

HARVEST HUB: A PLATFROM FOR AGRI-BUSINESS

FOURTH SEMESTER

MAIN PROJECT REPORT

Submitted to the University of Kerala

*In partial fulfillment of the requirements for the award of the
degree of*

Master of Computer Applications

Submitted By

Naveena Ninan Sam (95522821037)



Department of Computer Applications

SREE NARAYANA INSTITUTE OF TECHNOLOGY

Vadakkevila, Kollam-10

www.snit.ac.in

2024

SREE NARAYANA INSTITUTE OF TECHNOLOGY

Vadakkevila, Kollam-10

Department of Computer Applications

Certificate

Certified that the project entitled

“HARVEST HUB”

Is a bonafide report of the project

done by

Naveena Ninan Sam (95522821037)

During the year 2024 in partial fulfillment of the requirements

for the award of Master of Computer Applications

by University of Kerala

Examiner

Guide

Head of Department

Prof.Rajitha P.R

Dr.Sajeev J

DECLARATION

I, Naveena Ninan Sam, hereby declare that this project report entitled 'Harvest Hub' is the bonafide work carried out under the supervision of Mrs.Rajitha P.R, Asst. Professor, Sree Narayana Institute of Technology, Kollam. Declared further, that to the best of my knowledge, the work reported here does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion to any other candidate. The content of this report is not being presented by any other student to this or any other University for the award of a degree.

Signature

Naveena Ninan Sam (Reg.No.95522821037)

Countersigned

Head of Department, Master of Computer Applications

Sree Narayana Institute of Technology, Kollam

ACKNOWLEDGEMENT

First and foremost, I thank the Almighty God who gave me the knowledge and strength to complete this project successfully. I express my deep sense of gratitude to The Principal Dr.T.Mahalekshmi and Dr.Sajeev.J, Head, Department of Computer Applications for providing me with all the necessary facilities, which helped me a lot in the successful completion of this project. It is a great pleasure for me to acknowledge the assistance of my project guide Mrs.Rajitha P.R, Assistant Professor, Sree Narayana Institute of Technology, Kollam, for her valuable advice and kind support during the period of this project.

Naveena Ninan Sam (Reg.No.95522821037)

TABLE OF CONTENTS

CHAPTERNO	TITLE	PAGENO:
	ABSTRACT	i
	LIST OF TABLES	ii
	LIST OF FIGURES	iii
	LIST OF ABBREVIATIONS	iv
1	INTRODUCTION	1
	1.1 OVERVIEW OF THE PROJECT	1
	1.2 LITERATURE REVIEW	2
	1.2.1 E-Commerce platform and sustainability	2
	1.2.2 Market Dynamics	2
	1.2.3 The MERN Stack for E-Commerce	2
	1.2.4 User Experiences and Design in E-commerce	2
	1.2.5 Role and Impact	3
	1.2.6 User Adoption and Satisfaction	3
	1.2.7 Advantages & Disadvantages of MERN Stack	3
	1.3 PROPOSED SYSTEM OBJECTIVES & SCOPES	3-5
	1.3.1 Objectives of Project	4-5
	1.3.2 Scope of the Project	5
2	REQUIREMENTS SPECIFICATION	6
	2.1 OVERALL DESCRIPTION	6
	2.1.1 Functional Requirements	6-7
	2.2 PRODUCT PERSPECTIVE	7-8
	2.3 PRODUCT FUNCTIONS	8-9

2.3.1	User Authentication & Management	8
2.3.2	Marketplace & product listing	8
2.3.3	Order Management	8
2.3.4	Payment Gateway Integration	8
2.3.5	Inventory Management	8
2.3.6	Reviews & Ratings	8-9
2.3.7	Mobile Responsiveness	9
2.3.8	Security & Compliance	9
2.4	USER CHARACTERISTICS	9-10
2.4.1	System Administrator	9
2.4.2	Farmers	9
2.4.3	Users	9-10
2.5	OPERATING ENVIRONMENT CONSTRAINTS	10-11
2.5.1	Software Specification	10-11
2.5.2	Hardware Specification	11
2.6	SPECIFIC REQUIREMENTS	11-13
2.6.1	Functional Requirements	11
2.6.2	Non-Functional Requirements	12
2.6.3	Technical Requirements	12
2.6.4	Regulatory Requirements	12-13
2.7	EXTERNAL INTERFACE REQUIREMENTS	13-14
2.8	SYSTEM FEATURES	14-15
3	SYSTEM DESIGN AND TEST PLAN	16
3.1	USE CASE DIAGRAM	16-19
3.1.1	Admin Use Case Diagram	16-17
3.1.2	Farmer Use Case Diagram	17-18
3.1.3	User Use Case Diagram	18-19
3.2	DATA FLOW DIAGRAM	19

3.2.1 Level 0 DFD	20
3.2.2 Level 1 DFD	20-21
3.3 TABLE DESIGNS	22-26
3.3.1 Categories Table	22
3.3.2 Login Table	22
3.3.3 Register Table	23
3.3.4 Product Table	23
3.3.5 Feedback Table	24
3.3.6 Cultivation Table	24
3.3.7 Cart Table	25
3.3.8 Complaints Table	25
3.3.9 Review Table	26
3.4 PERFORMANCE REQUIREMENTS	26-27
3.5 SOFTWARE QUALITY ATTRIBUTES	27
3.6 DETAILED DESIGN	27-31
3.6.1 System Architecture	28
3.6.2 Frontend Design	28
3.6.3 Backend Design	29
3.6.4 Database Design	29
3.6.5 Security Measures	29-30
3.6.6 Testing Strategy	30-31
3.7 SAMPLING METHODS	31-32
3.7.1 Random Sampling	31
3.7.2 Stratified Sampling	31
3.7.3 Cluster Sampling	32
3.7.4 Convenience Sampling	32
3.7.5 Systematic Sampling	32

	3.8 TEST CASES	33-36
	3.8.1 Test Case for Admin	33-34
	3.8.2 Test Case for Farmer	34-35
	3.8.3 Test Case for User	36
4	IMPLEMENTATION AND RESULTS	37-40
	4.1 Implementation and Results	37
	4.2 Implementation Results	37-40
5	CONCLUSION AND FUTURE WORKS	41-42
	5.1 CONCLUSION	41
	5.2 FUTURE WORK	41-42
	5.2.1 Future Implementation Plan	42
6	REFERENCES	43-44
7	APPENDIX	44-71
	Appendix-A Screenshots	45-55
	Appendix-B Sample Code	56-71

ABSTRACT

The MERN-based Harvest Hub Portal revolutionizes the agricultural market by providing a platform that empowers farmers to directly connect with customers, eliminating the need for intermediary sellers. This innovative system is designed to facilitate a seamless and transparent exchange, fostering a direct link between farmers and consumers.

The portal introduces a user-friendly interface for farmers to showcase their products, including fruits, vegetables, and other agricultural goods. Customers, in turn, can explore a diverse range of locally sourced produce and make direct purchases from the farmers. The system incorporates features that allow farmers to provide detailed information about their products, including cultivation methods, freshness, and harvesting practices, enabling customers to make informed decisions.

Farmers can manage their product listings, update availability, and set prices, while customers can easily browse, add items to their carts, and make secure online payments directly to the farmers. This direct transaction model not only ensures fair compensation for farmers but also promotes community-driven agriculture.

The Harvest Hub emphasizes transparency and accountability, incorporating a review and rating system. Customers can share feedback on the quality of products and overall experiences, creating a trustworthy community and facilitating a symbiotic relationship between farmers and consumers.

With a responsive design, the portal ensures accessibility across various devices, making it convenient for both farmers and customers to engage in direct sales. The project aims to empower local farmers, foster sustainable agricultural practices, and redefine the way agricultural products reach consumers by providing a direct and efficient channel for transactions.

LIST OF TABLES

TABLE NO:	TITLE	PAGE NO
3.3.1	Categories Table	22
3.3.2	Login Table	22
3.3.3	Register Table	23
3.3.4	Product Table	23
3.3.5	Feedback Table	24
3.3.6	Cultivation Table	24
3.3.7	Cart Table	25
3.3.8	Complaints Table	25
3.3.9	Review Table	26

LIST OF FIGURES

FIGURE NO:	TITLE	PAGE NO:
3.1.1	Admin Use Case Diagram	17
3.1.2	Farmer Use Case Diagram	18
3.1.2	Users Use Case Diagram	19
3.2.1	Level 0 DFD	20
3.2.2	Level 1 DFD	21

LIST OF ABBREVIATION

List	Abbreviations	Meaning
1	MERN	MongoDB, Express.js, React, Node.js
2	API	Application Programming Interface
3	JWT	JSON Web Token
4	CRUD	Create, Read, Update, Delete
5	UX	User Experience
6	UI	User Interface
7	SSL	Secure Sockets Layer
8	CSS	Cascading Style Sheets
9	HTML	Hyper Text Markup Language

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW OF THE PROJECT

A Harvest Hub is an online platform specifically designed to cater to the needs of individuals and organizations operating within the agricultural sector. These portals serve as digital hubs where farmers, agribusinesses, suppliers, buyers, investors, and other stakeholders can converge to access a wide array of agricultural-related information, services, and opportunities. Through an agricultural business portal, users can engage in various activities such as buying and selling agricultural products and services. The portal introduces a user-friendly interface for farmers to showcase their products, including fruits, vegetables, and other agricultural goods. Customers, in turn, can explore a diverse range of locally sourced produce and make direct purchases from the farmers. The system incorporates features that allow farmers to provide detailed information about their products, including cultivation methods, freshness, and harvesting practices, enabling customers to make informed decisions. Farmers can manage their product listings, update availability, and set prices, while customers can easily browse, add items to their carts, and make secure online payments directly to the farmers. This direct transaction model not only ensures fair compensation for farmers but also promotes community-driven agriculture. The Harvest Hub emphasizes transparency and accountability, incorporating a review and rating system. Customers can share feedback on the quality of products and overall experiences, creating a trustworthy community and facilitating a symbiotic relationship between farmers and consumers. With a responsive design, the portal ensures accessibility across various devices, making it convenient for both farmers and customers to engage in direct sales. The project aims to empower local farmers, foster sustainable agricultural practices, and redefine the way agricultural products reach consumers by providing a direct and efficient channel for transactions. By leveraging digital technology, these portals contribute to the efficiency, productivity, and sustainability of the agricultural sector, ultimately benefiting farmers, businesses, and consumers alike.

1.2 LITERATURE REVIEW

1.2.1 E-commerce Platforms and Sustainability:

This research area examines how different E-commerce platforms address agricultural and sustainability concerns. Researchers would analyze various platforms, including Harvest Hub, to assess their features, user experience, and effectiveness in promoting user-friendly products. A comparative study would delve into the strengths and weaknesses of each platform, considering factors such as product selection, transparency of product information.

1.2.2 Market Dynamics:

These portals enable transparent and efficient transactions by providing real-time pricing information, reducing information asymmetry, and enabling direct interactions between buyers and sellers. Additionally, they may foster competition by allowing a diverse range of suppliers to access the market, thereby benefiting both producers and consumers.

1.2.3 The MERN Stack for E-Commerce

This topic explores the utilization of MongoDB, Express.js, React, and Node.js in developing Ecommerce platforms, with a focus on sustainability-oriented market places like Harvest Hub. Researchers would assess the advantages and challenges of using the MERN stack for building such platforms. Advantages may include scalability, real-time updates, and a seamless user experience, while challenges could involve complexity in implementation and potential performance bottlenecks. Case studies and technical analyses could provide valuable insights.

1.2.4 User Experience and Design in E-Commerce

The importance of user experience (UX) in e-commerce cannot be overstated. A study by Thakur (2018) emphasizes that a well-designed and intuitive interface significantly enhances customer satisfaction and retention. React's component-based architecture allows for the creation of reusable UI components, which can lead to a consistent and streamlined user experience across various devices. The focus on mobile responsiveness ensures that the platform caters to the growing number of users who shop via smartphones and tablets (Thakur, 2018).

1.2.5 Role and Impact

Research delves into the significant role played by agribusiness portals in driving agricultural development, enhancing market efficiency, and fostering economic growth. These portals contribute to improving access to markets by connecting farmers directly with buyers and reducing transaction costs associated with traditional supply chains.

1.2.6 User Adoption and Satisfaction:

Studies examine the factors influencing the adoption and usage of agribusiness portals among different user groups, including farmers, agribusinesses, government agencies, and consumers. These factors may include technological literacy, access to internet infrastructure, perceived usefulness and ease of use, and trust in the platform. Understanding user satisfaction levels and addressing usability issues are crucial for enhancing the effectiveness and sustainability of agribusiness portals.

1.2.7 Advantages and Disadvantages of the MERN Stack

The MERN stack offers several advantages, including the use of a single programming language (JavaScript) across the entire application, which simplifies development and maintenance (Tilkov&Vinoski, 2010). The open-source nature of these technologies makes them cost-effective and accessible, with strong community support and extensive documentation (Brown, 2020). However, the learning curve for mastering the full stack can be steep, particularly for developers who are not already proficient in JavaScript. Additionally, while Node.js excels in handling I/O-bound tasks, it may not be as efficient for CPU-intensive operations compared to other server-side technologies like Python or Java (Tilkov & Vinoski, 2010).

1.3 PROPOSED SYSTEM

Harvest Hub utilizes MongoDB, Express.js, React, and Node.js to create a robust, scalable, and efficient e-commerce platform. This platform empowers farmers to showcase and sell their products directly to consumers, eliminating intermediaries. A user-friendly interface allows farmers to provide detailed information about their produce, fostering an informed buying process. Customers can explore a variety of locally sourced goods, add items to their carts, and

make secure online payments directly to the farmers. The proposed system emphasizes transparency with features like product reviews and ratings, creating a trustworthy community. With a responsive design, the portal ensures accessibility, making it a convenient and direct channel for farmers and consumers.

1.3.1 Objectives of project

An Harvest Hub serves as a multifaceted digital platform designed to streamline and enhance various aspects of the agricultural industry. Its primary objectives include facilitating market access for agricultural products, both locally and globally, by connecting farmers, producers, distributors, and buyers. Ultimately, the agricultural business portal aims to empower the agricultural community, enhance productivity, and contribute to economic growth in rural areas.

The key objectives of the Harvest Hub project are as follows:

- **Eliminate Intermediaries:** Remove the dependency on intermediary sellers, empowering farmers to engage directly with consumers, ensuring fair compensation, and maximizing profits for agricultural produce.
- **Enhance Transparency and Accountability:** Introduce a review and rating system to foster a trustworthy community, promoting transparency in product quality and overall experiences. This builds confidence among consumers and strengthens the symbiotic relationship between farmers and customers.
- **Empower Local Farmers:** Provide a user-friendly platform for farmers to showcase their products, manage listings, and directly control pricing and availability. This empowerment enhances the economic standing of local farmers, encouraging community-driven agriculture.
- **Promote Sustainable Practices:** Encourage sustainable agricultural practices by showcasing locally sourced produce. By connecting consumers with the source of their food, the project aims to raise awareness about the environmental impact of their choices.

- **Efficient Transactions:** Enable secure online transactions directly between farmers and customers, streamlining the purchase process and ensuring a seamless experience for both parties.
- **Community-Driven Agriculture:** Foster a sense of community by connecting consumers with the individuals producing their food. This not only supports local economies but also strengthens the bond between farmers and their customer base.

1.3.2 Scope of the project

The scope of the Harvest Hub project encompasses the development of a comprehensive and user-friendly e-commerce platform dedicated to eco-friendly products using the MERN stack (MongoDB, Express.js, React, Node.js). The scope of an Harvest Hub project encompasses a comprehensive array of activities and objectives aimed at enhancing every facet of agricultural operations. It begins with optimizing production and farming techniques, employing modern practices and technologies to increase yields, improve crop quality, and efficiently manage livestock. Market analysis and strategy formulation play a crucial role in identifying consumer demand trends, market opportunities, and effective marketing channels to promote agricultural products effectively. Supply chain management becomes pivotal for ensuring seamless logistics, distribution, and inventory control from input suppliers to end consumers, thereby optimizing operational efficiency and customer satisfaction. Financial planning and management are essential components to ensure profitability and sustainable growth, involving budgeting, forecasting, and effective financial resource allocation. Emphasizing sustainability, agri-business projects prioritize eco-friendly farming practices, resource conservation, and compliance with environmental regulations to minimize ecological footprint and ensure long-term viability. Overall, the scope of an harvest hub project spans a holistic approach, integrating diverse disciplines and strategies to achieve sustainable growth, profitability, and positive socio-economic impact within the agricultural sector.

