An-Najah National University

Faculty of Engineering & Information Technology

Presented in partial fulfillment of the requirements for

Bachelor degree in Computer Engineering

Graduation Project 1

FarmX

Students:

Tala Yaseen

Samaa Yasin

Supervisor:

Dr. Luai Malhis

malhis@najah.edu

June 28, 2025

Disclaimer

This report was written by students Tala Yaseen and Samaa Yasin at the Computer Engineering Department, Faculty of Engineering, An-Najah National University. It has not been altered or corrected, other than editorial corrections, as a result of assessment and it may contain language as well as content errors. The views expressed in it together with any outcomes and recommendations are solely those of the students. An-Najah National University accepts no responsibility or liability for the consequences of this report being used for a purpose other than the purpose for which it was commissioned.

Acknowledgment

No success is achieved without the support of those around us. This project became possible thanks to Allah and the incredible people who supported us along the way. We are very grateful to our project supervisor, Dr. Luai Malhis, for his guidance, support, and expertise throughout the project. His dedication kept us motivated and focused.

We also express our thanks to our friends, classmates, and families for their constant support, valuable feedback, and helpful suggestions. Their unwavering patience, encouragement, and understanding were essential in refining and improving the quality of our work. Without their support, we would not have been able to fully dedicate ourselves to the project and bring it to completion.

We sincerely acknowledge the effort, time, and support of everyone who contributed to making this project a success. We are sincerely grateful to all those who supported us throughout this journey.

Abstract

Abstract

Agriculture remains a vital sector in Palestine, supporting livelihoods and food security. However, farmers face numerous challenges, including inefficient traditional practices, limited technological access, and weak connections with consumers. **FarmX** is a smart, web- and mobile-based farm management system designed to address these issues. The platform provides a comprehensive ecosystem for farmers to manage their farms, track planted crops, and sell products directly to consumers through an integrated digital marketplace.

The system supports multiple user roles, including farmers, consumers, administrators, and order handlers. It features a 3D interactive homepage, real-time communication using Firebase, and personalized notifications. FarmX was developed using Spring Boot for backend services, React and Next.js for the web interface, and React Native for the mobile application. The platform integrates modern technologies to enhance agricultural productivity, foster community interaction, and support sustainable development. This project demonstrates how software engineering can be applied to empower local farmers and revolutionize agricultural systems in the region.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 General background

Agriculture has always been the cornerstone of human civilization, sustaining populations and enabling societal development. In recent years, the intersection between agriculture and technology has given rise to innovative smart farming solutions. These systems aim to enhance productivity, optimize resource usage, and provide data-driven insights for better decision-making.

In Palestine, agriculture is not only a major economic sector but also a cultural and historical pillar. However, farmers face numerous challenges ranging from climate variability, limited water resources, and inefficient traditional practices to a lack of access to modern technologies. These issues demand a robust solution that empowers farmers and promotes sustainable agricultural practices.

To address these challenges, we introduce FarmX — an intelligent farm management system that leverages modern web and mobile technologies and real-time communication tools to simplify the entire farming experience. FarmX enables farmers to manage their farms, track planted crops and monitor orders, while consumers can explore a unified marketplace, order products, and interact with farms. The system also includes admin and order handler roles to ensure full operational support.

1.2 Objectives

The primary objective of this project is to develop an intelligent, user-friendly, and scalable farm management system tailored for the local agricultural context in Palestine. FarmX is designed to serve multiple user roles including farmers, consumers, administrators, and order handlers through an integrated platform.

The specific objectives of the system include:

- Enabling farmers to manage their farms, add products, and track planted crops efficiently.
- Providing consumers with a unified marketplace to explore and purchase fresh agricultural products from different farms.
- Offering administrators full access to oversee all operations, manage users, and maintain system integrity.
- Assisting order handlers in managing and tracking delivery tasks efficiently.
- Supporting real-time features such as chat and social posts using Firebase services.
- Enhancing user experience with a 3D interactive homepage built using Three.js.
- Providing a mobile application using React Native for on-the-go access and usage.

1.3 Significance of the Work

FarmX addresses key agricultural challenges in Palestine by introducing a comprehensive digital solution that enhances productivity, promotes sustainability, and strengthens the connection between farmers and consumers.

This project stands out for its:

- Local relevance — designed with a deep understanding of the Palestinian agricultural ecosystem.
- Technological innovation — combining 3D web interfaces, Firebase-powered real-time services, and mobile support.
- Multi-role support — enabling structured collaboration and operational flow among all actors involved in the agricultural supply chain.
- Educational and societal value — serving as a model of how technology can empower rural communities and promote food security.

By offering a smart ecosystem tailored for agriculture, FarmX contributes to the digital transformation of a sector that is vital to both economic development and cultural identity in the region.

1.4 Organization of the Report

This report is divided into chapters as follows:

- **1. Introduction:** Discusses the work's background, objectives, and importance, followed by an explanation of how the report is organized.
- **2. Theoretical Background and Previous Work:** Presents a review of the theoretical foundations and pertinent earlier research.
- **3. Constraints and Earlier Coursework:** Discusses the limitations encountered during the project as well as the previous coursework that contributed to its development.
- **4. Methodology:** Describes the tools, technologies, programming languages, frameworks, and database design employed. It also includes a full explanation of the FarmX application's features for all user roles.
- **5. Discussion and Results:** Provides an analysis of the collected data and interpretations.
- **6. Conclusion:** Summarizes the outcomes of the project, highlights the main conclusions, and reflects on the project's impact.
- **7. Future Work:** Suggests potential improvements and further developments for the system.
- **8. Bibliography:** Covers all references and sources utilized in the study.

Chapter 2

Theoretical Background and Previous Work

2.1 Theoretical Background

2.1.1 Farm Management Systems

Farm management systems are designed to assist farmers in efficiently organizing and monitoring their agricultural operations. These systems include functionalities such as crop management, order tracking, and inventory control, which help optimize productivity and resource usage.

2.1.2 3D Visualization Technologies

3D visualization technologies, like Three.js, allow web applications to present interactive and immersive environments. Incorporating 3D models enhances user engagement by providing realistic representations of farms and products, which can improve decision-making processes.

2.1.3 Real-time Communication and Social Features

Real-time communication tools improve interaction among users, such as farmers, consumers, and administrators. Firebase is commonly used for implementing these features due to its real-time database capabilities and ease of integration, supporting chat, notifications, and social

posts.

2.1.4 Soil Data Integration

Accurate soil data is essential for effective farm management. Resources like SoilGrids provide comprehensive soil composition information, enabling systems to recommend suitable crops based on location-specific soil characteristics.

2.2 Previous Work

Various farm management platforms exist globally, offering digital solutions for agricultural operations. Examples include FarmLogs, AgriWebb, and Cropio, which provide features like field mapping, crop tracking, and task management. However, these platforms often lack localization for Palestinian agricultural needs and integration of modern features such as 3D visualization and real-time social communication.

Some recent projects have started incorporating 3D user interfaces and Firebase-powered real-time services to enhance user experience and collaboration. However, a unified platform combining multi-role management, AI-driven crop suggestions, immersive 3D homepage, and mobile application support remains scarce, highlighting the significance of the FarmX project.

Chapter 3

Constraints and Earlier course work

3.1 Constraints

During the development of FarmX, several constraints and challenges were encountered that affected the design, implementation, and deployment phases:

- **Limited Soil Data Coverage:** SoilGrids does not provide comprehensive soil data coverage for all regions in Palestine, requiring manual data collection and interpolation to fill in missing information.
- **Outdated Agricultural Data:** Existing agricultural statistics and data for Palestine are often outdated, limiting the accuracy of AI crop suggestions and decision-making tools.
- **Notification System Limitations:** Standard technologies such as WebSockets were unsuitable for private channels and reliable delivery of notifications across different client platforms, which led to redesigning the notification system using Firebase Cloud Messaging for better compatibility and scalability.
- **Resource Constraints in Deployment:** Cloud hosting and resource allocation faced limitations due to cost constraints and availability, which affected system scalability and response times.
- **Multi-role Access Control Complexity:** Managing different user roles (Admin, Farmer, Consumer, Order Handler) with varying permissions required careful design to ensure data privacy, security, and appropriate access throughout the system.

- **Integration Challenges:** Combining multiple technologies such as Spring Boot backend, React/Next.js frontend, React Native mobile app, Firebase services, and AI modules introduced complexity in integration and maintenance.
- **Limited Network Infrastructure:** Variability in internet connectivity and speed in rural areas of Palestine posed challenges for real-time features like chat and notifications.
- **3D Visualization Performance:** Implementing a lightweight and responsive 3D homepage using Three.js required optimization to ensure acceptable performance across devices with varying hardware capabilities.

3.2 Earlier Coursework

The development of this project was supported by prior academic and practical coursework, including:

- Practical training at the university, providing hands-on experience in software development and project management.
- Advanced web development courses, which contributed to mastering frontend and backend technologies used in this project.
- Advanced software engineering courses, enhancing understanding of software architecture, design patterns, and best practices.
- Distributed Operating Systems (DOS) course, which provided knowledge about system design for handling multiple users and concurrency, aiding in implementing the multi-role access control and real-time features.

Chapter 4

Methodology

4.1 Methodology

This section describes the methodologies, tools, and technologies used to design, develop, and deploy the FarmX system. The project is divided into multiple components including frontend, backend, database management and real-time communication.

The frontend of FarmX is developed using React and Next.js for the web application, complemented by a mobile app built with React Native and Expo. The 3D interactive homepage utilizes Three.js to provide an engaging user experience.

The backend is implemented using Spring Boot, which handles business logic, data processing, and role-based access control. Firebase services are integrated to support real-time chat, notifications, and social posts.

The database is managed using MySQL hosted on Railway, ensuring persistent storage of users, products, farms, orders, and feedback.

4.1.1 System Design

The FarmX system adopts a multi-tier architecture integrating various components including frontend applications (web and mobile), backend services, database management, and cloud-based real-time communication.

Figure 4.1 illustrates the overall system architecture and how different components interact to provide a seamless user experience.

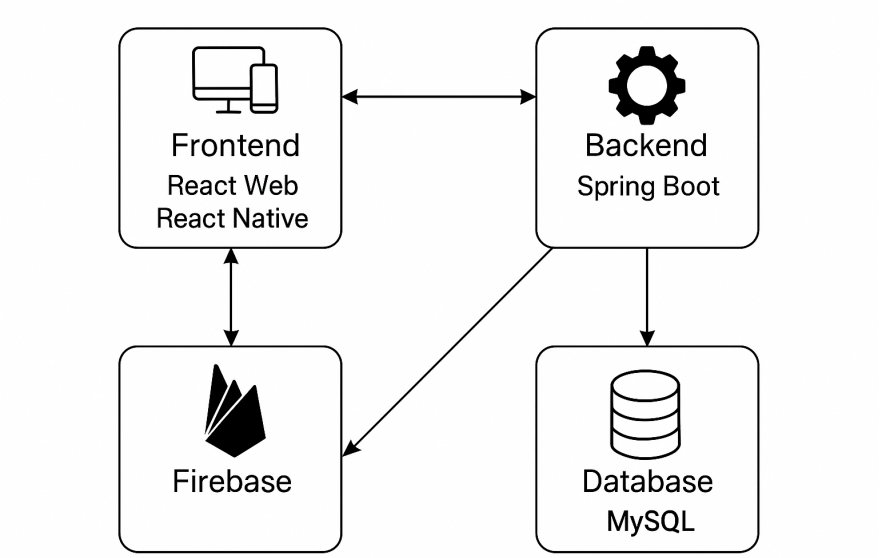

Figure 4.1: FarmX system architecture diagram showing the integration of frontend, backend, database, and Firebase services

4.1.2 Tools and Technologies Used

FarmX leverages a diverse set of modern tools and technologies to deliver a full-featured, scalable, and visually rich farm management platform. The following tools were used across different layers of the system:

- **Spring Boot:** The main backend framework used to implement business logic, RESTful APIs, and integrations with Firebase and the MySQL database.
- **Spring Security & JWT:** Ensures secure access through authentication and authorization using JSON Web Tokens and role-based access control.
- **Maven:** A project management and build automation tool used to manage dependencies and package the Spring Boot backend.
- **Swagger:** Used to document and visualize REST APIs, making it easier to test and integrate with the frontend.
- **Docker:** Used to containerize backend services for consistent deployment across environments.
- **React + Next.js (TypeScript):** Used to build the web application's frontend. TypeScript enhanced code quality and readability, while Next.js enabled server-side rendering and optimized routing.
- **React Three Fiber:** A powerful abstraction over Three.js, used to build the 3D homepage and interactive scenes within the web interface.
- **Chakra UI:** A component-based UI library used in the React frontend to create consis-

tent, responsive, and accessible user interfaces.

- **React Native with Expo:** Used for building the cross-platform mobile application with fast development cycles and native performance.
- **React Native Styles:** Utilized for styling the mobile UI in a flexible and modular manner.
- **Firebase:** Integrated for real-time features such as messaging, cloud notifications (via FCM), and cloud storage for user-generated content.
- **MySQL (hosted on Railway):** A relational database used to persist user information, farms, products, orders, feedback, and more.
- **Railway:** Cloud platform for deploying the Spring Boot backend and hosting the MySQL database. It supports seamless integration and auto-deployment pipelines.
- **Render:** Used to host the frontend application (React + Next.js) and manage production deployments with ease.
- **Postman:** A vital tool for testing and debugging backend APIs during development.
- **Git & GitHub:** Used for version control, collaboration, and continuous integration/deployment (CI/CD) workflows.
- **Canva:** Used to design diagrams, presentation slides, and promotional materials.

4.1.3 Database Design

The database schema for FarmX is designed using MySQL and hosted on Railway. The schema consists of multiple tables representing the core entities of the system including users, farms, products, orders, and more. Figure 4.2 shows the Entity-Relationship Diagram illustrating the tables and their relationships.

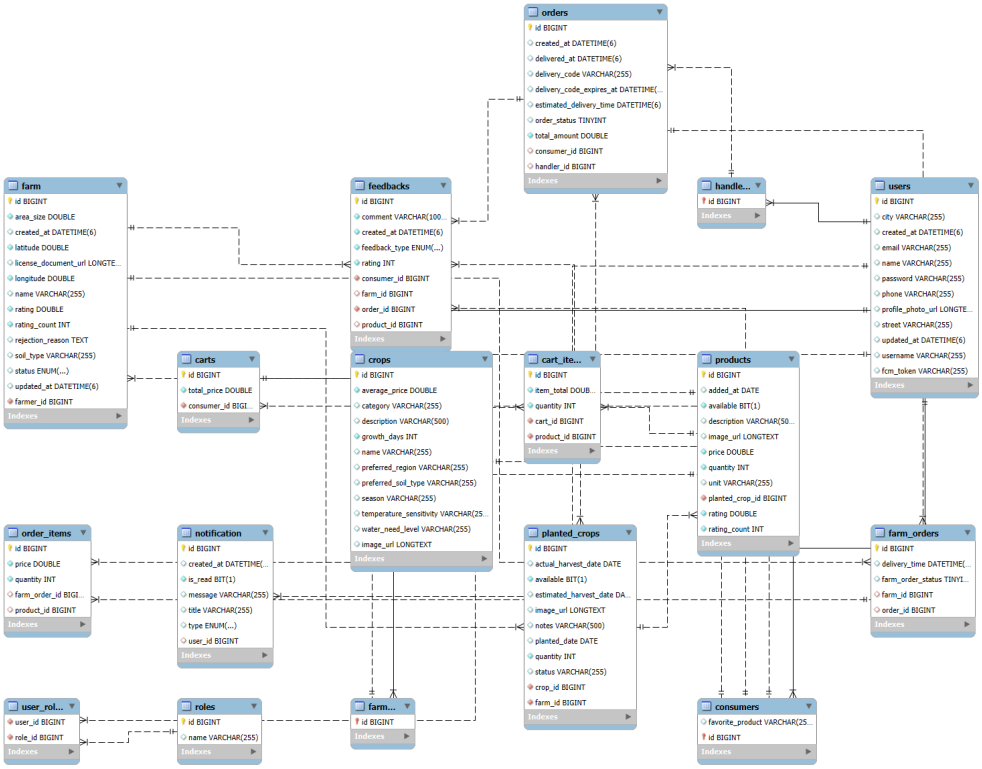

Figure 4.2: ER Diagram of the FarmX database schema

The main tables in the FarmX database include:

- `users`: Stores all user accounts including farmers, consumers, and handlers.
- `roles`: Defines possible user roles within the system.
- `user_roles`: Maps users to their assigned roles.
- `farmers`, `consumers`, `handlers`: Role-specific user details extending `users`.
- `farm`: Contains details about farms such as location, owner, and rating.
- `crops`: Defines crop types managed in the system.
- `planted_crops`: Tracks crops planted in each farm with dates and quantities.
- `products`: Products listed for sale, linked to farms.
- `carts` & `cart_items`: Manage shopping carts and their contents.
- `orders` & `order_items`: Handle finalized orders and ordered products.
- `feedbacks`: Stores ratings and comments from consumers about farms and products.
- `notification`: Stores system notifications sent to users.

These tables work together to support the complex workflows of FarmX, enabling multi-role user management, product ordering, feedback, and notifications.

Table List from the database:

`cart_items`, `carts`, `consumers`, `crops`, `farm`, `farm_orders`, `farmers`, `feedbacks`, `handlers`, `notification`, `order_items`, `orders`, `planted_crops`, `products`, `roles`, `user_roles`, `users`

4.1.4 Backend Development

The backend of FarmX is implemented using **Spring Boot**, a powerful framework that facilitates building scalable and secure RESTful APIs. The backend handles core business logic, data persistence, security, and integration with external services such as Firebase for notifications.

1. Architecture Overview The backend follows a layered architecture pattern, divided into:

- **Controller Layer:** Handles HTTP requests and responses, maps endpoints.
- **Service Layer:** Contains business logic and transaction management.
- **Repository Layer:** Manages data persistence with the database using Spring Data JPA.

2. Models and Data Persistence Entities represent the database tables and use **JPA annotations** for ORM mapping. Below are some key entities:

- `UserEntity`: Base class for all users, containing common fields such as `id`, `username`, `password`, and `roles`.
- `Farmer`, `Consumer`, `Handler`: Extend `UserEntity` for role-specific attributes.
- `Farm`: Contains farm details like name, location, owner (farmer), and rating.
- `Product`: Represents products linked to farms with details such as name, price, and quantity.
- `Order` and `Cart`: Manage orders and shopping cart items.
- `Feedback`: Stores user feedback and ratings for farms and products.

Repositories use Spring Data JPA to provide CRUD operations and custom queries without boilerplate code.

3. Data Transfer Objects (DTOs) To decouple API contracts from the database entities, FarmX uses DTOs. These are simple POJOs used to transfer data between client and server. Examples include:

- `FarmDto`: Contains only necessary fields like farm name, location, and average rating.
- `UserDto`: Used for user registration and profile data.
- `ProductDto`, `OrderDto`, `FeedbackDto`: Serve similar purposes for their respective entities.

Mapping between Entities and DTOs occurs within service methods.

4. Authentication and Authorization FarmX uses **Spring Security** with **JWT** for stateless authentication and role-based access control:

- On login, credentials are validated against stored users.
- A signed JWT token is issued containing user identity and roles.
- The client includes the JWT in the `Authorization` header on subsequent requests.
- Spring Security's JWT filter validates tokens and populates the security context.
- Endpoints are secured using annotations like `@PreAuthorize("hasRole('FARMER')")` to restrict access by roles.

Example of a security configuration class:

```
@Configuration
@EnableWebSecurity
@EnableMethodSecurity
public class SecurityConfig {

    @Bean
    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
        http
            .cors(cors -> cors.configurationSource(corsConfigurationSource()))
            .csrf(csrf -> csrf.disable())
            .exceptionHandling(exception -> exception.authenticationEntryPoint(authEntryPoint))
            .sessionManagement(session -> session.sessionCreationPolicy(
            SessionCreationPolicy.STATELESS))
            .authorizeHttpRequests(auth -> auth
                .requestMatchers("/signup", "/login").permitAll()
                 .requestMatchers("/ws-notifications/**").permitAll()
                .requestMatchers(HttpMethod.GET,
                "/swagger-ui/**", "/v3/api-docs/**", "/swagger-ui.html").permitAll()
                .anyRequest().authenticated()
            )
            .addFilterBefore(jwtAuthenticationFilter(), UsernamePasswordAuthenticationFilter.cl

        return http.build();
    }
```

```
        @Bean
    public CorsConfigurationSource corsConfigurationSource() {
        CorsConfiguration configuration = new CorsConfiguration();

        configuration.setAllowedOrigins(Arrays.asList("http://localhost:3000","https://farmx-ab
        configuration.setAllowedMethods(Arrays.asList("GET", "POST", "PUT", "DELETE", "OPTIONS"
        configuration.setAllowedHeaders(Arrays.asList("Authorization", "Content-Type"));
       configuration.setAllowCredentials(true); // Allow cookies, tokens, etc.

        UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();
        source.registerCorsConfiguration("/**", configuration);
        return source;
    }

}
```

5. Principal Usage Controllers receive the authenticated user information via the `Principal` object, enabling operations specific to the current user. Example:

```
@GetMapping("/farms")
public List<FarmDto> getMyFarms(Principal principal) {
    String username = principal.getName();
    return farmService.getFarmsByOwner(username);
}
```

6. Private Notifications Using Firebase Cloud Messaging FarmX integrates with **Firebase Cloud Messaging (FCM)** to send real-time push notifications to users:

- User devices register their unique FCM tokens which are stored in the backend.
- The backend sends notifications for events such as order status updates, new messages, and feedback responses.
- Notifications are targeted to individual users by their stored tokens, enabling private messaging without WebSocket overhead.

