

A Project Report
On
Modernizing Indian Agri Supply Chain
Empowering Farmers through an Online Platform

Submitted to
Acharya Nagarjuna University
In Partial Fulfillment of the Requirement for the Award of
Bachelor's Degree in
Information Technology

by

M. Srihari	Y20AIT507
K. Bhargava Rama	Y20AIT453
P. Sriram	Y20AIT490
T. Gopi Chand	Y20AIT511

Under the guidance of
Dr. B. Krishnaiah, M.E, Ph. D
Assistant Professor



Department of Information Technology
Bapatla Engineering College (Autonomous)
Mahatmaji Puram, Bapatla - 522102
2023-2024
Affiliated to
Acharya Nagarjuna University

Bapatla Engineering College

(Autonomous)

Department of Information Technology

Mahatmaji Puram, Bapatla -522102



CERTIFICATE

This is to certify that the project entitled

“Modernizing Indian Agri Supply Chain

Empowering Farmers through an Online Platform”

M. Srihari

Y20AIT507

is a record of Bonafide work carried out by him, in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology (Information Technology) at BAPATLA ENGINEERING COLLEGE, Bapatla under Acharya Nagarjuna University during the academic year 2023-2024, under our guidance.

Date:

Asst. Prof. Dr. B. Krishnaiah
Project Guide

Asst. Prof. Dr. B. Krishnaiah
Project coordinator

Dr. N. Sivaram Prasad
Prof & H.O.D, Dept of IT

Sign of External Examiner

DECLARATION

We declare that this project report titled “**Modernizing Indian Agri Supply Chain Empowering Farmers through an Online Platform**” submitted in partial fulfillment of the degree of **B. Tech in Information Technology** is a record of original work carried out by me under the supervision of **Dr. B. Krishnaiah, Asst.Professor**, and has not formed the basis for the award of any other degree or diploma, in this or any other Institution or University. In keeping with the ethical practice in reporting scientific information, due acknowledgements have been made wherever the findings of others have been cited.

Date:

Place: Bapatla

M. Srihari (Y20AIT507)

K. Bhargava Rama (Y20AIT453)

P. Sriram (Y20AIT490)

T. Gopi Chand (Y20AIT511)

ACKNOWLEDGEMENT

We are profoundly grateful to **Dr. B. Krishnaiah, Asst.Professor** for his expert guidance and continuous encouragement throughout to see that this Term Paper rights its target since its commencement to its completion.

We would like to express deepest appreciation towards **Dr. Nazeer Shaik**, Principal, Bapatla Engineering College, **Dr. N. Sivaram Prasad**, Head of Department of Information Technology and **Dr. B. Krishnaiah, Asst.Professor**, Project Coordinator whose invaluable guidance supported us in completing this term paper.

At last we must express our sincere heartfelt gratitude to all the staff members of Information Technology Department who helped us directly or indirectly during this course of work.

M. Srihari (Y20AIT507)

K. Bhargava Rama (Y20AIT453)

P. Sriram (Y20AIT490)

T. Gopi Chand (Y20AIT511)

ABSTRACT

The project involves creating an online platform named "KisanRaj" to modernize India's agricultural supply chain. This platform enables farmers to showcase their products, communicate directly with buyers, and negotiate prices efficiently. Buyers can access a diverse range of agricultural products and make offers seamlessly through filtering options. Leveraging technologies like React, Node.js, MongoDB, and Express.js, the portal ensures a user-friendly experience and secure transactions. Hosted on AWS servers, scalability and reliability are emphasized. Ultimately, the project aims to empower farmers and streamline the agricultural trade process for a sustainable future.

Keywords: Agricultural Supply Chain, KisanRaj, Direct Communication, Farmers, Buyers, Product Listings, Negotiation, Market Access, Transparency, User-Friendly Experience, Secure Transactions, React, Node.js, MongoDB, Express.js, AWS Servers, Scalability, Reliability, Sustainability.

TABLE OF CONTENTS

S.NO	DESCRIPTION	PAGE NO
	CERTIFICATE.....	i
	DECLARATION.....	ii
	ACKNOWLEDGEMENT	iii
	ABSTRACT.....	iv
	LIST OF FIGURES.....	vii
	LIST OF ABBREVIATIONS	viii
1	INTRODUCTION	1
	1.1 Purpose.....	1
	1.2 Scope.....	2
	1.3 Overview.....	3
2	LITERATURE SURVEY	4
	2.1 Existing system.....	5
	2.1.1 Disadvantages.....	5
3	SYSTEM REQUIREMENTS SPECIFICATION	6
	3.1 Functional Requirements.....	6
	3.1.1 Farmer Registration and Management.....	6
	3.1.2 Product Listing and Management.....	6
	3.1.3 Communication and Transaction Management.....	6
	3.1.4 User Authentication and Access Control.....	6
	3.2 Non-Functional Requirements.....	7
	3.2.1 Security	7
	3.2.2 Performance	7
	3.2.3 Scalability.....	7
	3.2.4 Reliability.....	7
	3.2.5 Maintainability.....	7
	3.2.6 Flexibility	8
	3.3 System Specifications.....	8
	3.3.1 Hardware Specifications.....	8
	3.3.2 Software Specifications.....	8
4	SYSTEM DESIGN	9
	4.1 UML Diagrams.....	9
	4.1.1 Use Case Diagram.....	10
	4.1.2 Sequence Diagram.....	11
	4.1.3 Activity Diagram.....	12
	4.1.4 ER Diagram.....	13

5	PROJECT PLANNING	15
5.1	Proposed System.....	15
5.1.1	System Architecture.....	15
5.1.2	Flow Chart Representation.....	16
5.2	Modules.....	16
5.2.1	Buy Commodity.....	16
5.2.2	Sell Commodity.....	16
5.2.3	Listing Posted.....	17
5.2.4	Responses Received for a Product.....	17
5.2.5	Post Potential Requirement.....	17
5.2.6	Negotiation History.....	18
5.2.7	Transaction History.....	18
5.2.8	Hire Equipment.....	19
5.2.9	Post Equipment.....	19
5.2.10	Cold Storage.....	19
6	IMPLEMENTATION	20
6.1	Authentication.....	20
6.1.1	Register.js.....	20
6.1.2	Login.js.....	24
6.2	Homepage.js.....	26
6.3	SellCommodity.js.....	29
6.4	BuyCommodity.js.....	34
6.5	NegotiationHistory.js.....	42
6.6	ColdStorages.js.....	50
7	TESTING	53
7.1	Types of Testing.....	53
7.1.1	Functional Testing.....	53
7.1.2	User Acceptance Testing (UAT).....	53
7.1.3	Performance Testing.....	53
7.1.4	Security Testing.....	53
7.2	Testing Approach.....	53
7.3	Test Cases.....	54
7.4	Testing Results.....	54
8	RESULT & DISCUSSION	55
9	CONCLUSION & FUTURE SCOPE	60
9.1	Conclusion.....	60
9.2	Future Scope.....	60
	REFERENCES	61

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
4.1	Use case diagram	10
4.2	Sequence diagram	11
4.3	Activity diagram for to order a product	12
4.4	Activity diagram for to Add a product	13
4.5	ER diagram for KisanRaj	14
5.1	System Architecture	15
5.2	Flow Chart for KisanRaj	16
8.1	Registration page	55
8.2	Login Page	55
8.3	Home page	56
8.4	Sell-Commodity	56
8.5	Buy-Commodity	57
8.6	Post Equipment	57
8.7	Hire Equipment	58
8.8	Post Potential Requirements	58
8.9	Cold Storage	59
8.10	Profile	59

LIST OF ABBREVIATIONS

ABBREVIATION	FULLFORM
APMCs	Agricultural Produce Market Committees
AWS	Amazon Web Service
E-CRM	Electronic Customer Relationship Management
E-NAM	Electronic National Agriculture Market
ER	Entity-Relationship
ICT	Information and Communication Technologies
UAT	User Acceptance Testing
UML	Unified Modelling Language

CHAPTER 1

INTRODUCTION

The agricultural sector in India is on the cusp of a transformative shift, poised to embrace a digital revolution that promises to redefine the way farmers interact with buyers and vice versa. At the forefront of this movement is our project, the development of an online web portal named "KisanRaj", which represents a significant stride towards modernizing the age-old agricultural supply chain in the country.

KisanRaj stands as a pioneering digital marketplace, acting as a bridge between farmers and buyers by facilitating direct communication and transactions. Harnessing the power of cutting-edge technologies like React, Node.js, MongoDB, and Express.js, we have meticulously crafted a platform that delivers a seamless and secure user experience for all stakeholders involved.

The driving force behind KisanRaj stems from the myriad challenges faced by farmers in India, ranging from limited market access and price exploitation by middlemen to inefficiencies in the traditional supply chain. KisanRaj represents a paradigm shift in the way farmers engage with buyers and access crucial resources for crop cultivation. By providing farmers with the ability to list their products for sale, pre-book crops, and access vast amounts of data on equipment and cold storage facilities, the platform serves as a comprehensive solution to enhance market access and promote transparency in agricultural trade.

By leveraging technology to eliminate intermediaries and streamline processes, KisanRaj aims to foster a fair and transparent marketplace where farmers can secure better prices for their produce and buyers can access high-quality agricultural products directly from the source. With a commitment to innovation and inclusivity, we envision KisanRaj as a catalyst for positive change in the agricultural sector, driving sustainable growth and prosperity for farmers across India.

1.1 Purpose

The main aim of the KisanRaj project is to create a simple and easy-to-use computer program that helps farmers and makes trading agricultural products smoother. This program cuts out the middlemen and lets farmers talk directly to the people who want to buy their crops. By doing this, it helps farmers reach more customers and get better prices for what they grow. We're using fancy technology like React, Node.js, MongoDB, and Express.js to make sure the program works well and can handle lots of users. The program will let farmers list their products for sale, talk to buyers to negotiate prices, and make sure the products meet quality standards. Overall, it's all about making life easier for farmers and making sure they get a fair deal for their hard work.

1.2 Scope

The scope of the KisanRaj project extends to modernizing the Indian agricultural supply chain through an online platform that caters to the needs of both farmers and buyers. The system's primary function is to facilitate direct communication and transactions between sellers and buyers, thereby enhancing market access and transparency in agricultural trade. Additionally, the application provides comprehensive stock balance information, enables location-based services, and supports multi-language functionality to promote inclusivity and accessibility. The project's use of cutting-edge technologies ensures scalability, reliability, and security, making it suitable for deployment across different regions and sectors.

Key aspects of the project's scope include:

- **Direct Communication and Transactions:** KisanRaj prioritizes enabling direct communication channels and transaction capabilities between farmers (sellers) and buyers. By eliminating intermediaries, the platform aims to foster greater transparency and efficiency in agricultural trade.
- **Enhanced Market Access:** One of the primary objectives of the project is to enhance market access for farmers by providing them with a centralized platform to showcase their products to a wider audience of potential buyers. This expanded market reach has the potential to increase sales opportunities and revenue for farmers.
- **Transparency and Information Availability:** KisanRaj seeks to promote transparency in agricultural trade by providing comprehensive stock balance information to both sellers and buyers. This feature allows users to make informed decisions based on real-time data about product availability and demand.
- **Location-Based Services:** The application incorporates location-based services to offer customized experiences for users based on their geographical location. This functionality enables farmers to connect with nearby buyers and vice versa, facilitating local trade relationships.
- **Multi-Language Functionality:** Recognizing the diverse linguistic landscape of India, KisanRaj includes multi-language support to ensure inclusivity and accessibility for users from different regions and linguistic backgrounds. This feature enhances user engagement and promotes widespread adoption of the platform.
- **Utilization of Cutting-Edge Technologies:** The project leverages cutting-edge technologies such as React, Node.js, MongoDB, and Express.js to develop a robust and scalable platform. These technologies ensure that KisanRaj can handle large volumes of data, provide a seamless user experience, and maintain high levels of security.
- **Scalability and Deployment Across Regions:** KisanRaj is designed to be highly scalable,

allowing for deployment across different regions and sectors of the agricultural industry. Whether in rural or urban settings, the platform can adapt to varying infrastructural and operational requirements, making it accessible to a wide range of users.

1.3 Overview

The KisanRaj project is a desktop application crafted using a combination of HTML, React.js, CSS, JavaScript, Node.js, Ajax, and MongoDB technologies. Its core objective is to revolutionize the traditional agricultural supply chain in India by introducing a digital platform. This platform serves as a medium for farmers to showcase their products and directly interact with potential buyers. Additionally, the application features an administrative component responsible for overseeing inventory management and ensuring the system remains updated.

By automating tedious manual processes and offering in-depth insights into agricultural products, KisanRaj seeks to empower farmers and improve their access to markets. Through direct communication between farmers and buyers, the platform aims to foster transparency and efficiency in agricultural trade. Ultimately, KisanRaj represents a significant leap forward in creating a more inclusive, resilient, and prosperous agricultural ecosystem in India.

CHAPTER 2

LITERATURE SURVEY

In the current era marked by rapid technological advancements, it has become increasingly essential for farmers and buyers alike to have access to a dependable platform for connecting and conducting transactions. This need is underscored by research findings indicating the significant impact of internet shopping on agricultural products [1], highlighting the crucial role of e-commerce in reshaping modern agriculture [2]. The integration of information technology has facilitated virtual interactions and knowledge sharing among stakeholders [3], which are vital for enhancing supply chain integration and improving overall efficiency in agricultural processes [4].

However, despite the potential benefits, the agricultural sector faces various challenges. For instance, issues related to rural freight transport necessitate the development of innovative solutions to ensure efficient transportation of goods [5]. In the context of Indian agriculture, efforts aimed at upliftment and effective risk management are crucial for sustainable growth [6]. Additionally, the implementation of e-learning initiatives can play a significant role in enhancing the skills and knowledge of agricultural workers, thereby contributing to overall sectoral development [7]. Moreover, leveraging Information and Communication Technologies (ICT) can further enhance the efficiency of horticultural supply chains [8], particularly in the context of e-commerce initiatives [9]. These technological advancements, driven by Industry Revolution 4.0, promise to revolutionize agricultural practices [10], enabling more sustainable production methods and increased efficiency [11].

However, it is essential to acknowledge the challenges associated with digital agriculture [12], including issues related to accuracy, interoperability, data storage, computation power, and farmers' reluctance to adopt new technologies [13]. Despite these challenges, digital platforms offer new opportunities for trade and market integration, albeit with certain complexities that need to be addressed [14]. Furthermore, the impact of initiatives such as the National Agriculture Market (eNAM) underscores the potential for enhancing market integration and price transparency within agricultural markets [15]. Overall, these findings suggest the importance of adopting a comprehensive framework that incorporates digital innovations to advance agricultural practices in the digital age.

