

A PROJECT REPORT

On

WEB-BASED FARMING ASSISTANCE SERVICES

Submitted in the Partial Fulfilment of the Requirement for the Award of

BACHELOR OF TECHNOLOGY

In

INFORMATION TECHNOLOGY

(2024)

By

Shruti Saumya (2001220130113)
Vrishi Raj Kesarwani (2001220130130)

Under the Guidance

Of

Dr Vibha Srivastava



**SHRI RAMSWAROOP MEMORIAL COLLEGE OF
ENGINEERING & MANAGEMENT, LUCKNOW**

Affiliated to

**Dr. APJ ABDUL KALAM TECHNICAL UNIVERSITY,
UTTAR PRADESH, LUCKNOW**



INFORMATION TECHNOLOGY
SHRI RAMSWAROOP MEMORIAL COLLEGE OF
ENGINEERING AND MANAGEMENT

CERTIFICATE

Certified that the project entitled “**Web-Based Farming Assistance Services**” submitted by **Shruti Saumya [2001220130113]** and **Vrishi Raj Kesarwani [2001220130130]** in the partial fulfilment of the requirements for the award of degree of Bachelor of Technology (Information Technology) of Dr APJ Abdul Kalam Technical University (Uttar Pradesh, Lucknow), is a record of student’s own work carried under our supervision and guidance. The project report embodies results of original work and studies carried out by students and the contents do not form the basis for the award of any other degree to the candidate or to anybody else.

Signature

Dr. Vibha Srivastava

Assistant Professor

(Project Guide)

Signature

Prof. Ajay KR. Srivastava

Professor

(Head Of Department)



INFORMATION TECHNOLOGY
SHRI RAMSWAROOP MEMORIAL COLLEGE OF
ENGINEERING AND MANAGEMENT

DECLARATION

We hereby declare that the project entitled “**Web-Based Farming Assistance Services**” submitted by us in the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology (Information Technology) of Dr. APJ Abdul Kalam Technical University (Uttar Pradesh, Lucknow), is record of our own work carried under the supervision and guidance of **Dr. Vibha Srivastava**, Assistant Professor.

To the best of our knowledge this project has not been submitted to Dr. APJ Abdul Kalam Technical University (Uttar Pradesh, Lucknow) or any other University or Institute for the award of any degree.

Shruti Saumya

(2001220130113)

Vrishi Raj Kesarwani

(2001220130130)

ACKNOWLEDGEMENT

We extend our deepest gratitude to **Dr. Vibha Srivastava**, our mentor, for her invaluable guidance, supervision, and continuous support throughout the development of this project. Her profound expertise and insightful feedback have been fundamental in steering this project towards its successful completion. Dr Vibha Srivastava's encouragement and dedication not only enriched our learning experience but also greatly enhanced our research capabilities. We would also like to acknowledge our heartiest thanks to our H.O.D. Prof. A.K. Srivastava (Head of Department- IT Department), for his support and continued guidance.

We are also immensely grateful to the **Information Technology Department at SRMCEM, Lucknow**, for providing us with the necessary resources and a conducive environment that enabled us to carry out our research effectively. The support from the department faculty and staff has been pivotal in overcoming various challenges during our project. Their readiness to assist at every step fostered an atmosphere of enthusiasm and innovation.

Additionally, we would like to express our appreciation to our peers and family members, whose constant support and belief in our abilities have been a source of motivation and strength. Their contributions, though not as visible, have played a significant role in our journey. We are also thankful to all those who contributed in any capacity to this project, your insights and suggestions have indeed made a valuable impact. This journey has not only been a learning experience but also an opportunity to grow and we are thankful to everyone involved for being a part of it.

Shruti Saumya
(2001220130113)

Vrishi Raj Kesarwani
(2001220130130)

PREFACE

Agriculture, the backbone of Indian society, has sustained the livelihoods of millions for generations, embodying not just an economic sector but a way of life deeply ingrained in tradition and culture. Yet, despite its profound importance, Indian agriculture faces formidable challenges, chief among them being the pervasive influence of middlemen in the agricultural supply chain. These intermediaries exploit farmers, perpetuating cycles of poverty and hindering the sector's growth and sustainability.

In this era of rapid technological advancement and societal transformation, there arises a unique opportunity to harness technology's power to revolutionize Indian agriculture. The Farming Assistance Web Application emerges as a beacon of hope, a digital solution designed to address the multifaceted needs of farmers across the nation. Its overarching goal is to bridge the gap between farmers and consumers, ensuring fair compensation for farmers while enhancing market transparency and fostering sustainable livelihoods.

This preface sets the stage for exploring the challenges plaguing Indian agriculture, the potential offered by technology, and the transformative impact envisioned through the Farming Assistance Web Application. It signifies a paradigm shift in how agriculture is practiced, offering a comprehensive suite of features and capabilities to empower farmers in navigating the complexities of modern agriculture with confidence and resilience. As India stands at the cusp of technological advancement and agricultural transformation, the Farming Assistance Web Application represents a pivotal innovation poised to shape the future of Indian agriculture for generations to come.

This preface aims to set the stage for the detailed project report that follows, outlining the structure of our documentation. The project is divided into several key chapters, each dedicated to a specific aspect of the project:

Chapter 1: Introduction

This chapter provides an overview of the project, including its background, objectives, and significance.

Chapter 2: Literature Review

Here, we delve into previous studies, technologies, and platforms related to the agricultural sector, role of middlemen, drawing parallels and distinctions with our work.

Chapter 3: Proposed Methodology

This section details the methodologies employed in the development of Web-Based Farming Assistance Services, from initial design to final implementation.

Chapter 4: Results

We present the outcomes of our project, showcasing the functionality and features of the Farming Services.

Chapter 5: Conclusion

This chapter reflects on the project's achievements, challenges encountered, and the learning outcomes.

Chapter 6: Future Scope

Looking ahead, we discuss potential enhancements and expansions for Farming Assistance Services, considering emerging technologies and user feedback.

ABSTRACT

In the contemporary era, Information and Communication Technologies (ICTs) have emerged as indispensable tools, revolutionizing various sectors including education, banking, healthcare, and e-commerce. One of the crucial domains where ICTs can significantly impact livelihoods is agriculture. This paper proposes the development of a web-based system titled "Farming Portal" aimed at empowering farmers and transforming the agricultural landscape by leveraging ICTs.

In countries like India, traditional supply chains often result in farmers receiving minimal profits while intermediaries reap significant benefits. The "Farming Portal" seeks to disrupt this paradigm by directly connecting farmers with consumers, thereby ensuring fair compensation for their produce. The core objective of this project is to integrate ICT features into the daily lives of people, facilitating the direct purchase of agricultural products.

The farming system offers a multifaceted approach to addressing the challenges faced by farmers. Firstly, it serves as an information hub, providing farmers with vital data on various aspects of agriculture including crop cultivation techniques, disease management strategies, market prices for different crops, and government schemes tailored to farmers' welfare. By equipping farmers with knowledge and resources, the system empowers them to make informed decisions, optimize crop yield, and mitigate risks.

Moreover, the "Farming Portal" functions as an efficient marketplace, enabling farmers to showcase their produce to a broader audience and negotiate fair prices directly with consumers. By eliminating intermediaries, farmers can maximize their profits and enhance their economic stability. Simultaneously, the platform facilitates the procurement of farming essentials from vendors, ensuring timely access to quality inputs.

Agricultural agencies and businesses involved in the agricultural value chain stand to benefit from the platform's comprehensive features. By fostering transparency, efficiency, and equitable distribution of resources, the system contributes to the overall development of the agricultural sector and enhances food security.

In conclusion, the "Farming Portal" represents a transformative initiative leveraging ICTs to empower farmers, streamline agricultural operations, and foster socio-economic development. Through its innovative features and inclusive design, the system has the potential to catalyze positive change and create a more sustainable and prosperous future for agricultural communities.

TABLE OF CONTENTS

Certificate	ii
Declaration	iii
Acknowledgment	iv
Preface	v
Abstract	vi
Table of Contents	vii
1. INTRODUCTION	01-06
1.1. Research Objective	04
1.2. Problem Statement	05
1.3. Objectives of the Project	06
1.4. Scope of the Project	06
2. LITERATURE REVIEW	08-11
3. PROPOSED METHODOLOGY	12-39
3.1. Methodology Overview	12
3.2. Hardware/Software Requirements	14
3.3. Implementation	18
3.3.1. Design	18
3.3.2. Coding	26
4. RESULT ANALYSIS AND DISCUSSION	38-52
4.1. Testing	39
4.2. Snapshots	48

4.3. Inference	50
4.4. Advantages and Limitations	51
5. CONCLUSION	53
6. FUTURE SCOPE OF THE PROJECT	54-55
APPENDICES	x-xxviii
List of Figures	xi
List of Tables	xii
List of Abbreviation	xii
Coding Implementation	xiii
References	xxviii

CHAPTER-1

INTRODUCTION

Agriculture, deeply entrenched in the fabric of Indian society, not only drives economic growth but also sustains the livelihoods of millions of people across the country. At its core are the farmers, the unsung heroes who work tirelessly to cultivate the land and produce the food that nourishes the nation. Their role is not merely economic but also societal, as they fulfil the fundamental need for sustenance. However, despite their indispensable contribution, many farmers find themselves trapped in a cycle of poverty and exploitation, primarily due to the pervasive influence of middlemen in the agricultural supply chain.

As India strides into the modern age, characterized by rapid technological advancements and societal transformations, the potential for leveraging technology to address age-old agricultural challenges becomes increasingly apparent. In this digital era, where smartphones have become indispensable tools for communication, commerce, and productivity, there exists a unique opportunity to integrate technological innovations into the agricultural domain. The overarching goal of this project is to harness the power of technology to bridge the gap between farmers and consumers, thereby revolutionizing the traditional farm-to-table supply chain. By enabling direct transactions between farmers and consumers, the project seeks to ensure that farmers receive fair compensation for their labour while circumventing the exploitative practices perpetuated by middlemen.[1]

Agriculture in India is not just an economic sector; it is a way of life, deeply rooted in tradition and culture. The agricultural landscape is as diverse as the nation itself, encompassing a myriad of crops, farming practices, and regional variations. From the verdant rice paddies of the east to the arid farmlands of the west, Indian agriculture reflects the rich tapestry of the country's geography and heritage. Moreover, agriculture serves as the primary source of livelihood for a significant portion of the population, particularly in rural areas where the majority of farmers reside.

However, despite its importance, Indian agriculture faces numerous challenges that hinder its growth and sustainability. One of the most pressing issues is the prevalence of middlemen in the agricultural supply chain. These intermediaries, often operating

with impunity, exploit farmers by offering low prices for their produce while selling it to consumers at inflated rates. This exploitative system not only deprives farmers of their rightful income but also perpetuates a cycle of poverty and indebtedness.[2]

Furthermore, inadequate access to market information and infrastructure exacerbates the challenges faced by farmers. Many farmers lack timely and accurate information about market prices, demand trends, and agricultural best practices, making it difficult for them to make informed decisions. Additionally, the lack of proper storage facilities leads to post-harvest losses, further diminishing farmers' income.

In this context, the emergence of technology offers a glimmer of hope for Indian agriculture. By leveraging digital tools and platforms, it is possible to empower farmers, enhance market transparency, and streamline agricultural operations. The advent of smartphones and the internet has facilitated the development of innovative solutions that can revolutionize the way farmers engage with markets, access information, and manage their farms.

The Farming Assistance Web Application represents one such innovative solution, designed to address the multifaceted needs of Indian farmers. By providing real-time market information, access to resources, and a platform for direct transactions, the application aims to empower farmers and transform the agricultural landscape. Through its comprehensive features and user-friendly interface, the application seeks to bridge the gap between farmers and consumers, ensuring fair prices for agricultural produce and fostering sustainable livelihoods for farming communities across India.

In India, agriculture is not just an economic activity; it's deeply intertwined with the country's cultural and social fabric. From the lush rice fields of the south to the wheat farms of the north, agriculture sustains millions of livelihoods and forms the backbone of rural communities. However, despite its importance, Indian agriculture faces a myriad of challenges that hinder its progress and perpetuate the cycle of poverty among farmers.

One of the most pressing issues plaguing Indian agriculture is the dominance of middlemen in the agricultural supply chain. These intermediaries often exploit farmers by offering low prices for their produce and selling it to consumers at inflated rates. This exploitative system not only deprives farmers of their rightful income but also

perpetuates a cycle of indebtedness and poverty. Moreover, the lack of transparency in pricing and transactions further exacerbates the plight of farmers, leaving them vulnerable to exploitation.

Another critical challenge facing Indian agriculture is the lack of access to timely and accurate market information. Many farmers, especially those in remote or rural areas, struggle to obtain information about market prices, demand trends, and agricultural best practices. This lack of information hampers their ability to make informed decisions about what crops to grow, when to sell, and where to sell their produce. Additionally, inadequate storage facilities lead to post-harvest losses, further diminishing farmers' income and exacerbating food insecurity in the country.

In this context, the emergence of technology offers a ray of hope for Indian agriculture. Digital tools and platforms have the potential to revolutionize the way farmers engage with markets, access information, and manage their farms. By leveraging the power of technology, it is possible to empower farmers, enhance market transparency, and improve agricultural productivity and sustainability.[3]

The Farming Assistance Web Application represents a bold step towards addressing these challenges and transforming Indian agriculture. This comprehensive digital platform is designed to provide farmers with real-time market information, access to resources, and a platform for direct transactions. By aggregating data from various sources, the application offers farmers insights into market dynamics, enabling them to make informed decisions about what crops to grow and when to sell. Additionally, the application provides tools for effective resource management, such as water and fertilizer usage, helping farmers optimize their farming practices and reduce waste.

Moreover, the Farming Assistance Web Application fosters a sense of community among farmers, enabling them to connect with peers and agricultural experts for knowledge sharing and support. The application is designed to be mobile-friendly, ensuring that farmers can access its features anytime, anywhere, using their smartphones or tablets. Furthermore, robust encryption and data protection measures are employed to safeguard user data, ensuring the security and privacy of farmers' information.

Overall, the Farming Assistance Web Application holds the potential to revolutionize Indian agriculture, empowering farmers, enhancing market transparency, and fostering sustainable livelihoods for farming communities across the country. By offering a comprehensive suite of features and capabilities, the application aims to serve as a digital partner for farmers, helping them navigate the complexities of modern agriculture with confidence and resilience.

The introduction of the Farming Assistance Web Application signifies a paradigm shift in how farmers engage with markets, access information, and manage their farms. Through real-time market data, access to resources, and a platform for direct transactions, the application aims to break the cycle of exploitation perpetuated by middlemen [2] and ensure fair compensation for farmers. Moreover, by providing tools for effective resource management and fostering a sense of community among farmers, the application seeks to enhance agricultural productivity, sustainability, and resilience.

As India stands at the cusp of technological advancement and agricultural transformation, the Farming Assistance Web Application offers a beacon of hope for the millions of farmers who toil tirelessly to feed the nation. By offering a comprehensive suite of features and capabilities, the application aims to serve as a digital partner for farmers, empowering them to navigate the complexities of modern agriculture with confidence and resilience.

In summary, the Farming Assistance Web Application represents not just a technological innovation but a beacon of hope for the future of Indian agriculture. By leveraging technology to empower farmers and enhance market transparency, the application holds the potential to revolutionize the way agriculture is practiced in India, ensuring the continued prosperity of farmers and the nation as a whole.

1.1 Research background:

The agricultural sector in India is confronted by a multitude of challenges, ranging from the pervasive influence of middlemen in the supply chain to the lack of access to timely market information and inadequate infrastructure. Academic research and scholarly literature have extensively documented these challenges and their adverse effects on farmers' livelihoods and agricultural sustainability.

Studies have highlighted the exploitative practices of middlemen who often offer low prices to farmers for their produce while selling it to consumers at inflated rates, resulting in diminished incomes for farmers and perpetuating cycles of poverty and indebtedness. Moreover, research has emphasized the importance of addressing the lack of transparency in pricing and transactions, which leaves farmers vulnerable to exploitation and exacerbates their plight.

Furthermore, academic inquiry has underscored the significance of providing farmers with access to timely and accurate market information, as well as essential resources and infrastructure. Many studies have explored the impact of inadequate market information on farmers' decision-making processes regarding crop cultivation and sales, as well as the detrimental effects of insufficient storage facilities on post-harvest losses and farmers' incomes.

In light of these challenges, researchers have recognized the potential of technology, particularly digital tools and platforms, to revolutionize Indian agriculture. The emergence of smartphones and the internet has paved the way for innovative solutions aimed at empowering farmers, enhancing market transparency, and improving agricultural productivity and sustainability.

The Farming Assistance Web Application project represents a significant step towards addressing these challenges and transforming Indian agriculture. By leveraging technology to provide farmers with real-time market information, access to resources, and a platform for direct transactions, the project aims to empower farmers and bridge the gap between farmers and consumers. The project builds upon existing research findings and insights to offer a comprehensive solution to the multifaceted challenges facing Indian agriculture.

