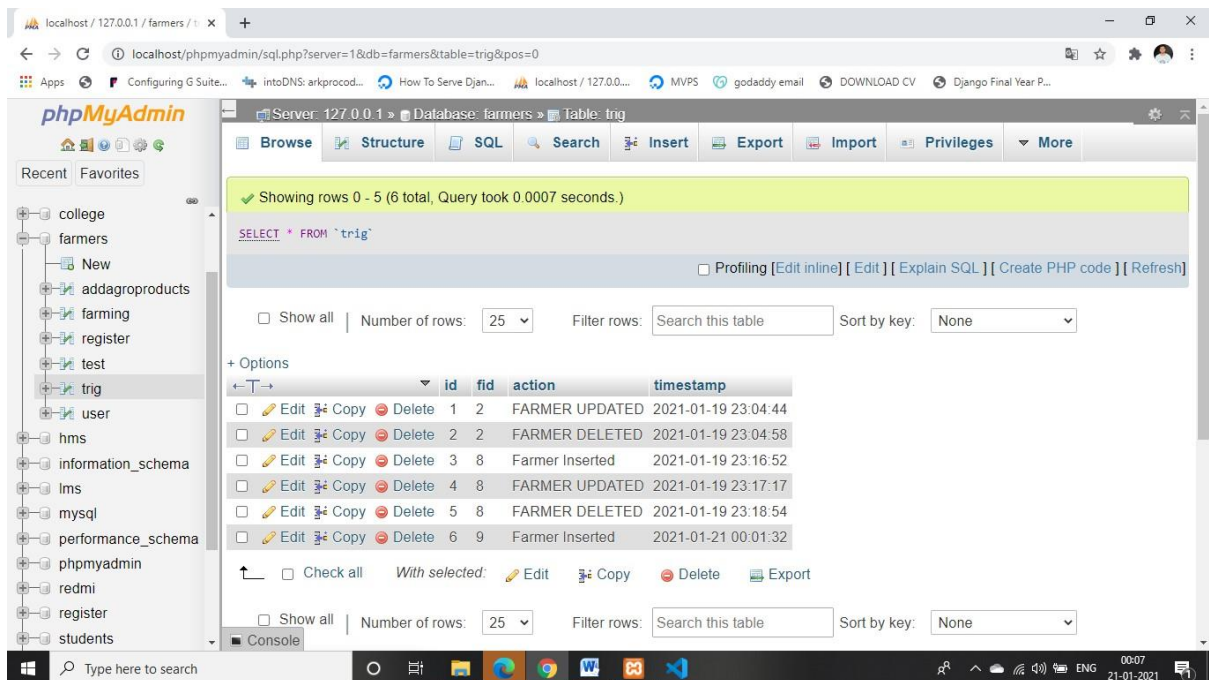


Figure 7.10: Trigger records in phpmyadmin

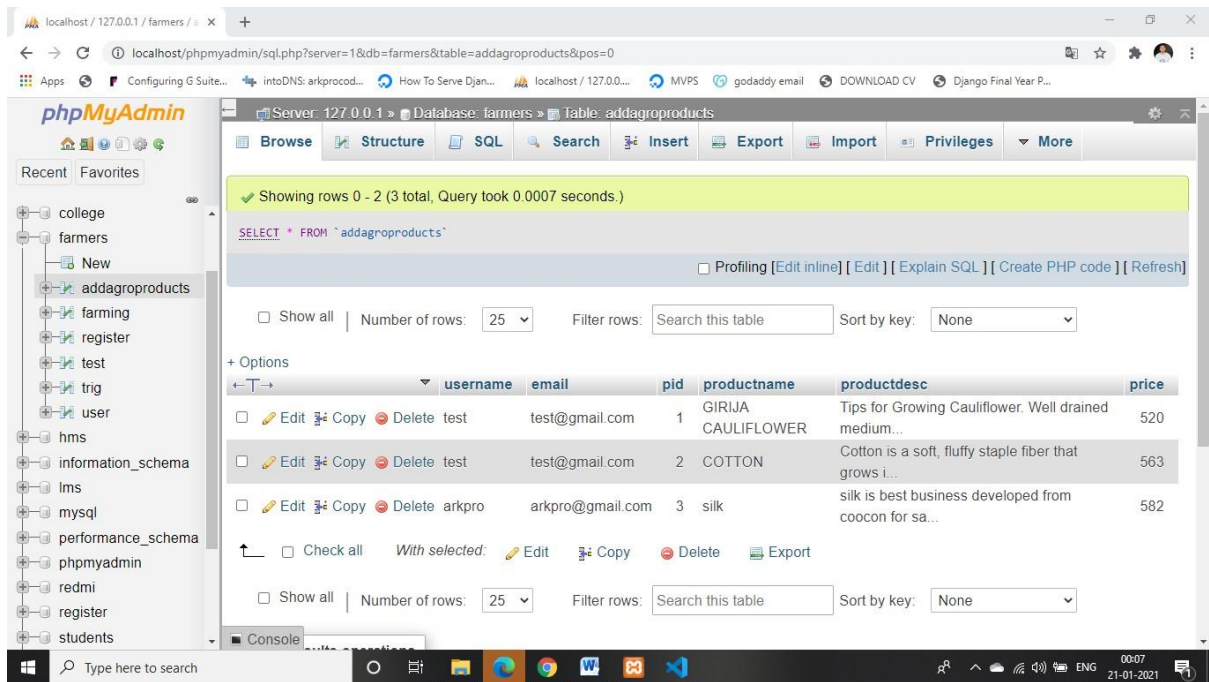


Figure 7.11: Agro product records in phpmyadmin

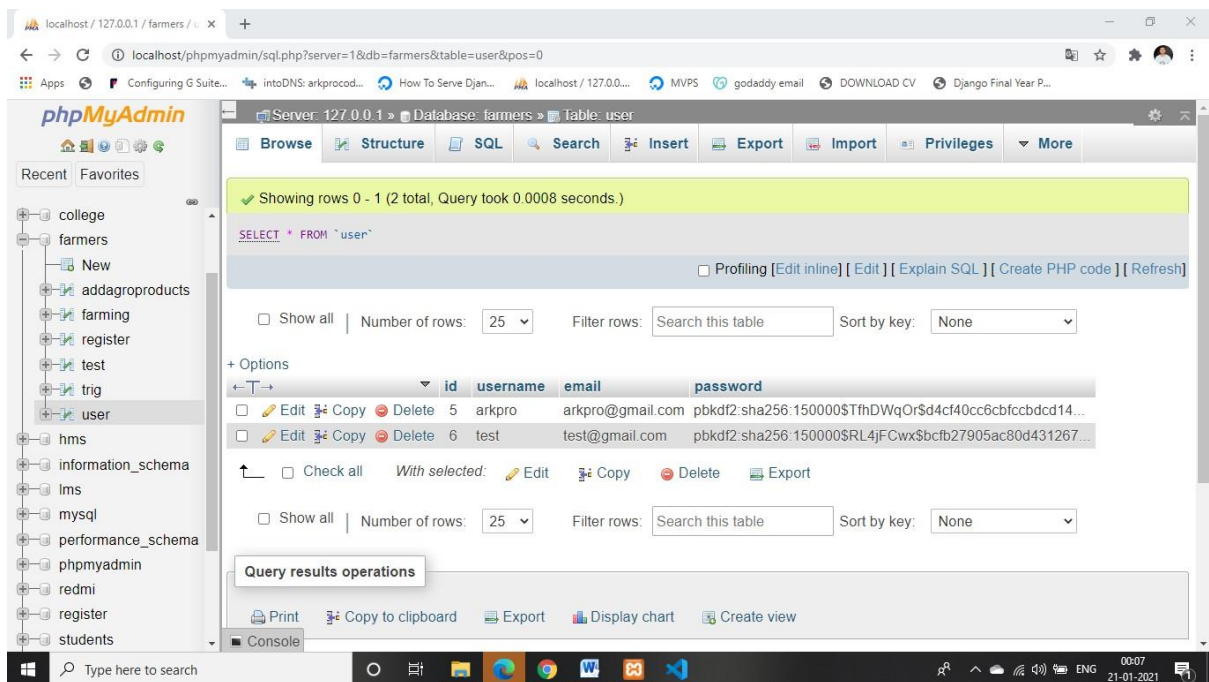


Figure 7.12: User records in phpmyadmin

REFERENCES

- Y. Li, J. Huang, F. Tian, H.-A. Wang, and G.-Z. Dai, “Farm management systems for small-scale farmers: A review of technologies and solutions,” *Agricultural Systems*, vol. 168, pp. 150–164, Jun. 2019.
- R. L. Kelly, A. C. Peters, and M. L. White, “Building an agricultural information system using web-based technology,” *International Journal of Agricultural Management*, vol. 22, no. 4, pp. 230-245, Aug. 2021.
- P. J. Lemaire and S. D. Smith, “E-commerce and digital platforms in agriculture: Impacts on the supply chain and productivity,” *Journal of Digital Agriculture*, vol. 12, no. 3, pp. 116–130, 2020.
- H. S. Murthy, K. B. Patel, and V. R. Yadav, “Design and implementation of farm management software for smallholder farmers,” *Computers and Electronics in Agriculture*, vol. 78, no. 2, pp. 109–119, Nov. 2018.
- J. G. Lewis, “Database management in agricultural management systems,” *Database Systems for Agricultural Applications*, vol. 45, no. 1, pp. 35-42, 2017.
- M. M. Holmes, P. B. Jackson, and K. M. Mendez, “Improving farm-to-market information systems with open-source solutions,” *International Journal of Agricultural Information Systems*, vol. 24, no. 2, pp. 99-106, 2021.
- C. L. Foster, A. J. Morton, and J. B. Williams, “IoT and sensor technology in precision agriculture: A review of applications,” *Sensors and Actuators B: Chemical*, vol. 263, pp. 289–296, Dec. 2018.
- R. M. Stone, “Web-based farm management solutions for optimizing agricultural productivity,” *Journal of Agricultural Informatics*, vol. 8, no. 1, pp. 27–36, Jan. 2020.