CHAPTER 2

REQUIREMENT SPECIFICATION

2.1 OVERALL DESCRIPTION

The Harvest Hub project is an innovative e-commerce platform designed to facilitate the purchase of agricultural products, addressing the growing demand for sustainable shopping options. Built on the MERN stack (MongoDB, Express.js, React, Node.js), the platform combines a robust backend with an intuitive frontend to create a seamless user experience. An agricultural business portal is an online platform specifically designed to cater to the needs of individuals and organizations operating within the agricultural sector. The portal introduces a user-friendly interface for farmers to showcase their products, including fruits, vegetables, and other agricultural goods. Customers, in turn, can explore a diverse range of locally sourced produce and make direct purchases from the farmers that is this portal serves as digital hubs where farmers, agribusinesses, suppliers, buyers, investors, and other stakeholders can converge to access a wide array of agricultural-related information, services, and opportunities. The Harvest Hub emphasizes transparency and accountability, incorporating a review and rating system. Customers can share feedback on the quality of products and overall experiences, creating a trustworthy community and facilitating a symbiotic relationship between farmers and consumers. Overall, agricultural business portals play a vital role in facilitating communication, collaboration, and commerce within the agricultural community. By leveraging digital technology, these portals contribute to the efficiency, productivity, and sustainability of the agricultural sector, ultimately benefiting farmers, businesses, and consumers alike.

2.1.1. Functional Requirements:

- **Farmers registration and approve:** Admins should be able to approve and reject registered farmers
- **User Account Management:** Users should be able to create, edit, and delete their accounts, manage passwords, and view order history.

- **Product Catalog Management:** The platform must allow administrators to add, edit, and remove products, including details such as descriptions, images, prices, and eco-friendly attributes.
- **Reviews and Ratings:** Implement a system for buyers to leave reviews and ratings for products and sellers. This helps build trust and credibility within the marketplace, influencing purchase decisions and fostering a community-driven approach to quality assurance.
- **Shopping Cart:** Users must have the ability to add, remove, and modify items in their shopping cart before proceeding to checkout.
- **Checkout Process:** The system should facilitate a secure and efficient checkout process, including payment processing and order confirmation.
- **Complaints Management:** Admins have the ability to effectively manage the issues raised by users.

2.2 PRODUCT PERSPECTIVE

A MERN-based Harvest Hub project represents a sophisticated platform designed to revolutionize the agricultural sector by leveraging modern technology to address various challenges and opportunities. At its core, the project aims to create a comprehensive marketplace and e-commerce platform where farmers can list their produce and agricultural products for sale, and buyers ranging from wholesalers to end consumers can easily browse, purchase, and track orders. This functionality not only streamlines transactions but also enhances market access and transparency in pricing and availability. The user interface is crafted to be intuitive and visually appealing, ensuring that customers can easily navigate the site, access detailed information about each item's sustainable attributes. The project aims to empower local farmers, foster sustainable agricultural practices, and redefine the way agricultural products reach consumers by providing a direct and efficient channel for transactions. By leveraging digital technology, these portals contribute to the efficiency, productivity, and sustainability of the agricultural sector, ultimately benefiting farmers, businesses, and consumers alike. A robust feedback and rating system enhances transparency and credibility within the marketplace, encouraging trust among users. Scalability and customization are key considerations, allowing the platform to grow alongside user needs and

adapt to regional agricultural practices and market dynamics. Data security measures are paramount to safeguard user information and transactions, complying with stringent data protection regulations to build trust and ensure privacy.

2.3 PRODUCT FUNCTIONS

The Harvest Hub platform offers several key functions designed to enhance the user experience and facilitate sustainable shopping:

- 2.3.1. User Authentication and Management:** Implement robust user authentication and management systems to securely handle farmer, buyer, and admin accounts. This includes features for registration, login, profile management, and role-based access control.
- 2.3.2. Marketplace and Product Listings:** Develop a marketplace where farmers can create detailed listings for their agricultural products. Include fields for product descriptions, images, pricing, quantities, and relevant tags (e.g., organic, local). Implement search and filter functionalities to help buyers easily find desired products.
- 2.3.3. Order Management:** Provide comprehensive order management capabilities for both farmers and buyers. Farmers should be able to view incoming orders, manage order statuses (e.g., pending, fulfilled), and generate invoices. Buyers should have options to track their orders, communicate with sellers, and provide feedback.
- 2.3.4. Payment Gateway Integration:** Integrate secure payment gateways to facilitate online transactions. Support various payment methods (e.g., credit/debit cards, digital wallets) and ensure compliance with payment industry standards for secure transactions.
- 2.3.5. Inventory Management:** Enable farmers to manage their inventory effectively by tracking stock levels, updating quantities as products are sold, and receiving notifications for low stock levels. Implement alerts and reminders for farmers to restock or update their listings.
- 2.3.6. Reviews and Ratings:** Implement a system for buyers to leave reviews and ratings for products and sellers. This helps build trust and credibility within the marketplace,

influencing purchase decisions and fostering a community-driven approach to quality assurance

2.3.7. Mobile Responsiveness: Ensure the platform is responsive and accessible on mobile devices through responsive web design and/or dedicated mobile applications. This caters to users who prefer to access the platform on smartphones or tablets, especially in rural areas with limited access to desktop computers.

2.3.8. Security and Compliance: Implement strong security measures to protect user data, transactions, and communications. Adhere to data protection regulations and industry standards to ensure confidentiality, integrity, and availability of information stored on the platform.

2.4. USER CHARACTERISTICS

2.4.1. System Administrators:

Administrators manage and maintain the overall system to ensure smooth operation. Admin ensure the platform is regularly updated with the latest features and security patches. Administrators have the capability to approve registrations from new farmers. They can also add, edit, or remove farmer profiles as needed, maintaining an updated database of active participants. Monitoring and addressing issues that arise within the platform. Administrators provide technical support to users and address any issues promptly.

2.4.2. Farmers:

Farmers can showcase their agricultural products on the platform, providing detailed information such as cultivation methods, freshness, and harvesting practices. Farmers can update the availability of their products and manage inventory in real-time. Farmers have the ability to set prices for their products. Farmers handle direct transactions with customers, including receiving online payments securely.

2.4.3. Users:

Consumers can explore a diverse range of locally sourced agricultural products showcased by farmers. They also have access to detailed information about products, including cultivation methods, freshness, and harvesting practices. They can add items to their carts

for convenient purchasing. Consumers can make secure online payments directly to farmers.

2.5 OPERATING ENVIRONMENT CONSTRAINTS

The operating environment constraints for the Harvest Hub platform encompass several factors that may impact its functionality and performance. Harvest Hub faces several critical considerations as an eco-conscious e-commerce platform. First, ensuring stable internet connectivity is essential for optimal user experience, minimizing potential disruptions during shopping sessions. Compatibility across various devices and operating systems, accommodating different screen sizes and resolutions, is crucial to enhance accessibility and user satisfaction. Adherence to stringent security regulations is paramount to safeguard user data through encryption and secure payment processing methods. While designed for scalability, monitoring server capacity and infrastructure is necessary to manage potential constraints from high traffic or increased data storage requirements. Keeping abreast of technological advancements in e-commerce and web development is crucial for maintaining competitiveness and operational efficiency. Adapting to market competition from other eco-friendly platforms requires continuous innovation in product offerings and pricing strategies. Educating users on the benefits of agricultural products and providing effective navigation support fosters engagement and enhances user experience. Offering a diverse range of payment options addresses regional disparities in payment gateways, ensuring seamless transactions. Additionally, prioritizing environmental sustainability involves minimizing energy consumption and adopting eco-friendly server hosting practices, aligning with Harvest Hub's core mission. These factors collectively shape Harvest Hub's strategy to deliver a secure, accessible, and environmentally responsible shopping experience for its users.

2.5.1. Software Specification

Software requirements are technical specifications for software products. The goal of software requirements definition is to completely and consistently specify the technical requirements for the software product concisely and unambiguously.

Operating System : Windows 10 and above

Web Browser	:	Any Standard web browser
Front End	:	React.js
Back End	:	Node.js
Database	:	MongoDB

2.5.2. Hardware Specification

The hardware requirement is the minimum required specification for the system to work.

Processor	:	Intel core i3
RAM	:	8GB
Hard Disk	:	Volume 1

2.6. SPECIFIC REQUIREMENTS

2.6.1. Functional Requirements:

The system described offers a comprehensive suite of features designed to enhance the functionality and user experience of an online marketplace. Admins have the capability to oversee and manage farmer registrations, approving or rejecting applications based on set criteria. For user account management, individuals can create, modify, and delete their accounts, as well as handle password changes and review their order histories. Product catalog management is streamlined for administrators, allowing them to add, update, or remove products while managing details such as descriptions, images, prices, and eco-friendly attributes. A robust reviews and ratings system enables buyers to leave feedback on products and sellers, fostering a trustworthy and community-driven marketplace. The shopping cart functionality permits users to effortlessly add, remove, or modify items before checkout. The checkout process is designed to be secure and efficient, handling payment processing and order confirmation seamlessly. Lastly, the complaints management feature empowers admins to address and resolve issues raised by users effectively, ensuring a responsive and supportive platform.

2.6.2. Non-Functional Requirements:

The Farmer's Direct Sales Portal is designed with critical performance, security, usability, scalability, and compatibility requirements to ensure a high-quality user experience. Performance is key, with the platform engineered to load within 3 seconds under standard traffic conditions and capable of handling up to 1,000 concurrent users without compromising responsiveness. Security measures are robust, including SSL encryption for secure data transmission, stringent user authentication protocols, and adherence to data protection regulations such as GDPR to safeguard sensitive information. Usability is prioritized by creating an intuitive and accessible user interface that facilitates easy navigation and smooth transaction completion with minimal effort from users. To address scalability, the architecture is built to support future growth, accommodating an expanding number of products and users without affecting performance. Additionally, compatibility ensures that the platform functions seamlessly across major web browsers such as Chrome, Firefox, Safari, and Edge and remains fully responsive across various devices, including desktops, tablets, and smartphones. This comprehensive approach guarantees that the portal meets current demands while being prepared for future developments and diverse user needs.

2.6.3. Technical Requirements:

- **Database:** Use MongoDB for data storage, ensuring efficient retrieval and management of user and product information.
- **Backend Framework:** Utilize Express.js for the backend API, facilitating secure communication between the client and server.
- **Frontend Framework:** Implement React to create a dynamic and responsive user interface.
- **Payment Processing:** Integrate with trusted payment gateways (e.g., Stripe, PayPal) to handle transactions securely.

2.6.4. Regulatory Requirements:

To ensure the Farmer's Direct Sales Portal meets all necessary regulatory requirements, it must comply with relevant data protection regulations such as GDPR (General Data Protection Regulation) and CCPA (California Consumer Privacy Act). This involves

implementing robust measures for the secure collection, processing, and storage of user data to protect privacy and ensure transparency. Additionally, adhering to web accessibility standards, such as WCAG (Web Content Accessibility Guidelines), is crucial for making the platform usable by individuals with disabilities. By integrating these practices, the portal not only meets legal obligations but also provides an inclusive and secure user experience, fostering trust and broadening its reach. These specific requirements will guide the development and implementation of the Harvest Hub platform, ensuring it meets user needs while maintaining a focus on sustainability and security.

2.7 EXTERNAL INTERFACE REQUIREMENTS:

The Harvest Hub platform must seamlessly interact with various external systems and interfaces to provide a comprehensive and efficient user experience. These requirements include interactions with payment gateways, third-party services, and external databases, as well as ensuring compatibility with different browsers and devices.

2.7.1. User Interfaces:

- **Web Browsers:** The platform must be compatible with major web browsers, including Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge. This ensures that users have a consistent experience regardless of their browser choice.
- **Mobile Devices:** The website must be fully responsive, providing an optimal user experience on smartphones and tablets running on various operating systems, such as iOS and Android.

2.7.2. Payment Gateways:

- **Integration with Payment Services:** The platform will integrate with secure and a reliable payment gateways such as Stripe and PayPal to handle online transactions. This includes supporting multiple payment methods like credit/debit cards and bank transfers.
- **Secure Transactions:** Payment processing must ensure encrypted data transmission to ensure security of user information.

2.7.3. Database Interfaces:

- **MongoDB:** The platform will use MongoDB as its primary database management system to store and retrieve product information, user data, and transaction records efficiently. This requires seamless interaction between the application server and the MongoDB database.
- **Backup and Recovery:** Integration with backup solutions is necessary to ensure data integrity and availability, allowing for regular data backups and recovery in case of system failures.

2.7.4. Authentication and Authorization Services:

- **User Management Systems:** Integration with external user management systems may be required for advanced authentication features and single sign-on (SSO) capabilities.

2.8 SYSTEM FEATURES:

2.8.1. User Account Management:

- **Registration and Login:** Users can create accounts using email and password including secure authentication mechanisms.

2.8.2. Farmer Account Management:

- **Registration and Login:** Farmers can create accounts using email and password including secure authentication mechanisms.
- **Farmer's approval and Rejection:** Admin can approve as well as reject farmers those who have registered.

2.8.3. Product Catalog:

- **Product Listings:** Comprehensive listings with detailed descriptions, high-quality images, prices, and eco-friendly attributes (e.g., materials, certifications).

- **Categories Listing:** Products are organized into categories, with available price, category.
- **Product Management:** Products that are approved by admin are only displayed on user's view products page.

2.8.4. Shopping Cart:

- **Cart Management:** Users can add, remove, and modify items in their shopping cart, with a summary of total costs, including taxes and shipping fees.

2.8.5. Checkout and Payment:

- **Secure Checkout:** A streamlined and secure checkout process that supports multiple payment methods, including credit/debit cards, PayPal, and other digital wallets.

2.8.6. User Reviews, Ratings & Feedback:

- **Product reviews:** Users can leave reviews and ratings for products they have purchased, helping others make informed decisions.
- **Product feedback:** Allowing users to provide feedbacks for products.
- **Product ratings:** Allows users to provide ratings for each products.

CHAPTER 3

SYSTEM DESIGN AND TEST PLAN

3.1 USE CASE DIAGRAMS:

The use case diagrams for the Harvest Hub platform depict the interactions between different user roles (Admin, Farmer, User) and the system, highlighting their primary functionalities. For users, the diagram includes actions like logging in, searching for products, adding items to the cart, checking out, making payments, and writing reviews. Admins are shown with capabilities to log in, approve or reject farmer applications, view product categories, approve products added by farmer and view complaints, ensuring platform integrity and organization. Farmers, on the other hand, can log in, view categories, add categories, add products, and monitor customer reviews and feedbacks, facilitating efficient product management and customer interaction. These diagrams collectively provide a clear visual representation of the system's functionality, ensuring a comprehensive understanding of user interactions within the platform.

3.1.1 Admin Use Case Diagram:

The Admin use case diagram outlines managing the site's overall operations and ensuring smooth functionality. Administrators have the capability to approve registrations from new farmers. They can also add, edit, or remove farmer profiles as needed, maintaining an updated database of active participants. Monitoring and addressing issues that arise within the platform.

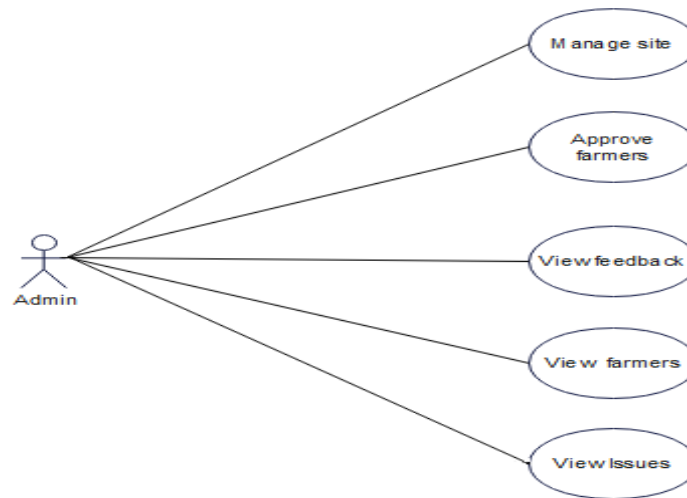


Fig 3.1.1 Admin use case Diagram

3.1.2 Farmer Use Case Diagram:

Use case diagram for Farmers depicts Registration and login processes enabling users to establish and manage their accounts effectively. Farmers can easily add new products to their inventory, complete with detailed descriptions and images, while also monitoring customer feedback through review features to gauge satisfaction and make improvements. Viewing current orders and accessing payment details and reviewing order histories provides comprehensive insights into financial transactions and sales performance.

