ABSTRACT

The increasing disconnect between local farmers and consumers often leads to inefficiencies in the agricultural supply chain, resulting in reduced farmer income and limited access to fresh, locally sourced produce for urban populations. Traditional distribution channels can involve multiple intermediaries, driving up costs and extending the time between harvest and consumption, potentially impacting the quality and nutritional value of produce. Furthermore, farmers often lack direct insight into market demand, making it difficult to optimize their production and pricing strategies. This project, CropConnect, aims to bridge this gap by creating a direct-to-consumer platform that empowers farmers and provides consumers with convenient access to fresh, local agricultural products.

CropConnect is a web-based application developed using modern web technologies, including React for the frontend, Node.js and Express for the backend API, and a PostgreSQL database for data persistence. The platform facilitates direct interaction between farmers and consumers, allowing farmers to list their available produce, set prices, and manage inventory, while consumers can browse, search, and purchase products directly from local sources. A key aspect of CropConnect is the integration of artificial intelligence to provide valuable services to farmers. This includes features for delivering relevant farming news, predicting optimal product pricing based on location and market data, and providing AI-powered analytics to offer insights into sales performance and strategies for improving income. Additionally, the platform incorporates accessibility features such as a read-aloud option for content and support for regional languages to ensure broader usability and inclusivity for farmers from diverse backgrounds.

In conclusion, CropConnect provides a viable solution to address the challenges in the local agricultural supply chain by creating a direct and efficient marketplace enhanced by intelligent features. The platform has the potential to significantly benefit both farmers, through increased income, market access, and data-driven decision-making, and consumers, through access to fresher produce and a connection to their local food system. Future extensions for CropConnect could include the development of native mobile applications for both farmers and consumers, the integration of advanced logistics and delivery tracking features, and the incorporation of features supporting community-supported agriculture (CSA) programs and farm-to-table initiatives.

1. INTRODUCTION

In today's rapidly evolving world, the agricultural sector faces numerous challenges, from fluctuating market prices and unpredictable weather patterns to the complexities of reaching consumers effectively. Traditional agricultural supply chains often involve multiple intermediaries, leading to reduced profitability for farmers and increased costs for consumers. Furthermore, the lack of direct communication channels between producers and consumers can hinder the development of a transparent and efficient food system. This project documentation outlines the development and implementation of CropConnect, a platform designed to address these challenges by creating a direct and streamlined connection between local farmers and consumers.

CropConnect is envisioned as a transformative tool that empowers farmers by providing them with direct control over the sale of their produce and access to valuable market insights. By eliminating unnecessary intermediaries, farmers can potentially increase their income and gain a better understanding of consumer demand. For consumers, CropConnect offers the convenience of accessing fresh, locally sourced produce directly from the farmers who grow it, fostering a sense of community and supporting local economies. This documentation will delve into the design principles, technical architecture, and key features of CropConnect, demonstrating how it facilitates this direct interaction and aims to create a more sustainable and equitable agricultural ecosystem.

CropConnect's objectives, scope, and the technologies employed in its development are thoroughly detailed. The platform's functional and non-functional requirements, the design decisions made during the development process, and the implementation details of its key modules are covered. Additionally, the integration of artificial intelligence features aimed at enhancing farmer capabilities and the inclusion of accessibility features to ensure wide usability are explored. Finally, the potential impact of CropConnect is discussed, and potential future enhancements to further improve its functionality and reach are outlined.

1.1 Motivation

The primary motivation behind the CropConnect project is to create a more equitable and efficient agricultural ecosystem by directly connecting local farmers with consumers. Many farmers face challenges in achieving fair compensation due to complex supply chains and limited direct market access. CropConnect addresses this by providing a platform for farmers to sell their produce directly, increasing their income and business control.

Furthermore, CropConnect is driven by the need to leverage technology to support farmers in the face of modern agricultural challenges. The project incorporates artificial intelligence to provide valuable tools such as price prediction based on market data, sales analytics for business optimization, and crucial weather updates to help mitigate risks from unpredictable conditions. This empowers farmers with data-driven insights to improve their practices and income. Simultaneously, the platform benefits consumers by offering direct access to fresh, high-quality local produce at potentially lower costs, fostering a stronger connection to their food sources and supporting local economies.

1.2 Problem Definition

Agriculture forms the backbone of the Indian economy, with a significant portion of the population engaged in farming activities. Despite their crucial role in food production, a large number of these farmers struggle to earn a sustainable and deserving income. A primary contributor to this issue is the prevalence of multi-layered supply chains involving numerous intermediaries. These middlemen often absorb a substantial portion of the profit margin, leaving farmers with a disproportionately small share of the final selling price of their produce. This not only impacts the financial well-being of farmers but also disincentivizes investment in modern farming practices and technology, perpetuating a cycle of low productivity and limited economic growth within the agricultural sector.

The lack of direct access to markets and consumers also leaves farmers vulnerable to price fluctuations and market volatility. Without real-time information on demand and pricing, farmers are often forced to sell their produce at unfavourable rates to intermediaries. This opaque system limits their bargaining power and hinders their ability to make informed decisions about planting, harvesting, and selling. The absence of a direct channel for farmers to connect with consumers exacerbates these problems, creating an inefficient and often exploitative system that benefits intermediaries more than the producers themselves. CropConnect aims to directly address this fundamental problem by creating a transparent and efficient platform that bypasses traditional middlemen, enabling farmers to connect directly with consumers and reclaim a larger share of the value they create.

1.3 Existing System

The current agricultural supply chain largely operates through a traditional multi-tiered system involving farmers selling to intermediaries like aggregators and wholesalers in markets, who then distribute to retailers before reaching consumers. This established system, while facilitating distribution, often lacks transparency and efficiency, leaving farmers with limited bargaining power and a small share of the final profit due to multiple intermediaries. Consumers, in turn, may face higher prices and reduced freshness due to the extended chain. Although limited direct sales occur, there is no widespread, unified platform connecting farmers directly with consumers to streamline the process and ensure fairer outcomes for producers and better access to fresh produce for consumers.

1.4 Proposed System

The proposed system, CropConnect, is designed to be a direct-to-consumer platform that fundamentally transforms the existing agricultural supply chain by creating a direct link between local farmers and end consumers. This web-based application will serve as a centralized marketplace where farmers can register, list their available produce with details such as type, quantity, price, and origin, and manage their inventory directly. Consumers will be able to browse, search, and filter produce based on various criteria, add items to a shopping cart, and complete secure online transactions.

A key aspect of the proposed system is the integration of intelligent features powered by artificial intelligence. This includes providing farmers with valuable market insights through data analytics on their sales performance, enabling them to optimize their production and pricing strategies for increased income. AI will also be utilized to offer features like price prediction based on market trends and location, and to provide timely and relevant farming news and weather updates to help farmers make informed decisions and mitigate risks. The platform will also prioritize accessibility with features such as regional language support and a read-aloud option to ensure inclusivity. By bypassing traditional intermediaries, CropConnect aims to create a more transparent, efficient, and equitable system that benefits both farmers through increased profitability and consumers through access to fresh, high-quality local produce at potentially lower prices. The system will manage user authentication and authorization for both farmer and customer roles, handle product listings and inventory, manage the shopping cart and checkout process, and facilitate communication between users where necessary.

1.5 Requirements Specification

Requirement Specifications describe the software and hardware requirements for building and running the CropConnect application. This system involves developing a direct-to-consumer platform for agricultural produce, connecting farmers and customers, and incorporating AI features.

1.5.1 Software Requirements

1. **Platform:** Node.js, React.js
2. **Front-end:** React.js, Tailwind CSS
3. **Back-end:** Node.js, Express.js
4. **Database:** PostgreSQL
5. **AI Integration:** Genkit, @genkit-ai/googleai (for integrating AI services like price prediction, analytics, and weather data processing)
6. **Authentication:** Firebase Authentication
7. **Tools:** VS Code

1.5.2 Hardware Requirements

- RAM – 4GB minimum

2.LITERATURE SURVEY

In recent years, agricultural digitization has emerged as a transformative force addressing persistent issues such as fragmented supply chains, reliance on intermediaries, volatile market prices, and limited access to consumer markets. While several digital agriculture platforms have been introduced to mitigate these challenges, gaps remain—particularly in delivering user-centric, scalable solutions that cater to the needs of smallholder farmers and support direct farmer-to-consumer interactions.

Shetigram - Building a Digital Bridge Between Farmers and Consumers (ICSES 2024)
This study introduces *Shetigram*, a digital platform aimed at connecting farmers and consumers directly. It incorporates advanced technologies such as BERT (Bidirectional Encoder Representations from Transformers) for natural language understanding, ECDSA for secure authentication, and machine learning algorithms like XGBoost for price and demand forecasting. It uniquely utilizes USSD interfaces to increase accessibility in rural areas. The integration of blockchain enhances data integrity, making this solution highly secure and predictive, though its reliance on complex models may increase computational requirements and implementation difficulty in low-resource settings. [1]

AGROCART - An Online Platform for Farmers to Sell Products without Middleman (2023)
AGROCART is a web-based application designed to eliminate intermediaries in the agricultural supply chain. It enables farmers to list products directly to consumers through a user-friendly online portal. This system focuses on increasing transparency and fair pricing while leveraging standard web technologies. Its main advantage lies in simplicity and user accessibility. However, it lacks AI integration or predictive analytics for supporting decision-making by the stakeholders, potentially limiting its scalability and smart automation. [2]

Enhancing Crop Yield Prediction and Provide Direct Market Access Using Web and ML (2024)
This paper explores an integrated web-based platform that uses machine learning models—LightGBM, KNN Regressor, Gradient Boosting, and CatBoost—for crop yield prediction and market trend analysis. The system supports farmers in forecasting productivity and accessing markets directly. The blend of accurate ML forecasting and accessible web technologies makes it practical and effective. However, the challenge remains in ensuring accurate data collection from the field and managing model complexity in real-world deployments. [3]

KisanConnect: Reshaping Agricultural Trade with Dynamic Pricing Model (ICNEWS 2024)
KisanConnect focuses on real-time pricing and analytics using regression models and heuristic algorithms. It offers dynamic pricing for crops based on market conditions, empowering farmers with timely economic insights. The solution enhances market efficiency and farmer revenue. Nevertheless, its dependency on continuous real-time data streams may face challenges in areas with limited internet connectivity or lack of reliable data sources. [4]

Design and Development of a Mobile Application for Direct Agricultural Market Access (2025)
This research outlines the development of a mobile app that improves rural connectivity and enables direct market access for farmers. It emphasizes UI simplicity and supply chain optimization using

heuristic algorithms and visual analytics. The app supports better investment planning and product distribution. While highly relevant for underserved rural communities, its success hinges on digital literacy and access to smartphones, which can vary significantly across regions. [5]

Table 2.1: Comparison of Literature survey

Application	Author	Title	Year	Methodology / Concept	Merits	Demerits
Mobile App for Direct Agricultural Market Access	G. Manikandan, N. S. B, S. S. S. S. T, N. J B and S. RM	Design and Development of a Mobile Application for Direct Agricultural Market Access	2025	Heuristic algorithms, visual analytics, rural connectivity enhancement	Improves rural access to markets, user-friendly design, promotes investment planning	Requires digital literacy, smartphone access
KisanConnect	S. Bhukta, A. Shil, A. Dutta and B. Sadhukhan	KisanConnect: Reshaping Agricultural Trade with Dynamic Pricing Model	2024	Dynamic pricing, real-time market analytics, regression tree models	Empowers farmers with real-time price data, increases income opportunities	Depends on reliable internet and real-time data
Crop Yield Prediction and Direct Market Access	S. G, N. S, R. A and A. N	Enhancing Crop Yield Prediction and Provide Direct Market Access to Farmers Using Web Technologies and Machine Learning Models	2024	ML algorithms (LightGBM, KNN, CatBoost), web technologies	Accurate yield prediction, data-driven decision-making	Model complexity, data collection challenges
Shetigram	N. G, A. G, S. K. M, H. V and B. L	Shetigram - Building a Digital Bridge Between Farmers and Consumers	2024	BERT, ECDSA, XGBoost, USSD, blockchain integration	Secure platform, smart analytics, accessible via USSD	High system complexity, resource-intensive
AGROCART	R. B. Lincy, D. MD, R. MK, M. S and B. G	AGROCART – An Online Platform for Farmers to Sell Products without Middleman	2023	Web portal using standard web technologies	Simple interface, eliminates middlemen, transparent pricing	Lacks smart automation, no AI integration

3. SYSTEM DESIGN

The CropConnect project is being managed and developed using an **Agile-inspired methodology**, adapted to the specific needs and context of building a platform that requires iterative development and responsiveness to user feedback. This approach prioritizes flexibility, collaboration, and continuous improvement throughout the project lifecycle.

Our methodology is characterized by the following principles and practices:

- **Iterative Development:** The project is broken down into smaller, manageable iterations (similar to sprints in Scrum). Each iteration focuses on delivering a set of functional features, allowing for regular testing, feedback collection, and adjustments to the development plan. This approach enables us to quickly respond to changing requirements and ensure that the platform is evolving in a way that best meets the needs of farmers and customers.
- **Cross-functional Collaboration:** Development involves close collaboration between frontend developers, backend developers, and potentially individuals working on AI model integration and data management. Regular communication and knowledge sharing are encouraged to ensure alignment and efficient problem-solving.
- **User-Centric Approach:** User feedback is a critical driver of the development process. We aim to gather feedback from potential users (farmers and consumers) throughout the iterations, incorporating their insights into the design and functionality of the platform. This ensures that CropConnect is intuitive, user-friendly, and addresses the real-world challenges faced by its target audience.
- **Flexible Planning and Adaptability:** While initial planning provides a roadmap, the methodology allows for flexibility in adapting to new information, unforeseen challenges, and evolving requirements. The iterative nature of development enables us to reprioritize tasks and adjust the plan as needed, rather than adhering to a rigid upfront plan.
- **Continuous Integration and Testing:** Code changes are integrated frequently, and automated testing is employed where possible to identify and address issues early in the development cycle. This helps maintain code quality and stability.
- **Risk Management:** Potential risks, such as technical challenges in AI integration, user adoption rates, and data security concerns, are identified and assessed throughout the project. Mitigation strategies are developed and implemented as part of the ongoing project management process.
- **Focus on Value Delivery:** Each iteration aims to deliver tangible value to the potential users of the platform. This could be in the form of new features, improved usability, or enhanced performance. The focus is on building a functional and impactful platform incrementally.

By adopting this Agile-inspired methodology, the CropConnect project aims to deliver a robust, user-centric, and effective platform that can adapt to the dynamic needs of the agricultural sector and provide meaningful benefits to both farmers and consumers. This approach provides a framework for planning, executing, and managing the project in a way that maximizes responsiveness and the likelihood of success.

Technical Methodology and System Architecture

The CropConnect platform is engineered based on a robust **client-server architecture**, employing a layered approach to ensure scalability, maintainability, and a clear separation of concerns. This technical methodology outlines the key components, their roles, the technologies employed, and the interaction patterns that govern the system's operation.

At the **frontend layer**, the user interfaces for both Farmers and Customers are built using **React**, a declarative, component-based JavaScript library. This layer is responsible for rendering the user interface, managing client-side state, handling user interactions, and presenting information retrieved from the backend. The visual design and responsiveness are primarily achieved through a utility-first CSS framework, likely **Tailwind CSS**, which is integrated into the React build process. The frontend communicates with the backend exclusively through well-defined **RESTful APIs**, initiating asynchronous HTTP requests to fetch data, submit user input, and trigger backend processes. This clear API contract between the frontend and backend is a cornerstone of the architecture, enabling independent development and deployment of both layers.

The **backend layer** serves as the application's core processing engine, implemented using **Node.js** and the **Express.js** framework. This layer is responsible for executing the business logic, managing data persistence, handling user authentication and authorization, and orchestrating interactions with external services. It exposes the RESTful API endpoints consumed by the frontend, processing incoming requests, validating data, and generating appropriate responses. The backend interacts with the **PostgreSQL** relational database for all data storage and retrieval operations, leveraging SQL queries to manage user data, product information, orders, and other critical application data. User authentication is managed through integration with **Firebase Authentication**, with the backend validating tokens and enforcing role-based access control rules to ensure secure access to resources.

A key aspect of the technical methodology is the seamless **integration of Artificial Intelligence**. The backend communicates with a dedicated **AI service**, likely implemented using the **Genkit** framework. When AI-powered functionalities are required, such as generating a price prediction for a product, the backend sends relevant data to the AI service. The AI service, which encapsulates the specific AI models (e.g., machine learning models trained for price forecasting), processes this data and returns the results to the backend. The backend then incorporates these AI results into the data sent back to the frontend for display to the user. This decoupled approach to AI integration allows for flexibility in model management and scaling the AI capabilities independently of the core backend. Furthermore, the backend manages interactions with external services like a **Payment Gateway** for secure transaction processing, following established protocols and ensuring data integrity.

This technical methodology, based on a layered architecture, RESTful API communication, and strategic integration of specialized services like AI and authentication, provides a robust and scalable foundation for the CropConnect platform. It defines a clear structure for development, facilitates collaboration between frontend and backend teams, and ensures the efficient and secure operation of the system.