The backend uses Firebase Admin SDK, configured as follows:

```
@Configuration
public class FirebaseConfig {
```

```
  @Value("${FIREBASE_CREDENTIALS:}")
    private String firebaseEnvCredentials;

    @PostConstruct
    public void initializeFirebase() throws IOException {
        InputStream serviceAccount;

        if (!firebaseEnvCredentials.isEmpty()) {
            // running on Render, decode string
            byte[] decoded = Base64.getDecoder().decode(firebaseEnvCredentials);
            serviceAccount = new ByteArrayInputStream(decoded);
            System.out.println(" Firebase loaded from ENV variable");
        } else {
            // running locally, use JSON file
            serviceAccount = new FileInputStream("src/main/resources/firebase/serviceAccountKey
            System.out.println(" Firebase loaded from local file");
        }

        FirebaseOptions options = new FirebaseOptions.Builder()
                .setCredentials(GoogleCredentials.fromStream(serviceAccount))
                .build();

        if (FirebaseApp.getApps().isEmpty()) {
            FirebaseApp.initializeApp(options);
        }
    }
}
```

7. Exception Handling FarmX implements centralized exception handling with `@ControllerAdvice` to return consistent and informative error responses. For example:

```
@ControllerAdvice
public class GlobalExceptionHandler {

    @ExceptionHandler(AccessDeniedException.class)
    public ResponseEntity<String> handleAccessDenied(AccessDeniedException ex) {
```

```
        return ResponseEntity.status(HttpStatus.FORBIDDEN).body("Access Denied: " + ex.getMessa
    }

    @ExceptionHandler(EntityNotFoundException.class)
    public ResponseEntity<String> handleNotFound(EntityNotFoundException ex) {
        return ResponseEntity.status(HttpStatus.NOT_FOUND).body(ex.getMessage());
    }
}
```

Summary FarmX backend combines a clean architecture, strong security with JWT, efficient database interaction via JPA, and real-time communication using Firebase notifications. This results in a robust and maintainable platform supporting multiple user roles and complex workflows.

4.1.5 Frontend Development

The frontend of FarmX is built with a focus on providing a seamless and intuitive user experience across web and mobile platforms. The project uses React and Next.js for the web interface, and React Native with Expo for the mobile application. This section provides a comprehensive overview of the user interfaces, supported features, and key screens along with corresponding screenshots.

Web Application (React + Next.js)

The web frontend offers users a rich interface to interact with all core functionalities of FarmX. Below are key pages and their descriptions with corresponding screenshots:

Home Page & About

The homepage offers an immersive 3D virtual farm environment that introduces users to FarmX, along with general platform information.

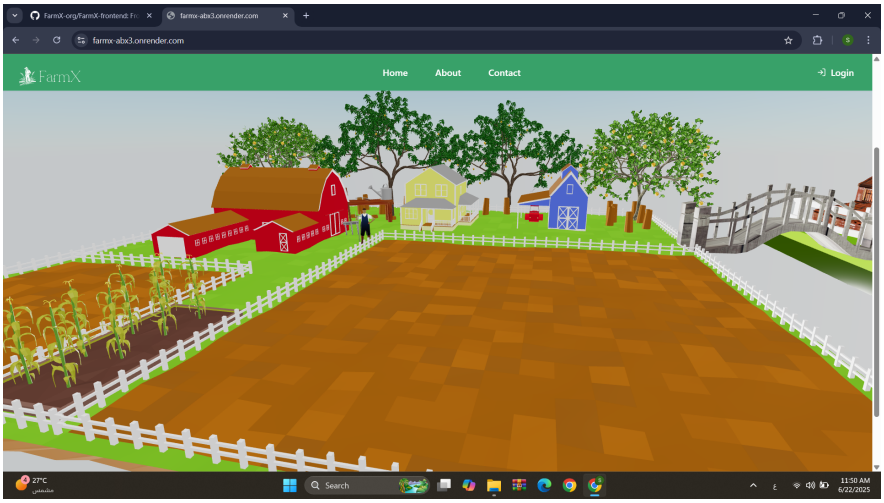

Figure 4.3: FarmX Web Home Page - Desktop View

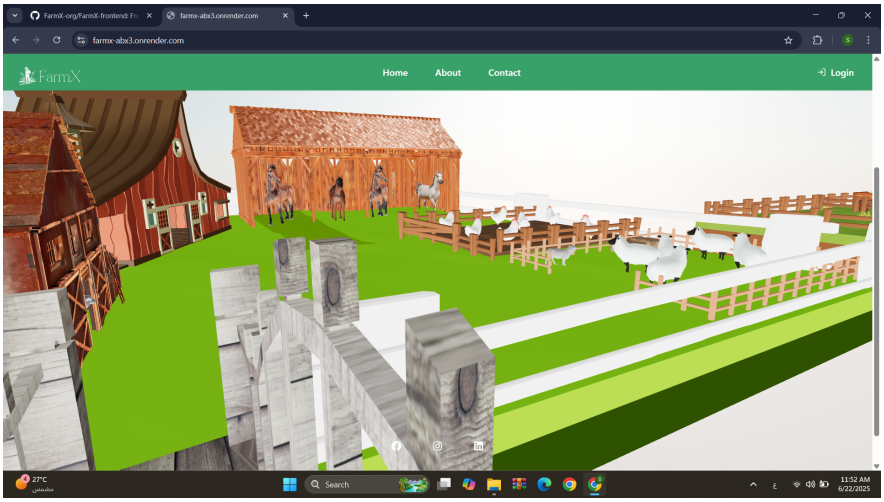

Figure 4.4: Additional Homepage View 1

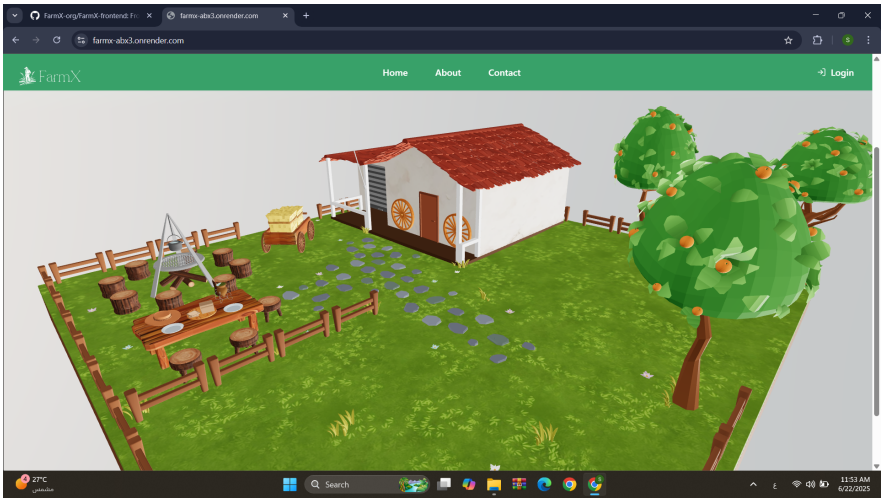

Figure 4.5: Additional Homepage View 2

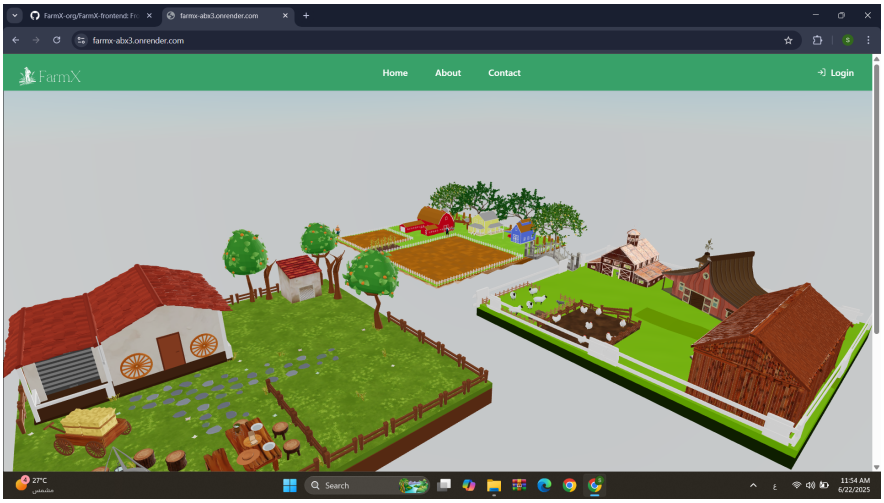

Figure 4.6: Additional Homepage View 3

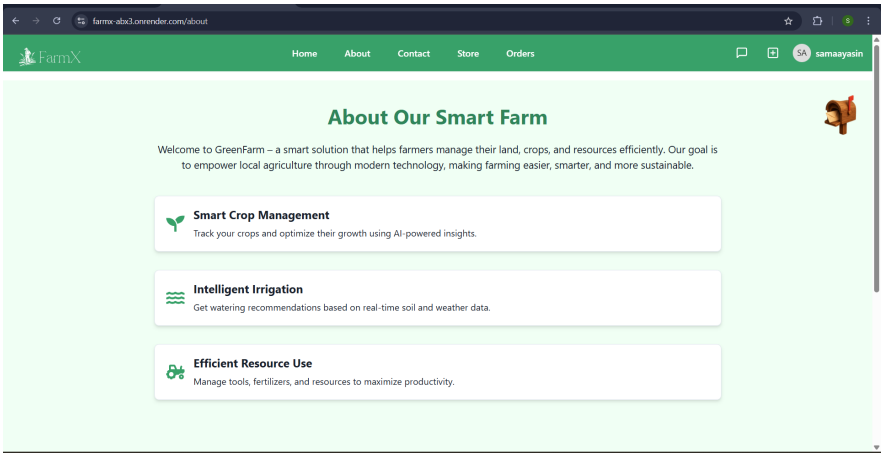

Figure 4.7: About Page Screenshot

Farm Management

Farmers can add, edit, and manage farms and crops efficiently.