In the dynamic realm of online shopping behavior, recent studies have focused on unraveling the drivers behind consumer decisions and the impact of technology on these behaviors. A study in Pakistan investigated the relationship between consumer purchase intention and online shopping behavior, highlighting the moderating role of attitude [16]. Findings revealed a significant positive impact of consumer purchase intention on online shopping behavior, with attitude playing a key moderating role. Another study underscored the importance of website content and trust in internet

shopping, indicating their positive influence on online purchasing intention [17]. However, the moderating effect of trust in internet shopping was deemed insignificant, despite managerial suggestions for improvement. In China's agricultural sector, the burgeoning internet marketing landscape, particularly in rural areas, presents vast opportunities in communication, branding, and efficiency enhancement [18]. While basic internet applications are prevalent, the utilization of advanced tools remains limited. Meanwhile, in Sri Lanka, the escalating trend of e-commerce transactions necessitates an understanding of factors driving online purchase intention, with trust emerging as a significant mediator [19]. Finally, research on e-marketing effectiveness in commercial saffron companies highlighted the role of e-trust as a partial mediator between website content and e-marketing effectiveness, emphasizing implications for practitioners seeking to optimize website content and trust mechanisms in the e-CRM environment [20].

2.1 Existing System

The Electronic National Agriculture Market (E-NAM), is essentially a digital platform introduced by the Government of India to revolutionize the way agricultural commodities are bought and sold across the country. Traditionally, farmers in India would sell their produce through local agricultural produce market committees (APMCs) or mandis. However, these markets often suffered from inefficiencies, lack of transparency, and limited reach. With E-NAM, the goal is to create a unified national market where farmers can sell their produce directly to buyers without the need for physical presence in the mandis. This is facilitated through an online portal that connects various APMCs or mandis electronically. Through this platform, farmers can list their products, specify quality parameters, and set prices, which can then be accessed by buyers from different regions and states.

2.1.1 Disadvantages

- **Slow Loading Times:** ENAM suffers from slow loading times, causing frustration and delays for users.
- **Lack of Negotiation Platform for Farmers:** ENAM lacks a dedicated negotiation platform where farmers can engage in discussions and bargaining with potential buyers.
- **Subscription and Network Access for Farmers:** ENAM may require subscription fees or have limited network access for farmers, hindering their participation.
- **User-Friendly UI/UX Design:** ENAM's user interface and experience design may not be intuitive or user-friendly, leading to confusion and difficulty in navigating the platform.
- **Preselling Before Crop Harvest:** ENAM may not support preselling options for farmers to secure buyers before their crops are harvested, leading to uncertainty and risk.
- **User Profiles for Farmers:** ENAM may lack comprehensive user profiles for farmers, limiting their ability to showcase their products and establish credibility.

CHAPTER 3

SYSTEM REQUIREMENTS SPECIFICATION

Functional and Non-Functional Requirements

Requirement's analysis is very critical process that enables the success of a system or software project to be assessed. Requirements are generally split into two types: Functional and non-functional requirements.

3.1 Functional Requirements

3.1.1 Farmer Registration and Management

- **Register Farmer:** Allow farmers to register with the system by providing necessary details such as name, contact information, location, and crop details.
- **Update Farmer Information:** Provide functionality for farmers to update their profile information, including crop details, contact information, and location.
- **Delete Farmer Profile:** Allow farmers to deactivate or delete their profile from the system.

3.1.2 Product Listing and Management

- **List Products:** Enable farmers to list their agricultural products for sale on the platform, capturing details such as product name, category, quantity, quality, price, and availability.
- **Update Product Information:** Provide farmers with the ability to update product details, including quantity, price, and availability status.
- **Remove Products:** Allow farmers to remove or deactivate their listed products from the platform.

3.1.3 Communication and Transaction Management

- **Direct Communication:** Enable direct communication between buyers and sellers (farmers) for negotiating prices, discussing product details, and finalizing transactions.
- **Transaction Recording:** Record transaction details such as product name, quantity, buyer-seller communication, negotiation details, and transaction date.
- **Transaction History:** Maintain a history of transactions for both buyers and sellers, providing a record of purchases, sales, and negotiation outcomes.

3.1.4 User Authentication and Access Control

- **User Registration:** Allow users (farmers and buyers) to register with their credentials.
- **User Authentication:** Authenticate users during login to ensure secure access to the platform.

3.2 Non-Functional Requirements

3.2.1 Security

Ensuring the security of the KisanRaj platform is paramount to safeguarding user data and preventing unauthorized access. This involves the implementation of robust encryption protocols to protect sensitive information during transmission and storage. Secure authentication mechanisms, such as multi-factor authentication and strong password policies, should be in place to verify the identity of users and prevent unauthorized access to accounts. Additionally, regular security audits should be conducted to identify and address potential vulnerabilities, ensuring that the platform remains resilient against emerging threats and cyberattacks.

3.2.2 Performance

Optimizing the performance of the platform is essential to provide users with a seamless and responsive experience. This requires careful attention to factors such as loading times, responsiveness to user input, and overall system efficiency. Pages should load quickly, even under heavy traffic conditions, and actions such as product listing and communication should be executed promptly to enhance user satisfaction. Performance testing should be conducted regularly to identify bottlenecks and optimize system resources for maximum efficiency.

3.2.3 Scalability

As the user base of KisanRaj expands, the platform must be able to scale seamlessly to accommodate increased demand and user interactions. This necessitates a scalable architecture that can dynamically allocate resources based on workload fluctuations, ensuring that the platform remains responsive and reliable even during periods of peak usage. Horizontal scaling, vertical scaling, and cloud-based solutions should be considered to support the platform's scalability requirements effectively.

3.2.4 Reliability

Ensuring the reliability of the KisanRaj platform is crucial to maintain user trust and satisfaction. The platform should be available to users at all times, with minimal downtime or service interruptions. This requires deploying the platform on reliable hosting infrastructure with built-in redundancy measures to mitigate the impact of hardware failures or network disruptions. Continuous monitoring of system health and performance is essential to detect and address issues proactively, minimizing downtime and ensuring uninterrupted access for users.

3.2.5 Maintainability

Designing the platform with maintainability in mind is essential to facilitate ongoing updates, bug fixes, and enhancements. This involves writing clean and well-documented code that is easy to understand and modify. Adopting modular architecture and efficient development practices enables developers to make changes to the platform quickly and effectively, reducing the time and

effort required for maintenance and support tasks.

3.2.6 Flexibility

The KisanRaj platform should be flexible enough to accommodate future enhancements and changes in requirements, allowing for seamless integration of new features or functionalities. This requires adopting scalable and adaptable technologies and architectures that can evolve with the evolving needs of the agricultural industry. An agile development approach and continuous feedback from users can help identify emerging requirements and prioritize them for implementation, ensuring that the platform remains relevant and competitive in the long run.

3.3 System Specifications

3.3.1 Hardware Specifications

- Processor i3/Intel Processor
- Hard Disk 128 GB
- A multi-core processor, at least 4GB of RAM, SSD storage, gigabit Ethernet interface, and compatibility with Linux-based operating systems.
- Client-side hardware requirements: Any device with a modern web browser

3.3.2 Software Specifications

- Operating System Windows 10
- React for front-end development
- Node.js and Express.js for server-side development
- MongoDB for database management
- AWS for hosting and deployment

CHAPTER 4

SYSTEM DESIGN

4.1 UML Diagrams

UML stands for Unified Modelling Language. UML is a standardized general purpose modelling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modelling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modelling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems.

The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

Goals:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modelling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of programming languages and development process.
4. Provide a formal basis for understanding the modelling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns, and components.
7. Integrate best practices.

4.1.1 Use Case Diagram

A use case diagram, a vital component of the Unified Modeling Language (UML), serves as a visual representation derived from use-case analysis. Its fundamental objective is to offer a graphical depiction of the functionalities offered by a system, including actors, their objectives (depicted as use cases), and any interdependencies among these use cases.

Essentially, a use case diagram illustrates the specific functions performed by a system for each actor involved. It provides insight into the roles played by these actors within the system's context. Actors, which can be users or external systems interacting with the primary system, are represented as stick figures, while use cases are depicted as ovals. Arrows between actors and use cases indicate interactions, showcasing how actors utilize the system's functionalities to achieve their objectives.

By visualizing the relationships between actors and use cases, stakeholders can gain a comprehensive understanding of the system's functionality and how it caters to various user roles. This aids in requirements elicitation, system design, and communication among project stakeholders, ultimately contributing to the successful development and implementation of the system.

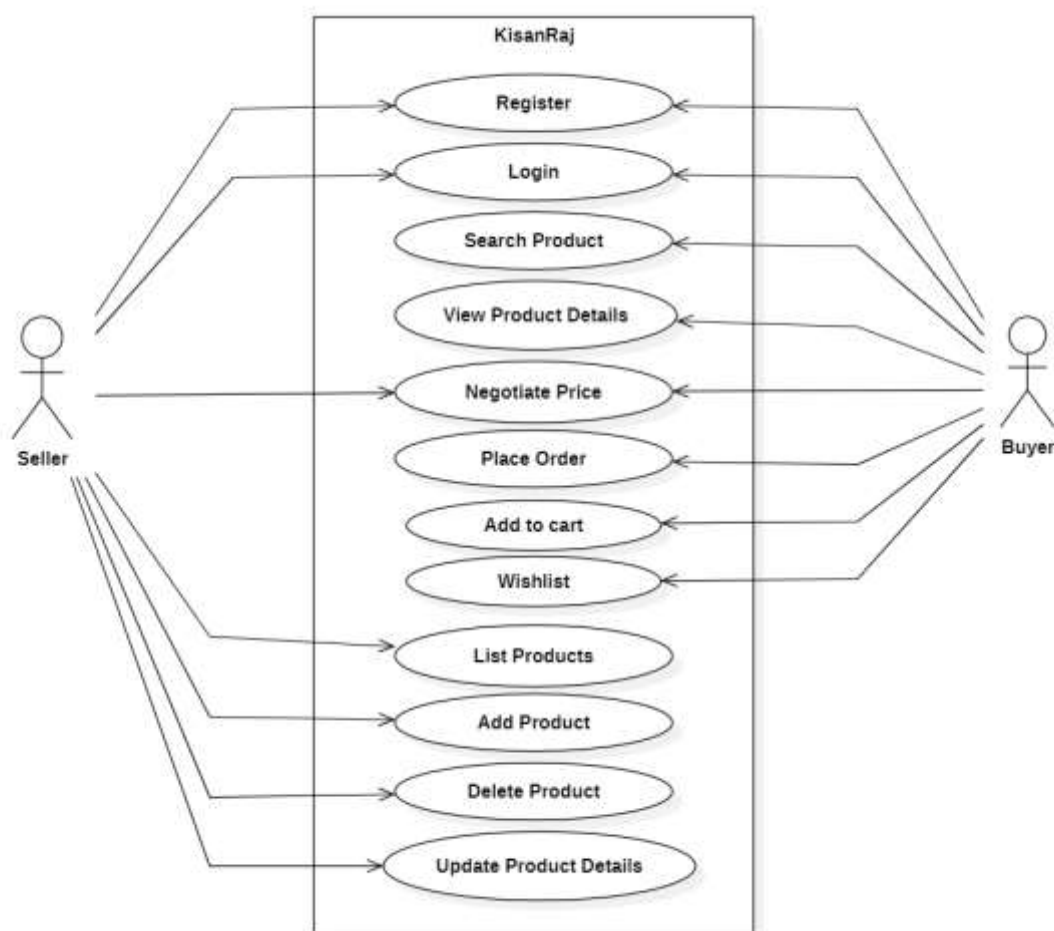


Figure 4.1 Use case diagram

4.1.2 Sequence Diagram

The sequence diagram represents the flow of messages in the system and is also termed as an event diagram. It helps in envisioning several dynamic scenarios. It portrays the communication between any two lifelines as a time-ordered sequence of events, such that these lifelines took part at the run time. In UML, the lifeline is represented by a vertical bar, whereas the message flow is represented by a vertical dotted line that extends across the bottom of the page. It incorporates the iterations as well as branching.

Purpose of a Sequence Diagram

1. To model high-level interaction among active objects within a system.
2. To model interaction among objects inside a collaboration realizing a use case.
3. It either models generic interactions or some certain instances of interaction.

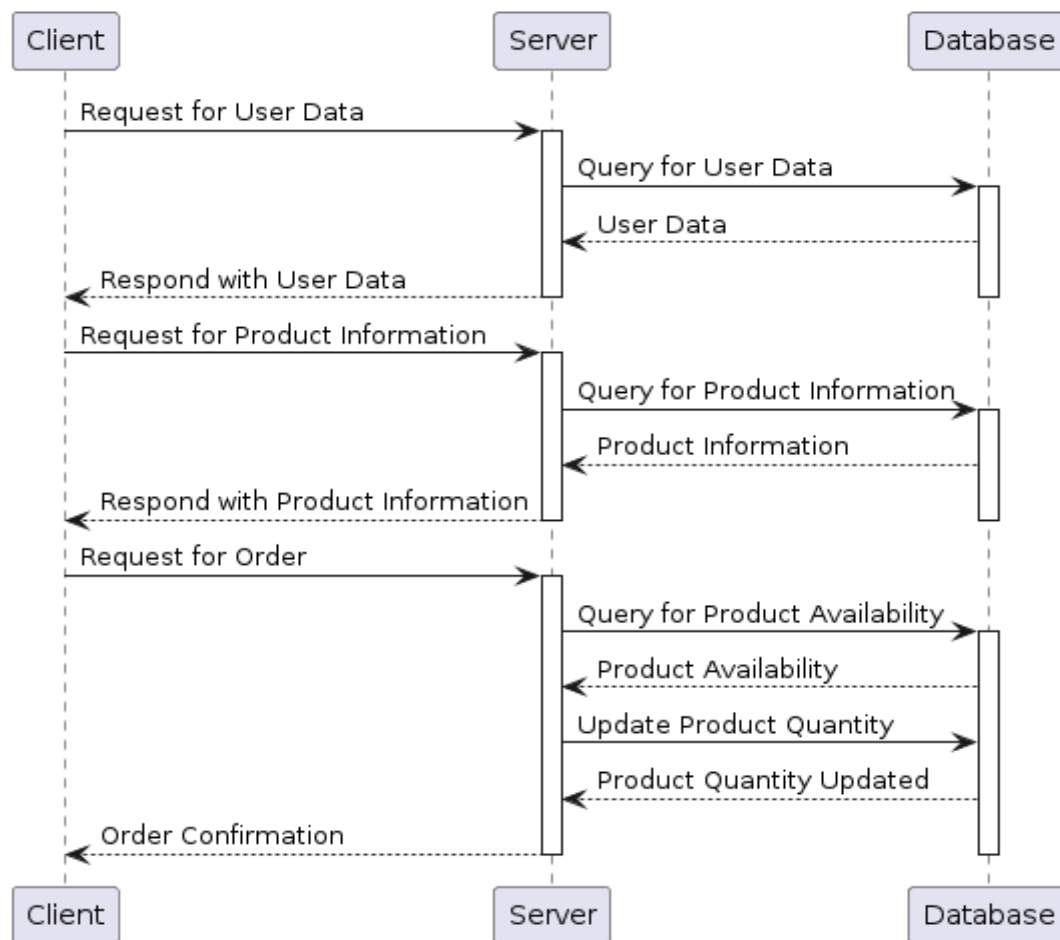


Figure 4.2 Sequence diagram

4.1.3 Activity Diagram

An activity diagram serves as a visual representation of a system's workflow, elucidating the sequence of activities or actions required to accomplish a specific task or goal. It offers a comprehensive depiction of the flow of control among various activities and decision points within a process. Whether modeling business processes, software workflows, or system behaviors, activity diagrams play a pivotal role in clarifying and illustrating complex processes. In essence, activity diagrams serve as a valuable tool for analyzing, designing, and documenting processes, enabling stakeholders to visualize and understand the intricacies of system workflows.