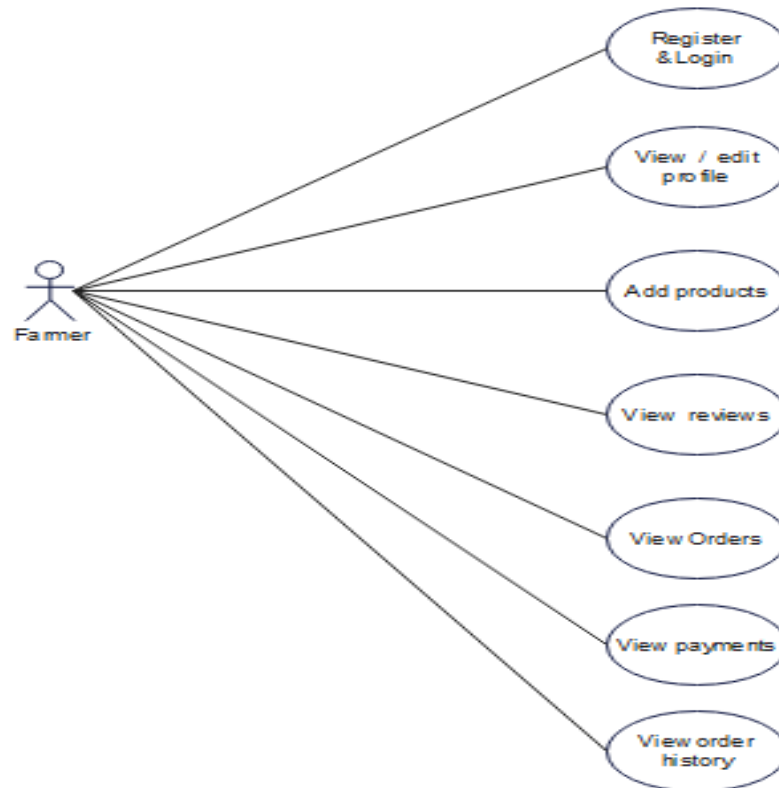


Fig:3.1.2 Farmer use case diagram

3.1.3 User Use Case Diagram:

Use case diagram for Customers depicts Users can register and create accounts securely, followed by logging in to access personalized features. They can view their profiles to manage information such as contact details and farm specifics, with options to update these details as necessary. Farmers showcase their products through detailed listings that include descriptions and images, which buyers can browse and purchase directly through the platform. Seamless payment processing ensures transactions are conducted smoothly and securely. Rating functionalities enable feedback on products and sellers.

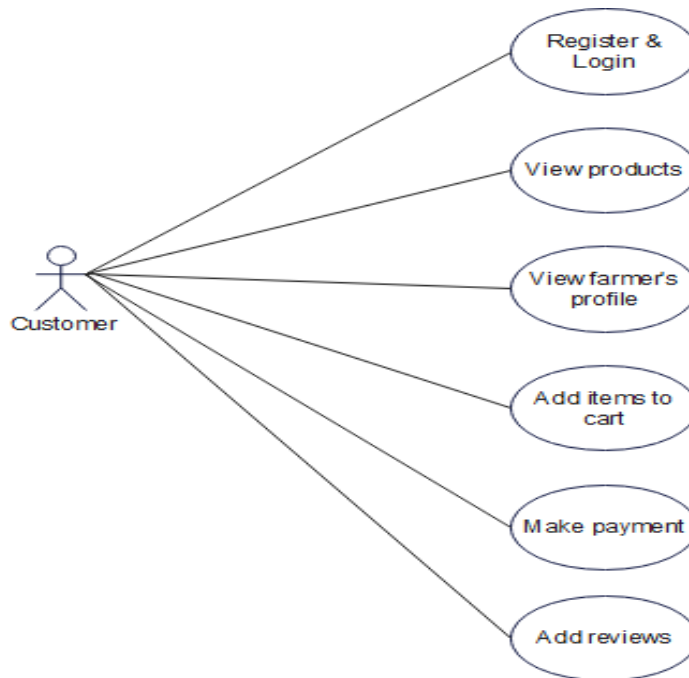


Fig:3.1.3 User use case diagram

3.2 DATA FLOW DIAGRAMS

A Data Flow Diagram (DFD) is a graphical representation used to depict the flow of data within a system or process. It illustrates how data is processed by a system in terms of inputs and outputs. The main components of a DFD include data sources and destinations (external entities), data flows, processes, and data stores. External entities represent sources or sinks of data outside the system boundaries. Data flows indicate the movement of data between processes, data stores, and external entities. Processes represent actions that transform data, and data stores depict places where data is held for later use. By visually mapping out these components, a DFD helps stakeholders understand how information moves through the system, identifies potential bottlenecks or inefficiencies, and aids in the design and analysis of system functionalities. DFDs are typically used in software engineering, system design, and business process modeling.

3.2.1 Level 0 DFD

This diagram depicts a simple system with three components: an Admin, Farmer, and a User. The Admin and User both have connections to the Agri-Business, suggesting that they can interact with it. The bidirectional arrows indicate that information or actions can flow in both directions between these components. This could represent an e-commerce platform where the admin manages the Harvest Hub and the User browses and purchases products.

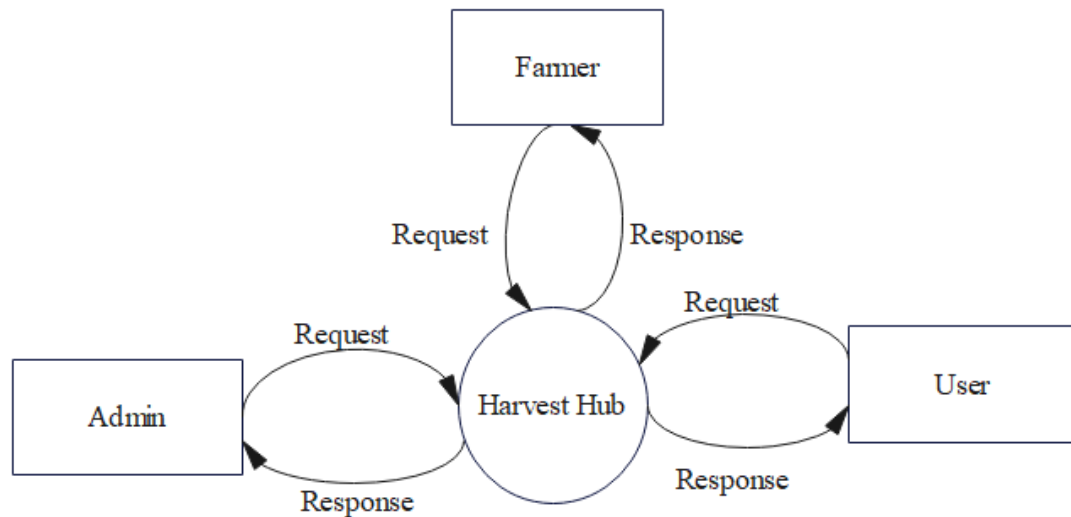


Fig:3.2.1 Level 0 DFD

3.2.2 Level 1 DFD

The data flow diagram outlines admin managing the site's overall operations and ensuring smooth functionality. Administrators have the capability to approve registrations from new farmers. They can also add, edit, or remove farmer profiles as needed, maintaining an updated database of active participants. Monitoring and addressing issues that arise within the platform. Farmers depicts Registration and login processes enabling users to establish and manage their accounts effectively. Farmers can easily add new products to their inventory, complete with detailed descriptions and images, while also monitoring customer feedback through review features to gauge satisfaction and make improvements. Viewing current orders and accessing payment details and reviewing order histories provides comprehensive insights into financial transactions and sales performance. Users can register and create accounts securely, followed by

logging in to access personalized features. They can view their profiles to manage information such as contact details and farm specifics, with options to update these details as necessary. Farmers showcase their products through detailed listings that include descriptions and images, which buyers can browse and purchase directly through the platform. Seamless payment processing ensures transactions are conducted smoothly and securely. Rating functionalities enable feedback on products and sellers.

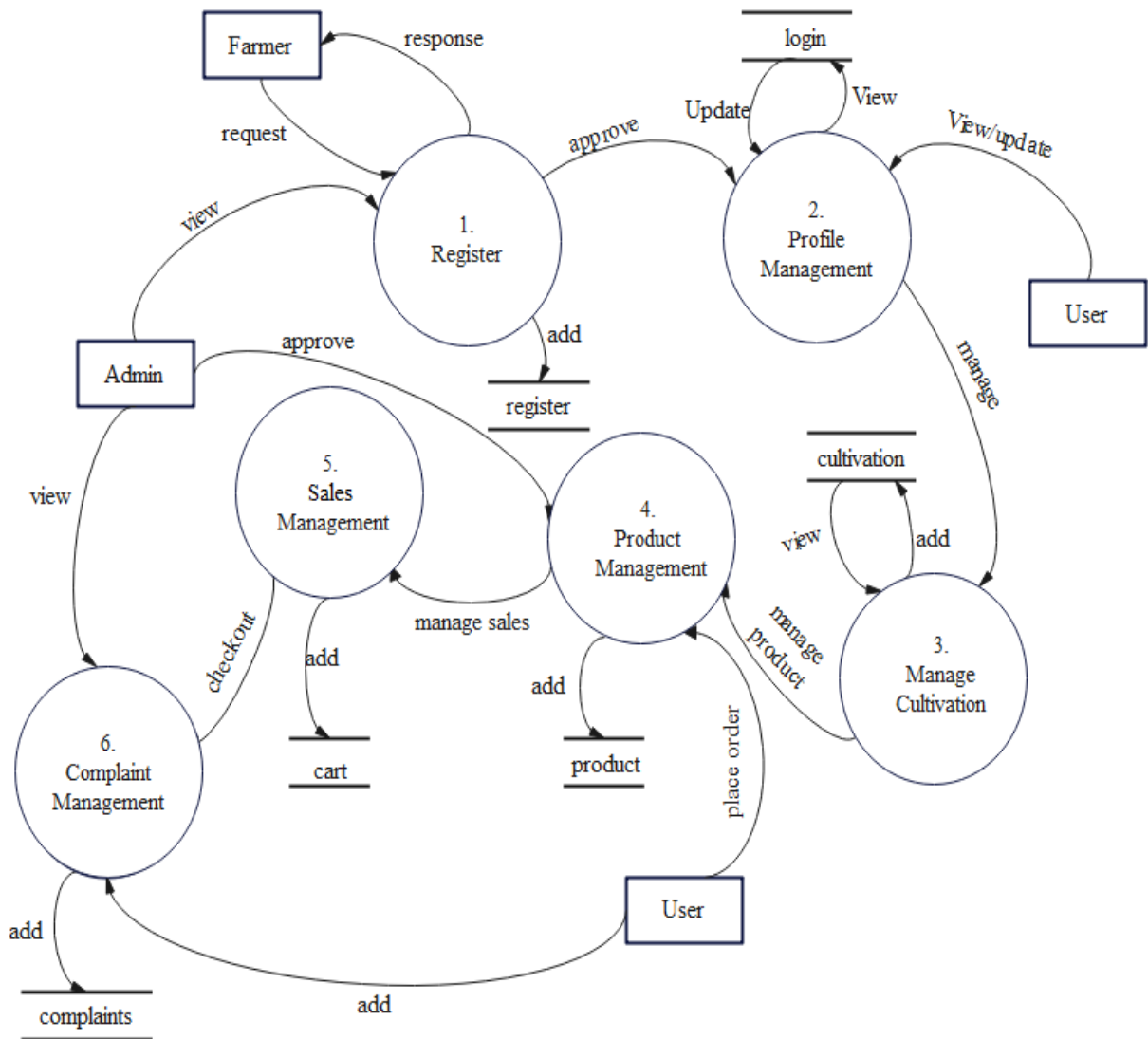


Fig:3.2.2 Level 1 DFD

3.3 TABLE DESIGN

3.3.1 Categories Table:

- This table stores information about category in the Harvest Hub application.

Field	Data Type	Constraints	Required	Unique	Description
Id	Object Id	Primary key	Yes	Yes	Stores category id
Category	String		Yes	No	Stores category name

3.3.2 Login Table

- This table stores information about login details in the Harvest Hub application.

Field	Data Type	Constraints	Required	Unique	Description
Id	Object Id	Primary Key	Yes	Yes	Stores login id
Email	String		Yes	Yes	Stores Email id of user
Password	String		Yes	No	Stores the password
User status	String		No	No	Stores the User status

3.3.3 Register Table

- This table stores information about login details in the Harvest Hub application.

Field	Data Type	Constraints	Required	Unique	Description
Id	Object Id	Primary Key	Yes	Yes	Stores registration id
Name	String		Yes	Yes	Stores Name
Category	String		Yes	No	Stores Category type
Address	String		Yes	No	Stores Address
UserId	Object Id	Foreign Key	Yes	Yes	Stores id of user
Image	Object		Yes	No	Stores Image
Phone	String		Yes	Yes	Stores Phone number
Dob	String		Yes	No	Stores the Date of Birth

3.3.4 Product Table

- This table stores information about products in the Harvest Hub application.

Field	Data Type	Constraints	Required	Unique	Description
Id	Object Id	Primary Key	Yes	Yes	Stores product id
Sub Category	String		Yes	No	Stores sub category details
Weight	Number		Yes	No	Stores weight of product
Price	Int		Yes	No	Stores price of product
Description	String		Yes	No	Stores Description
farmer_id	Object Id	Foreign Key	Yes	Yes	Stores Farmer id

3.3.5 Feedback Table

➤ This table stores information about feedbacks in the Harvest Hub application.

Field	Data Type	Constraints	Required	Unique	Description
Id	ObjectId	Primary Key	Yes	Yes	Stores feedback id
Title	String		Yes	No	Stores feedback title
Description	String		Yes	No	Stores feedback description
User_id	Object Id	Foreign Key	Yes	Yes	Stores id of user
farmer_id	Object Id	Foreign Key	Yes	Yes	Stores id of farmer
Email	String		Yes	Yes	Stores Email id of user

3.3.6 Cultivation Table

➤ This table stores information about cultivation in the Harvest Hub application.

Field	Data Type	Constraints	Required	Unique	Description
Id	ObjectId	Primary Key	Yes	Yes	Stores Cultivation id
Item	String		Yes	No	Stores item details
sub_category	String		Yes	No	Stores sub category details
Cultivation	String		Yes	Yes	Stores cultivation details
farmer_id	Object Id	Foreign key	Yes	Yes	Stores id of farmer

3.3.7 Cart Table

➤ This table stores information about cart in the Harvest Hub application.

Field	Data Type	Constraints	Required	Unique	Description
Id	Object Id	Primary Key	Yes	Yes	Stores Cart id
User_id	Object Id	Foreign Key	Yes	Yes	Stores User id
Product_id	Object Id	Foreign Key	Yes	Yes	Stores Product id
Status	Number		No	Yes	Stores User Status
Updated_At	Date		Yes	No	Stores Updated date and time

3.3.8 Complaints Table

➤ This table stores information about complaints in the Harvest Hub application.

Field	Data Type	Constraints	Required	Unique	Description
Id	Object Id	Primary Key	Yes	Yes	Stores cart id
Title	String		Yes	No	Stores complaint title
description	String		Yes	No	Stores complaint description
Contact	Object Id		Yes	Yes	Stores contact of user
User_id	Object Id	Foreign Key	Yes	Yes	Stores id of user
Username	String		Yes	Yes	Stores username
Status	Number		No	Yes	Stores User Status
Created_At	Date		Yes	No	Stores created date and time

3.3.9 Review Table

➤ This table stores information about cultivation in the Harvest Hub application.

Field	Data Type	Constraints	Required	Unique	Description
Id	ObjectId	Primary Key	Yes	Yes	Stores review id
Product_id	ObjectId	Foreign Key	Yes	Yes	Stores product id
Farmer_id	ObjectId	Foreign Key	Yes	Yes	Stores id of farmer
Rating	Number		No	No	Stores rating count
Created_At	Date		Yes	No	Stores created date and time

3.4 PERFORMANCE REQUIREMENTS

The performance requirements for the Harvest Hub project are essential to ensure a smooth user experience and efficient operations. The Harvest Hub Portal must adhere to rigorous performance and reliability standards to ensure an optimal user experience. The website should achieve a response time of less than 2 seconds for loading pages and retrieving product data, crucial for maintaining user engagement and satisfaction. It must also be scalable, supporting at least 1,000 concurrent users without performance degradation, with a database that efficiently scales to handle growing volumes of products, users, and orders. To guarantee reliability, the platform should maintain a minimum uptime of 99.9%, ensuring continuous access for users through redundant systems and failover mechanisms. Data retrieval speed is critical, with product searches and retrievals completed within 300 milliseconds, even under high load, necessitating optimized database queries and caching strategies. Payment transactions must be securely processed and completed within 5 seconds to facilitate a smooth checkout experience, achieved through optimized integration with payment gateways and secure communication protocols. Additionally, robust security measures are essential, including data encryption for sensitive information and secure payment processing to protect against unauthorized access and ensure compliance with regulations. These combined efforts will ensure that the portal operates

efficiently, scales with demand, and maintains user trust through reliable and secure performance.

