B.M.S. COLLEGE OF ENGINEERING BENGALURU

Autonomous Institute, Affiliated to VTU

OOMD Mini Project Report

Digital Market Access Platform for Farmers

Submitted in partial fulfillment for the award of degree of

Bachelor of Engineering
in
Computer Science and Engineering

Submitted by:

PADMA DEEPAK
1BM23CS222

NITHYA S
1BM23CS219

PARASH KASHYAP
1BM23CS225

Department of Computer Science and Engineering
B.M.S. College of Engineering
Bull Temple Road, Basavanagudi, Bangalore 560 019
2025-26

B.M.S. COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We, PADMA DEEPAK (1BM2CS222), NITHYA(1BM2CS222),PARASH KASHYAP(1BM2CS222) students of 5th Semester, B.E, Department of Computer Science and Engineering, BMS College of Engineering, Bangalore, hereby declare that, this OOMD Mini Project entitled " **Digital Market Access Platform for Farmer**" has been carried out in Department of CSE, B.M.S. College of Engineering, Bangalore during the academic semester August 2025-December 2025. I also declare that to the best of our knowledge and belief, the OOMD mini Project report is not from part of any other report by any other students.

Signature of the Candidate

NTHYA(1BM2CS222)

PADMA(1BM23CS222)

PARASH KASHYAP(1BM23CS225)

B.M.S. COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the OOMD Mini Project titled "**DIGITAL MARKETS ACCESS PLATFORM**" has been carried out by ,PADMA DEEPAK(1BM2CS222),NTHYA(1BM2CS222),PARASH KASHYAP(1BM2CS222) during the academic year 2025-2026.

Signature of the Faculty in Charge (Your Guide Name)

Table of Contents

Chapter 1: Problem Statement

1. Problem Statement

Agriculture remains the primary source of livelihood for millions of farmers, yet a significant proportion of them continue to face major challenges in accessing fair and transparent markets. The existing agricultural marketing system is heavily dependent on intermediaries, which often leads to reduced profit margins for farmers. Due to limited access to real-time price information and market demand trends, farmers are frequently forced to sell their produce at unfavorable prices. This lack of pricing transparency restricts their ability to make informed decisions regarding when, where, and how to sell their crops.

Furthermore, farmers struggle with fragmented and unreliable market information. Current channels—such as local markets, middlemen, or manual announcements—do not provide accurate or timely updates on commodity prices, market fluctuations, or buyer requirements. As a result, farmers face uncertainty regarding the true value of their produce, leading to financial losses and reduced motivation to adopt modern agricultural practices.

In addition to market information challenges, farmers often lack efficient digital tools that can simplify the process of marketing, logistics coordination, and secure payments. Many farmers rely on manual communication methods that are slow, inconsistent, and susceptible to miscommunication. The absence of streamlined logistics support means farmers may face difficulties finding reliable transport, tracking deliveries, or ensuring that produce reaches buyers in good condition. This increases post-harvest losses and reduces overall profitability.
Another major issue is the lack of a secure and trustworthy payment mechanism. Farmers often depend on delayed cash payments or informal credit arrangements, which create risks of fraud, payment defaults, and prolonged financial uncertainty. Many small-scale farmers also have limited awareness of or access to digital payment systems, further widening the gap between them and modern agricultural markets.

Overall, the lack of an integrated, digital, and transparent marketplace prevents farmers from participating fully and fairly in the agricultural supply chain. To empower them and promote better market participation, there is a strong need for a unified digital platform that connects farmers directly with buyers, provides real-time market data, facilitates logistics, and ensures secure payment processes. Such a solution can significantly enhance pricing transparency, reduce dependency on intermediaries, increase efficiency, and ultimately improve the financial well-being of farmers.

Chapter 2: Software Requirement Specification

1. Problem Statement

Farmers face significant challenges in accessing markets due to limited pricing transparency, dependency on intermediaries, lack of real-time market data, and inadequate digital tools. These issues reduce profit margins and prevent informed decision-making. A comprehensive digital solution is required to connect farmers directly with buyers, provide accurate market data, streamline logistics, and ensure secure payments. The proposed system aims to overcome these barriers by digitizing the agricultural marketplace and enabling farmers to participate efficiently and profitably.

---

2. Introduction

2.1 Purpose

This SRS document outlines the functional, non-functional, design, and interface requirements for the *Market Access Platform for Farmers*. The system aims to empower farmers with a digital marketplace to enhance transparency, accessibility, and profitability.

2.2 Scope

The platform will:

- Allow farmers to list produce and set prices
- Allow buyers to search, filter, and order products
- Provide real-time market prices using government APIs
- Enable online payments and invoice generation
- Integrate transporters for logistics handling
- Support multiple regional languages

2.3 Overview

This SRS includes system structure, constraints, requirements, and planning details such as schedule and budget.

---

3. General Description

3.1 System Perspective

The platform integrates modules for:

- Product Listings

- Payments
- Logistics
- Communication
- Market Intelligence

3.2 User Types and Characteristics

- **Farmers:** Require simple UI, local language support
- **Buyers:** Need efficient product search and ordering tools
- **Transporters:** Manage delivery and logistics updates
- **Administrators:** Oversee verification, maintenance, and system operations

3.3 Product Features

- User profiles
- Product catalog
- Market price dashboard
- Direct messaging
- Order tracking
- Payment gateway integration

3.4 Operating Environment

- Android and iOS mobile applications
- Web browsers (Chrome, Firefox, Edge, Safari)

3.5 System Constraints

- Must comply with agricultural policies and data privacy laws
- Must work effectively in low-bandwidth rural environments

---

4. Functional Requirements

FR1: User registration via mobile/email
FR2: Login authentication via OTP/password
FR3: KYC verification for farmers
FR4: Product listing and editing
FR5: Real-time market price display
FR6: Product search and filtering
FR7: Order placement and tracking
FR8: Online payment support
FR9: Logistics request and status updates
FR10: Feedback and rating system

---

5. Interface Requirements

5.1 User Interface

- Clean, mobile-friendly layout
- Large icons and simple navigation
- Multi-language support

5.2 Hardware Interface

- Smartphones with 2GB RAM or higher
- Web-enabled laptops/desktops

5.3 Software Interface

- Payment Gateway APIs
- Market Price APIs
- Database Management System

5.4 Communication Interface

- HTTPS protocol
- SMS/Email alerts

---

6. Design Constraints

- Must follow PCI-DSS standards
- TLS 1.2+ encryption
- Support for low-end mobile devices
- Compliance with Data Privacy Laws (e.g., DPDP Act)