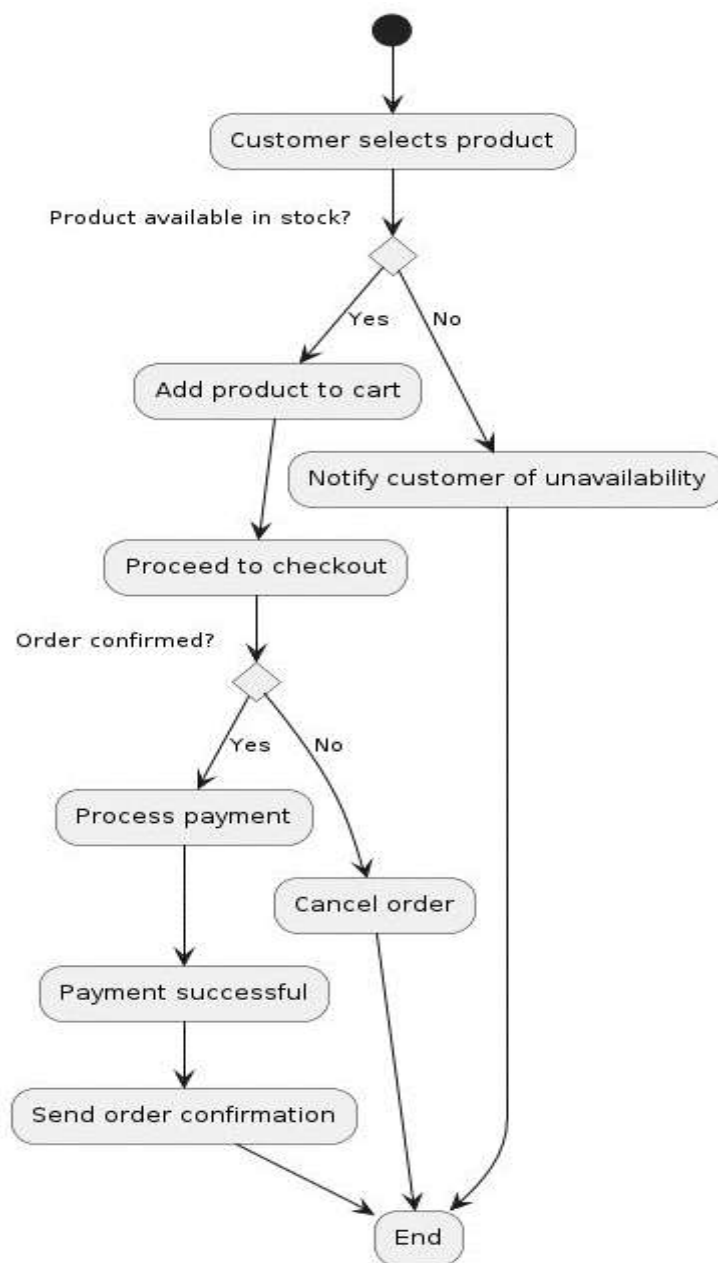


Figure 4.3 Activity diagram for to Order a Product

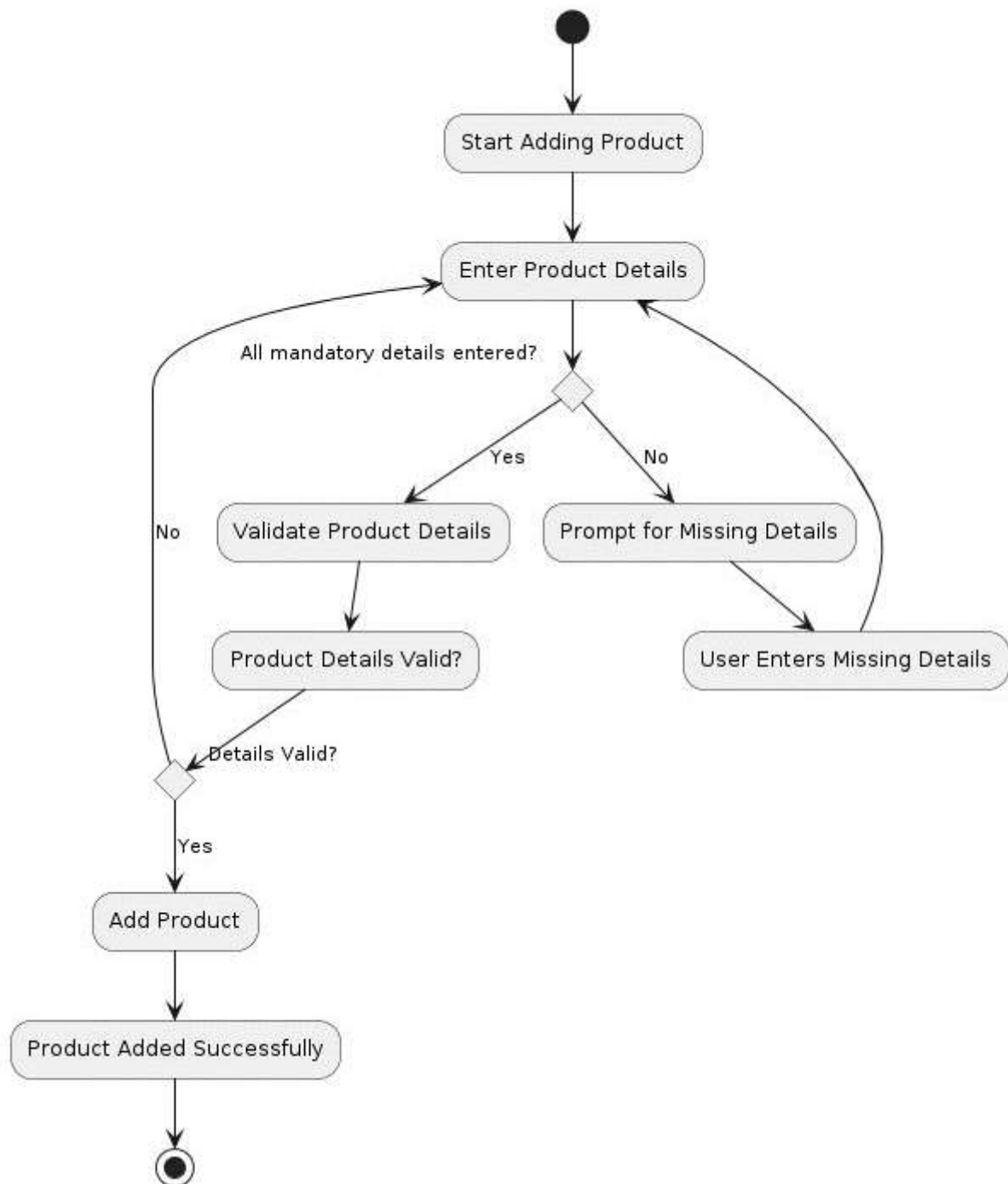


Figure 4.4 Activity diagram for to Add a Product

4.1.4 ER Diagram

An Entity-Relationship (ER) diagram is a visual representation of the structure of a database, illustrating the entities, their attributes, and the relationships between them. It helps to design and understand the logical structure of a database before its implementation. Entities represent objects or concepts, while relationships depict connections or associations between entities. Attributes describe the properties or characteristics of entities. ER diagrams are crucial for database modeling and serve as a blueprint for database development.

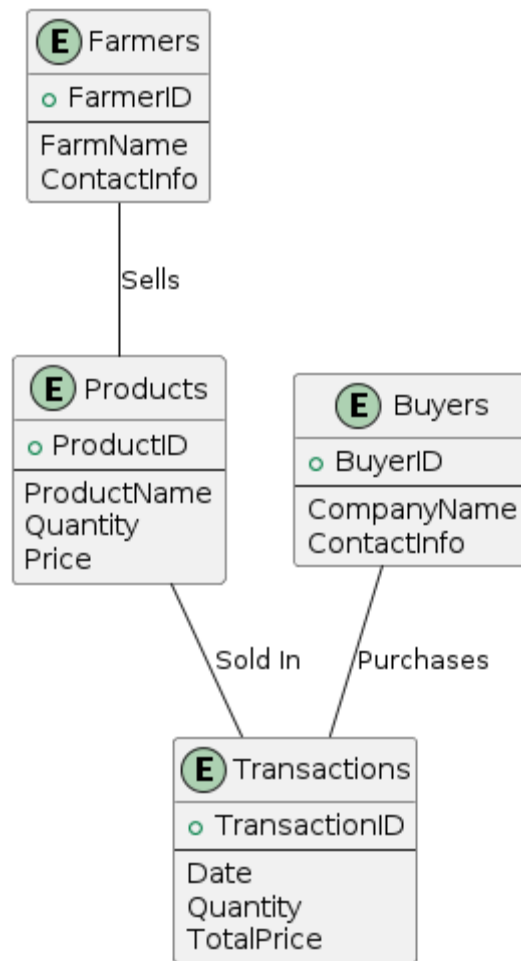


Figure 4.5 ER diagram for KisanRaj

CHAPTER 5

PROJECT PLANNING

5.1 Proposed System

The proposed system "KisanRaj" is designed to modernize the Indian agricultural supply chain through an innovative online platform. Our solution aims to bridge the gap between farmers and buyers by facilitating direct communication and transparent transactions. The proposed system "KisanRaj" is designed to modernize the Indian agricultural supply chain through an innovative online platform. Our solution aims to bridge the gap between farmers and buyers by facilitating direct communication and transparent transactions.

5.1.1 System Architecture

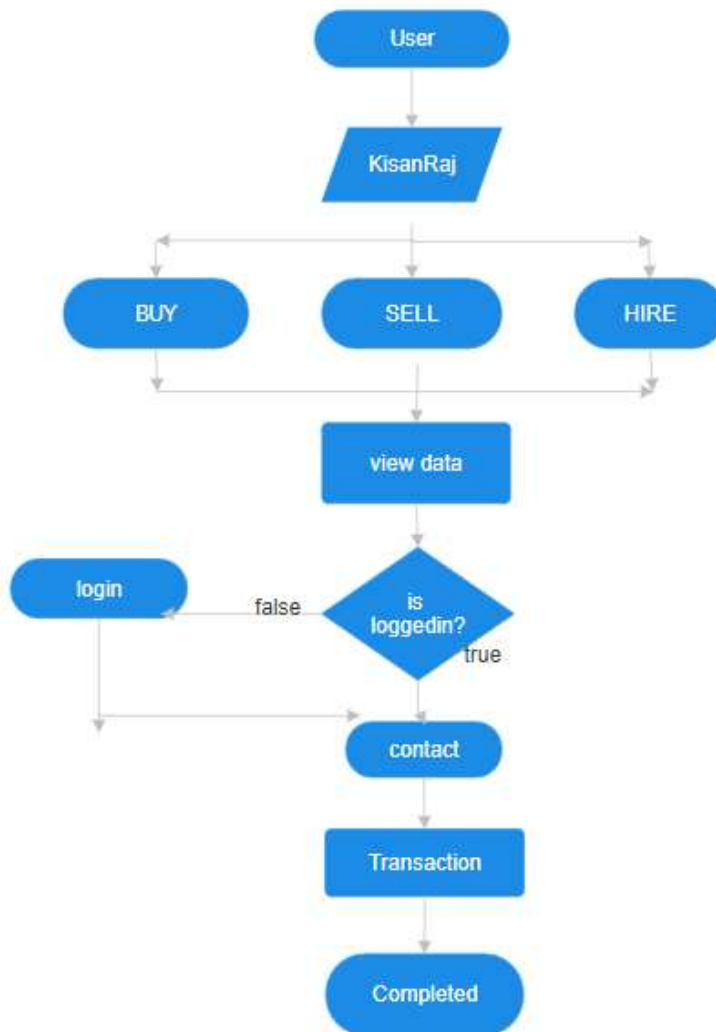


Figure 5.1 System Architecture

5.1.2 Flow Chart Representation

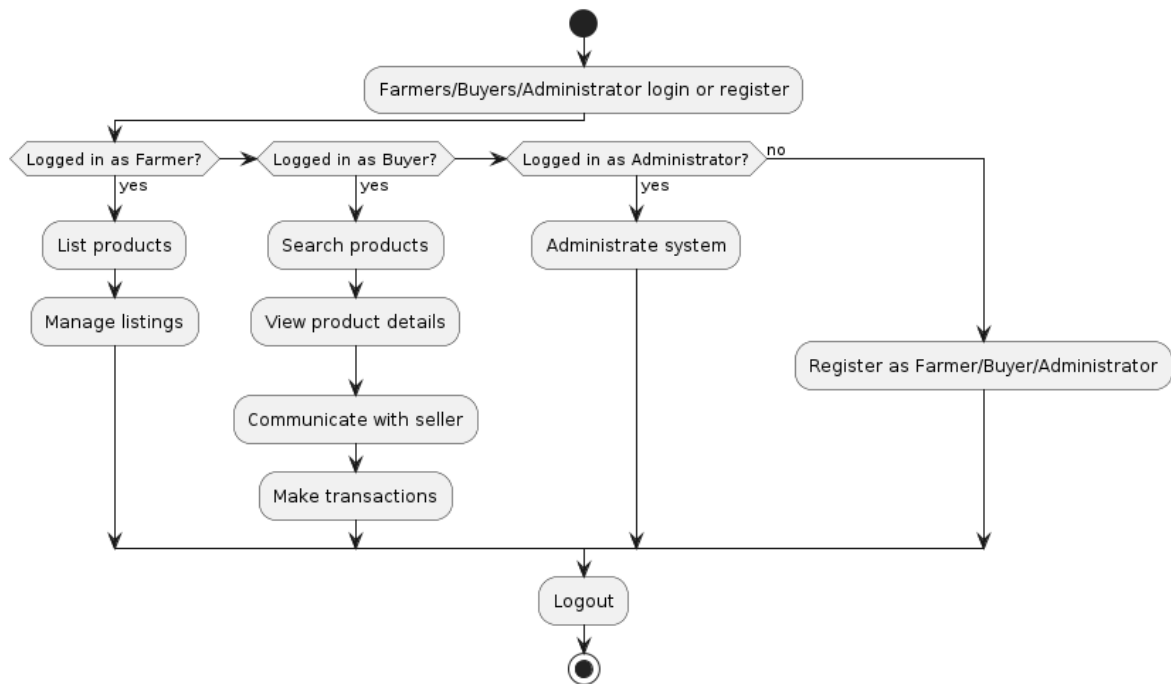


Figure 5.2 Flow chart for KisanRaj

5.2 Modules

5.2.1 Buy Commodity

A buyer or farmer may need farm products and be prepared to buy them in a few days or months. In this instance, a farmer should be given the ability to view every product that is for sale from every seller, as well as a section where they can filter through all of the products and look for the specific item they need based on factors like category, price, and seller's distance from them, delivery time, quality, and seller rating

- **Product Listing:** Displays all available products for purchase.
- **Search Bar:** Allows users to search for specific products.
- **Filtering:** Enables filtering of products based on various criteria.
- **Card View:** Each product is presented in a visually appealing card format.
- **User-Friendly Interface:** Provides an intuitive interface for easy navigation and product discovery.