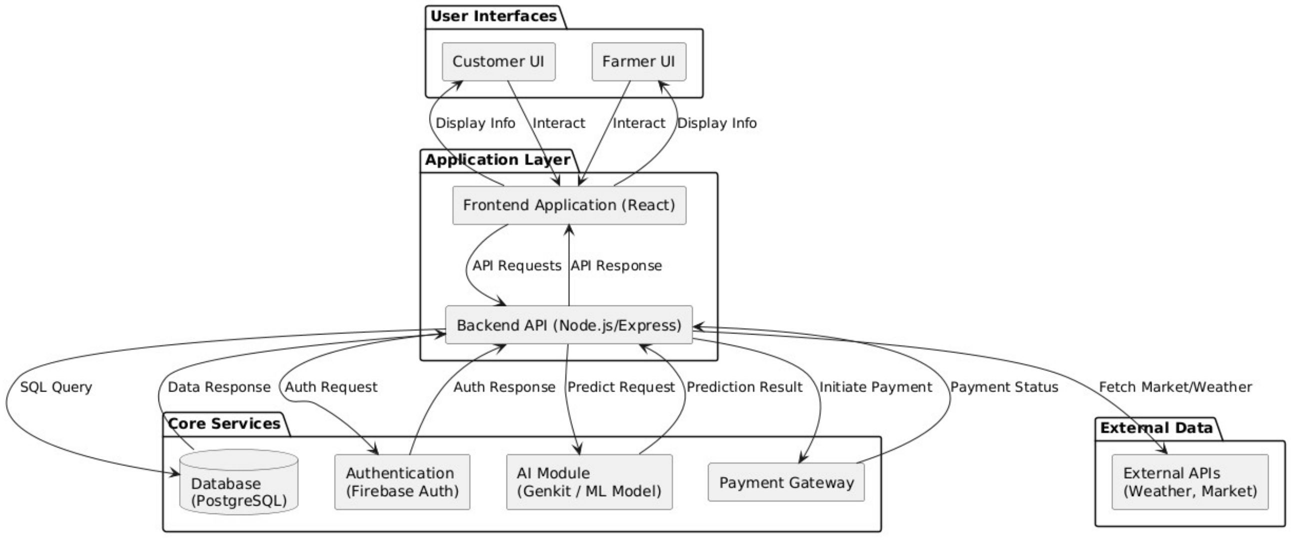


Figure 3.1 System Architecture Diagram

Figure 3.1 depicts the interaction between user interfaces, application layer, core services, and external data sources within the CropConnect platform.

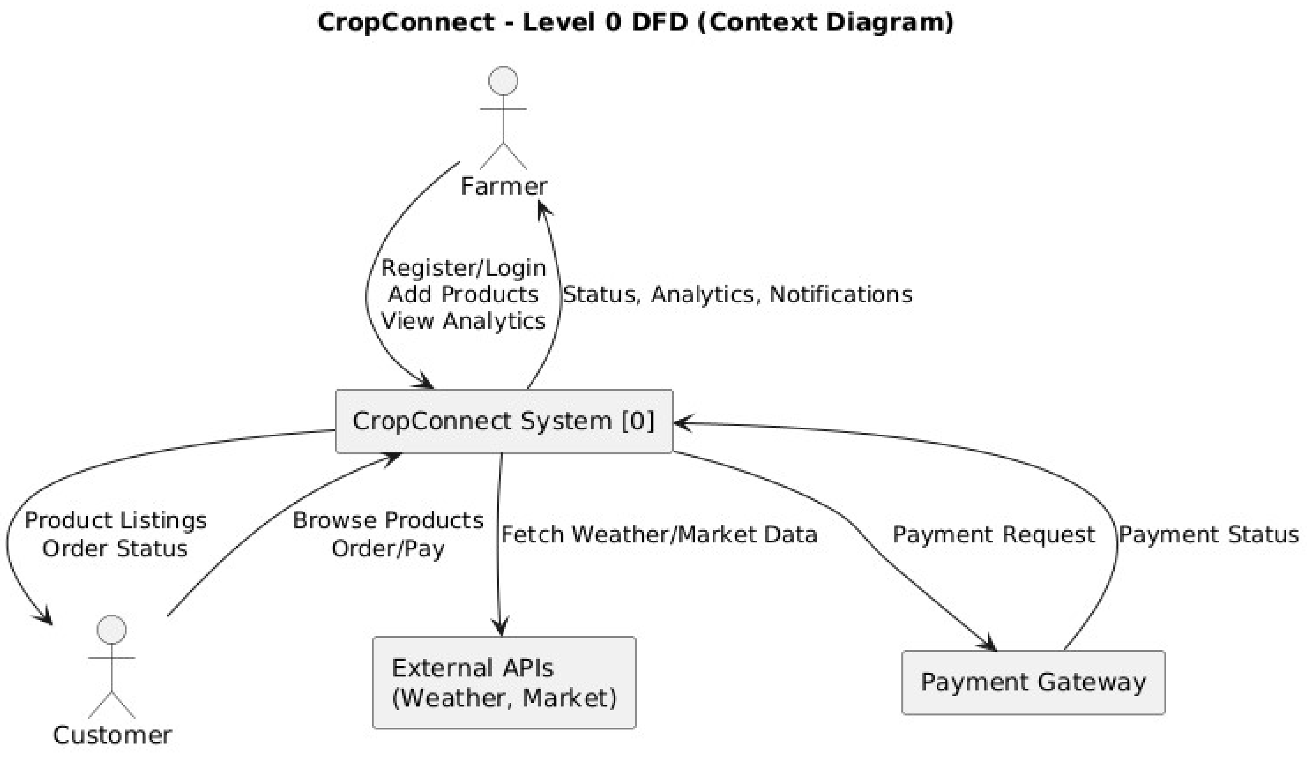


Figure 3.2 Level 0 Data Flow Diagram

The Figure 3.2 gives a high-level view of the CropConnect system as a single process. It shows the interaction between the system and external entities like **Farmers**, **Customers**, **Payment Gateway**, and **External APIs**. It outlines basic data flow such as **registration/login**, **product listings**, **orders**, **payments**, and **analytics requests**, offering a simple overview of the system's boundaries and external communication.

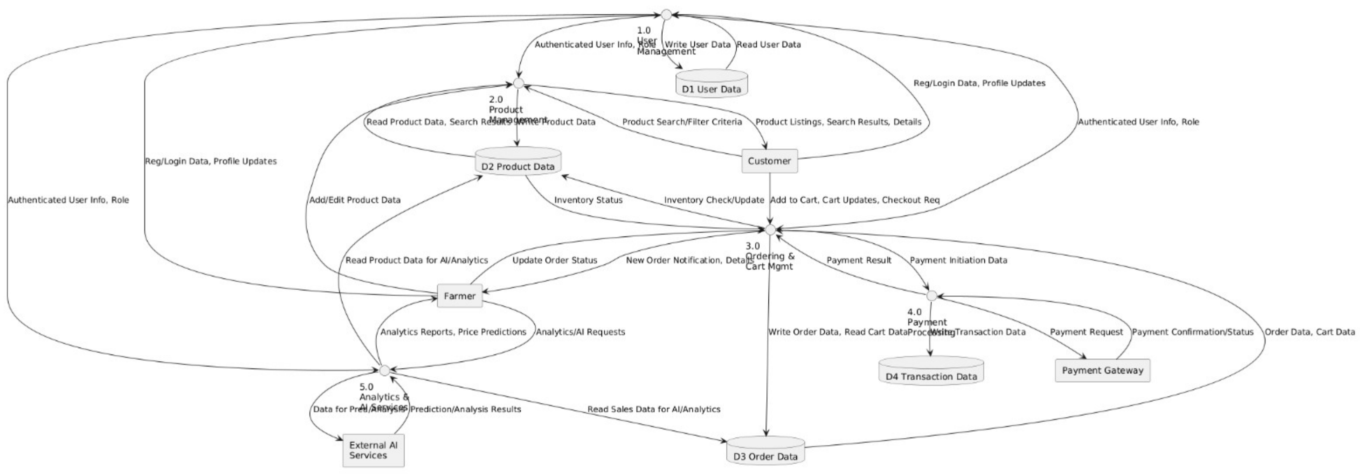

1.0
User
Management
Authenticated User Info, Role
Write User Data
Read User Data
D1 User Data
Reg/Login Data, Profile Updates
2.0
Product
Management
Read Product Data, Search Results
Write Product Data
Product Search/Filter Criteria
Product Listings, Search Results, Details
Authenticated User Info, Role
Reg/Login Data, Profile Updates
D2 Product Data
Customer
Authenticated User Info, Role
Add/Edit Product Data
Inventory Status
Inventory Check/Update
Add to Cart, Cart Updates, Checkout Req
3.0
Ordering &
Cart Mgmt
Read Product Data for AI/Analytics
Update Order Status
New Order Notification, Details
Payment Result
Payment Initiation Data
Farmer
4.0
Payment
Processing
Analytics Reports, Price Predictions
Analytics/AI Requests
Write Order Data, Read Cart Data
Write Transaction Data
Payment Request
Payment Confirmation/Status
Order Data, Cart Data
D4 Transaction Data
Payment Gateway
5.0
Analytics &
AI Services
Data for Pred/Analysis
Prediction/Analysis Results
Read Sales Data for AI/Analytics
External AI Services
D3 Order Data

Figure 3.3 Level 1 Data Flow Diagram

The Level 1 DFD breaks down the core process of CropConnect into **five major functional components**:

- **1.0 User Management**: Handles registration, login, authentication, and user profile management.
- **2.0 Product Management**: Manages product listings, inventory, and updates by farmers.
- **3.0 Ordering & Cart Management**: Manages customer carts, order placement, and inventory updates.
- **4.0 Payment Processing**: Handles payment requests, gateway interaction, and transaction logging.
- **5.0 Analytics & External APIs**: Processes AI-based predictions and integrates with external APIs for weather and market trends.

The Figure 3.3 also shows **data stores** (User DB, Product DB, Order DB, Transaction DB) and how they interact with processes, providing a more detailed understanding of **internal data flows, system logic, and dependencies**.

3.1 UML Diagrams

3.1.1 Class Diagram

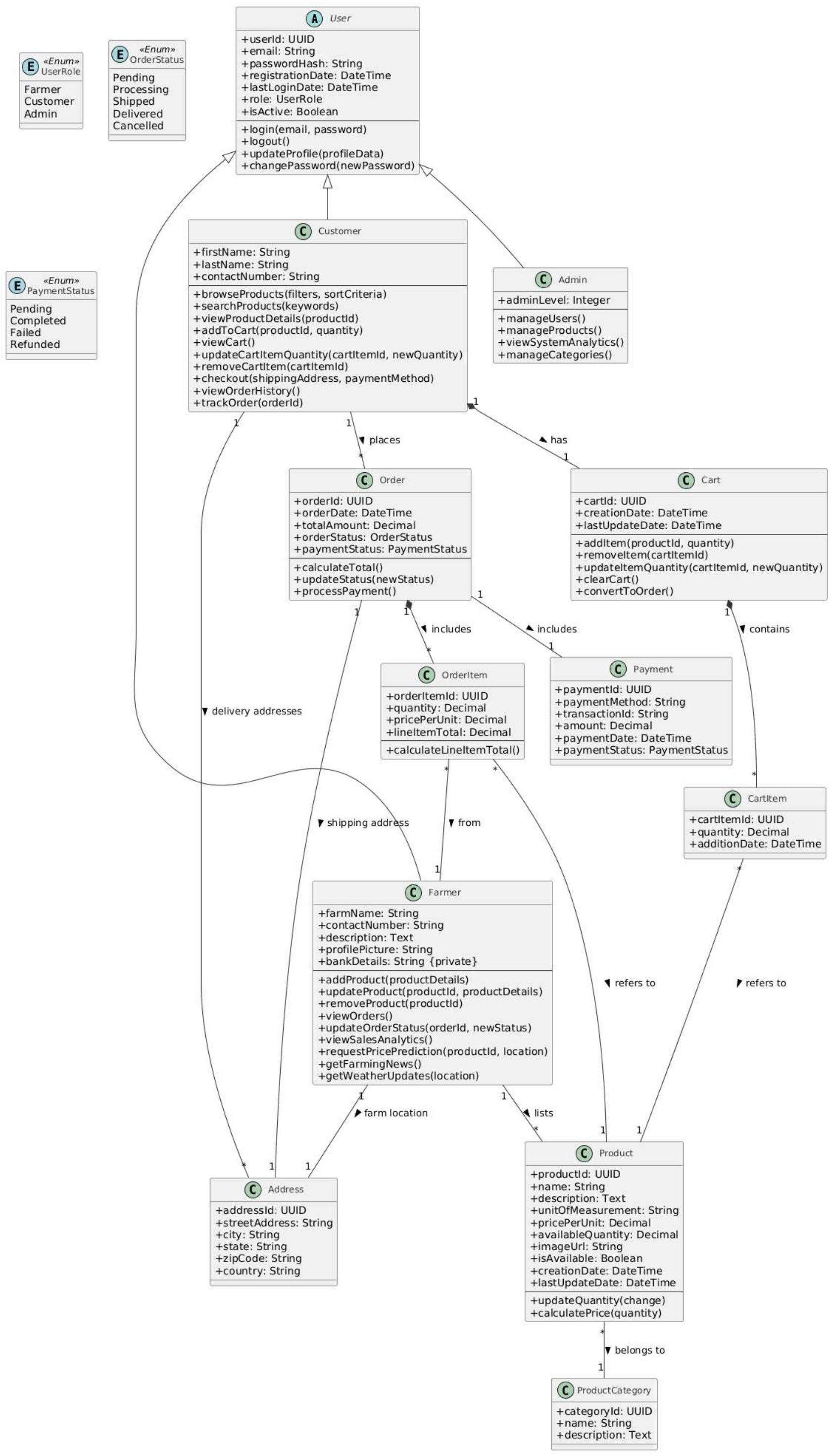


Figure 3.4 Class Diagram

Figure 3.4 illustrates the static structure of the CropConnect platform, detailing the key entities, their attributes, operations, and the relationships between them. It provides a foundational view of the system's data model and object interactions.

3.1.2 UseCase Diagram

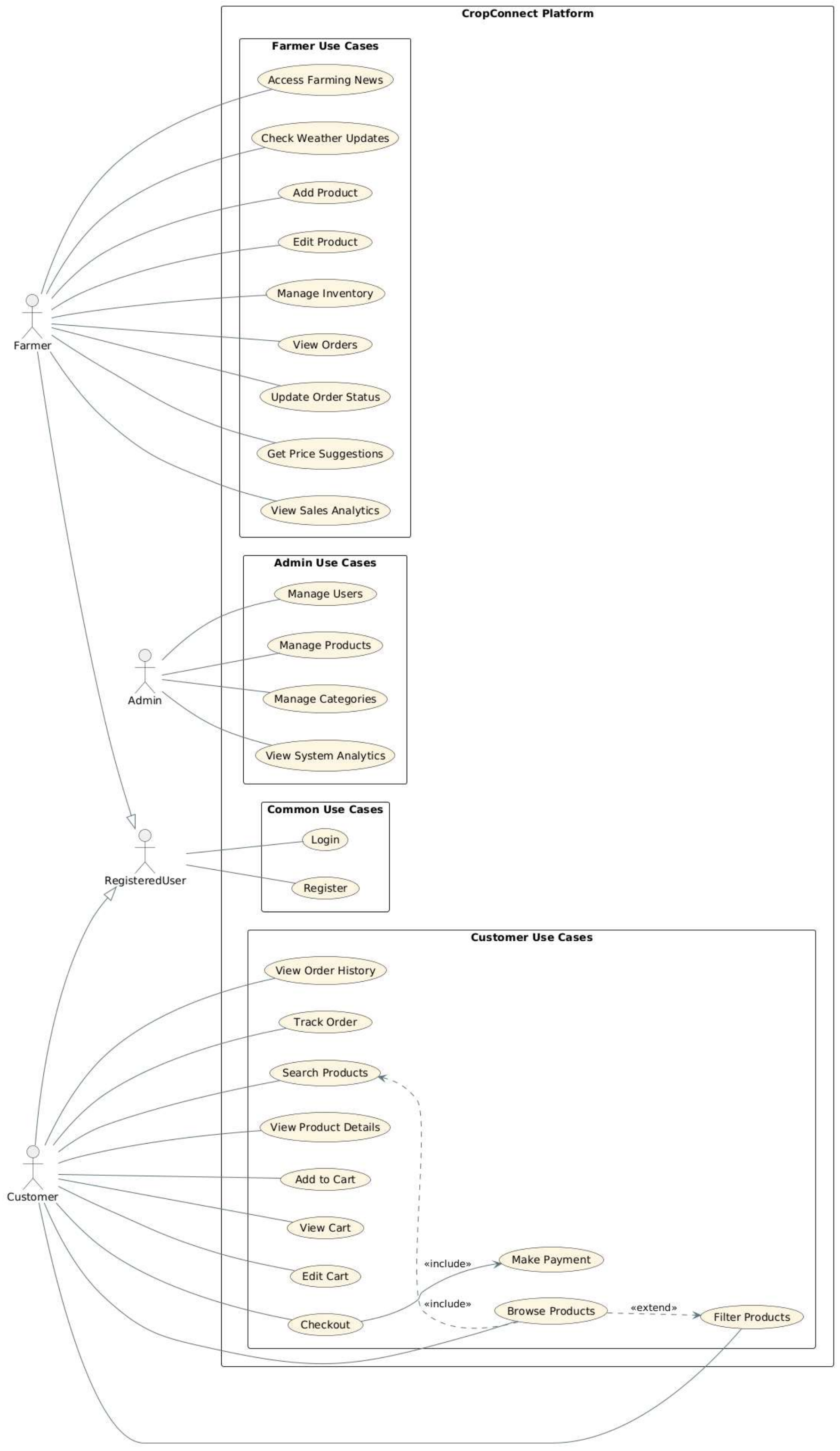


Figure 3.5 Use Case Diagram

Figure 3.5 illustrates the interactions of farmers, customers, and admins with the core functionalities of the CropConnect platform.

