Journal of Computer and Electrical Sciences (JCES)- Volume-1 Issue-4, April 2026
ISSN: 3049-2602

E-Farm: A Web-Based Marketplace for Direct Farmer-to-Consumer Sales

Parapathi Vasu
Student, Department of Computer Science and Engineering
Andhra Loyola Institute of Engineering and Technology (ALIET), Vijayawada
*Roll No: **22HP1A05C0***
Mrs. K. Pushpavalli
Assistant Professor, Department of Computer Science and Engineering
Andhra Loyola Institute of Engineering and Technology (ALIET), Vijayawada
Affiliated to Jawaharlal Nehru Technological University Kakinada (JNTUK)
Email ID:vasuparapathi946@gmail.com

Abstract: E-Farm (Farm Fresh Direct) is a web-based agricultural marketplace designed to connect farmers directly with consumers, eliminating intermediaries from the supply chain. The system enables farmers to register, list agricultural products, and manage their product details. Customers can browse available farm products, view product information, add items to the cart, and place orders through the platform. An admin module manages user approvals, product oversight, and platform monitoring. The application is built using React, TypeScript, Vite, and Tailwind CSS for the frontend, with Supabase as the backend service for authentication, database management, and API integration using PostgreSQL. Results from functional and system testing confirm correct user authentication, accurate product management, successful order processing, and a smooth, responsive user experience across all modules.

Keywords: E-Farming, Agricultural Marketplace, React, Supabase, PostgreSQL, Role-Based Access Control, Farm-to-Consumer, Direct Sales Platform

1.INTRODUCTION

Agriculture plays a vital role in the economy and is the primary source of livelihood for many people. Farmers produce a variety of crops, vegetables, and fruits, but they often face challenges in selling their products at fair prices. In traditional markets, farmers usually depend on middlemen or intermediaries to sell their produce. This process reduces farmers' profits and increases the price for consumers.

With the advancement of technology and internet services, digital platforms have become an effective way to connect farmers directly with consumers. An online farming platform allows farmers to list their products and sell them directly to customers without the involvement of middlemen. This approach helps farmers receive better prices for their produce while allowing consumers to purchase fresh agricultural products at reasonable prices.

The E-Farm (Farm Fresh Direct) system is designed to create a digital marketplace where farmers can showcase and sell their products directly to customers. The system enables farmers to add product details such as name, price, quantity, and description. Customers can browse available products, view product information, add items to their cart, and place orders through the platform. The application is developed using modern web technologies — React, TypeScript, and Vite for the frontend — with Supabase as the backend handling authentication, database storage, and API services using PostgreSQL.

Motivation

Many farmers face difficulties in selling their products at fair prices due to the involvement of middlemen. These intermediaries often reduce the profits earned by farmers and increase the cost for customers. With the growth of digital technology, online platforms can help farmers connect directly with consumers. The motivation of this project is to create a digital platform that simplifies the buying and selling of farm products, helping farmers increase their income and providing customers with fresh products. By using modern technologies like React and Supabase, the platform becomes efficient and user-friendly.

Problem Statement

In the traditional agricultural market system, farmers depend on local markets or middlemen to sell their products. This often results in lower profits for farmers and higher prices for consumers. Farmers also face challenges in reaching a larger customer base due to the lack of proper digital platforms. Customers may not always get fresh products directly from farms. Therefore, there is a need for a system that connects farmers and consumers directly. The main objective of this project is to develop an E-Farming web application that allows farmers to sell their products online, manage

https://doi.org/10.5281/zenodo.19598288

listings, and interact with customers — while customers can browse products, add items to the cart, and place orders easily.

2. LITERATURE SURVEY

The development of digital technology has significantly influenced many sectors, including agriculture. Traditional agricultural marketing systems often rely on intermediaries, which reduces the profit earned by farmers and increases the cost for consumers. With the increasing use of the internet and online platforms, many researchers and developers have explored digital solutions to connect farmers directly with customers. Online agricultural marketplaces help farmers display their products and allow consumers to purchase fresh farm products conveniently.

Modern web technologies have played an important role in developing efficient agricultural platforms. Frameworks such as React are widely used to create responsive and user-friendly interfaces. Backend services like Supabase provide powerful tools for handling database management, authentication, and APIs. Databases such as PostgreSQL are used to store product details, user accounts, and transaction information securely.