3.5 SOFTWARE QUALITY STANDARDS

The Harvest Hub project should focus on several key software quality attributes to ensure a robust and effective e-commerce platform. The Farmer's Direct Sales Portal must meet rigorous criteria to ensure it provides a superior user experience and operates effectively across various dimensions. Usability is paramount, requiring an intuitive and user-friendly interface that facilitates easy navigation and product discovery for users of all technical levels. Performance is also critical; the platform must deliver fast loading times and quick response rates, particularly during peak traffic periods, with efficient data retrieval and transaction processing to maintain user satisfaction. Scalability ensures that the system can grow seamlessly to accommodate increased user traffic, expanded product inventory, and higher transaction volumes without compromising performance. Security is a top priority, demanding robust encryption, secure authentication, and strict adherence to data protection regulations to safeguard user information and ensure secure transactions. Reliability is essential, with the platform designed to minimize downtime, handle errors gracefully, and provide users with clear feedback to maintain trust and consistency. Maintainability is achieved through a modular and well-documented code base, which supports easy updates, feature additions, and bug fixes with minimal disruption. Interoperability is crucial for integrating with third-party services like payment gateways and shipping providers, enhancing overall functionality and user experience. The portal must also adhere to accessibility standards, ensuring that it is usable by individuals with disabilities by following best practices for web accessibility. Lastly, portability is required for the platform to operate efficiently across various devices and operating systems, ensuring a consistent and reliable experience for all users. This comprehensive approach ensures that the portal not only meets current demands but is also prepared for future growth and technological advancements.

3.6 DETAILED DESIGN

The detailed design of the Eco Market project involves breaking down the architecture into specific components, including frontend, backend, and database design. It involves a comprehensive approach to architecture, components, and interactions to fulfill both functional

and non-functional requirements.

3.6.1 System Architecture:

The system architecture for the MERN-based Farmer's Direct Sales Portal is designed to deliver a seamless, scalable, and secure platform that enhances the connection between farmers and consumers. At its core, the frontend leverages React to provide a dynamic and responsive user interface, employing component-based architecture to ensure a fluid and interactive experience. This includes features such as product listings, shopping carts, and user profiles, with navigation managed by React Router and state handled by Redux or Context API to maintain a coherent data flow. On the backend, Node.js and Express.js work together to manage server-side operations, creating RESTful APIs that handle user authentication, product management, and order processing while incorporating middleware for security and data validation. Node.js's asynchronous capabilities support efficient handling of multiple concurrent requests, ensuring high performance. MongoDB serves as the NoSQL database, providing a flexible schema to accommodate diverse product and user data through collections and documents. Mongoose facilitates interaction with MongoDB, ensuring efficient data management with indexing and optimized queries for quick retrieval. Payment processing is integrated through trusted gateways like Stripe or PayPal, ensuring secure and rapid handling of transactions.

3.6.2 Frontend Design:

- **Technology:** React.js
- **Components:**
 - **Landing Page:** Showcases featured products, categories, and cart.
 - **Product Listing:** Displays a grid of products based on category.
 - **Product Detail Page:** Provides detailed information about the product, including images, description, price, reviews, and an "Add to Cart" button.
 - **Cart Page:** Allows users to view selected items, modify quantities, and proceed to checkout.
 - **Responsive Design:** Ensures optimal user experience across desktops, tablets, and mobile devices using CSS frameworks like Bootstrap or Material-UI.

3.6.3 Backend Design

- **Technology:** Node.js with Express.js
- **APIs:**
 - **User Authentication:** Sign up, login using username and password .
 - **Product Management:** CRUD operations for products (for admin), including adding new products, updating stock, and managing product categories.
 - **Order Processing:** End points for creating, retrieving, and updating orders, including payment status.
 - **Review Management:** Endpoints for submitting and retrieving product reviews in form of feedbacks and ratings.

3.6.4 Database Design

- **Technology:** MongoDB
- **Collections:**
 - **Login:** Stores user details and authentication information.
 - **Register:** Stores registered users as well as farmers information.
 - **Product:** Contains product attributes, including name, description, price, image of product, category etc.
 - **Orders:** Records user orders with product details and status with ordered date and time details.
 - **Feedbacks:** Captures user reviews for products.
 - **Ratings:** Stores ratings provided by users for each products.
 - **Cart:** Temporarily stores user selections before purchase.
 - **Complaints:** Stores complaints raised by users.

3.6.5 Security Measures

Strong security measures are implemented to protect user data and system integrity. Sensitive data, including passwords and personal information, is encrypted, and HTTPS is enforced for secure communication. Role-based access control (RBAC) ensures that only authorized users and authorities can access specific features and data. The system undergoes regular security

audits and updates to address vulnerabilities. Input Validation sanitizes and validates user inputs to prevent SQL injection and XSS attacks. By sanitizing and validating user inputs, the portal ensures that data submitted by users is both safe and correctly formatted before it is processed or stored.

3.6.6 Testing Strategy

- **Unit Testing:** Unit testing involves evaluating individual components or functions of the software to verify that each performs correctly in isolation. In the context of the MERN-based Harvest Hub Portal, unit testing would focus on ensuring that specific frontend elements, such as product cards or shopping cart functionalities, operate as intended when provided with various inputs. For the backend, it would test functions like API endpoints or authentication logic to confirm they return the expected results. By catching bugs at this granular level, unit testing helps maintain high code quality and simplifies the process of isolating and fixing issues, ultimately leading to a more robust and reliable system.
- **Integration Testing:** Integration testing focuses on validating the interactions between different components of the system to ensure they work together seamlessly. For the Harvest Hub Portal, this means testing how the frontend React components interact with the backend Node.js services and how data flows between them. For instance, integration tests would verify that when a user requests product information, the frontend correctly receives and displays data from the backend API. This testing is crucial for detecting issues that may arise when components or systems interact, ensuring that the end-to-end functionality of the portal is coherent and efficient.
- **User Acceptance Testing (UAT):** User Acceptance Testing (UAT) is a critical phase where real users interact with the Farmer's Direct Sales Portal to assess whether it meets their needs and expectations. During UAT, end-users are asked to perform tasks such as browsing products, making purchases, and managing their profiles. Their feedback is crucial for evaluating the portal's usability, functionality, and overall user experience. This phase helps identify any usability issues or feature gaps, allowing for refinements that enhance the portal's effectiveness and ensure it provides a satisfactory experience for

its target audience. UAT ensures that the final product is not only technically sound but also aligns well with user expectations and practical use cases.

3.7 SAMPLING METHODS

Sampling methods for an e-commerce platform like Harvest Hub typically involve gathering data from a subset of users or products to derive insights or make decisions. Here are some relevant sampling methods that could be applied:

3.7.1 Random Sampling:

In the context of the Farmer's Direct Sales Portal, random sampling can be used to gather feedback or conduct market research. For instance, if the portal aims to assess customer satisfaction or gather opinions on product quality, random sampling involves selecting a subset of users from the entire customer base at random. This method ensures that every customer has an equal chance of being chosen, providing a representative sample of the overall user experience. By using random sampling, the portal can obtain unbiased insights into the effectiveness of the platform and make data-driven decisions to enhance user satisfaction and improve service quality.

3.7.2 Stratified Sampling:

Stratified sampling can be employed to understand different segments of users more effectively. For example, the portal can divide its user base into distinct strata such as “farmers”, “first-time customers”, and “regular customers”, and then perform sampling within each stratum. This approach ensures that each subgroup is proportionally represented in the research. For instance, to evaluate the impact of the platform on different types of farmers, stratified sampling allows for an in-depth analysis of each farmer segment, such as small-scale versus large-scale farmers. By analyzing each stratum separately, the portal can tailor its features and services to meet the specific needs of each group, enhancing the overall user experience.

3.7.3 Cluster Sampling:

Cluster sampling involves dividing the user base into clusters, such as geographical regions, and then randomly selecting entire clusters to sample. This method is particularly cost-effective for analyzing geographically dispersed users, as it reduces the need for extensive data collection across all regions. By focusing on specific clusters, Harvest Hub can gain valuable insights into regional preferences and logistical challenges, enabling the platform to tailor its offerings and operations to meet the unique needs of different areas effectively.

3.7.4 Convenience Sampling:

Convenience sampling involves selecting participants who are easiest to reach, which can be practical for initial user feedback or pilot testing. For instance, the portal might use convenience sampling to gather feedback from a subset of farmers who are already active users of the platform. This method allows for quick and cost-effective data collection, providing immediate insights into user experiences. Although it may not always yield a representative sample of the entire user base, convenience sampling can still offer valuable information for refining platform features and addressing any immediate issues before rolling out changes more broadly.

3.7.5 Systematic Sampling:

Systematic sampling can be utilized to ensure a uniform approach to selecting participants. For example, the portal might select every 10th user from the customer database to participate in a survey about the platform's features. This method involves choosing a starting point randomly and then selecting every n th individual from a list. Systematic sampling provides a structured and efficient way to gather feedback while minimizing the risk of bias. By applying this method, the portal can systematically collect data on user satisfaction and identify trends or issues in a manageable and organized manner. Each sampling method offers distinct advantages for gathering data and insights, helping the Harvest Hub to enhance its services and better serve its community of farmers and customers

3.8 Test Cases:

3.8.1 Test case for Admin:

Test Case	Test Case Description	Steps	Expected Result
1	Administrator Login	1. Enter valid administrator credentials 2. Click on the "Login" button.	The administrator successfully logs into the application.
2	Approve farmers	Click on the "New farmer" option.	The administrator is redirected to the page where they can manage shop-related activities, such as approving or rejecting the farmers.
3	View farmers	Click on the "Farmer List" option.	The administrator is redirected to the page where they can view approved farmers.
4	View profile	Click on the "Profile" option.	The administrator is redirected to the page where they can view as well as update their profile details.
5	View Category	Click on the "Category" option.	The administrator is redirected to the page where they can manage categories such as adding, editing, deleting.

6	View Orders	Click on the "Order List" option.	The administrator is redirected to the page where they can view orders with ordered date and time details
7	Approve Product	Click on the "Product" option.	The administrator is redirected to the page where they can manage activities related to the Product details and they can approve or reject them.
8	View Issues	Click on the "Issues" option.	The administrator is redirected to the page displaying issues added by customer.
9	Administrator Logout	Click on the "Logout" button.	The administrator is successfully Logged out and redirected to the home page.

Table3.8.2 Test Case For Farmer:

Test case	Test case Description	Steps	Expected Result
1	Farmer Login	1. Enter valid farmer credentials 2. Click on the "Login" button.	The farmer successfully logs into the application and is directed to the farmer dashboard.
2	Farmer Registration	1. Fill in the required fields with valid farmer information.	The farmer is successfully registered and

		2. Click on the "Register" button.	receives a confirmation message.
3	Add category	Click on the "Category Option"	The farmer can add the category details and also view them.
4	Product Details	1. Click on product details button 2. fill the details of products including image. 3. Click on Submit	The farmer can view the products that they have added .
5	Add cultivation	Click on the "Cultivation" option	The farmer can add the category details .
6	Sales History	Click on the "Sales History" option.	The administrator is redirected to the page where they can view sales history with ordered date and time details
7	View Ratings	Click on the "Rating" button .	The farmer can view the Ratings added by the customer for products.
8	View feedbacks	Click on the "feedback" button .	The farmer can view the feedbacks added by the customer.
9	Logout	Click on the "Logout" button.	The farmer successfully Logged out and redirected to the home page.

Table3.8.3 Test Case For User:

Test Case	TestCase Description	Steps	Expected Result
1	User Login	1. Enter valid User credentials 2. Click on the "Login" button.	The User successfully logs into the application and is directed to the User home page.
2.	User Registration	1. Fill in the required fields with valid User information. 2. Click on the "Register" button.	The User successfully registered and receives a confirmation message.
3.	View Profile	Click on the "Profile" option	The user can view as well as update their profile details.
4.	View Products	Click on the "Home" option	The user can view all the products that have been added by the farmer in the home page.
5.	Add to Cart	Click on "add to cart" button.	The products will be successfully added to the cart.
6.	Check out	Click on the payment option	The user can successfully pay for the products.
7.	Feedback	Click on the feedback option	The user can add feedback about the products that is viewed by farmer.
8.	Complaints	Click on the complaints option	The user can add complaints about the products and farmers that is viewed by admin.
9.	Logout	Click on the "Logout" button.	The User successfully Logged out and redirected to the login page.

CHAPTER 4

IMPLEMENTATION AND RESULTS

4.1 Implementation and Results:

The implementation of the Harvest Hub project involved developing a comprehensive e-commerce platform using the MERN (MongoDB, Express.js, React, Node.js) stack. The frontend, built with React.js, provided a seamless user experience with features like product listings, detailed product pages, a user-friendly cart, and a robust user account dashboard. The backend, powered by Node.js and Express.js, facilitated secure user authentication, efficient product management, and smooth order processing, all integrated with a MongoDB database for scalable data management. Farmers benefit from enhanced control over product listings, pricing, and inventory management, leading to increased profitability and market reach. For consumers, the portal offers a transparent marketplace where they can access detailed information about locally sourced products, including cultivation practices and freshness, enabling informed purchasing decisions that align with their preferences and values. The integration of a review and rating system fosters trust and accountability within the community, while responsive design ensures accessibility across various devices. Ultimately, the Farmer's Direct Sales Portal not only facilitates economic empowerment for farmers but also promotes sustainability through reduced food miles and supports local agricultural economies, marking a transformative shift towards a more transparent and consumer-centric agricultural market.

4.2 Implementation Results

1. Development Environment Setup

- Set up the necessary development tools and software, including code editors, integrated development environments (IDEs), and version control systems.
- Install and configure the required software dependencies, such as Node.js, MongoDB, and other additional libraries.

2. Farmers Authentication and Authorization

Implement a secure farmers authentication system, allowing farmer to register, log in, and manage their accounts.

3. Add products

After successfully logging in, farmers can add products to the system. The system provides a form specifically designed for adding new products. This form typically includes fields to capture relevant information about the products, such as product name, category, description, image, pricing details, and any other necessary details.

4. View Products

After successfully logging in, farmers can view, make changes to, and remove existing products. The system provides a list of all the products added to the system. This view may include details of products, including name, category, price, and other relevant information. Farmers have the option to make changes to their product information. They may click on the 'Update' Button to access the product editing functionality. If farmers decide to remove a product from the system, they can select the product from the list or grid view and choose the "Delete" option.

5. View feedbacks and ratings

After successfully logging in, users can view complaints. Within the farmer dashboard, there is typically a dedicated section for viewing feedbacks and ratings provided by users.

6. View sales history

After successfully logging in, admin can view sales history.

7. View Profile

Within the farmer profile page, farmers can access and view their profile information. This may include details such as their name, email address, contact number, registration number, and any other relevant information. Farmers have the option to make changes to their profile information. They may click on the 'Update' Button to access the profile editing functionality. Farmers have the option to log out of the application when they are done. This can be achieved by clicking on the 'Logout' Button.

8. Customers Authentication and Authorization

Implement a secure customers authentication system, allowing customer to register, log in, and manage their accounts.

9. Buy Product

After successfully logging in, customers can view the products available on the application. The system provides a powerful search bar where customers can enter product names to find specific items quickly. The system represents detailed information about the products, including product name, category, price, rating, and other relevant details. The system includes an "Add to cart" button associated with each product. When customers find a product they are interested in, they can click the button to add it to their cart for purchase. The system provides a list of all the products added to the cart. This view may include details of products, including name, category, price, and other relevant information. It also provides options to update the quantity, remove items from the cart, or proceed to checkout.

10. Add feedbacks, ratings and complaints

After successfully logging in, customers can add feedbacks, ratings for each products and also add complaints that is viewable to admin.

11. View profile & logout

After successfully logging in, customers can view, delete, and edit their profiles. Within the customer profile page, customers can access and view their profile information. This may include details such as their name, email address, contact number, and any other relevant information. Customers have the option to make changes to their profile information. They may click on the 'Update' Button to access the profile editing functionality. Customers have the option to log out of the application when they are done. This can be achieved by clicking on the 'Logout' Button.

12. Admin Authentication and Authorization

Implement a secure Admin authentication system, allowing admin to log in and manage their accounts.

13. Approve farmers and products

After successfully logging in, admin can approve or reject the registered farmers. Within the admin dashboard or management page, there is an option to view the products listed by farmers within the system admin can approve or reject the products added by farmers.

14. View sales history

After successfully logging in, admin can view sales history.