3.1.3 Sequence Diagram 1 – Customer Placing an Order

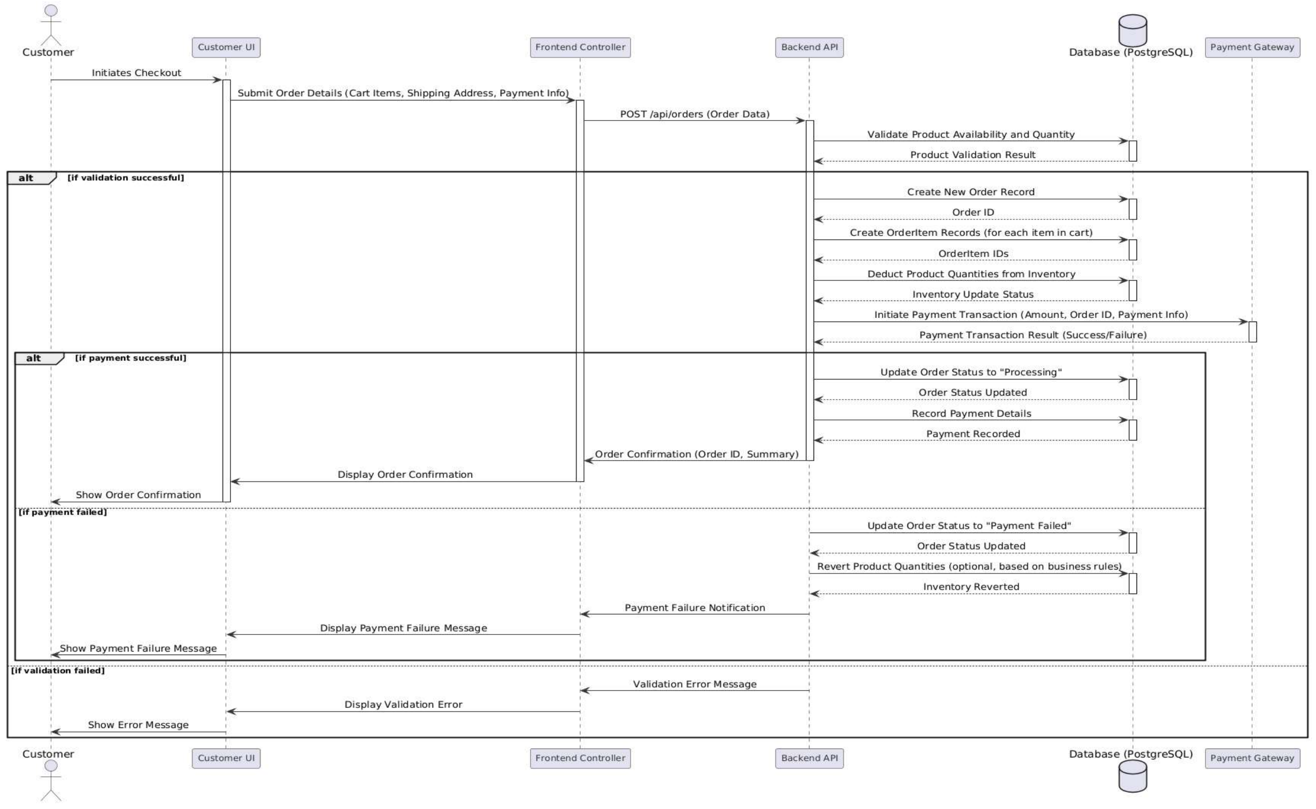

Customer
Customer UI
Frontend Controller
Backend API
Database (PostgreSQL)
Payment Gateway
Initiates Checkout
Submit Order Details (Cart Items, Shipping Address, Payment Info)
POST /api/orders (Order Data)
Validate Product Availability and Quantity
Product Validation Result
alt [if validation successful]
Create New Order Record
Order ID
Create OrderItem Records (for each item in cart)
OrderItem IDs
Deduct Product Quantities from Inventory
Inventory Update Status
Initiate Payment Transaction (Amount, Order ID, Payment Info)
Payment Transaction Result (Success/Failure)
alt [if payment successful]
Update Order Status to "Processing"
Order Status Updated
Record Payment Details
Payment Recorded
Order Confirmation (Order ID, Summary)
Display Order Confirmation
Show Order Confirmation
[if payment failed]
Update Order Status to "Payment Failed"
Order Status Updated
Revert Product Quantities (optional, based on business rules)
Inventory Reverted
Payment Failure Notification
Display Payment Failure Message
Show Payment Failure Message
[if validation failed]
Validation Error Message
Display Validation Error
Show Error Message

Figure 3.6 Sequence Diagram 1 – Customer Placing an Order

Figure 3.6 shows the sequence of interactions when a customer checks out and places an order.

3.1.4 Sequence Diagram 2 – Farmer Adding a Product

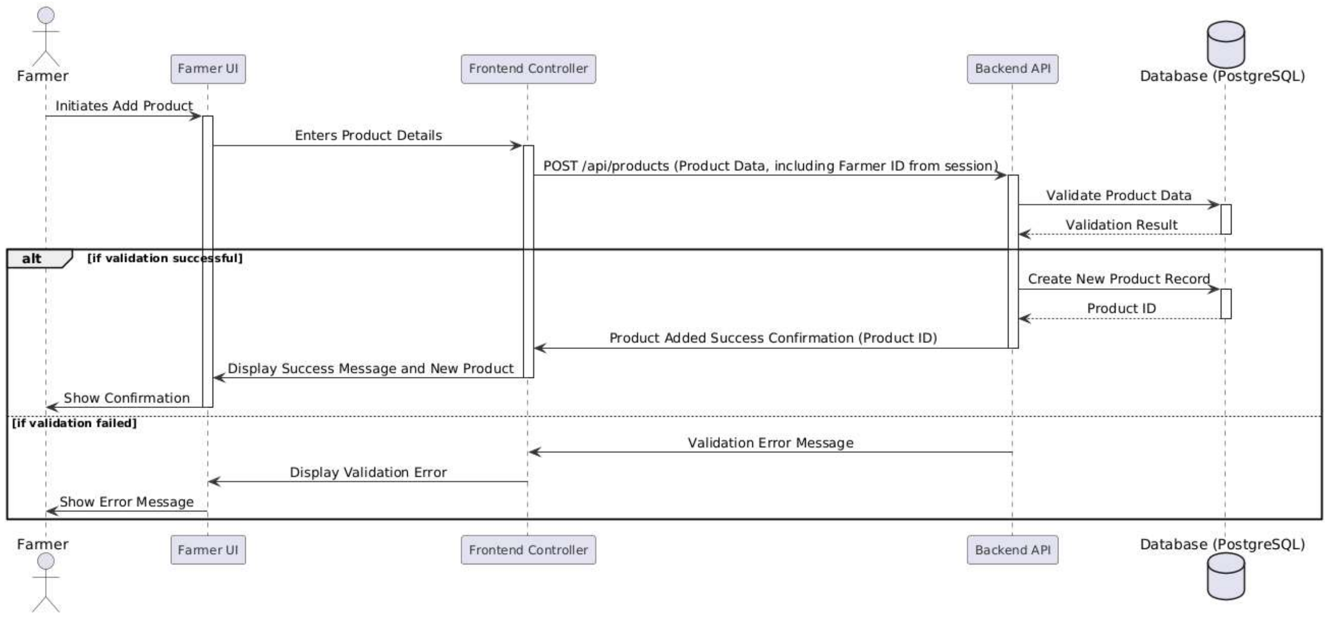


Figure 3.7 Sequence Diagram 2 – Farmer Adding a Product

Figure 3.7 shows the sequence of interactions when a farmer adds a new product to their catalog.

3.1.5 Sequence Diagram 3 – Farmer Requesting Price Prediction

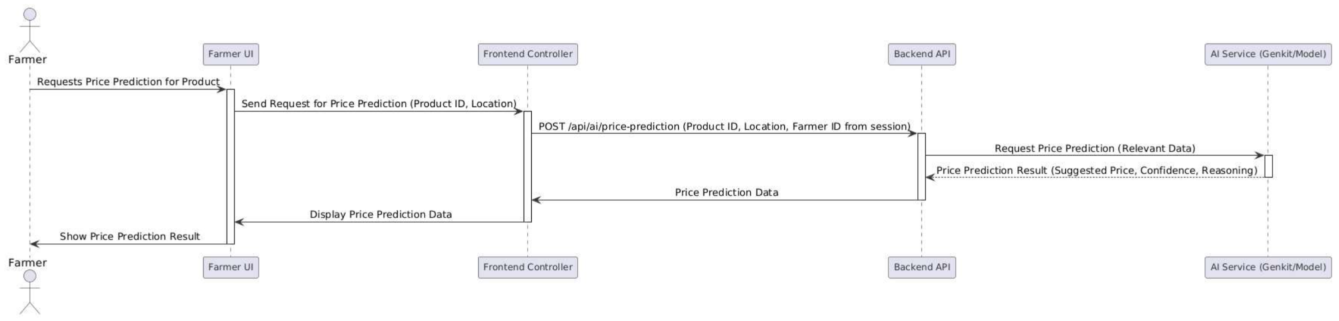

Farmer
Farmer UI
Frontend Controller
Backend API
AI Service (Genkit/Model)
Requests Price Prediction for Product
Send Request for Price Prediction (Product ID, Location)
POST /api/ai/price-prediction (Product ID, Location, Farmer ID from session)
Request Price Prediction (Relevant Data)
Price Prediction Result (Suggested Price, Confidence, Reasoning)
Price Prediction Data
Display Price Prediction Data
Show Price Prediction Result
Farmer
Farmer UI
Frontend Controller
Backend API
AI Service (Genkit/Model)

Figure 3.8 Sequence Diagram 3 – Farmer Requesting Price Prediction

Figure 3.8 shows the sequence of interactions when a farmer adds a new product to their catalog.

4. IMPLEMENTATION AND RESULTS

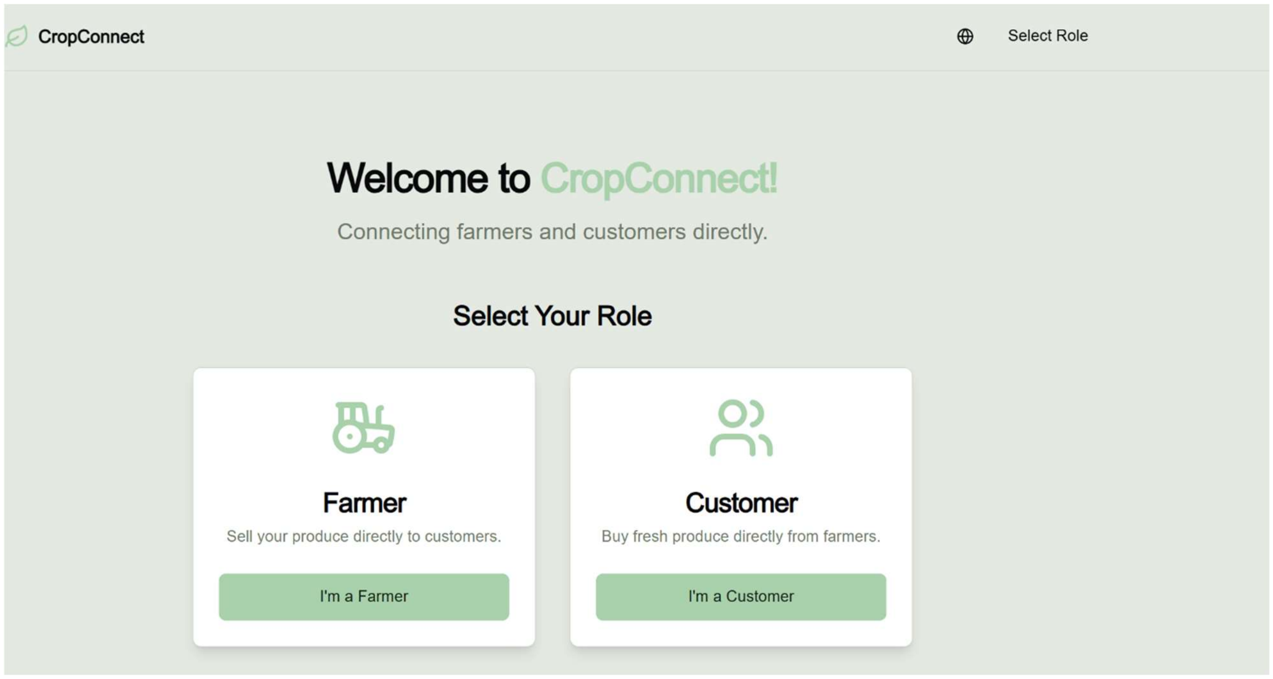

CropConnect
Select Role
Welcome to CropConnect!
Connecting farmers and customers directly.
Select Your Role
Farmer
Sell your produce directly to customers.
I'm a Farmer
Customer
Buy fresh produce directly from farmers.
I'm a Customer

Figure 4.1: CropConnect Role Selection Interface

Figure 4.1 displays the initial welcome screen of the CropConnect platform, allowing users to choose their role as either a 'Farmer' or a 'Customer' to begin their interaction with the marketplace.

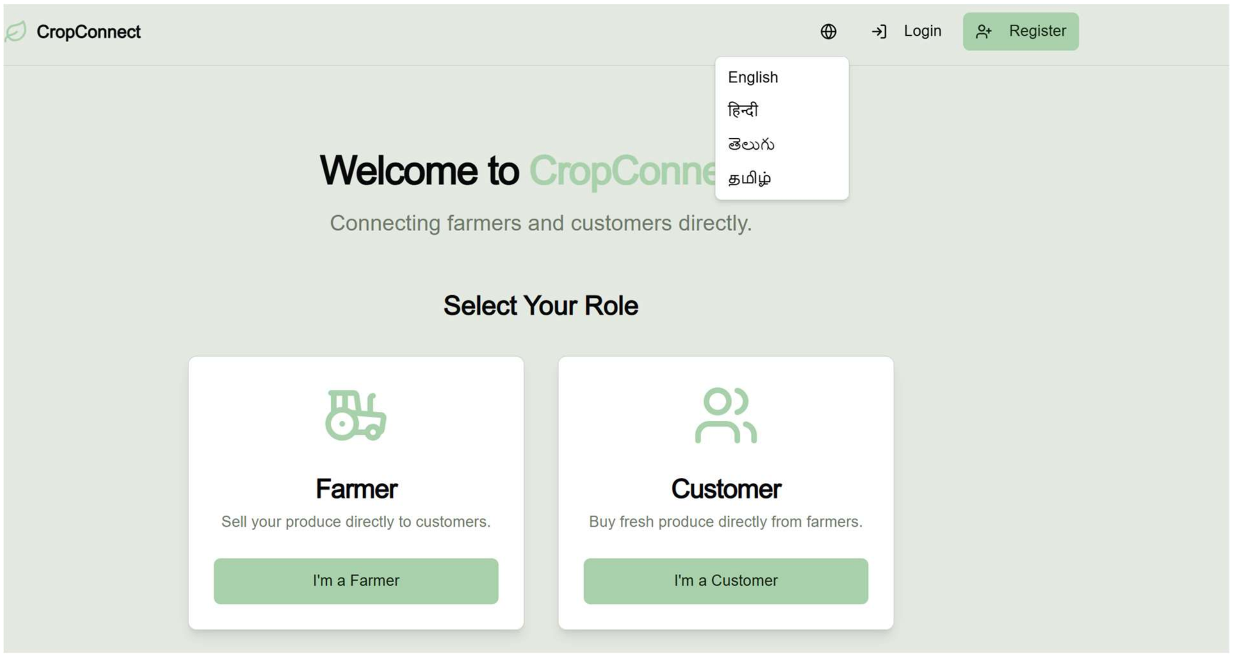
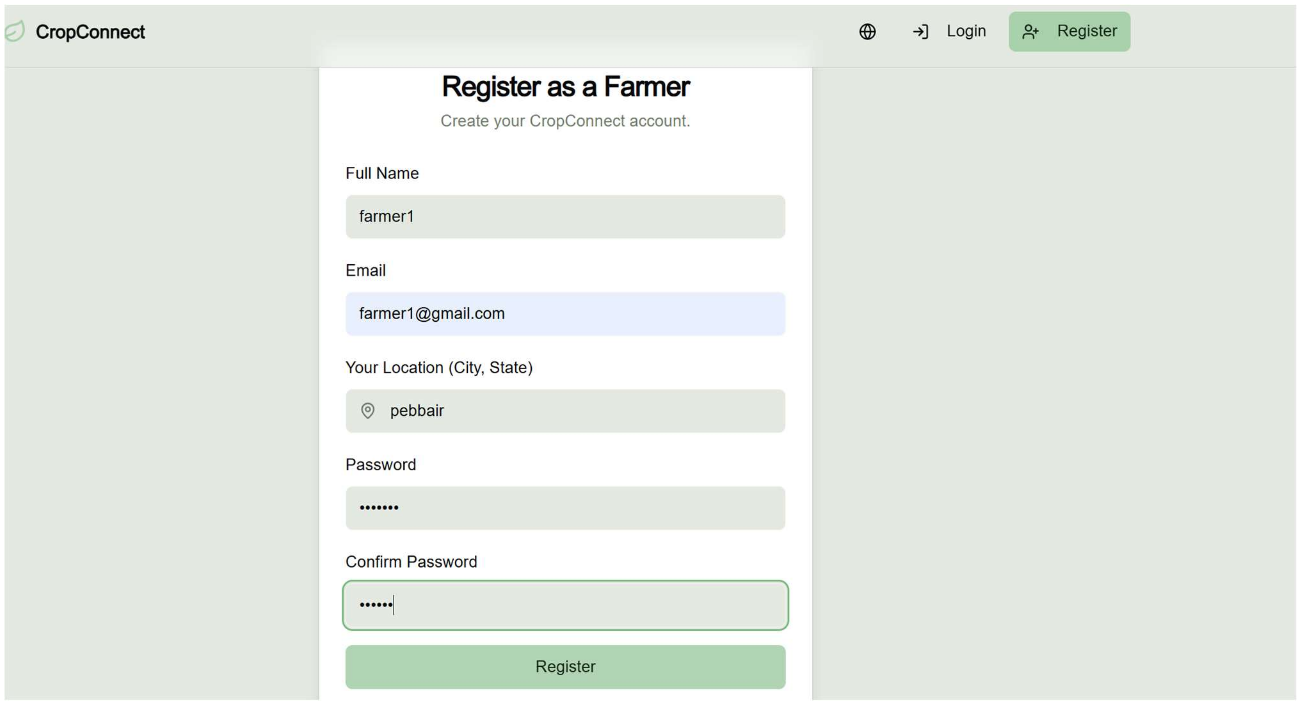


Figure 4.2: Language Selection Feature

Figure 4.2 highlights the multi-language support available on the CropConnect welcome screen, allowing users to select their preferred language for interface interaction.