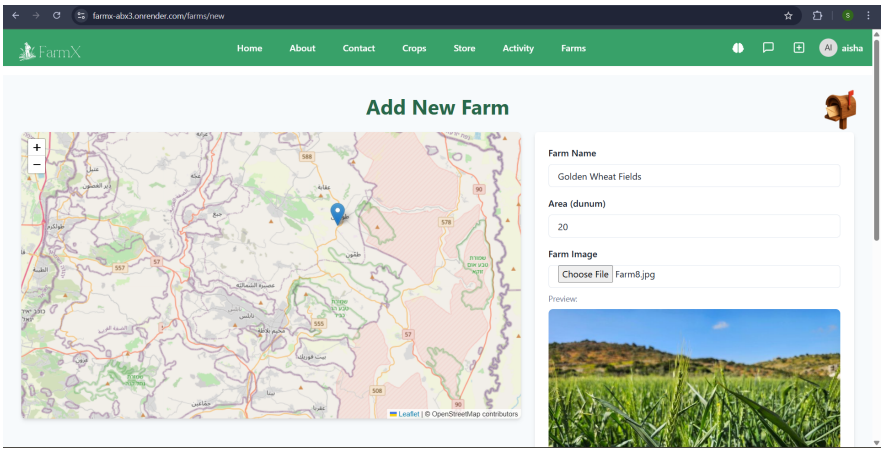

Figure 4.8: Add Farm

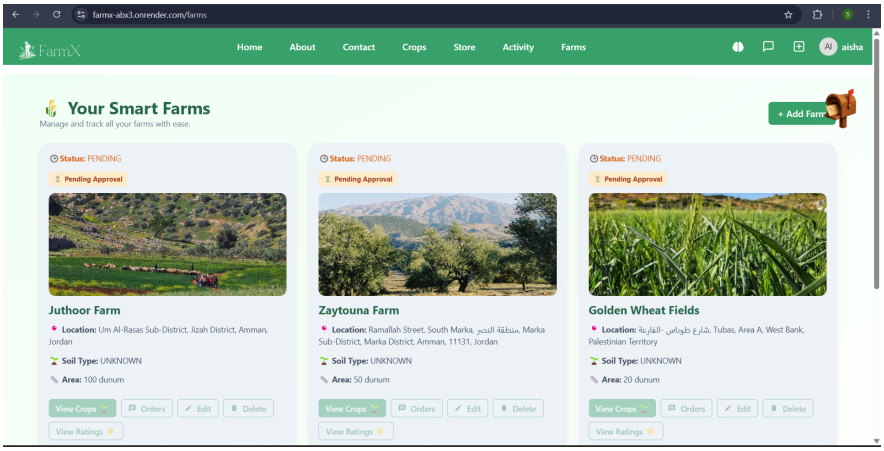

Figure 4.9: My Farms - View

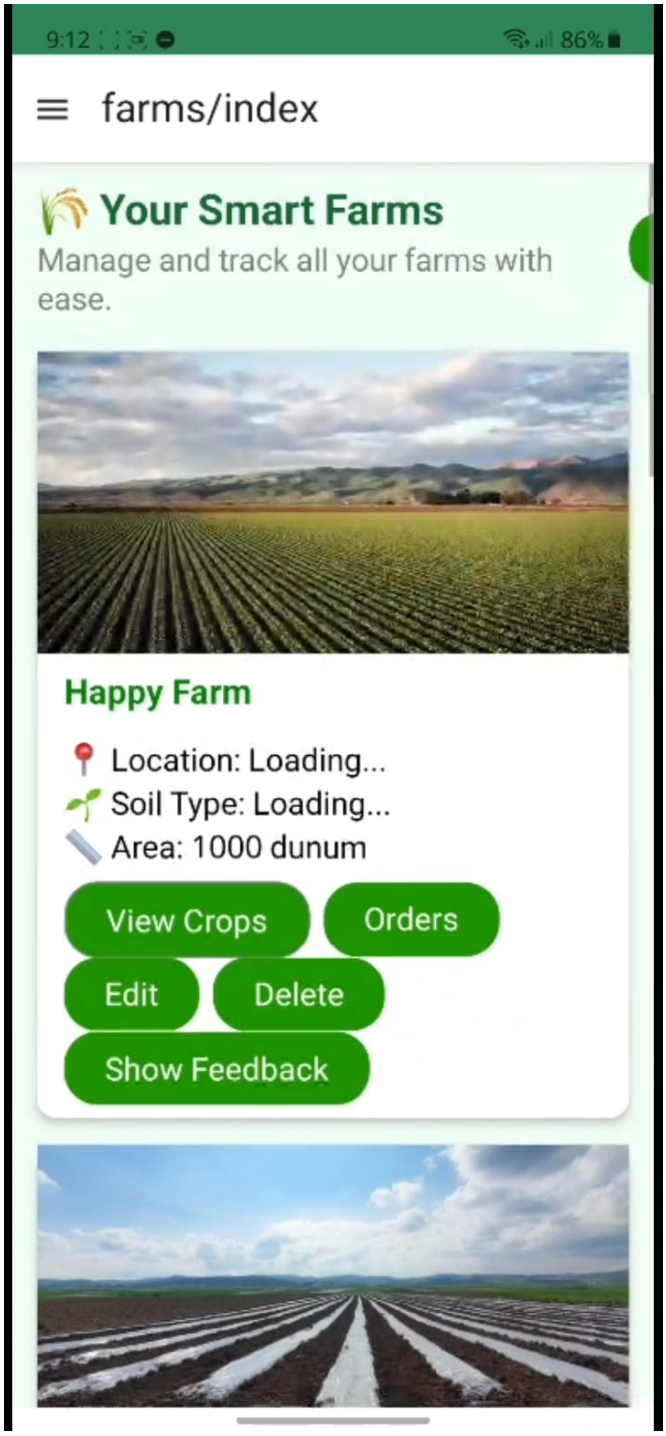

Figure 4.10: My Farms - Mobile View

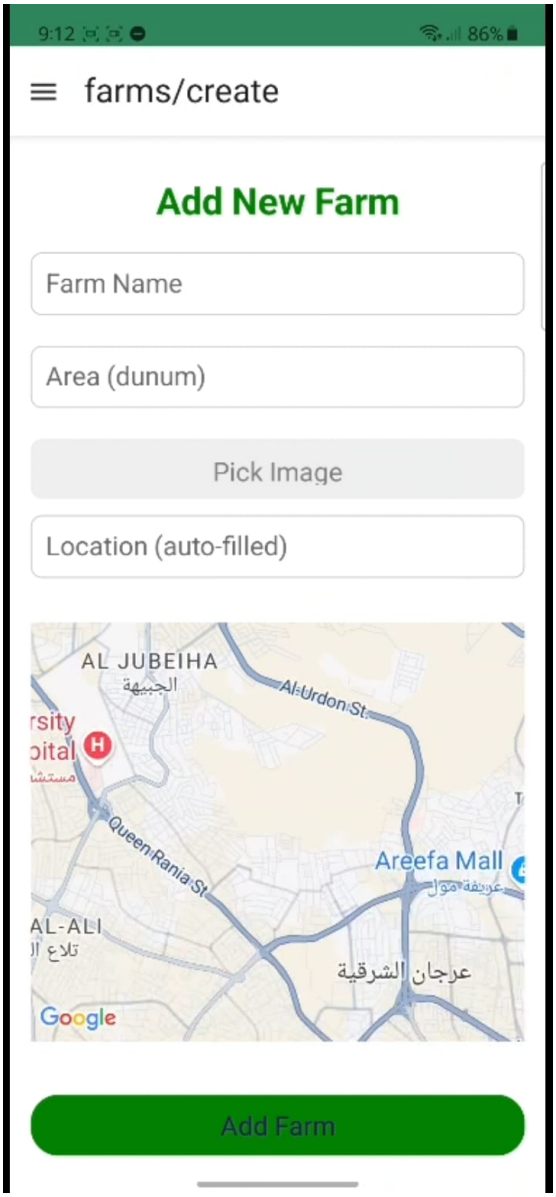

Figure 4.11: Add Farm - Mobile

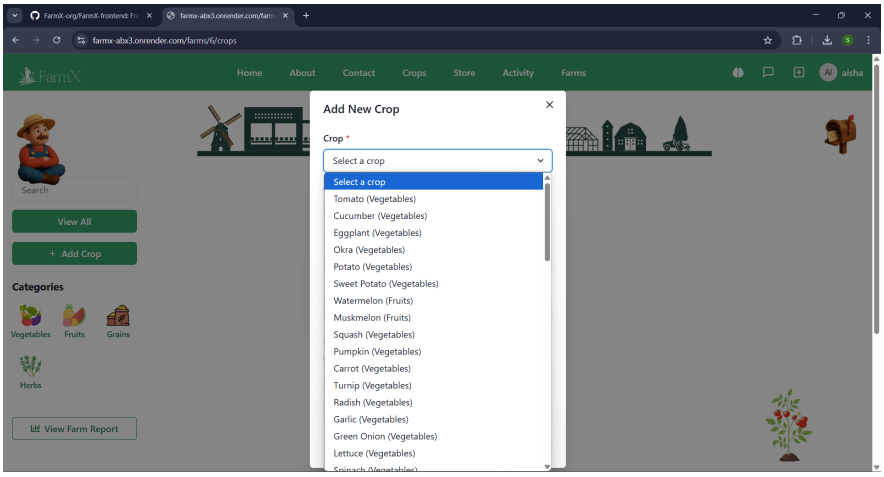

Figure 4.12: Add Crop 2 Screen

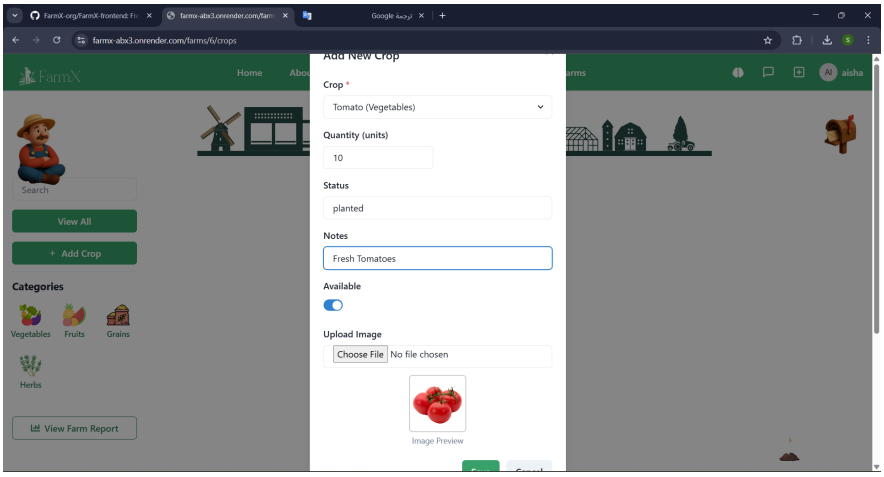

Figure 4.13: Add Crop 3 Screen

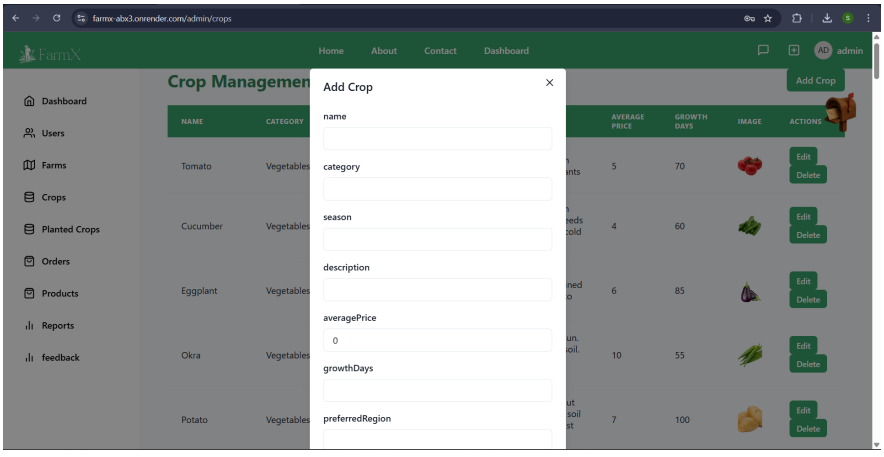

Figure 4.14: Add Crop Admin Panel

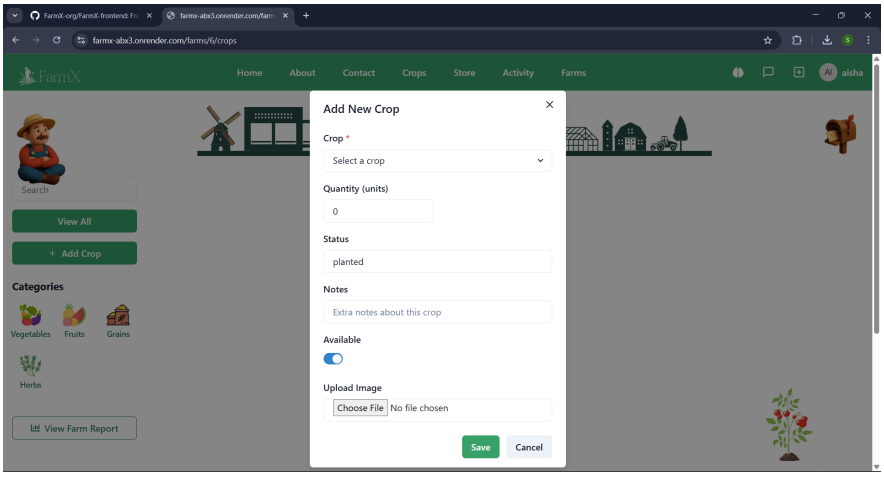

Figure 4.15: Add Crop Screen

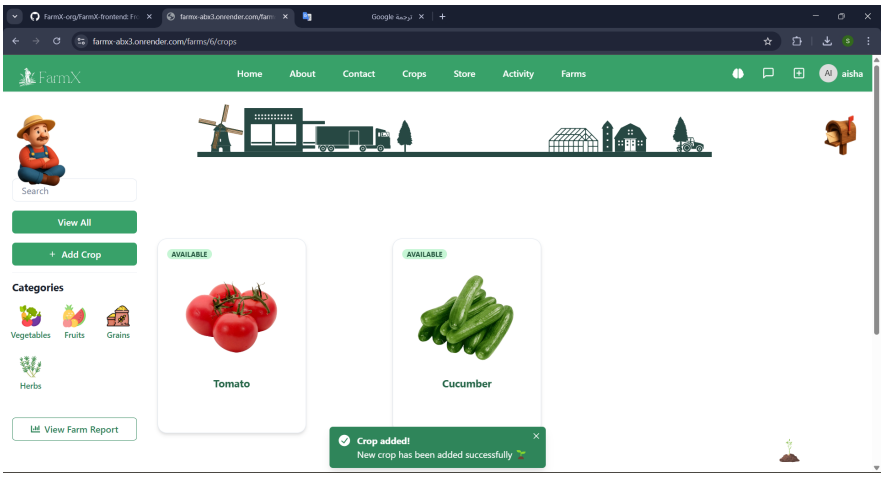

Figure 4.16: Crop Added Confirmation

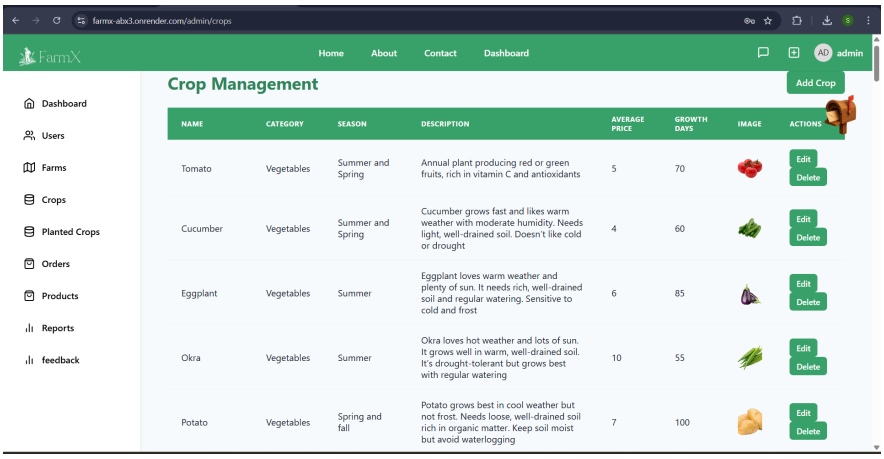

Figure 4.17: Crop Management Interface

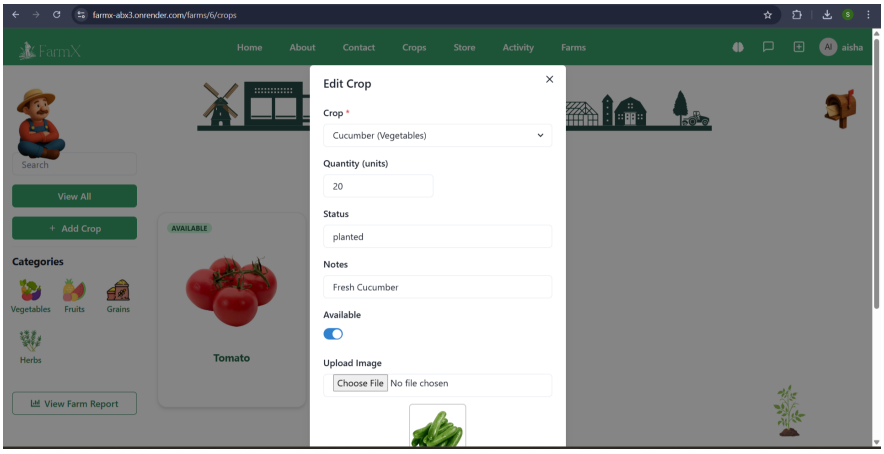

Figure 4.18: Edit Crop Screen

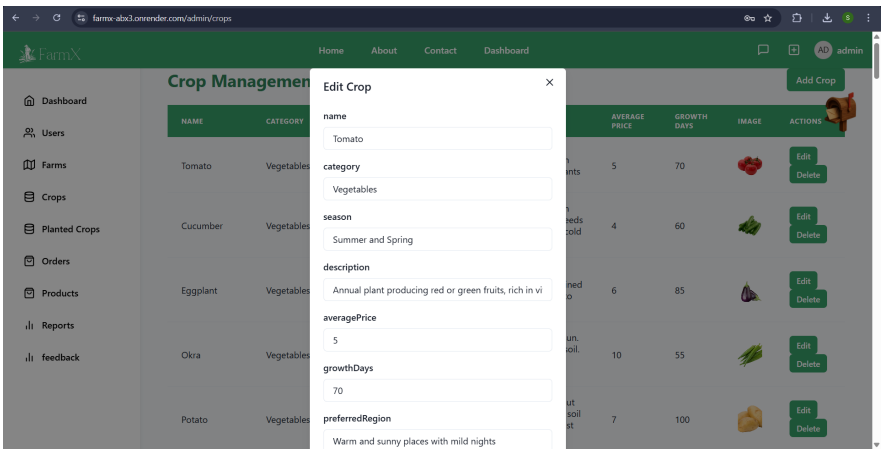

Figure 4.19: Edit Crop Admin Panel

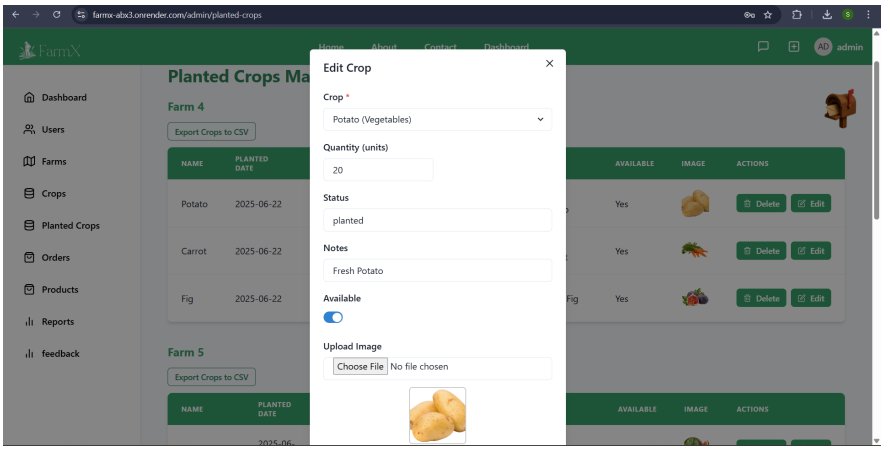

Figure 4.20: Edit Planted Crop Admin

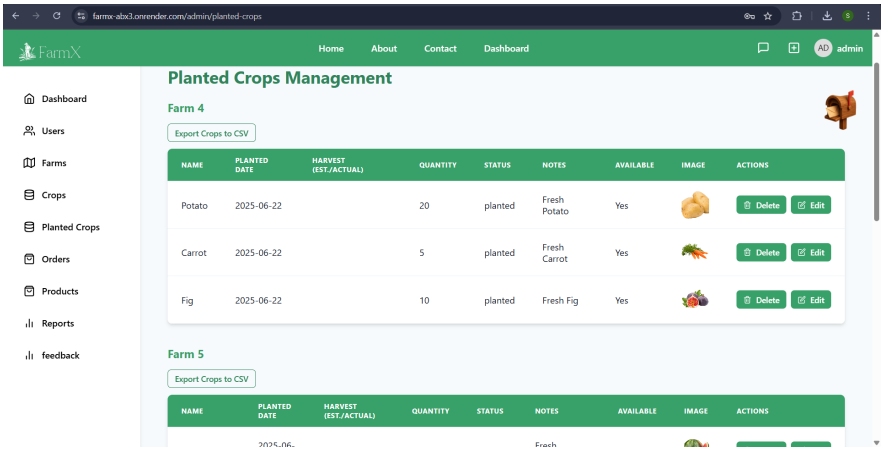

Figure 4.21: Planted Crops Overview

Farm Status & Admin Operations

Management of farm approvals and rejections by the admin.

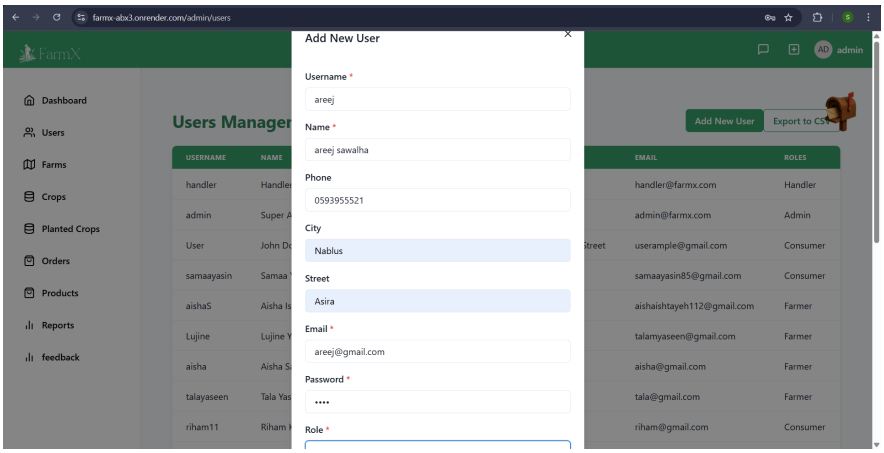

Figure 4.22: Add User Screen

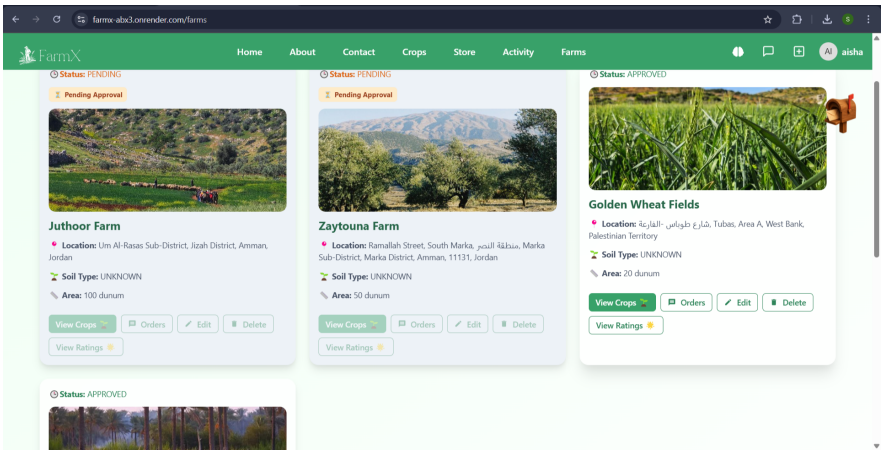

Figure 4.23: After Farm Approval

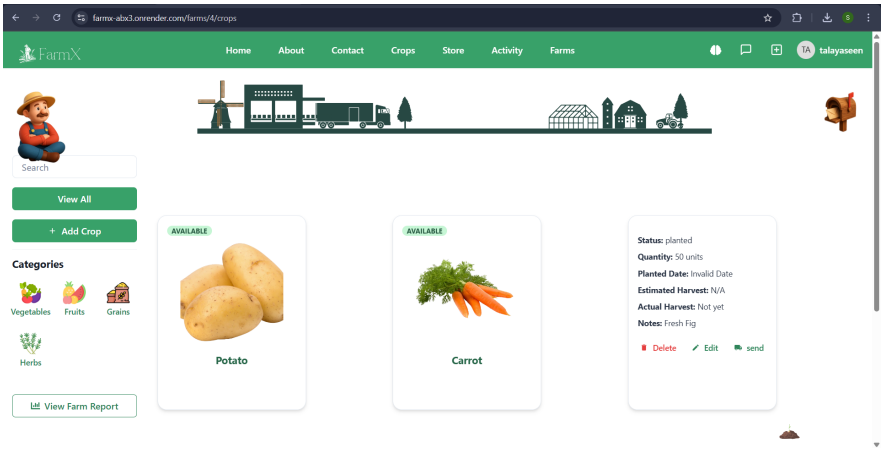

Figure 4.24: After Sending Farm for Approval

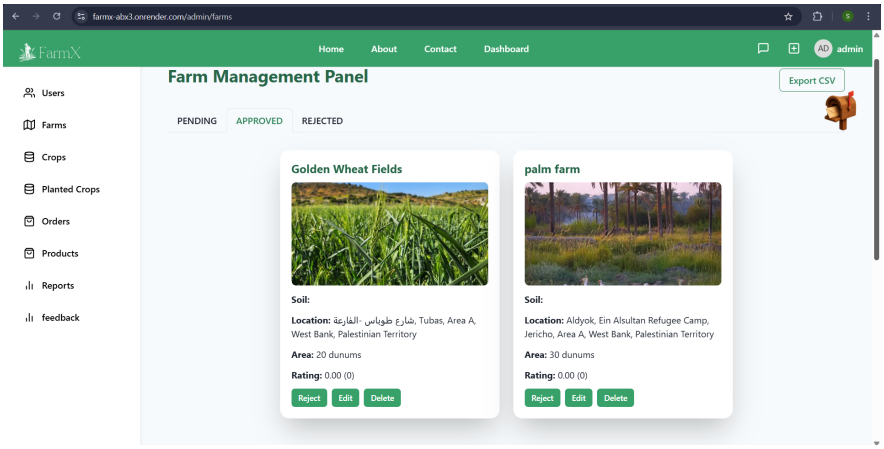

Figure 4.25: Approved Farm View

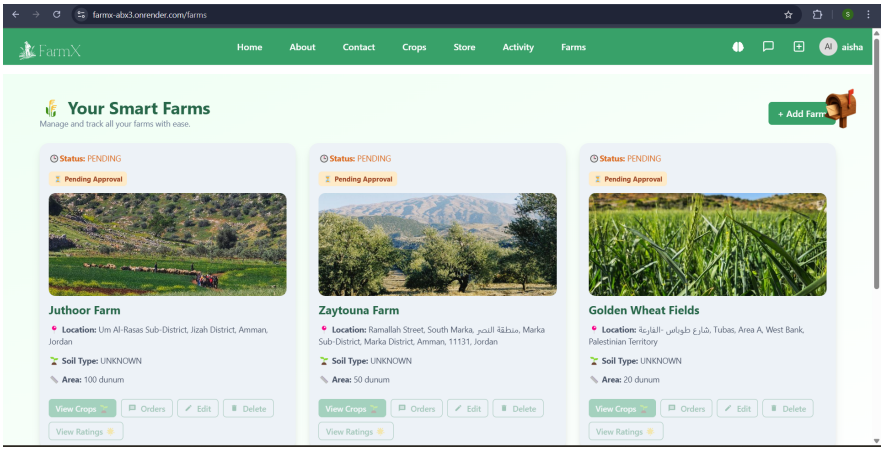

Figure 4.26: Pending Farms List

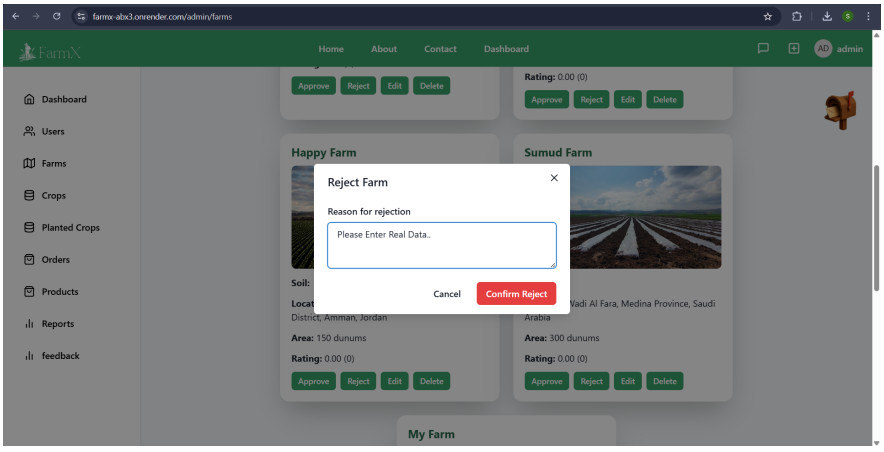

Figure 4.27: Reject Farm Screen

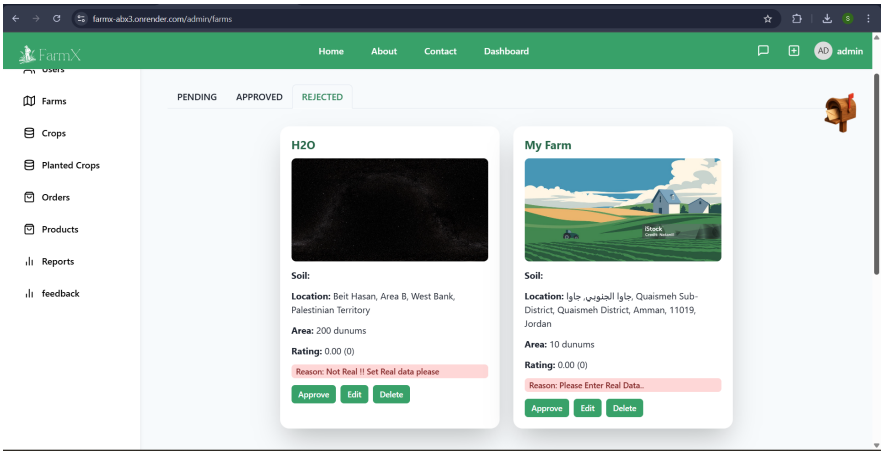

Figure 4.28: Rejected Farm View

Store & Cart

Users can browse products from multiple farms, use filters, and manage their cart efficiently. Quantity restrictions are handled gracefully.