15. View Complaints

After successfully logging in, admin view complaints raised by users.

16. View profile & logout

After successfully logging in, admin can view and edit their profiles. Within the admin profile page, admin can access and view their profile information.. They may click on the 'Update' Button to access the profile editing functionality. Customers have the option to log out of the application when they are done. This can be achieved by clicking on the 'Logout' Button.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 CONCLUSION:

Harvest Hub serve as digital hubs that catalyze significant transformations within the agricultural sector. By seamlessly connecting farmers, suppliers, buyers, and service providers, these platforms revolutionize traditional agricultural practices. Through streamlined transactions and improved market access, agribusiness portals empower farmers to reach a broader customer base, secure fair prices for their produce, and expand their economic opportunities. Additionally, these portals serve as invaluable repositories of agricultural knowledge, offering insights into market trends, best practices, and technological advancements. User adoption and satisfaction are pivotal to the success of agribusiness portals, necessitating user-centric designs that prioritize usability and accessibility. Furthermore, the market dynamics facilitated by these portals foster transparency, efficiency, and healthy competition, driving innovation and growth within the agricultural ecosystem.

5.2 FUTURE WORK:

In the coming years, Harvest Hub portals are poised to undergo significant expansion and innovation to meet the evolving needs of the agricultural sector. One of the primary areas of development lies in diversifying the services offered by these platforms. Moving beyond their traditional role as marketplaces, agribusiness portals are expected to integrate a wide range of services such as agricultural advisory, financial management tools, and specialized expertise catering to the unique requirements of farmers, suppliers, and buyers. Furthermore, the future of agribusiness portals will witness a deeper integration of cutting-edge technologies like artificial intelligence (AI), machine learning, Internet of Things (IoT), and blockchain.

These advancements will empower users with predictive analytics, optimized resource management, and enhanced transparency across the agricultural supply chain. Through Mobile accessibility will remain a key priority for agribusiness portals, particularly in rural and remote

areas. Leveraging the widespread penetration of smartphones, these platforms will provide convenient access to services and information, bridging the digital divide and empowering users with real-time data and insights, even in areas with limited internet connectivity.

5.2.1 Future Implementation Plan:

1. **Mobile App:** Developing a mobile app for Harvest Hub can make the platform more accessible to customers and farmers who prefer using their smartphones for online shopping.
2. **Voice Assistant:** Incorporating a voice assistant feature can make Harvest Hub more userfriendly for customers who have difficulty typing or navigating through the website.
3. **Smart Farming:** Integrating IoT and smart farming technologies can help farmers monitor their crops, soil, and weather conditions remotely and optimize their yields.
4. **Sustainable Agriculture:** Encouraging and promoting sustainable agriculture practices and eco-friendly products can make Harvest Hub more attractive to environmentally conscious customers and farmers.
5. **Regional Languages:** Providing Harvest Hub in regional languages can help farmers and customers who are more comfortable communicating in their local languages

REFERENCES

Journal Articles:

- 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.
- Gershon Feder, Roger Slade. The Acquisition of Information and the Adoption of New Technology, American Journal of Agricultural Economics. 1984.
- Goyal Aprajita. Information, Direct Access to Farmers, and Rural Market Performance in Central India, American Economic Journal: Applied Economics, 2010.
- Ganguly Sonali and Patra Sujeet Prakash "Digitization Paradigm shift of Agriculture in "International Journal of Advance Research, Ideas and Innovations, in Technology"
- Shital Chaudhari, Vaishnavi Mhatre, Pooja Patil, Sandeep Chavan, “Smart Farm Application: A Modern Farming Technique Using Android Application”, International Research Journal of Engineering and Technology, volume 5, issue 2, 2018,318-320p.
- NITI Aayog. (2018). Agriculture Index Report: Transforming Indian Agriculture. Received from https://niti.gov.in/writereaddata/files/document_publication/Agriculture_Index_Report.pdf
- Sharma, A., & Pramanik, A. (2020). Agriculture in India: Challenges and way forward. Journal of Community Mobilization and Sustainable Development, 15(2), 232-237
- Government of India, Ministry of Agriculture & Farmers' Welfare. (2021). Krishi Bhavan. Retrieved from <https://krishibhavan.nic.in/>

Textbooks:

- Agri Business Management by Prof.Smita Diwase.
- Agribusiness and Technology: Revolutionizing the Future of Farming by Sujit

Sahgal.

- Objective Agribusiness Management 3rd Ed P/B 2020 by Shakti Ranjan et al Panigrahy.
- "The Mythical Man-Month: Essays on Software Engineering" by Frederick P. Brooks Jr. provides timeless insights on software development challenges, project management, and team dynamics in its original edition.
- "Practical Object-Oriented Design in Ruby: An Agile Primer" by Sandi Metz, in its latest edition (2nd edition), provides invaluable insights into designing objectoriented software using Ruby, focusing on principles and techniques for writing clean and maintainable code
- In "Software Engineering: A Practitioner's Approach" by Roger S. Pressman, 8th edition, the author provides a comprehensive guide to software engineering principles and practices.
- "Analysis and Design of Information Systems" by Arthur M. Langer, 3rd edition, provides comprehensive coverage of the concepts and techniques involved in the analysis and design of information systems.

Web Sources:

- MongoDB.(n.d.).MongoDB. Retrieved from <https://www.mongodb.com/>
- Chodorow, K. (2019). MongoDB: The Definitive Guide: Powerful and Scalable Data Storage.
- Express.js.(n.d.).Express.js.Retrieved from <https://expressjs.com/>
- React.(n.d.).React.Retrieved from <https://reactjs.org/>
- Node.js.(n.d.).Node.js.Retrieved from <https://nodejs.org/>
- Soumalya Ghosh, A. B. Garg, Sayan Sarcar, P.S.V.S Sridhar, Ojasvi Maleyvar, and Raveesh kapoor, “Krishi-Bharati: An Interface for Indian Farmer”, Proceedings of the 2014 IEEE Students' Technology Symposium, 28 Feb.-2 March 2014,Kharagpur, India. <https://ieeexplore.ieee.org/document/123456789>

APPENDIXES

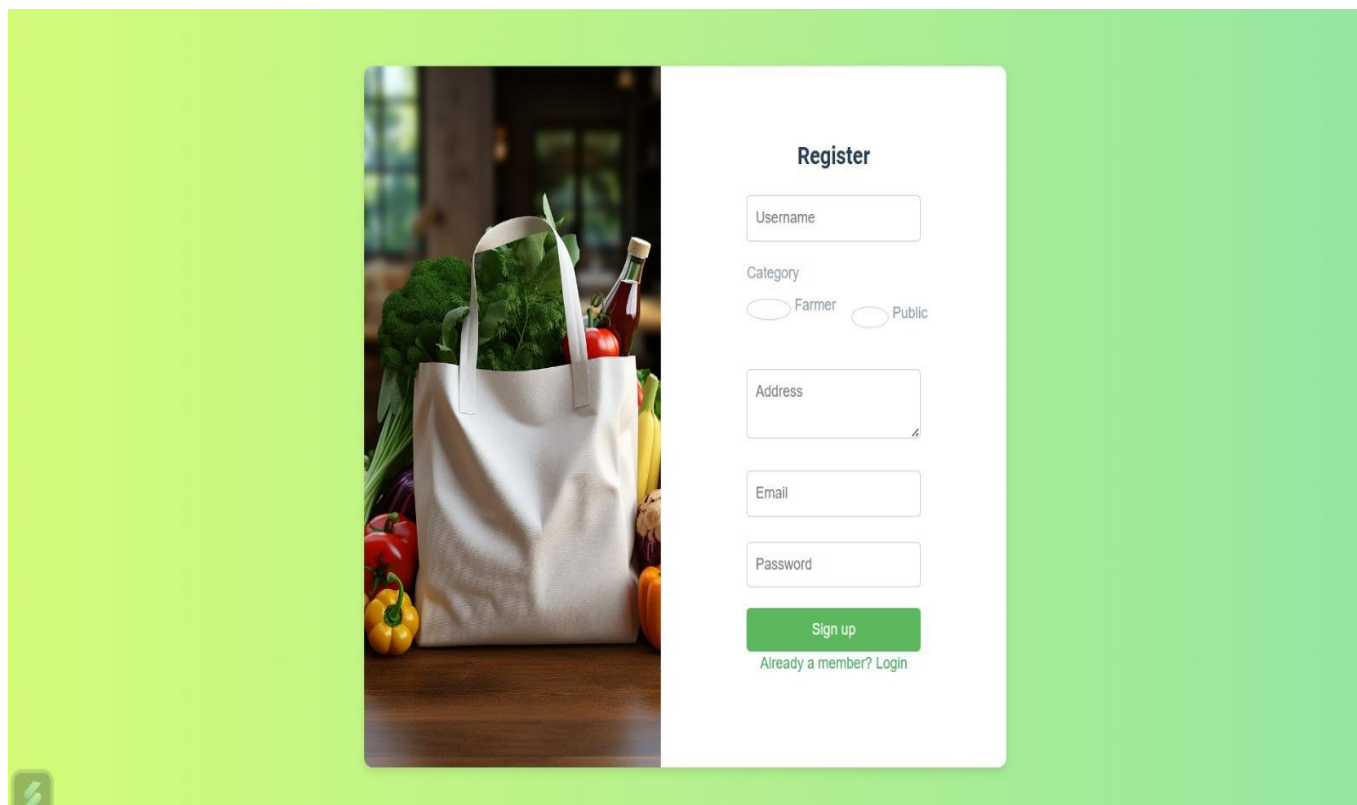
Appendix-A Screenshots



Fig A.1 Home Page



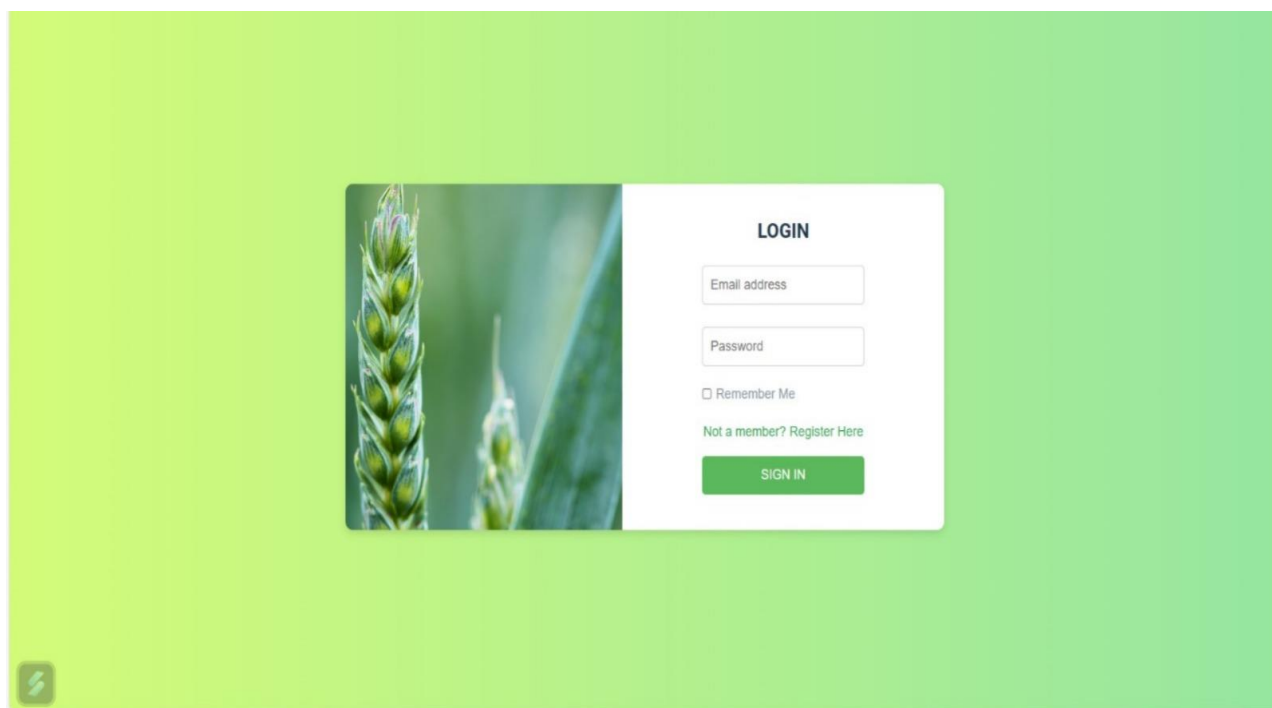
Fig A.2 About us



The registration page features a light green background. On the left, there is a photograph of a white canvas tote bag filled with fresh vegetables, including broccoli, tomatoes, and bell peppers, sitting on a wooden surface. To the right of the image is a white registration form titled "Register". The form contains the following fields and options:

- Register** (Section Header)
- Username
- Category
 - ☐ Farmer
 - ☐ Public
- Address
- Email
- Password
-
- [Already a member? Login](#)

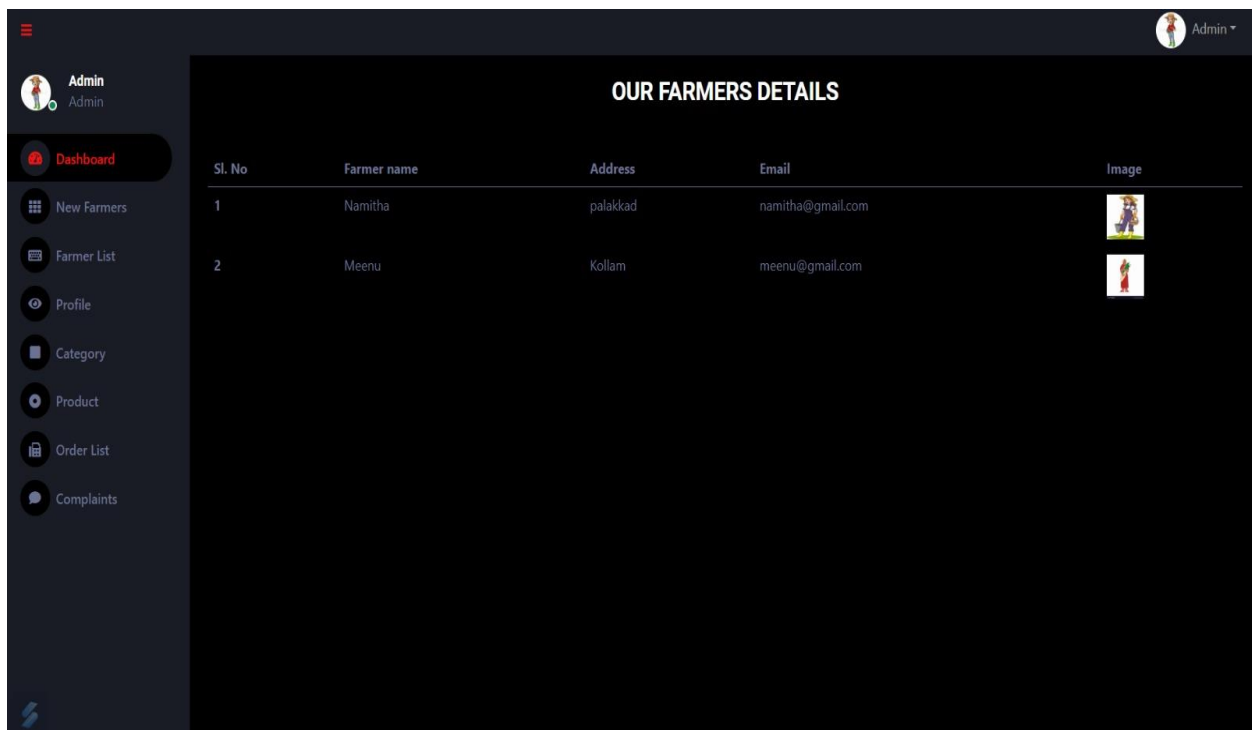
Fig: A.3 Registration Page



The login page features a light green background. On the left, there is a photograph of a close-up of green wheat stalks. To the right of the image is a white login form titled "LOGIN". The form contains the following fields and options:

- LOGIN** (Section Header)
- Email address
- Password
- ☐ Remember Me
- [Not a member? Register Here](#)
-