CropConnect

Login

Register

Register as a Farmer

Create your CropConnect account.

Full Name

farmer1

Email

farmer1@gmail.com

Your Location (City, State)

pebbair

Password

•••••••

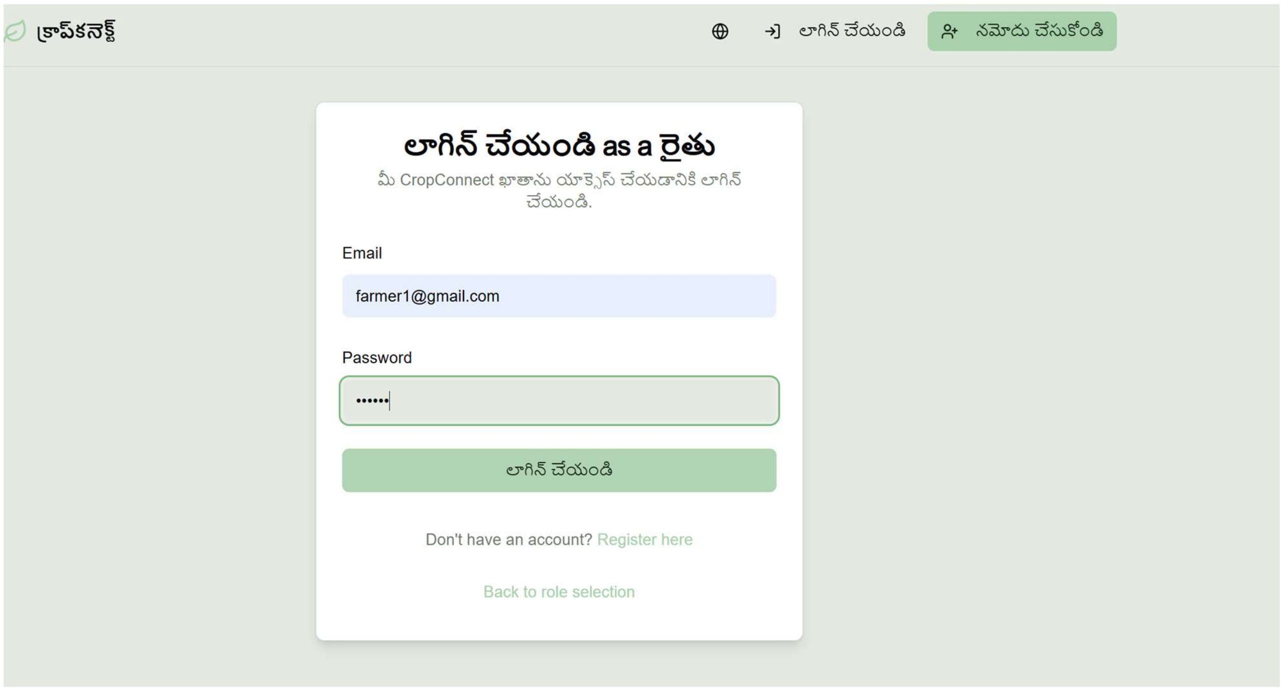
Confirm Password

••••••

Register

Figure 4.3: Farmer Registration Form

Figure 4.3 illustrates the registration interface for farmers on CropConnect, where they can input their personal details and create an account to begin selling their produce.


క్రాప్‌కనెక్ట్

లాగిన్ చేయండి

నమోదు చేసుకోండి

లాగిన్ చేయండి as a రైతు

మీ CropConnect ఖాతాను యాక్సెస్ చేయడానికి లాగిన్ చేయండి.

Email

farmer1@gmail.com

Password

••••••

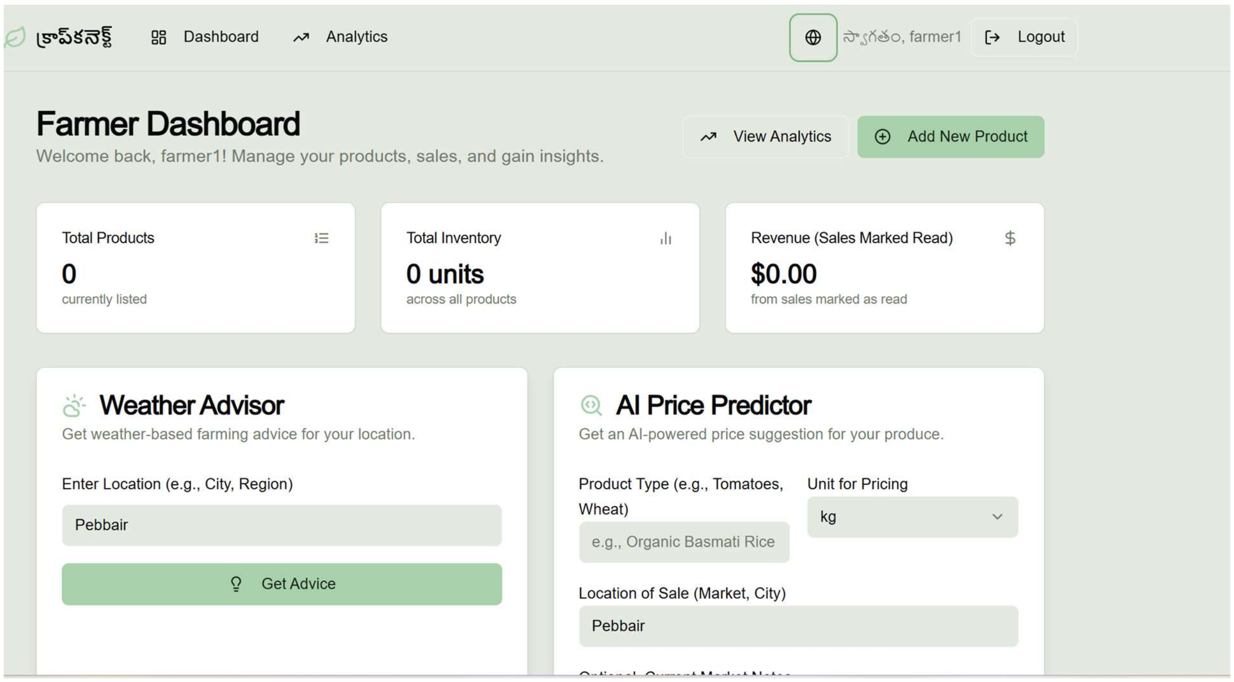
లాగిన్ చేయండి

Don't have an account? Register here

Back to role selection

Figure 4.4: Farmer Login Interface

Figure 4.4 displays the login screen for farmers on CropConnect, allowing registered farmers to access their accounts to manage products and orders.



Farmer Dashboard

Welcome back, farmer1! Manage your products, sales, and gain insights.

Total Products
0
currently listed

Total Inventory
0 units
across all products

Revenue (Sales Marked Read)
$0.00
from sales marked as read

Weather Advisor

Get weather-based farming advice for your location.

Enter Location (e.g., City, Region): Pebbair

AI Price Predictor

Get an AI-powered price suggestion for your produce.

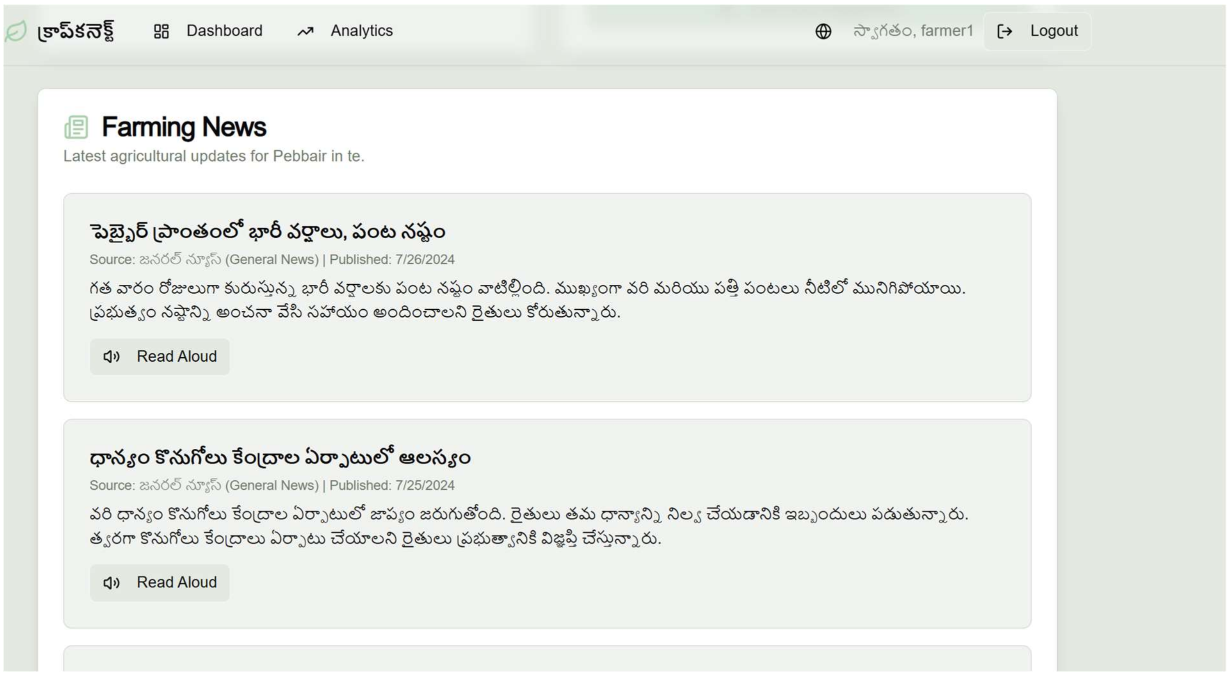
Product Type (e.g., Tomatoes, Wheat): e.g., Organic Basmati Rice

Unit for Pricing: kg

Location of Sale (Market, City): Pebbair

Figure 4.5: Farmer Dashboard Overview

This figure displays the main dashboard for farmers on CropConnect, offering a summary of their product listings, inventory, sales, and providing quick access to features like weather advisories and AI-powered price predictions.



Farming News

Latest agricultural updates for Pebbair in te.

పెబ్బైర్ ప్రాంతంలో భారీ వర్షాలు, పంట నష్టం

Source: జనరల్ న్యూస్ (General News) | Published: 7/26/2024

గత వారం రోజులుగా కురుస్తున్న భారీ వర్షాలకు పంట నష్టం వాటిల్లింది. ముఖ్యంగా వరి మరియు పత్తి పంటలు నీటిలో మునిగిపోయాయి. ప్రభుత్వం నష్టాన్ని అంచనా వేసి సహాయం అందించాలని రైతులు కోరుతున్నారు.

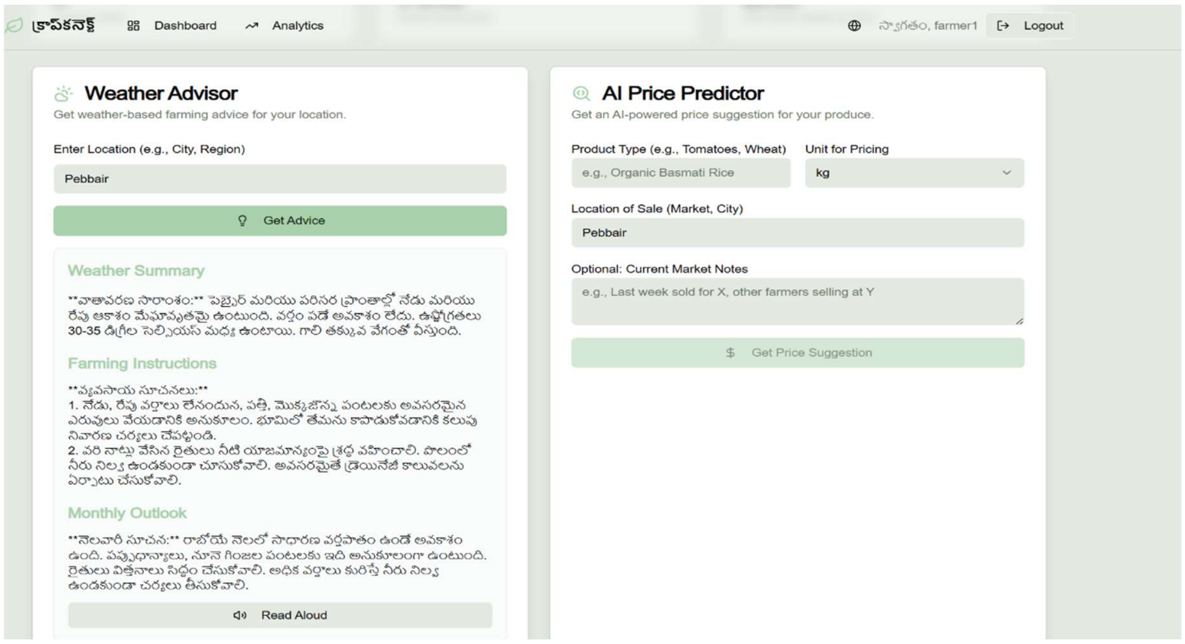
ధాన్యం కొనుగోలు కేంద్రాల ఏర్పాటులో ఆలస్యం

Source: జనరల్ న్యూస్ (General News) | Published: 7/25/2024

వరి ధాన్యం కొనుగోలు కేంద్రాల ఏర్పాటులో జాప్యం జరుగుతోంది. రైతులు తమ ధాన్యాన్ని నిల్వ చేయడానికి ఇబ్బందులు పడుతున్నారు. త్వరగా కొనుగోలు కేంద్రాలు ఏర్పాటు చేయాలని రైతులు ప్రభుత్వానికి విజ్ఞప్తి చేస్తున్నారు.

Figure 4.6: Farming News Dashboard

Figure 4.6 presents the "Farming News" section within the farmer's dashboard, providing farmers with the latest agricultural updates and news relevant to their location (Pebbair), presented in Telugu.



Weather Advisor

Get weather-based farming advice for your location.

Enter Location (e.g., City, Region): Pebbair

Weather Summary

వాతావరణ సారాంశం: పెబ్బైర్ మరియు పరిసర ప్రాంతాల్లో నేడు మరియు రేపు ఆకాశం మేఘావృతమై ఉంటుంది. వర్షం పడే అవకాశం లేదు. ఉష్ణోగ్రతలు 30-35 డిగ్రీల సెల్సియస్ మధ్య ఉంటాయి. గాలి తక్కువ వేగంతో వీస్తుంది.

Farming Instructions

వ్యవసాయ సూచనలు:
1. నేడు, రేపు వర్షాలు లేనందున, పత్తి, మొక్కజొన్న పంటలకు అవసరమైన ఎరువులు వేయడానికి అనుకూలం. భూమిలో తేమను కాపాడుకోవడానికి కలుపు నివారణ చర్యలు చేపట్టండి.
2. వరి నాట్లు వేసిన రైతులు నీటి యాజమాన్యంపై శ్రద్ధ వహించాలి. పొలంలో నీరు నిల్వ ఉండకుండా చూసుకోవాలి. అవసరమైతే డ్రెయినేజీ కాలువలను ఏర్పాటు చేసుకోవాలి.

Monthly Outlook

నెలవారీ సూచన: రాబోయే నెలలో సాధారణ వర్షపాతం ఉండే అవకాశం ఉంది. పప్పుధాన్యాలు, నూనె గింజల పంటలకు ఇది అనుకూలంగా ఉంటుంది. రైతులు విత్తనాలు సిద్ధం చేసుకోవాలి. అధిక వర్షాలు కురిస్తే నీరు నిల్వ ఉండకుండా చర్యలు తీసుకోవాలి.