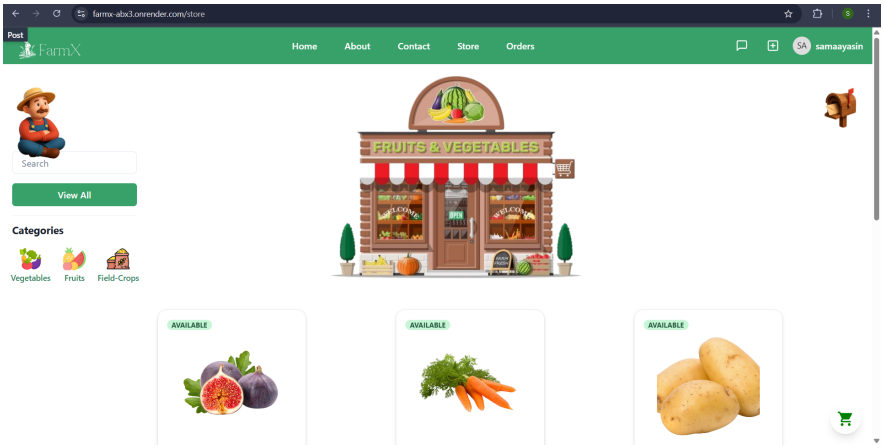

Figure 4.29: FarmX Store - Product Listing

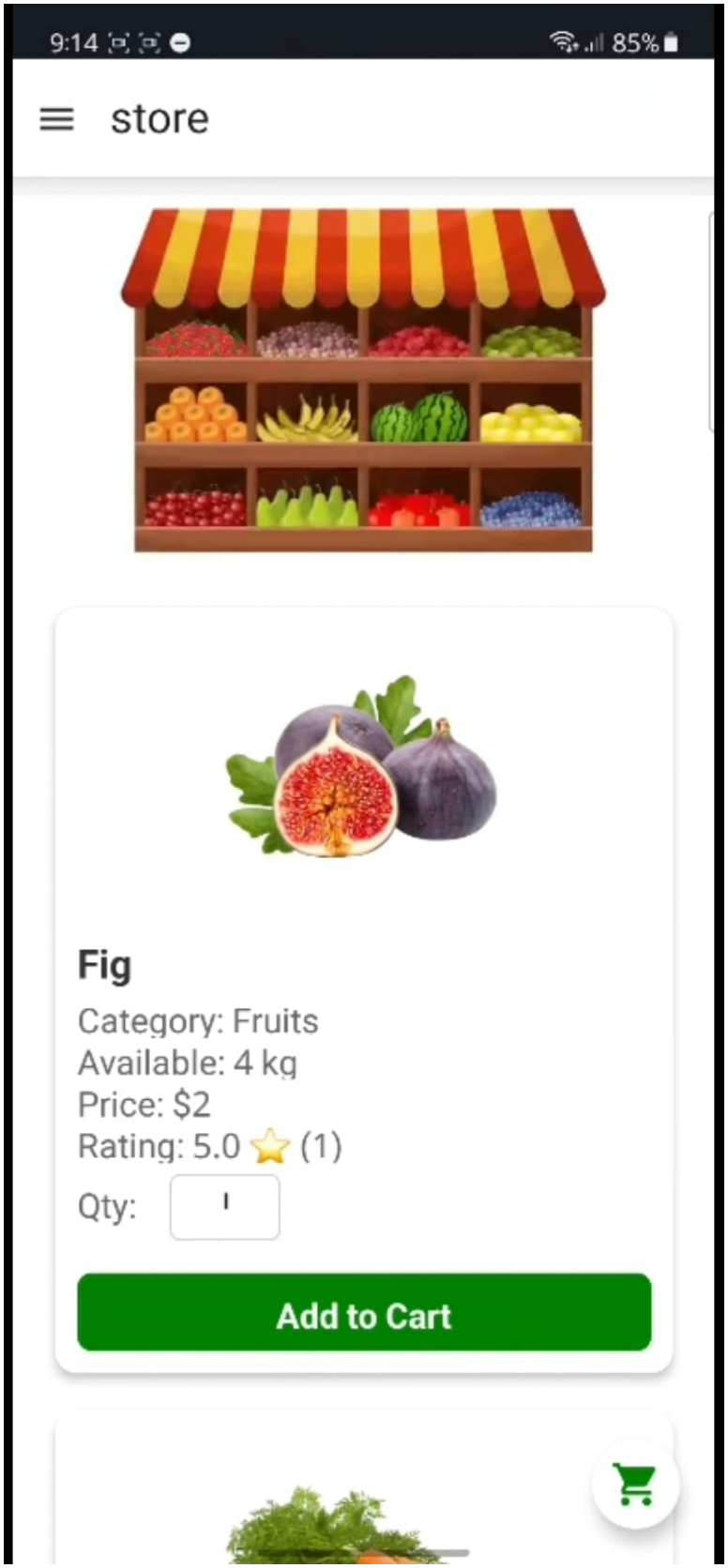

Figure 4.30: Mobile Store View

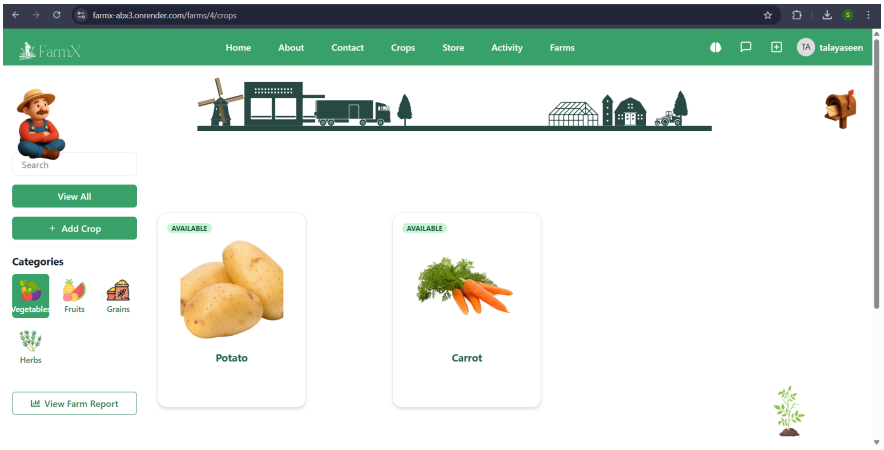

Figure 4.31: Filtering Options in Store

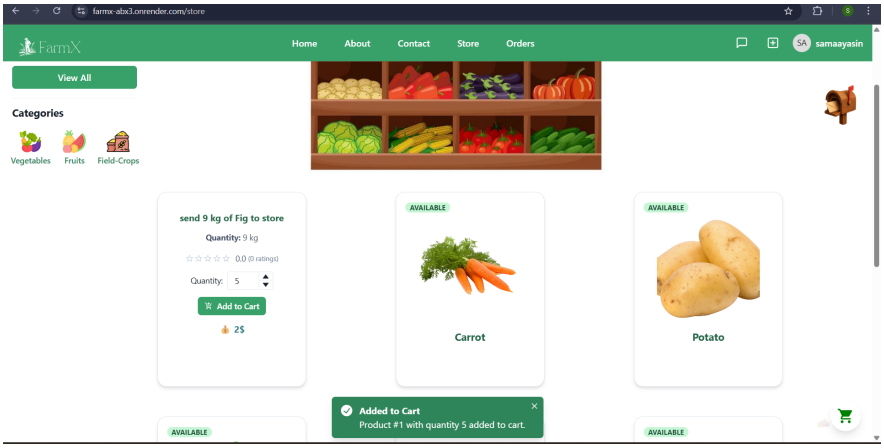

Figure 4.32: Add to Cart Action

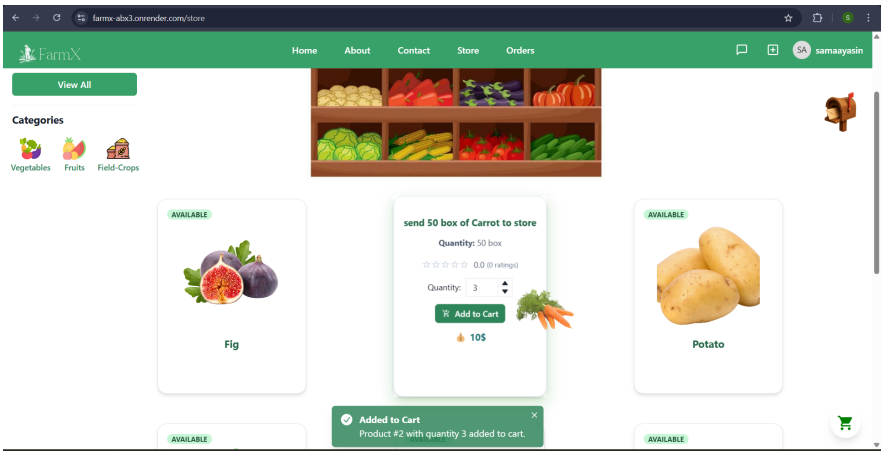

Figure 4.33: Add to Cart Confirmation

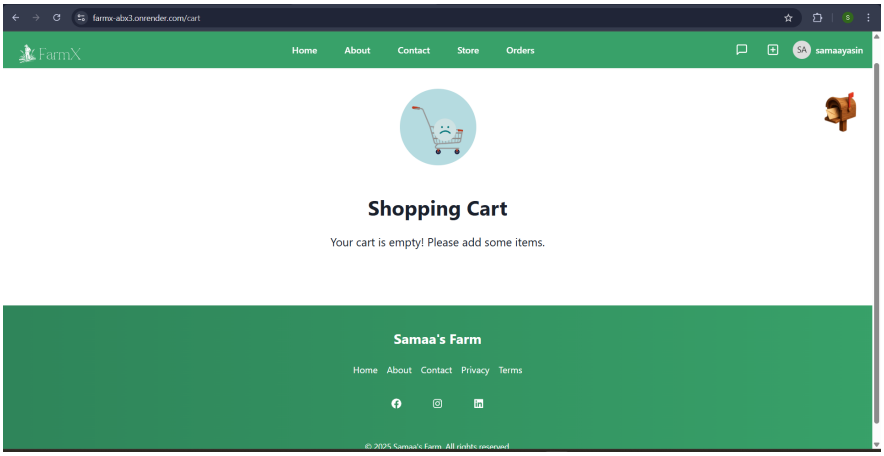

Figure 4.34: Empty Cart State

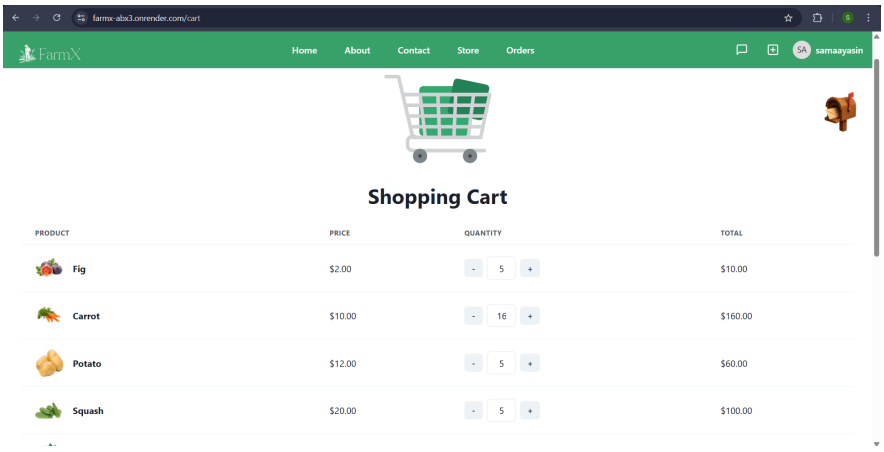

Figure 4.35: Cart View 1

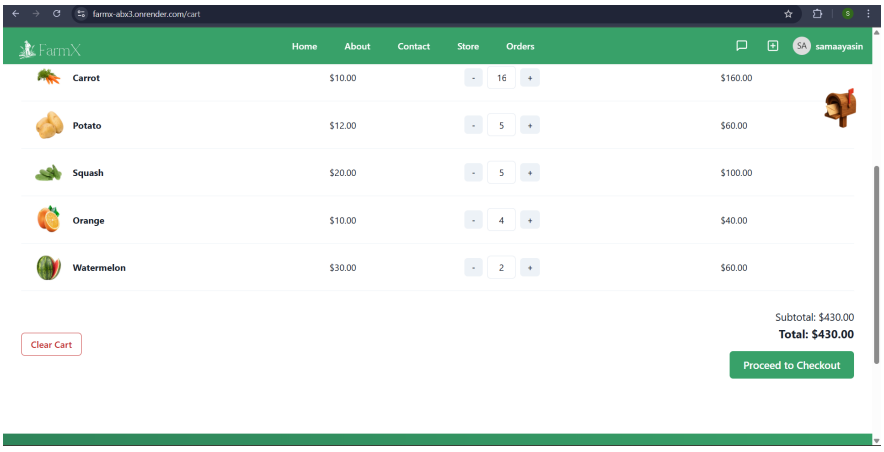

Figure 4.36: Cart View 2

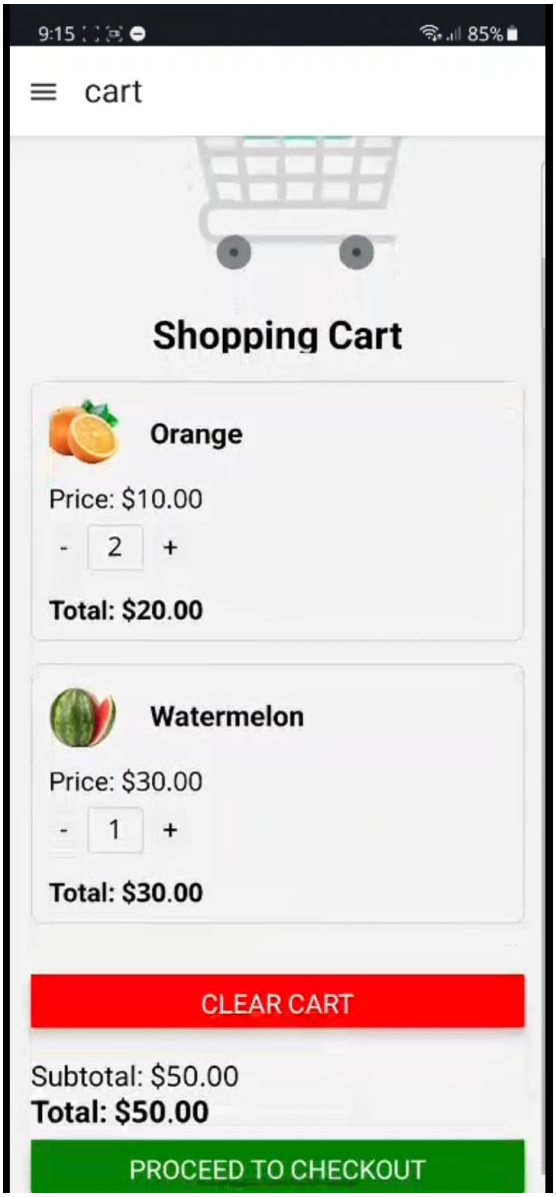

Figure 4.37: Mobile Cart View

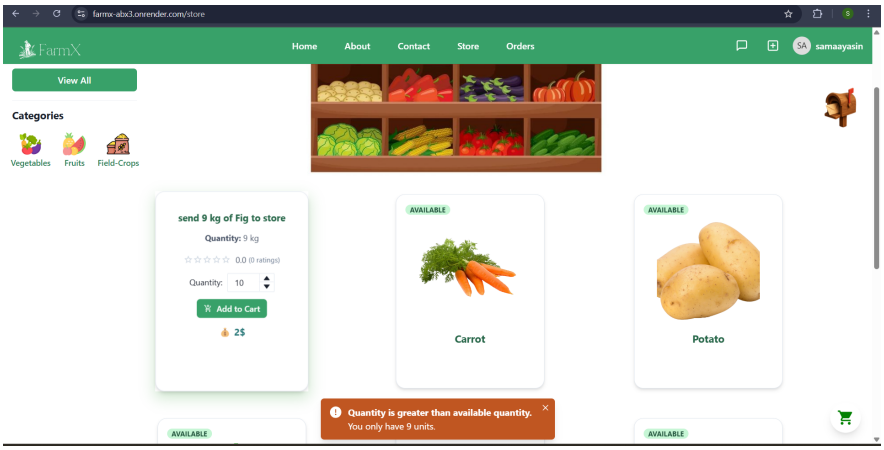

Figure 4.38: Quantity Limit Warning when Ordering

User Authentication & Profile

Users can register, login, and manage their profile across web and mobile platforms.