1.2 Problem Statement:

The agricultural sector in India faces multifaceted challenges that hinder its growth, sustainability, and the well-being of farmers. Foremost among these challenges is the pervasive influence of middlemen in the agricultural supply chain. These intermediaries exploit farmers by offering low prices for their produce while selling it to consumers at inflated rates, trapping farmers in a cycle of poverty and exploitation.

Additionally, farmers encounter obstacles such as inadequate access to timely and accurate market information, leading to difficulties in making informed decisions about crop cultivation and sales. Furthermore, the lack of proper infrastructure, including storage facilities, contributes to post-harvest losses and diminishes farmers' incomes.

Overall, the central problem revolves around the need to address the exploitative practices of middlemen, improve market transparency, and provide farmers with the necessary resources and infrastructure to thrive in the agricultural sector. This problem statement underscores the urgency of implementing innovative solutions, such as the Farming Assistance Web Application, to empower farmers and transform the agricultural landscape in India.

1.3 Objectives of the Project:

- To facilitate connections between farmers and potential buyers, such as wholesalers, retailers, and consumers.
- To reduce the cost and develop an economical and user-friendly GUI application.
- To develop and create a digital platform that provides valuable support and resources to the framers, agricultural workers, and stakeholders in the agricultural industry which could address various challenges and needs within the farming sector.
- To provide suggestions based on a current word to eliminate the need to translate the full term.

1.4 Scope of the Project:

The scope of the study encompasses several key aspects aimed at addressing the challenges outlined in the problem statement and advancing the goals set forth in the introduction:

- **Middlemen Exploitation:**

The study will examine the extent of middlemen exploitation in the agricultural supply chain and its impact on farmers' livelihoods. This will involve analysing pricing mechanisms, transactional processes, and the economic disparities faced by farmers.

- **Market Transparency:**

The study will investigate strategies to enhance market transparency within the agricultural sector. This will include exploring the role of technology in providing farmers with access to real-time market information, pricing trends, and demand dynamics.

- **Access to Information:**

The study will assess the effectiveness of digital tools and platforms in improving farmers' access to timely and accurate market information. This will involve evaluating the usability and functionality of existing applications and identifying areas for improvement.

- **Infrastructure Development:**

The study will examine the importance of infrastructure development, particularly in terms of storage facilities, in reducing post-harvest losses and enhancing farmers' incomes. This will involve analysing current infrastructure gaps and proposing strategies for improvement.

- **Technology Integration:**

The study will explore the integration of technology, including smartphones, internet connectivity, and digital platforms, into the agricultural domain. This will include assessing the feasibility of implementing innovative solutions such as the Farming Assistance Web Application and evaluating their potential impact on farmers' lives.

- **Socioeconomic Implications:**

The study will consider the socioeconomic implications of empowering farmers and improving market transparency in the agricultural sector. This will involve analysing the broader implications for rural development, poverty alleviation, and food security.

The scope of the study is comprehensive, encompassing a wide range of factors that contribute to the challenges faced by farmers in India's agricultural sector. By addressing these factors and proposing innovative solutions, the study aims to contribute to the advancement of knowledge and the improvement of agricultural practices in India.

CHAPTER-2

LITERATURE REVIEW

To develop this system, we studied some previous papers.

The agricultural sector, the backbone of the global economy, has been undergoing significant transformations in recent years due to the integration of digital technologies and web-based services [4]. This literature survey provides an overview of key studies, research, and trends related to farming assistance web services and removing middlemen in agriculture.

Historical Role of Mediators in Agriculture

Historically, intermediaries have played a vital role in connecting farmers with markets and resources. Studies such as Williamson (1979) emphasized the importance of go-betweens in reducing transaction costs. However, various scholars have also highlighted the exploitative practices and inefficiencies associated with middlemen, as discussed in the work of Reardon et al. (1992) and Tilman et al. [5]

The literature review segment explores the historical role of intermediaries in agriculture, acknowledging their vital functions in connecting farmers with markets and providing value-added services. Initially praised for reducing transaction costs and facilitating efficient market transactions, intermediaries later faced scrutiny for exploitative practices, market distortions, and hindering transparency and innovation. Scholars began advocating for reducing intermediaries to achieve more equitable outcomes and transparent pricing mechanisms, potentially benefiting farmers. Overall, the segment provides a comprehensive view of intermediaries' contributions, challenges, and evolving perspectives in agricultural markets, setting the stage for further discussions on their removal.

Benefits of Removing Middlemen

Several studies have highlighted the advantages of removing middlemen from the agricultural supply chain. Swinnen (2007) argues that direct farmer-buyer interactions can lead to higher profits for farmers and more transparent pricing. Additionally,

Babcock [6] and Clemens (2004) found that eliminating intermediaries can reduce transaction costs.

This segment of the literature review outlines the benefits of removing intermediaries from agricultural supply chains:

- **Higher Profits for Farmers:** Direct engagement with buyers allows farmers to negotiate fair prices, leading to increased profits by bypassing intermediaries who often take a significant cut.
- **Transparent Pricing Mechanisms:** Eliminating intermediaries promotes transparent pricing negotiations, reducing asymmetries of information and fostering trust between farmers and buyers.
- **Reduction in Transaction Costs:** Farmers can save on fees and commissions charged by intermediaries, leading to cost savings and increased efficiency in agricultural transactions.
- **Enhanced Market Access:** Removing intermediaries enables farmers, especially small-scale producers, to access a broader range of markets and buyers, diversifying their customer base.
- **Empowerment of Farmers:** Direct engagement gives farmers greater control over their businesses, allowing them to negotiate fair prices, make informed decisions, and build long-term relationships with buyers.

Overall, these benefits highlight the potential for improved efficiency, equity, and sustainability in agricultural markets through the removal of intermediaries, emphasizing the importance of promoting direct farmer-buyer interactions.

Challenges and Concerns

While the idea of removing brokers is promising, it also raises concerns. Research conducted by the World Bank (2015) underscores the challenge faced by farmers in developing countries, who often lack the essential digital literacy and internet access required to leverage web-based services. Concerns regarding privacy and data security have been articulated by scholars like Smith et al. (2018).

This segment of the literature review examines challenges associated with removing middlemen from agricultural supply chains:

- **Digital Divide:** Many farmers lack access to technology and digital literacy, particularly in developing countries, limiting their participation in web-based agricultural services.
- **Privacy and Data Security:** Concerns about data misuse and privacy undermine trust in digital platforms, especially regarding the collection and sharing of farmers' personal information.
- **Dependence on Intermediaries:** Farmers may resist change due to their reliance on intermediaries for market access, transportation, and credit, hindering the transition to direct interactions.
- **Market Volatility:** Without intermediaries, farmers face increased exposure to market fluctuations, risking income instability and financial insecurity.
- **Infrastructure Challenges:** Limited access to reliable electricity and internet connectivity in rural areas impedes farmers' adoption of web-based agricultural services, further exacerbating the digital divide.

Addressing these challenges is crucial for realizing the potential of digital agricultural platforms and ensuring equitable access for farmers worldwide.

Case Studies of Successful Implementation

Numerous case studies showcase successful implementations of web-based farming assistance services. The case of AgroStar in India, described by Kumar [7] and Qureshi (2018), demonstrates how a mobile app can connect farmers with agricultural inputs and advisory services. Similarly, the FarmLogs platform in the United States, highlighted in Lowenberg-DeBoer and Erickson (2018), provides insights into the benefits of data-driven farm management [8].

"Design of Web Portal for E-Trading for Farmers" by Vishi Purushottam Paliwal et al. describes the design and development of a web portal aimed at facilitating e-trading between farmers and buyers. The authors of this article also highlighted the importance of educating farmers about e-trading and providing them with the necessary training and support.

These case studies demonstrate how web-based platforms like AgroStar and FarmLogs are transforming agriculture by empowering farmers with insights, improving efficiency, and maximizing profitability. They highlight the potential of digital

technologies to revolutionize farming practices globally, bridging the digital divide and promoting sustainable agriculture.[9]

Future Prospects and Policy Considerations

The future of web-based farming assistance services is a topic of ongoing research. Scholars like Hobbs (2018) discuss the role of artificial intelligence and machine learning in shaping the future of agriculture. Policy and regulatory considerations are also significant, as highlighted by Marette and Messéan (2017). The article "A Study of Blockchain Technology [10] in Farmer's Portal," published on IEEE Xplore, delves into the transformative potential of blockchain technology within the realm of farmer's portals. The authors propose a blockchain-based farmer's portal architecture that integrates components such as smart contracts, digital identities, and data storage. The article also discusses the potential benefits of this architecture, such as increased efficiency, reduced costs, and improved data security and privacy.[11]

This segment of the literature review discusses prospects and policy considerations for web-based farming assistance services:

- **Emerging Technologies and Trends:** Technologies like artificial intelligence (AI) and blockchain hold promise for improving farm efficiency and transparency. AI can optimize resource allocation, while blockchain [12] enhances trust and traceability in supply chains.
- **Policy and Regulatory Considerations:** Supportive policy frameworks are crucial for fostering innovation and digital adoption in agriculture. Governments should incentivize investment in digital infrastructure, address data privacy concerns, and promote digital literacy among farmers.[13]

Addressing these considerations will be vital for realizing the potential of web-based farming assistance services in improving agricultural productivity and sustainability while ensuring fair outcomes for farmers. This literature survey highlights the multifaceted nature of the transition towards web-based farming assistance services and the removal of intercessors in agriculture. With the agricultural sector's ongoing evolution, research in this domain remains pivotal for comprehending the shifting landscape and facilitating informed decision-making.

CHAPTER-3

PROPOSED METHODOLOGY

The system proposed by us aims to streamline the marketing of agricultural products for agriculturists, benefiting both farmers and buyers alike. The system is developed as a web platform using HTML, CSS, JavaScript, PHP, Python-Flask and MySQL, featuring interfaces tailored for both large and small screens. Both the ranchers and end user need to log in to the system by providing all necessary details to access its features.

This system is a website as well as a mobile application. Cultivators can use the system directly by entering the URL of our website or just by opening the tool. At this stage, users will receive fundamental husbandry-related information. If they wish to engage in selling or purchasing, such as rural workers intending to sell their produce, registration through the provided form and subsequent login are mandatory. Similarly, for purchasing, registration and login procedures are also required.

Apart from farmers, two types of people can also benefit from this system:

- Consumer
- Supplier

The information provided by the farmer regarding their products will be stored in the database. All details of the farmer, including their product, price, location, and contact number, will be showcased to the end user during the purchase process. Additionally, this system offers multi-language support to enhance user-friendliness and accessibility across various local languages.

3.1 Methodology Overview

To achieve these objectives, the project will follow an Iterative Waterfall methodology, which integrates sequence phases of the traditional Waterfall model and their iterative elements to gain flexibility and adaptability. The Iterative Waterfall Approach suits the requirements for the projects with a defined approach to the build running along a sequentially. Only the Iterative Waterfall Approach offers the advantage of both sequentially developing along the process and iteratively refining the end product while at the same time incorporating any changes or feedback throughout the development

cycle. This approach ensures systematic progress while allowing for flexibility and responsiveness to feedback. The phases include:

Feasibility Study: The feasibility study serves as a pivotal step in the inception phase of our project, providing a comprehensive evaluation of the project's viability and potential for success. Through meticulous analysis and assessment, we aim to ascertain the feasibility of implementing our accident detection and prevention system, considering various factors including technical, economic and operational considerations.

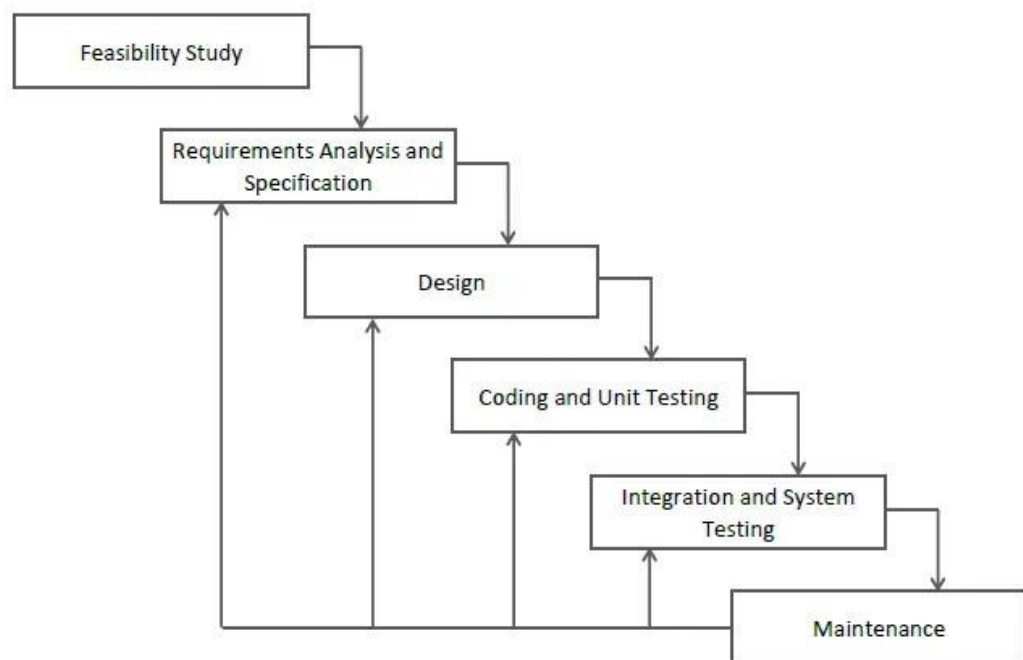


Fig 3.1- Iterative Water Fall [14]

Requirements Gathering: At the onset of the project, we embark on a comprehensive journey of requirements gathering, collaborating closely with drivers, road safety experts, law enforcement officials, and potential end-users. Through direct engagement and consultation, we aim to discern the precise needs and aspirations surrounding our project. This phase serves as the bedrock for our endeavour, enabling us to define clear objectives, delineate project scope, and elucidate functional requirements essential for the success of our drowsiness and accident detection system.

System Design: With a firm grasp of the project's requirements, we transition seamlessly into the system design phase. Here, we meticulously craft detailed architectural blueprints, elaborate database, and intuitive user mock-ups. This iterative

process allows us to refine and enhance our initial designs. Our focus remains steadfast on creating a robust and user-centric system architecture that seamlessly aligns with the project's objectives and requirements.

Implementation (Iterative Development): The implementation phase embodies a series of iterative cycles, each dedicated to implementing and testing specific features or modules of our accident detection and prevention system. Guided by agile principles, our development team engages in incremental stages of development, ensuring continuous integration and validation of our system's functionalities. Frequent reviews and feedback sessions with project advisors and mentors ensure that our development efforts remain aligned with the expectations and needs of our project.

Testing and Quality Assurance: Quality is paramount in every facet of our project, and as such, rigorous testing and quality assurance procedures are ingrained into our development process. We conduct exhaustive rounds of testing, encompassing unit testing, integration testing, system testing, and user acceptance testing. Through meticulous scrutiny and validation, we ensure the reliability, robustness, and functionality of our healthcare management system, thereby instilling confidence in its efficacy and performance.

Deployment and Maintenance: Upon reaching the culmination of our development journey, we deploy the system in real-world environments and provide ongoing maintenance and updates. This phase ensures that the system remains effective and up-to-date with the latest advancements in technology. Installing the system in vehicles and integrating it with existing infrastructure. Providing training to users on how to use the system effectively. Regularly updating the system to fix bugs, improve performance, and add new features.

3.2 HARDWARE AND SOFTWARE REQUIREMENTS

SOFTWARES USED:

These technologies are fundamental building blocks for creating modern web applications, each serving different purposes in the development process.

HTML: HTML, short for Hypertext Markup Language, is the backbone of the internet. It's a markup language that web developers use to structure the content of web pages.

Each web page you visit is written in HTML, which tells your web browser how to display the content. HTML consists of a series of elements, enclosed in opening and closing tags, that define the structure and semantics of the content. These elements can range from headings and paragraphs to images, links, forms, and more.

CSS: Cascading Style Sheets, is the design language of the web. Its primary function is to dictate the presentation and layout of web pages written in HTML. In simpler terms, CSS is responsible for the visual appearance of a website, determining how elements are displayed, styled, and positioned on the page. At its core, CSS operates by selecting HTML elements and applying styling rules to them. One of the key benefits of CSS is its ability to separate the content and structure of a web page (handled by HTML) from its presentation and style.

JavaScript: JavaScript is a versatile programming language that adds interactivity and dynamic behavior to web pages. Unlike HTML and CSS, which primarily deal with the structure and presentation of content, JavaScript focuses on enhancing the functionality and behavior of web pages. In a nutshell, JavaScript allows developers to create interactive elements, handle user interactions, and manipulate the content of web pages. JavaScript is a client-side scripting language, meaning it runs on the user's web browser rather than on a web server.

jQuery: It is a fast, lightweight, and feature-rich JavaScript library that simplifies the process of interacting with HTML elements, manipulating the DOM, and handling events in web development. It provides a concise and powerful set of tools for front-end developers to write more efficient and expressive JavaScript code. It provides a unified interface for selecting HTML elements, applying CSS styles, animating elements, and handling events across different web browsers, ensuring consistent behavior and compatibility.