AI Price Predictor

Get an AI-powered price suggestion for your produce.

Product Type (e.g., Tomatoes, Wheat): e.g., Organic Basmati Rice

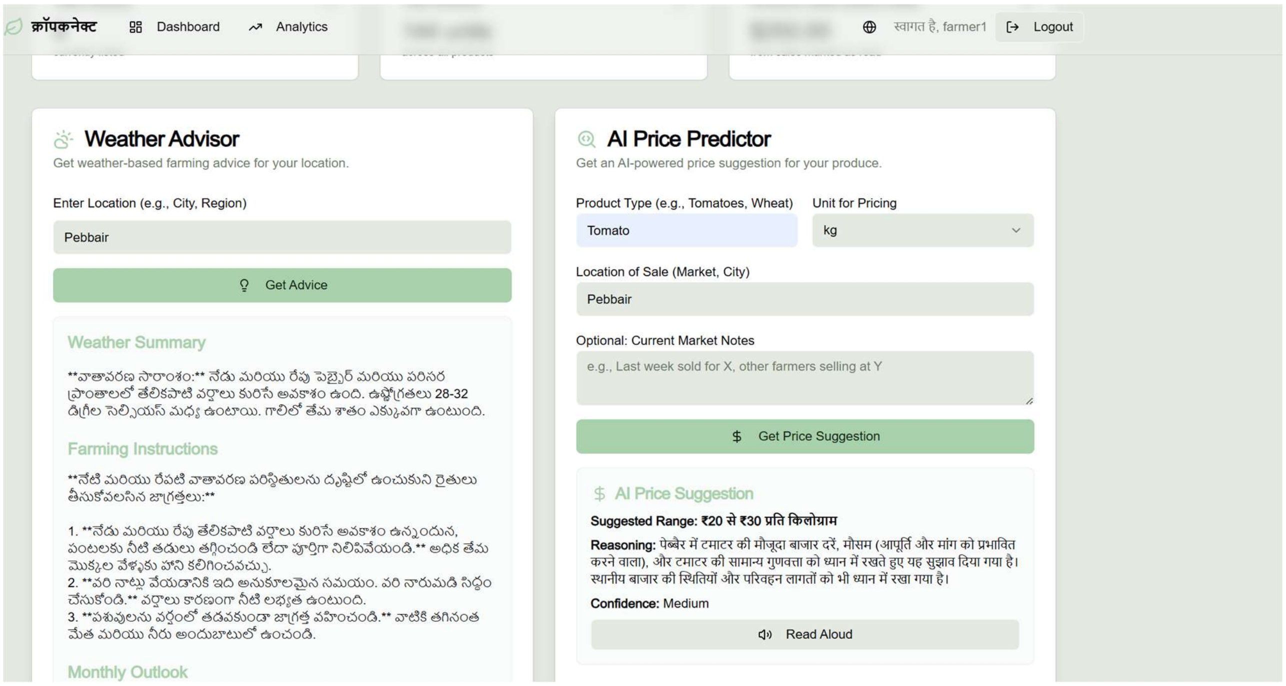
Unit for Pricing: kg

Location of Sale (Market, City): Pebbair

Optional: Current Market Notes: e.g., Last week sold for X, other farmers selling at Y

Figure 4.7: Weather Advisor Feature

Figure 4.7 highlights the "Weather Advisor" section within the farmer's dashboard, providing farmers with weather-based farming advice and a detailed summary of current conditions and a monthly outlook for their specified location.



Weather Advisor

Get weather-based farming advice for your location.

Enter Location (e.g., City, Region): Pebbair

Weather Summary

వాతావరణ సారాంశం: నేడు మరియు రేపు పెబ్బైర్ మరియు పరిసర ప్రాంతాలలో తేలికపాటి వర్షాలు కురిసే అవకాశం ఉంది. ఉష్ణోగ్రతలు 28-32 డిగ్రీల సెల్సియస్ మధ్య ఉంటాయి. గాలిలో తేమ శాతం ఎక్కువగా ఉంటుంది.

Farming Instructions

నేటి మరియు రేపటి వాతావరణ పరిస్థితులను దృష్టిలో ఉంచుకుని రైతులు తీసుకోవలసిన జాగ్రత్తలు:

1. **నేడు మరియు రేపు తేలికపాటి వర్షాలు కురిసే అవకాశం ఉన్నందున, పంటలకు నీటి తడులు తగ్గించండి లేదా పూర్తిగా నిలిపివేయండి.** అధిక తేమ మొక్కల వేళ్ళకు హాని కలిగించవచ్చు.
2. **వరి నాట్లు వేయడానికి ఇది అనుకూలమైన సమయం. వరి నారుమడి సిద్ధం చేసుకోండి.** వర్షాలు కారణంగా నీటి లభ్యత ఉంటుంది.
3. **పశువులను వర్షంలో తడవకుండా జాగ్రత్త వహించండి.** వాటికి తగినంత మేత మరియు నీరు అందుబాటులో ఉంచండి.

Monthly Outlook

AI Price Predictor

Get an AI-powered price suggestion for your produce.

Product Type (e.g., Tomatoes, Wheat): Tomato

Unit for Pricing: kg

Location of Sale (Market, City): Pebbair

Optional: Current Market Notes: e.g., Last week sold for X, other farmers selling at Y

AI Price Suggestion

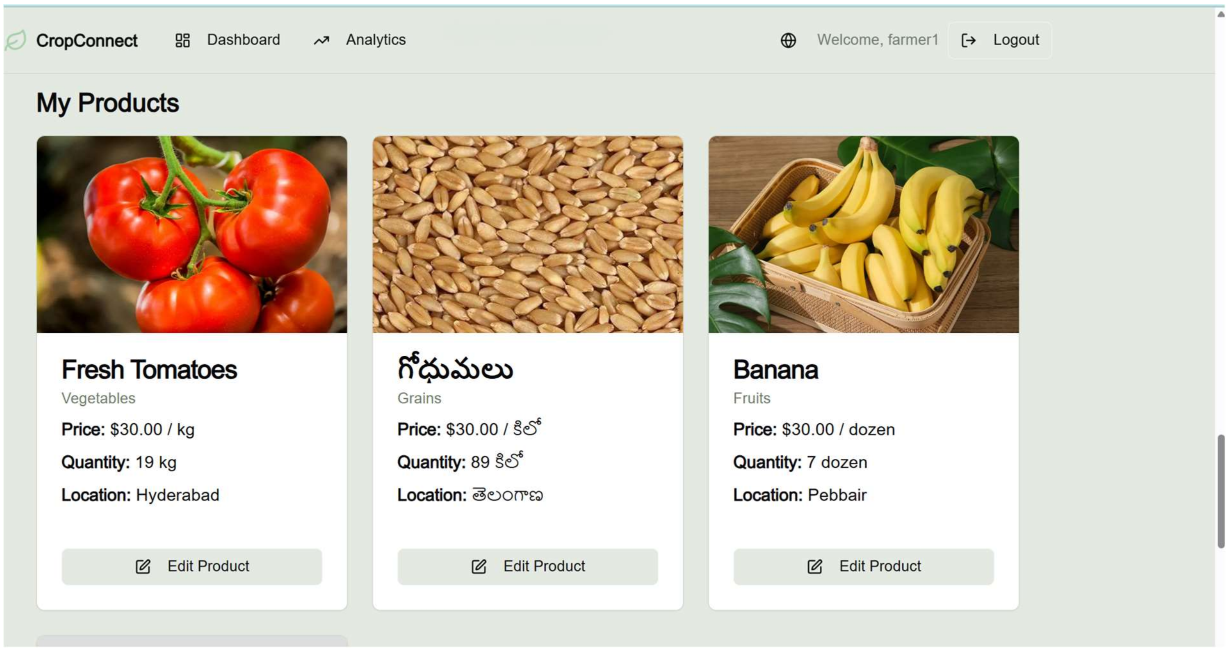
Suggested Range: ₹20 से ₹30 प्रति किलोग्राम

Reasoning: पेब्बैर में टमाटर की मौजूदा बाजार दरें, मौसम (आपूर्ति और मांग को प्रभावित करने वाला), और टमाटर की सामान्य गुणवत्ता को ध्यान में रखते हुए यह सुझाव दिया गया है। स्थानीय बाजार की स्थितियों और परिवहन लागतों को भी ध्यान में रखा गया है।

Confidence: Medium

Figure 4.8: AI Price Predictor Feature

Figure 4.8 displays the "AI Price Predictor" tool on the farmer's dashboard, demonstrating how farmers can input product details to receive AI-powered price suggestions and reasoning for their produce, such as tomatoes.



My Products

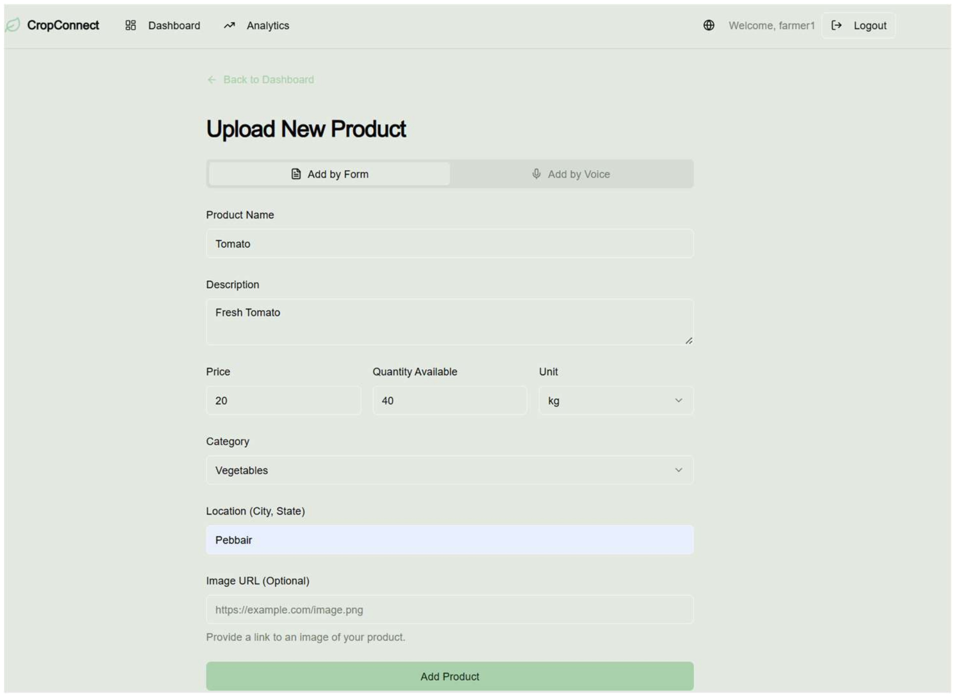
Fresh Tomatoes
Vegetables
Price: $30.00 / kg
Quantity: 19 kg
Location: Hyderabad
Edit Product

గోధుమలు
Grains
Price: $30.00 / కిలో
Quantity: 89 కిలో
Location: తెలంగాణ
Edit Product

Banana
Fruits
Price: $30.00 / dozen
Quantity: 7 dozen
Location: Pebbair
Edit Product

Figure 4.9 Farmer's Active Product Listings

Figure 4.9 displays the "My Products" section of the farmer's dashboard, showcasing their active product listings with details such as product name, category, price, available quantity, and location, along with an option to edit each product.



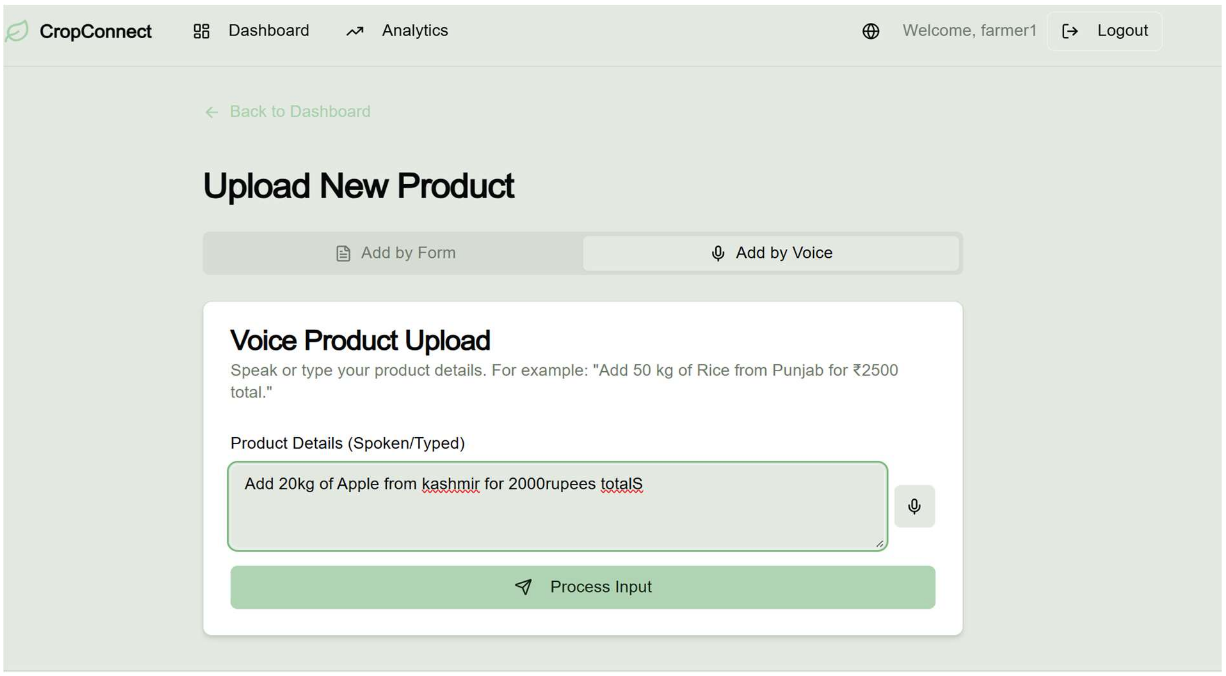
Upload New Product

Product Name	Tomato
Description	Fresh Tomato
Price	20
Quantity Available	40
Unit	kg
Category	Vegetables
Location (City, State)	Pebbair
Image URL (Optional)	https://example.com/image.png

Provide a link to an image of your product.

Figure 4.10 Farmer adding new product manually

Figure 4.10 displays the "Upload New Product" form, enabling farmers to manually enter details such as product name, description, price, quantity, category, and location before listing their produce on CropConnect.



Upload New Product

Voice Product Upload

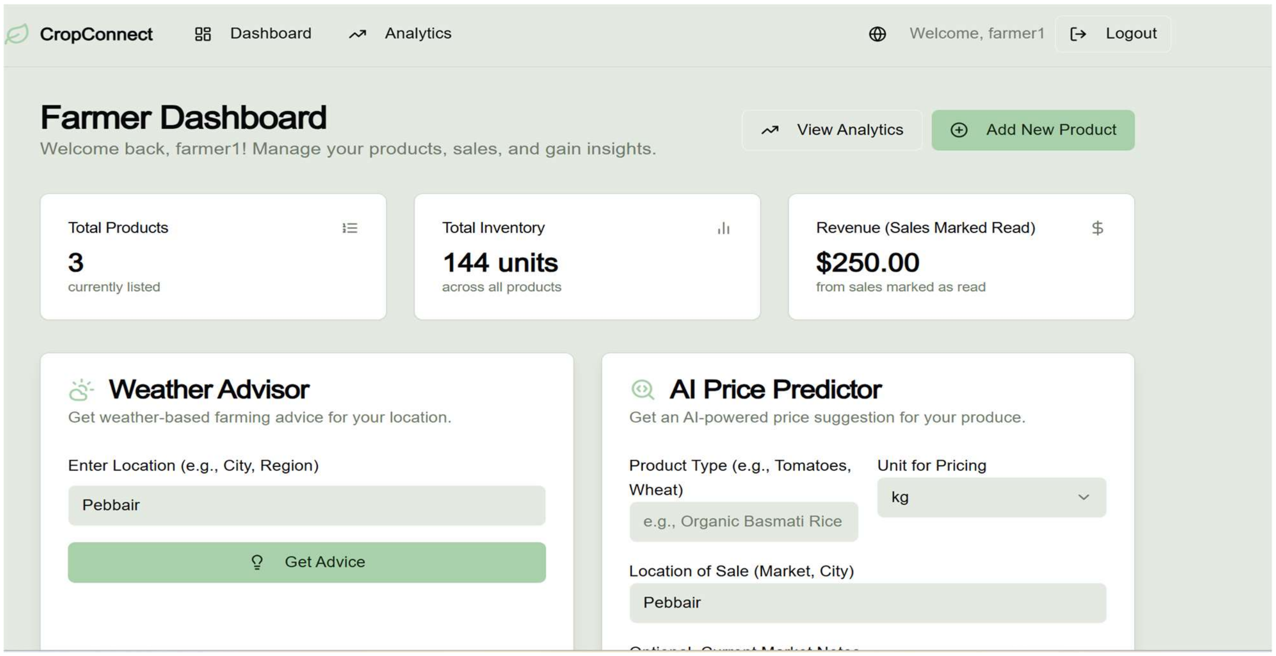
Speak or type your product details. For example: "Add 50 kg of Rice from Punjab for ₹2500 total."