---

7. Non-Functional Requirements

7.1 Performance Requirements

- Support up to 20,000 concurrent users
- API response time: 1–2 seconds

7.2 Security Requirements

- End-to-end encryption
- Role-based access control
- Regular data backups

7.3 Reliability Requirements

- 99% system uptime
- Recovery time < 30 minutes

7.4 Usability Requirements

- Simple UI suitable for rural users
- Local language availability

7.5 Scalability Requirements

- Horizontal scalability for growing users

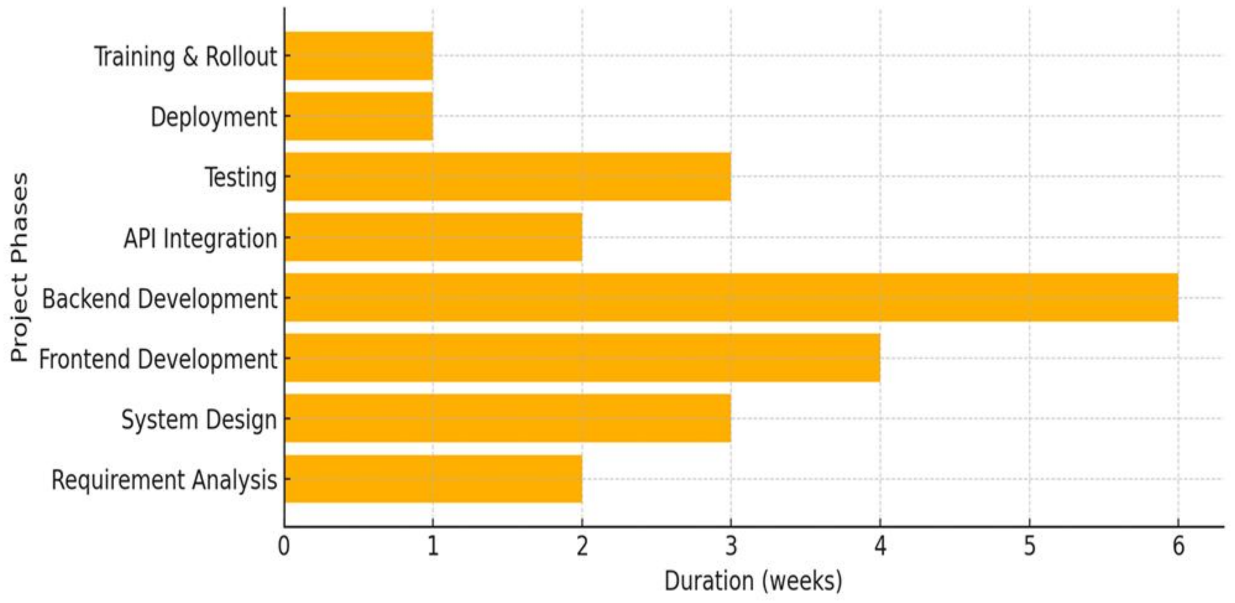
7.6 Maintainability Requirements

- Modular architecture
- Clear technical documentation

---

8. Preliminary Schedule and Budget

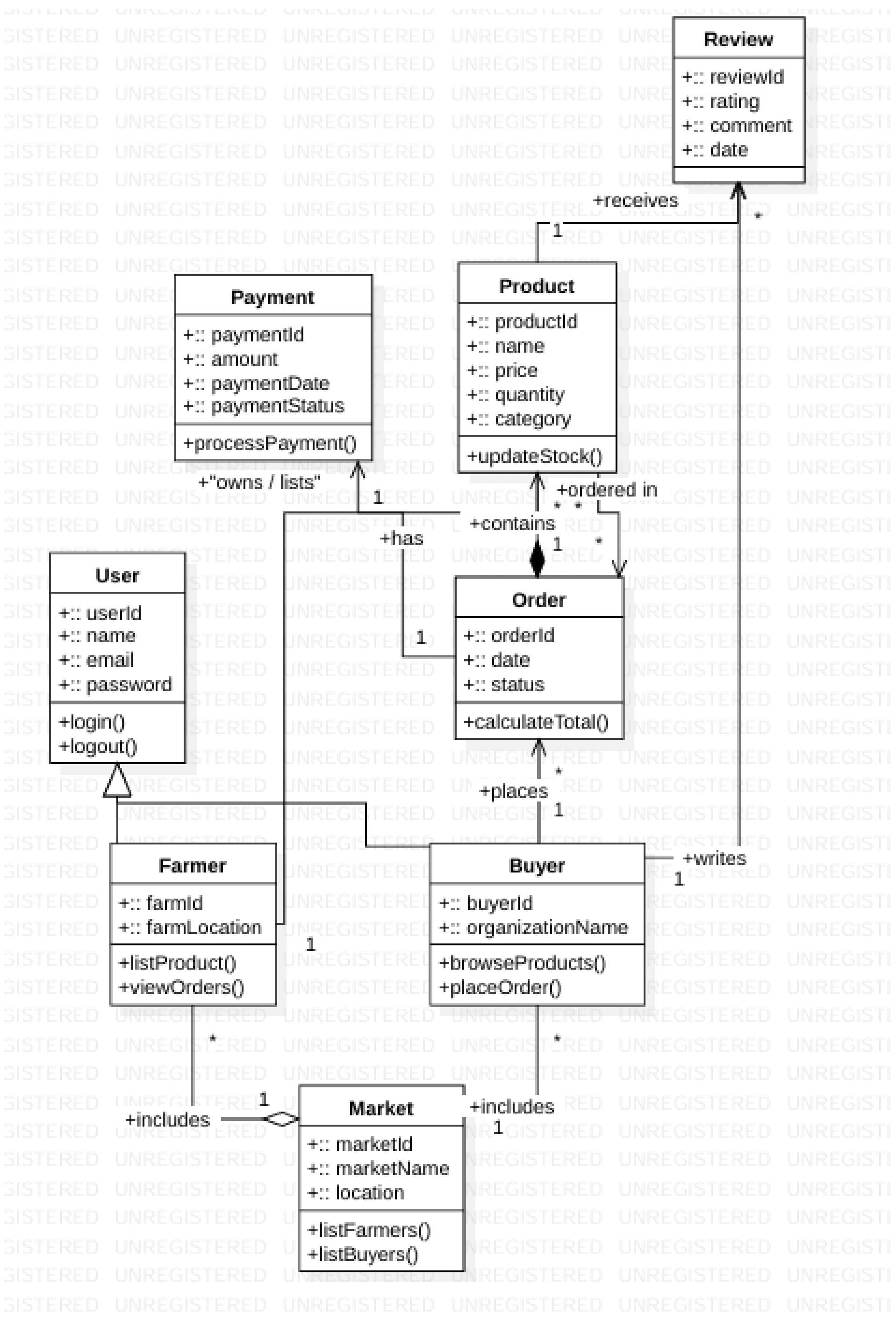
8.1 Development Schedule



8.2 Estimated Budget

Cost Component	Amount (INR)
Development Team	18,00,000
UI/UX Design	2,00,000
Cloud Hosting	1,50,000
API Integrations	50,000
Testing	1,00,000
Contingency	3,30,000
Total	**26,30,000**

Chapter 3: Class Modeling



Review
+:: reviewId
+:: rating
+:: comment
+:: date
+receives
1
*
Payment
+:: paymentId
+:: amount
+:: paymentDate
+:: paymentStatus
+processPayment()
Product
+:: productId
+:: name
+:: price
+:: quantity
+:: category
+updateStock()
+"owns / lists"
+ordered in
+contains
+has
User
+:: userId
+:: name
+:: email
+:: password
+login()
+logout()
Order
+:: orderId
+:: date
+:: status
+calculateTotal()
+places
Farmer
+:: farmId
+:: farmLocation
+listProduct()
+viewOrders()
Buyer
+:: buyerId
+:: organizationName
+browseProducts()
+placeOrder()
+writes
Market
+:: marketId
+:: marketName
+:: location
+listFarmers()
+listBuyers()
+includes
+includes

1. User

Purpose: Represents a general system user with basic authentication details.