Table 1 provides a comparative summary of related works and the positioning of the proposed E-Farm system:

Table 1: Comparison of Related Work

Author / Source	Method	Contribution	Limitation
e-Choupal (ITC Initiative)	Internet Kiosks	Provides real-time commodity prices; improved farmer income by 15-20%	Targets wholesale buyers, not direct consumers
eNAM (Govt. of India)	Digital Mandi Auction	Digitized auction process across 22 states for 150+ commodities	Preserves aspects of intermediary structure
BigBasket / Ninjacart	Farm-to-Retailer Model	Demonstrated viable direct commerce at scale	Capital-intensive; not open/accessible to small developers
Proposed System (E-Farm)	React + Supabase Web App	Direct farmer-to-consumer marketplace with role-based access control	Requires internet connectivity; payment gateway integration pending

3. SYSTEM ANALYSIS

System Analysis is the process of studying and understanding the existing system and identifying the requirements for developing a new system. It helps in analysing the problems in the current system and designing a better solution.

Project SDLC

The Software Development Life Cycle (SDLC) is a process used for planning, creating, testing, and deploying software systems. It ensures that the software is developed systematically and meets the requirements of users. The main stages of SDLC followed in this project include: Requirement Analysis, System Design, Implementation, Testing, Deployment, and Maintenance.

Fig. 3.1 illustrates the SDLC model adopted for this project. These stages help in developing the system in an organized and efficient manner.

Stage 1 — Planning and Requirement Analysis: The requirements of the E-Farm system are collected and analysed. Key features such as user registration, product listing, and order management are identified. These requirements help define the overall structure of the system.

Stage 2 — Defining Requirements: After analysing the problem, the system requirements are clearly defined. Farmers must be able to add and manage their products, while customers can browse and purchase products online. The system must also provide secure login and data storage.

Stage 3 — Designing the Software: The architecture and structure of the system are designed. The system includes modules like authentication, product management, cart, and order processing. The frontend is designed using React and the backend services are handled using Supabase.

https://doi.org/10.5281/zenodo.19598288

Stage 4 — Developing the Project: The frontend is built using React, TypeScript, and Tailwind CSS. The backend and database operations are managed using Supabase with PostgreSQL.

Stage 5 — Testing: Different types of testing such as functional testing and system testing are performed. This verifies whether farmers can add products and customers can place orders successfully.

Stage 6 — Deployment: After the system is successfully tested, it is deployed so that farmers and customers can access the platform through a web browser.

Stage 7 — Maintenance: Maintenance involves updating and improving the system after deployment. New features can also be added. This stage ensures that the E-Farm platform remains reliable and efficient.

Functional Requirements

Functional requirements describe the main functions that the system must perform:

- The system should allow users to register and log in to the platform.
- The system should allow farmers to add, update, and delete product information.
- The system should allow customers to view available farm products.
- The system should allow customers to add products to the cart and place orders.
- The system should provide an admin module to manage users and approve products.
- The system should store and manage user and product data in the database.

Non-Functional Requirements

Usability: The system should be easy to use and provide a simple interface for farmers and customers.

Reliability: The system should work consistently without errors and ensure that product and order information is stored correctly.

Performance: The system should load quickly and allow users to browse products and place orders without delays.

Security: The system should protect user accounts and data through secure authentication and row-level security policies.

Maintainability: The system should be easy to update and maintain so that new features can be added in the future.

System Requirements

Software Requirements — Operating System: Windows 10/11 or Linux. Programming Language: TypeScript / JavaScript. Frontend Framework: React with Vite. Styling: Tailwind CSS. Backend Service: Supabase. Database: PostgreSQL. IDE: Visual Studio Code.

Hardware Requirements — Processor: Intel Core i3 or higher. RAM: 4 GB minimum (8 GB recommended). Storage: 10 GB or more free disk space. Network: Internet or local network access. Device: Desktop, laptop, or smartphone with a compatible browser.

RELATED WORK

Existing System

In the traditional agricultural marketing system, farmers usually sell their products through local markets or intermediaries. Farmers depend on middlemen to distribute their agricultural products to customers. Because of this process, farmers often receive lower prices for their produce while consumers have to pay higher prices. In many rural areas, farmers do not have a proper digital platform to directly connect with customers. Some existing online systems provide basic product listings but lack proper management of product details, orders, and user accounts.

Disadvantages of the existing system include:

- Farmers depend heavily on middlemen to sell their products.
- Farmers often receive lower profits for their agricultural produce.
- Limited market access and reach for farmers.
- Customers may not always get fresh products directly from farms.
- The traditional system lacks an efficient digital platform.

https://doi.org/10.5281/zenodo.19598288

Proposed System

The proposed system introduces an E-Farm web application that connects farmers directly with consumers through an online platform. This system allows farmers to add and manage their agricultural products such as vegetables, fruits, and other farm produce. Customers can browse available products, view product details, and purchase products directly through the platform. The system is developed using React for the frontend and Supabase for backend services and database management.