Product Details (Spoken/Typed)

Add 20kg of Apple from kashmir for 2000rupees totalS

Figure 4.11: Voice-Based Product Upload Interface

Figure 4.11 illustrates the "Voice Product Upload" feature, allowing farmers to add new products to CropConnect by speaking or typing product details in a natural language format, which the system then processes.



Farmer Dashboard

Welcome back, farmer1! Manage your products, sales, and gain insights.

Total Products 3 currently listed

Total Inventory 144 units across all products

Revenue (Sales Marked Read) $250.00 from sales marked as read

Weather Advisor

Get weather-based farming advice for your location.

Enter Location (e.g., City, Region): Pebbair

AI Price Predictor

Get an AI-powered price suggestion for your produce.

Product Type (e.g., Tomatoes, Wheat): e.g., Organic Basmati Rice

Unit for Pricing: kg

Location of Sale (Market, City): Pebbair

Figure 4.12: Farmer Dashboard After Product Additions

Figure 4.12 showcases the farmer's dashboard reflecting updated metrics after new products have been added, displaying the total number of listed products, total inventory units, and accumulated revenue from sales.

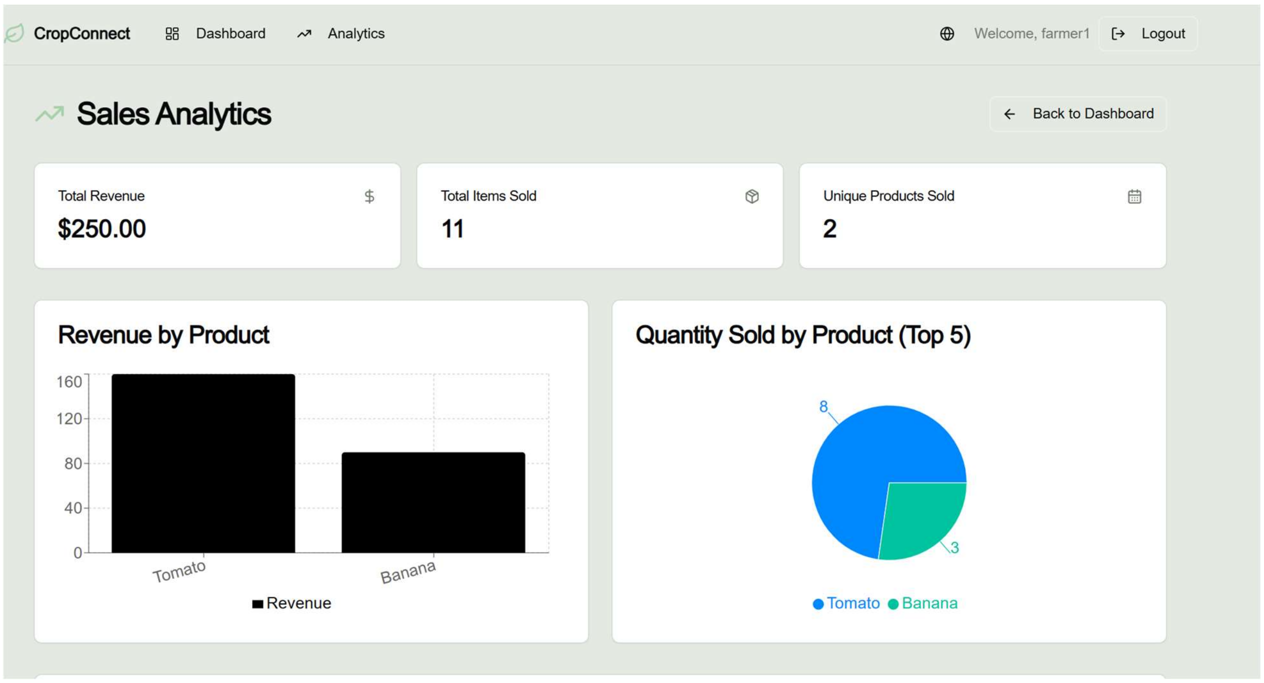
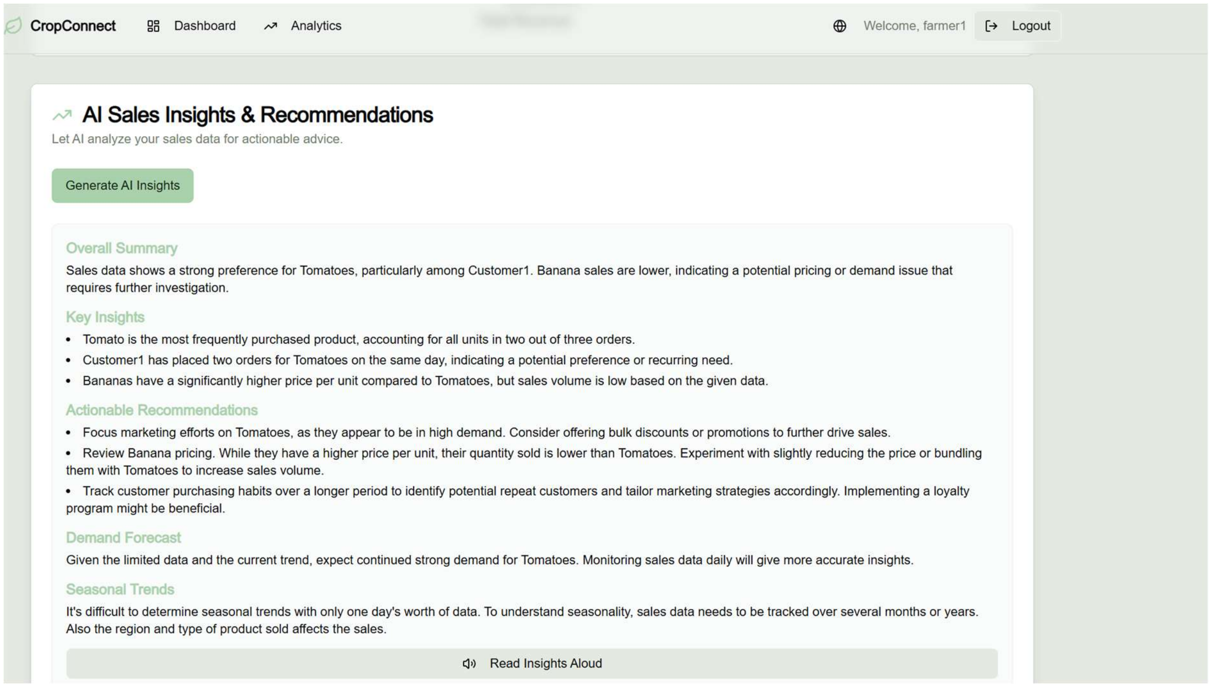


Figure 4.13: Farmer Sales Analytics Dashboard

This figure presents the "Sales Analytics" dashboard for farmers, providing a comprehensive overview of their sales performance including total revenue, total items sold, unique products sold, and visual breakdowns of revenue and quantity sold by product.



AI Sales Insights & Recommendations

Let AI analyze your sales data for actionable advice.

Generate AI Insights

Overall Summary

Sales data shows a strong preference for Tomatoes, particularly among Customer1. Banana sales are lower, indicating a potential pricing or demand issue that requires further investigation.

Key Insights

- Tomato is the most frequently purchased product, accounting for all units in two out of three orders.
- Customer1 has placed two orders for Tomatoes on the same day, indicating a potential preference or recurring need.
- Bananas have a significantly higher price per unit compared to Tomatoes, but sales volume is low based on the given data.

Actionable Recommendations

- Focus marketing efforts on Tomatoes, as they appear to be in high demand. Consider offering bulk discounts or promotions to further drive sales.
- Review Banana pricing. While they have a higher price per unit, their quantity sold is lower than Tomatoes. Experiment with slightly reducing the price or bundling them with Tomatoes to increase sales volume.
- Track customer purchasing habits over a longer period to identify potential repeat customers and tailor marketing strategies accordingly. Implementing a loyalty program might be beneficial.

Demand Forecast

Given the limited data and the current trend, expect continued strong demand for Tomatoes. Monitoring sales data daily will give more accurate insights.

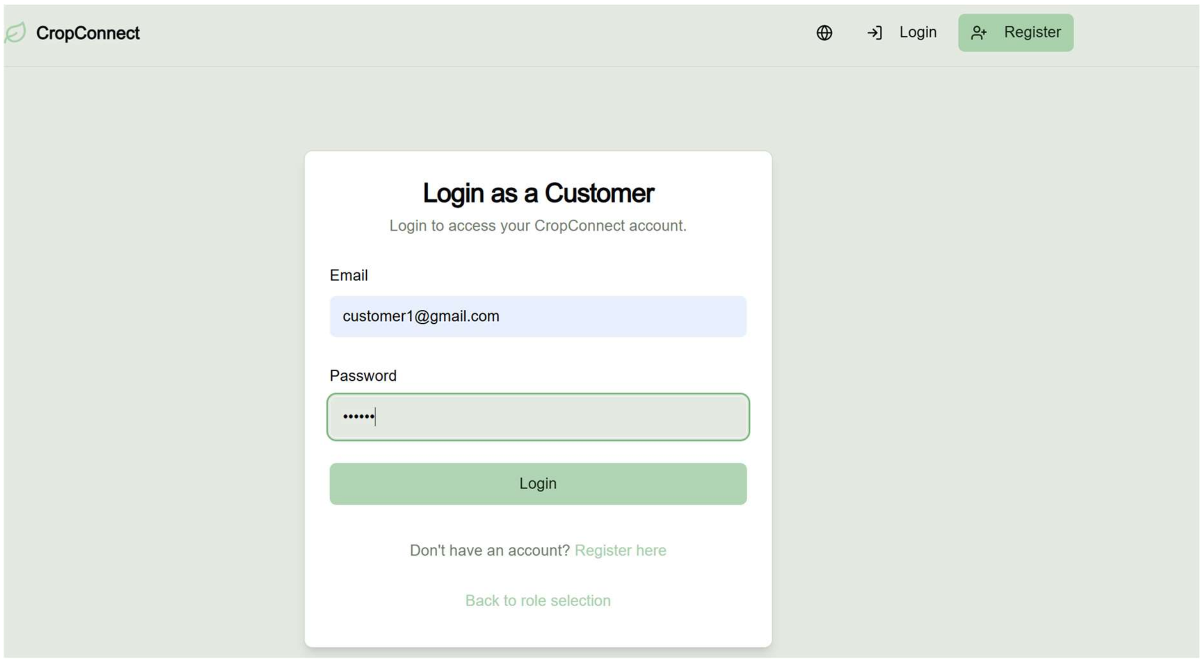
Seasonal Trends

It's difficult to determine seasonal trends with only one day's worth of data. To understand seasonality, sales data needs to be tracked over several months or years. Also the region and type of product sold affects the sales.

Read Insights Aloud

Figure 4.14: AI Sales Insights and Recommendations

This figure displays the "AI Sales Insights & Recommendations" feature, which analyzes a farmer's sales data to provide actionable advice, including an overall summary, key insights, demand forecasts, and seasonal trends to help optimize sales strategies.



Login as a Customer

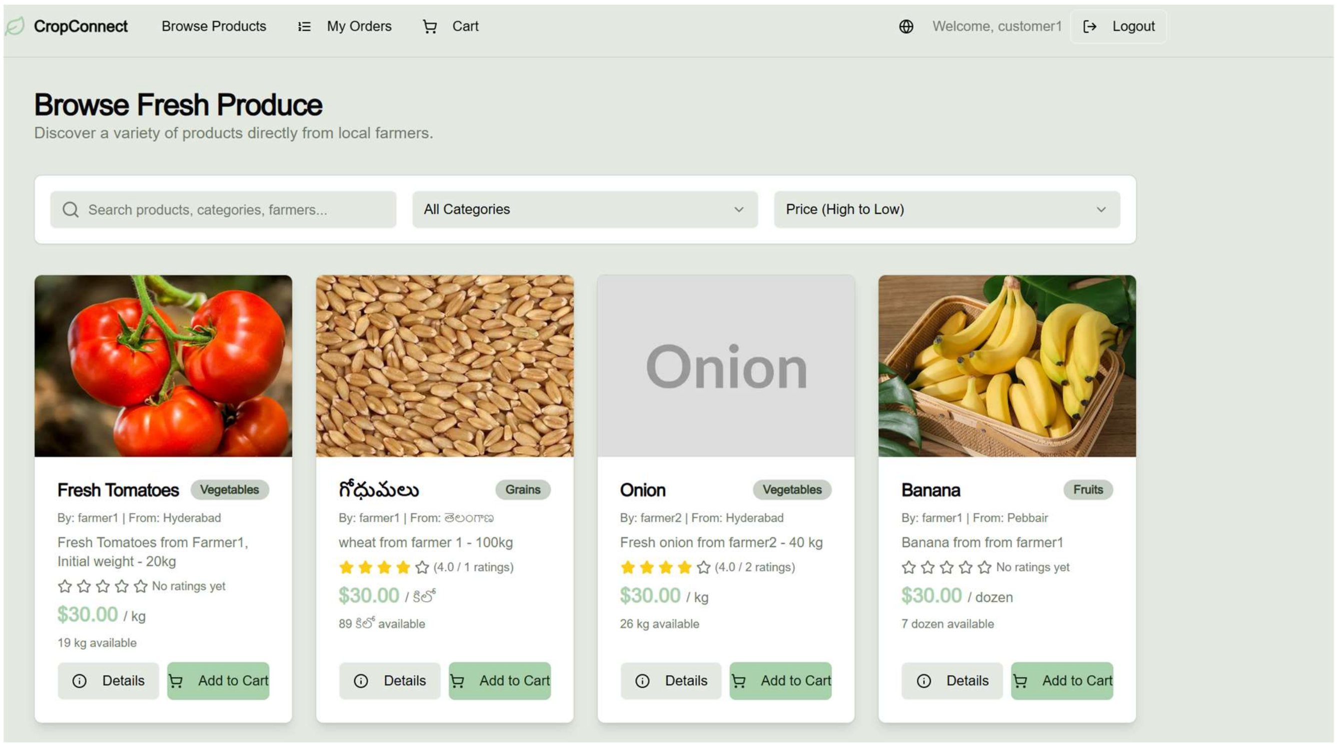
Login to access your CropConnect account.

Email: customer1@gmail.com

Password: ••••••

Figure 4.15: Customer Login Interface

Figure 4.15 presents the login screen for customers on CropConnect, allowing registered customers to enter their email and password to access their accounts and browse products.



Browse Fresh Produce

Discover a variety of products directly from local farmers.

Search products, categories, farmers...

All Categories

Price (High to Low)

Fresh Tomatoes — Vegetables
By: farmer1 | From: Hyderabad
Fresh Tomatoes from Farmer1, Initial weight - 20kg
No ratings yet
$30.00 / kg
19 kg available

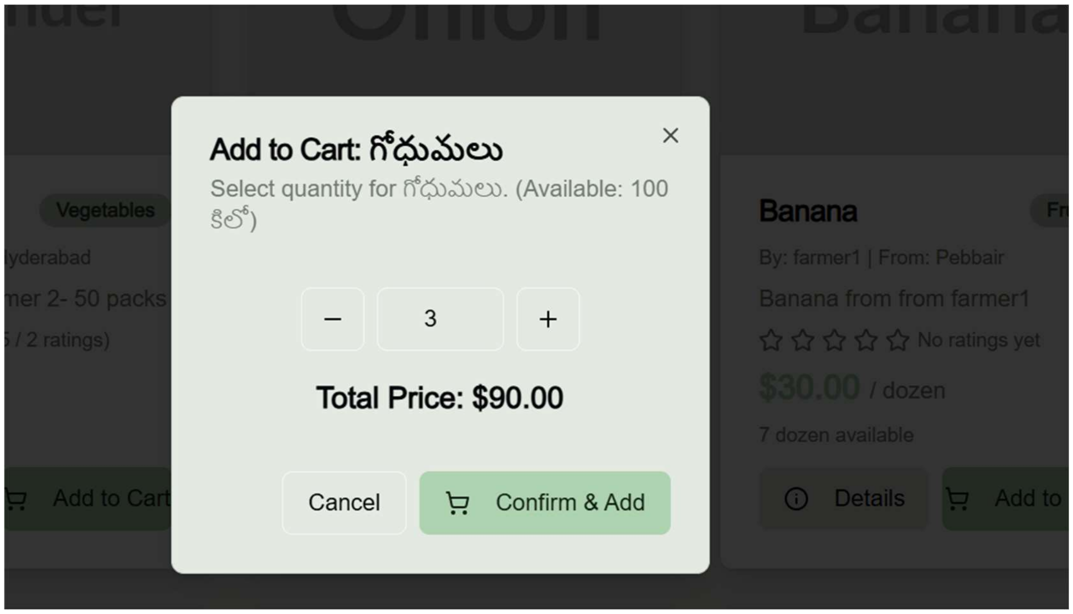
గోధుమలు — Grains
By: farmer1 | From: తెలంగాణ
wheat from farmer 1 - 100kg
(4.0 / 1 ratings)
$30.00 / కిలో
89 కిలో available

Onion — Vegetables
By: farmer2 | From: Hyderabad
Fresh onion from farmer2 - 40 kg
(4.0 / 2 ratings)
$30.00 / kg
26 kg available

Banana — Fruits
By: farmer1 | From: Pebbair
Banana from from farmer1
No ratings yet
$30.00 / dozen
7 dozen available

Figure 4.16: Customer Product Browse Interface

This figure displays the main interface for customers to browse fresh produce on CropConnect, allowing them to search, filter, and view various products listed by local farmers, along with options to view details or add items to their cart.