Bootstrap: It is a popular open-source front-end framework for building responsive and mobile-first websites and web applications. Developed by Twitter, Bootstrap provides a comprehensive set of pre-styled HTML, CSS, and JavaScript components, as well as powerful layout utilities, to streamline the process of web development. These components are fully customizable and can be easily integrated into web projects to achieve a polished and professional look.

PHP (Hypertext Preprocessor): PHP, which stands for Hypertext Preprocessor, is a server-side scripting language primarily used for web development and building dynamic websites and web applications. PHP is an open-source language, meaning it is freely available for anyone to use and modify, and it is widely adopted for its simplicity, versatility, and compatibility with various web servers and operating systems. This dynamic nature of PHP enables developers to create interactive and data-driven.

Python - Flask: Flask is a lightweight and flexible web framework written in Python. It is designed to make web development simple and straightforward by providing developers with the tools they need to build web applications quickly and efficiently. One of the key features of Flask is its routing system, which allows developers to map URLs to Python functions, known as view functions. Flask also provides support for URL parameters, variable routing, and URL redirection, making it flexible enough to handle a wide range of routing scenarios.

MY SQL: SQL (Structured Query Language) is a programming language used in backend development to interact with relational databases. It allows developers to create, read, update, and delete data in databases, making it essential for managing data in web applications and other backend systems.

Database:

A Database Management System (DBMS) is complex software designed to manage large sets of structured data and perform operations requested by multiple users. Examples of DBMSs include Oracle, DB2, Microsoft Access, Microsoft SQL Server, Firebird, PostgreSQL, MySQL, SQLite. These systems are typically used by database administrators to create and maintain database systems in various applications like accounting, human resources, and customer support. They control data organization, storage, management, and retrieval, featuring a modelling language to define database schemas and often supporting the Open Database Connectivity API for standard access methods. DBMSs handle data structures optimized for large volumes of data stored on permanent devices, and provide query languages and report writers for user interaction. They ensure data security through access controls and sub schemas and support personal database management while lacking multi-user controls.

Table 3.1: Hardware Requirements for Web-Based Farming Assistance Services

NUMBER	DESCRIPTION	TYPE
1	Processor	1.1 GHz or faster processor
2	RAM	Minimum 2 GB of RAM or more
3	HDD	2.5 GB free disk space
4	Network Interface	Ethernet or Wi-Fi for Internet connection
5	Input Devices	DVD-ROM Drive, keyboard & Mouse
6	Monitor/Screen	1366 * 768 or higher resolution display
7	Hard Drive	54000 RPM hard drive

Table 3.2: Software Requirements for Web-Based Farming Assistance Services

NUMBER	DESCRIPTION	TYPE
1	Technology	Python main editor (user interface): PyCharm Community
2	Integrated Development Environment (IDE)	Visual Studio Code
3	Operating System	Windows 10
4	Browser	Google Chrome/Internet Explorer
5	Stack	XAMPP (Version-3.7)
6	Monitor/Screen	Sublime text 3

3.3 Implementation

3.3.1 Design

Project Module Overview

The provided modules and their components are part of a software system designed to manage user information, farmer registration, agricultural products, farming information, action logging, and relationships between entities. Here's a detailed explanation of each module and its components:

Module 1 (User Management Module):

The User Management Module is essential for handling all user-related functionalities within the system. It facilitates new user registrations, allowing individuals to create accounts by providing necessary details like username, password, and email. Once registered, users can log in and out securely, with the system authenticating their credentials to ensure secure access. Additionally, this module enables users to manage their profiles by updating their personal information such as username, password, and email, ensuring that user data remains current and accurate.

Components:

- **User Registration:** This component allows new users to create an account in the system by providing necessary information such as username, password, and email.
- **User Authentication (Login/Logout):** This component manages the authentication process, enabling users to log in to and out of the system securely.
- **User Profile Management:** This component allows users to manage their profiles, including updating their username, password, and email information.

Module 2 (Farmer Registration Module):

The Farmer Registration Module is designed to manage the detailed information of farmers within the system. It allows farmers to register by providing comprehensive details including their unique ID (Rid), name, type of farming they practice, Aadhaar number, address, phone number, age, and gender. This module not only captures this information but also enables the farmers to manage and update their profiles. This

ensures that all farmer-related data is up-to-date and readily available for other system functionalities.

Components:

- **Register Farmer Information:** This component collects and stores detailed information about farmers, including their unique ID (Rid), name, type of farming, Aadhaar number (a unique identifier used in India), address, phone number, age, and gender.
- **Farmer Profile Management:** This component enables the editing and updating of farmer profiles, ensuring their information is current and accurate.

Module 3 (Product Management Module):

The Product Management Module oversees the addition and management of agricultural products. It allows users to add new products by entering details such as product ID (P-Id), name, description, price, and the user's contact information (username, email). Once added, the module also manages the detailed information of these products, allowing for updates and modifications.

Components:

- **Add Agricultural Products:** This component allows users to add new agricultural products to the system. It includes information such as the user who added the product (username, email), product ID (P-Id), product name, description, and price.
- **Product Details Management:** This component manages the details of each product, allowing for updates and modifications to the product information as needed.

Module 4 (Farming Information Module):

The Farming Information Module is responsible for managing data related to different farming practices. It categorizes various farming types by assigning them unique farming IDs (Fid) and manages this classification effectively. Additionally, it integrates a wide range of farming-related data into the system, making this information accessible for other modules and aiding in comprehensive data management and analysis for farming practices.

Components:

- **Farming Type Management:** This component deals with the categorization and management of different farming types, identified by a unique farming ID (Fid).
- **Farming Data Integration:** This component integrates various farming-related data into the system, ensuring it is accessible and usable for other modules.

Module 5 (Action Logging Module):

The Action Logging Module tracks and logs user actions within the system. It records specific actions taken by users, including the action type, a unique action ID, timestamp, and any related farming ID (F-Id). This module also manages the history of these logged actions, allowing for detailed retrieval and analysis of past user activities. This functionality is crucial for auditing, monitoring user behaviour, and system security.

Components:

- **Action Trigger Logging:** This component logs specific actions taken by users, including the action ID, type of action, timestamp, and related farming ID (F-Id).
- **Action History Management:** This component manages the history of logged actions, allowing for the retrieval and analysis of past actions.

Module 6 (Relationship Management Module):

The Relationship Management Module manages the intricate relationships between different entities within the system. It tracks and maintains associations between users and products, ensuring that interactions are correctly logged. Additionally, it manages the relationships between users and their actions, providing a clear history of activities. This module also oversees the relationships between products and farming types, ensuring that product data is accurately linked to relevant farming information. This comprehensive relationship management supports efficient data organization and retrieval across the system.

Components:

- **User-Product Relationships:** This component manages the relationships between users and the products they interact with, ensuring that user-product associations are tracked and maintained.
- **User-Action Relationships:** This component manages the relationships between users and their actions within the system, tracking what actions each user has taken.
- **Product-Farming Relationships:** This component handles the relationships between products and farming types, ensuring that each product is correctly associated with relevant farming data.

Each module and its components work together to create a comprehensive system that manages user accounts, farmer registrations, agricultural products, farming information, action logs, and relationships between various entities. This structure ensures that the system is organized, scalable, and capable of supporting the needs of users, farmers, and administrators.

BLOCK DIAGRAM

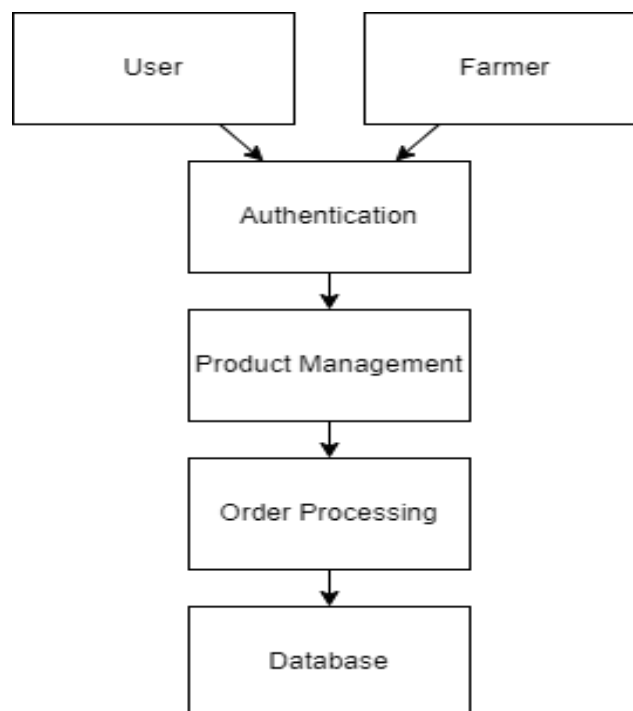


Fig 3.2: The Block Diagram of the Application for buying and selling of the product showing the connection between the cultivators and customers.

A block diagram, is a visual representation of a system or process using blocks to represent components or stages, and lines or arrows to show connections or flow between them.

In Figure 3.2. the block diagram of the proposed system is explained. This block diagram represents a system designed to streamline farming assistance services by eliminating middlemen in the agricultural business. The system has two main users: farmers and consumers. The top-level blocks include "User" and "Farmer," indicating the primary participants in the system. The core functional blocks are "Authentication," "Product Management," "Order Processing," and "Database." The Authentication block handles the login and registration processes for both users and farmers, ensuring secure access to the system. The Product Management block allows farmers to manage their product listings, including adding, updating, and removing products. The Order Processing block manages the entire order lifecycle, from placement by users to confirmation and fulfillment by farmers, including payment processing and order tracking. The Database block underpins the system by storing all relevant data, such as user profiles, product information, and order histories, ensuring data security and integrity.

SCHEMA DIAGRAM

A schema diagram is a visual representation of a database's structure, illustrating how data is organized into tables and fields and how these tables are related to each other. Key components include entities or tables, which are depicted as rectangles containing fields or attributes, such as 'UserID', 'Name', and 'Email'. Primary keys (PK) are unique identifiers for each record in a table, while foreign keys (FK) are attributes that link tables together to maintain referential integrity.

The schema diagram serves multiple purposes: it helps in the design and planning of the database, acts as a documentation tool for developers and administrators, facilitates communication among team members, and assists in the maintenance and troubleshooting of the database.

Fig 3.3 represents a relational database schema designed to manage agricultural data, including information about users, farmers, farming types, agricultural products, and triggered actions. The database consists of five main tables: USER, REGISTER, FARMING, ADDAGROPRODUCTS, and TRIG.

This schema efficiently organizes and retrieves information about users, farmers, farming practices, agricultural products, and triggered actions, ensuring a structured approach to managing agricultural data.

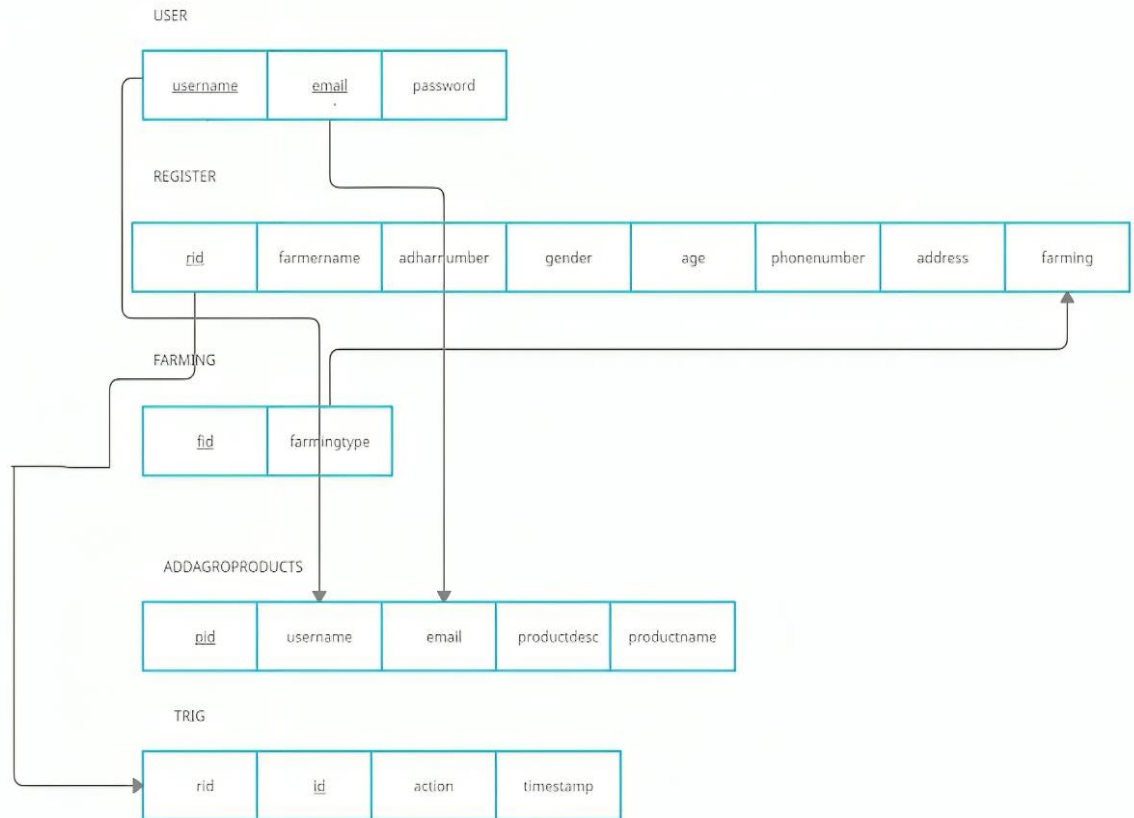


Fig 3.3: Schema Diagram of the platform containing the modules giving the visual representation of the database's structure.

USE CASE DIAGRAM

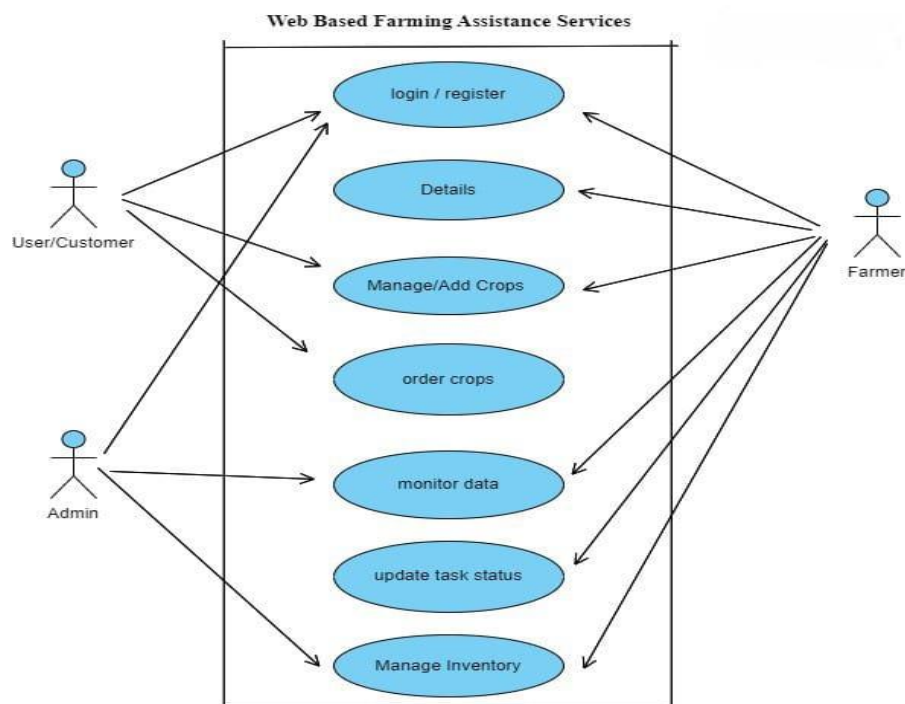


Fig 3.4: Use Case Diagram of the platform of Farming Assistance Services.

DATA FLOW DIAGRAM (0-Level)

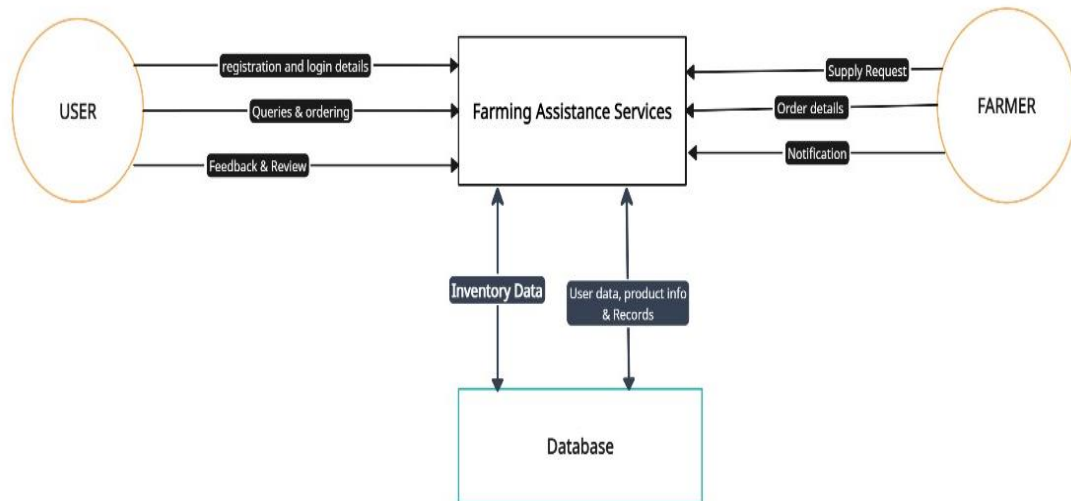


Fig 3.5: Level 0 Data Flow Diagram: providing an entire view of the system.