Advantages of the proposed system include:

- Farmers can sell their products directly to customers without middlemen.
- The system helps farmers reach a larger number of customers through the online platform.
- Customers can easily browse and purchase fresh farm products from farmers.
- The platform provides better transparency in pricing and product information.
- The system offers a simple and user-friendly interface for both farmers and customers.
- It helps increase farmers' profits and improve the efficiency of agricultural product distribution.

Table 2: Proposed vs. Existing Systems

Feature	Existing Systems	E-Farm (Proposed)
Data Entry	Manual / Offline	Digital Forms (Web-Based)
Product Listing	Basic or None	Full product management with approval flow
Order Management	Not Available	Cart, checkout, and order history
Admin Control	Limited or None	Admin dashboard with user/product approval
Authentication	None / Insecure	JWT-based session with Supabase Auth
Personalization	Generic	Role-based dashboards for farmers, buyers, admin

CODING

The coding phase is an important stage in software development where the system design is converted into a working application. In this project, the E-Farm system is implemented as a web-based application using modern web technologies. The implementation is divided into two main parts: Frontend and Backend.

Frontend

Frontend Technologies: React, TypeScript, Vite, Tailwind CSS, shadcn-ui / Radix UI components, React Router.

HTML (index.html): Defines the application entry point with SEO meta tags, Open Graph tags, and Twitter Card metadata.

The root HTML file sets up the document structure, loads the React application via the /src/main.tsx module entry point, and declares application title and description metadata for E-Farm.

Main.tsx: Mounts the React application onto the DOM root element and imports global styles.

The application root renders the App component which contains the routing configuration using React Router, providing navigation between the Home, Products, Cart, Checkout, Login, Register, Farmer Dashboard, and Admin Dashboard pages.

Admin Dashboard: Implements the administrative interface for managing farmers, products, and orders.

https://doi.org/10.5281/zenodo.19598288

The Admin Dashboard fetches farmer profiles, products, and orders from Supabase. Administrators can approve or reject farmer registrations, approve or reject product listings, edit product and farmer details, and delete products. The dashboard uses Tabs to organize content by Farmers, Products, and Orders sections, with card-based summary metrics showing pending counts.

Register Page: Implements dual-role registration for Farmer and Buyer accounts.

The registration form collects full name, email, phone, location, and password. Users select their role (Farmer or Buyer) before submitting. Supabase Auth handles account creation, with a database trigger creating a corresponding profiles table entry with the selected role.

Products Page (Products.tsx): Displays the product catalogue with filtering and sorting controls.

Active and approved products are fetched from Supabase. Users can search by name or farmer, filter by category (vegetables, fruits, grains, dairy, poultry), filter by location, and sort by popularity, price (low to high, high to low), or rating. Products are displayed in a responsive grid layout.

Cart Page (Cart.tsx): Manages the shopping cart and order summary.

Cart state is persisted in the Supabase cart_items table for cross-device synchronization. Users can update quantities, remove individual items, or clear the cart. The order summary panel displays subtotal, delivery charges, and total, with a Proceed to Checkout action.

Index.css: Defines the E-Farm design system using CSS custom properties.The CSS establishes a nature-inspired colour palette using HSL variables: primary deep forest green (#2D5A3D), secondary warm wheat, accent fresh lime, and cream background. Custom utilities include gradient classes (hero, nature, wheat), shadow tokens, blob shapes, leaf patterns, and animation keyframes (float, grow, slideUp, pulse-soft) for UI interactions.

Backend

Backend / Database Technologies: Supabase, PostgreSQL.

Supabase Client (supabaseClient.ts): Initializes the Supabase client with the project URL and anonymous public key using the @supabase/supabase-js SDK.

Row-Level Security (RLS) Policies: PostgreSQL RLS policies enforce role-based data access at the database tier.

Key policies include: (1) Buyers can only insert order items for their own orders; (2) Farmers can view, create, and update their own products only; (3) Only admins can delete products; (4) Admins can view and update all profiles, products, orders, and user roles; (5) Approved and active products are visible to all authenticated users. Additional database constraints enforce price positivity (price > 0), non-negative quantity, and length limits on name (200 chars) and description (2000 chars) fields.

TEST CASES

Testing is an important phase in software development used to verify whether the system works correctly according to the specified requirements. It helps identify errors, bugs, and performance issues in the system. In this project, testing is performed to ensure that the E-Farm system functions properly and allows farmers and customers to interact with the platform without errors.