5.2.2 Sell Commodity

Enables farmers to list their farm products for sale on the platform. Similar to how one purchase a commodity, a seller must put something for sale. The seller must now have the ability to list a product for sale along with information about the product, including a photo, quantity available, quality, projected date of availability, price, and location of availability.

- **Sell Commodity Form:** A form for users to input details about the commodity they wish to sell.

- **Data Storage:** Upon submission, the entered data, including product and user details, is stored in the database.
- **Product Details:** Captures information such as product type, quantity, quality, and price.
- **User Details:** Collects information about the seller, including contact details and location.
- **Database Integration:** Integrates with the backend database to ensure secure and efficient data storage.

5.2.3 Listings Posted

The farmers can view the products they have uploaded for sale on the platform. This section serves as a centralized hub for sellers to manage their inventory and ensure that their offerings are accurately represented on the platform. Additionally, it allows farmers to make necessary updates or modifications to their listings as needed.

- **Listings Posted:** Displays a list of items that users have posted for sale.
- **User Listings:** Each item is associated with the user who posted it.
- **Product Details:** Shows details of the posted items, including product type, quantity, quality, and price.
- **Status Updates:** Users can view the status of their listings, such as active or inactive.
- **Edit and Delete Options:** Provides options for users to edit or delete their posted listings as needed.

5.2.4 Responses Received

The "Responses Received" section presents sellers with a summary of the responses or offers they have received for the products they have listed on the platform. When sellers upload products for sale, potential buyers have the opportunity to interact with these listings by making inquiries, submitting offers, or expressing interest in purchasing.

- **Buyer Inquiries:** Displays inquiries received from potential buyers.
- **Inquiry Details:** Includes details such as buyer name, contact information, and message.
- **Product of Interest:** Specifies the product(s) the buyer is interested in.
- **Response Handling:** Provides options for the seller to respond to inquiries, such as accepting, declining, or negotiating offers.
- **Communication Platform:** Facilitates communication between sellers and potential buyers for further negotiation and transaction arrangements.

5.2.5 Post potential requirements

Sometimes a buyer needs a certain product, but it's not available on the platform. In this situation, the buyer posts his criteria, and the seller keeps an eye out to see if any other buyers have set requirements for any of the products. For ease of communication with customers who have previously expressed a desire to buy, the seller should also have a function whereby, whenever he

lists a product, all possible customers for that specific product should be displayed to him. Let's buyers post their requirements for specific farm products they are looking to purchase.

- **User Requirements:** Enables users to submit their product requirements.
- **Listing Requirements:** Allows sellers to view and list all requirements posted by users.
- **Requirement Details:** Displays details of each requirement, including product specifications, quantity, and any additional notes provided by the user.
- **Matching Algorithm:** Utilizes a matching algorithm to connect user requirements with suitable sellers, facilitating efficient transactions.
- **Transaction Initiation:** Provides a platform for initiating transactions based on matched requirements, promoting trade between users.

5.2.6 Negotiation History

After a buyer selects a product from the platform's inventory, he must get in touch with the seller for more information and to negotiate a better price. In order to facilitate real-time communication for farmers, a chat window may be a good idea. This will enable direct communication between buyers and sellers and ensure that the seller receives fair compensation through negotiations. Records the negotiation process between buyers and sellers, capturing the exchange of offers, counteroffers, and agreements made during the transaction.

- **Record Offer Exchanges:** Capture offers and counteroffers during negotiations.
- **Timeline View:** Track negotiation progression through a chronological timeline.
- **Message History:** Access a complete record of communication during negotiations.
- **Offer Management:** Manage and modify offers before final agreement.
- **Negotiation Resolution:** Finalize negotiations and initiate transactions.

5.2.7 Transaction History

Once the negotiation is completed between buyer and seller, the buyer and seller needs to track the product and transaction need to be completed. This feature is also mandatory to be provided. Tracks the history of completed transactions between buyers and sellers, providing a record of purchases and sales.

- **View Transaction Details:** Access detailed information about past transactions.
- **Filter Transactions:** Filter transaction history based on criteria like date and product category.
- **Export Transaction Data:** Download transaction history for record-keeping.
- **Transaction Status Tracking:** Receive real-time updates on transaction statuses.
- **Transaction Analytics:** Analyze transaction data for insights into sales performance.

5.2.8 Hire Equipment

When growing crops, farmers require a lot of equipment and are always looking for ways to get it. This can occasionally be an issue if the farmer is unaware of where to find information about the equipment that is closest to them. In order to make farming easier and prevent time wastage, a farmer has to have access to all relevant data and equipment. Facilitates the hiring of farming equipment by users, allowing farmers to rent out their equipment and enabling others to find and hire equipment as needed.

- **Browse Equipment Listings:** Explore available farming equipment for rent.
- **Equipment Details:** Access detailed information about each equipment listing.
- **Request Rental:** Submit rental requests for desired equipment.
- **Owner Communication:** Communicate with equipment owners for rental arrangements.
- **Rental Management:** Manage rentals, payments, and feedback.

5.2.9 Post Equipment

The "Post Equipment" feature allows users to share details about farming equipment they possess and are willing to rent out to others through the platform. This functionality serves as a marketplace for agricultural machinery, enabling users to list various types of equipment such as tractors, plows, harvesters, and more. Users can provide comprehensive information about the equipment they wish to rent out, including its specifications, availability, rental terms, and any additional notes or conditions.

- **Create Equipment Listings:** List farming equipment for rent with detailed information.
- **Edit/Delete Listings:** Modify or remove equipment listings as needed.
- **Rental Inquiry Management:** Manage rental inquiries and negotiate terms.
- **Rental Agreement Generation:** Generate rental agreements for finalized rentals.
- **Feedback and Ratings:** Leave feedback and ratings after rental completion.

5.2.10 Cold Storages

Farmers often want to store their products if they can't sell them at a good price right away. This way, they can sell them later, maybe during the next crop season. But to do this, they need to know where they can find cold storage places nearby. These places keep their products fresh for longer periods until they're ready to sell them.

- **Search Cold Storage Listings:** Search for cold storage facilities based on location and capacity.
- **View Facility Details:** Access detailed information about each cold storage facility.
- **Contact Facility Owner:** Communicate directly with facility owners for inquiries.
- **Booking Management:** Manage cold storage bookings, payments, and confirmations.
- **Review and Feedback:** Leave reviews and feedback based on cold storage experience.

CHAPTER 6

IMPLEMENTATION

6.1 Authentication

6.1.1 Register.js

```
import React, { useState } from "react";
import Header from "../../components/layouts/Header";
import Footer from "../../components/layouts/Footer";
import axios from "axios";
import { NavLink, useNavigate } from "react-router-dom";
import toast, { Toaster } from "react-hot-toast";
import Navbar from "../../components/UIComponents/Navbar";
import Nav from "../../components/UIComponents/Nav";
import "./Register.css"
export const Register = () => {
  const [name, setName] = useState("");
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [phone, setPhone] = useState("");
  const [answer, setAnswer] = useState("");
  const [address, setAddress] = useState("");
  const [pincode, setPincode] = useState("");
  const [latitude, setLatitude] = useState("");
  const [longitude, setLongitude] = useState("");
  const navigate = useNavigate();
  const handleSubmit = async (e) => {
    getLocation();
    if (pincode.length > 6) {
      toast("Please enter valid pincode");
    } else {
      e.preventDefault();
      try {
        const res = await axios.post(
          `${process.env.REACT_APP_API}/api/v1/auth/register`,
          {
```

```

    name,
    email,
    password,
    phone,
    answer,
    address,
    pincode,
    latitude,
    longitude,
  }
);
if (res.data.success) {
  toast.success(res.data.message);
  navigate("/login");
} else {
  toast.error(res.data.message);
}
} catch (error) {
  console.error(error);
  toast.error("Error in registration!");
}
}
};

const getLocation = async() => {
  if (navigator.geolocation) {
    await navigator.geolocation.getCurrentPosition(
      (position) => {
        setLatitude(position.coords.latitude);
        setLongitude(position.coords.longitude);
      },
      (error) => {
        console.error("Error getting location:", error.message);
      }
    );
  } else {
    console.error("Geolocation is not supported by this browser.");
  }
}

```

```

    });
    return (
      <>
        <Nav />
        <Toaster />
        <div className="container">
          <div className="card bg-light">
            <article className="card-body mx-auto" style={{ maxWidth: "400px" }}>
              <h4 className="card-title mt-1 text-center">Create Account</h4>
              <p className="text-center">Get started with your free account</p>
              <form onSubmit={handleSubmit}>
                <div className="form-group input-group">
                  <div className="input-group-prepend">
                    <span className="input-group-text"> <i className="fa fa-user"></i> </span>
                  </div>
                  <input value={name} onChange={(e)=>setName(e.target.value)} className="form-
control" placeholder="Full name" type="text" />
                </div>
                <div className="form-group input-group">
                  <div className="input-group-prepend">
                    <span className="input-group-text"> <i className="fa fa-envelope"></i> </span>
                  </div>
                  <input name="" value={email} onChange={(e)=>setEmail(e.target.value)}
className="form-control" placeholder="Email address" type="email" />
                </div>
                <div className="form-group input-group">
                  <div className="input-group-prepend">
                    <span className="input-group-text"> <i className="fa fa-phone"></i> </span>
                  </div>
                  <input name="" value={" +91"} className="form-control" placeholder="+91"
style={{ maxWidth:"60px" }} />
                  <input name="" value={phone} onChange={(e)=>setPhone(e.target.value)}
className="form-control" placeholder="Enter phone" type="number" />
                </div>
                <div className="form-group input-group">
                  <div className="input-group-prepend">

```

```

        <span className="input-group-text"> <i className="fa fa-building"></i> </span>
    </div>

    <input value={ address } onChange={ (e)=>setAddress(e.target.value)} className="form-
control" placeholder="Enter address" type="text" />
</div>

<div className="form-group input-group">
    <div className="input-group-prepend">
        <span className="input-group-text"> <i className="fa fa-location-dot"></i> </span>
    </div>

    <input value={ pincode } onChange={ (e)=>setPincode(e.target.value)} className="form-
control" placeholder="Enter pincode" type="text" />

</div>

<div className="form-group input-group">
    <div className="input-group-prepend">
        <span className="input-group-text"> <i className="fa fa-lock"></i> </span>
    </div>

    <input className="form-control" value={ password }
onChange={ (e)=>setPassword(e.target.value)} placeholder="Create password" type="password" />
</div>

<div className="form-group input-group">
    <div className="input-group-prepend">
        <span className="input-group-text"> <i className="fa fa-lock"></i> </span>
    </div>

    <input className="form-control" value={ answer }
onChange={ (e)=>setAnswer(e.target.value)} placeholder="Create secret code" type="password" />
</div>

<div className="form-group">
    <button type="submit" className="btn btn-primary btn-block"> Create Account
    </button>
</div>

<p className="text-center">Have an account?<NavLink to="/login"> Log In
</NavLink></p>
</form>
</article>
</div>

```



```

    </div>
    <Footer />
  </>
);
};

```

6.1.2 Login.js

```

import React, { useState } from "react";
import Header from "../../components/layouts/Header";
import Footer from "../../components/layouts/Footer";
import axios from "axios"
import { useContext } from 'react';
import { useNavigate, useLocation, NavLink } from "react-router-dom";
import toast from "react-hot-toast";
import AuthContext from "../../context/AuthContext";
import Navbar from "../../components/UIComponents/Navbar";
import Nav from "../../components/UIComponents/Nav";
export const Login = () => {
  const [email, setemail] = useState("");
  const [password, setpassword] = useState("");
  const location = useLocation();
  const [auth, setAuth] = useContext(AuthContext);
  const navigate = useNavigate();
  const handlesubmit = async (e) => {
    e.preventDefault();
    try {
      const res = await axios.post(`${process.env.REACT_APP_API}/api/v1/auth/login`, { email,
password });
      if (res && res.data.success) {
        toast.success(res.data.message)
        setAuth({
          ...auth,
          user: res.data.user,
          token: res.data.token
        })
        localStorage.setItem("auth", JSON.stringify(res.data))
        navigate(location.state || "/")
      }
    }
  }
}

```

```

    }
    else {
        toast.error(res.data.message)
    }
} catch (error) {
    console.log(error)
    toast.error("Please check credentials")
}
}
return (
    <>
    <Nav />
    <div className="d-flex justify-content-between">
        <div className=""></div>
        <div className="my-2" style={{ marginTop: "10vh" }}>
            <div className="register my-3">
                <h1 className="my-2 text-center">Login user</h1>
                <form className="text-center m-3" onSubmit={handlesubmit}>
                    <div className="mb-3">
                        <input
                            type="text"
                            className="form-control"
                            value={email}
                            onChange={(e) => { setemail(e.target.value) }}
                            placeholder="Enter Email"
                            required
                        />
                    </div>
                    <div className="mb-3">
                        <input
                            type="password"
                            className="form-control"
                            value={password}
                            onChange={(e) => { setpassword(e.target.value) }}
                            placeholder="Enter password"
                            required

```

```

    />
  </div>
  <button type="submit" className="btn btn-primary">
    Login
  </button>
  <p className="text-center mt-3"><NavLink to="/register"> Forgot
passssword?</NavLink></p>
  <div className="m-3">
    <p className="divider-text">
      <span className="bg-light">OR</span>
    </p>
    <p className="text-center">Dont Have an account?<NavLink to="/register"> Create
Account</NavLink></p>
  </div>
</form>
</div>
</div>
<div className=""></div>
</div>
<Footer />
</>
);};

```

6.2 Homepage.js

```

import React, { useContext, useState } from "react";
import Header from "../components/layouts/Header";
import Footer from "../components/layouts/Footer";
import { Toaster } from "react-hot-toast";
import "../Homepage.css";
import { useNavigate } from "react-router-dom";
import Navbar from "../components/UIComponents/Navbar";
import Nav from "../components/UIComponents/Nav";
import AuthContext from "../context/AuthContext";
import HomeSummaryCards from
"../components/CardRelated/SummaryCards/HomeSummaryCards";
import CategoriesCardHome from
"../components/CardRelated/SummaryCards/CategoriesCardHome";