The Data Flow Diagram of Web Based Farming Assistance Services developed using the Web Application Development is shown in above figure. This DFD provides a high-level overview of how data flows within a farming assistance services system, from input gathering to providing personalized advice and facilitating market access for farmers.

DATA FLOW DIAGRAM (1-Level)

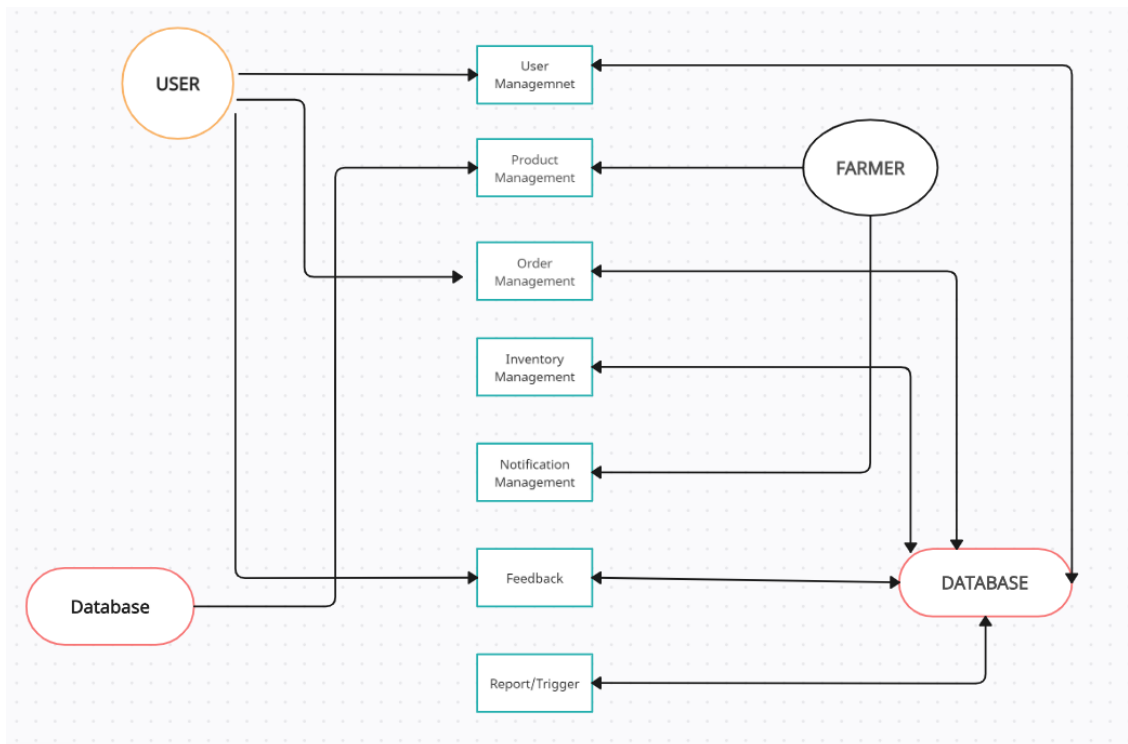


Fig 3.6: Level 1 Data Flow Diagram: detailed subprocess of the system.

A Level 1 Data Flow Diagram (DFD) breaks down the main process of a system (shown in a Level 0 DFD) into sub-processes. It shows detailed interactions within the system, including processes, data stores, data flows, and external entities. This level of DFD helps to understand how data moves and is transformed within the system, offering a more detailed view for system analysis and design.

This Data Flow Diagram (DFD) Level 1 outlines the interactions between users, farmers, and a database through various management processes. Users can manage profiles and products, place orders, provide feedback, and receive notifications. Farmers manage their products, inventory, and orders, and also receive notifications. The system includes user management for handling user operations, product management for adding and updating products, order management for processing orders, inventory management for overseeing stock levels, notification management for sending alerts, feedback for collecting user input, and report/trigger for generating reports and actions. The central database stores and retrieves data for all these processes. The arrows depict the data flow between users, farmers, management modules, and the database.

ENTITY RELATIONSHIP DIAGRAM

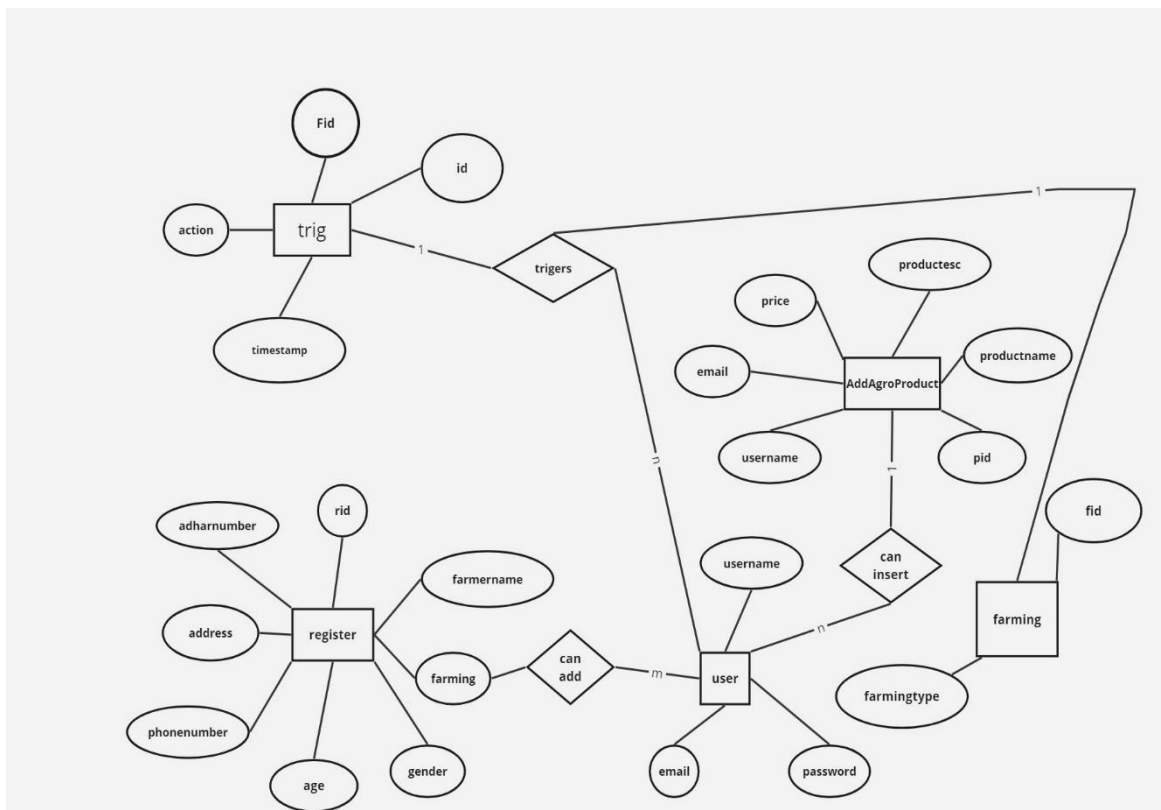


Fig 3.7: The ER Diagram showing the logical representation of the database

An Entity-Relationship (ER) diagram is a graphical representation of the logical structure of a database, showcasing the entities (such as people, objects, or concepts) within a system and the relationships between them. Attributes of entities are shown within the rectangles, helping to define the properties or characteristics of each entity.

ER diagrams are valuable tools for database designers and developers to visualize and communicate the structure of a database system before its implementation. They help ensure that all necessary entities and relationships are identified and properly defined, laying the groundwork for efficient and effective database design.

3.3.2. Coding

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

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/databas_table_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=productn
ame,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')

        phonenumber=request.form.get('onenumber')

        address=request.form.get('address')

        farmingtype=request.form.get('farmingtype')

    #query=db.engine.execute(f'UPDATE`register`SET`farmername`='{farmername}',`a
    dharnumber`='{adharnumber}',`age`='{age}',`gender`='{gender}',`onenumber`='{p
    honenumber}',`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.phonenumber=phonenumber

    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`,`a
ge`,`gender`,`phonenumber`,`address`,`farming`)VALUES('{farmername}','{adharnu
mber}','{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)

```

Signup.html:

```

{% extends 'auth.html' %}

{% block title %}

Signup

{% endblock title %}{% block content %}

<h1 class="mb-1"><span>Sign Up</span> </h1>

{% with messages=get_flashed_messages(with_categories=true) %}

{% if messages %} {% for category, message in messages %}

<div class="alert alert-{{category}} alert-dismissible fade show" role="alert">

    {{message}}</div>

{% endfor %} {% endif %} {% endwith %}

<form class=" text-white py-3 px-3" action="/signup" method="post">

```

```

<div class="form-group"> <label for="exampleInputPassword1">UserName</label>

  <input type="text" class="form-control mb-2" name="username" id="username"
required></div>

<div class="form-group"><label for="exampleInputEmail1">Email address</label>

  <input type="email" class="form-control mb-2" id="exampleInputEmail1"
name="email" aria-describedby="emailHelp" required>

  <small id="emailHelp" class="form-text text-muted">We'll never share your email
with anyone else.</small></div>  <div class="form-group">  <label
for="exampleInputPassword1">Password</label>

  <input type="password" class="form-control mb-2" name="password"
id="exampleInputPassword1" required></div><br>  <button type="submit"
class="form-control bg-success text-white">Sign In</button>

  <p class="mb-2 pb-0">Already User? </p><a href="/login" class="about-btn
scrollto">Login</a></form>

```

```
{% endblock content %}
```

login.html:

```

{% extends 'auth.html' %} {% block title %}login {% endblock title %} {% block
content %}

<h1 class="mb-4 pb-0"><span>Login</span> </h1>

{% with messages=get_flashed_messages(with_categories=true) %} {% if messages
%}

{% for category, message in messages %}

<div class="alert alert-{{category}} alert-dismissible fade show" role="alert">

  {{message}} </div> {% endfor %} {% endif %} {% endwith %}

<form class=" text-white py-3 px-3" action="/login" method="post">

  <div class="form-group">

```

```

<label for="exampleInputEmail1">Email address</label>

<input type="email" class="form-control mb-3" id="exampleInputEmail1"
name="email" aria-describedby="emailHelp" required></div> <div class="form-
group">

<label for="exampleInputPassword1">Password</label>

<input type="password" class="form-control" id="exampleInputPassword1"
name="password" required> </div><br><button type="submit" class="form-control
bg-success text-white">Login</button>

<p class="mb-2 pb-0">New User?</p> <a href="/signup" class="about-btn
scrollto">Signup</a></form>

```

```
{% endblock content %}
```

index.html:

```
{% extends 'base.html' %} {% block title %}
```

Student

```
{% endblock title %} {% block home %}
```

menu-active

```
{% endblock home %} {% block body %}
```

```
<footer id="footer"> <div class="footer-top"> </div><div class="container">
```

```

<div class="credits"><br> Developed by <a href="/">Vrishi and Shruti
</a></div></div> </footer><!-- End Footer -->

```

```
{% endblock body %}
```

addagroproducts.html:

```
{% extends 'base.html' %} {% block title %}
```

Add AgroProducts

```
{% endblock title %} {% block body %}
```

```

<h3 class="text-center"><span>Add Agro Products</span> </h3>

{% with messages=get_flashed_messages(with_categories=true) %}

{% if messages %} {% for category, message in messages %}

<div class="alert alert-{{category}} alert-dismissible fade show" role="alert">

    {{message}}</div> {% endfor %} {% endif %} {% endwith %}

<br><div class="container"><div class="row">

<div class="col-md-4"></div><div class="col-md-4">

<form action="/addagroproduct" method="post"><div class="form-group">

<label for="username">Farmer Name</label>

<input type="text" class="form-control" name="username" value="{{current_user.username}} " id="{{current_user.username}} " required></div><br>

<div class="form-group"><label for="username">Farmer Email</label>

<input type="email" class="form-control" name="email" value="{{current_user.email}} " id="{{current_user.email}} " required></div><br>

<div class="form-group"><label for="rollno">Product Name</label>

<input type="text" class="form-control" name="productname" id="productname" required></div><br>

<div class="form-group"> <label for="productdesc">Product Description</label>

<textarea class="form-control" name="productdesc" id="productdesc" required></textarea>

</div><br><div class="form-group"><label for="price">Price</label>

<input type="number" class="form-control" name="price" id="price" required></div><br>

<button type="submit" class="btn btn-success btn-block">Add Product</button></form><br><br></div>