Fig: A.4 Login Page





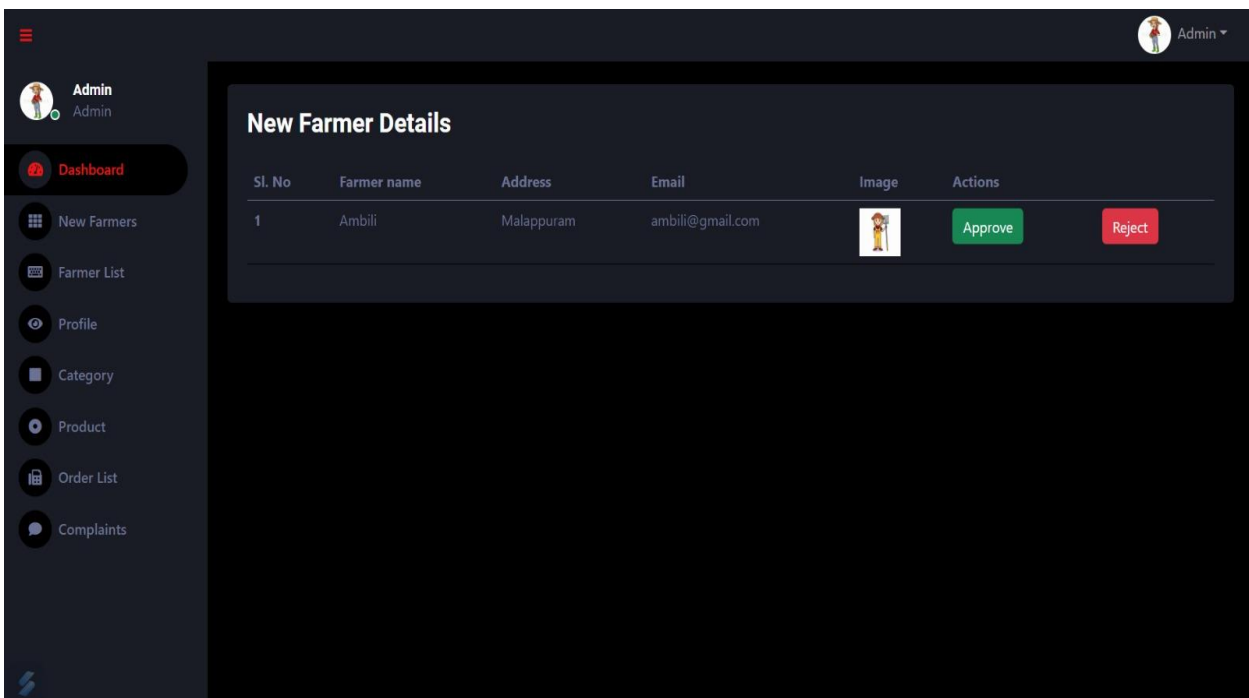
Sl. No	Farmer name	Address	Email	Image
1	Namitha	palakkad	namitha@gmail.com	
2	Meenu	Kollam	meenu@gmail.com	

Fig: A.5 Admin List Farmers




Sl. No	Farmer name	Address	Email	Image	Actions
1	Ambili	Malappuram	ambili@gmail.com		<button>Approve</button> <button>Reject</button>

Fig: A.6 Admin approve farmers

The screenshot shows the 'Add Category' form in the Harvest Hub admin interface. The form is titled 'Add Category' and is located in the main content area. It has a dark theme. The form fields are:

- Category:** A text input field containing the word 'Nuts'.
- Description:** A text input field.
- Image:** A file upload section with a 'Choose File' button and a file name '1.jpeg'.
- Submit:** A red button at the bottom of the form.

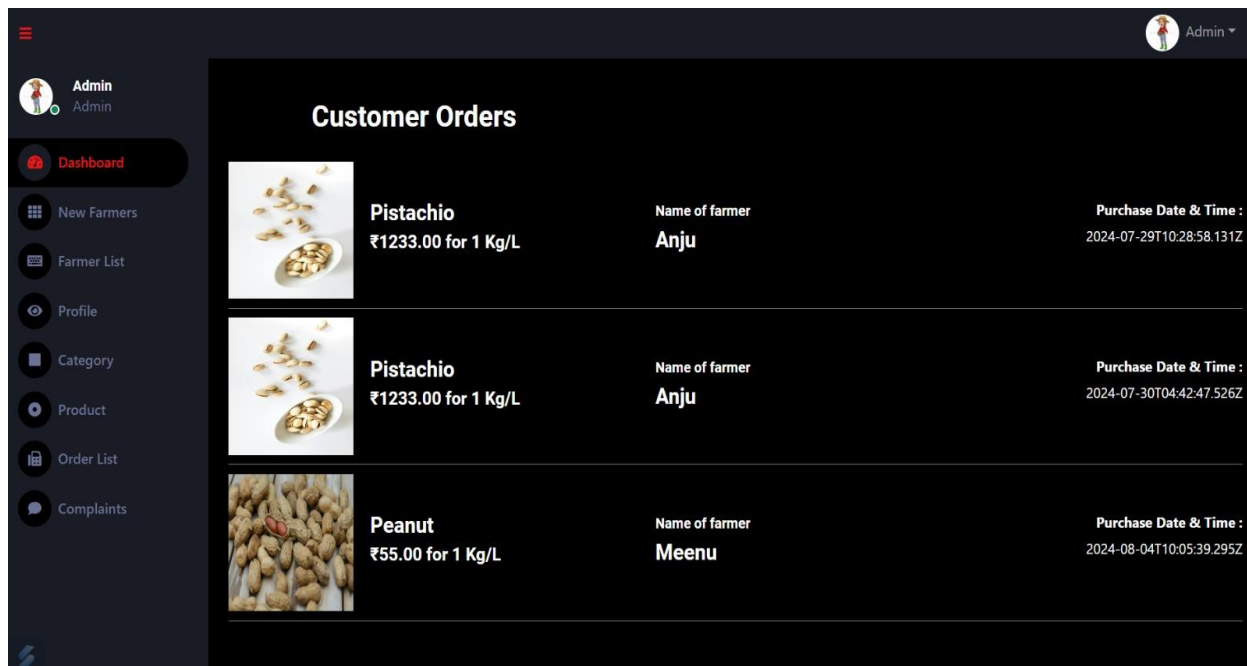
The left sidebar contains the following menu items: Dashboard (selected), New Farmers, Farmer List, Profile, Category, Product, Order List, and Complaints. The top right corner shows the user 'Admin' with a dropdown arrow.

Fig: A.7 Admin add category

The screenshot shows the 'PRODUCT LIST' table in the Harvest Hub admin interface. The table has the following columns: Sl. No, Item, Subcategory, Description, Weight, Price, Picture, and Actions. The table contains 5 rows of product data. The first three rows have a single 'Approved' button in the Actions column. The last two rows have 'Approve' and 'Reject' buttons in the Actions column.

Sl. No	Item	Subcategory	Description	Weight	Price	Picture	Actions
1	Fruits	Apple	ASDEFTGYUIOP	85	115		Approved
2	Nuts	cashew	asadfg	45	1233		Approved
3	Nuts	zxdfv	dfgtyuyio	78	62		Rejected
4	Fruits	Apple	An apple is a round, edible fruit produced by an apple tree	78	125		<button>Approve</button> <button>Reject</button>
5	Nuts	Peanut	The peanut, also known as the groundnut, goober	25	55		<button>Approve</button> <button>Reject</button>

Fig: A.8 Admin approve product



Customer Orders




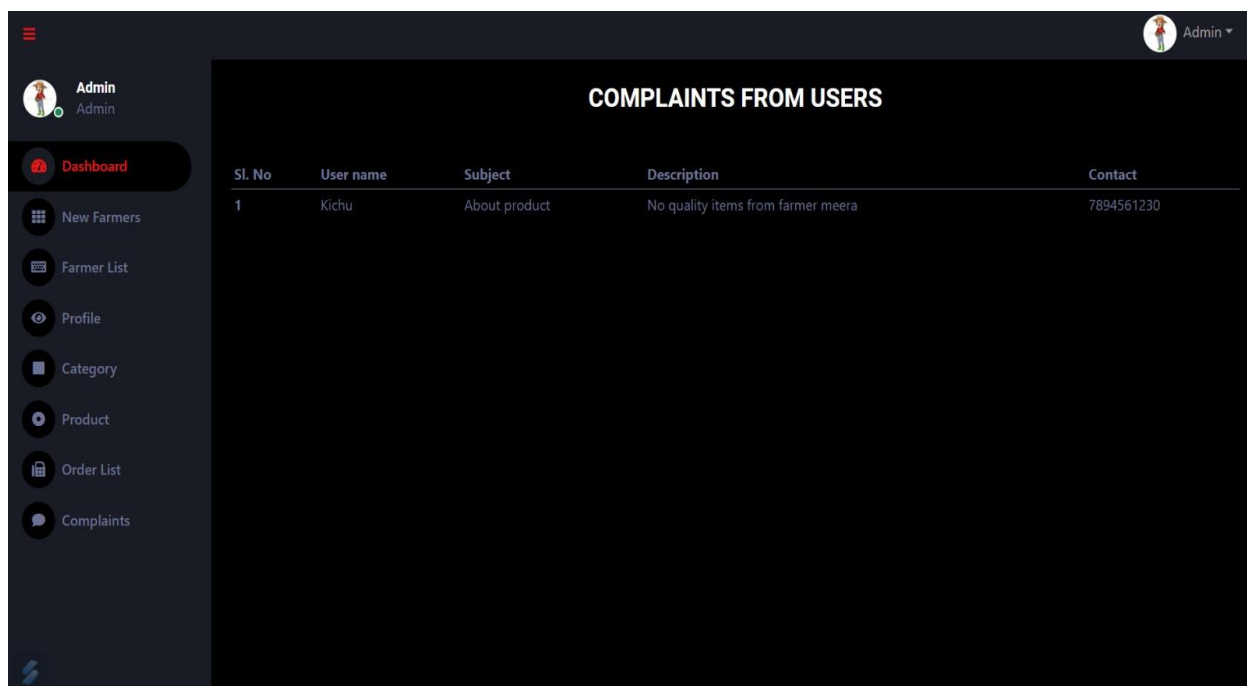
	Pistachio ₹1233.00 for 1 Kg/L	Name of farmer Anju	Purchase Date & Time : 2024-07-29T10:28:58.131Z
	Pistachio ₹1233.00 for 1 Kg/L	Name of farmer Anju	Purchase Date & Time : 2024-07-30T04:42:47.526Z
	Peanut ₹55.00 for 1 Kg/L	Name of farmer Meenu	Purchase Date & Time : 2024-08-04T10:05:39.295Z

Fig: A.9 Admin view sales



COMPLAINTS FROM USERS

Sl. No	User name	Subject	Description	Contact
1	Kichu	About product	No quality items from farmer meera	7894561230

Fig: A.10 Admin view complaints

The screenshot shows the 'Add Product' form in the Harvest Hub application. The form is titled 'Add Product' and is located in the center of the screen. The left sidebar contains a menu with the following items: Dashboard, Profile, Category, Product, Cultivation, Sale History, Rating, and feedback. The user is logged in as 'Meenu Farmer'. The form fields are as follows:

- Category: Fruits
- Subcategory: Apple
- Description: An apple is a round, edible fruit produced by an apple tree
- Total weight/Litre: 78
- Price for 1Kg/1L: 125
- Image: Choose File pexels-mareefe-672101.jpg

A red 'Submit' button is located at the bottom of the form.

Fig: A.11 Farmer Add products

The screenshot shows the 'Add cultivation' form in the Harvest Hub application. The form is titled 'Add cultivation' and is located in the center of the screen. The left sidebar contains a menu with the following items: Dashboard, Profile, Category, Product, Cultivation, Sale History, Rating, and feedback. The user is logged in as 'Meenu Farmer'. The form fields are as follows:

- Item: Nuts
- Subcategory: Penut
- Cultivation: Groundnut agriculture, also known as peanut farming.

A red 'Submit' button is located at the bottom of the form.