```

```

import products from "../src/Data/MOCK_DATA"
import commodities from "../Data/Commodities";
import "../Hero.css"
import random from "random-int"
import slugify from "slugify"
export const Homepage = () => {
  const navigate = useNavigate();
  const [auth, setAuth] = useContext(AuthContext);
  const [value, setvalue] = useState("")
  const [cat, setcat] = useState("")
  const [id, setid] = useState("")
  return (
    <
      <Nav />
      <Toaster />
      <div className="hero" style={{ display: 'flex', flexDirection: 'column', alignItems: 'center',
justifyContent: 'center', textAlign: 'center' }}>
        
        <div className="hero-text">
          <h1 className="text-warning">KisanRaj<br />Buy and Sell in advance</h1>
          <input
            type="search"
            value={value}
            placeholder={`Try Searching for "${commodities[random(1, 190)].name}`} `}
            className="loc"
            onChange={(e) => setvalue(e.target.value)}
            list={value.length >= 2 ? "commodities" : null} // Render datalist if value length is at least 3
          />
          {value.length >= 2 && (
            <datalist id="commodities">
              {commodities
                .filter((product) => product.name.toLowerCase().startsWith(value.toLowerCase())) //

```

Filter options based on starting characters

```

        .map((product) => (
          <option key={product._id} value={product.name} />
        )))
    </datalist>
  )}
  <div className="buttons">
    <button
      className="btn buy btn-outline-info mx-2 my-2"
      onClick={() => {
        if (auth?.user) {
          if (value) {
            const selectedProduct = commodities.find(product => product.name.toLowerCase()
            === value.toLowerCase());
            if (selectedProduct) {
              const { category, slug } = selectedProduct;
              navigate(`/dashboard/user/buy-
commodity/${slugify(category.toLowerCase())}/${slugify(value.toLowerCase())}`);
            } else {
              navigate("/dashboard/user/buy-commodity/all");
            }
          } else {
            navigate("/dashboard/user/buy-commodity/all");
          }
        } else {
          navigate("/buy-commodity");
        }
      }}
    >
    Buy
  </button>
  <button
    className="btn buy btn-outline-info mx-2 my-2"
    onClick={() => {
      auth?.user && navigate("/dashboard/user/sell-commodity")
    }}
  >
  Sell
</button>

```

```

    >
    Sell
  </button>

  <button className="btn buy btn-outline-info mx-2 my-2" onClick={() =>
navigate("/dashboard/user/hire-equipment")}>Hire</button>
  </div>
</div>
</div >
  <CategoriesCardHome />
  <Footer />
</>
);
};

```

6.3 SellCommodity.js

```

import React, { useState, useContext, useEffect } from 'react';
import Footer from '../components/layouts/Footer';
import { Radio } from 'antd';
import axios from 'axios';
import toast from 'react-hot-toast';
import { NavLink, useNavigate } from 'react-router-dom';
import AuthContext from '../context/AuthContext';
import commodities from "../../Data/Commodities"; // Import the data from Commodities.js
import Nav from '../components/UIComponents/Nav';
const SellCommodity = () => {
  const navigate = useNavigate();
  const [photo, setPhoto] = useState("");
  const [name, setName] = useState("");
  const [description, setDescription] = useState("");
  const [price, setPrice] = useState("");
  const [availableDate, setAvailableDate] = useState("");
  const [organic, setOrganic] = useState("")
  const [quantity, setQuantity] = useState("");
  const [shipping, setShipping] = useState("");
  const [quantityUnit, setQuantityUnit] = useState("")
  const [commodityId, setCommodityId] = useState("")
  const [suggestions, setSuggestions] = useState([]);

```

```

const [isFocused, setIsFocused] = useState(false);
const [auth] = useContext(AuthContext);
const filterSuggestions = (input) => {
  const filtered = commodities.filter(product =>
product.name.toLowerCase().includes(input.toLowerCase()));
  setSuggestions(filtered.map(product => product.name));
  /// i want to assign commodityId to product._id
  setCommodityId(filtered.map(product => product._id))
}
useEffect(() => {
  if (name && isFocused) { // Only filter suggestions when name is not empty and input is focused
    filterSuggestions(name);
  } else {
    setSuggestions([]); // Clear suggestions when name is empty or input is not focused
  }
}, [name, isFocused]);
const handleSelect = (suggest) => {
  setName(suggest);
  setSuggestions([]); // Clear suggestions when item is selected
};
const handleBlur = () => {
  setTimeout(() => {
    setIsFocused(false); // Set input focus state to false after a delay
  }, 200);
};
const handleCreate = async (e) => {
  e.preventDefault();
  try {
    const productData = new FormData();
    const sellerId=auth?.user?._id;
    productData.append("name", name);
    productData.append("description", description);
    productData.append("price", price);
    productData.append("quantity", quantity);
    productData.append("photo", photo);
    productData.append("shipping", shipping);

```

```

productData.append("sellerId", sellerId)
productData.append("availableDate", availableDate)
productData.append("organic", organic)
productData.append("quantityUnit", quantityUnit)
productData.append("commodityId", commodityId[0])
const { data } = await axios.post(`${process.env.REACT_APP_API}/api/v1/products/create-product`, productData);
if (data?.success) {
  toast.success("Product created successfully");
  navigate("/dashboard/user/listings-posted");
} else {
  toast.error(data?.message);
}
} catch (error) {
  console.log(error);
  toast.error("Error occurred while creating product here");
}
}
const Breadcrumb = () => {
  return (
    <nav aria-label="breadcrumb">
      <ol class="breadcrumb">
        <li class="breadcrumb-item"><NavLink to="/">Home</NavLink></li>

        <li class="breadcrumb-item active" aria-current="page">sell-commodity</li>
      </ol>
    </nav>
  );
};
return (
  <>
    <Nav />
    <Breadcrumb/>
    <div className="container-fluid m-3 p-3 d-flex justify-content-center">
      <div className="col-md-8 text-center m-1" style={{ minHeight: "50vh" }}>
        <h3>Fill Product details to post</h3>

```



```

<div className="m-1">
  <div className="mb-3">
    <label className='btn btn-primary btn-prinary'>
      {photo ? photo.name : "Upload photo"}
      <input type="file" name="photo" id="" accept='image/*' onChange={(e) =>
setPhoto(e.target.files[0])} hidden />
    </label>
  </div>
  {photo && (
    <div className="mb-3 text-center">
      <img src={URL.createObjectURL(photo)} alt="Product "
style={{ height:"200px",width:"300px", objectFit:"cover" }} className="img img-responsive" />
    </div>
  )}
  { /* Form Inputs */}
  <div className="mb-3">
    <input
      type="text"
      value={name}
      placeholder='Enter name'
      className='form-control'
      onChange={(e) => setName(e.target.value)}
      onFocus={() => setIsFocused(true)} // Add onFocus event handler to set focus state to
true
      onBlur={handleBlur} // Add onBlur event handler
    />
    {isFocused && suggestions.length > 0 && name && (
      <ul style={{ listStyle: "none" }}>
        {suggestions.map((suggest, index) => (
          <li key={index} className='bg-info p-1 m-1' onClick={() => handleSelect(suggest)}>
            {suggest}
          </li>
        ))}
      </ul>
    )}
  </div>

```

```

<div className="mb-3 d-flex align-items-center">
  <input type="number" value={price} placeholder='Rs. Price ' className='form-control'
onChange={(e) => setPrice(e.target.value)} />
  <span className='m-3' style={{ fontWeight: "600" }}>per</span>
  <Radio.Group onChange={(e) => setQuantityUnit(e.target.value)} value={quantityUnit}
className="d-flex align-items-center">
    <Radio value={ "ton" } className="me-3">ton</Radio>
    <Radio value={ "box" } className="me-3">box</Radio>
    <Radio value={ "quintal" } className="me-3">quintal</Radio>
    <Radio value={ "dozen" } className="me-3">dozen</Radio>
    <Radio value={ "kg" }>kg</Radio>
    <Radio value={ "item" }>item</Radio>
    <Radio value={ "bag" }>bag</Radio>
  </Radio.Group>
</div>

<div className="d-flex align-items-center">
  <input type="number" value={quantity} placeholder='Total quantity available'
className='form-control me-2' onChange={(e) => setQuantity(e.target.value)} />
  <label className='m-2' htmlFor="">{quantityUnit}s</label>
</div>

<div className="m-4 d-flex align-items-center">
  <label htmlFor="" className="m-0 me-3 text-dark" style={{ fontWeight: "600"
}}>Available by :</label>
  <div className="dater">
    <input type="date" value={availableDate} placeholder='Available date'
className='form-control' onChange={(e) => setAvailableDate(e.target.value)} />
  </div>
  <div className="data">
    <label htmlFor="" className='m-4 text-dark' style={{ fontWeight: "600" }}>Is shipping
available?</label>
    <Radio.Group onChange={(e) => setShipping(e.target.value)} value={shipping}>
      <Radio value={1}>Yes</Radio>
      <Radio value={0}>No</Radio>
    </Radio.Group>
    <label htmlFor="" className='m-4 text-dark' style={{ fontWeight: "600"
}}>Organic</label>

```

```

    <Radio.Group onChange={(e) => setOrganic(e.target.value)} value={organic}>
      <Radio value={1}>Yes</Radio>
      <Radio value={0}>No</Radio>
    </Radio.Group>
  </div>
</div>
<div className="mb-3">
  <input type="text" value={description} placeholder='Enter description' className='form-control' onChange={(e) => setDescription(e.target.value)} />
</div>
<div className="mb-3">
  <button onClick={handleCreate} className='btn btn-primary'>Create
Commodity</button>
</div>
</div>
</div>
</div>
<Footer />
</>
)
}export default SellCommodity;

```

6.4 BuyCommodity.js

```

import React, { useContext, useEffect, useState } from "react";
import Spinner from "../../components/UIComponents/Spinner";
import Footer from "../../components/layouts/Footer";
import Filtersbar from "../../components/Filters/Filtersbar";
import AuthContext from "../../context/AuthContext";
import toast from "react-hot-toast";
import axios from "axios";
import commodities from "../../Data/Commodities";
import Nav from "../../components/UIComponents/Nav";
import { NavLink, useNavigate } from "react-router-dom";
import ProductCard, { Prod } from "../../components/CardRelated/buycommodity/ProductCard";
const BuyCommodity = () => {
  const [filteredProducts, setFilteredProducts] = useState([]); // To store filtered products
  const [loading, setLoading] = useState(true)

```

```

const [searchitem, setSearchitem] = useState("");
const [auth] = useContext(AuthContext);
const [quantity, setquantity] = useState("");
const [price, setprice] = useState("");
const [date, setdate] = useState("");
const [notes, setnotes] = useState("");
const [isFocused, setIsFocused] = useState(false);
const [proposedlist, setProposedlist] = useState([]);
const [products, setProducts] = useState([]);
const handleCategoryFilter = (selectedCategory) => {
  if (selectedCategory) {
    const filtered = products.filter((p) => {
      const category = p?.commodityId?.category;
      return category && category.toLowerCase() === selectedCategory.toLowerCase();
    });
    if (filtered.length > 0) {
    } else {
      setFilteredProducts([])
    }
    setFilteredProducts(filtered);
  } else {
    setFilteredProducts(products)
  }
};
const filterSuggestions = (input) => {
  const filtered = commodities.filter((product) =>
    product.name.toLowerCase().includes(input.toLowerCase())
  );
};
useEffect(() => {
  if (searchitem && isFocused) {
    filterSuggestions(searchitem);
  } else {
  }
}, [searchitem, isFocused]);
const proposeOffer = async (pid, sellerId) => {

```

```

const sentBy = auth?.user?._id;
const buyerId = sentBy;
const productId = pid;
try {
  const res = await axios.post(`${process.env.REACT_APP_API}/api/v1/requirements/propose-
offer`, { quantity, price, notes, date, sentBy, buyerId, productId, sellerId })
  if (res.data.success) {
    setquantity("")
    setprice("")
    setnotes("")
    setdate("")
    setProposedlist([...proposedlist, pid]);
    toast.success("offer proposed!")
  }
} catch (error) {
  console.log(error)
}
finally {
  setLoading(false)
}
}

const getUserdata = async () => {
  // setLoading(true)
  const uid = auth?.user?._id;
  try {
    const userdata = await
axios.get(`${process.env.REACT_APP_API}/api/v1/users/profile/${uid}`);
    if (userdata.data.success) {
      console.log(userdata.data.user.wishlisted.length)
    }
  } catch (error) {
    console.log(error)
  }
  finally {
    setLoading(false)
  }
}

```

```

    }
    useEffect(()=>{
      getUserdata()
    },[])
    const getProposedList = async () => {
      try {
        const userid = auth?.user?._id;
        const { data } = await
        axios.post(`${process.env.REACT_APP_API}/api/v1/products/proposedlist`, { userid });
        if (data?.success) {
          setProposedlist(data?.proposedList);
        }
      } catch (error) {
        console.error("Error fetching proposed list:", error);
      }
      finally {
        setLoading(false)
      }
    };
    useEffect(() => {
      getProposedList();
    }, []);
    const handleDecline = async (pid, sellerid) => {
      try {
        console.log("clicked on hndledecline")
        const buyerid = auth?.user._id;
        const res = await axios.post(`${process.env.REACT_APP_API}/api/v1/requirements/decline`,
        { buyerid, pid, sellerid });
        console.log(res)
        if (res?.data?.success) {
          toast.success("Offer declined!");
          setProposedlist(proposedlist.filter((id) => id !== pid));
        }
      } catch (error) {
        console.log(error);
        toast.error("Decline failed!");
      }
    }
  }

```

```

    finally {
      setLoading(false)}
  };
  const getAllProducts = async () => {
    try {
      if (auth?.user) {
        const { data } = await axios.get(`${process.env.REACT_APP_API}/api/v1/products/get-all-products`);
        if (data?.success) {
          setProducts(data?.products);
        }
      }
    } else {
      const { data } = await axios.get(`${process.env.REACT_APP_API}/api/v1/products/get-all-product`); //products not posted by him and posted by others
      if (data?.success) {
        setProducts(data?.products);
      }
    }
  } catch (error) {
    console.log(error);
    toast.error("Something went wrong in getting all products!");
  }
  finally {
    setLoading(false)
  }
};

useEffect(() => {
  getAllProducts();
}, []);

const handleProductFilter = (productName) => {
  const filtered = products.filter((product) => product.name === productName);
  setFilteredProducts(filtered);
};

if (loading) {
  return (<>

```

```

    <Nav />
    <div className="container" style={{ minHeight: "50vh" }}>
      <Spinner />
    </div>
    <Footer />
  </>
)}

const Breadcrumb = () => {
  return (
    <nav aria-label="breadcrumb">
      <ol className="breadcrumb">
        <li className="breadcrumb-item"><NavLink to="/">Home</NavLink></li>
        <li className="breadcrumb-item active" aria-current="page">buy-commodity - all
categories</li>
      </ol>
    </nav>
  );
};

const FilterSearch = () => {
  return (
    <div className="d-flex align-items-center" style={{ display: 'flex', flexDirection: "row" }}>
      <input
        className="form-control mr-sm-2 m-3"
        type="search"
        placeholder={`search among ${products.length} products available`}
        value={searchitem}
        onChange={(e) => {
          setSearchitem(e.target.value);
        }}
        aria-label="Search"
        onFocus={() => setIsFocused(true)}
        // onBlur={handleBlur}
      />
      {searchitem && ( // Render the cross button only when search item is not empty
        <button