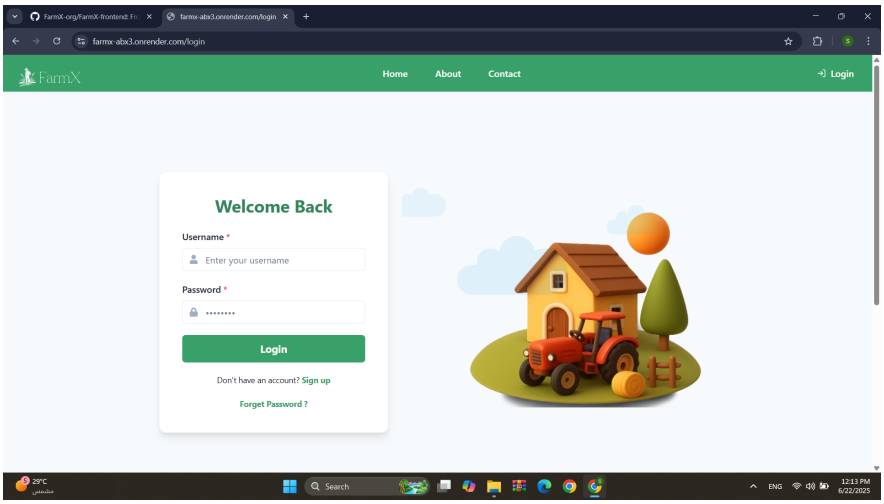

Figure 4.39: Login Screen - Web

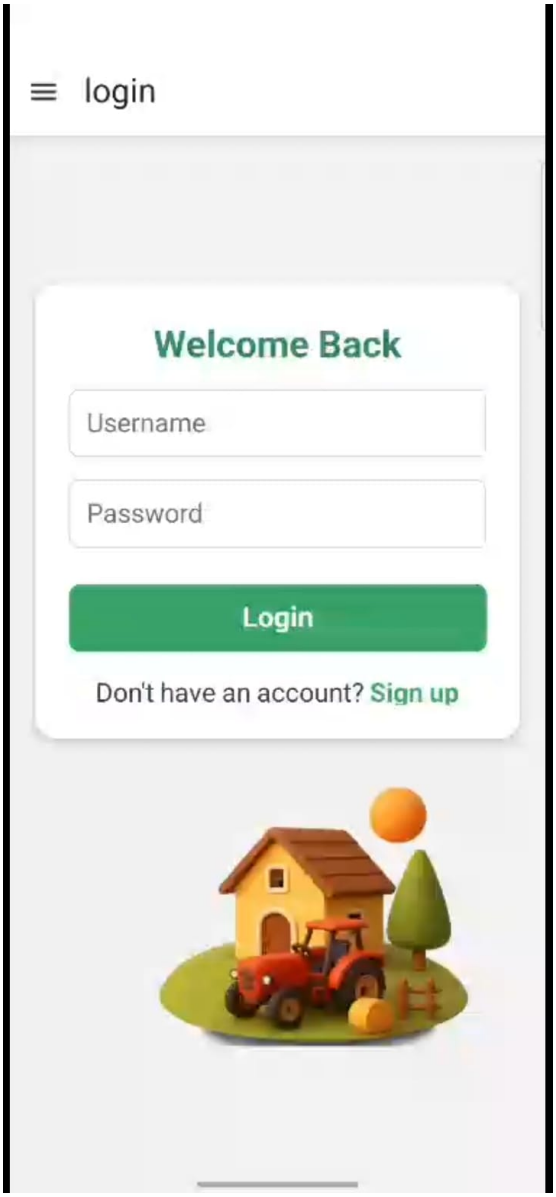

Figure 4.40: Login Screen - Mobile

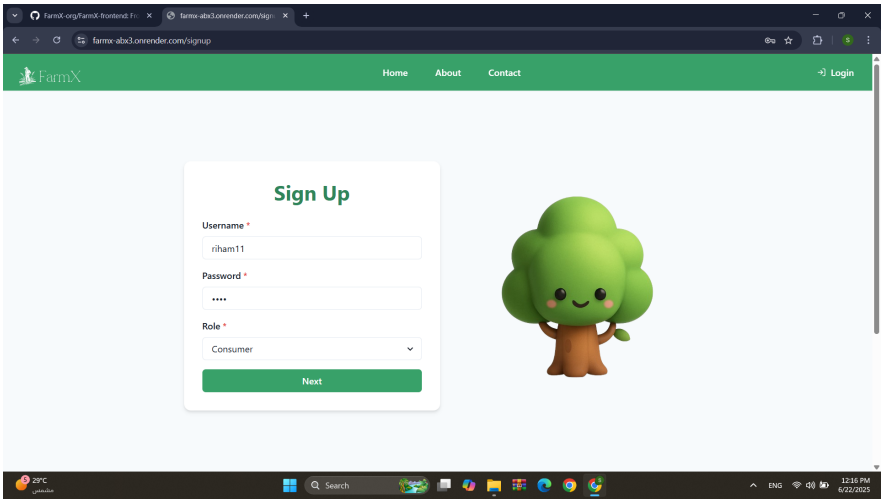

Figure 4.41: Signup Screen - Web

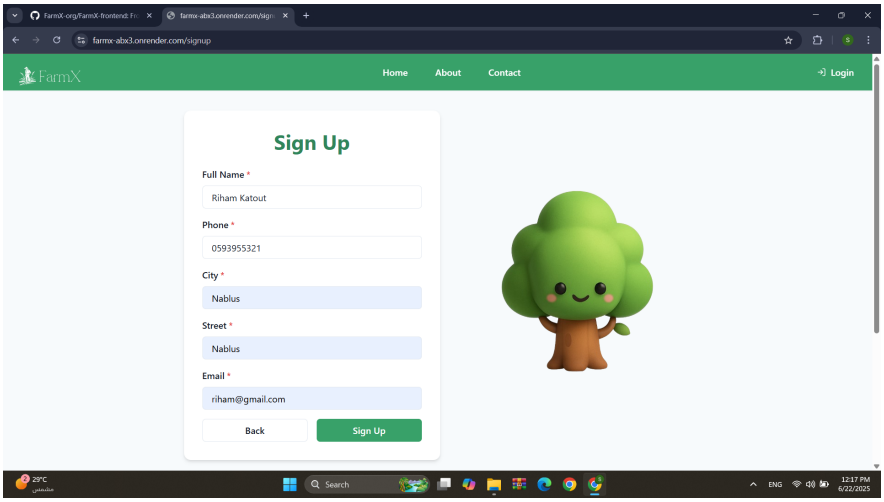

Figure 4.42: Alternate Signup Screen

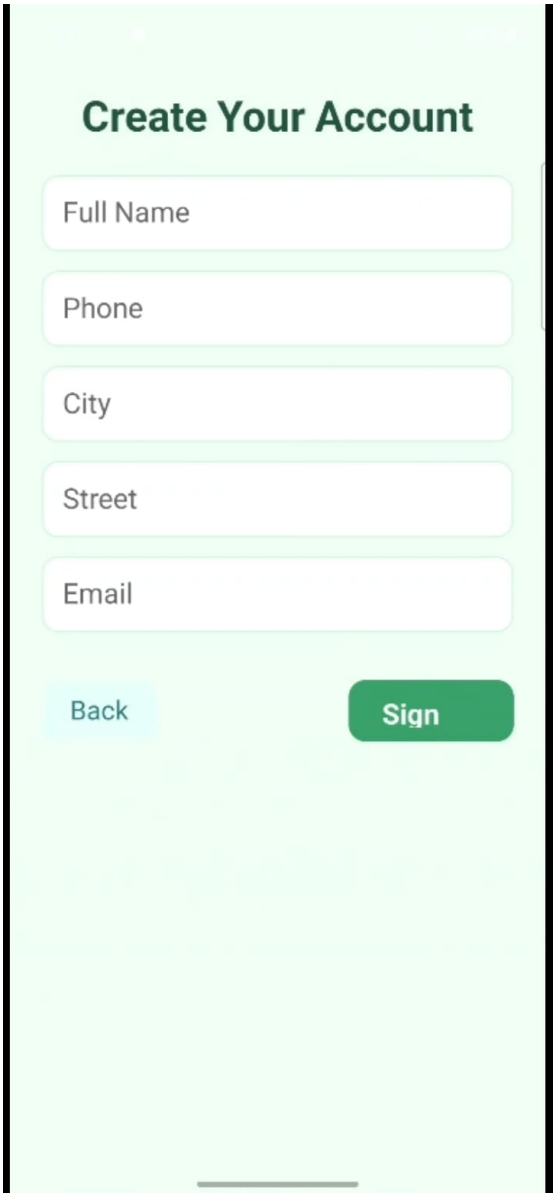

Figure 4.43: Signup Screen - Mobile

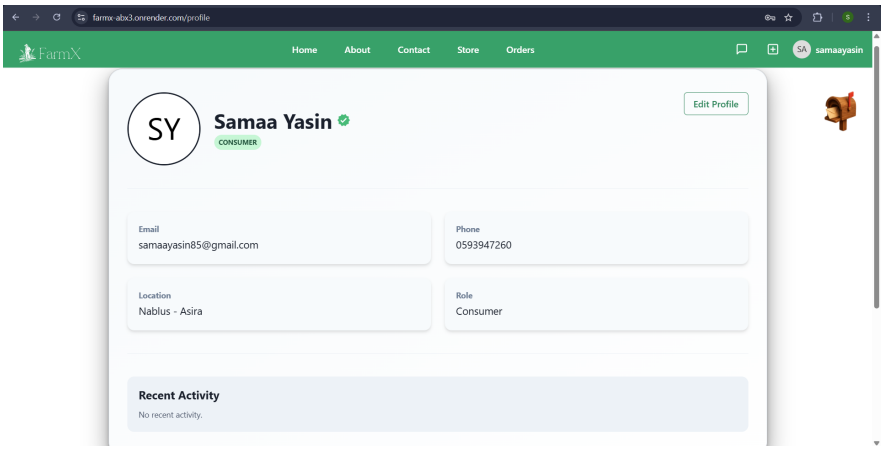

Figure 4.44: User Profile Management

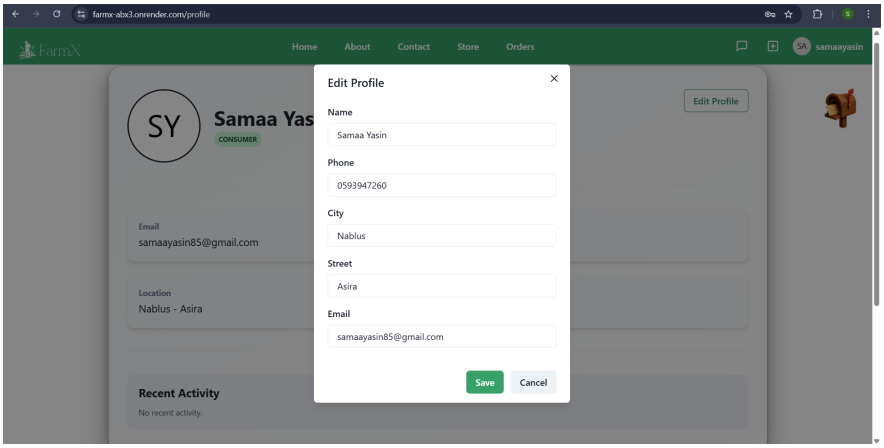

Figure 4.45: Edit Profile Screen

Admin Panel

Admins have control over users, farms, products, orders, and feedback with powerful management dashboards.

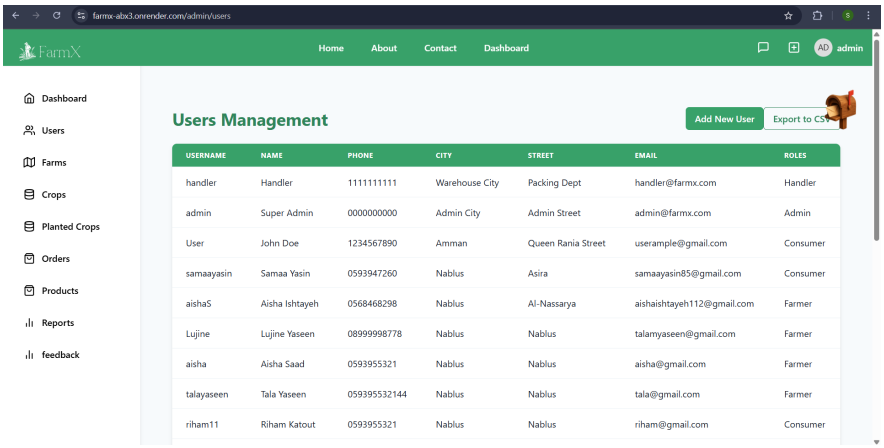

Figure 4.46: Admin User Management

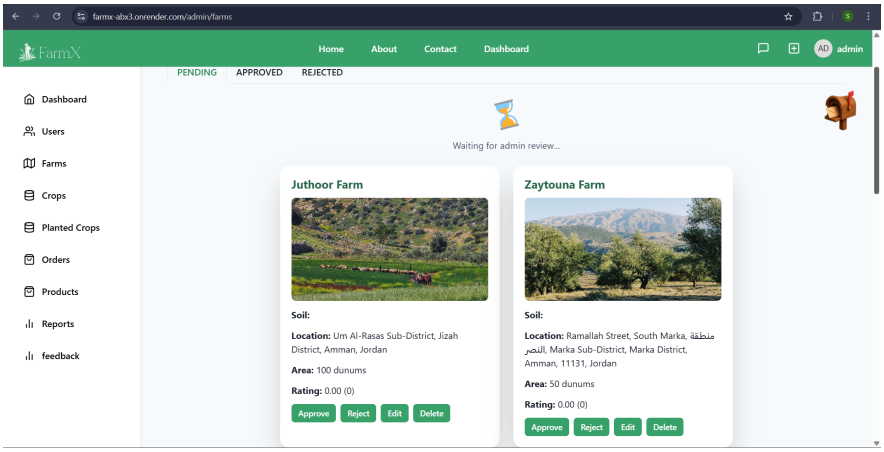

Figure 4.47: Admin Farm Management

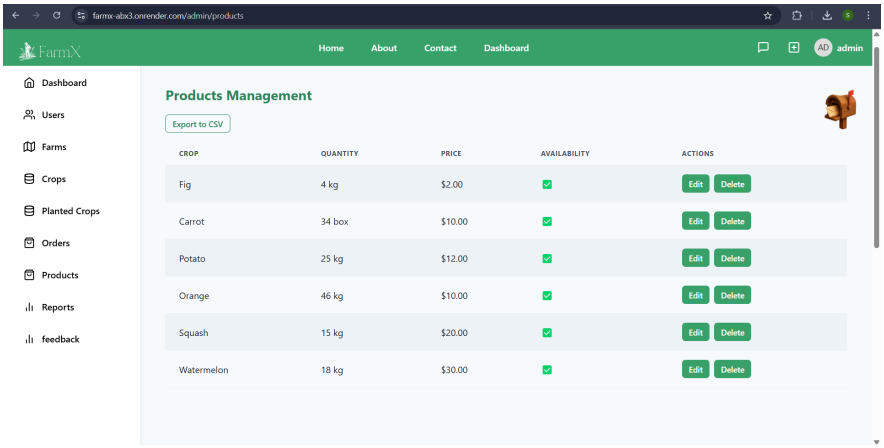

Figure 4.48: Product Management Interface

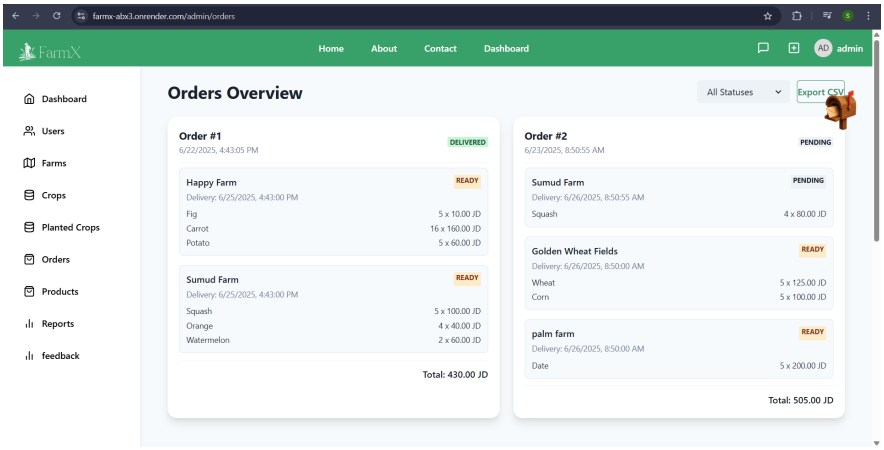

Figure 4.49: Order Management in Admin Panel

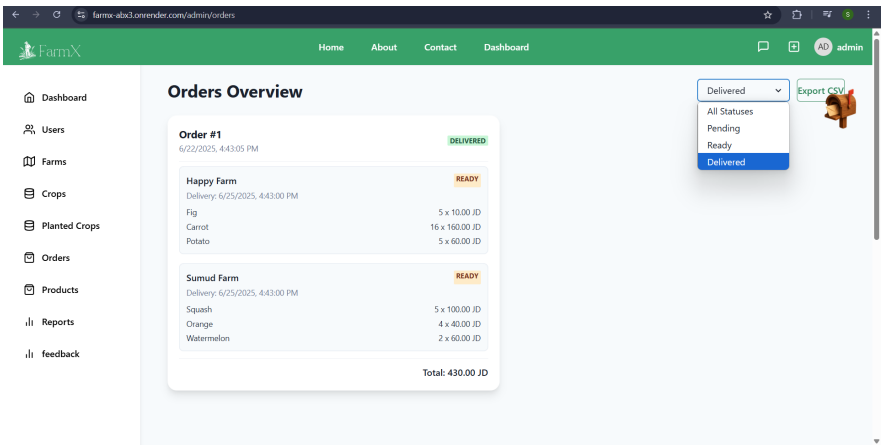

Figure 4.50: Order Filtering Options

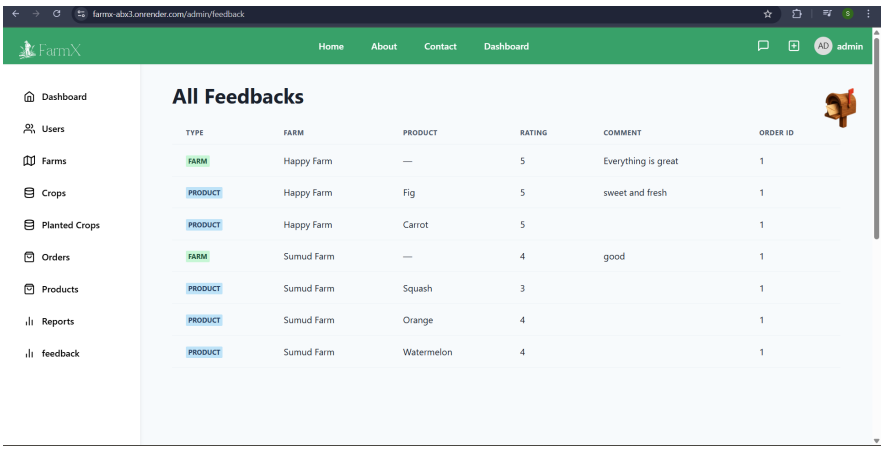

Figure 4.51: Admin Feedback Overview

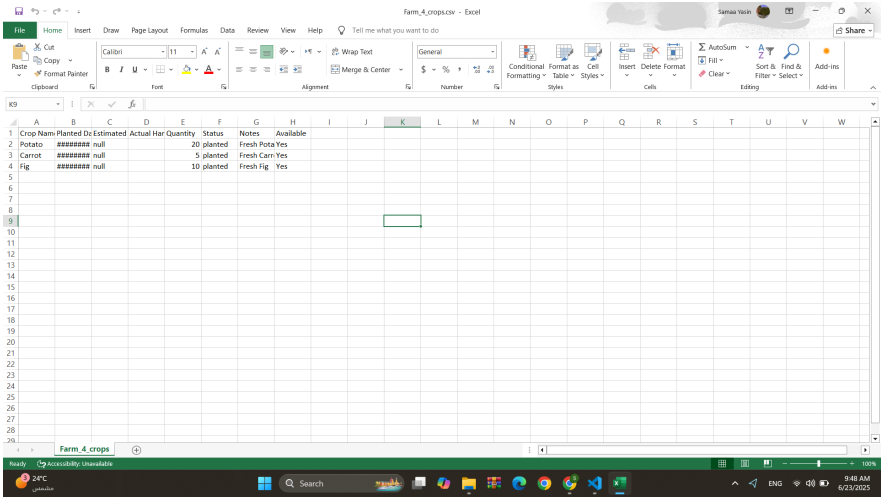

Figure 4.52: CSV Import - Planted Crops Data

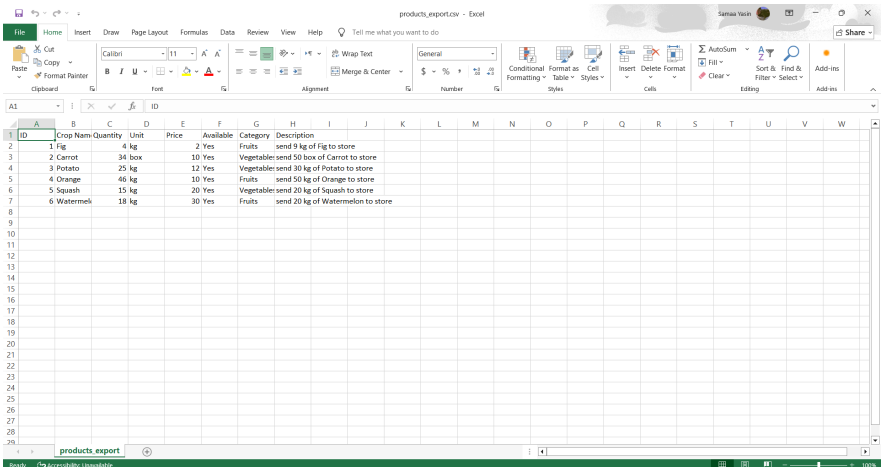

Figure 4.53: CSV Import - Products Data

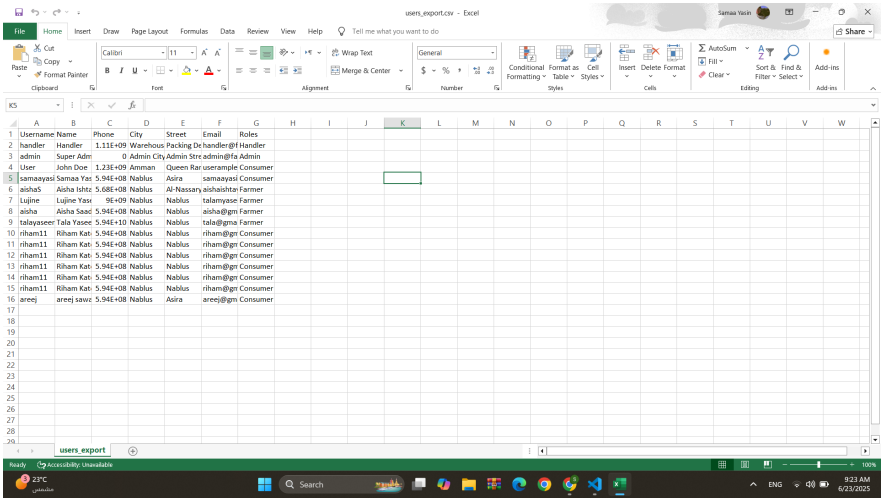

Figure 4.54: CSV Import - Users Data

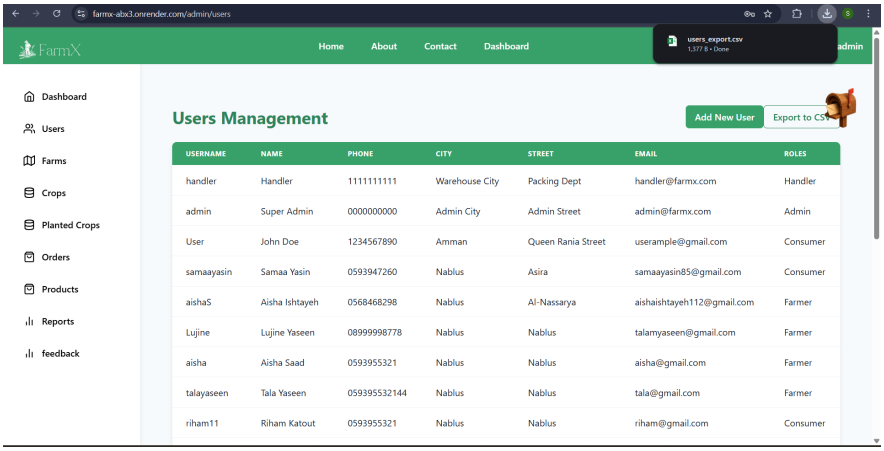

Figure 4.55: User Data Export Feature

Orders & Delivery Management

FarmX supports multi-farm orders with automatic splitting, order status tracking, and delivery confirmation.

