International Journal of Applied Review
www.multiappliedjournal.com
Online ISSN: 3107-8443
Received: 28-10-2025, Accepted: 27-11-2025, Published: 12-12-2025
Volume 1, Issue 2, 2025, Page No. 82-88

Web-based farm management system

Osuan Diamond Owosa, John Temitope Ogbiti
Department of Computer Science, Edo University Iyamho, Auchi, Nigeria

Abstract
The advancement of technology in the 21st century has significantly transformed traditional sectors, including agriculture, by introducing digital tools that improve efficiency, data management, and decision-making. However, many farming operations, particularly within academic institutions and smallholder settings in Nigeria, still rely on manual processes that hinder productivity, transparency, and collaboration. In response to this challenge, this paper presents the design and implementation of a Web-Based Farm Management System developed for Edo University Iyamho. The system aims to digitize farm operations, streamline resource tracking, and facilitate real-time data access across devices. Key features of the platform include modules for farm activity logging, inventory control, task scheduling, and reporting. Users can create accounts, log farm operations such as planting and harvesting, monitor input usage, and visualize data through interactive dashboards. An administrative panel is also provided to manage system-wide tasks and user roles. The system was built using the MERN stack—MongoDB, Express.js, React.js, and Node.js—with Google Sheets serving as a lightweight database for structured data storage and real-time updates. State management was achieved through React's useState, and the application was deployed using Vercel and Render for frontend and backend services, respectively.

Keywords: Farm management, web-base, database, inventory control

Introduction

Agriculture remains a vital sector globally, providing food, employment, and economic stability. However, the industry faces significant challenges, including inefficient traditional farming practices, fragmented data collection, poor record-keeping, and limited access to real-time information. In response to these issues, modern technological innovations are being applied to agriculture, notably through the development of Farm Management Information Systems (FMIS), which support planning, decision-making, monitoring, and documentation of farm operations.

Web-based farm management systems represent an important subset of FMIS, particularly because of their ability to operate remotely, integrate various digital tools, and offer real-time data access and decision support. Such systems leverage the internet and cloud computing to manage data related to crop growth, resource usage, disease detection, and more, offering efficiency and precision in agricultural production Cloud-based FMISs are especially useful in fragmented agricultural systems such as vegetable farms in China, where they enable the tracking and tracing of production through tools like QR codes and RFID, ensuring compliance with regulatory standards and improving transparency.

The Internet of Things (IoT) plays a pivotal role in modern smart farming, especially in integrating sensors, automation, and machine learning into web-based platforms

Literature Review

Edo University Iyamho, established in 2016, is a fast-growing institution committed to academic excellence and innovation. The Department of Computer Science, under the Faculty of Science, trains students in areas such as software development, programming, web applications, and database systems. The department promotes hands-on learning through practical projects and research, encouraging students to solve real-world problems using technology. One such area is agriculture, a vital sector in Nigeria's economy. A web-based farm management system allows farmers to plan, monitor, and manage activities such as crop production, inventory, and financial records through a central platform. Designing such systems, students gain practical skills in web development, database integration, and user interface design. The department supports innovations that align with national development goals and digital transformation, preparing students to contribute to modernising agriculture and improving food productivity through technology-driven solutions.

Research Methodology

The Prototyping Methodology was adopted for the design and development of the Web-Based Farm Management System. This approach facilitates the iterative creation of functional prototypes, allowing early user involvement and feedback, which contributes significantly to refining system requirements and improving usability. It consists of 6 stages which includes

1. **Requirement Gathering:** Through interaction with potential users such as farm managers, agricultural extension officers, and other stakeholders, core requirements such as user registration, account management, farm record tracking, transaction logging, and data access control were documented.

2. **Quick Design:** Initial sketches of the user interface were created, focusing on the homepage, dashboard, transaction pages, and settings. These visual layouts served as a guide for prototype development.

3. **Prototype Building:** A basic, functional prototype of the farm management platform was constructed using minimal features to showcase the general structure and navigation flow.

4. **User Evaluation:** Selected users were allowed to interact with the prototype. Their feedback was collected through interviews and questionnaires, identifying usability issues and missing features.

5. **Prototype Refinement:** Modifications were implemented based on user suggestions. This stage often involved multiple iterations to achieve a polished system model.

6. **Final Product Development:** With validated and refined specifications, the complete web-based system was developed. This final phase involved frontend, backend, and database integration with improved stability and performance.