```



```

        className="fa-solid fa-multiply btn-sm btn"
        onClick={() => {
            setSearchitem("")
            setFilteredProducts([])
        }} // Clear the search input when the cross button is clicked
    </button>
)}
<button
    className="btn btn-outline-info btn-sm m-3"
    onClick={() => {
        handleProductFilter(
            searchitem.charAt(0).toUpperCase() + searchitem.slice(1)
        );
    }}
>
    Search
</button>
</div>
</>
)}
return (
    <>
        <Nav />
        <Breadcrumb />
        <div className="row m-3" style={{ display: 'flex', flexDirection: 'row', justifyContent: 'space-around' }}>
            <div style={{ minHeight: "50vh", width: "100%" }}>
                <div className="container" style={{ position: "relative" }}>
                    <FilterSearch />
                </div>
                <div className="container" style={{ display: 'flex', flexDirection: "row" }}>
                    <div className="col-3" style={{ border: 'solid 1px' }}>
                        <Filtersbar onProductSelect={handleCategoryFilter} />
                    </div>
                    <div style={{ display: "flex", flexDirection: "row", flexWrap: "wrap" }} >
                        {filteredProducts.length > 0

```

```

      ? filteredProducts.map((p) => (
        auth?.user ? <ProductCard key={p._id} product={p} /> : <Prod key={p._id}
product={p} />
      ))
    : products
      .filter((p) => p.name.toLowerCase().includes(searchitem.toLowerCase()))
      .map((p) => auth?.user ? <ProductCard key={p._id} product={p} /> : <Prod
key={p._id} product={p} />)
    }
  </div>
</div>
</div>
</div>
<Footer />
</>
);};
export default BuyCommodity;
export const FilterSearch = ({ products, handleProductFilter }) => {
  const [searchitem, setSearchitem] = useState("");
  const [isFocused, setIsFocused] = useState(false);
  const clearSearch = () => {
    setSearchitem("");
    handleProductFilter("");
  };
  return (
    <div className="d-flex align-items-center" style={{ display: 'flex', flexDirection: "row" }}>
      <input
        className="form-control mr-sm-2 m-3"
        type="search"
        placeholder={`search among ${products.length} products available`}
        value={searchitem}
        onChange={(e) => setSearchitem(e.target.value)}
        aria-label="Search"
        onFocus={() => setIsFocused(true)}
      />
      {searchitem && (
        <button

```

```

        className="fa-solid fa-multiply btn-sm btn"
        onClick={clearSearch}
      ></button>
    )}
    <button
      className="btn btn-outline-info btn-sm m-3"
      onClick={() => {
        handleProductFilter(
          searchitem.charAt(0).toUpperCase() + searchitem.slice(1)
        );
      }}
    >
      Search
    </button>
  </div>
);
};

```

6.6 NegotiationHistory.js

```

import React, { useContext, useEffect, useState } from "react";
import Header from "../../components/layouts/Header";
import Footer from "../../components/layouts/Footer";
import UserMenu from "../user/UserMenu";
import axios from "axios";
import toast from "react-hot-toast";
import { format } from "date-fns"
import { useParams } from "react-router-dom";
import AuthContext from "../../context/AuthContext";
import { Button, Modal } from 'antd';
import Nav from "../../components/UIComponents/Nav";
const Responses = () => {
  const params = useParams();
  const [proposal, setProposal] = useState([]); // to propose a new offer by seller to buyer for
negotiation
  const [product, setProduct] = useState({}); //for product details maintenance
  const [leads, setLeads] = useState([]);

```

```

const [requirements, setRequirements] = useState([]); //to fetch all requirements from
requirements model

const [auth, setAuth] = useContext(AuthContext); //to maintain user state and logged-in details
for profile page

const [open, setOpen] = useState(false); //for modal in negotiate button

const [requirement, setRequirement] = useState("") //to attach details of requirement to it and use
eg: requirement.productId

const [responsesState, setResponsesState] = useState(true)
const [negotiation, setnegotiation] = useState("")
const [quantity, setquantity] = useState("")
const [price, setprice] = useState("")
const [date, setdate] = useState("")
const [notes, setnotes] = useState("")

const showModal = (r) => {
  setRequirement(r)
  setOpen(true);
};

const handleOk = async (pid, buyerid) => {
  try {
    const sellerId = auth?.user._id;
    const productId = pid;
    const buyerId = buyerid;
    const sentBy = auth?.user?._id;
    const { data: proposeData } = await
    axios.post(`${process.env.REACT_APP_API}/api/v1/products/propose`, { buyerId, productId,
    sellerId });

    const { data: requirementData } = await
    axios.post(`${process.env.REACT_APP_API}/api/v1/requirements/post-requirement`, { quantity,
    price, date, notes, buyerId, sellerId, productId, sentBy });

    if (proposeData?.success && requirementData?.success) {
      toast.success("Offer proposed!");
      setOpen(false);
      setquantity("")
      setprice("")
      setdate("")
      setnotes("")

```

```

    }
  } catch (error) {
    console.log(error);
    toast.error("Propose failed!");
  } };
const handleCancel = () => {
  setOpen(false);
};
const getRequirements = async () => {
  try {
    const sellerId = auth?.user?._id;
    const productId = params.pid;
    const { data } = await
axios.post(`${process.env.REACT_APP_API}/api/v1/requirements/get-requirement`, { sellerId,
productId });
    if (data?.success) {
      setRequirements(data.result);
    } else {
      console.log("Failed to fetch requirements");
    }
  } catch (error) {
    console.log(error);
  }
};
useEffect(() => {
  getRequirements();
}, []);
const getPotentials = async (productName) => {
  try {
    const response = await axios.post(`http://localhost:8000/api/v1/requirements/get-product-
potentials`, { productName });
    if (response?.data.success) {
      setLeads(response.data.potentials);
    } else {
      console.log("if else case");
    }
  }

```

```

    } catch (error) {
      console.log(error)
      console.log("something wrong");
    }
  };

  const formattedDate = product.createdAt ? format(new Date(product.createdAt), 'dd MMM
yyyy') : "";

  const pid = params.pid;
  const getProductData = async () => {
    try {
      const { data } = await axios.get(`${process.env.REACT_APP_API}/api/v1/products/get-
product/${pid}`);
      if (data?.success) {
        const proposalsReceived = data.product?.sellerId?.proposalsReceived?.[pid] || [];
        setProduct(data.product);
        setProposal(proposalsReceived);
      } else {
        console.log("Failed to get product data");
      }
    } catch (error) {
      console.log(error.message);
      toast.error("Failed to get product data responses");
    }
  };

  useEffect(() => {
    getProductData();
  }, []);

  return (
    <
    <Nav />
    <div className="container-fluid">
      <div className="row">

        <div style={{ minHeight: "50vh" }}>
          <div className="titler">
            <div className="topper p-3">

```

```

<div style={{ flexDirection: "column" }}>
  <p>
    <span className="text-primary m-2" style={{ fontWeight: "500",
fontSize: "1.5rem" }}>
      {product.name}
    </span>{ " "}
    <span className="text-warning bg-dark">Rs.{product.price}</span>
per { " "} {product.quantityUnit}
    <span className="bg-warning">{product.quantity}
{product.quantityUnit}</span> Lot id: <span className="text-danger">{product._id} </span>{ "
"}

    Posted on : <span className="text-info">{formattedDate}</span>{ " "}
  </p>
  <div className="buttons">
    <button className="btn btn-lg btn-warning m-1" style={{ width: "49% "
}} onClick={() => {
      setResponsesState(true)
    }}>
      Buyer Responses
    </button>
    <button className="btn btn-lg btn-warning m-1" style={{ width: "49% "
}} onClick={() => {
      setResponsesState(false)
      getPotentials(product.name);
    }}>
      Potential Leads
    </button>
  </div>
  <>
    {responsesState && (
      <div className="container">
        <h1 className="text-center">Responses</h1>
        <div className="row">
          {requirements.map((requirement, index) => (
            <div className="card p-1 mx-2 mb-2" key={index}>
              <div style={{ display: "flex", flexDirection: "row" }}>

```

```

<div className="p-2" style={{ width: "70%" }}>
  <div className="d-flex justify-content-between">
    <p>{product.name}</p>
    <p><i className="fa-regular fa-clock p-1">
</i>{format(new Date(requirement.date), 'dd MMM yy')}
    </p>
  </div>
  <div className="d-flex justify-content-between p-1"
  style={{ display: "flex", flexDirection: "row" }}>
    <p>Rs.{requirement.price} /-</p>
    <p>Quantity : {requirement.quantity}
      {product.quantityUnit}s</p>
    <p>Description : {requirement.notes}</p>
  </div>
</div>
<div className="card mx-2 p-1" style={{ width: "30%",
  display: "flex", flexDirection: "column" }}>
  <div style={{ display: "flex", flexDirection: "row" }} >
    <div style={{ width: "30%" }}>
      <i className="fa-solid fa-circle-user fa-2x p-
1"></i>
    </div>
    <div className="p-1" style={{ width: "70%" }}>
      <p style={{ fontSize: "15px"
}}>{requirement.sentBy.name}</p>
    </div>
  </div>
  <div>
    <button className="btn btn-primary btn-sm m-
1">Accept</button>
    <button onClick={() => showModal(requirement)}
className="btn btn-primary btn-sm m-1">Negotiate</button>
    <i className="fa-solid fa-phone m-2" style={{
cursor: "pointer" }}></i>
    <i className="fa-brands fa-whatsapp m-2" style={{
cursor: "pointer" }}></i>

```



```

        </div>
    </div>
</div>
</div>
    )}
</div>
</div>
)}
{
    !responsesState && (
        <div className="container">
            <h1 className="text-center">Potential Leads</h1>
            {leads.length?
                (<div className="row"
                    style={{
                        display: "flex",
                        flexDirection: "row",
                        flexWrap: "wrap",
                        justifyContent: "space-around",
                    }}>
                    {leads.map((lead, index) => (
                        <div className="card p-1 m-1" key={index}>
                            <div className="d-flex justify-content-between">

                                <p>Quantity Required : {lead.quantity}
{lead.quantityUnit}</p>

                                <p className="">Cost offered : {lead.price} /- (
Rs.{lead.price / lead.quantity} per {lead.quantityUnit} )</p>

                                <div className="card mx-2 p-1" style={{ width:
"30%", display: "flex", flexDirection: "column" }}>
                                    <div style={{ display: "flex", flexDirection: "row" }} >
                                        <div style={{ width: "30%" }}>
                                            <i className="fa-solid fa-circle-user fa-2x p-1">
                                                </i>
                                            </div>
                                        </div>
                                    </div>
                                </div>
                            </div>
                    )
                )
            }
        </div>
    )
}

```



```

    footer=[
      <Button key="back" onClick={handleCancel}>Cancel</Button>,
      <Button key="submit" type="primary" onClick={() =>
handleOk(requirement?.productId, requirement?.buyerId)}>Submit</Button>,
    ]
  >
    <p>Product Name : {requirement?.productId?.name}</p>
    <div className="p-1 m-1" >
      <input type="number" required={true} value={quantity} onChange={(e) =>
setquantity(e.target.value)} style={{ borderRadius: "5px" }} className="p-1 m-1"
placeholder="Quantity" /> <label htmlFor="">{requirement?.productId?.quantityUnit}s</label>
    </div>
    <div className="p-1 m-1">
      Rs. <input type="number" required={true} value={price} onChange={(e) =>
setprice(e.target.value)} style={{ borderRadius: "5px" }} className="p-1 m-1"
placeholder="Offered price" /> per {quantity} {requirement?.productId?.quantityUnit}s
    </div>
    <div className="p-1 m-1" >
      Available date: <input type="date" required={true} value={date} onChange={(e) =>
setdate(e.target.value)} style={{ borderRadius: "5px" }} className="p-1 m-1"
placeholder="Available date" />
    </div>
    <div className="p-1 m-1" >
      <input type="text" required={true} value={notes} onChange={(e) =>
setnotes(e.target.value)} style={{ borderRadius: "5px" }} className="p-1" placeholder="Some
notes..." />
    </div>
  </Modal>
</>
);
};
export default Responses;

```

6.7 ColdStorages.js

```

import React, { useEffect, useState } from 'react';
import Nav from '../components/UIComponents/Nav';
import Footer from '../components/layouts/Footer';

```

```

import axios from 'axios';
import toast from 'react-hot-toast';
import Spinner from '../components/UIComponents/Spinner';
import './usercss/cold.css'
import randint from "random-int"

const ColdStorages = () => {
  const [coldstorages, setColdStorages] = useState([]);
  const [loading, setLoading] = useState(true);

  const images=["https://dce0qyjkutl4h.cloudfront.net/wp-content/webp-express/webp-
images/uploads/2020/01/cold-storage-1.jpg.webp", "https://stellarfoodforthought.net/wp-
content/uploads/2018/08/IMG_2007-e1600808052986.jpg",
"https://media.licdn.com/dms/image/C4E12AQGfpTZRLTLLkA/article-cover_image-
shrink_720_1280/0/1634835177119?e=2147483647&v=beta&t=qRyRzxImCCThsXwpMsovUN4I
oweNAH0mFl89ztGgEug"]

  const getCold = async () => {
    setLoading(true);
    try {
      const res = await axios.get(`${process.env.REACT_APP_API}/api/v1/cold/getcold`);
      if (res.data.success) {
        console.log("got cold storages");
        setColdStorages(res.data.result);
      } else {
        console.log("failed to get cold storages");
      }
    } catch (error) {
      console.log("failed to get them");
    } finally {
      setLoading(false);
    }
  };

  useEffect(() => {
    getCold();
  }, []);

  if (loading) {
    return (

```

```

    </>
    <Nav />
    <Spinner />
    <Footer />
  </>
); }
return (
  <
    <Nav />
    <div className="container" style={{ marginTop: '20px', minHeight: "50vh" }}>
      <div className="r">
        {coldstorages.map((coldStorage) => (
          <div key={coldStorage._id}>
            <div className="card" style={{ width: "18rem", minHeight: "24rem" }}>
              <div>
                <img src={images[randint(0,2)]} style={{ height: "150px" }} alt="cold" />
              </div>
              <div className="card-body">
                <h5 className="card-title">{coldStorage.name}</h5>
                <p className="card-text"><i className='fa-solid fa-phone'></i>
{coldStorage.phone}</p>
                <p className="card-text"><i className='fa-solid fa-location-dot'></i>
{coldStorage.address}</p>
                <p className="card-text"><i class="fa-solid fa-warehouse"></i>
{coldStorage.capacity ? coldStorage.capacity : "NA"} Metric tons</p>
              </div>
            </div>
          </div>
        ))}
      </div>
    <Footer />
  </>
);
};
export default ColdStorages;

```

CHAPTER 7

TESTING

Our testing process ensures the robustness, functionality, and security of the KisanRaj website, an online e-commerce platform designed to connect farmers and buyers. Testing is crucial to guaranteeing a seamless user experience and reliable performance.