<div class="col-md-4"></div></div></div> {% endblock body}

```


CHAPTER-4

RESULT ANALYSIS AND DISCUSSION

4.1 TESTING

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include the process of executing a program or application with the intent of finding software bugs (errors or other defects).

Software testing involves the execution of a software component or system component to evaluate one or more properties of interest. In general, these properties indicate the extent to which the component or system under test:

- meets the requirements that guided its design and development,
- responds correctly to all kinds of inputs,
- performs its functions within an acceptable time,
- is sufficiently usable.

Black-box testing

Black-box testing treats the software as a "black box", examining functionality without any knowledge of internal implementation, without seeing the source code. The testers are only aware of what the software is supposed to do, not how it does it. Black-box testing methods include: equivalence partitioning, boundary value analysis, all-pairs testing, state transition tables, decision table testing, fuzz testing, model-based testing, use case testing, exploratory testing and specification-based testing.

White-box testing

White-box testing (also known as clear box testing, glass box testing, transparent box testing and structural testing, by seeing the source code) tests internal structures or workings of a program, as opposed to the functionality exposed to the end-user. In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the

code and determine the appropriate outputs. This is analogous to testing nodes in a circuit, e.g. in-circuit testing (ICT).

Testing levels: There are generally four recognized levels of tests: unit testing, integration testing, component interface testing, and system testing. Tests are frequently grouped by where they are added in the software development process, or by the level of specificity of the test.

Unit testing: Unit testing, also known as component testing, refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors. Unit testing is a software development process that involves synchronized application of a broad spectrum of defect prevention and detection strategies in order to reduce software development risks, time, and costs. It is performed by the software developer or engineer during the construction phase of the software development lifecycle.

Integration testing: Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design. Software components may be integrated in an iterative way or all together ("big bang"). Normally the former is considered a better practice since it allows interface issues to be located more quickly and fixed. Integration testing works to expose defects in the interfaces and interaction between integrated components (modules).

Component interface testing: The practice of component interface testing can be used to check the handling of data passed between various units, or subsystem components, beyond full integration testing between those units. The data being passed can be considered as "message packets" and the range or data types can be checked, for data generated from one unit, and tested for validity before being passed into another unit. One option for interface testing is to keep a separate log file of data items being passed, often with a timestamp logged to allow analysis of thousands of cases of data passed between units for days or weeks. Tests can include checking the handling of some extreme data values while other interface variables are passed as normal values.

System testing: System testing, or end-to-end testing, tests a completely integrated system to verify that it meets its requirements. For example, a system test might involve

testing a logon interface, then creating and editing an entry, plus sending or printing results, followed by summary processing or deletion (or archiving) of entries, then logoff.

Operational Acceptance testing: Operational Acceptance is used to conduct operational readiness (pre-release) of a product, service or system as part of a quality management system. OAT is a common type of non-functional software testing, used mainly in software development and software maintenance projects. This type of testing focuses on the operational readiness of the system to be supported, and/or to become part of the production environment. Hence, it is also known as operational readiness testing (ORT) or Operations readiness and assurance (OR&A) testing.

Testing Strategies for Each Module

User Management Module:

- **Unit Testing:** This entails meticulously testing functions handling user registration, authentication, and profile management. Each function should be probed to verify its accuracy in processing and storing user data.
- **Integration Testing:** Integration tests evaluate the interaction between user registration, authentication, and profile management components. This ensures that data flows accurately between components, and authentication processes rely on precise registration data.

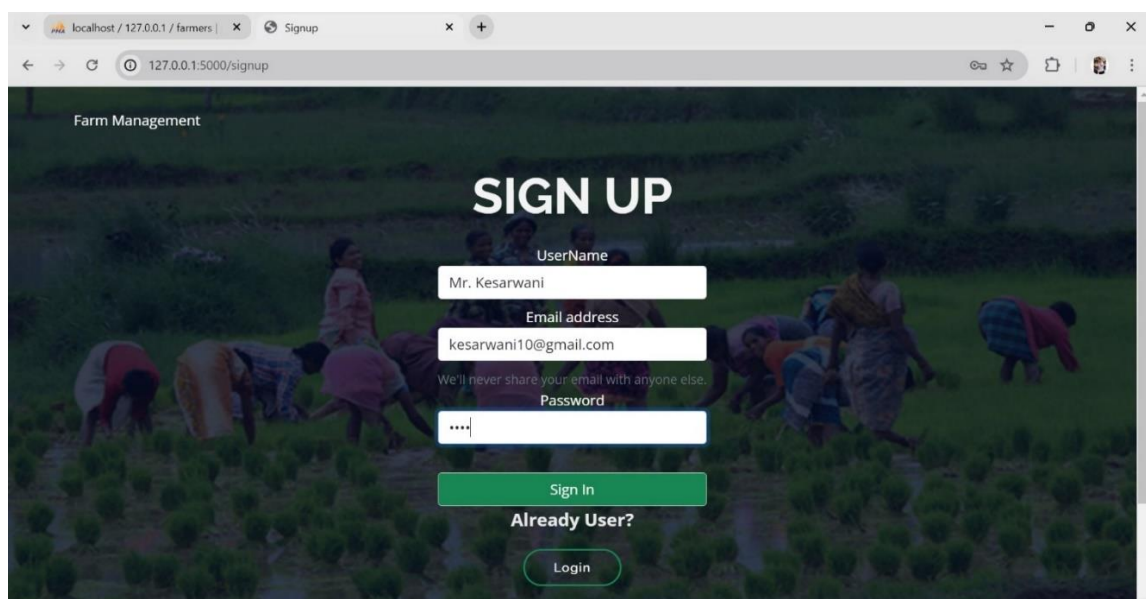


Fig 4.1: The Screenshot Image showing New User Registration or Sign-Up using mail-id, password and user name.

- **Component Interface Testing:** Component interfacing tests ensure seamless data transmission between user registration, authentication, and profile management components. They validate that data exchanged between components is consistent and that error handling mechanisms are effectively communicated.
- **System Testing:** This phase thoroughly evaluates the user management process, including registration, authentication, and profile management. It validates seamless user interactions and ensures robustness throughout the process.

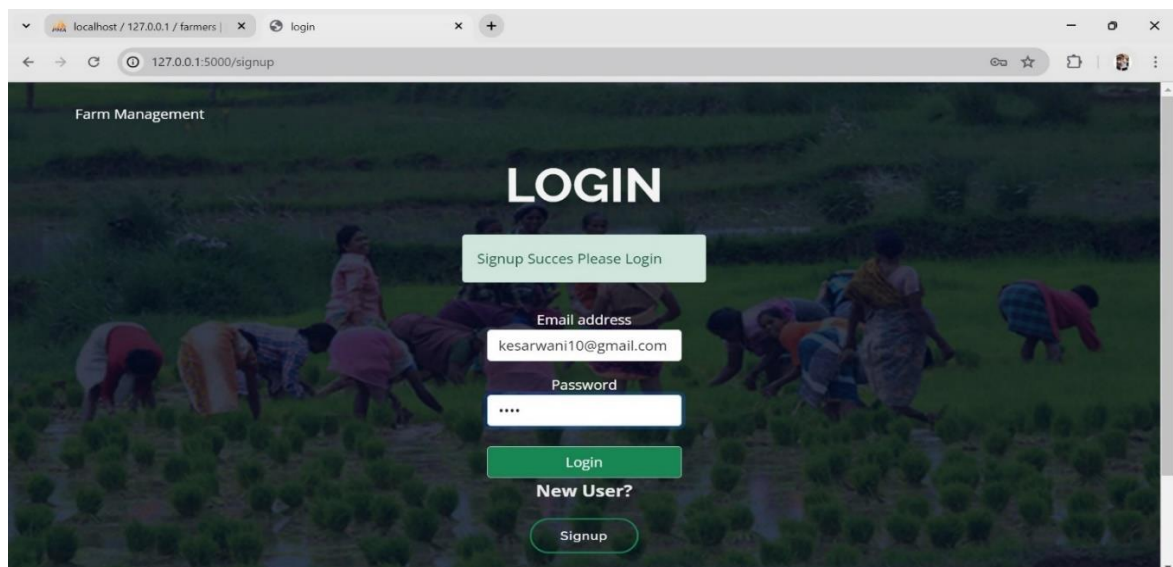


Fig 4.2: The Screenshot Image where upon successful New User Sign-Up, the user is asked to Login using their credentials.

- **Operational Acceptance Testing (OAT):** This phase validates scalability requirements, ensuring the system can handle a large volume of users. It also tests the usability of the user management interface to ensure it's intuitive for end-users.

Farmer Registration Module:

- **Unit Testing:** Test the farmer registration and profile management functions separately. This allows us to verify that farmer data is processed correctly and that the profile management functions properly interact with the registration process.
- **Integration Testing:** Integration testing will focus on how farmer registration and profile management components interact. This ensures that farmer data is

passed accurately between components and that profile updates are reflected in the registration process.

- **Component Interface Testing:** Testing will focus on how farmer registration and profile management components interface. This ensures that farmer registration data is correctly passed to the profile management component and that profile updates are properly communicated back.

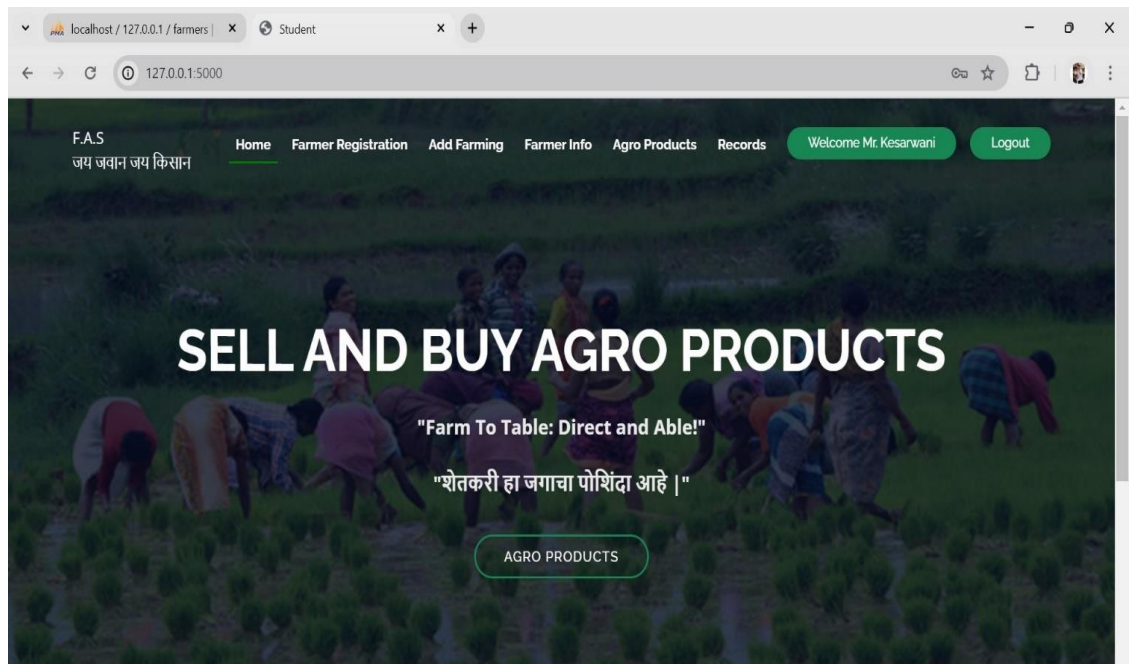


Fig 4.3: The Screenshot Image where after Login the user enters the home page with a Welcome Message alongside their name.

- **System Testing:** Testing will focus on the entire process of farmer registration and profile management. This ensures that farmer information is accurately processed and stored throughout the registration process.
- **Operational Acceptance Testing (OAT):** Testing will focus on scalability requirements and the usability of the farmer registration interface.

Product Management Module:

- **Unit Testing:** Unit tests here are vital for functions associated with adding products and managing their details. These tests validate that product data is stored accurately and that updates are reflected correctly, maintaining data integrity throughout the product lifecycle.
- **Integration Testing:** Integration testing for this module focuses on how adding products and managing product details components interact. This validates that

product details are correctly linked to added products and that updates to product details are communicated effectively.

- **Component Interface Testing:** Component interfacing tests ascertain how adding products and managing product details components interact. This validates that product data transmission between components is accurate and that updates are communicated effectively.

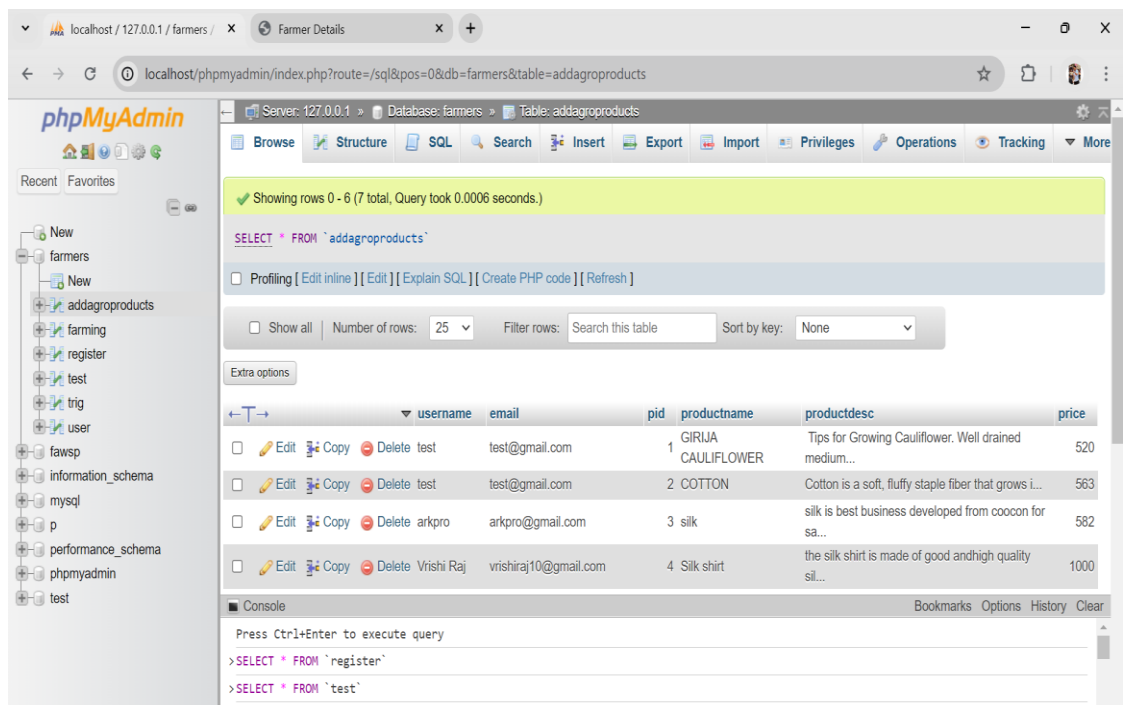


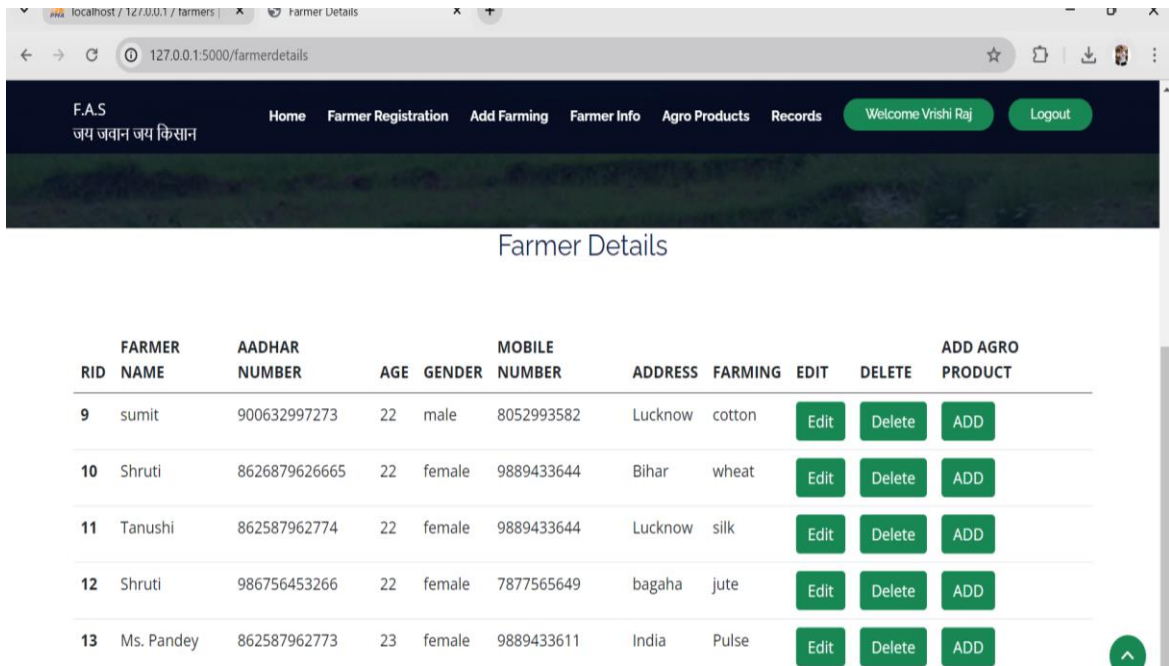
Fig 4.4: The Screenshot Image showing the database of the agricultural products/items which are being displayed or put for display by the registered farmers for the customers.

- **System Testing:** This phase evaluates the product management process holistically, ensuring smooth handling from product addition to details management. It validates data consistency and integrity across the product lifecycle.
- **Operational Acceptance Testing (OAT):** This phase validates scalability requirements and tests the usability of the product management interface to ensure it's easy to navigate.

Farming Information Module:

- **Unit Testing:** Here, the unit testing regimen zeroes in on functions managing farming types and integrating farming data. Ensuring these functions operate correctly guarantees that farming data integration is accurate and updates to farming types are appropriately propagated.

- **Integration Testing:** Integration tests here ascertain how managing farming types and integrating farming data components interact. It guarantees accurate farming data integration and effective communication of updates.
- **Component Interface Testing:** Testing here ensures effective interaction between managing farming types and integrating farming data components. It validates the proper communication of updates between components, maintaining data consistency.



The screenshot shows a web application titled 'Farmer Details'. It features a navigation bar with links: Home, Farmer Registration, Add Farming, Farmer Info, Agro Products, and Records. A user is logged in as 'Welcome Vrishi Raj' with a 'Logout' button. Below the navigation bar, there is a table listing registered farmers. Each row contains farmer details and three action buttons: 'Edit', 'Delete', and 'ADD'.

FARMER RID	FARMER NAME	AADHAR NUMBER	AGE	GENDER	MOBILE NUMBER	ADDRESS	FARMING	EDIT	DELETE	ADD AGRO PRODUCT
9	sumit	900632997273	22	male	8052993582	Lucknow	cotton	Edit	Delete	ADD
10	Shruti	8626879626665	22	female	9889433644	Bihar	wheat	Edit	Delete	ADD
11	Tanushi	862587962774	22	female	9889433644	Lucknow	silk	Edit	Delete	ADD
12	Shruti	986756453266	22	female	7877565649	bagaha	jute	Edit	Delete	ADD
13	Ms. Pandey	862587962773	23	female	9889433611	India	Pulse	Edit	Delete	ADD

Fig 4.5: The Screenshot Image showing every Registered Farmer's Registration Detail where they can choose to edit/delete/add their agricultural products and its details.

- **System Testing:** System testing here ensures the seamless integration of farming type management and data integration processes. It validates accurate farming data integration and consistent updates.
- **Operational Acceptance Testing (OAT):** Operational acceptance testing validates scalability requirements and tests the usability of the farming information interface, ensuring it's user-friendly.

Action Logging Module:

- **Unit Testing:** Unit tests for action logging functions delve into ensuring user actions are logged accurately and action history is maintained properly. Validating these functions independently assures accurate data logging and seamless action history management.

- **Integration Testing:** Integration tests validate how action logging and action history management components interact. They ensure accurate logging data transmission and seamless updates across components.
- **Component Interface Testing:** Component interfacing tests validate how action logging and action history management components interact. They ensure accurate data transmission and seamless updates between components.

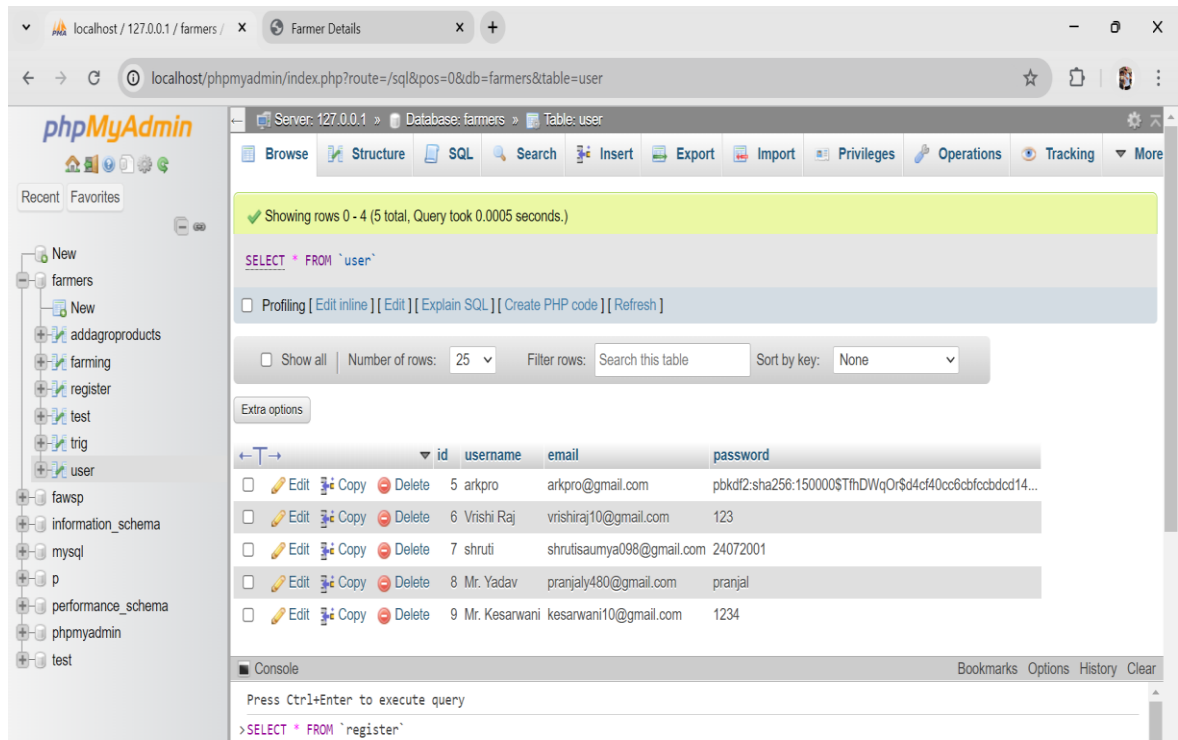


Fig 4.6: The Screenshot Image showing the Xampp software where in all the users who have registered, their id and password is stored in the database.

- **System Testing:** System testing validates the entire action logging and action history management processes. It ensures accurate logging of user actions and maintains a seamless action history.
- **Operational Acceptance Testing (OAT):** This phase validates scalability requirements and tests the usability of the action logging interface to ensure it meets user expectations.

Relationship Management Module:

- **Unit Testing:** Unit testing within this module aims to validate functions managing various relationships (user-product, user-action, product-farming).

By isolating these functions, you ensure relationship data is stored accurately and changes propagate correctly throughout the system.

- **Integration Testing:** Integration tests within this module assess how different relationship management components interact. This ensures that changes in one component are accurately communicated to others, maintaining data consistency.
- **Component Interface Testing:** Testing within this module ensures effective interaction between different relationship management components. This guarantees accurate communication of changes between components, upholding data integrity.