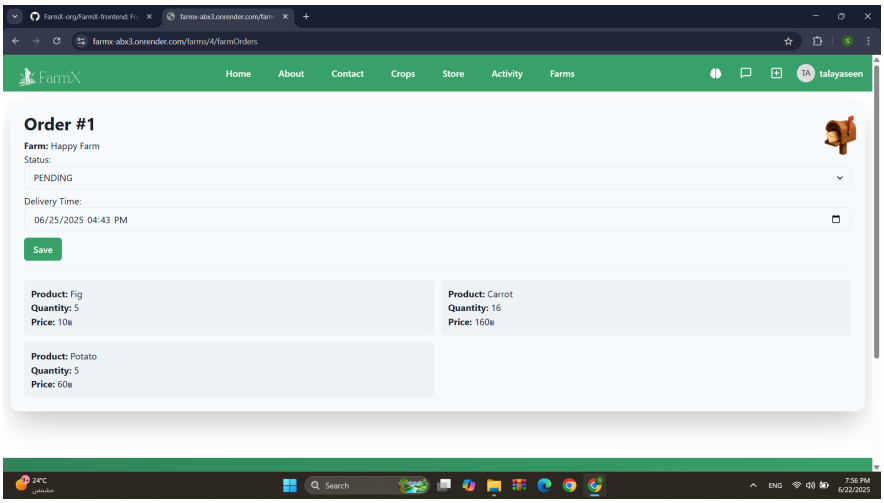

Figure 4.56: Pending Order Details

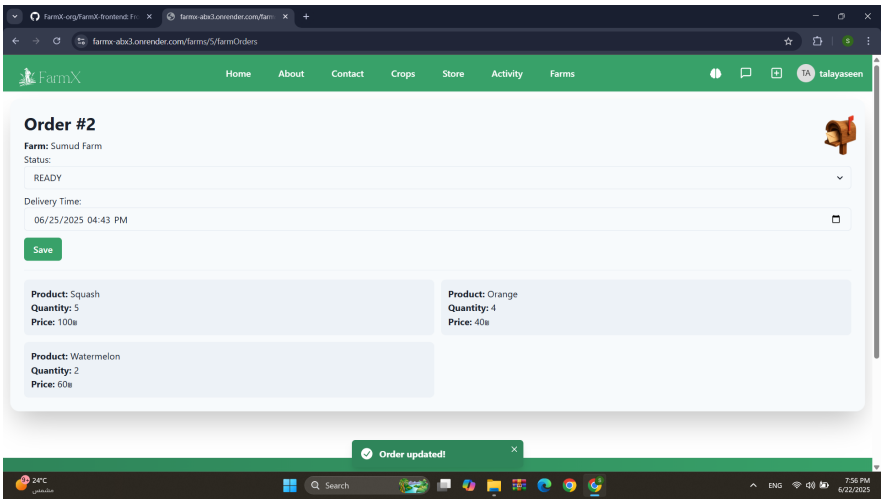

Figure 4.57: Ready Order Details

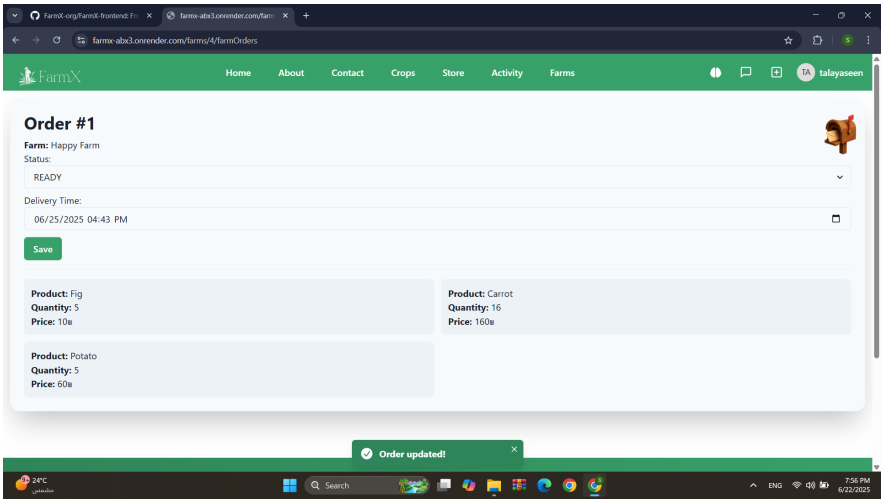

Figure 4.58: Mixed Ready and Pending Orders

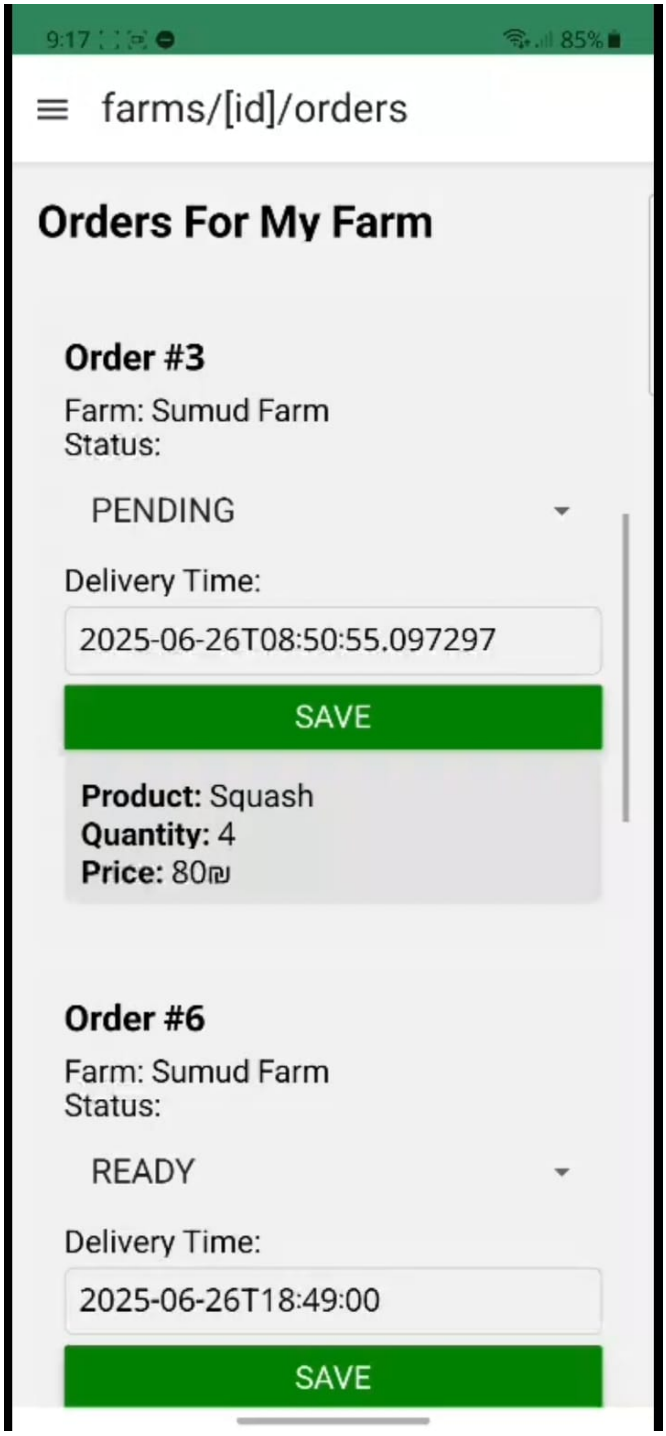

Figure 4.59: Orders for My Farm - Mobile View

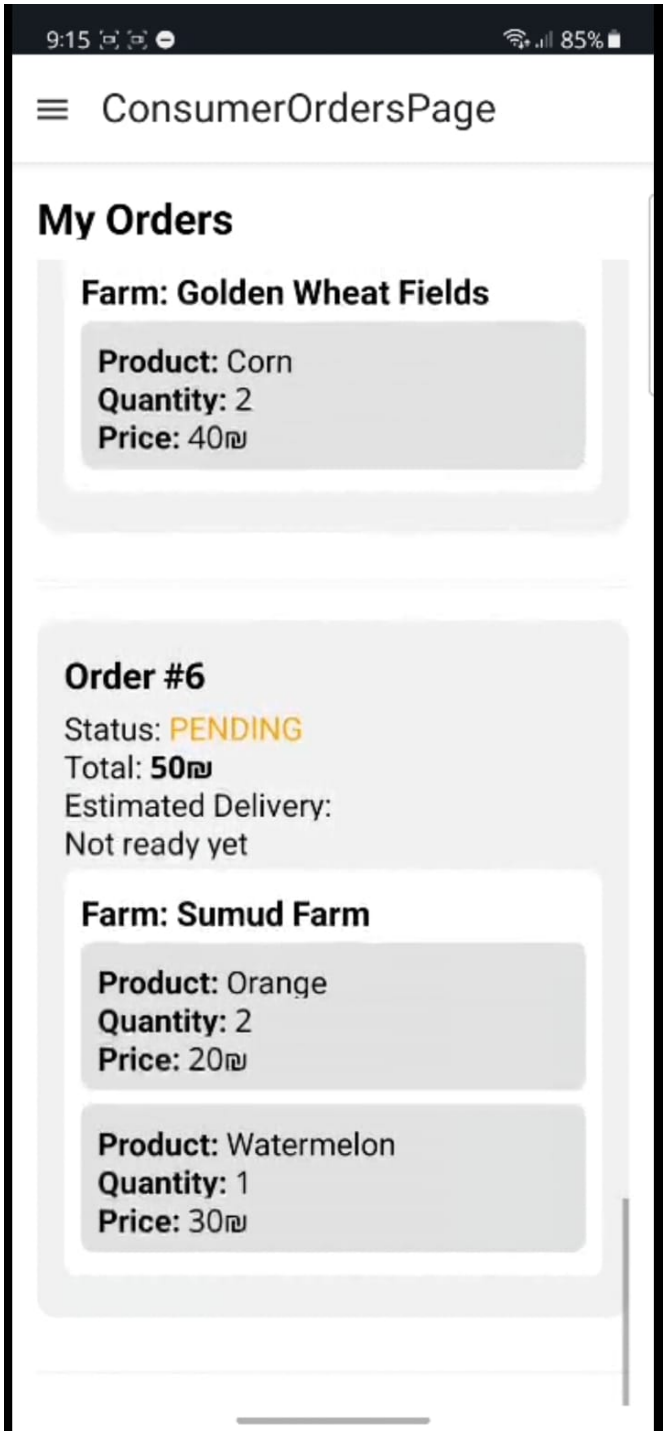

Figure 4.60: Consumer Orders - Mobile View

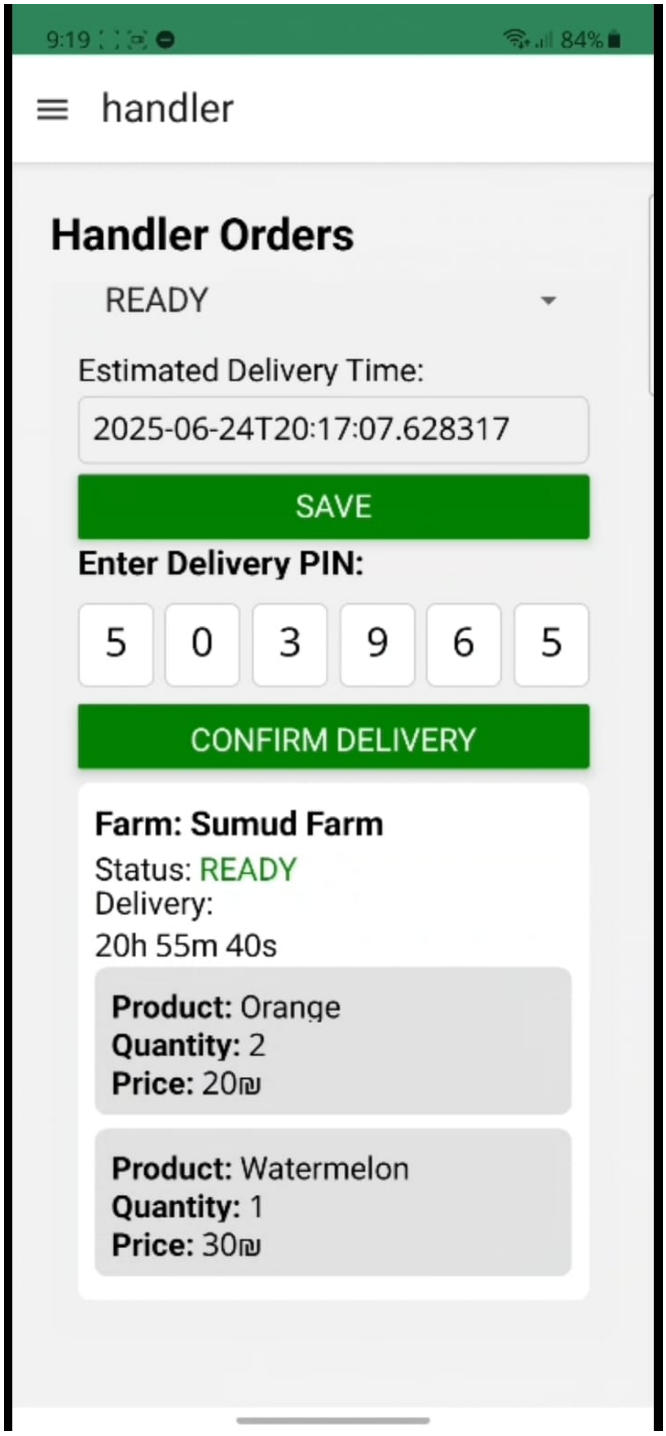

Figure 4.61: Orders Assigned to Handler - Mobile View

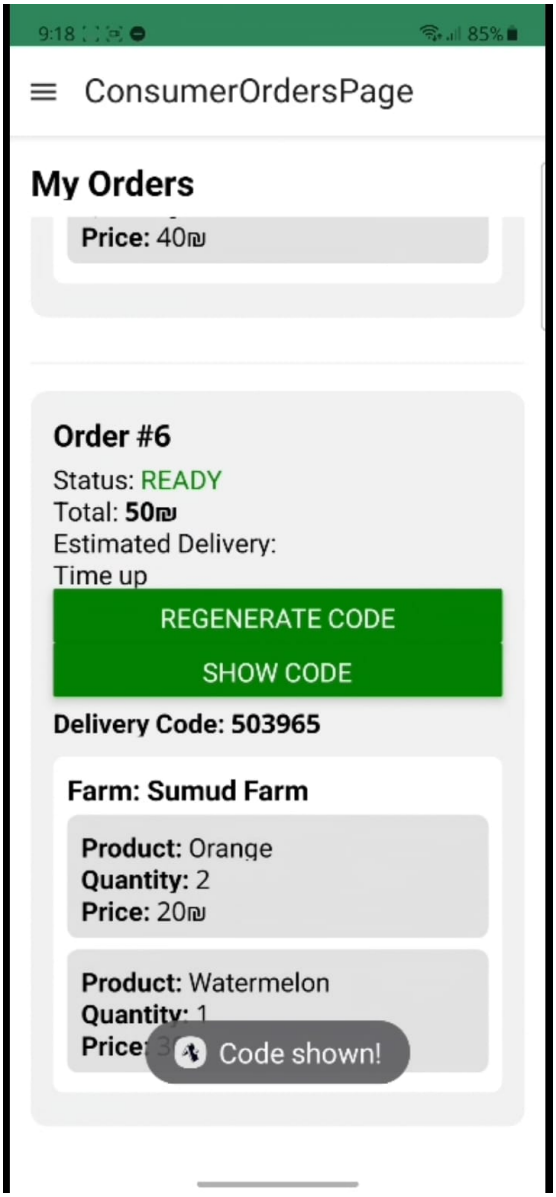

Figure 4.62: Order Code for Consumer - Mobile View

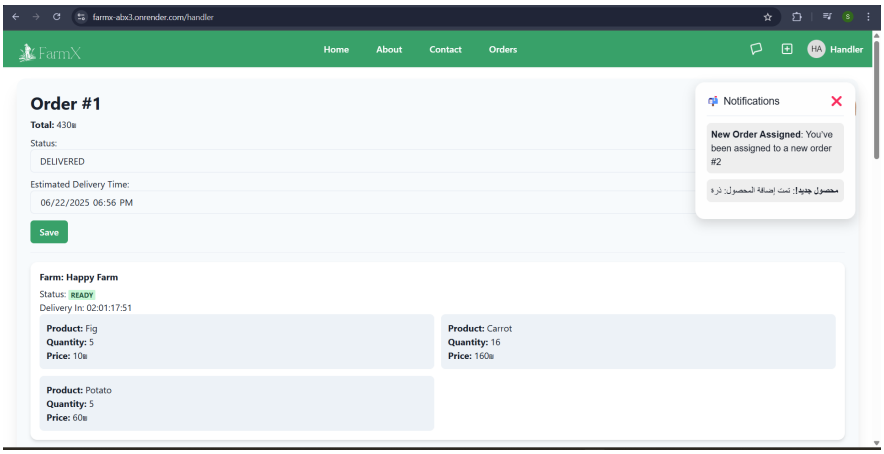

Figure 4.63: Handler Notification for Orders

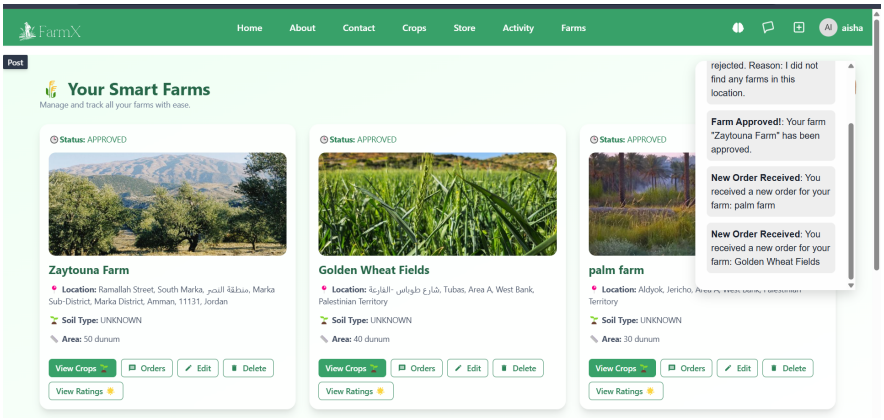

Figure 4.64: Farm Notification for Orders

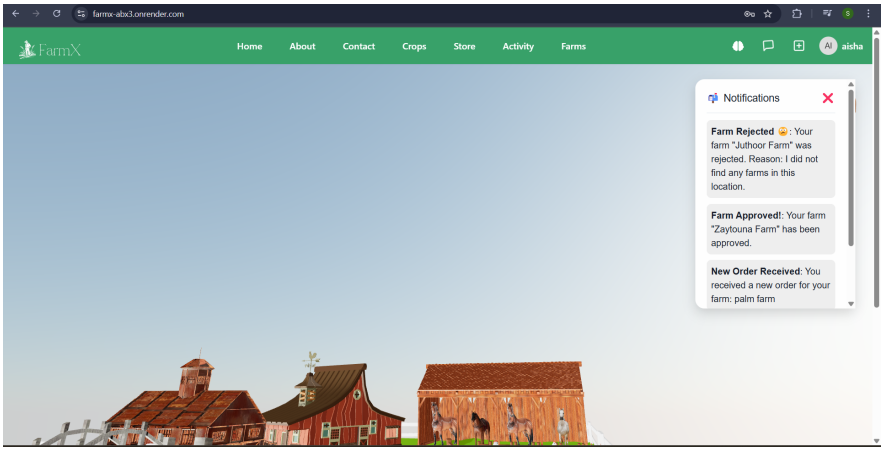

Figure 4.65: Notification for Order Approval/Rejection

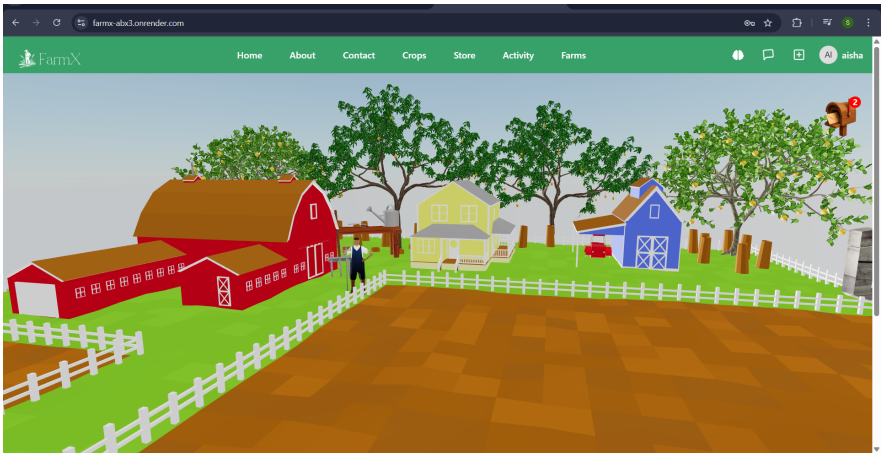

Figure 4.66: General Notification Center

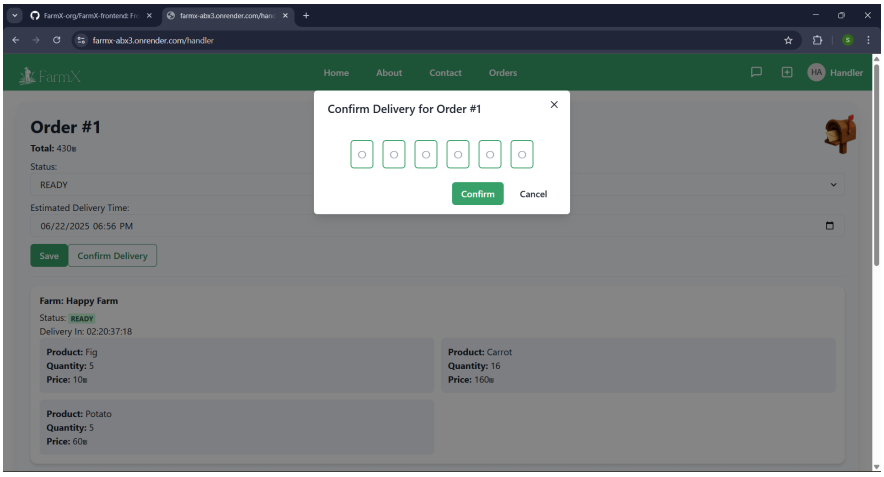

Figure 4.67: Delivery Confirmation - Code Screen 1

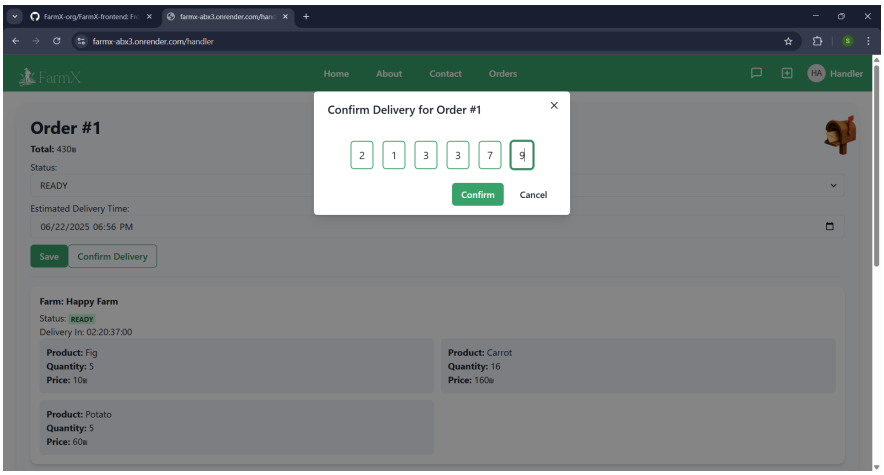

Figure 4.68: Delivery Confirmation - Code Screen 2

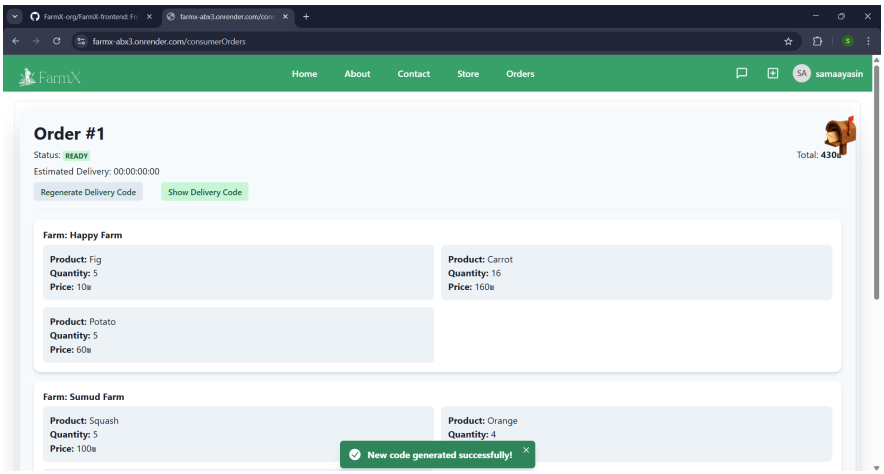

Figure 4.69: Ability to Regenerate Delivery Code

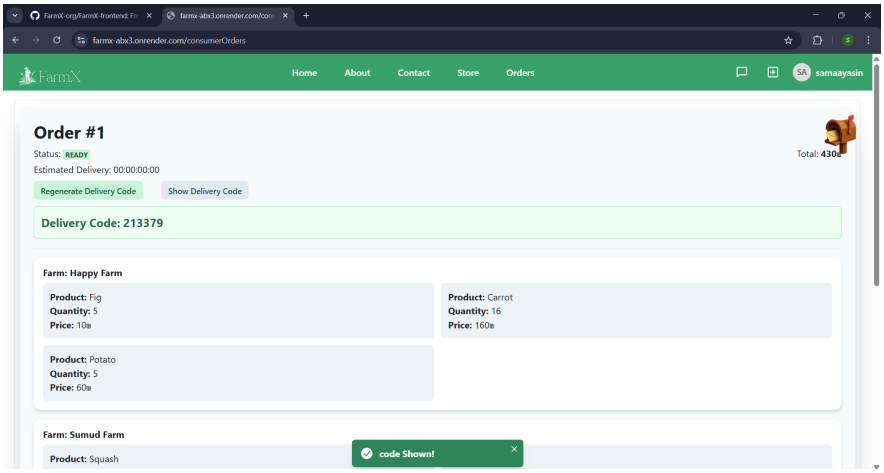

Figure 4.70: Show Delivery Code to Customer

Order Handlers & Notifications

Dedicated interfaces for order handlers to manage and track assigned orders.