Attributes:

- userId, name, email, password

Methods:

- login(), logout()

Relevance:

The **User** class is the base entity for both *Farmers* and *Buyers*. It supports login, logout, and general user identity management.

---

2. Farmer (inherits from User)

Purpose: Represents farmers who sell products.

Attributes:

- farmId, farmLocation

Methods:

- listProduct() – farmer adds products to the marketplace
- viewOrders() – view received orders

Relevance:

Farmers are the *sellers* on the platform. This class connects them to products and orders.

---

3. Buyer (inherits from User)

Purpose: Represents customers purchasing products.

Attributes:

- buyerId, organizationName

Methods:

- browseProducts()
- placeOrder()

Relevance:

Buyers are the *customers* of the marketplace. They browse available products and place orders.

4. Product

Purpose: Represents agricultural items listed by farmers.

Attributes:

- productId, name, price, quantity, category

Methods:

- updateStock()

Relationships:

- A farmer **lists** products
- An order **contains** multiple products
- A product **receives** reviews

Relevance:

The Product class is central to the marketplace. All transactions revolve around products.

---

5. Order

Purpose: Represents a purchase request placed by a buyer.

Attributes:

- orderId, date, status

Methods:

- calculateTotal()

Relationships:

- A buyer **places** orders
- An order **contains** products
- An order **has** a payment

Relevance:

The Order class handles the transaction workflow — collecting items, tracking status, and calculating total amount.

6. Payment

Purpose: Handles payment processing for an order.

Attributes:

- paymentId, amount, paymentDate, paymentStatus

Methods:

- processPayment()

Relationships:

- An order **has** one payment
- Payment is associated with a buyer/farmer through the order

Relevance:

Ensures that every order is backed by a secure and trackable payment record.

---

7. Review

Purpose: Allows buyers to give feedback on purchased products.

Attributes:

- reviewId, rating, comment, date

Relationships:

- A buyer **writes** reviews
- A product **receives** reviews

Relevance:

Reviews build trust, help other buyers decide, and allow farmers to improve offerings.

---

8. Market

Purpose: Represents different market locations within the system.

Attributes:

- marketId, marketName, location

Methods:

- listFarmers()
- listBuyers()

Relationships:

- A market **includes** many farmers and buyers

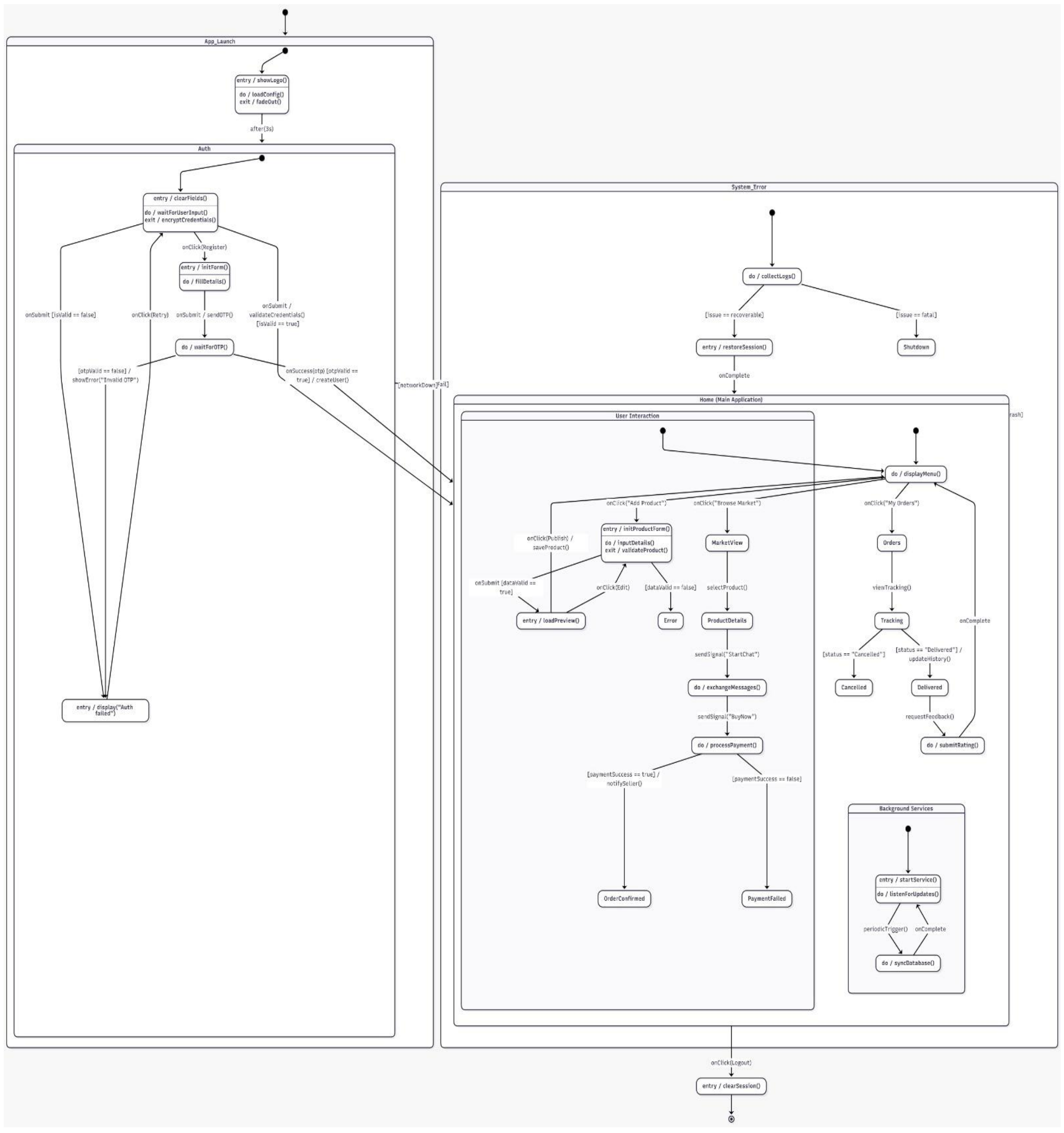
Relevance:

This class organizes the marketplace geographically, allowing grouping by area, region, or market hub.

How the System Works Together

- Farmers (subclass of User) **list products** in the market.
- Buyers (subclass of User) **browse and place orders**.
- Orders **contain products** and have an associated **payment**.
- Buyers can **review products** after purchase.
- All actors operate within one or more **Markets**, which group farmers and buyers by location.

Chapter 4: State Modeling



The state machine diagram illustrates the complete behavioral flow of the application from launch to shutdown, including system-level processes, user-initiated actions, and background operations. The diagram is organized into four primary regions: **App Launch**, **System Flow**, **User Interaction**, and **Background Synchronization**. Each section represents a distinct phase of application behavior, and the transitions between states depict how the system responds to events, inputs, and internal conditions.