Objectives of Testing

- To verify the correctness of the E-Farm system.
- To check whether the system correctly accepts user inputs such as registration details and product information.
- To ensure that farmers can add products and customers can browse and purchase products successfully.
- To identify and fix errors in the application.
- To improve the reliability and performance of the web application.

https://doi.org/10.5281/zenodo.19598288

Types of Testing

Levels of Testing

Unit Testing: Unit testing is performed to verify the individual components of the E-Farm system. Each module such as user authentication, product management, and order processing is tested separately.

Integration Testing: Integration testing checks whether different modules of the system work correctly when combined together. Modules such as user login, farmer product upload, product browsing, and order placement are integrated and tested.

System Testing: System testing verifies the complete functionality of the E-Farm application including user registration, login, product management, product approval by admin, browsing products, and order processing.

User Acceptance Testing: User acceptance testing ensures that the system meets the requirements of farmers, customers, and administrators, confirming that the system works as expected in a real-world environment.

Table 3: System Test Cases — Key Scenarios

Test Case	Input	Expected Output	Result
User Registration	Name, email, role, password	Account created; redirect to login	Pass
Farmer Login	Valid farmer credentials	Redirect to Farmer Dashboard	Pass
Add Product	Name, price, category, image	Product listed; pending admin approval	Pass
Admin Approve Product	Product ID	Product visible to customers	Pass
Add to Cart	Product selection	Item added to cart_items table	Pass
Place Order	Cart items, delivery address	Order created; cart cleared; stock decremented	Pass
Admin View Orders	Admin credentials	All orders displayed with buyer details	Pass

All test cases produced correct results. No critical bugs were found during the testing phase. The system correctly manages user authentication, product management, order processing, and administrative control across all modules.

4. CONCLUSION

This project presented E-Farm (Farm Fresh Direct), a web-based agricultural marketplace developed using React, TypeScript, Supabase, and PostgreSQL. The system successfully connects farmers directly with consumers through a digital platform, eliminating intermediaries from the supply chain. The platform integrates user authentication, product management, shopping cart, order processing, and administrative control under a single, unified system.

System testing confirms that the platform accurately processes user data, manages product listings, processes orders, enforces role-based access control, and stores records securely. The responsive interface and intuitive design make the system accessible to both farmers and consumers.

E-Farm demonstrates how modern web technologies can be leveraged to create a transparent, efficient digital agricultural marketplace — enabling farmers to receive fairer prices for their produce while ensuring that customers can easily purchase fresh agricultural products directly from farms.

5. FUTURE SCOPE

Future enhancements to the E-Farm platform may include:

- Integration of full payment gateways (UPI, credit/debit cards, digital wallets) to support secure in-app transactions.
- Development of a Progressive Web App (PWA) with offline capability and push notifications for order status updates.
- Real-time delivery tracking so that customers can monitor their orders from the farm to their location.

https://doi.org/10.5281/zenodo.19598288

- Product rating and review systems to help customers share feedback and help farmers improve their services.
- Data analytics and AI-powered recommendation systems to suggest products to customers based on previous purchases and seasonal demand.
- Multilingual support (Telugu, Hindi) to extend accessibility to farmers from diverse linguistic backgrounds.
- Cloud deployment (AWS / Azure / GCP) with containerization for scalability and improved performance.

REFERENCES

[1] React Documentation, https://react.dev. Used for understanding the React framework and building the user interface.

[2] Supabase Documentation, https://supabase.com/docs. Used for backend services including authentication, database management, and API integration.

[3] Vite Documentation, https://vitejs.dev. Used as the development build tool for creating and running the React application efficiently.

[4] Tailwind CSS Documentation, https://tailwindcss.com/docs. Used for designing the user interface with responsive and modern styling.

[5] TypeScript Documentation, https://www.typescriptlang.org/docs. Used to implement type-safe JavaScript in the frontend application.

[6] PostgreSQL Documentation, https://www.postgresql.org/docs. Used for understanding the database structure and data management through Supabase.

[7] MDN Web Docs, https://developer.mozilla.org. Reference for HTML, CSS, and JavaScript concepts during development.

[8] Lucide Icons Documentation, https://lucide.dev. Used for implementing icons in the user interface.

[9] Trebbin, A. (2014). Linking Small Farmers to Modern Retail through Producer Organizations. Food Policy, 45, 35-44.

[10] FAO. (2019). The State of Food and Agriculture: Moving Forward on Food Loss and Waste Reduction. Food and Agriculture Organization of the United Nations, Rome.

https://doi.org/10.5281/zenodo.19598288