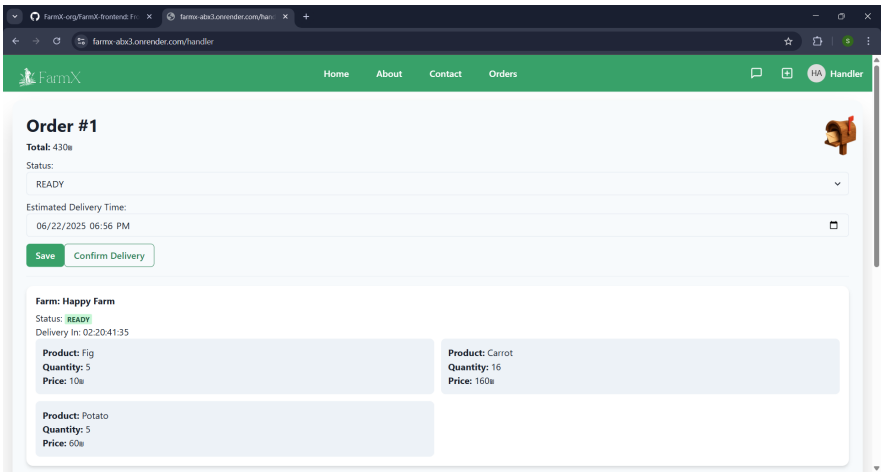

Figure 4.71: Handler Dashboard 1

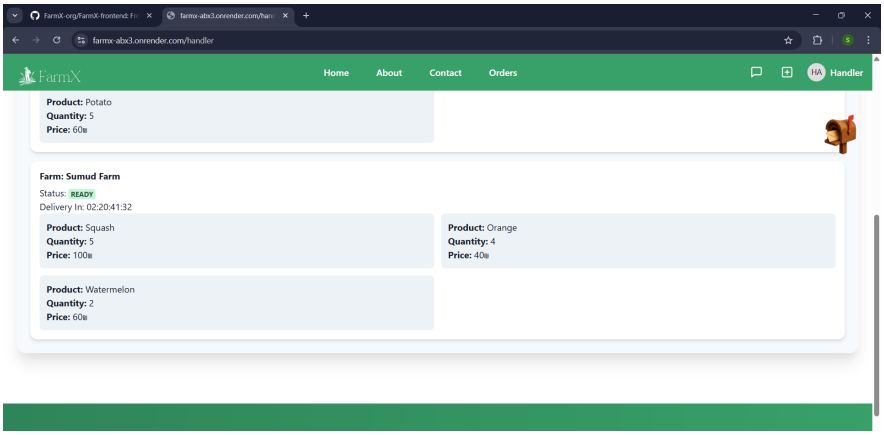

Figure 4.72: Handler Dashboard 2

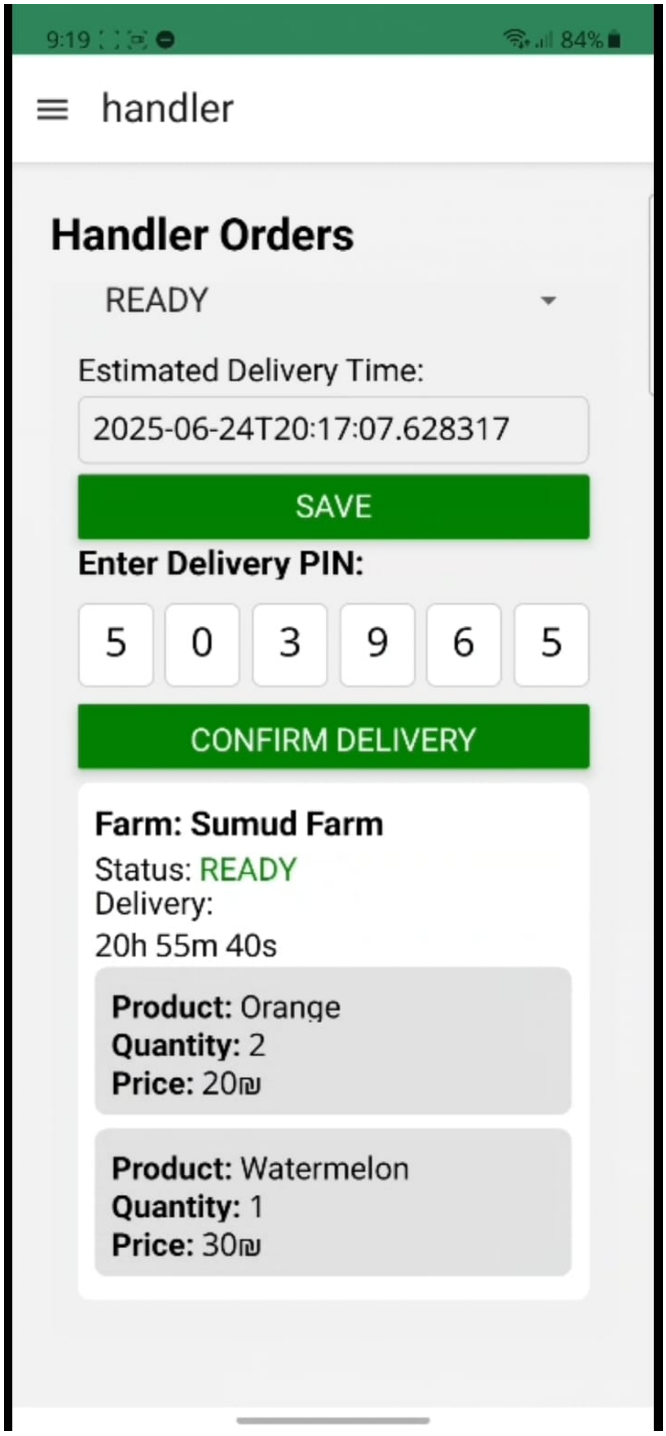

Figure 4.73: Orders Assigned to Handler - Mobile

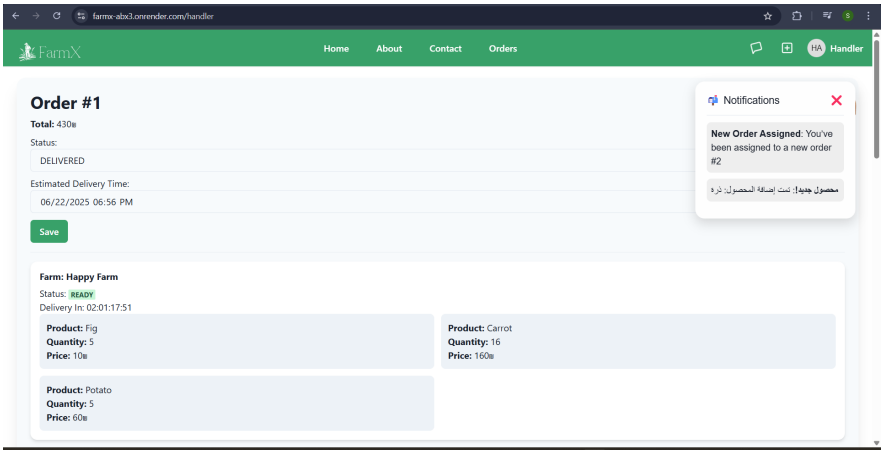

Figure 4.74: Handler Notifications

Posts, Chat & Messaging

Real-time interaction through posts and chat with instant feedback via toast messages.

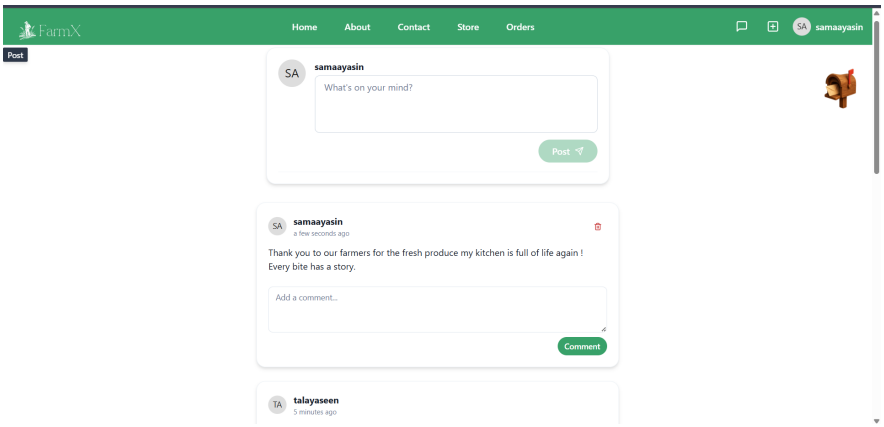

Figure 4.75: Posts and Social Feed

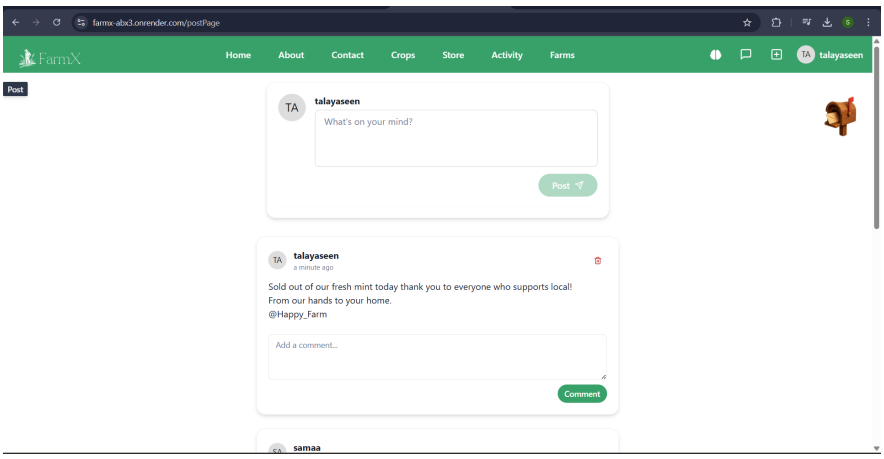

Figure 4.76: Farmer's Post Example

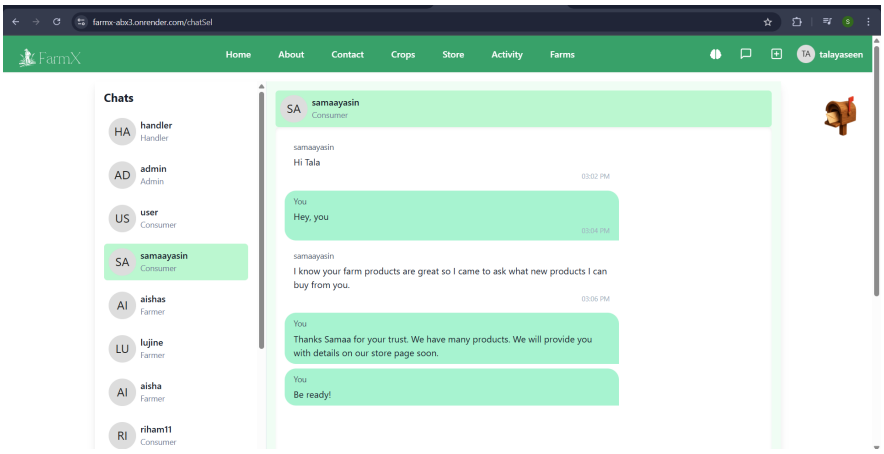

Figure 4.77: Chat Interface

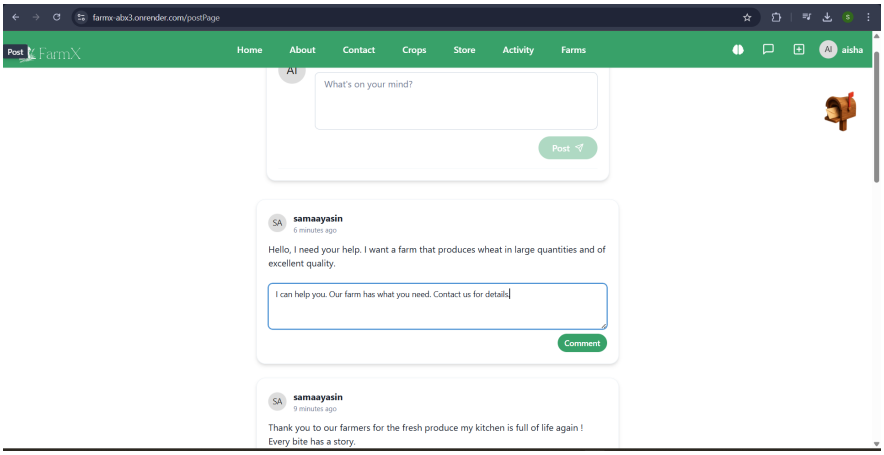

Figure 4.78: Comment Section

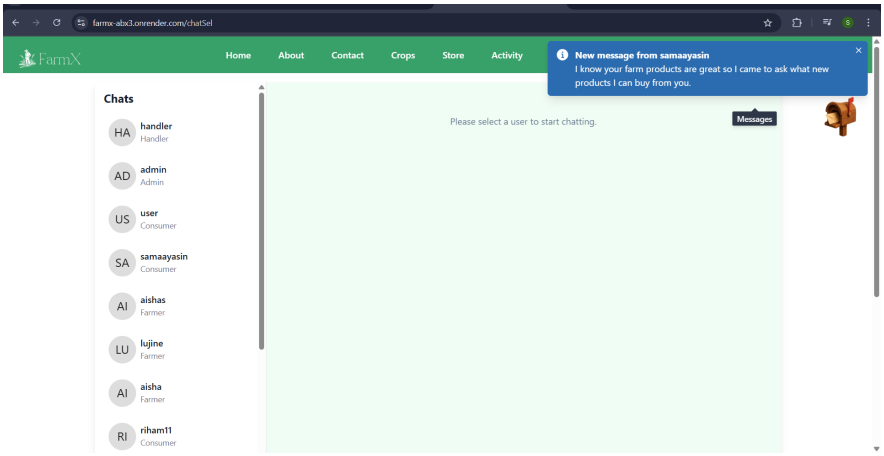

Figure 4.79: Toast Notification for Messages

Feedback & Ratings

Consumers rate products and farms, providing valuable feedback visible to admins.