---

1. App Launch Section

This section defines the initial behavior of the application immediately after it is opened.

1.1 AppLaunch

This is the entry point of the system where the application initializes.

- **Relevance:** Establishes the first point of interaction and triggers the initial loading sequence.
- **Event:** entry / loadSplashScreen() displays the splash screen while system resources begin loading.

1.2 Init (Initialization State)

This composite state manages all essential startup operations required before the application becomes interactive.

It contains the following substates:

a. CheckVersion

- Ensures the application version is valid and up-to-date.
- **Relevance:** Prevents deprecated versions from running and ensures compatibility.

b. VerifyUser

- Validates whether a user is already authenticated.
- **Relevance:** Determines whether to redirect to the login screen or directly to the home screen.

c. LoadResources

- Loads graphical assets, cached data, and configuration files.
- **Relevance:** Prepares the UI and internal resources for optimal performance.

d. InitDB

- Initializes or opens the local database.
- **Relevance:** Required for offline storage, caching, and transactional support.

1.3 Transitions

- If the application version is outdated, the transition leads to a mandatory update state.
- If authentication fails, the system navigates to the login screen.
- Upon successful initialization, the system transitions to the Home screen.

---

2. System Flow Section

This section covers system-level processes that execute automatically, independent of user actions.

2.1 CollectGPS()

- Retrieves the user's geographic location.
- **Relevance:** Enables location-based services such as nearby markets or localized pricing.

Transition Conditions:

- If GPS is enabled → proceed with geofencing.
- If permission is denied or GPS fails → transition to **ShutDown**.

2.2 InitializeServices

- Starts essential backend services including API handlers, notification engines, and resource synchronizers.
- **Relevance:** Ensures the core functional modules are active before user interaction.

2.3 ShutDown

- Represents the termination of the application due to permission denial, critical error, or user exit.
- **Relevance:** Formal end state for the system flow.

---

3. User Interaction Section

This is the largest region of the diagram, representing all user-driven navigation and actions within the application.

3.1 Home (Main Application Screen)

- Serves as the primary navigation hub.
- **Relevance:** Allows the user to access the main modules of the application.

3.2 MarketScreen

- Displays available markets, products, or farmer listings.

Events:

- onProductSelect() → transitions to **ProductDetails**
- onSearch() → transitions to **FilterScreen**
- onBackPressed() → returns to Home

3.3 ProductDetails

- Shows description, price, category, and stock information for a selected product.
- **Relevance:** Enables purchase decisions.

Events:

- onAddToCart(quantity > 0) → transitions to **ViewCart**
- Invalid input results in transition to the error state.

3.4 ViewCart

- Displays the list of selected products and allows modification of quantities.
- **Relevance:** Acts as a review step before payment.

Events:

- onProceedToPayment() → transitions to **ProcessPayment**

3.5 ProcessPayment

- Executes the payment workflow.
- **Relevance:** Facilitates transaction completion.

Events:

- onPaymentSuccess() → **PaymentStatus (Success)**
- onPaymentFailure() → **PaymentStatus (Failed)**

3.6 Orders

- Shows all current and past orders placed by the user.
- **Relevance:** Supports order tracking and history management.

Events:

- onOrderSelect() → **OrderDetails**

3.7 OrderDetails

- Displays the details of a specific order.
- **Relevance:** Provides information on items, status, and delivery.

Events:

- onTrackOrder() → **TrackOrder**
- onWriteReview() → **ReviewScreen**

3.8 TrackOrder

- Provides live tracking of the order delivery process.
- **Relevance:** Enhances transparency and user assurance.

3.9 ReviewScreen

- Allows a user to provide a rating and comments for a completed purchase.
- **Relevance:** Supports feedback collection and product quality evaluation.

---

4. Background Synchronization Section

This section handles automatic updates executed without user involvement.

4.1 BackgroundSync

- Controls periodic synchronization activities.
- **Relevance:** Maintains data consistency and keeps the system updated.

4.2 FetchMarketPrice()

- Retrieves updated government or market price information.
- **Relevance:** Ensures real-time pricing accuracy.

4.3 SyncDatabase()

- Pushes locally stored data to the server and retrieves updated information.
- **Relevance:** Enables offline functionality and data reliability.

---

5. Logout and Exit Section

5.1 UserLogout

- Triggered when the user logs out.

- **Relevance:** Ends the active session and returns to the login state.

5.2 Final State

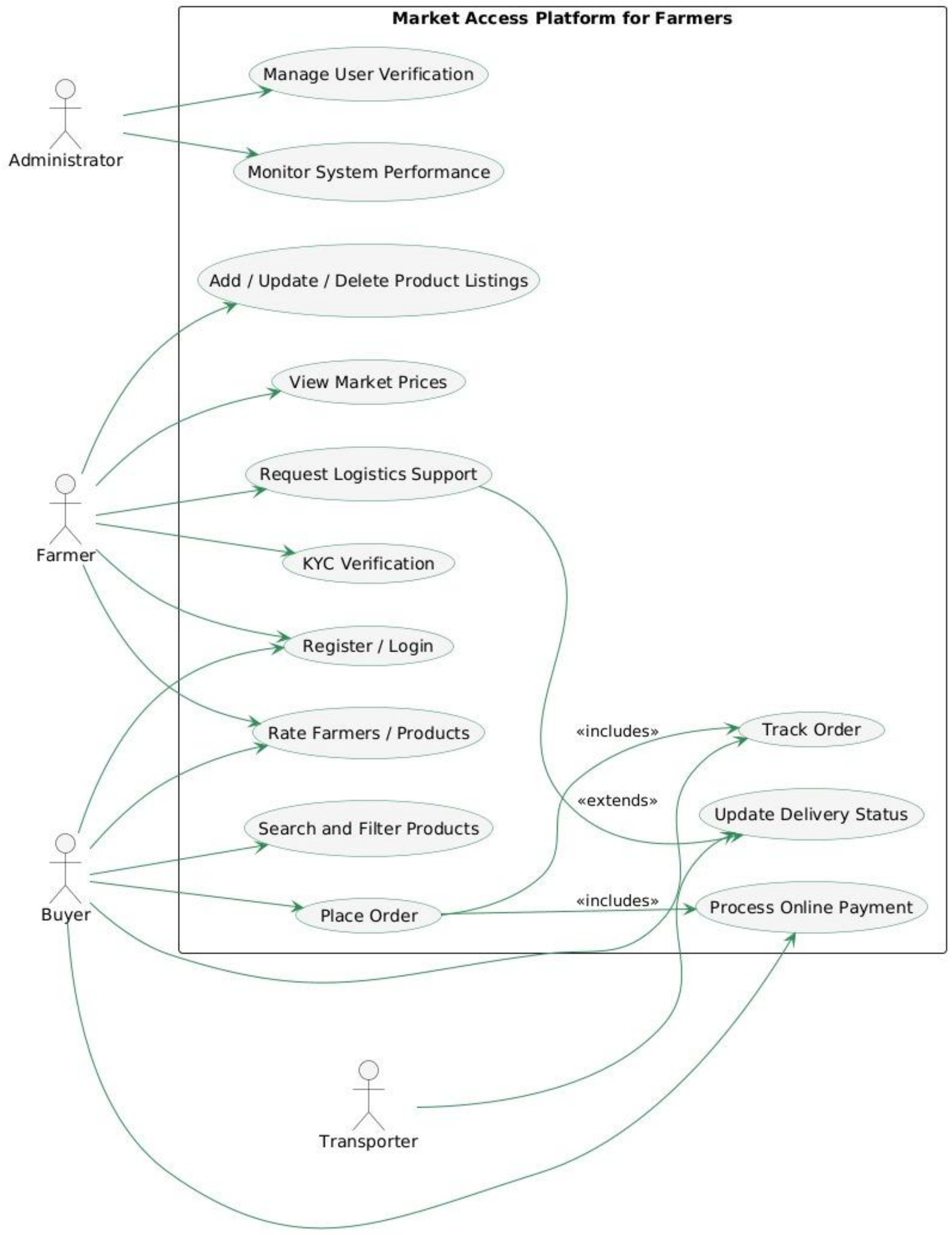
- Indicates full termination of application behavior.