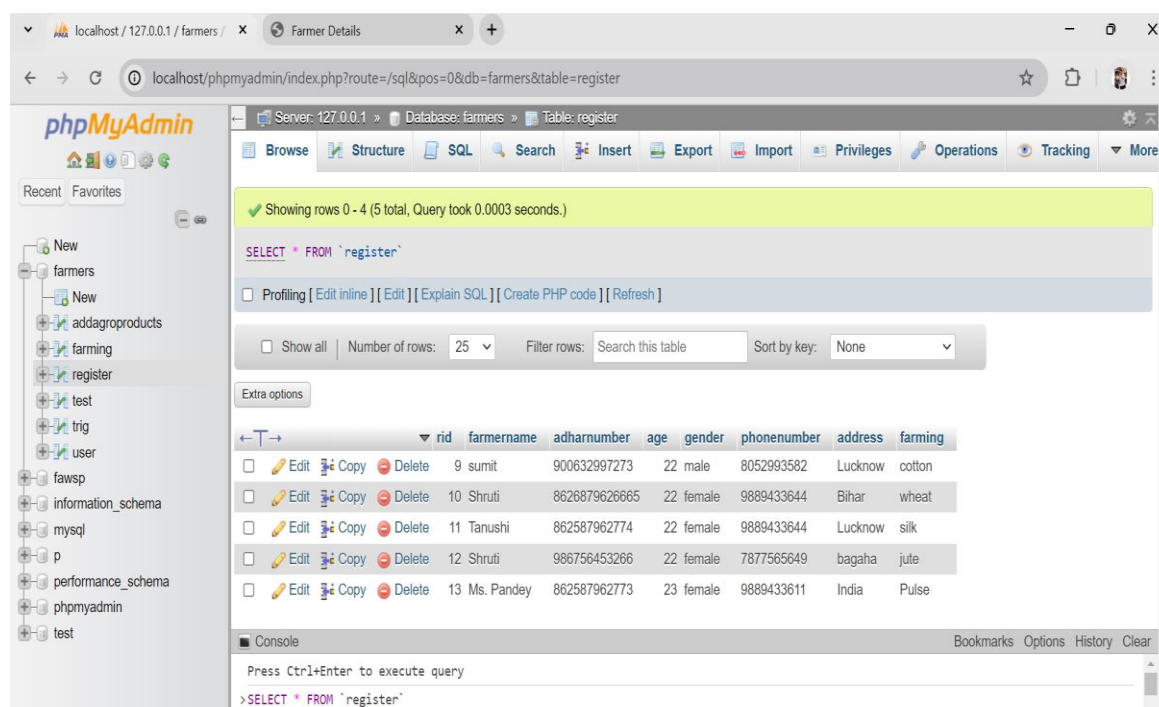


Fig 4.7: The Screenshot Image showing the Xampp software where in all the users who have registered, their id and password is stored in the database.

- **System Testing:** System testing within this module evaluates the complete relationship management process, guaranteeing accurate communication of changes and upholding data consistency.
- **Operational Acceptance Testing (OAT):** Operational acceptance testing ensures scalability requirements are met and tests the usability of the relationship management interface, ensuring it's intuitive for users.

4.2 SNAPSHOTS

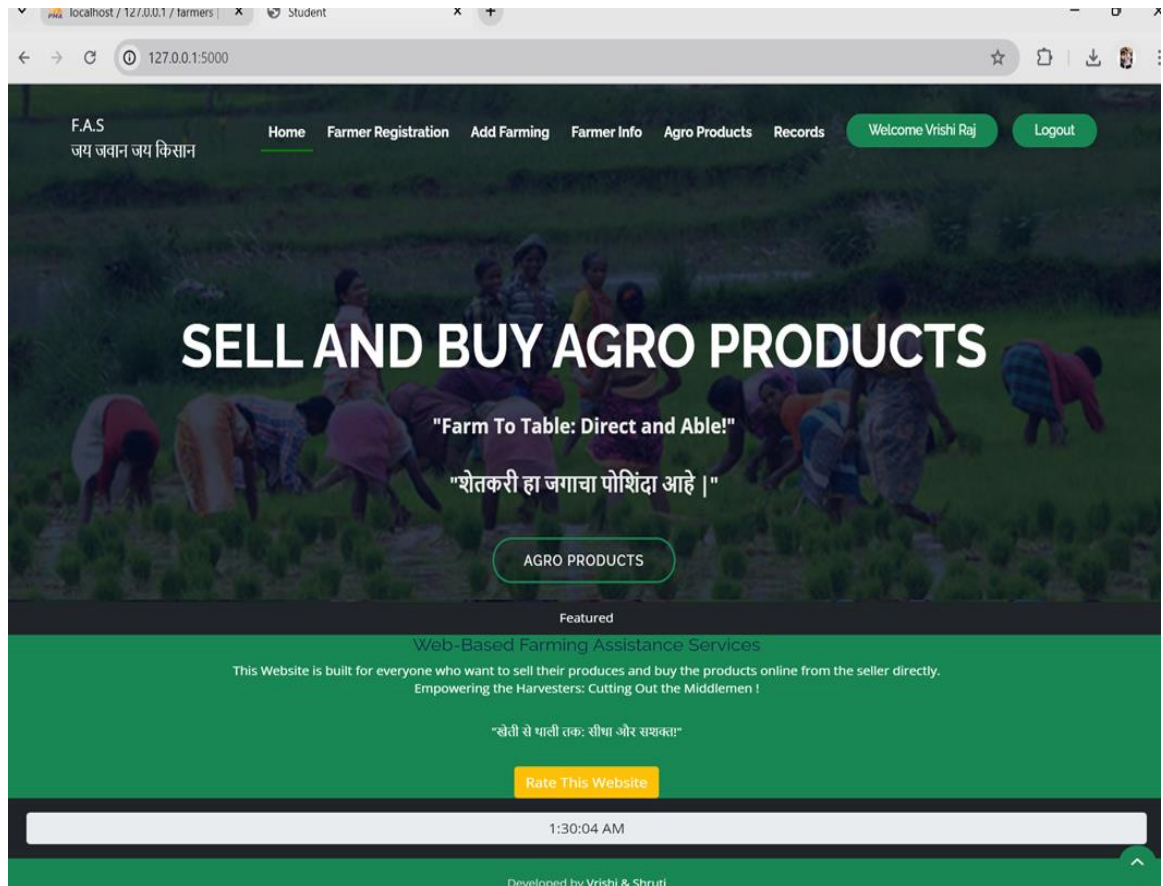


Fig 4.8: The Screenshot Image showing the Home Page of the Web-Based Farming Assistance Services.

A screenshot of the 'Register Farmer's Details' page. The page has a dark green header with the same navigation and user information as the home page. The main content area is white and contains a registration form with the following fields: 'Farmer's Name' (Shruti Saumya), 'Aadhar Number' (862587962756), 'Age' (22), 'Gender' (Female), 'Mobile Number' (7381022379), 'Address' (West Champaran, Bihar), and a product category 'jute'. A green 'Save Records' button is at the bottom.

Fig 4.9: The Screenshot Image showing the Farmer's Registration Page where all the information from the cultivator is collected for successful addition of the agriculturists.

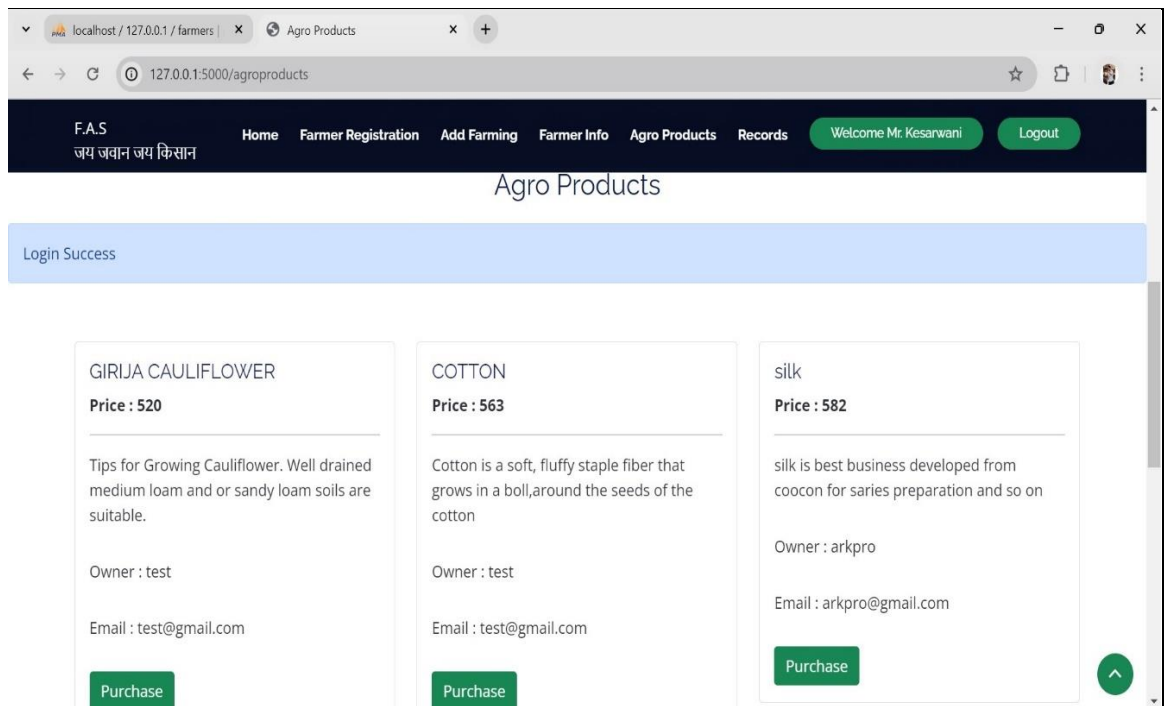


Fig 4.10: The Screenshot Image showing the Agronomists Produces listing along with its price and details of the product which are being put for sale by the agrarians for the purchase.

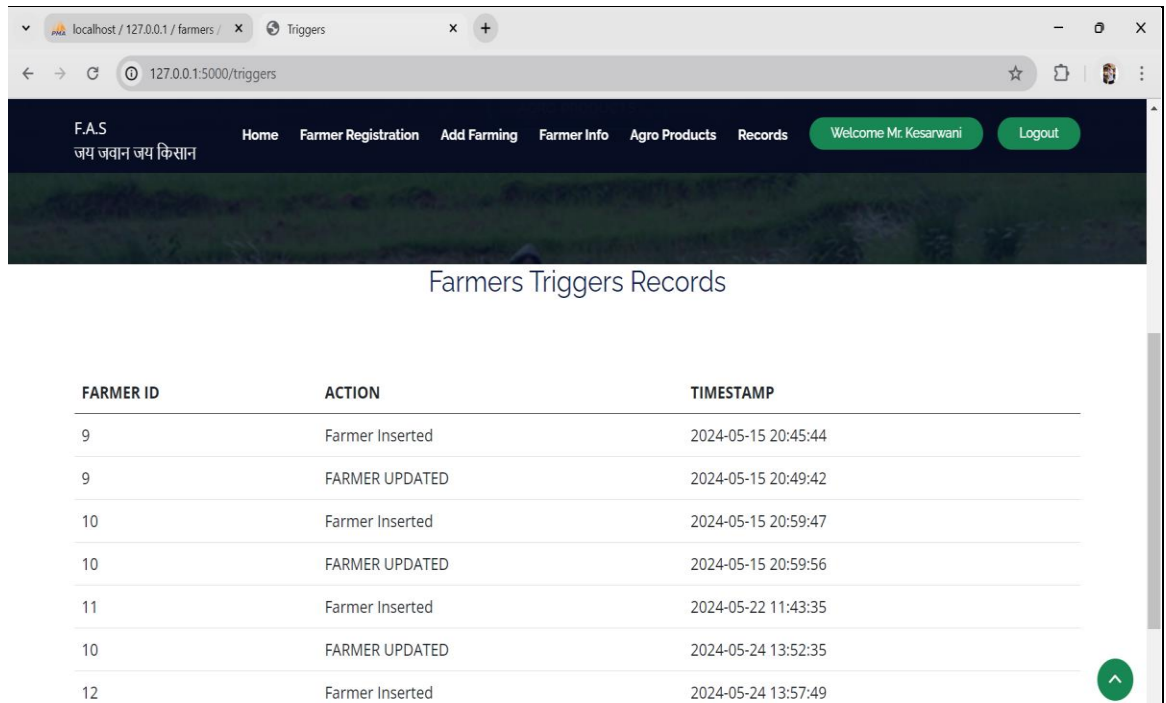


Fig 4.11: The Screenshot Image showing the Agrarians Trigger Records where the records which are inserted/updated/deleted, all are shown along with the timestamp and Id Number.

The screenshot shows a web browser window with the address bar displaying 'localhost / 127.0.0.1 / farmers / edit' and the URL '127.0.0.1:5000/edit/12'. The page title is 'Edit Farmer Details'. The form fields are as follows:

Field	Value
Farmer Name	Shruti
Aadhar Number	986756454266
Age	22
Gender	female
Mobile Number	7877565649
Address	bagaha
Crop	jute

At the bottom of the form is a button labeled 'Submit Record'.

Fig 4.12: The Screenshot Image showing the Page where the Cultivator's/User's detail can be edited if in case it is to be corrected.

4.3 INFERENCE

The "Farming Portal" project leverages Information and Communication Technologies (ICTs) to address critical challenges in the agricultural sector, particularly in regions like India where traditional supply chains often disadvantage farmers. The project aims to empower farmers by directly connecting them with consumers, thereby ensuring fair compensation for their produce and enhancing their economic stability. Here are the key inferences from the abstract and conclusion of the project:

- 1. Empowerment through ICT:** The project seeks to empower farmers by providing them with essential agricultural information and resources through a web-based platform. This information includes crop cultivation techniques, disease management strategies, market prices, and government schemes, which helps farmers make informed decisions and optimize crop yield.
- 2. Direct Market Access:** By creating a marketplace that connects farmers directly with consumers, the project aims to eliminate intermediaries, thus enabling farmers to

receive fair prices for their produce. This direct access to markets is expected to enhance farmers' profitability and economic stability.

3. Comprehensive Agricultural Support: The platform serves as a one-stop hub for various agricultural needs, including information dissemination, market access, and procurement of farming essentials. This holistic approach supports farmers in multiple aspects of their operations, from production to sales.

4. Economic Development: By fostering transparency and efficiency in the agricultural value chain, the project contributes to the overall development of the agricultural sector. It enhances food security and promotes equitable distribution of resources, benefiting not only farmers but also agricultural agencies and businesses.

5. Sustainability and Inclusivity: The project aims to create a more sustainable agricultural ecosystem by promoting practices that optimize resource use and reduce risks. Its inclusive design ensures that all farmers, regardless of their scale of operation, can benefit from the platform's features.

6. Transformative Potential: The "Farming Portal" has the potential to catalyze significant positive changes in agricultural communities by leveraging ICTs. It aims to create a more prosperous and sustainable future for farmers, contributing to broader socio-economic development.

Overall, the project underscores the transformative power of ICTs in agriculture, aiming to enhance farmers' livelihoods, streamline agricultural operations, and foster socio-economic development through innovative and inclusive digital solutions.

4.4 ADVANTAGES AND LIMITATIONS

Advantages

- The growers can access a wealth of information on various subjects at their fingertips.
- The portal empowers agriculturists by disrupting middlemen dominance.

- Informed decisions can result in increased productivity and profitability.
- The cultivators can connect with potential buyers, expanding their market reach and increasing sales opportunities, which can lead to better income and market stability.
- The cultivators can advertise their crops to showcase their products to potential suppliers.
- The producers can sell their products directly to the supplier without the involvement of the mediators.
- Farmers also have the option to post complaints, which will be addressed by the administration.
- It provides and works towards digital literacy.
- It provides efficient and scalable supply chain management feature.

Limitations

- The growers in remote or underserved areas may have limited internet access, hindering their ability to use the system effectively.
- Rural workers with low literacy levels or who speak languages not supported by the software may struggle to use it effectively.
- The cost of developing, deploying, and maintaining such a system, including hardware, software, and support, may be a barrier to widespread adoption.
- The digital divide can exacerbate disparities, as growers with limited access to equipment may be left behind.
- Too much data without proper guidance can lead to fraud in some cases, also making it challenging to extract actionable insights.

CHAPTER-5

CONCLUSION

The conclusion of the project proposal highlights both the promise and challenges of the "Farming Portal" system in India's agricultural sector. It emphasizes the transformative potential of the platform in challenging existing structures dominated by middlemen and enabling direct engagement between producers and consumers. The use of cutting-edge technologies is positioned as a means to empower farmers and ensure transparency in pricing for consumers. However, the conclusion also acknowledges significant hurdles, particularly in authentication and addressing the digital divide. Robust mechanisms are needed to protect user data and ensure secure transactions, while efforts to bridge disparities in digital literacy and internet access are crucial to prevent marginalization of certain farming communities.

Despite these challenges, the conclusion remains optimistic, presenting the "Farming Portal" as a beacon of hope for a more equitable, efficient, and resilient agricultural landscape in India. It emphasizes the need for concerted efforts to overcome authentication challenges and bridge the digital gap, underlining the platform's potential to revolutionize the agricultural paradigm, promote prosperity among growers, and enhance food security for future generations.

Importance:

- Disrupts middlemen dominance.
- Ensures transparent pricing.
- Empowers agriculturists.

Features:

- Secure authentication.
- Digital literacy initiatives.
- Facilitates direct producer-consumer interaction.
- Efficient supply chain management.
- Scalability and resilience.

CHAPTER-6

FUTURE SCOPE OF THE PROJECT

Here's an outline of the future scope of a farming assistance project incorporating AI and ML, blockchain, and agriculture:

AI and ML Integration:

Implementation of AI and ML algorithms for predictive analytics in agriculture, aiding in crop yield forecasting, pest detection, and disease diagnosis.

Development of smart farming solutions utilizing AI-powered drones and sensors for real-time monitoring of crops, soil health, and weather conditions.

Adoption of AI-driven decision support systems to provide personalized recommendations to farmers regarding crop selection, irrigation scheduling, and fertilization practices based on data analytics.

Utilization of machine learning algorithms for crop classification and weed identification, enabling targeted interventions and minimizing herbicide usage.

Blockchain Technology:

Integration of blockchain technology to establish transparent and immutable records of agricultural transactions, ensuring traceability and enhancing trust in the supply chain.

Implementation of smart contracts on blockchain platforms to automate agreements and payments between farmers, suppliers, and buyers, reducing administrative overhead and disputes.

Creation of digital identities for agricultural products using blockchain-based tracking systems, enabling consumers to verify the authenticity and quality of produce.

Facilitation of access to finance for farmers through blockchain-based lending platforms, leveraging digital assets and smart contracts to streamline loan approval processes and reduce risk for financial institutions.

AI, ML, and Blockchain Synergies:

Exploration of synergies between AI, ML, and blockchain technologies to develop advanced solutions for agricultural challenges, such as predictive modelling of market demand, supply chain optimization, and fraud detection.

Integration of blockchain-enabled data marketplaces where farmers can securely share agricultural data with AI and ML researchers, fostering innovation and knowledge exchange in the agricultural sector.

Development of AI-driven blockchain platforms for carbon trading and sustainable agriculture initiatives, incentivizing farmers to adopt environmentally friendly practices through tokenized rewards and carbon credits.

Capacity Building and Adoption:

Implementation of capacity-building programs to enhance farmers' understanding and adoption of AI, ML, and blockchain technologies, including training workshops, demonstration farms, and digital literacy initiatives.

Collaboration with agricultural extension services, research institutions, and technology companies to co-create and deploy AI, ML, and blockchain solutions tailored to the needs and contexts of smallholder farmers in diverse agricultural systems.

Policy and Regulatory Considerations:

Engagement with policymakers and regulators to address legal and regulatory challenges related to the use of AI, ML, and blockchain in agriculture, including data privacy, intellectual property rights, and interoperability standards.

Advocacy for supportive policies and incentives to encourage investment in AI, ML, and blockchain initiatives for agriculture, fostering innovation and digital transformation in the sector while ensuring equitable access and benefits for all stakeholders.

LIST OF FIGURES

Figure No.	Description	Page No.
Figure 3.1	Screenshot: Iterative Water Fall Model	7
Figure 3.2	The Block Diagram of Farming Assistance Services.	21
Figure 3.3	The Schema Diagram of the Platform	22
Figure 3.4	The Use Case Diagram of the Platform	23
Figure 3.5	DFD(0-level)	24
Figure 3.6	DFD(1-level)	24
Figure 3.7	Entity Relationship Diagram	25
Figure 4.1	Screenshot: New User Sign Up/Registration Page using id, user name and password.	42
Figure 4.2	Screenshot: New User Login Page upon successful registration.	43
Figure 4.3	Screenshot: Home Page with a Welcome Message	43
Figure 4.4	Screenshot: Database of Agricultural Products which are put for display for customers.	44
Figure 4.5	Screenshot: Registered Farmers Registration Detail where their details can be edited/deleted/added.	46
Figure 4.6	Screenshot: Xampp software showing registered users details stored in the database.	46
Figure 4.7	Screenshot: Database of the cultivators showing all details.	47
Figure 4.8	Screenshot: Home Page of Web-Based Farming Assistance Services	47

Figure 4.9	Screenshot: Agrarians Registration Page where all the information from them is collected for addition of the product.	48
Figure 4.10	Screenshot: Agronomists Produces listing along with its prices and details which are put sale.	48
Figure 4.11	Screenshot: Agrarians Trigger Records	49
Figure 4.12	Screenshot: Page where the Cultivators Detail can be edited if in case it is to be corrected.	49

LIST OF TABLES

Table No.	Description	Page No.
Table 3.1	Hardware Requirements of Web-Based Farming Assistance Services	17
Table 3.2	The Software Requirements of Web-Based Farming Assistance Services	18

LIST OF ABBREVIATIONS USED

IEEE	Institute of Electrical and Electronics Engineers
URL	Uniform Resource Locator
API	Application Programming Interface
HDD	Hard Disk Drive
RAM	Random Access Memory
ICT	Information and Communication Technologies

Coding Implementation:

farmer.sql:

```
-- phpMyAdmin SQL Dump
-- https://www.phpmyadmin.net/
-- Host: 127.0.0.1
-- Generation Time: Jan 20, 2021 at 06:31 AM
-- Server version: 10.4.11-MariaDB
-- PHP Version: 7.2.29

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";

START TRANSACTION;

SET time_zone = "+00:00";

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

-- Database: `farmers`

-- Table structure for table `addagroproducts`

CREATE TABLE `addagroproducts` (
  `username` varchar(50) NOT NULL,
  `email` varchar(50) NOT NULL,
  `pid` int(11) NOT NULL,
  `productname` varchar(100) NOT NULL,
  `productdesc` text NOT NULL,
  `price` int(100) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- Dumping data for table `addagroproducts`

INSERT INTO `addagroproducts` (`username`, `email`, `pid`, `productname`, `productdesc`, `price`) VALUES
('test', 'test@gmail.com', 1, 'GIRIJA CAULIFLOWER', ' Tips for Growing Cauliflower. Well drained medium loam and or sandy loam soils are suitable.', 520),
('test', 'test@gmail.com', 2, 'COTTON', 'Cotton is a soft, fluffy staple fiber that grows in a boll, around the seeds of the cotton ', 563),

-- Table structure for table `farming`
```

```

CREATE TABLE `farming` (
  `fid` int (11) NOT NULL,
  `farmingtype` varchar (200) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- Dumping data for table `farming`

INSERT INTO `farming` (`fid`, `farmingtype`) VALUES
(1, 'Seed Farming'),
(2, 'coccon'),
(3, 'silk');

-- Table structure for table `register`

CREATE TABLE `register` (
  `rid` int (11) NOT NULL,
  `farmername` varchar (50) NOT NULL,
  `adharnumber` varchar (20) NOT NULL,
  `age` int (100) NOT NULL,
  `gender` varchar (50) NOT NULL,
  `phonenummer` varchar (12) NOT NULL,
  `address` varchar (50) NOT NULL,
  `farming` varchar (50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- Triggers `register`

DELIMITER $$

CREATE TRIGGER `deletion` BEFORE DELETE ON `register` FOR EACH ROW INSERT INTO trig VALUES
(null,OLD.rid,'FARMER DELETED',NOW())

$$

DELIMITER;

DELIMITER $$

CREATE TRIGGER `insertion` AFTER INSERT ON `register` FOR EACH ROW INSERT INTO trig VALUES
(null,NEW.rid,'Farmer Inserted',NOW())

$$

DELIMITEM;

DELIMITER $$

```

```

CREATE TRIGGER `updater` AFTER UPDATE ON `register` FOR EACH ROW INSERT INTO trig VALUES
(null,NEW.rid,'FARMER UPDATED',NOW())

$$

DELIMITER;

-- Table structure for table `test`

CREATE TABLE `test` (

  `id` int(11) NOT NULL,

  `name` varchar(50) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- Dumping data for table `test`

INSERT INTO `test` (`id`, `name`) VALUES

(1, 'harshith');

-- Table structure for table `trig`

CREATE TABLE `trig` (

  `id` int(11) NOT NULL,

  `fid` varchar(50) NOT NULL,

  `action` varchar(50) NOT NULL,

  `timestamp` datetime NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- Dumping data for table `trig`

INSERT INTO `trig` (`id`, `fid`, `action`, `timestamp`) VALUES

(1, '2', 'FARMER UPDATED', '2024-05-19 23:04:44'),

(2, '2', 'FARMER DELETED', '2024-05-19 23:04:58'),

CREATE TABLE `user` (

  `id` int(11) NOT NULL,

  `username` varchar(50) NOT NULL,

  `email` varchar(50) NOT NULL,

  `password` varchar(500) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- Dumping data for table `user`

INSERT INTO `user` (`id`, `username`, `email`, `password`) VALUES

(5,'arkpro', 'arkpro@gmail.com','pbkdf2: sha256:150000$TfhDWqOr$d4cf40');

```

```

-- Indexes for dumped tables

-- Indexes for table `addagroproducts`

ALTER TABLE `addagroproducts`

  ADD PRIMARY KEY (`pid`);

-- Indexes for table `farming`

ALTER TABLE `farming`

  ADD PRIMARY KEY (`fid`);

-- Indexes for table `register`

ALTER TABLE `register`

  ADD PRIMARY KEY (`rid`);

-- Indexes for table `test`

ALTER TABLE `test`

  ADD PRIMARY KEY (`id`);

-- Indexes for table `trig`

ALTER TABLE `trig`

  ADD PRIMARY KEY (`id`);

-- Indexes for table `user`

ALTER TABLE `user`

  ADD PRIMARY KEY (`id`);

-- AUTO_INCREMENT for dumped tables

-- AUTO_INCREMENT for table `addagroproducts`

ALTER TABLE `addagroproducts`

  MODIFY `pid` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=4;

-- AUTO_INCREMENT for table `farming`

ALTER TABLE `farming`

  MODIFY `fid` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=4;

-- AUTO_INCREMENT for table `register`

ALTER TABLE `register`

  MODIFY `rid` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=9;

-- AUTO_INCREMENT for table `test`

ALTER TABLE `test`

  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=2;

```

```
-- AUTO_INCREMENT for table `trig`

ALTER TABLE `trig`

  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=6;

-- AUTO_INCREMENT for table `user`

ALTER TABLE `user`

  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=6;

COMMIT;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;

/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;

/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

Triggers.html:

```
{% extends 'base.html' %}

{% block title %}

Triggers

{% endblock title %}

{% block body %}

<h3 class="text-center"><span>Farmers Triggers Records</span> </h3>

{% with messages=get_flashed_messages(with_categories=true) %}

{% if messages %}

{% for category, message in messages %}

<div class="alert alert-{{ category }} alert-dismissible fade show" role="alert">

  {{ message }} </div>

{% endfor %} {% endif %}

{% endwith %} <br>

<div class="container mt-4">

<table class="table">

  <thead class="thead-light"><tr>

    <th scope="col">FARMER ID</th> <th scope="col">ACTION</th>

    <th scope="col">TIMESTAMP</th></tr> </thead> <tbody>

    {% for post in query %} <tr>

      <td>{{ post.fid }} </td> <td>{{ post.action }} </td> <td>{{ post.timestamp }} </td></tr>

    {% endfor %}
```



```

</tbody></table></div>

{% endblock body %}

agroproducts.html:

{% extends 'base.html' %} {% block title %}

Agro Products

{% endblock title %} {% block body %}

<h3 class="text-center"><span>Agro Products</span></h3>

{% with messages=get_flashed_messages(with_categories=true) %}

{% if messages %} {% for category, message in messages %}

<div class="alert alert-{{category}} alert-dismissible fade show" role="alert">

    {{message}} </div>

{% endfor %} {% endif %} {% endwith %}<br>

<div class="container mt-3"><div class="row">

    {% for p in query %} <div class="col-sm-4"><div class="card">

        <div class="card-body"><b><h5 class="card-title">{{p.productname}}</h5></b>

        <b>Price : {{p.price}}</b><hr><p class="card-text">{{p.productdesc}}</p>

        <p>Owner: {{p.username}}</p><p>Email: {{p.email}}</p>
<a href="https://mail.google.com/mail/?view=cm&fs=1&tf=1&to={{p.email}}" target="_blank" class="btn btn-
success ">Purchase</a> </div> </div></div>

    {% endfor %} </div></div>

{% endblock body %}

auth.html:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="utf-8">

    <meta content="width=device-width, initial-scale=1.0" name="viewport">

    <title> {% block title %}

    {% endblock title %} </title>

    <meta content="" name="description">

    <meta content="" name="keywords">

    {% block style %}