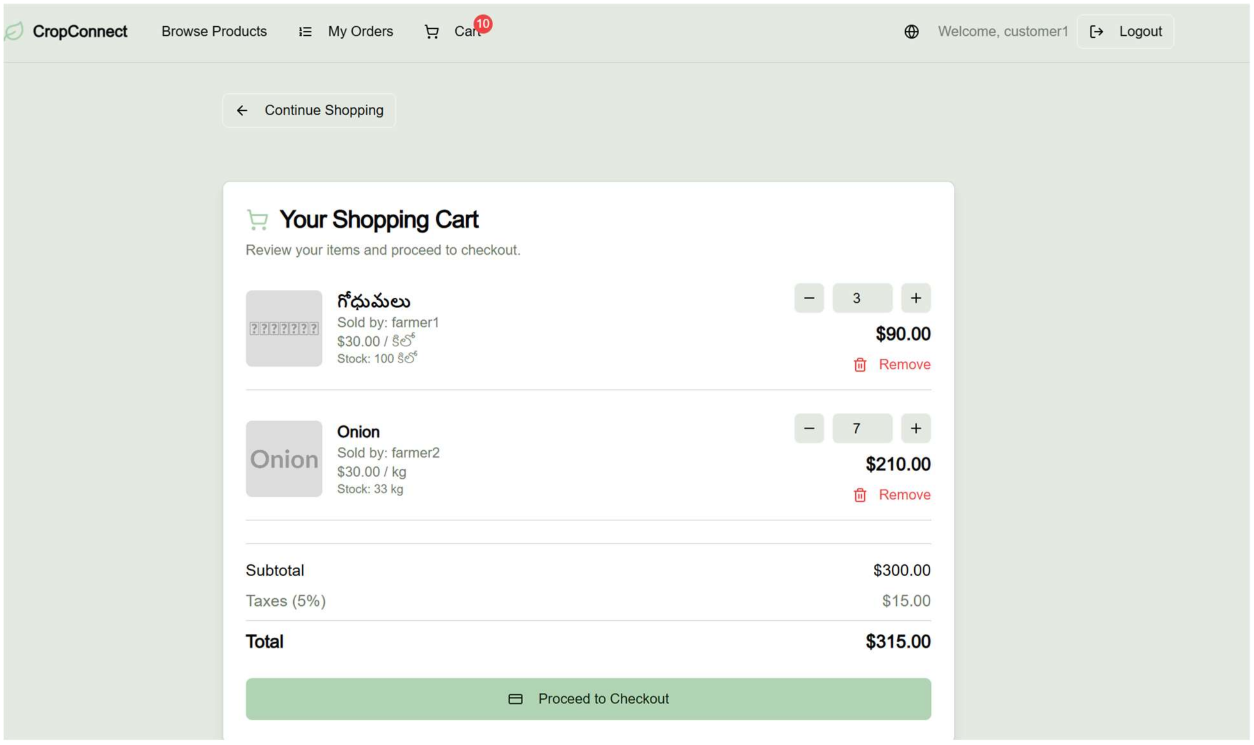

Add to Cart: గోధుమలు

Select quantity for గోధుమలు. (Available: 100 కిలో)

Total Price: $90.00

Figure 4.17: Customer Adding Product to Cart

Figure 4.17 shows the pop-up modal displayed when a customer adds a product to their cart, allowing them to specify the desired quantity and view the calculated total price before confirming the addition.



Your Shopping Cart

Review your items and proceed to checkout.

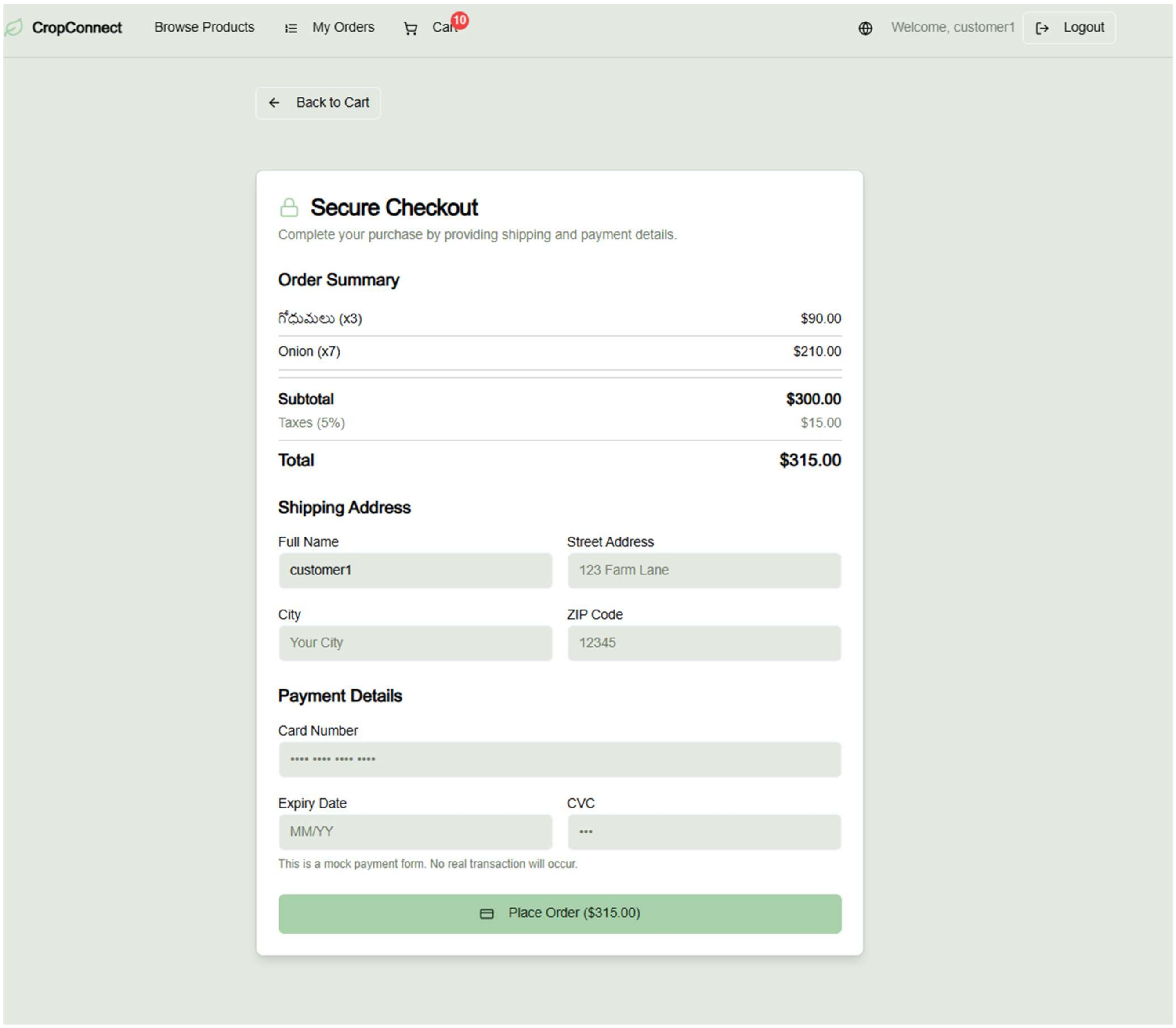
గోధుమలు
Sold by: farmer1
$30.00 / కిలో
Stock: 100 కిలో
3 — $90.00 — Remove

Onion
Sold by: farmer2
$30.00 / kg
Stock: 33 kg
7 — $210.00 — Remove

Subtotal	$300.00
Taxes (5%)	$15.00
Total	$315.00

Figure 4.18: Customer Shopping Cart

Figure 4.18 displays the customer's shopping cart section, where they can review selected products, adjust quantities, remove items, view the subtotal, taxes, and grand total, before proceeding to checkout.



Secure Checkout

Complete your purchase by providing shipping and payment details.

Order Summary

గోధుమలు (x3)	$90.00
Onion (x7)	$210.00
Subtotal	$300.00
Taxes (5%)	$15.00
Total	$315.00

Shipping Address

Full Name	customer1
Street Address	123 Farm Lane
City	Your City
ZIP Code	12345

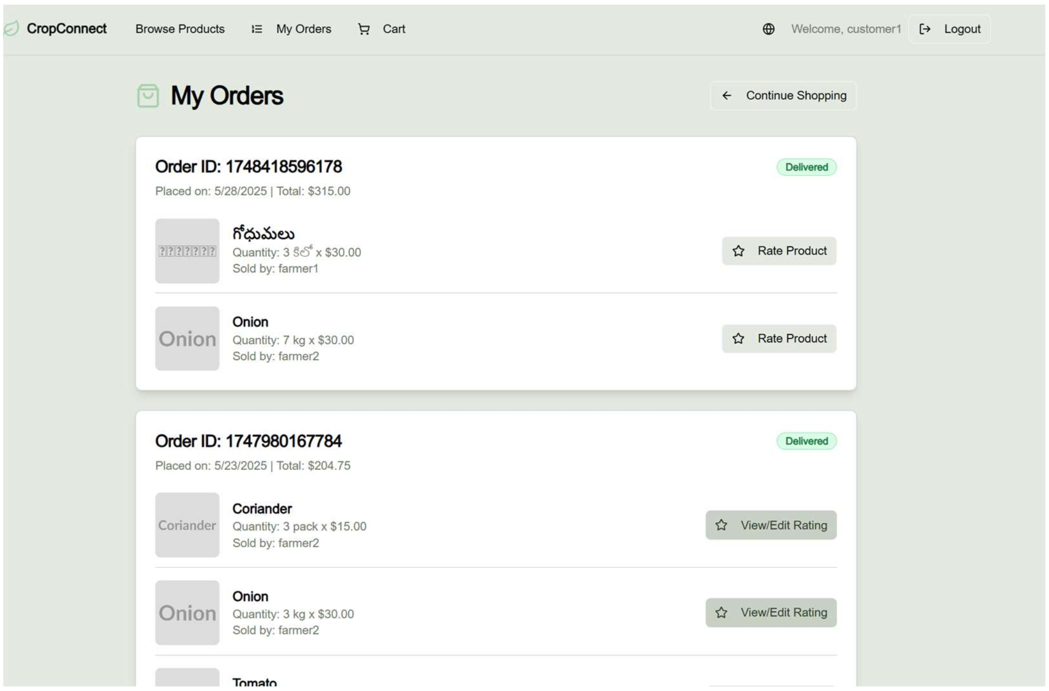
Payment Details

Card Number	•••• •••• •••• ••••
Expiry Date	MM/YY
CVC	•••

This is a mock payment form. No real transaction will occur.

Figure 4.19: Customer Checkout Section

Figure 4.19 illustrates the secure checkout interface for customers on CropConnect, displaying the order summary, allowing entry of shipping address details, and capturing payment information to complete the purchase.



My Orders

Order ID: 1748418596178 — Delivered
Placed on: 5/28/2025 | Total: $315.00

గోధుమలు
Quantity: 3 కిలో x $30.00
Sold by: farmer1
Rate Product

Onion
Quantity: 7 kg x $30.00
Sold by: farmer2
Rate Product

Order ID: 1747980167784 — Delivered
Placed on: 5/23/2025 | Total: $204.75

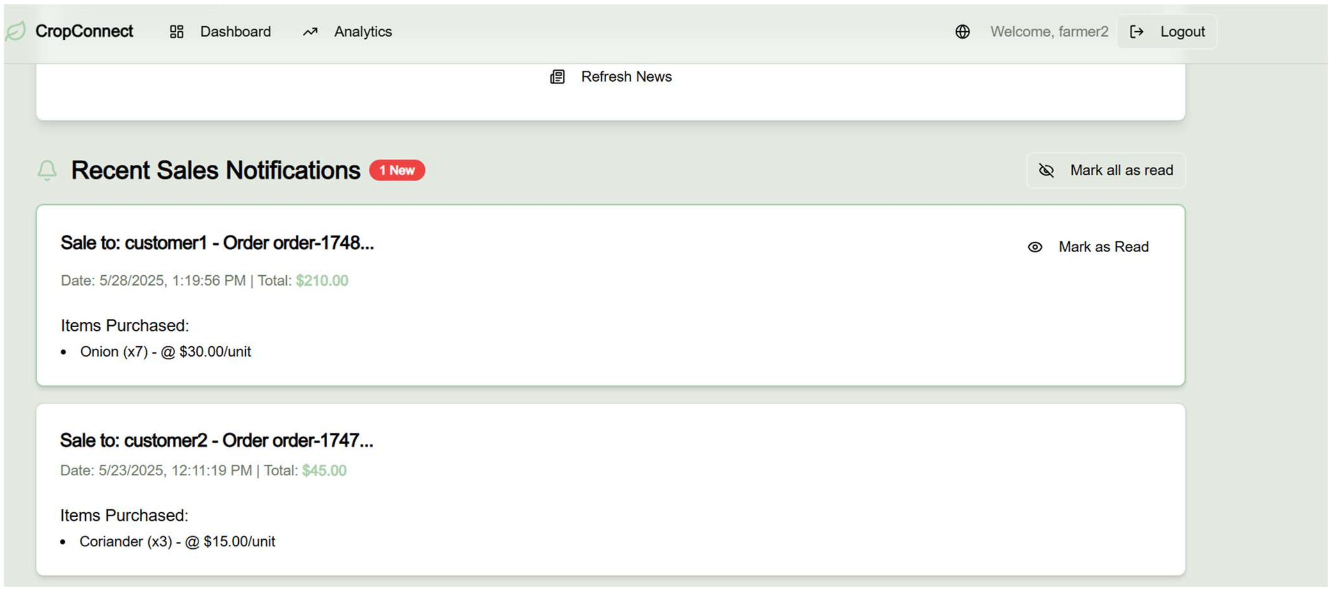
Coriander
Quantity: 3 pack x $15.00
Sold by: farmer2
View/Edit Rating

Onion
Quantity: 3 kg x $30.00
Sold by: farmer2
View/Edit Rating

Tomato

Figure 4.20: Customer 'My Orders' Section

Figure 4.20 displays the "My Orders" section for customers, showcasing a comprehensive history of their previous purchases with details like Order ID, date, total amount, and the status of each order, alongside options to rate products or view/edit existing ratings.



Recent Sales Notifications 1 New

Sale to: customer1 - Order order-1748...
Date: 5/28/2025, 1:19:56 PM | Total: $210.00
Items Purchased:
- Onion (x7) - @ $30.00/unit

Sale to: customer2 - Order order-1747...
Date: 5/23/2025, 12:11:19 PM | Total: $45.00
Items Purchased:
- Coriander (x3) - @ $15.00/unit

Figure 4.21: Recent Sales Notifications for Farmers

Figure 4.21 illustrates the "Recent Sales Notifications" section on the farmer's dashboard, providing a clear overview of recent orders, including buyer details, order ID, total amount, and items purchased.

5. CONCLUSION AND FUTURE WORK

5.1 Conclusion

The CropConnect platform represents a significant step towards fostering a more direct, transparent, and efficient agricultural marketplace, effectively bridging the gap between local farmers and consumers. By leveraging technology to streamline the buying and selling process, CropConnect provides a comprehensive solution that addresses key challenges in the traditional agricultural supply chain, benefiting both producers and consumers. The platform's design prioritizes ease of use, accessibility, and the integration of smart features to enhance the experience for all users.

CropConnect offers a suite of tailored functionalities designed to empower its diverse user base. For farmers, the platform provides intuitive tools for managing product listings, tracking inventory, and handling orders, enabling them to reach a wider customer base and potentially increase their income. The integration of AI-powered features like price prediction and access to relevant farming news and weather updates equips farmers with valuable insights to make informed decisions. Consumers, on the other hand, benefit from easy access to fresh, local produce, the ability to browse and search for specific items, and a straightforward checkout process, ensuring a convenient and reliable way to support local agriculture and access high-quality food.

In summary, CropConnect stands as a versatile and impactful platform, demonstrating the potential of technology to revolutionize the agricultural sector. By creating a direct connection between farmers and consumers and incorporating intelligent features, CropConnect not only simplifies transactions but also contributes to building a more sustainable and localized food system. This project exemplifies how thoughtful application design and the integration of advanced capabilities can create a tool that delivers tangible benefits, strengthening local economies and promoting healthier food choices.

5.2 Future Work

Building upon the foundational CropConnect platform, the future development roadmap will prioritize several key areas to enhance its functionality, expand its accessibility, and ensure its long-term sustainability and impact. A critical focus will be on increasing the platform's realism and responsiveness to real-world agricultural dynamics and user needs. This includes refining the accuracy of AI-powered features, such as price prediction, by incorporating more diverse and real-time market data. Furthermore, enhancing the platform's capacity to handle logistical complexities, such as varying delivery options and coordinating pickups between farmers and customers, will be crucial for a truly seamless user experience. This evolution aims to make CropConnect an even more reliable and indispensable tool for both farmers and consumers.