✓ Conclusion

This state machine diagram provides a comprehensive representation of the application's dynamic behavior. It captures initialization routines, system-controlled processes, user-initiated interactions, transaction flows, and background synchronization. The diagram ensures clarity in understanding application behavior, supports software design decisions, and facilitates error-free system implementation.

Chapter 5: Interaction Modeling

USE CASE DIAGRAM



➢

The Use Case Diagram for the *Market Access Platform for Farmers* identifies the key actors interacting with the system and the major functionalities they access. Each actor plays a distinct role within the digital agricultural ecosystem, and each use case represents a critical function that supports the operational flow of the platform.

---

1. Actors and Their Relevance

1.1 Administrator

Relevance:

The Administrator is responsible for overseeing, managing, and maintaining the integrity of the platform. This actor ensures that all users (farmers and buyers) are verified, the system is operating efficiently, and data remains secure and accurate.

Key Responsibilities:

- User verification and approval
- Monitoring system performance
- Managing product listing rules and compliance
- Supporting policy enforcement

1.2 Farmer

Relevance:

The Farmer is a primary actor who uses the platform to sell agricultural produce directly to buyers. This actor benefits from transparency, digital tools, real-time market data, and reduced dependency on intermediaries.

Key Responsibilities:

- Listing and managing products
- Viewing market prices for informed decision-making
- Requesting logistics support for deliveries
- Completing KYC to validate identity
- Managing their own profile and transactions

1.3 Buyer

Relevance:

The Buyer represents individuals or organizations purchasing agricultural products from farmers through the platform. This actor drives the demand side of the marketplace and ensures a functioning digital trade environment.

Key Responsibilities:

- Registering and authenticating in the system
- Searching and filtering products
- Placing orders and making payments
- Rating farmers and products
- Tracking orders after purchase

1.4 Transporter

Relevance:

The Transporter is involved in order fulfillment by providing delivery and logistics services. Although not a frequent system user, this actor plays a crucial operational role in ensuring successful product delivery.

Key Responsibilities:

- Updating the status of deliveries
- Coordinating logistics with farmers and buyers
- Ensuring safe transport of goods

---

2. Use Cases and Their Relevance

2.1 Manage User Verification (Administrator)

Relevance:

Ensures only genuine farmers and buyers access the platform. Prevents fraud and maintains trust.

2.2 Monitor System Performance (Administrator)

Relevance:

Allows administrators to supervise system health, server usage, user activity, and potential issues.

2.3 Add / Update / Delete Product Listings (Farmer)

Relevance:

Enables farmers to manage product availability, ensuring buyers have access to accurate information and current stock levels.

2.4 View Market Prices (Farmer)

Relevance:

Provides real-time government or market price data. Helps farmers price their produce competitively and make informed selling decisions.

2.5 Request Logistics Support (Farmer)

Relevance:

Allows farmers to initiate delivery or transportation services for their sold products.

2.6 KYC Verification (Farmer)

Relevance:

Ensures farmer identity is authenticated before listing products, increasing buyer trust and platform credibility.

2.7 Register / Login (Farmer, Buyer)

Relevance:

Provides basic authentication access to the platform. Both farmers and buyers must register and log in before using the system.

2.8 Rate Farmers / Products (Buyer)

Relevance:

Enhances transparency and trust by allowing buyers to provide feedback on product quality and seller reliability.

2.9 Search and Filter Products (Buyer)
Relevance:
Allows buyers to efficiently explore available agricultural products based on criteria such as category, price, or location.

2.10 Place Order (Buyer)
Relevance:
Core purchasing functionality where buyers select items, confirm quantities, and initiate the order workflow.

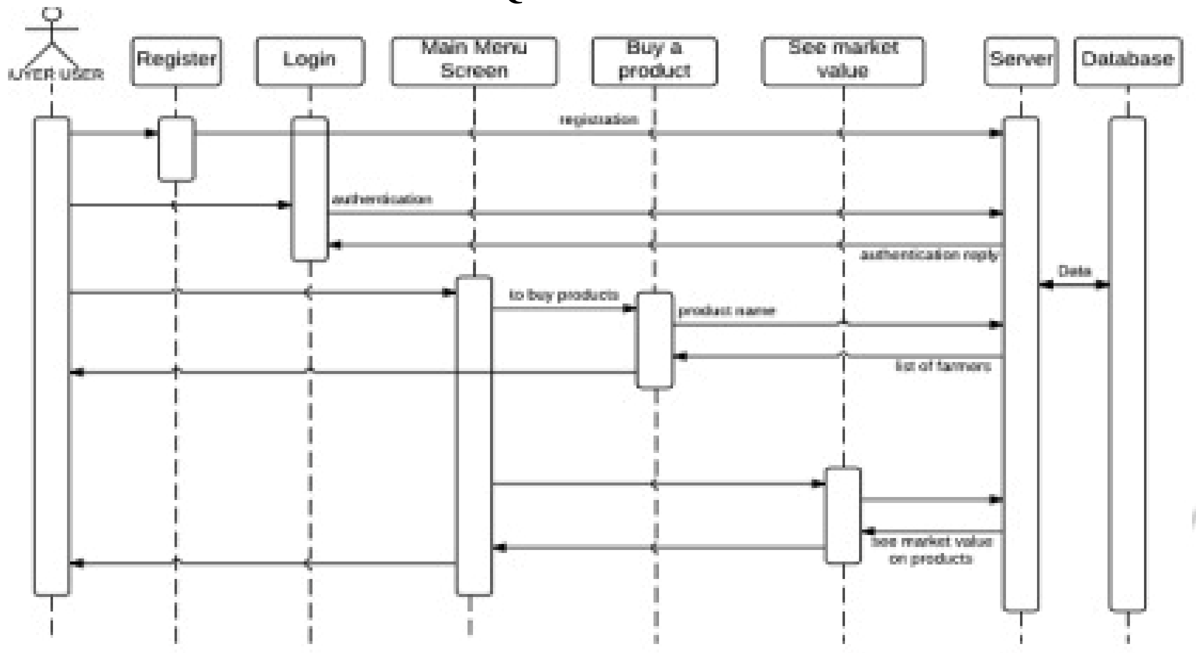
2.11 Process Online Payment (Buyer) — «includes»
Relevance:
A required step during order placement. Handles secure transaction completion through digital payment gateways.