Fig: A.12 Farmer Add Cultivation

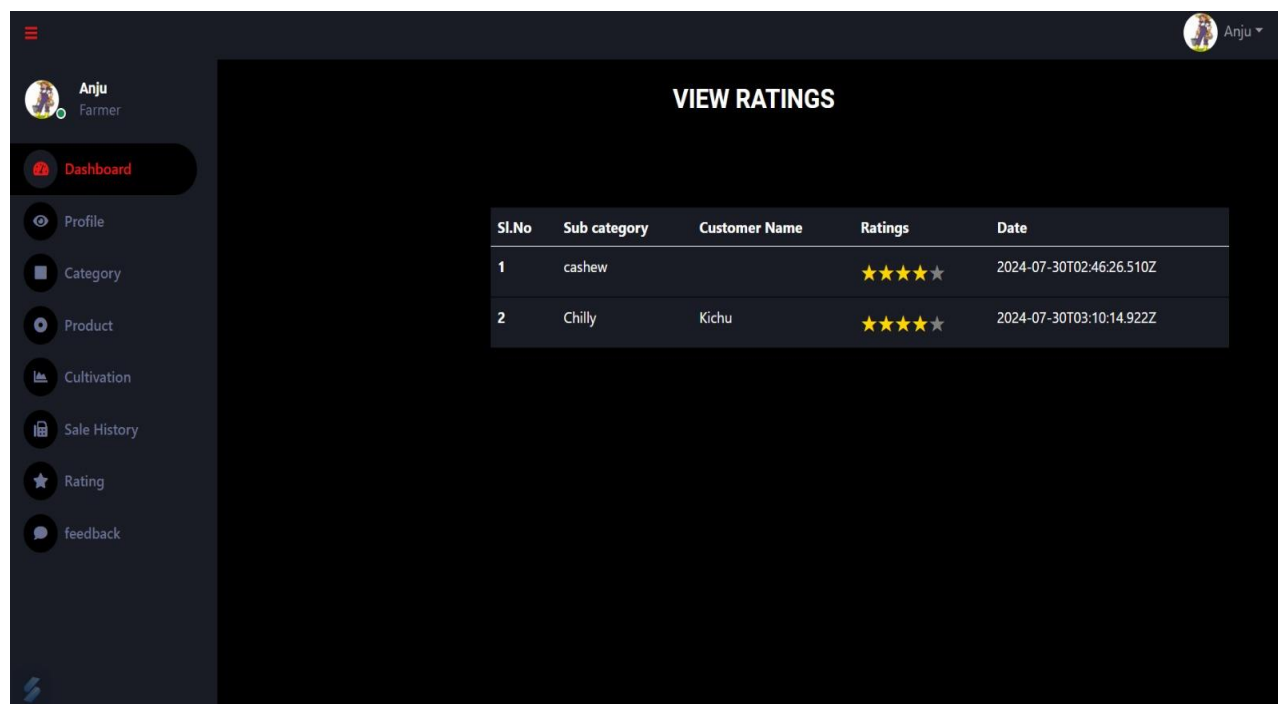


Fig: A.13 Farmer view ratings

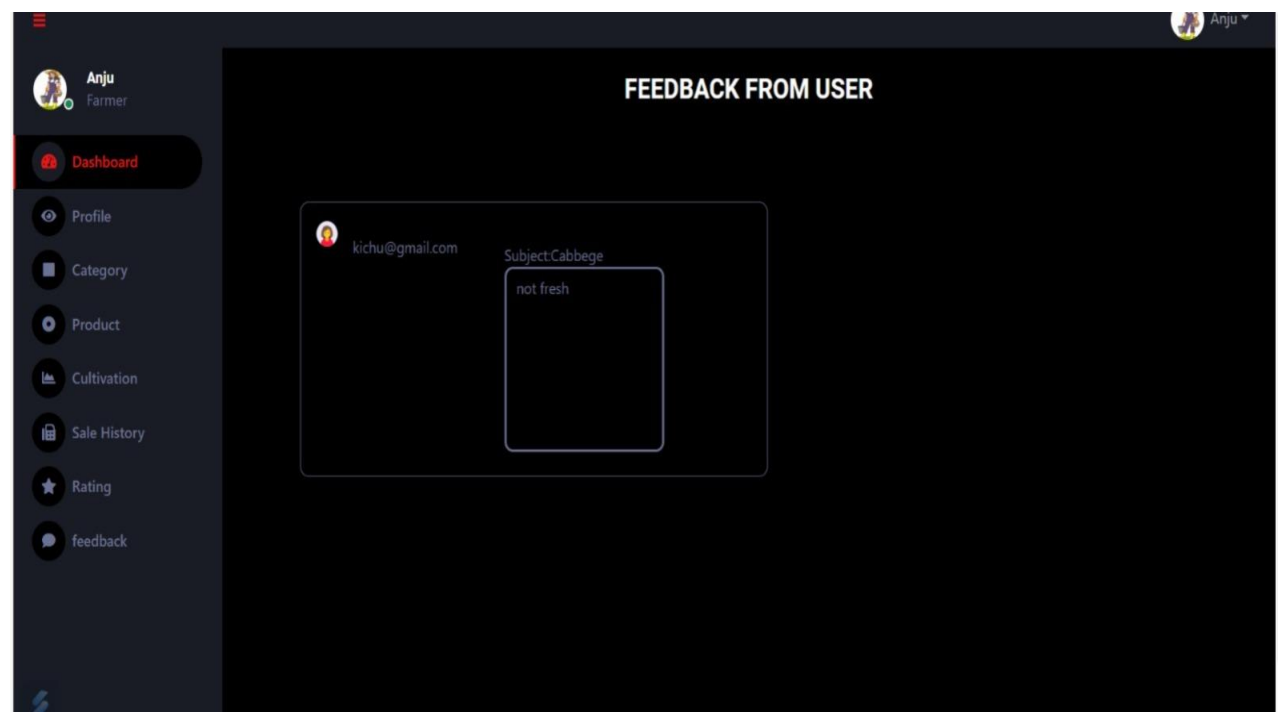


Fig: A.14 Farmer view feedbacks

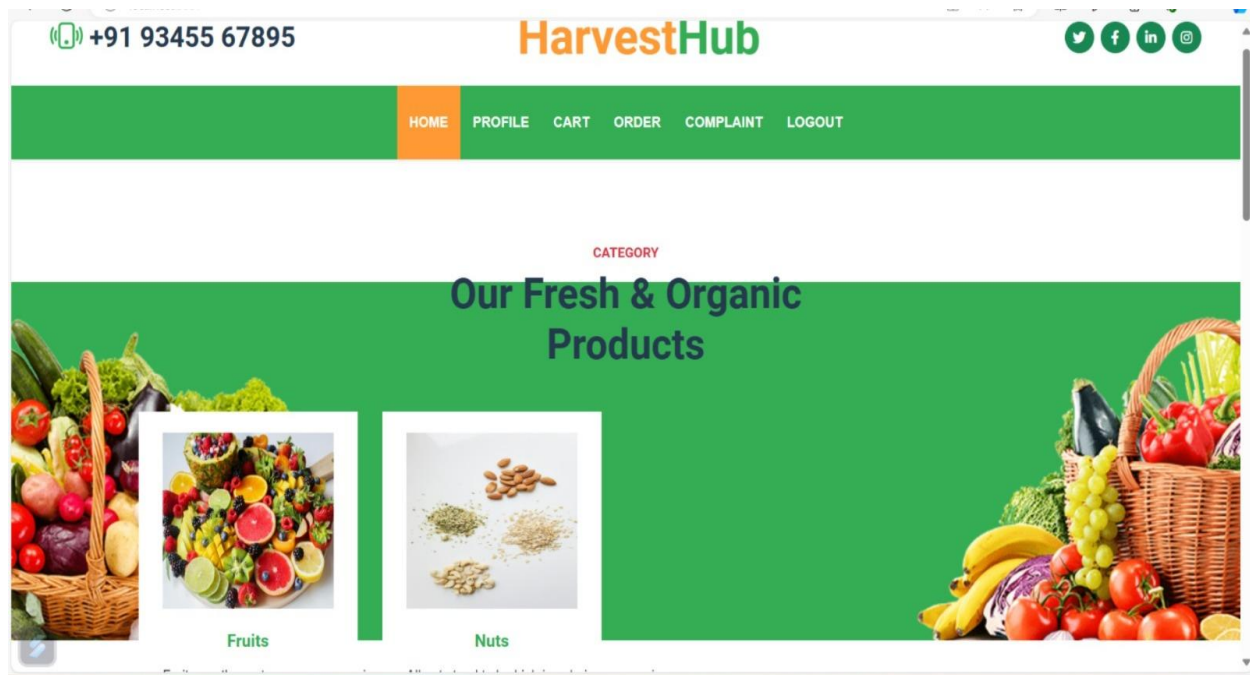


Fig: A.15 User Home page

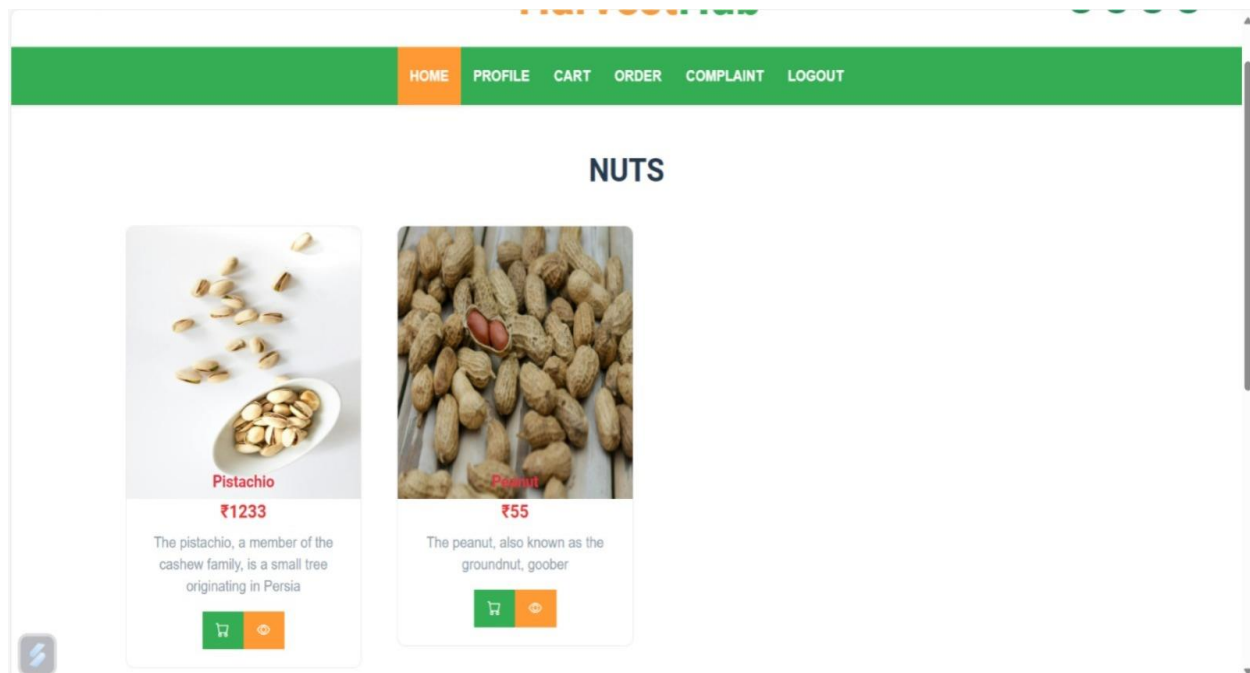


Fig: A.16 View Product categories

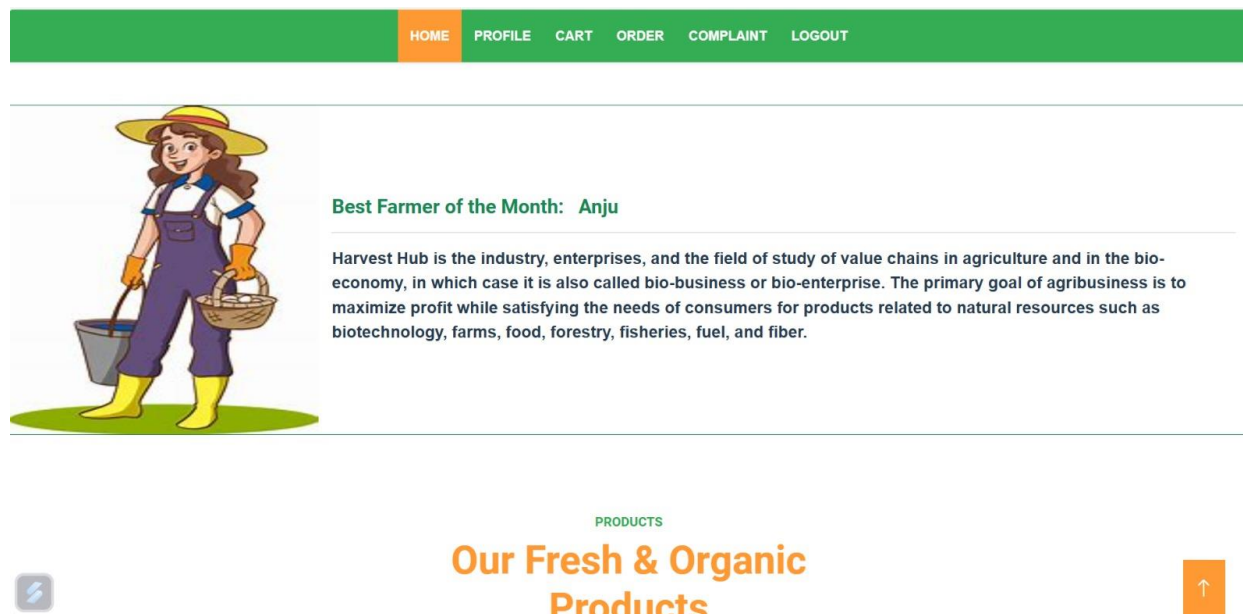


Fig: A.17 View Best Farmer of a month

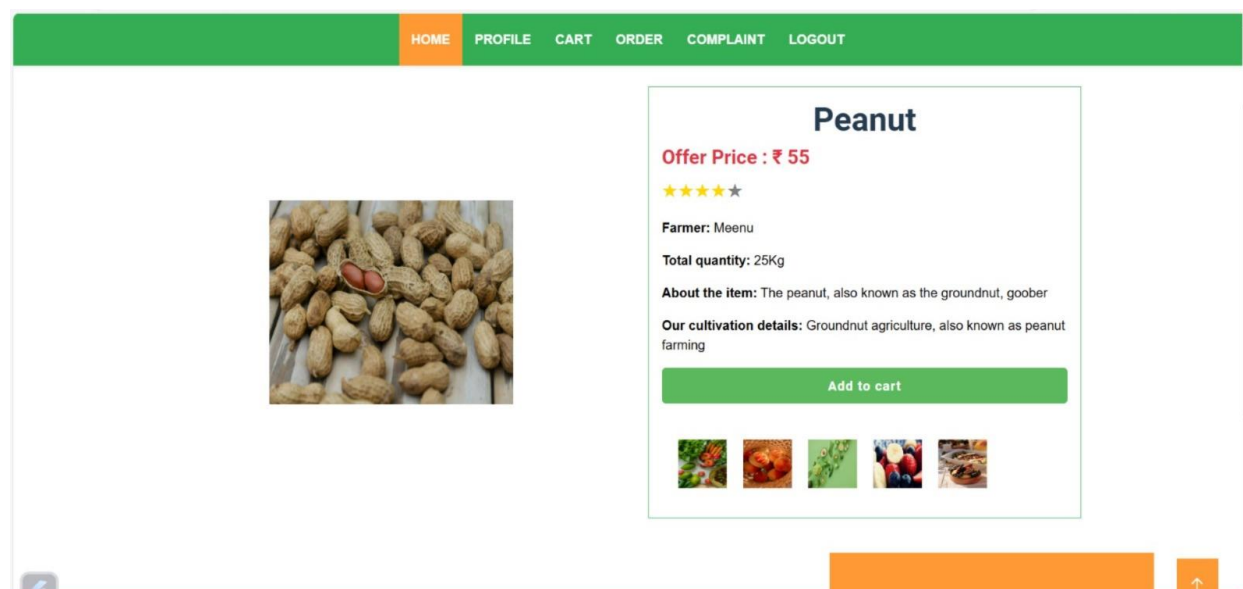


Fig: A.18 User Add to cart

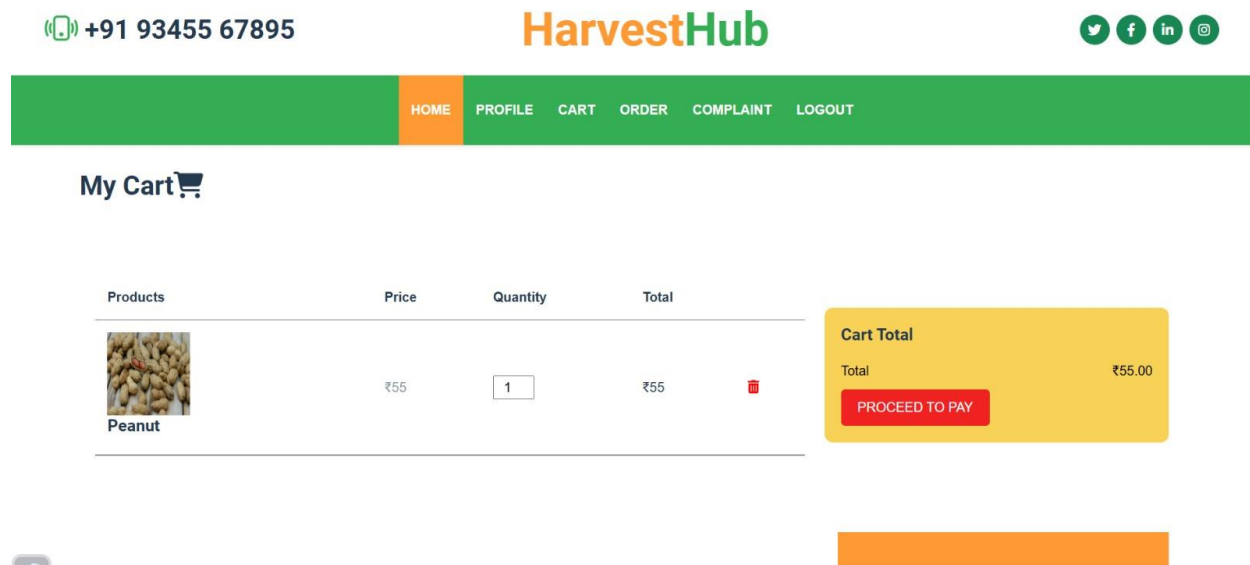


Fig: A.19 Check Out

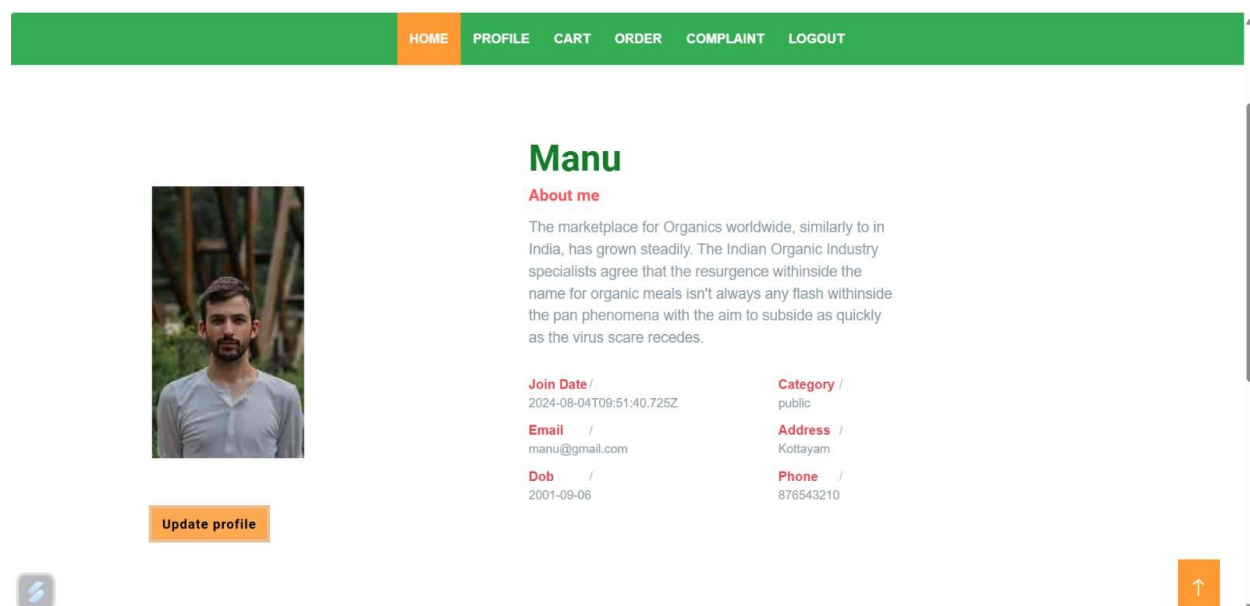



Fig: A.20 User View and Update profile

The screenshot shows the 'My Orders' section of the Harvest Hub application. The top navigation bar is green with links: HOME, PROFILE, CART, ORDER, COMPLAINT, and LOGOUT. The main content area displays an order for 'Peanut' priced at ₹55.00, purchased on 2024-08-04T10:05:39.295Z. The farmer is 'Meenu'. Below the order details is a feedback form titled 'Add Feedback To Farmers'. The form includes a 'Farmer' dropdown menu (currently showing 'Select Farmer'), a 'Subject' text input field, and a 'Description' text area. A green 'Submit' button is at the bottom of the form. The browser's address bar shows 'localhost:3001/order'. The Windows taskbar at the bottom shows the time as 15:40 on 04-08-2024.