7.1 Types of Testing

7.1.1 Functional Testing

Functional testing verifies that each feature of the website, such as posting and selling commodities, negotiating prices, hiring equipment, and renting cold storage, performs as intended. Test cases are designed to validate the functionality of these key features, ensuring they meet user requirements.

7.1.2 User Acceptance Testing (UAT)

UAT involves testing the website from the perspective of farmers and buyers. It ensures that the platform meets the needs and expectations of its target users. Test scenarios mimic real-world usage to validate the overall usability and satisfaction of the website.

7.1.3 Performance Testing

Performance testing evaluates the responsiveness and scalability of the website under different loads. It assesses the website's ability to handle concurrent users, process transactions efficiently, and maintain acceptable response times. Metrics such as response time, throughput, and resource utilization are measured to identify performance bottlenecks.

7.1.4 Security Testing

Security testing is conducted to identify and mitigate potential vulnerabilities in the website. It ensures the safety and privacy of user data by testing authentication mechanisms, data encryption, and access controls. Security testing aims to prevent unauthorized access, data breaches, and other security threats.

7.2 Testing Approach

Our testing approach combines automated and manual testing methodologies to ensure comprehensive coverage of the website's functionalities. We utilize React for front-end testing and MongoDB for back-end testing. Automated testing tools such as Jest and Enzyme are employed for unit and integration testing, while manual testing is conducted to validate user workflows and edge cases.

7.3 Test Cases

A suite of test cases is developed to validate the core functionalities of the KisanRaj website, including:

- Posting and selling commodities
- Negotiating prices
- Hiring equipment
- Renting cold storage

Each test case specifies the expected behavior and verifies that the actual result matches the expected outcome.

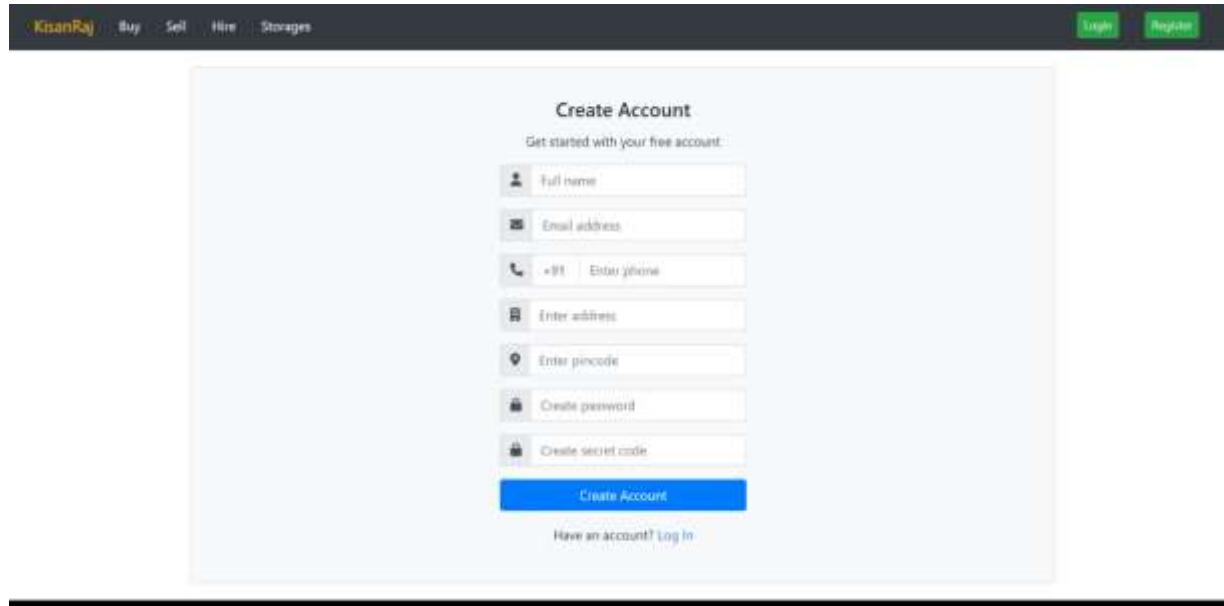
7.4 Testing Results

Throughout the testing process, issues and bugs were identified, prioritized, and addressed accordingly. Regular testing cycles enabled us to refine the website and improve its performance, usability, and security. The testing results validate the reliability and functionality of the KisanRaj platform, ensuring a seamless experience for farmers and buyers.

CHAPTER 8

RESULT & DISCUSSION

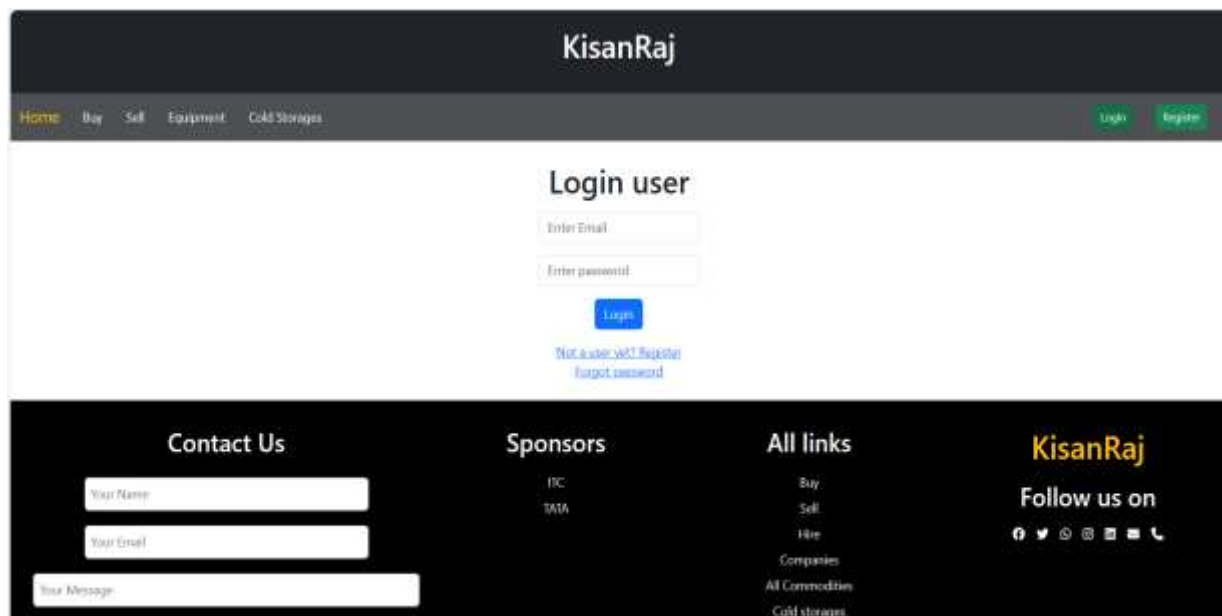
Registration Page:



The screenshot shows the 'Create Account' page of the KisanRaj website. The header includes the KisanRaj logo and navigation links: Buy, Sell, Hire, and Storages. There are 'Login' and 'Register' buttons in the top right corner. The main content area is titled 'Create Account' with the subtitle 'Get started with your free account:'. Below this, there are several input fields: 'Full name', 'Email address', '+91 Enter phone', 'Enter address', 'Enter pincode', 'Create password', and 'Create secret code'. A blue 'Create Account' button is at the bottom of the form. Below the button, it says 'Have an account? Log in'.

Figure 8.1 Registration page

Login Page:



The screenshot shows the 'Login user' page of the KisanRaj website. The header includes the KisanRaj logo and navigation links: Home, Buy, Sell, Equipment, and Cold Storages. There are 'Login' and 'Register' buttons in the top right corner. The main content area is titled 'Login user' and contains two input fields: 'Enter Email' and 'Enter password'. A blue 'Login' button is below the fields. Below the button, there are links for 'Not a user yet? Register' and 'Forgot password'. The footer is divided into four sections: 'Contact Us' with fields for 'Your Name', 'Your Email', and 'Your Message'; 'Sponsors' listing ITC and TATA; 'All links' listing Buy, Sell, Hire, Companies, All Commodities, and Cold storages; and 'KisanRaj Follow us on' with social media icons for Facebook, Twitter, Instagram, YouTube, and LinkedIn.

Figure 8.2 Login Page

Home Page:

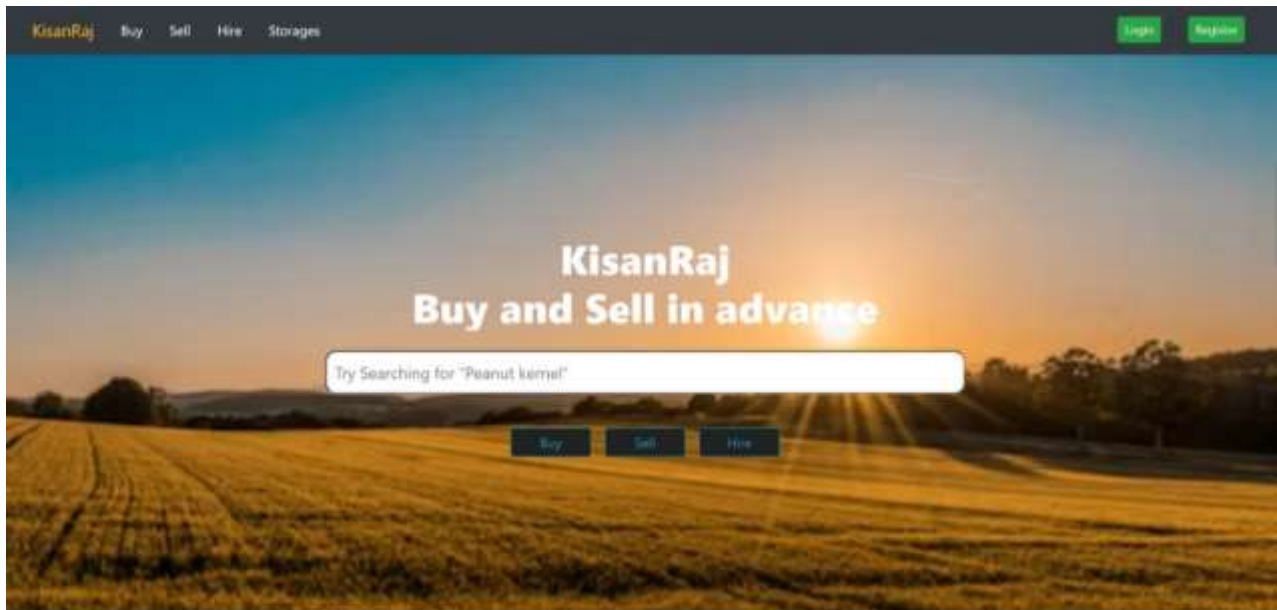


Figure 8.3 Home page

Sell-Commodity Page:

Figure 8.4 Sell-Commodity

Buy Commodity:

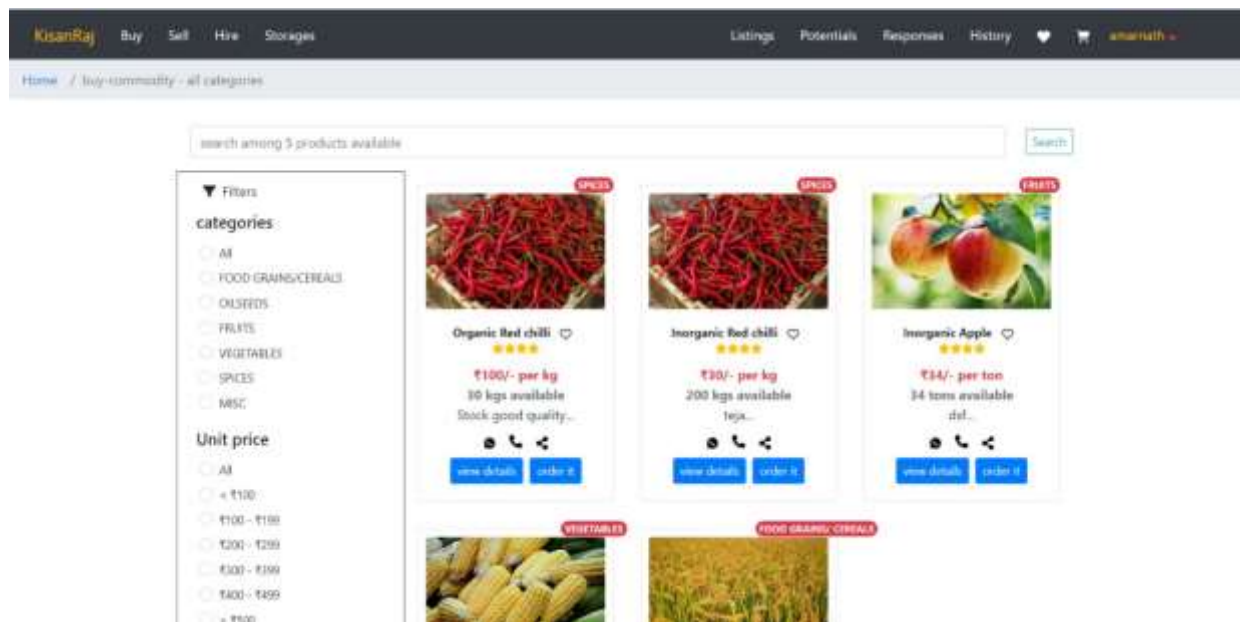


Figure 8.5 Buy-Commodity

Post-Equipment Page:

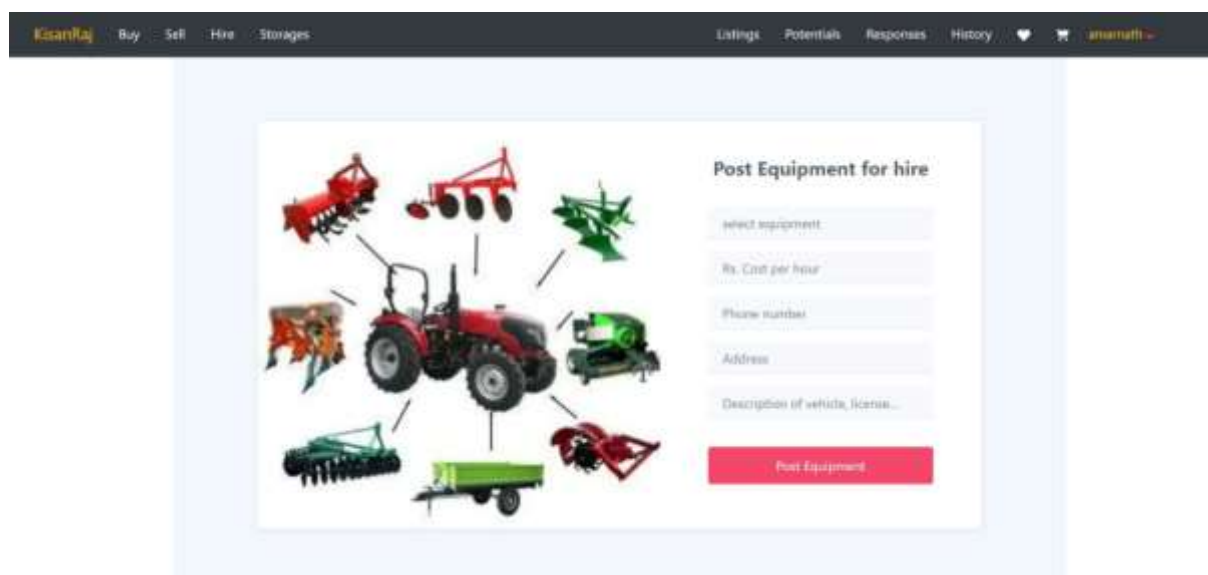


Figure 8.6 Post Equipment

Hire-Equipment Page:

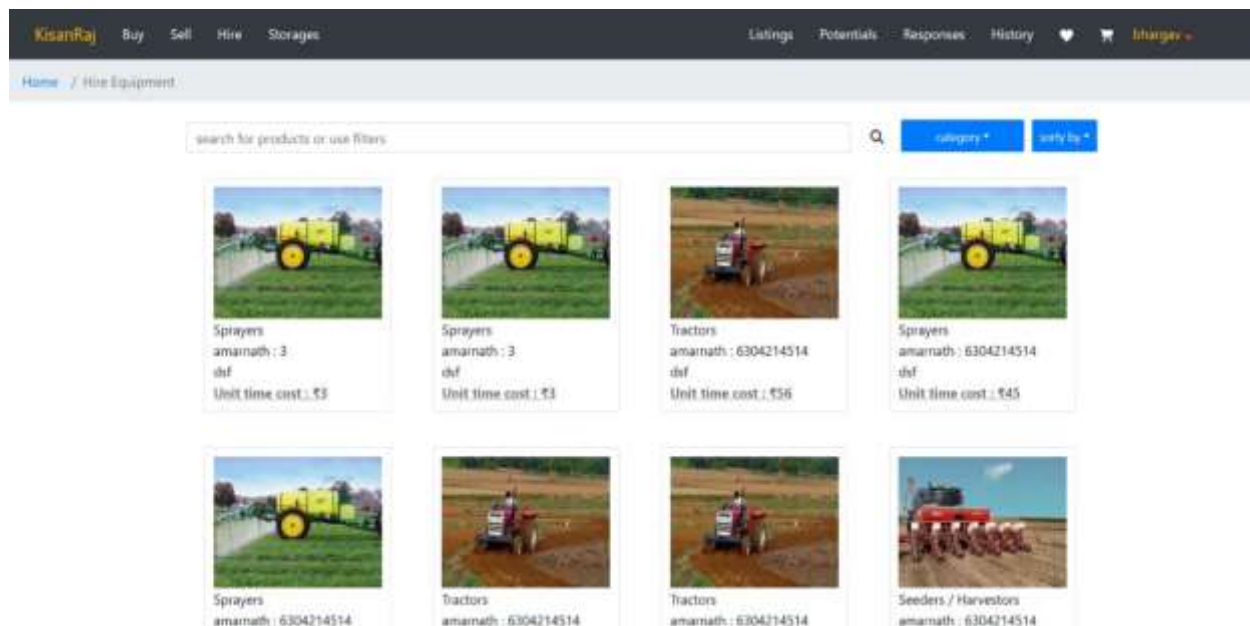


Figure 8.7 Hire Equipment

Post Potential Requirements Page:

The screenshot shows the 'Post Potential Requirements' page on the KisanRaj platform. The page is titled 'Fill Product details'. The form includes the following fields and options:

- Enter name**: A text input field.
- Rs. Price**: A dropdown menu with options: per, ton, box, quintal, dozen, kg, item, bag.
- Total Quantity Needed**: A text input field.
- Required by**: A date picker (dd/mm/yyyy).
- Is shipping needed?**: Radio buttons for Yes and No.
- Organic**: Radio buttons for Yes and No.
- Post Requirement**: A blue button.
- Hide posted potentials**: A yellow button.

Below the form, two posted potentials are displayed:

Product Name	Unit Price (₹)	Quantity Needed
Inorganic Orange	₹2540	20 bags
Organic Apple	₹1090	100 kgs

Figure 8.8 Post Potential Requirements

Cold Storage Page:

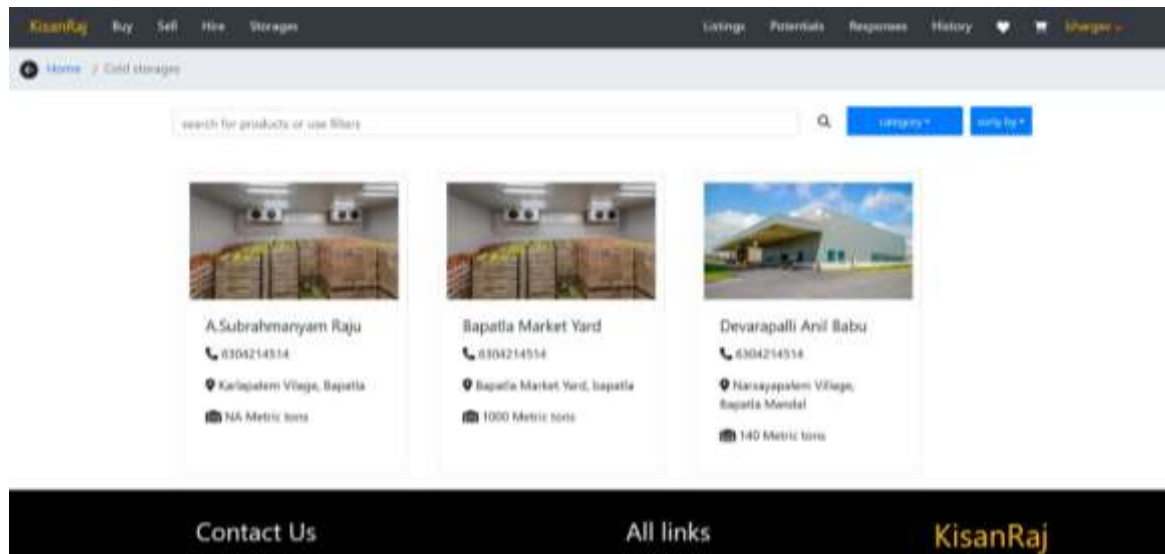


Figure 8.9 Cold Storage

Profile Page:

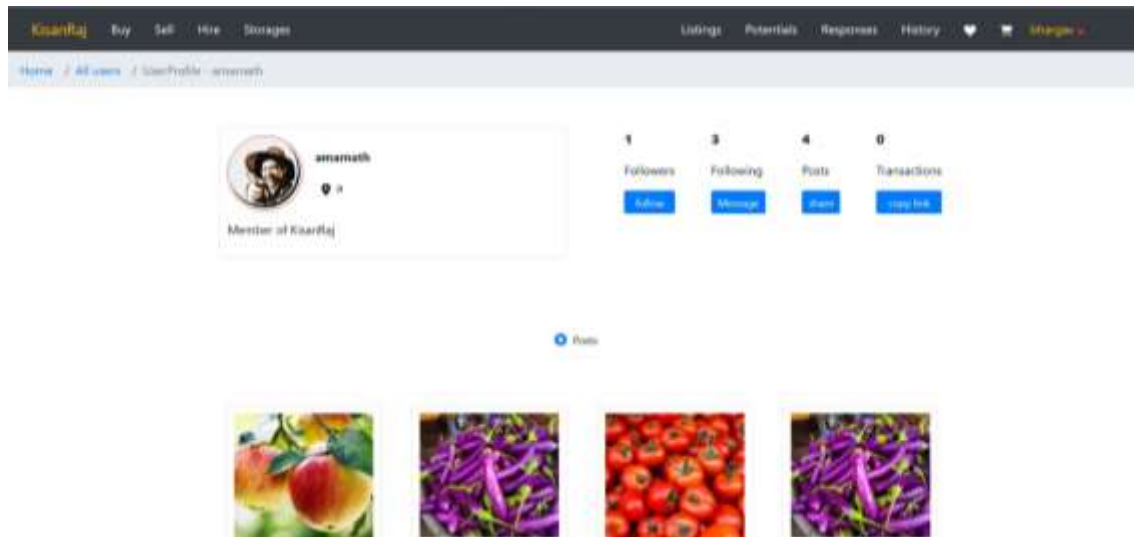


Figure 8.10 Profile

CHAPTER 9

CONCLUSION & FUTURE SCOPE

9.1 Conclusion

In conclusion, our project represents a significant leap forward in modernizing the Indian agricultural supply chain. By leveraging cutting-edge technologies and integrating key features such as direct communication, secure transactions, and connectivity with E-NAM, we are poised to revolutionize farmer-buyer interactions and enhance market access. Through our platform, farmers will have unprecedented opportunities to engage directly with buyers, negotiate fair prices, and expand their reach across different regions and states. Moreover, our commitment to sustainability and scalability ensures that our solution is not only effective in addressing current challenges but also adaptable and resilient in the face of future developments. By empowering farmers, fostering transparency, and promoting efficiency, our project contributes to building a more inclusive, resilient, and prosperous agricultural ecosystem in India

9.2 Future Scope

- **Enhanced Features:** Continuously innovate and add new features to the platform, such as real-time market insights, weather forecasts, and agricultural advisory services, to provide comprehensive support to farmers.
- **Mobile App Development:** Develop a mobile application for the platform to make it more accessible to farmers who may have limited access to desktop computers or internet connectivity.
- **Integration with IoT:** Explore integration with Internet of Things (IoT) devices to enable smart monitoring of agricultural processes, such as soil moisture levels, crop health, and irrigation systems, for better decision-making.
- **Blockchain Integration:** Implement blockchain technology to enhance transparency and traceability in agricultural transactions, ensuring fair pricing and reducing the risk of fraud.
- **Partnerships and Collaborations:** Forge partnerships with government agencies, NGOs, and agricultural organizations to leverage their resources and expertise in reaching and supporting farmers effectively.
- **Training and Capacity Building:** Offer training programs and capacity-building initiatives to empower farmers with the skills and knowledge needed to leverage the platform effectively and maximize their agricultural productivity.

REFERENCES

- [1] Yi-chang, C., & Lee, H. (2020, January), The Effects of Website Content and Trust on Online Purchasing Intention of Agricultural Products. <https://doi.org/10.2991/aebmr.k.200626.062>
- [2] Warlina, L., Siddiq, F., & Valentina, T. (2019, December 1). Designing website for online business in the agricultural sector. IOP Publishing, 1402(6), 066080-066080. <https://doi.org/10.1088/1742-6596/1402/6/066080>
- [3] Ong, R J., Raof, R A A., Sudin, S., & Choong, K Y. (2021, February 1). A Review of Chatbot development for Dynamic Web-based Knowledge Management System (KMS) in Small Scale Agriculture. <https://doi.org/10.1088/1742-6596/1755/1/012051>
- [4] <http://www.manage.gov.in/studymaterial/scm/E.pdf> Reading material on Supply chain management in agriculture.
- [5] Kunaka, C 2011, 'Logistics in Lagging Regions: Overcoming Local Barriers to Global Connectivity', Washington DC, World Bank.
- [6] Sazzad, P 2014, 'Food supply chain management in Indian Agriculture: Issues, opportunities and further research'. African journal of management, vol. 8, no.14, pp. 572-581. doi. 10.5897/ajbm2013.729
- [7] Shaik Meera, N, Arunkumar, S, Amtul Waris, Vara Prasad, C, Muthuraman, P, Mangalsen, and Vikranth, BC 2010, 'E-Learning in extension systems- Emperical study in Agricultural Extension in India'. Indian Journal of Extension Education, vol. 46, no. 3&4, pp. 94-101.
- [8] Shalendra, Shalendra & Gummagolmath, Karabasayya & Sharma, Purushottam. (2013). User Centric ICT Model for Supply Chain of Horticultural Crops in India. Agricultural Economics Research Review. 26. 91-100.
- [9] Solomon, E., & Klyton, A V. (2020, December 1). The impact of digital technology usage on economic growth in Africa. <https://doi.org/10.1016/j.jup.2020.101104>
- [10] Finger, Robert. (2023). Digital innovations for sustainable and resilient agricultural systems. European Review of Agricultural Economics. 50. 1277-1309. 10.1093/erae/jbad021.
- [11] Rambod Abiri, Nastaran Rizan, Siva K. Balasundram, Arash Bayat Shahbazi, Hazandy Abdul-Hamid, Application of digital technologies for ensuring agricultural productivity, Heliyon, Volume 9, Issue 12, 2023, e22601, ISSN 2405-8440, <https://doi.org/10.1016/j.heliyon.2023.e22601>.
- [12] B., Shibi & Aithal, Sreeramana. (2022). A Study on Challenges Faced by Farmers Using ECommerce in Agriculture - A Survey of Thrissur District in the State of Kerala, India. International Journal of Case Studies in Business, IT, and Education. 600-610. 10.47992/IJCSBE.2581.6942.0220.
- [13] Fountas, Spyros & García, Borja & Kasimati, Aikaterini & Mylonas, Nikolaos & Darra, Nicoleta. (2020). The Future of Digital Agriculture: Technologies and Opportunities. IT Professional. 22. 24-28. 10.1109/MITP.2019.2963412.

- [14] Banker, Rajiv & Mitra, Sabyasachi & Sambamurthy, V.(2011). The Effects of Digital Trading Platforms on Commodity Prices in Agricultural Supply Chains. MIS Quarterly.35.599611. 10.2307/23042798.
- [15] Chaudhary S., Suri P.K. (2022): The impact of digitalisation on the agricultural wholesale prices to aid agrarian income. Agric. Econ. – Czech.
- [16] Bhatti, A.: Consumer purchase intention effect on online shopping behavior with the moderating role of attitude. International Journal of Academic Management Science Research. Vol. 2, No. 7, pp. 44-50 (2018).
- [17] Yi-Chang Chen.,Hui-Ho Lee.: The Effects of Website Content and Trust on Online Purchasing Intention of Agricultural Products(2019).
- [18] Heang, J.F. and Khan, H.U.: The role of internet marketing in the development of agricultural industry: a case study of China. Journal of Internet Commerce, Vol. 14, pp:65–113 (2015).
- [19] Athapaththu, J.C. and Kulathunga, K.M.S.D.: Factors affecting online purchase intention: a study of Sri Lankan online customers. International Journal of Scientific & Technology Research. Vol. 7, No. 9, pp. 120-128 (2018).
- [20] Rahimnia, J. and Hassanzadeh, F.: The impact of website content dimension and e-trust on e-marketing effectiveness: the case of Iranian commercial saffron corporations. Information & Management, Vol. 50, pp. 240–247 (2013).