2.12 Track Order (Buyer) — «includes»
Relevance:
Buyers can monitor delivery progress after placing an order. Ensures transparency and improves user experience.

2.13 Update Delivery Status (Transporter) — «extends»
Relevance:
Allows transporters to update shipment status, which extends the order tracking use case. Keeps buyers and farmers informed of real-time delivery updates.

Summary
This Use Case Diagram clearly defines the interaction between four primary actors—Administrator, Farmer, Buyer, and Transporter—and the key operations provided by the platform. The system ensures seamless agricultural trade by supporting essential functions such as product management, logistics, payments, and performance monitoring. Overall, these use cases work together to deliver a comprehensive digital marketplace solution that supports transparency, efficiency, and trust between agricultural producers and buyers.

SEQUENCE DIAGRAM



Sequence Diagram Explanation
The sequence diagram illustrates the step-by-step interaction between the **User**, **Application Interface Modules**, **Server**, and **Database** in the *Market Access Platform for Farmers*. It shows how a user registers, logs in, buys products, and views market prices, while the server and database coordinate in the background to provide the required data. The diagram presents the temporal order of operations and highlights how messages flow through the system.

---

1. Actor and Components
User
- The end-user interacting with the application (farmer or buyer).
- Initiates all primary actions in the sequence.

Register Module
- Manages new user registration requests.

Login Module
- Handles user authentication.

Main Menu Screen
- Provides navigation to different system functionalities after successful login.

Buy a Product Module
- Enables users to browse and purchase products.

See Market Value Module
- Retrieves and displays current market prices.

Server

- Acts as the backend application layer.
- Processes client requests, performs business logic, and communicates with the database.

Database

- Stores user credentials, product information, farmer lists, and market price data.

---

2. Sequence of Operations

2.1 User Registration

1. The user sends a **registration request** to the *Register* module.
2. The Register module forwards the data to the **Server**, which then stores it in the **Database**.
3. This establishes a new user record.

Relevance: Enables new users (farmers or buyers) to enter the platform.

2.2 User Login

1. The user enters credentials in the *Login* module.
2. The credentials are forwarded to the **Server** for authentication.
3. The server queries the **Database** and returns an **authentication reply**.
4. If successful, the user proceeds to the Main Menu Screen.

Relevance: Ensures secure, authenticated access to the system.

2.3 Navigation to Main Menu

- After login, the user is presented with the *Main Menu Screen*.
- From here, the user chooses between different actions such as buying products or viewing market prices.

Relevance: Acts as the central navigation hub of the platform.

2.4 Buying a Product

1. The user selects the **Buy a Product** option.
2. The system sends a request (e.g., "to buy products") to the **Server**.
3. The server queries the **Database** to retrieve the **list of farmers offering the selected product**.
4. The data is returned to the user.

Relevance: Facilitates purchasing by connecting buyers with available farmers and products.

2.5 Viewing Market Value

1. The user chooses the **See Market Value** option from the Main Menu.
2. A request for **market prices** is sent to the **Server**.
3. The server retrieves market price data from the **Database**.
4. Updated market value information is displayed to the user.

Relevance: Allows users (especially farmers) to access real-time price data for informed decision-making.

---

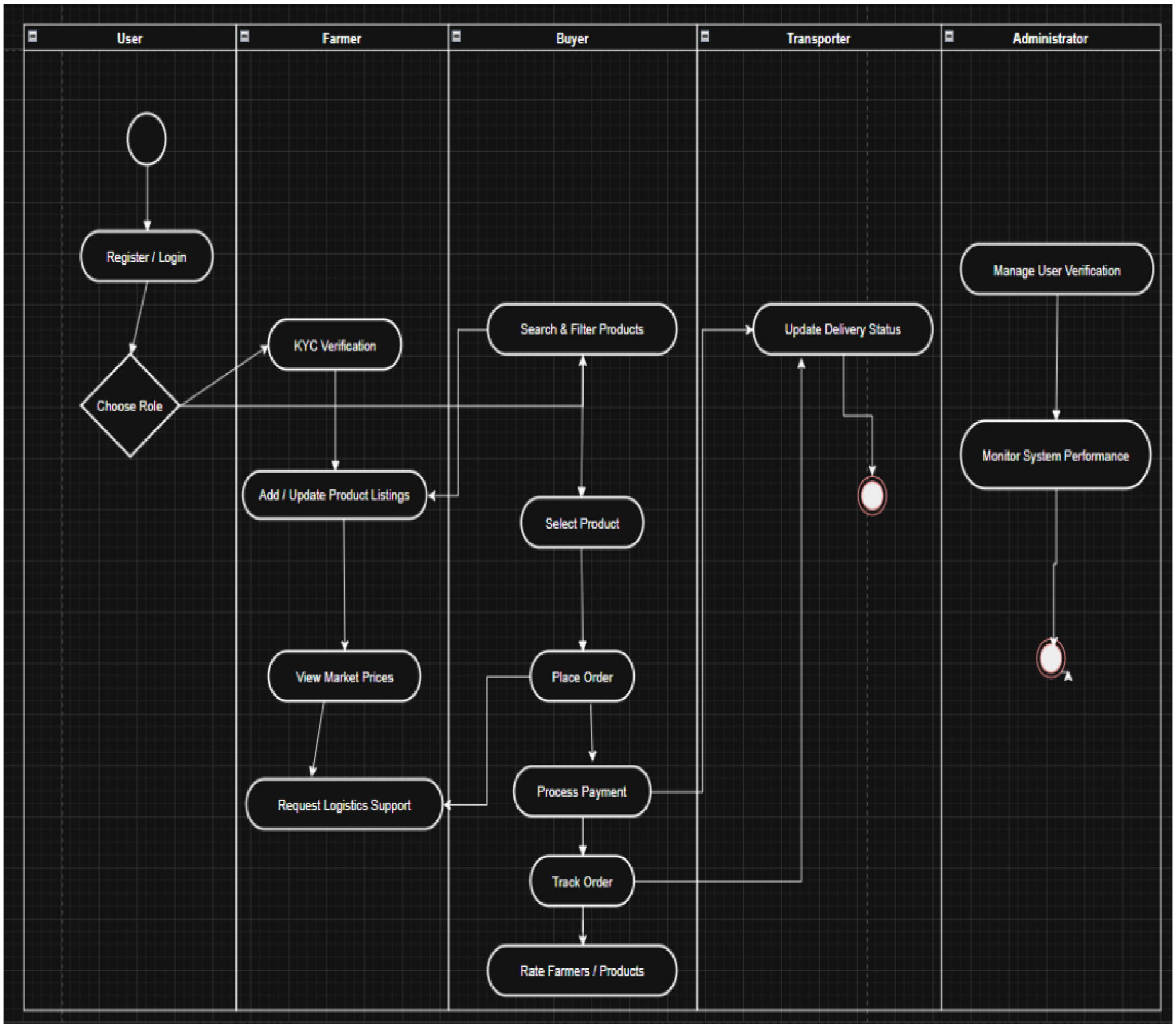
3. Summary of System Behavior

- Every user action travels through the application's UI modules to the **Server**, which acts as the central processing component.
- The **Database** plays a crucial role by storing and retrieving all relevant data (user details, product

listings, farmer lists, market prices).