My Orders

 **Peanut**
₹55.00

Purchase Date & Time :
2024-08-04T10:05:39.295Z

Farmer: Meenu **Add rating**
★★★★★

Add Feedback To Farmers

Farmer:

Subject:

Description:

Fig: A.21 User Add feedbacks

The screenshot shows the 'Add Complaints' section of the Harvest Hub application. The top navigation bar is green with links: HOME, PROFILE, CART, ORDER, COMPLAINT, and LOGOUT. The main content area displays a form titled 'Add Complaints'. The form includes a 'Subject' text input field, a 'Description' text area, and a 'Contact Information' text input field. A green 'Submit' button is at the bottom of the form. Below the form is a section titled 'Complaint Details' which contains a table with the following columns: SI.NO, Title, Description, and Phone no. The browser's address bar shows 'localhost:3001/complaint'. The Windows taskbar at the bottom shows the time as 15:40 on 04-08-2024.

Add Complaints

Subject:

Description:

Contact Information:

Complaint Details

SI.NO	Title	Description	Phone no
-------	-------	-------------	----------

Fig: A.22 User Add Complaints

Appendix-B Sample Code

7.1 CLIENT SIDE

7.1.1. App.js

```
import React, { useState } from 'react';
import './App.css';
import Login from './Components/Login';
import Register from './Components/Register';
import { BrowserRouter, Route, Routes } from 'react-router-dom';
import Home from './Components/Home';
import Header from './Components/Header';
import Footer from './Components/Footer';
import Cart from './Components/Cart';
import Orders from './Components/Orders';
import FarmerHome from './Farmer/FarmerHome';
import AddProduct from './Farmer/AddProduct';
import FarmerSidebar from './Farmer/FarmerSidebar';
import ProductView from './Farmer/ProductView';
import UpdateProduct from './Farmer/UpdateProduct';
import AddCategory from './Farmer/AddCategory';
import ViewCategory from './Farmer/ViewCategory';
import AddCultivation from './Farmer/Addcultivation';
import ViewCultivation from './Farmer/ViewCultivation';
import UpdateCultivation from './Farmer/Updatecultivation';
import UpdateCategory from './Farmer/UpdateCategory';
import NewFarmers from './Admin/NewFarmers';
import FarmerList from './Admin/FarmerList';
import AdminSidebar from './Admin/AdminSidebar';
import AdminHome from './Admin/AdminHome';
```

```
import UserHome from './User/UserHome';
import Carts from './User/Carts';
import Order from './User/Order';
import Matchproduct from './User/Matchproduct';
import AddFeedback from './User/AddFeedback';
import Payment from './User/Payment';
import CategoryProducts from './User/CategoryProducts';
import ViewFeedback from './Farmer/ViewFeedback';
import SingleProduct from './User/Singleproduct'
function App() {
  const [auth, setAuth] = useState(JSON.parse(localStorage.getItem('userdata')))
  return (
    <BrowserRouter>
      { /* public */ }
      { auth == null ? (
        <>
          <Routes>
            <Route path="/" element={ <Home /> } />
            <Route path="/login" element={ <Login /> } />
            <Route path="/register" element={ <Register /> } />
            <Route path="" element={ <Header /> } />
            <Route path="" element={ <Footer /> } />
            <Route path="/cart" element={ <Cart /> } />
            <Route path="/orders" element={ <Orders /> } />
          </Routes>
        </>
        // admin
      ) : auth.userStatus == 0 ? (
        <>
          <Routes>
            <Route path="/newfarmers" element={ <NewFarmers /> } />
```

```

    <Route path="/farmerlist" element={<FarmerList />} />
    <Route path="/sidebar" element={<AdminSidebar />} />
    <Route path="/" element={<AdminHome />} />
    <Route path="/cart" element={<Cart />} />
    <Route path="/orders" element={<Orders />} />
  </Routes>
</>
// farmer
) : auth.userStatus == 1 ? (
  <>
    <Routes>
      { /* <Route path="" element={<Footer />} /> */ }
      <Route path="/" element={<FarmerHome />}></Route>
      <Route path="/addproduct" element={<AddProduct />} />
      <Route path="/side" element={<FarmerSidebar />}></Route>
      <Route path="/viewproduct" element={<ProductView />}></Route>
      <Route path="/updateproduct" element={<UpdateProduct />}></Route>
      <Route path="/addcategory" element={<AddCategory />} />
      <Route path="/viewcategory" element={<ViewCategory />} />
      <Route path="/updatecategory" element={<UpdateCategory />} />
      <Route path="/addcultivation" element={<AddCultivation />} />
      <Route path="/viewcultivation" element={<ViewCultivation />} />
      <Route path="/updatecultivation" element={<UpdateCultivation />} />
      <Route path="/viewfeedback" element={<ViewFeedback />} />
      <Route path="/cart" element={<Cart />} />
      <Route path="/orders" element={<Orders />} />
    </Routes>
  </>
  // user
) : (

```

```
<>
  <Routes>
    <Route path="/" element={<UserHome />} />
    <Route path='/carts' element={<Carts />} />
    <Route path='/order' element={<Order />} />
    <Route path='/matchproduct' element={<Matchproduct />} />
    <Route path='/addfeedback' element={<AddFeedback />} />
    <Route path='/payment' element={<Payment />} />
    <Route path="/matchproduct/:id" element={<CategoryProducts />} />
    <Route path="/singleproduct/:id" element={<SingleProduct />} />

  </Routes>
</>
)}
</BrowserRouter>
);
}
export default App;
```

7.1.2. List Farmers.js

```
import React, { useEffect, useState } from "react";
import { Link, useNavigate } from "react-router-dom";
import Header from "../Components/Header";
import AdminSidebar from "../AdminSidebar";

function FarmerList() {
  const [farmers, setFarmers] = useState([]);
  const navigate = useNavigate();
  const [auth, setAuth] = useState(JSON.parse(localStorage.getItem('userdata')));
  useEffect(() => {
```

```
fetch('http://localhost:3000/farmerlist')
  .then(res => res.json())
  .then(result => {
    console.log(result);
    setFarmers(result || []);
  })
  .catch(error => console.error('Error fetching:', error));
}, []);

const deletenewfarmer = (id) => {
  let params = {
    id: id
  };
  fetch('http://localhost:3000/deleteNewfarmer', {
    method: 'post',
    headers: {
      Accept: 'application/json',
      'Content-Type': 'application/json'
    },
    body: JSON.stringify(params)
  })
  .then((res) => res.json())
  .then((result) => {
    if (result.success) {
      setFarmers(farmers.filter(farmer => farmer._id !== id));
    } else {
      console.error("Error deleting:", result.error);
    }
  })
  .catch((error) => {
    console.error("Error deleting:", error);
  });
});
```



```

}
return (
  <>
    <Header />
    <div style={{ display: 'flex' }}>
      <AdminSidebar/>
      <main style={{ flex: 1, backgroundImage: "url('img/banner/adlist.jpg')",
backgroundSize: 'cover', backgroundRepeat: 'no-repeat', backgroundPosition: 'center' }}>
        <div style={{ margin:'50px', width: '900px', padding: '30px', border: '1px solid #ccc',
borderRadius: '10px', boxShadow: '0 4px 15px rgba(0, 0, 0, 0.2)', backgroundColor: 'rgba(255,
255, 255, 0.8)' }}>
          <h3 className="mb-5 text-uppercase text-center"> Farmer Details</h3>
          <table className="table table-striped table-sm">
            <thead>
              <tr>
                <th scope="col">Sl. No</th>
                <th scope="col">Farmer name</th>
                <th scope="col">Address</th>
                <th scope="col">Email</th>
                <th scope="col">Actions</th>
              </tr>
            </thead>
            <tbody>
              {farmers.map((data, index) => (
                <tr key={index}>
                  <th scope="row">{index + 1}</th>
                  <td>{data.register.name}</td>
                  <td>{data.register.address}</td>
                  <td>{data.email}</td>
                  <td>
                    <button className="btn btn-danger" type="button" onClick={() =>

```

```
deletenewfarmer(data._id)}>Reject</button>
      </td>
    </tr>
  )}
</tbody>
</table>
</div>
</main>
</div>
</>
);
}
```

export default FarmerList;

7.1.3. Add Category.js

```
import React, { useState } from "react";
import { useNavigate } from "react-router-dom";
import Header from '../Components/Header';
import FarmerSidebar from "../FarmerSidebar";
function AddCategory() {
  const [category, setCategory] = useState("");
  const [description1, setDescription1] = useState("");
  const [img, setImg] = useState(null);
  const [error, setError] = useState("");
  const navigate = useNavigate();
  const [auth, setAuth] = useState(JSON.parse(localStorage.getItem('userdata')))
  const addProduct = (e) => {
    e.preventDefault();
    setError("");
    // Validation
    if (!category || !description1 || !img) {
      setError('All fields are required.');
```

```

    return;
  }
  let formData = new FormData();
  formData.append('category', category);
  formData.append('description1', description1);
  formData.append('img', img);
  formData.append('farmerid', auth._id)

  fetch('http://localhost:5000/addcategory', {
    method: 'post',
    body: formData
  })
    .then((res) => res.json())
    .then((result) => {
      console.log(result);
      navigate('/viewcategory');
    })
    .catch((error) => {
      console.error("Error adding product:", error);
    });
};

return (
  <>
  <Header />
  <div style={{ display: 'flex' }}>
    <FarmerSidebar />
    <main style={{ flex: 1, backgroundImage: "url('img/banner/kenan-kitchen-Bbq3H7eGids-unsplash.jpg')", backgroundSize: 'cover', backgroundRepeat: 'no-repeat', backgroundPosition: 'center' }}>
      <form onSubmit={addProduct} style={{ margin: '70px auto', width: '500px', padding: '20px', border: '1px solid #ccc', borderRadius: '10px', boxShadow: '0 4px 15px rgba(0, 0, 0, 0.2)',

```

```

backgroundColor: 'rgba(255, 255, 255, 0.8)' }}>
  <h3 style={{ textAlign: 'center' }}>Add Category</h3>
  {error && <p style={{ color: 'red', textAlign: 'center' }}>{error}</p>}

  <div className="form-outline mb-4">
    <input type="text" name='category' id="categoryInput" className="form-control"
onChange={(e) => setCategory(e.target.value)} />
    <label className="form-label" htmlFor="categoryInput">Category</label>
  </div>
  <div className="form-outline mb-4">
    <textarea name='description1' id="descriptionInput" className="form-control"
onChange={(e) => setDescription1(e.target.value)} />
    <label className="form-label" htmlFor="descriptionInput">Description</label>
  </div>
  <div className="form-outline mb-4">
    <input type="file" name='img' id="imgInput" className="form-control"
onChange={(e) => setImg(e.target.files[0])} />
    <label className="form-label" htmlFor="imgInput">Image</label>
  </div>
  <button type="submit" className="btn btn-primary btn-block mb-4">Submit</button>
</form>
</main>
</div>
</>
)
}
export default AddCategory;

```

7.2. SERVER SIDE

7.2.1 Database.js

```
const MongoClient = require('mongodb').MongoClient
const client = new MongoClient('mongodb://localhost:27017')

function dbase() {
  return client.connect().then((dbs) => {
    var database = dbs.db('AgriBusiness')
    return database
  })
}

module.exports = dbase()
```

7.2.2. Server.js

```
var express = require('express')
var bodyParser = require('body-parser')
var cors=require('cors')
var mongodb= require('mongodb')
var expressFileupload = require('express-fileupload');
var session = require('express-session')
var database= require('./database')
var path = require ('path')
var app = express()
const Razorpay = require('razorpay');
const crypto = require('crypto');
const fs = require('fs');
const dotenv = require("dotenv");
const { v4: uuidv4 } = require('uuid');
const razorpay = new Razorpay({
  key_id: 'rzp_test_4Ex6Tyjkg79GFy',
```

```
key_secret: 'IVGcQB0HSAttEhr7mq4AbM7Z'
});
// Setting up environment variables
dotenv.config();
app.use(cors({
  origin: 'http://localhost:3000', // Allow requests from this origin
  methods: 'GET,HEAD,PUT,PATCH,POST,DELETE', // Allow these HTTP methods
  credentials: true // Allow sending cookies along with requests
}));
app.use(bodyParser.urlencoded({ extended: false }))
app.use(bodyParser.json())
app.use(cors())
app.use(session({
  secret:'cat',
  resave:false,
  saveUninitialized:true
}))
app.use(express.static(path.join(__dirname, 'public')));
app.use(expressFileupload());
// registration
app.post('/register', async (req, res) => {
  let registerData = {
    name: req.body.username,
    category: req.body.category,
    address:req.body.address,
  }
  let loginData = {
    email: req.body.email,
    password: req.body.password,
    userStatus: req.body.status
  }
}
```

```
try {
  const db = await database
  const loginResult= await db.collection('login').insertOne(loginData)
  registerData.userId = loginResult.insertedId
  await db.collection('register').insertOne(registerData)
  res.json("success")
} catch (err) {
  console.error(err)
}
})
// login
app.post('/login', (req, res) => {
  let logindetails = {
    email: req.body.username,
    password: req.body.userPassword
  }
  // console.log(email);
  // console.log(password);
  database.then((db) => {
    return db.collection('login').findOne({ email: logindetails.email }).then((result) => {
      console.log(result);
      let user = result
      if (user) {
        if (user.password === logindetails.password) {
          req.session.user = user
          if (user.userStatus === 0) {
            res.json(user)
          }
          else if (user.userStatus === 1) {
            res.json(user)
          }
        }
      }
    })
  })
})
```

```
        else if (user.userStatus == 2) {
            res.json(user)
        }
        else {
            res.json('invalid')
        }
    }
    else {
        res.json('invalid')
    }
}
else {
    res.json('invalid')
}
})
})
})

// new farmers
app.get('/newfarmer', (req, res) => {
    try {
        database.then(async(db) => {
            // Check if the database connection is initialized
            if (!database) {
                console.error('Database connection not initialized');
                return res.status(500).json({ error: 'Database connection not initialized' });
            }
            // Fetch farmers from the "register" collection
            const registers = await db.collection('register').find({ category: "farmer" }).toArray();
            // Extract user IDs from the farmer registers
            const userIds = registers.map(user => user.userId);
```



```
// Fetch logins from the "login" collection based on user IDs
let logins = [];
if (userIds.length > 0) {
  logins = await db.collection('login').find({ _id: { $in: userIds }, userStatus: 2 }).toArray();
}
// console.log(logins);
const farmers = logins.map(login => {
  const register = registers.find(register => register.userId.equals(login._id));
  return {
    ...login,
    register
  };
});
// console.log(farmers);
res.json(farmers);
})
} catch (error) {
  console.error('Error during fetching farmers:', error);
  res.status(500).json({ error: 'Internal Server Error' });
}
});
app.get('/farmerlist', (req, res) => {
  try {
    database.then(async(db) => {
      // Check if the database connection is initialized
      if (!database) {
        console.error('Database connection not initialized');
        return res.status(500).json({ error: 'Database connection not initialized' });
      }
      // Fetch farmers from the "register" collection
      const registers = await db.collection('register').find({ category: "farmer" }).toArray();
```

```
// Extract user IDs from the farmer registers
const userIds = registers.map(user => user.userId);
// Fetch logins from the "login" collection based on user IDs
let logins = [];
if (userIds.length > 0) {
  logins = await db.collection('login').find({ _id: { $in: userIds }, userStatus: 1 }).toArray();
}
// console.log(logins);
// Augment farmer data with login information
const farmers = logins.map(login => {
  const register = registers.find(register => register.userId.equals(login._id));
  return {
    ...login,
    register
  };
});
// Return the augmented farmer data
res.json(farmers);
})
} catch (error) {
  console.error('Error during fetching farmers:', error);
  res.status(500).json({ error: 'Internal Server Error' });
}
});
// delete farmer
app.post('/deleteNewfarmer', async (req, res) => {
  let delid = req.body.id;

  try {
    const db = await database;
    const farmer = await db.collection('login').findOne({ _id: new mongodb.ObjectId(delid) });
```

```
if (!farmer) {  
  return res.status(404).json({ error: 'Farmer not found' });  
}  
await db.collection('login').deleteOne({ _id: new mongodb.ObjectId(delid) });  
await db.collection('register').deleteOne({ userId: new mongodb.ObjectId(delid) });  
  
res.json({ success: true, message: 'Farmer and associated user deleted successfully' });  
} catch (error) {  
  console.error("Error deleting farmer:", error);  
  res.status(500).json({ error: "Internal Server Error" });  
}  
});
```