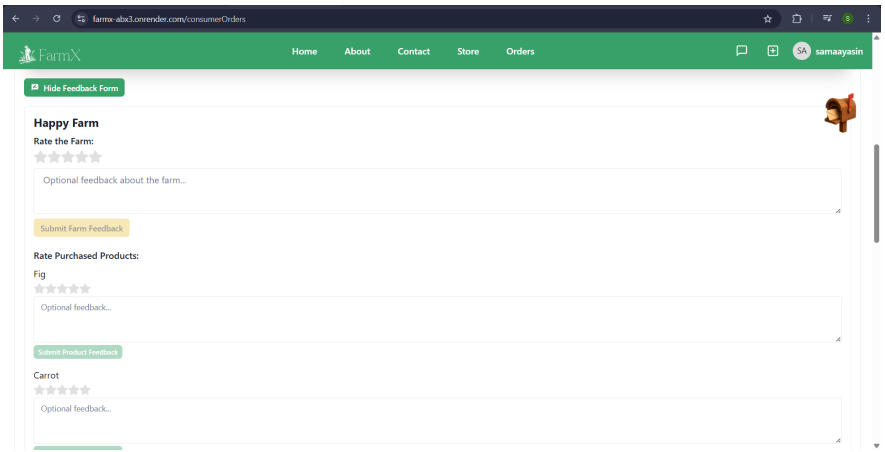

Figure 4.80: Consumer Feedback

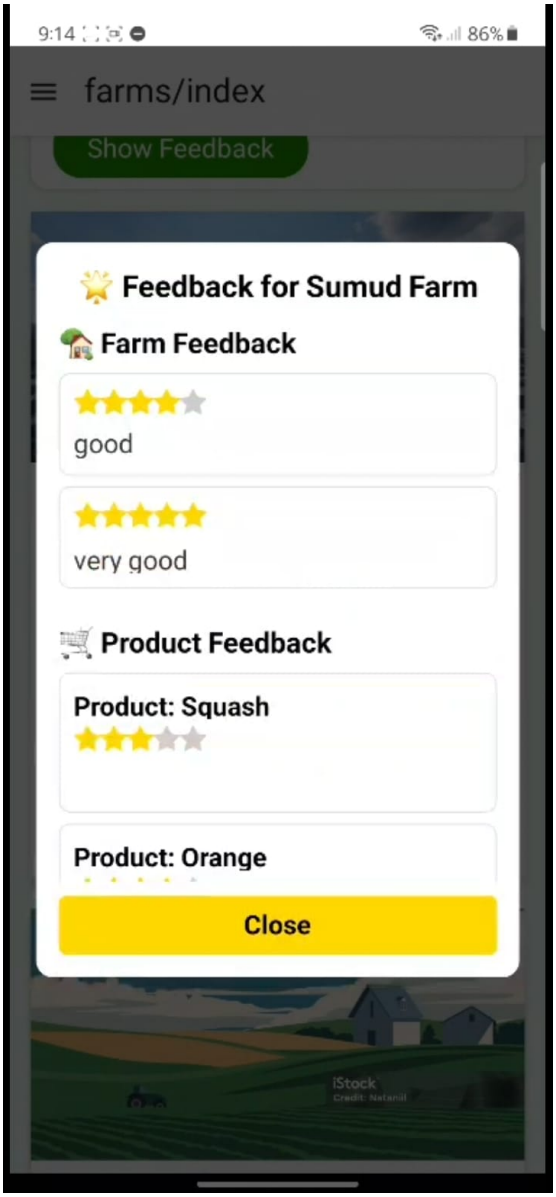

Figure 4.81: Feedback on Mobile

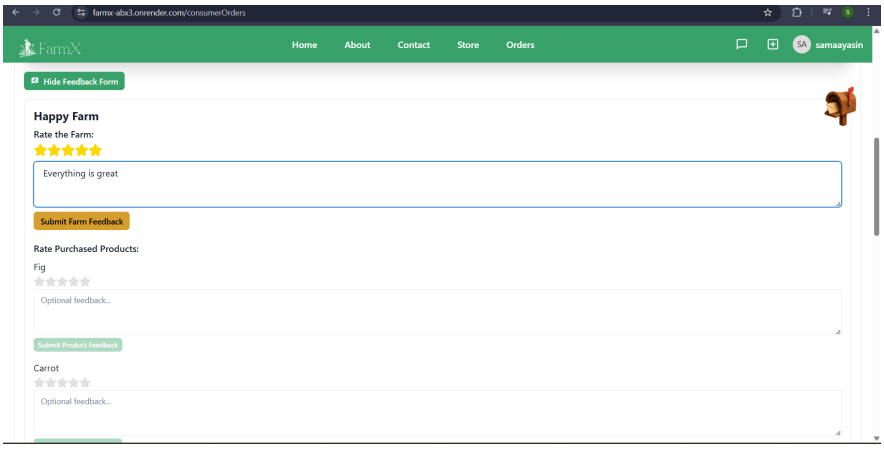

Figure 4.82: Feedback Detail 2

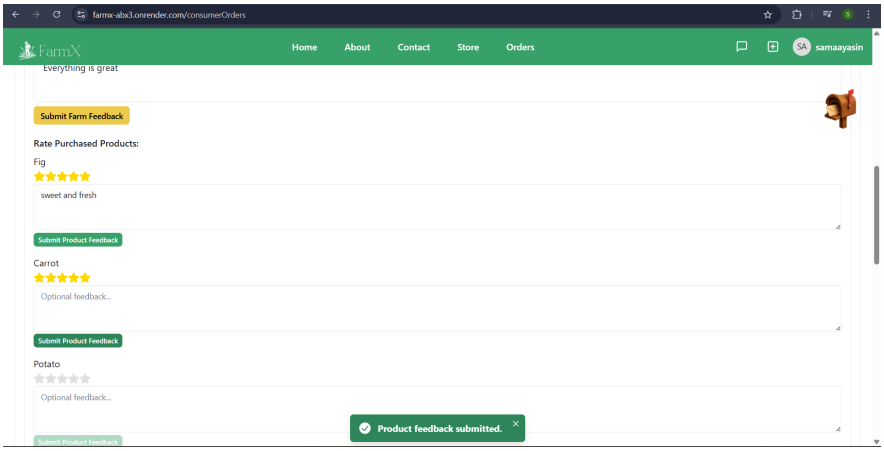

Figure 4.83: Feedback Detail 3

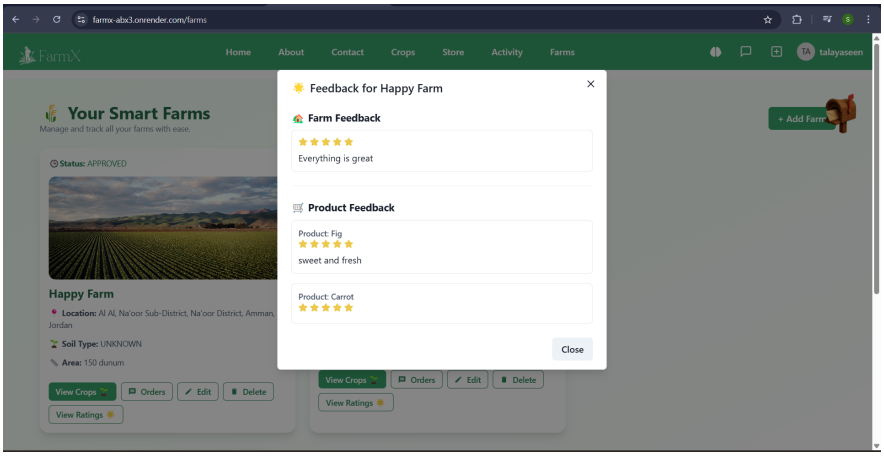

Figure 4.84: Farmer Feedback Overview

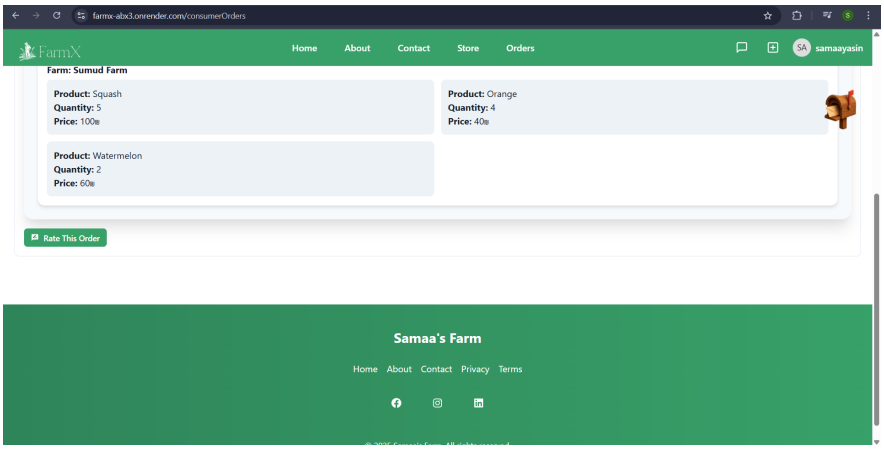

Figure 4.85: Rating Cards for Consumers

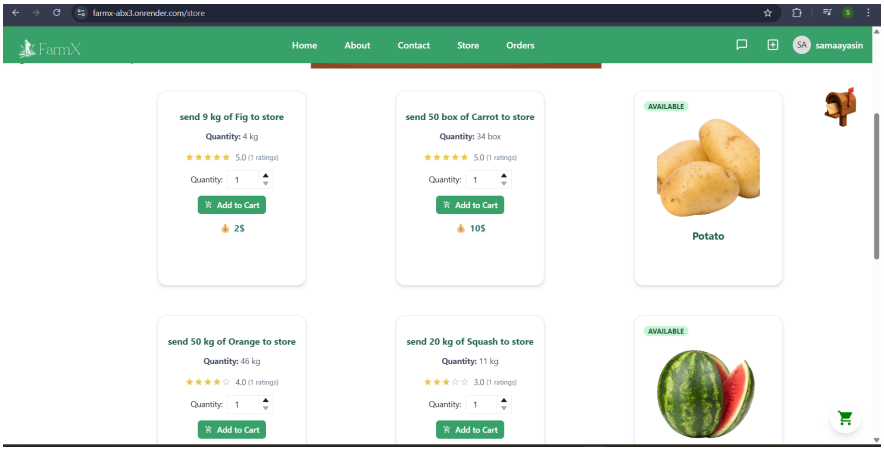

Figure 4.86: Ratings Display in Cards

Navigation & Miscellaneous

Various navigation bars and utility screens.

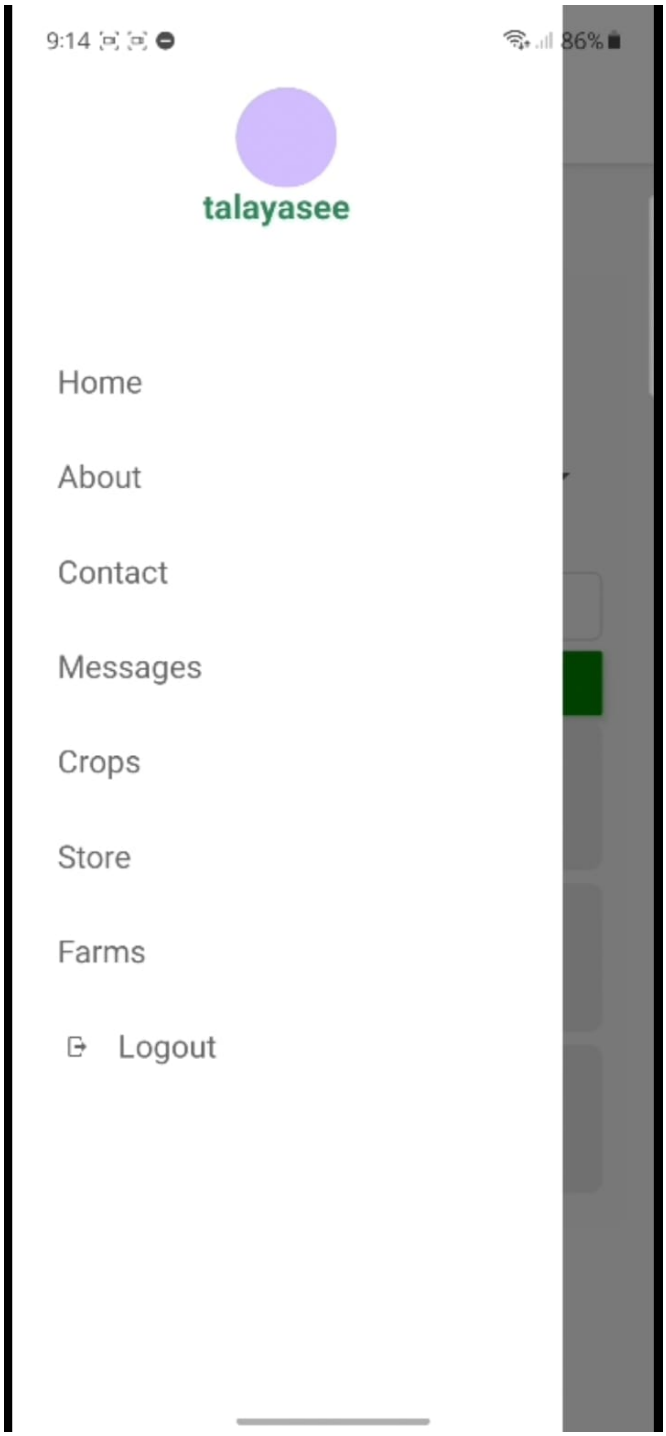

Figure 4.87: Mobile Navigation Bar

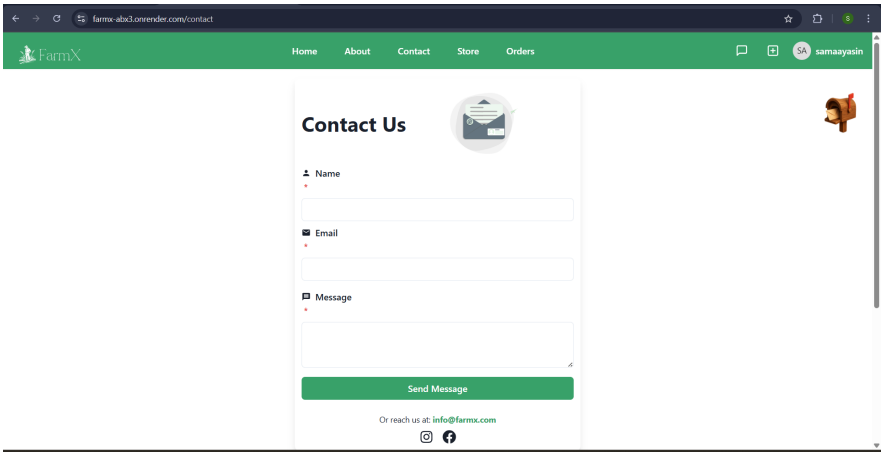

Figure 4.88: Contact Screen

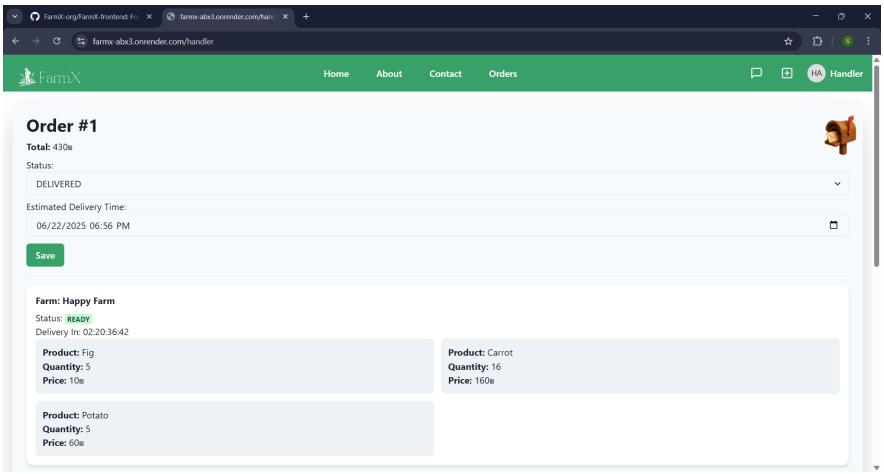

Figure 4.89: Delivery Status Screen

UI/UX Considerations

The frontend leverages Chakra UI on the web to provide a consistent and accessible design system. For the mobile app, React Native Styles ensure modular and reusable styling components. The interface is designed to be fully responsive and intuitive, offering seamless navigation and fast interactions. Toast notifications provide instant feedback to users for their actions.

Summary

The frontend development of FarmX presents a comprehensive, user-friendly experience across platforms, combining modern technologies and thoughtful design to empower farmers, consumers, and administrators alike.

4.1.6 Deployment and Hosting

FarmX was deployed using modern cloud-based platforms to ensure high availability, easy scalability, and a smooth CI/CD workflow. The deployment architecture involves three main components:

- **Backend API (Spring Boot)** – Deployed on **Render**.
- **Frontend (React + Next.js)** – Also deployed on **Render**.
- **Database (MySQL)** – Hosted on **Railway**.

1. Backend Deployment (Render) The backend is deployed as a Render Web Service using Docker and GitHub integration.

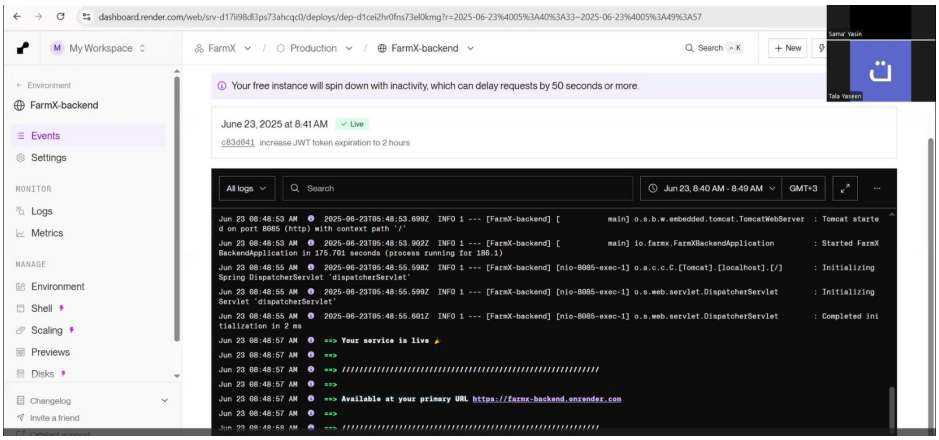

Figure 4.90: Backend deployment on Render

- **Framework:** Spring Boot (Java)
- **Start Command:** `java -jar target/.jar`
- **Build Command:** `./mvnw clean package`
- **Auto Deploy:** Triggered on GitHub push to `main`

2. Frontend Deployment (Render) The frontend is built with React and Next.js, deployed as a static site.

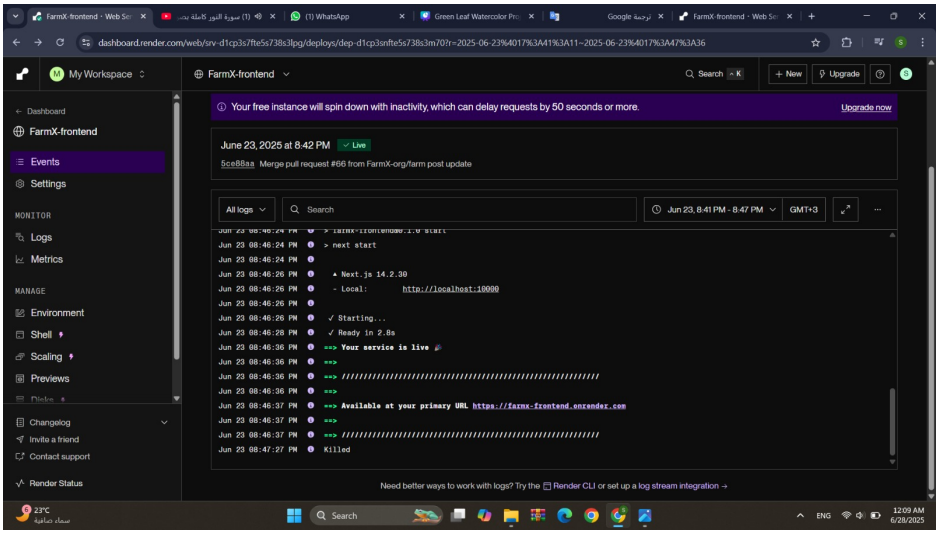

Figure 4.91: Frontend deployment on Render

- **Framework:** Next.js (React)
- **Build Command:** `npm run build`
- **Publish Directory:** `.next`
- **Environment:** Includes backend URL and Firebase config

3. Database Hosting (Railway) The MySQL database is hosted on Railway with connection credentials shared via environment variables in both frontend and backend.

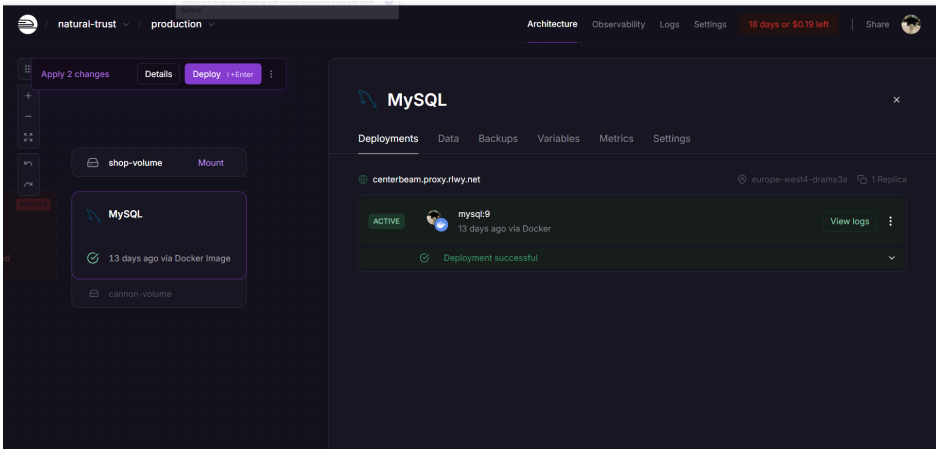

Figure 4.92: MySQL database hosted on Railway

- **Database:** MySQL 8
- **Security:** SSL, private credentials
- **Usage:** Spring Boot connects via JDBC

4. Environment Variables Sensitive information is managed using platform-specific environment dashboards:

- `SPRING_DATASOURCE_URL`, `JWT_SECRET`, `FIREBASE_CREDENTIALS`
- `NEXT_PUBLIC_BACKEND_URL`, `NEXT_PUBLIC_FIREBASE_CONFIG`

5. Summary This cloud-based deployment strategy ensures high uptime, fast delivery, and maintainability of the FarmX application across different platforms and roles.

Chapter 5

Literature Review

The development of FarmX was influenced by a wide range of research related to agriculture technology, user-centric design, and real-time systems.

Several studies have explored the role of digital platforms in transforming agricultural operations. Farm management systems such as AgriWebb and Cropio have shown how digitization can enhance productivity, but they often lack real-time features and localized support for specific communities. Research highlights that integrating soil data and local climate information can greatly improve crop selection and yield optimization.

Real-time communication technologies like Firebase have gained attention for enabling live updates and interactions in mobile and web apps. Studies show that these tools enhance user engagement, particularly in social and commerce platforms. Firebase Cloud Messaging (FCM) offers scalable, cross-platform push notifications critical for modern systems like FarmX.

Research on UI/UX design emphasizes the importance of accessibility and responsiveness in user interfaces, particularly in systems targeting users with varying technical skills. Tools like React Native and Expo simplify the development of performant mobile applications, while frameworks like Chakra UI contribute to consistent and accessible design in web interfaces.

Studies also highlight that the combination of 3D visualization (such as Three.js) with functional data displays increases user immersion and understanding, especially in educational and geographical applications. This insight shaped FarmX's 3D homepage and crop visualization features.

Furthermore, publications from the Palestinian Central Bureau of Statistics (PCBS) provided foundational data on agricultural practices, crop distribution, and regional farming challenges.

These insights ensured the system's design was tailored to the local ecosystem.

In summary, existing research underscores the potential of integrating modern web technologies, real-time communication, and interactive design to empower agricultural communities — a vision that FarmX brings to life for the Palestinian farming context.

Chapter 6

Results and Discussion

The implementation of FarmX has demonstrated promising results across both technical and user experience dimensions.

System Performance

FarmX achieved a reliable and responsive performance under realistic usage scenarios. Backend endpoints responded with low latency, and Firebase-based notifications delivered messages instantly. The system maintained consistent behavior across different user roles.

User Interaction

User feedback from testing sessions with farmers and consumers showed high satisfaction, especially regarding the simplicity of navigation, real-time notifications, and mobile access. Farmers appreciated features like crop tracking and order management, while consumers found the unified store intuitive and efficient.

Real-time Features

The use of Firebase for chat and notifications provided seamless real-time communication between users. Notifications were successfully used for order updates, delivery confirmations, and social interactions.

Visualization

The 3D homepage received positive attention for its innovative approach to engaging users visually. While it posed performance challenges on low-end devices, optimizations allowed acceptable performance across most modern devices.

Admin & Security

Spring Security with JWT tokens ensured robust authentication and authorization across roles. Admins were able to moderate users, farms, and feedback effectively, thanks to the centralized dashboard.

Challenges

- Occasional delays during Firebase sync on weak network connections.
- Mobile responsiveness required fine-tuning for different screen sizes.
- Some legacy devices had trouble rendering 3D content smoothly.

Chapter 7

Conclusion and Future Work

Conclusion

FarmX has shown that it is possible to digitize and modernize the agricultural process in Palestine using scalable and user-centric technologies. The platform serves multiple stakeholders—farmers, consumers, admins, and handlers—by offering tools to manage farms, visualize crop data, handle orders, and communicate in real time. It bridges the digital divide for rural agricultural communities while promoting transparency, efficiency, and sustainability in the agricultural supply chain.

By integrating features such as role-based access, real-time communication, 3D visual interfaces, and responsive mobile design, FarmX has established itself as a foundational platform for modern farming systems tailored to the Palestinian context.

Future Work

Several enhancements can further elevate the FarmX platform:

- **AI-based Crop Health Monitoring:** Use satellite imagery and AI models to detect crop diseases, suggest treatments, and prevent losses.
- **Livestock Management:** Extend the platform to support animal husbandry, including health tracking, feeding schedules, and breeding management.
- **Offline Functionality:** Implement offline data entry and syncing to support remote areas with poor internet connectivity.

- **Arabic NLP Assistance:** Add Arabic-language support and AI-based chatbots for easier access and self-service help.
- **Integration with Local Agricultural Databases:** Collaborate with Palestinian institutions to link real-world agricultural data and improve system intelligence.

Bibliographic

[1] Google Firebase. *Firebase Documentation*. `https://firebase.google.com/docs`. [Online]. 2024.

[2] Pivotal Software. *Spring Boot Reference Documentation*. `https://docs.spring.io/spring-boot/docs/current/reference/html/`. [Online]. 2024.

[3] Meta Open Source. *React – A JavaScript library for building user interfaces*. `https://reactjs.org`. [Online]. 2024.

[4] Three.js Documentation. *JavaScript 3D library*. `https://threejs.org/docs/`. [Online]. 2024.

[5] Chakra UI. *Simple, Modular and Accessible UI Components for React*. `https://chakra-ui.com`. [Online]. 2024.

[6] ISRIC. *SoilGrids — Global gridded soil information*. `https://soilgrids.org`. [Online]. 2024.

[7] Palestinian Central Bureau of Statistics (PCBS). *Agricultural Statistics for Palestinian Territories*. `https://www.pcbs.gov.ps`. [Online]. 2024.

[8] S. Sladojevic et al. "Deep Neural Networks Based Recognition of Plant Diseases by Leaf Image Classification". In: *Computational Intelligence and Neuroscience* (2016).

[9] M. Ayaz et al. "Internet-of-Things (IoT)-Based Smart Agriculture: Toward Making the Fields Talk". In: *IEEE Access* 7 (2019), pp. 129551–129583.

[10] R. Tsouros et al. "A Review on UAV-Based Applications for Precision Agriculture". In: *Information* 10.11 (2019).

[11] K. Wolfert et al. "A review on the possibilities for smart livestock farming using digital technology". In: *Computers and Electronics in Agriculture* 157 (2019).

Appendix