- The system ensures smooth transitions between registration, authentication, purchase processes, and market data retrieval.
- The diagram clearly shows the linear and coordinated flow of information necessary for the functioning of the platform.

ACTIVITY DIAGRAM



The activity diagram illustrates the end-to-end workflow of the Market Access Platform for Farmers, organized across **five swimlanes** corresponding to different actors: **User, Farmer, Buyer, Transporter, and Administrator**. Each swimlane represents the responsibilities and actions performed by that actor, while the flows show how activities interact and depend on each other.

This diagram demonstrates how a user progresses from registration to role-based operations, including product management, purchasing, logistics handling, payments, and system administration.

Swimlane-Based Explanation

1. User Lane

This lane contains activities common to all system users before they assume their specific roles.

1.1 Register / Login

The process begins when an individual accesses the platform and registers or logs into their account.

1.2 Choose Role

After login, the user must select whether they are accessing the platform as a **Farmer** or **Buyer**.
This decision directs the workflow toward the corresponding swimlane.

---

2. Farmer Lane

This swimlane represents all activities performed by farmers who list products and manage sales.

2.1 KYC Verification

Farmers undergo identity verification (KYC) before using platform features.
This step ensures credibility and builds trust among buyers.

2.2 Add / Update Product Listings

After verification, farmers can add new products to the marketplace or update existing listings.
This includes setting price, quantity, and product details.

2.3 View Market Prices

Farmers can check real-time market prices, helping them make informed decisions about pricing strategy.

2.4 Request Logistics Support

Once an order is placed by a buyer, farmers may request logistics assistance for delivery coordination.
This action interacts with the Transporter lane.

---

3. Buyer Lane

The buyer interacts with the marketplace to discover, evaluate, and purchase products.

3.1 Search & Filter Products

Buyers browse the marketplace using filters such as price, category, location, or product name.

3.2 Select Product

The buyer selects a specific item based on interest and product details.

3.3 Place Order

The selected product is added to an order, and the buyer proceeds to confirm their purchase.

3.4 Process Payment

Once the order is confirmed, the buyer completes the payment through the platform's integrated payment system.

3.5 Track Order

After payment, buyers can track the progress of their order in real time.
Tracking updates are coordinated with the transporter.

3.6 Rate Farmers / Products

Upon successful delivery, buyers may provide feedback to improve platform transparency and quality.

---

4. Transporter Lane

This lane represents the logistics service provider responsible for fulfilling deliveries.

4.1 Update Delivery Status

Transporters update the progress of order delivery (e.g., dispatched, in transit, delivered).
This information feeds back to the buyer's **Track Order** activity.

---

5. Administrator Lane

The administrator oversees user credibility and system functionality.

5.1 Manage User Verification

Admins verify the identities of users (especially farmers) to maintain platform security.

5.2 Monitor System Performance

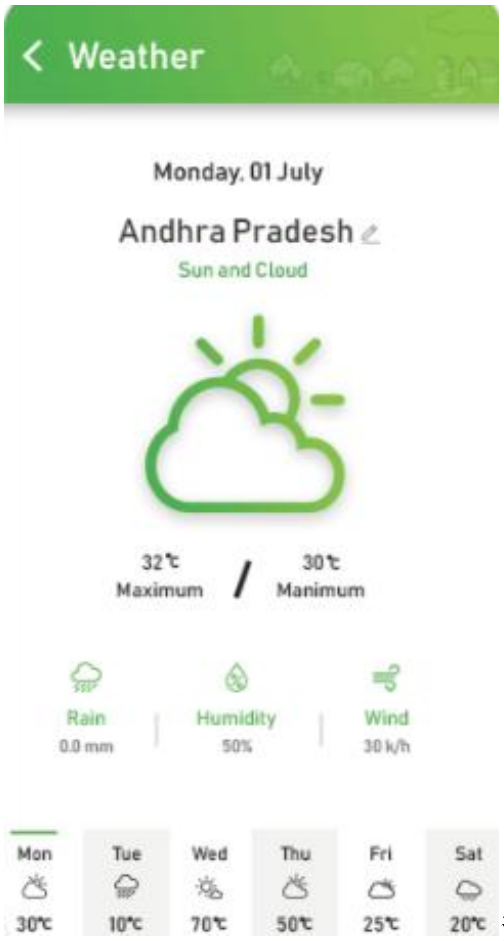
Admins also track the operational health of the system, ensuring stable performance and handling exceptions.

Overall Workflow Summary

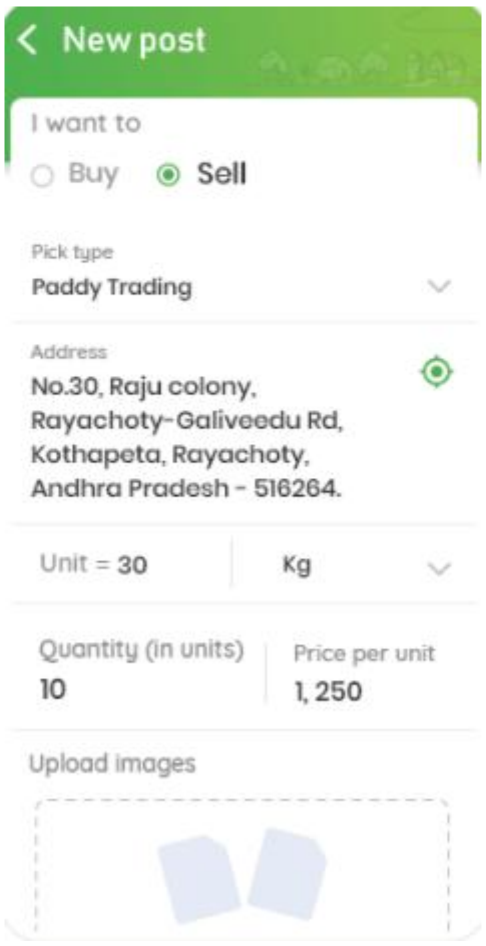
1. **User authentication and role selection** determine the navigation path.
2. **Farmers** manage product listings, check market prices, and handle logistics requests.
3. **Buyers** browse and purchase products, complete payments, track orders, and provide ratings.
4. **Transporters** update delivery status to keep buyers informed.
5. **Administrators** ensure the system remains secure, validated, and operational.

The activity diagram effectively models the **complete lifecycle of interactions** between the various stakeholders in the Market Access Platform and highlights how different roles synchronize to deliver a seamless agricultural marketplace experience.