Expanding the reach and usability of CropConnect to a broader audience is another primary objective for future work. This involves a significant effort towards integrating support for a wider array of regional languages, going beyond the initial set to ensure that farmers and customers from diverse linguistic backgrounds can fully utilize the platform. Concurrently, substantial improvements will be made to the voice assistant feature, focusing on enhancing its speech recognition accuracy for various accents and languages, and expanding its capabilities to handle more complex product uploads and inquiries. A major step in this direction will be the development of a full-fledged, native mobile application for both Android and iOS platforms. This will provide users with a more convenient and optimized experience, enabling access to CropConnect anytime, anywhere, even in areas with limited internet connectivity by potentially incorporating offline capabilities for certain features.

Beyond these core enhancements, future work will explore additional features and integrations to further solidify CropConnect's position as a leading agricultural marketplace. This could include developing a rating and review system to build trust and transparency within the community, implementing loyalty programs to encourage repeat business, and exploring partnerships with agricultural cooperatives or local government initiatives. Integrating with external data sources for pest and disease advisories, or even soil health information, could provide farmers with even more comprehensive support. Ultimately, the future of CropConnect lies in its continuous evolution, driven by user feedback and technological advancements, to become an even more comprehensive, accessible, and impactful platform for the agricultural sector.

BIBLIOGRAPHY

[1] N. G, A. G, S. K. M, H. V, and B. L. (2024). *Shetigram - Building a Digital Bridge Between Farmers and Consumers*. In **Proceedings of the 2024 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES)**, Chennai, India, pp. 1–7. doi: https://ieeexplore.ieee.org/document/10910863

[2] R. B. Lincy, D. MD, R. MK, M. S, and B. G. (2023). *AGROCART – An Online Platform for Farmers to Sell Products without Middleman*. In **Proceedings of the 2023 Fourth International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE)**, Bengaluru, India, pp. 1–8. doi: https://ieeexplore.ieee.org/document/10585208

[3] S. G, N. S, R. A, and A. N. (2024). *Enhancing Crop Yield Prediction and Provide Direct Market Access to Farmers Using Web Technologies and Machine Learning Models*. In **Proceedings of the 2024 Eighth International Conference on Parallel, Distributed and Grid Computing (PDGC)**, Solan, India, pp. 842–847. doi: https://ieeexplore.ieee.org/document/10984269

[4] S. Bhukta, A. Shil, A. Dutta, and B. Sadhukhan. (2024). *KisanConnect: Reshaping Agricultural Trade with Dynamic Pricing Model*. In **Proceedings of the 2024 2nd International Conference on Networking, Embedded and Wireless Systems (ICNEWS)**, Bangalore, India, pp. 1–7. doi: https://ieeexplore.ieee.org/document/10730822

[5] G. Manikandan, N. S. B, S. S. S. S. T, N. J. B, and S. R. M. (2025). *Design and Development of a Mobile Application for Direct Agricultural Market Access*. In **Proceedings of the 2025 International Conference on Visual Analytics and Data Visualization (ICVADV)**, Tirunelveli, India, pp. 836–845. doi: https://ieeexplore.ieee.org/document/10961104

APPENDIX

"C:\Btech\IOMP Mini Project\CropConnect\src\app\customer\cart\page.tsx"

```
"use client";

import { Button } from "@/components/ui/button";
import { Card, CardContent, CardDescription, CardFooter, CardHeader, CardTitle } from
"@/components/ui/card";
import { Input } from "@/components/ui/input";
import { ArrowLeft, ShoppingCart, CreditCard, Trash2, Minus, Plus } from "lucide-react";
import Link from "next/link";
import { useLanguage } from "@/contexts/LanguageContext";
import { useCart } from "@/contexts/CartContext";
import Image from "next/image"; // Use Next Image
import { useToast } from "@/hooks/use-toast";

export default function CartPage() {
  const { translate } = useLanguage();
  const { cart, removeFromCart, updateCartItemQuantity, getCartTotal, loadingCart } = useCart();
  const { toast } = useToast();

  const handleQuantityChange = (productId: string, currentCartQuantity: number, productStock:
number, change: number) => {
    const newQuantity = currentCartQuantity + change;
    if (newQuantity < 1) {
      // Optionally confirm removal or just do nothing/set to 1
      // For now, let's ensure it doesn't go below 1 via this button. Removal is separate.
      updateCartItemQuantity(productId, 1);
      return;
    }
    if (newQuantity > productStock) {
      toast({
        title: translate('notEnoughStockTitle', "Not Enough Stock"),
        description: translate('notEnoughStockDesc', `Maximum available quantity is ${productStock}.`),
        variant: "destructive",
      });
      updateCartItemQuantity(productId, productStock); // Set to max available
      return;
    }
  updateCartItemQuantity(productId, newQuantity);
};

const handleDirectQuantityInput = (productId: string, productStock: number, inputValue: string) => {
  const newQuantity = parseInt(inputValue, 10);
  if (isNaN(newQuantity) || newQuantity < 1) {
    updateCartItemQuantity(productId, 1); // Default to 1 if invalid input
    return;
  }
  if (newQuantity > productStock) {
    toast({
      title: translate('notEnoughStockTitle', "Not Enough Stock"),
      description: translate('notEnoughStockDesc', `Maximum available quantity is ${productStock}.`),
      variant: "destructive",
    });
    updateCartItemQuantity(productId, productStock);
    return;
  }
  updateCartItemQuantity(productId, newQuantity);
};

const subtotal = getCartTotal();
const taxes = subtotal * 0.05; // 5% tax (example)
const total = subtotal + taxes;

if (loadingCart) {
  return <div className="text-center py-10">{translate('loadingCart', 'Loading your cart...')}</div>;
}

return (
  <div className="max-w-3xl mx-auto space-y-8">
     <Button variant="outline" size="sm" asChild className="mb-6">
      <Link href="/customer/dashboard">
        <ArrowLeft className="mr-2 h-4 w-4" /> {translate('continueShopping', 'Continue Shopping')}
      </Link>
    </Button>

    <Card className="shadow-lg">
      <CardHeader>
        <CardTitle className="text-2xl flex items-center">
          <ShoppingCart className="mr-3 h-6 w-6 text-primary" />
```

```
            {translate('yourCart', 'Your Shopping Cart')}
          </CardTitle>
          <CardDescription>{translate('reviewItemsCheckout', 'Review your items and proceed to
checkout.')}</CardDescription>
        </CardHeader>
        <CardContent>
          {cart.length === 0 ? (
            <p className="text-muted-foreground text-center py-8">{translate('cartEmpty', 'Your cart is
empty.')}</p>
          ) : (
            <div className="space-y-6">
              {cart.map(item => (
                <div key={item.id} className="flex flex-col sm:flex-row items-start sm:items-center gap-4
border-b pb-4">
                  <div className="relative w-24 h-24 sm:w-20 sm:h-20 rounded-md overflow-hidden
shrink-0">
                    <Image
                      src={item.imageUrl || "https://placehold.co/100x100.png"}
                      alt={item.name}
                      layout="fill"
                      objectFit="cover"
                      data-ai-hint={`${item.category} produce`}
                    />
                  </div>
                  <div className="flex-grow">
                    <h3 className="font-semibold">{item.name}</h3>
                    <p className="text-sm text-muted-foreground">{translate('soldBy', 'Sold by:')}
{item.farmerName}</p>
                    <p className="text-sm text-muted-foreground">${item.price.toFixed(2)} /
{item.unit}</p>
                    <p className="text-xs text-muted-foreground">{translate('stockAvailable', 'Stock:')}
{item.quantity} {item.unit}</p>
                  </div>
                  <div className="flex flex-col items-end space-y-2 shrink-0">
                    <div className="flex items-center space-x-2">
                      <Button variant="outline" size="icon" className="h-8 w-8" onClick={() =>
handleQuantityChange(item.id, item.cartQuantity, item.quantity, -1)} disabled={item.cartQuantity <=
1}>
                        <Minus className="h-3 w-3"/>
                      </Button>
                      <Input
                        type="number"
                        value={item.cartQuantity}
                        onChange={(e) => handleDirectQuantityInput(item.id, item.quantity, e.target.value)}
                        className="w-16 h-8 text-center"
                        min="1"
                        max={item.quantity}
                      />
                      <Button variant="outline" size="icon" className="h-8 w-8" onClick={() =>
handleQuantityChange(item.id, item.cartQuantity, item.quantity, 1)} disabled={item.cartQuantity >=
item.quantity}>
                        <Plus className="h-3 w-3"/>
                      </Button>
                    </div>
                    <p className="font-semibold text-lg">${(item.price *
item.cartQuantity).toFixed(2)}</p>
                    <Button variant="link" size="sm" className="text-destructive p-0 h-auto
hover:underline" onClick={() => removeFromCart(item.id)}>
                      <Trash2 className="mr-1 h-4 w-4"/>{translate('remove', 'Remove')}
                    </Button>
                  </div>
                </div>
              ))}

              <div className="space-y-2 pt-4 border-t">
                <div className="flex justify-between">
                  <span>{translate('subtotal', 'Subtotal')}</span>
                  <span>${subtotal.toFixed(2)}</span>
                </div>
                <div className="flex justify-between text-muted-foreground">
                  <span>{translate('taxes', 'Taxes (5%)')}</span> {/* Example tax */}
                  <span>${taxes.toFixed(2)}</span>
                </div>
                <div className="flex justify-between font-bold text-lg pt-2 border-t">
                  <span>{translate('total', 'Total')}</span>
                  <span>${total.toFixed(2)}</span>
                </div>
              </div>
            </div>
          )}
        </CardContent>
        {cart.length > 0 && (
          <CardFooter>
            <Button size="lg" className="w-full" asChild>
              <Link href="/customer/checkout">
```

```
              <CreditCard className="mr-2 h-5 w-5" /> {translate('proceedToCheckout', 'Proceed to
Checkout')}
            </Link>
          </Button>
        </CardFooter>
      )}
    </Card>
  </div>
 );
}
```

"C:\Btech\IOMP Mini Project\CropConnect\src\app\farmer\dashboard\add-product\page.tsx"

```
"use client";

import { useState } from "react";
import { Button } from "@/components/ui/button";
import { Tabs, TabsContent, TabsList, TabsTrigger } from "@/components/ui/tabs";
import { AddProductForm } from "@/components/farmer/AddProductForm";
import { VoiceUpload } from "@/components/farmer/VoiceUpload";
import { useLanguage } from "@/contexts/LanguageContext";
import type { VoiceUploadResult, Product } from "@/lib/types";
import { ArrowLeft, Mic, FileText } from "lucide-react";
import Link from "next/link";

export default function AddProductPage() {
  const [initialFormData, setInitialFormData] = useState<Partial<Product> | null>(null);
  const [activeTab, setActiveTab] = useState("form");
  const { translate } = useLanguage();

  const handleVoiceDataProcessed = (data: VoiceUploadResult) => {
    setInitialFormData({
      name: data.productName,
      price: data.price,
      unit: data.unit,
      location: data.location,
      // Quantity and category will need to be filled by farmer
    });
    setActiveTab("form"); // Switch to form tab to complete details
  };
  return (
    <div className="max-w-2xl mx-auto space-y-6">
      <Link href="/farmer/dashboard" className="inline-flex items-center text-sm text-primary
hover:underline mb-4">
        <ArrowLeft className="mr-2 h-4 w-4" />
        {translate('backToDashboard', 'Back to Dashboard')}
      </Link>
      <h1 className="text-3xl font-bold tracking-tight">{translate('uploadProduct', 'Upload New
Product')}</h1>

      <Tabs value={activeTab} onValueChange={setActiveTab} className="w-full">
        <TabsList className="grid w-full grid-cols-2">
          <TabsTrigger value="form">
            <FileText className="mr-2 h-4 w-4" />
            {translate('addByForm', 'Add by Form')}
          </TabsTrigger>
          <TabsTrigger value="voice">
            <Mic className="mr-2 h-4 w-4" />
            {translate('addByVoice', 'Add by Voice')}
          </TabsTrigger>
        </TabsList>
        <TabsContent value="form" className="mt-6">
          <AddProductForm initialData={initialFormData} onClearInitialData={() =>
setInitialFormData(null)} />
        </TabsContent>
        <TabsContent value="voice" className="mt-6">
          <VoiceUpload onDataProcessed={handleVoiceDataProcessed} />
        </TabsContent>
      </Tabs>
    </div>
  );
}
```

"C:\Btech\IOMP Mini Project\CropConnect\src\app\farmer\layout.tsx"

```
"use client";
// Removed ProductProvider as it's now in RootLayout
import React from "react";

export default function FarmerLayout({
  children,
}: {
  children: React.ReactNode;
}) {
  // Add farmer-specific navigation or sidebar here if needed in the future
  return <>{children}</>;
}
```

"C:\Btech\IOMP Mini Project\CropConnect\src\app\login\[role]\page.tsx"

```
"use client";

import { useParams, useRouter } from "next/navigation";
import { useAuth } from "@/contexts/AuthContext";
import type { UserRole } from "@/lib/types";
import { Card, CardContent, CardDescription, CardHeader, CardTitle } from "@/components/ui/card";
import { LoginForm } from "@/components/auth/LoginForm";
import { useEffect } from "react";
import Link from "next/link";
import { useLanguage } from "@/contexts/LanguageContext";
import { APP_NAME } from "@/lib/constants";
import { Button } from "@/components/ui/button";

export default function LoginPage() {
  const params = useParams();
  const role = params.role as UserRole;
  const { isAuthenticated, user, setSelectedRole } = useAuth();
  const router = useRouter();
  const { translate } = useLanguage();

  useEffect(() => {
    if (role !== 'farmer' && role !== 'customer') {
      router.replace('/'); // Redirect if role is invalid
    } else {
      setSelectedRole(role);
    }
  }, [role, router, setSelectedRole]);

  useEffect(() => {
    if (isAuthenticated && user) {
      if (user.role === 'farmer') {
        router.replace('/farmer/dashboard');
      } else if (user.role === 'customer') {
        router.replace('/customer/dashboard');
      }
    }
  }, [isAuthenticated, user, router]);

  if (role !== 'farmer' && role !== 'customer') {
    return <div className="flex justify-center items-center min-h-screen"><p>Invalid role.</p></div>;
  }

  return (
    <div className="flex items-center justify-center min-h-[calc(100vh-15rem)] py-12">
      <Card className="w-full max-w-md shadow-lg">
        <CardHeader className="text-center">
          <CardTitle className="text-2xl">
            {translate('login', 'Login')} {translate('asA', 'as a')} {role === 'farmer' ? translate('farmer',
'Farmer') : translate('customer', 'Customer')}
          </CardTitle>
          <CardDescription>
            {translate('loginToAccess', `Login to access your ${APP_NAME} account.`)}
          </CardDescription>
        </CardHeader>
        <CardContent>
          <LoginForm role={role} />
          <p className="mt-6 text-center text-sm text-muted-foreground">
            {translate('dontHaveAccount', "Don't have an account?")}{" "}
            <Button variant="link" asChild className="p-0">
              <Link href={`/register/${role}`}>{translate('registerHere', 'Register here')}</Link>
            </Button>
          </p>
           <p className="mt-2 text-center text-sm">
            <Button variant="link" asChild className="p-0">
              <Link href="/">{translate('backToRoleSelection', 'Back to role selection')}</Link>
            </Button>
          </p>
        </CardContent>
      </Card>
    </div>
  );
}
```

C:\Btech\IOMP Mini Project\CropConnect\src\app\register\[role]\page.tsx

```
"use client";

import { useParams, useRouter } from "next/navigation";
import type { UserRole } from "@/lib/types";
import { Card, CardContent, CardDescription, CardHeader, CardTitle } from "@/components/ui/card";
import { RegisterForm } from "@/components/auth/RegisterForm";
import { useEffect } from "react";
import Link from "next/link";
import { useLanguage } from "@/contexts/LanguageContext";
import { useAuth } from "@/contexts/AuthContext";
import { Button } from "@/components/ui/button";
import { APP_NAME } from "@/lib/constants";

export default function RegisterPage() {
  const params = useParams();
  const role = params.role as UserRole;
  const router = useRouter();
  const { translate } = useLanguage();
  const { setSelectedRole } = useAuth();

  useEffect(() => {
    if (role !== 'farmer' && role !== 'customer') {
      router.replace('/'); // Redirect if role is invalid
    } else {
      setSelectedRole(role);
    }
  }, [role, router, setSelectedRole]);

  if (role !== 'farmer' && role !== 'customer') {
    return <div className="flex justify-center items-center min-h-screen"><p>Invalid role.</p></div>;
  }

  return (
    <div className="flex items-center justify-center min-h-[calc(100vh-15rem)] py-12">
      <Card className="w-full max-w-md shadow-lg">
        <CardHeader className="text-center">
          <CardTitle className="text-2xl">
            {translate('register', 'Register')} {translate('asA', 'as a')} {role === 'farmer' ? translate('farmer',
'Farmer') : translate('customer', 'Customer')}
          </CardTitle>
          <CardDescription>
            {translate('createCropConnectAccount', `Create your ${APP_NAME} account.`)}
          </CardDescription>
        </CardHeader>
        <CardContent>
          <RegisterForm role={role} />
          <p className="mt-6 text-center text-sm text-muted-foreground">
            {translate('alreadyHaveAccount', 'Already have an account?')}{" "}
            <Button variant="link" asChild className="p-0">
              <Link href={`/login/${role}`}>{translate('loginHere', 'Login here')}</Link>
            </Button>
          </p>
          <p className="mt-2 text-center text-sm">
            <Button variant="link" asChild className="p-0">
              <Link href="/">{translate('backToRoleSelection', 'Back to role selection')}</Link>
            </Button>
          </p>
        </CardContent>
      </Card>
    </div>
  );
}
```