```

```

{%endblockstyle%}
<linkhref="https://fonts.googleapis.com/css?family=Open+Sans:300,300i,400,400i,700,700i|Raleway:300,400,500,700,800" rel="stylesheet">

<!-- Vendor CSS Files -->

<link href="static/assets/vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet">

<link href="static/assets/vendor/font-awesome/css/font-awesome.min.css" rel="stylesheet">

<link href="static/assets/vendor/owl.carousel/assets/owl.carousel.min.css" rel="stylesheet">

<link href="static/assets/vendor/aos/aos.css" rel="stylesheet">

<link href="static/assets/css/style.css" rel="stylesheet">

</head><body>

<header id="header">

<div class="container">

<div id="logo" class="pull-left">

<a href="/" class="scrollto">Farm Management</a></div></div>

</header><!-- End Header -->

<section id="intro">

<div class="intro-container" data-aos="zoom-in" data-aos-delay="100">

{% block content %}

{% endblock content %} </div>

</section><!-- End Intro Section -->

<div class="main">

<footer id="footer">

<div class="footer-top"><div class="container"></div><div class="container"><div class="credits"><br>

Developed by <a href="/">Vrishi and Shruti</a>

</div></div></footer><!-- End Footer -->

<a href="#" class="back-to-top"><i class="fa fa-angle-up"></i></a>

<script src="static/assets/vendor/jquery/jquery.min.js"></script>

<script src="static/assets/vendor/bootstrap/js/bootstrap.bundle.min.js"></script>

<script src="static/assets/vendor/jquery.easing/jquery.easing.min.js"></script>

<script src="static/assets/vendor/php-email-form/validate.js"></script>

<script src="static/assets/vendor/venobox/venobox.min.js"></script>

<script src="static/assets/vendor/owl.carousel/owl.carousel.min.js"></script>

<script src="static/assets/vendor/superfish/superfish.min.js"></script>

```

```

<script src="static/assets/vendor/hoverIntent/hoverIntent.js"></script>

<script src="static/assets/vendor/aos/aos.js"></script>

script src="static/assets/js/main.js"></script></body></html>

```

base.html:

```

<!DOCTYPE html>

<html lang="en">

<head> <meta charset="utf-8">

<meta content="width=device-width, initial-scale=1.0" name="viewport">

<title> {% block title %}

{% endblock title %} </title>

<meta content="" name="description">

<meta content="" name="keywords">

{% block style %}

{%endblockstyle%}

<linkhref="https://fonts.googleapis.com/css?family=Open+Sans:300,300i,400,400i,700,700i|Raleway:300,400,500,700,800" rel="stylesheet">

<!-- Vendor CSS Files -->

<link href="static/assets/vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet">

<link href="static/assets/vendor/venobox/venobox.css" rel="stylesheet">

<link href="static/assets/vendor/font-awesome/css/font-awesome.min.css" rel="stylesheet">

<link href="static/assets/vendor/owl.carousel/assets/owl.carousel.min.css" rel="stylesheet">

<link href="static/assets/vendor/aos/aos.css" rel="stylesheet">

<!-- Template Main CSS File -->

<link href="static/assets/css/style.css" rel="stylesheet"></head>

<body>

<header id="header">

<div class="container">

<div id="logo" class="pull-left">

<a href="/" class="scrollto">F.A. S</a></div>

<nav id="nav-menu-container">

<ul class="nav-menu">

<li class="{% block home %}

{% endblock home %}"><a href="/">Home</a></li>

```

```

<li><a href="/register">Farmer Register</a></li>

<li><a href="/addfarming">Add Farming</a></li>

<li><a href="/farmerdetails">Farmer Details</a></li>

<li><a href="/agroproducts">Agro Products</a></li>

<li><a href="/triggers">Records</a></li>

    {% if current_user.is_authenticated %}

        <li class="buy-tickets"><a href="">Welcome {{current_user.username}} </a></li>

        <li class="buy-tickets"><a href="/logout">Logout</a></li>

    {% else %} <li class="buy-tickets"><a href="/signup">Signin</a></li> {% endif %}

</ul> </nav><!-- #nav-menu-container --></div> </header>

<section id="intro">

    <div class="intro-container" data-aos="zoom-in" data-aos-delay="100">

        <h1 class="mb-4 pb-0">SELL AND BUY AGRO PRODUCTS </span> </h1>

        <a href="/agroproducts" class="about-btn scrollto">AGRO PRODUCTS</a></div>

    </section><!-- End Intro Section -->

<main id="main">

    {% block body %}

{% with messages=get_flashed_messages(with_categories=true) %}

{% if messages %}

{% for category, message in messages %}

<div class="alert alert-{{category}} alert-dismissible fade show" role="alert">

    {{message}}

</div>

{% endfor %} {% endif %}

{% endwith %} {% endblock body %}

<a href="#" class="back-to-top"><i class="fa fa-angle-up"></i></a>

<!-- Vendor JS Files -->

<script src="static/assets/vendor/jquery/jquery.min.js"></script>

<script src="static/assets/vendor/bootstrap/js/bootstrap.bundle.min.js"></script>

<script src="static/assets/vendor/jquery.easing/jquery.easing.min.js"></script>

<script src="static/assets/vendor/php-email-form/validate.js"></script>

<script src="static/assets/vendor/venobox/venobox.min.js"></script>

```

```

<script src="static/assets/vendor/owl.carousel/owl.carousel.min.js"></script>

<script src="static/assets/vendor/superfish/superfish.min.js"></script>

<script src="static/assets/vendor/hoverIntent/hoverIntent.js"></script>

<script src="static/assets/vendor/aos/aos.js"></script>

<!-- Template Main JS File --><script src="static/assets/js/main.js"></script></body></html>

```

edit.html:

```

<!doctype html>

<html lang="en">

  <head>   <meta charset="utf-8">

           <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

           <!--BootstrapCSS-->
<linkrel="stylesheet"href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css"integrity="sha384
84TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkGIXeMed4M0jlfDPv6uqKI2xXr2" crossorigin="anonymous">

           <title>Edit</title>

  </head><body class="bg-success"> <h3 class="text-center"><span>Edit Farmer Details</span> </h3>

  {% with messages=get_flashed_messages(with_categories=true) %}

  {% if messages %} {% for category, message in messages %}

  <div class="alert alert-{{category}} alert-dismissible fade show" role="alert">

    {{message}}</div>

    {% endfor %} {% endif %} {% endwith %}<br>

  <div class="container"> <div class="row"><div class="col-md-4"></div>

  <div class="col-md-4"><form action="/edit/{{posts.riid}}" method="post">

  <div class="form-group"> <label for="rollno">Farmer Name</label>

  <input type="text" class="form-control" name="farmername" id="farmername" value={{posts.farmername}}
  required></div><br>

  <div class="form-group"> <label for="adharnumber">Adhar Number</label>

  <input type="number" class="form-control" name="adharnumber" id="adharnumber"
  value={{posts.adharnumber}} required></div><br>

  <div class="form-group"><label for="age">Age</label>

  <input type="number" class="form-control" name="age" id="age" value={{posts.age}} required></div><br>

  <div class="form-group">

  <select class="form-control" id="gender" name="gender" required>

    <option selected>{{posts.gender}}</option>

    <option value="male">Male</option>

```

```

        <option value="female">Female</option> </select></div><br>

<div class="form-group">

<label for="num">Phone Number</label>

<input      type="number"      class="form-control"      name="phonenum"      id="phonenum"
value={{posts.phonenum}} required>

</div><br><div class="form-group">

<label for="address">Address</label>

<input class="form-control" name="address" id="address" value={{posts.address}} required/>

</div><br><div class="form-group">

<select class="form-control" id="farmingtype" name="farmingtype" required>

    <option selected>{{posts.farming}}</option>

    {% for d in farming %}

    <option value="{{d.farmingtype}}">{{d.farmingtype}}</option>

    {% endfor %} </select></div><br>

<button type="submit" class="btn btn-light btn-sm btn-block">Submit Record</button></form>

<br><br></div><div class="col-md-4"></div></div></div></div>

<!-- Optional JavaScript; choose one of the two! -->
<!--Option1:jQueryandBootstrapBundle(includesPopper)<scriptsrc="https://code.jquery.com/jquery3.5.1.slim.min.js
"integrity="sha384DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
crossorigin="anonymous"></script>

<scriptsrc="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/js/bootstrap.bundle.min.js"integrity="sha384-
ho+j7jyWK8fNQe+A12Hb8AhRq26LrZ/JpcUGGOn+Y7RsweNrtN/tE3MoK7ZeZDyx"
crossorigin="anonymous"></script>

<!--Option2:jQuery,Popper.js,andBootstrapJS
<scriptsrc="https://code.jquery.com/jquery3.5.1.slim.min.js"integrity="sha384DfXdz2htPH0lsSSs5nCTpuj/zy4C+
OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj" crossorigin="anonymous"></script>

<script      src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"      integrity="sha384-
9/reFTGAw83EW2RDu2S0VKalZap3H66lZH81PoYlFhbGU+6BZp6G7niu735Sk7lN"
crossorigin="anonymous"></script>

<script      src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/js/bootstrap.min.js"      integrity="sha384-
w1Q4orYjBQndcko6MimVbzY0tgp4pWB4lZ7lr30WKz0vr/aWKhXdBNmNb5D92v7s"
crossorigin="anonymous"></script> </body>

</html>

```

farmer.html:

```
{% extends 'base.html' %} {% block title %}
```

Register Farmers Details

```
{% endblock title %} {% block body %}
```

```

<h3 class="text-center"><span>Register Farmers Details</span> </h3>

{% with messages=get_flashed_messages(with_categories=true) %}

{% if messages %}

{% for category, message in messages %}

<div class="alert alert-{{category}} alert-dismissible fade show" role="alert">

    {{message}}</div>

{% endfor %} {% endif %}

{% endwith %}<br>

<div class="container">

<div class="row"><div class="col-md-4"></div>

<div class="col-md-4"><form action="/register" method="post">

<div class="form-group"><label for="rollno">Farmer Name</label>

<input type="text" class="form-control" name="farmername" id="farmername" required>

</div><br><div class="form-group">

<label for="adharnumber">Adhar Number</label>

<input type="number" class="form-control" name="adharnumber" id="adharnumber" required></div><br>

<div class="form-group"><label for="age">Age</label>

<input type="number" class="form-control" name="age" id="age" required></div><br><div class="form-group">

<select class="form-control" id="gender" name="gender" required>

    <option selected>Select Gender</option>

    <option value="male">Male</option>

    <option value="female">Female</option></select></div><br>

<div class="form-group">

<label for="num">Phone Number</label>

<input type="number" class="form-control" name="phonenumber" id="phonenumber" required></div><br>

<div class="form-group"><label for="address">Address</label>

<textarea class="form-control" name="address" id="address" required></textarea>

</div><br><div class="form-group">

<select class="form-control" id="farmingtype" name="farmingtype" required>

    <option selected>Select Farming</option>

    {% for d in farming %}

    <option value="{{d.farmingtype}}">{{d.farmingtype}}</option>

```

```

        {% endfor %} </select></div><br>

<button type="submit" class="btn btn-success btn-block">Save Records</button>

</form><br><br></div><div class="col-md-4"></div></div></div>

{% endblock body %}

farmerdetails.html:

{% extends 'base.html' %} {% block title %}

Farmer Details

{% endblock title %} {% block body %}

<h3 class="text-center"><span>Farmer Details</span> </h3>

{% with messages=get_flashed_messages(with_categories=true) %}

{% if messages %}

{% for category, message in messages %}

<div class="alert alert-{{category}} alert-dismissible fade show" role="alert"> {{message}} </div>

{% endfor %} {% endif %} {% endwith %}<br>

<div class="container mt-3"> <table class="table">

<thead class="thead-light"><tr>

<th scope="col">RID</th> <th scope="col">FARMER NAME</th>

<th scope="col">ADHAR NUMBER</th> <th scope="col">AGE</th>

<th scope="col">GENDER</th> <th scope="col">PHONE NUMBER</th>

<th scope="col">ADDRESS</th> <th scope="col">FARMING</th>

<th scope="col">EDIT</th> <th scope="col">DELETE</th>

<th scope="col">ADD AGRO PRODUCT</th></tr> </thead> <tbody>

{% for post in query %} <tr>

<th scope="row">{{post.rid}}</th>

<td> {{post. farmer name}} </td>

<td> {{post. adharnumber}} </td> <td>{{post. Age}} </td>

<td> {{post. gender}} </td> <td>{{post. phonenumber}} </td>

<td> {{post. address}} </td> <td>{{post.farming}} </td>

<td><a href="/edit/{{post.rid}}"><button class="btn btn-success">Edit </button> </a> </td>

<td><a href="/delete/{{post.rid}}"><button onclick="return confirm ('Are you sure to Delete data');" class="btn

btn-success">Delete </button> </a> </td>

<td><a href="/addagroproduct"><button class="btn btn-success">ADD </button> </a> </td></tr>

```



```
{% endfor %} </tbody></table></div>

{% endblock body %}

farming.html:

{% extends 'base.html' %} {% block title %}

Add Farming

{% endblock title %} {% block body %}

<h3 class="text-center bg-success text-white"><span>Add Farming</span> </h3>

{% with messages=get_flashed_messages(with_categories=true) %}

{% if messages %} {% for category, message in messages %}

<div class="alert alert-{{category}} alert-dismissible fade show" role="alert"> {{message}}</div>

{% endfor %} {% endif %}

{% endwith %} <br><div class="container"><div class="row">

<div class="col-md-4"></div><div class="col-md-4">

<form action="/addfarming" method="post"> <div class="form-group">

<label for="dept">Enter Farming Type</label>

<input type="text" class="form-control" name="farming" id="farming"></div><br>

<button type="submit" class="btn btn-success btn-sm btn-block">Add Farming</button>

</form><br><br></div> <div class="col-md-4"></div></div></div>

{% endblock body %}
```

Stored Procedure:

Routine name: proc

Type: procedure

Definition: Select * from register;

Triggers:

It is the special kind of stored procedure that automatically executes when an event occurs in the database.

Triggers used:

- Trigger name: on insert Table: register Time: after Event: INSERT INTO trig VALUES (null, NEW.rid, 'Farmer Inserted', NOW ())
- Trigger name: on delete Table: register Time: after Event: delete Definition: INSERT INTO trig VALUES (null, OLD. Rid, 'FARMER DELETED', NOW ())
- Trigger name: on update Table: register Time: after Event: update Definition: INSERT INTO trig VALUES (null, NEW.rid,'FARMER UPDATED', NOW ())

REFERENCES

- 1) Soko Manav Singhal, Kshitij Verma, Anupam Shukla, “Krishi Ville - Android based solution for Indian agriculture”. 2011 Fifth IEEE International Conference on Advanced Telecommunication Systems and Networks (ANTS), 18-21 Dec. 2011, Bangalore, India.
- 2) R. Sneha Iyer, R. Shruthi, K. Shruthi and R. Madhumathi, "Spry Farm: A Portal for Connecting Farmers and End Users," 2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 2021, pp. 429433, doi:10.1109/ICACCS51430.2021.9441815.
- 3) Shankar M. Patil, Monika Jadhav, Vishakha Jagtap, “Android Application for Farmers”, International Research Journal of Engineering and Technology, volume 6, issue 4, 2019, 4200-4202p.
- 4) Abishek A. G., Bharathwaj M., Bhagyalakshmi L. “Agriculture Marketing Using Web and Mobile Based Technologies”, 2016 IEEE International Conference on Technological Innovations in ICT For Agriculture and Rural Development (TIAR 2016).
- 5) D. Tilman C. Balzer J. Hill B.L. Befort Global food demand and the sustainable intensification of agriculture Proceedings of the National Academy of Sciences 108 50 2011 20260 20264
- 6) Babcock B 2013 *Cutting Waste in the Crop Insurance Program* ed N Bruzelius (Washington, DC: EnvironmentalWorkingGroup) (http://cdn.ewg.org/sites/default/files/u118/2013%20Cutting%20Crop%20Insurance%20Waste_.pdf?_ga=1.110206349.96985284.1437058644)
- 7) Md Iqbal, Vimal Kumar and Vijay Kumar Sharma. Krishi Portal: Web Based Farmer Help Assistance. International Journal of Advanced Science and Technology Vol. 29, No. 6, (2020), pp. 4783 – 4786
- 8) Agriculture marketing using web and mobile based technologies. 2016 IEEE Technological Innovations in ICT for Agriculture and Rural Development. doi:10.1109/tiar.2016.7801211
- 9) Vishi Purushottam Paliwal et al, “Design of Web Portal for ETrading for Farmers”, International Journal on Future Revolution in Computer Science and Communication Engineering, vol. 4, pp. 220-222, 2018.
- 10) Specification for security management systems for the supply chain, ISO 28000-2007,2007.[Online].Available:http://www.iso.org/iso/catalogue_detail?csnumber=44641 C. Speier, J. M. Whipple, D. J. Closs, and M. D. Voss, “Global supply chain design

considerations: Mitigating product safety and security risks,” J. Oper. Manage., vol. 29, pp. 721 736, 2011.

- 11) C. Speier, J. M. Whipple, D. J. Closs, and M. D. Voss, “Global supply chain design considerations: Mitigating product safety and security risks,” J. Oper. Manage., vol. 29, pp. 721 736, 2011.
- 12) Shital Chaudhari, Vaishnavi Mhatre, Pooja Patil, Sandeep Chavan, “Smart Farm Application: A Modern Farming Technique Using Android Application”, IJRET, Feb 2018.
- 13) <https://ieeexplore.ieee.org/document/9182969> A Study of Blockchain Technology in Farmer's Portal Sheetal Bhagwat et al, “Survey Paper on E-Mandi A Market Exchanging Between Farmers and End-user”, International Research Journal of Engineering and Technology, vol. 6, 2019.
- 14) <https://prepinsta.com/software-engineering/iterative-waterfall-model/>
- 15) <https://internetcomputer.org/>