Chapter 6: UI Design with Screenshots



j



New post

I want to

Buy [✓ Sell]

Pick type
Paddy Trading

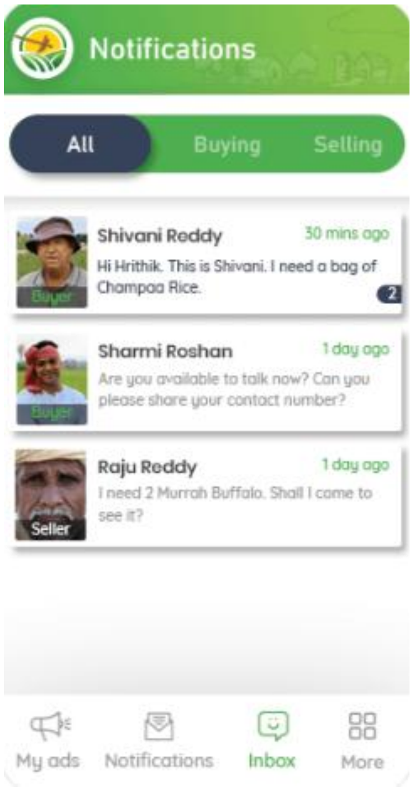
Address
No.30, Raju colony,
Rayachoty-Galiveedu Rd,
Kothapeta, Rayachoty,
Andhra Pradesh - 516264.

Unit = 30 | Kg

Quantity (in units)
10

Price per unit
1, 250

Upload images

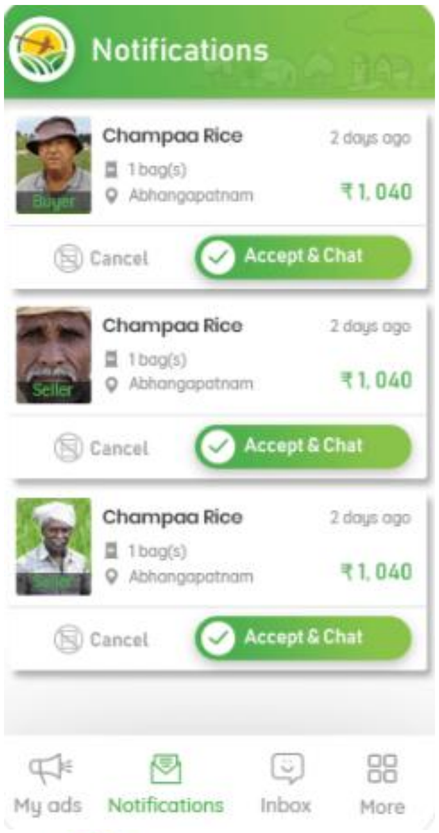

Notifications

All | Buying | Selling

Shivani Reddy — 30 mins ago — Buyer
Hi Hrithik. This is Shivani. I need a bag of Champaa Rice. 2

Sharmi Roshan — 1 day ago — Buyer
Are you available to talk now? Can you please share your contact number?

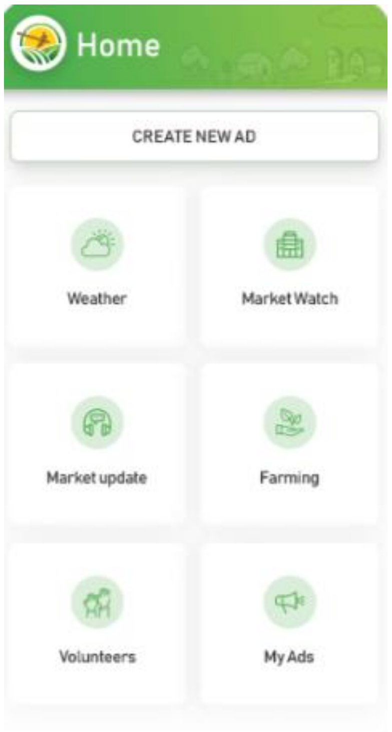
Raju Reddy — 1 day ago — Seller
I need 2 Murrah Buffalo. Shall I come to see it?

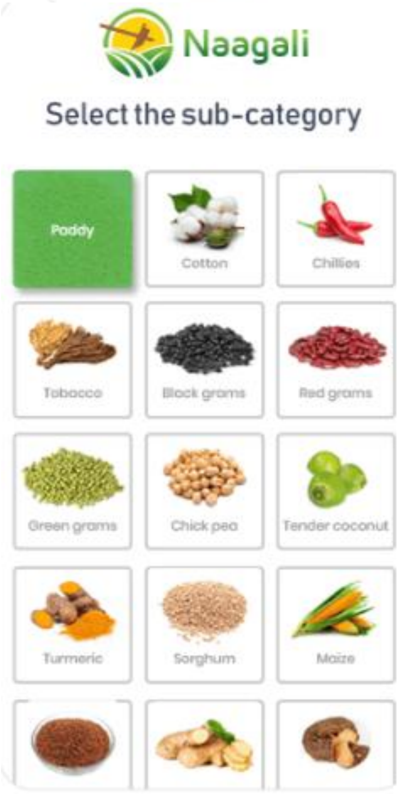

Notifications

Champaa Rice — 2 days ago — Buyer
1 bag(s)
Abhangapatnam
₹1, 040
Cancel | Accept & Chat

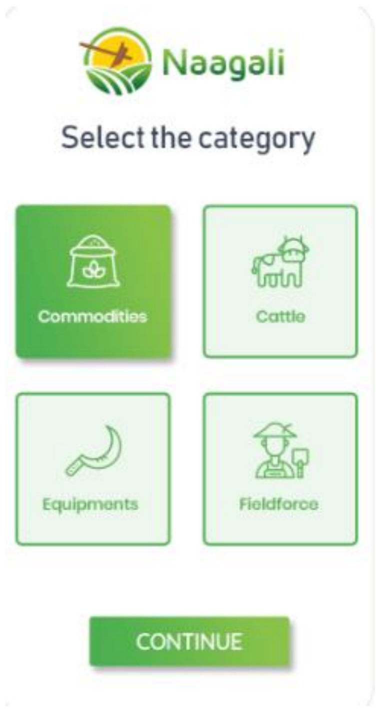
Champaa Rice — 2 days ago — Seller
1 bag(s)
Abhangapatnam
₹1, 040
Cancel | Accept & Chat

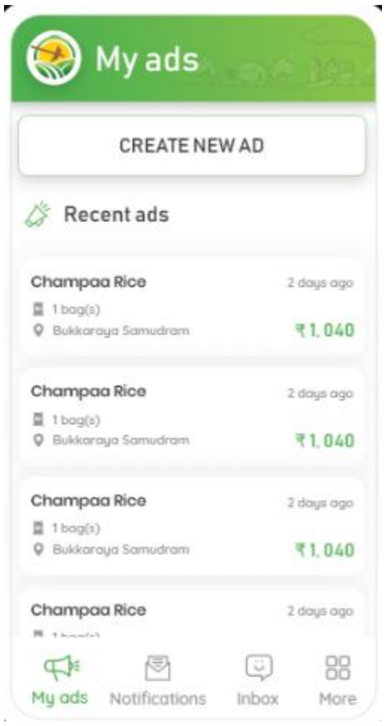
Champaa Rice — 2 days ago — Seller
1 bag(s)
Abhangapatnam
₹1, 040
Cancel | Accept & Chat


Home


Naagali

Select the sub-category





Champaa Rice 2 days ago
1 bag(s)
Bukkaraya Samudram ₹1,040

Champaa Rice 2 days ago
1 bag(s)
Bukkaraya Samudram ₹1,040

Champaa Rice 2 days ago
1 bag(s)
Bukkaraya Samudram ₹1,040

Champaa Rice 2 days ago