Programming Languages and Tools Used

The development of the Web-Based Farm Management System was carried out using a modern full-stack web development approach. Each layer of the system utilizes carefully selected technologies to ensure performance, reliability, and ease of maintenance. The table 1 below summarizes the technologies used across different layers of the application

Table 1: Technologies used across different layers of the application

Layer	Technology Used	Description
Frontend	React.js + Vite	The frontend was developed using React.js, a widely used JavaScript library for building dynamic, component-based user interfaces. Vite was used as the frontend bundler and development server, offering fast reloads and a streamlined development experience.
State Management	React useState	State within the application is managed using React's built-in useState hook. This approach is suitable for small to medium-scale applications, allowing for straightforward state handling within functional components.
Backend	Node.js + Express.js	Node.js provides the runtime environment for executing server-side JavaScript. Express.js, a minimalist web framework, is used to define routes, handle middleware, and manage API endpoints.
Deployment	Render / Vercel	The application was deployed using cloud platforms: Render for hosting the backend services and Vercel for deploying the frontend. These services support continuous deployment, version control integration, and scalable hosting.
Database	Google Sheets + Google Apps Script / Sheets API	Google Sheets was used as the database for storing and managing data. It was integrated using Google Apps Script and/or the Google Sheets API, allowing for basic CRUD operations and real-time synchronization. This approach is ideal for lightweight applications or prototypes due to its accessibility and ease of use.

1. System Design

The system design and specification describes how the software system will be built and how it will function. It incorporates various UML diagrams to represent different aspects of the system as well as the structure, processes, and interactions within the system.

2. Software Architecture

The software architecture serves as a blueprint of the Web based farm management system and is made up of four main parts:

Frontend: Built using ReactJS + Vite, with Axios for HTTP requests and Zustand for state management.

Backend: Powered by Node.js and Express.js. Handles authentication, routing, logic, and SQL queries.

Database: PostgreSQL. Ensures data integrity and supports relational queries.

Cloud Integration: For storing large files or backups securely.

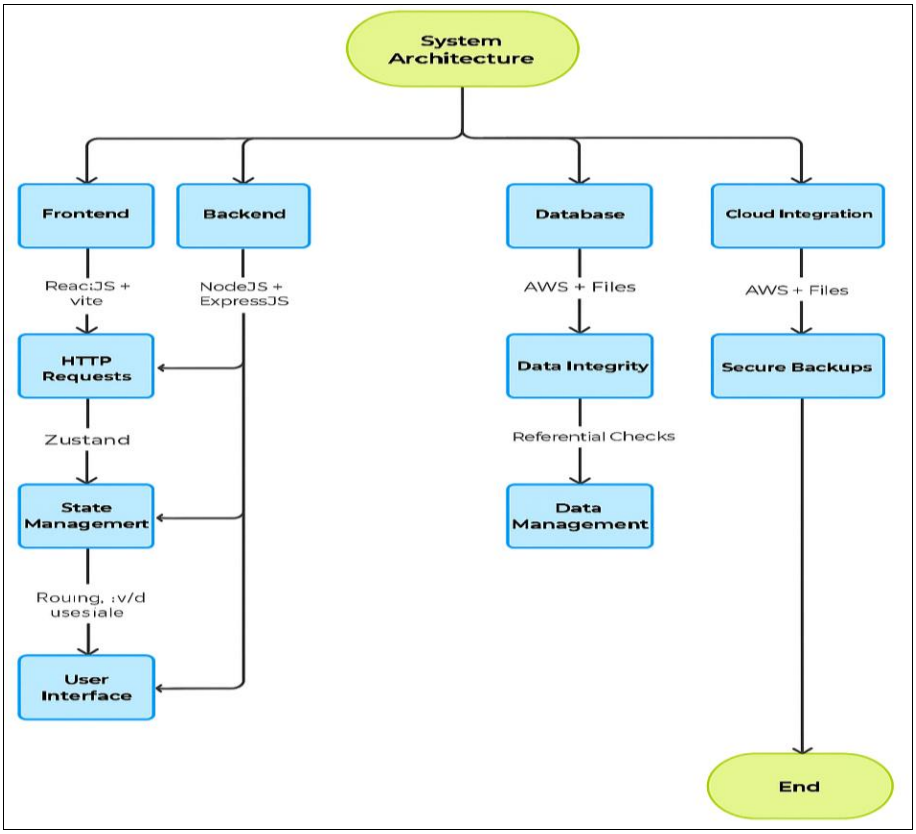

Fig 1: Software Architecture

3. Use Case Model

Use cases are valuable techniques in software development for outlining the different ways a user interacts with a system to achieve a specific goal. They provide a clear and concise description of the system's functionality from the user's perspective. Below are the main use cases for this project

- Sign Up / Log In
- View/Add Accounts
- Add/View Transactions
- Update Settings
- Access Dashboard

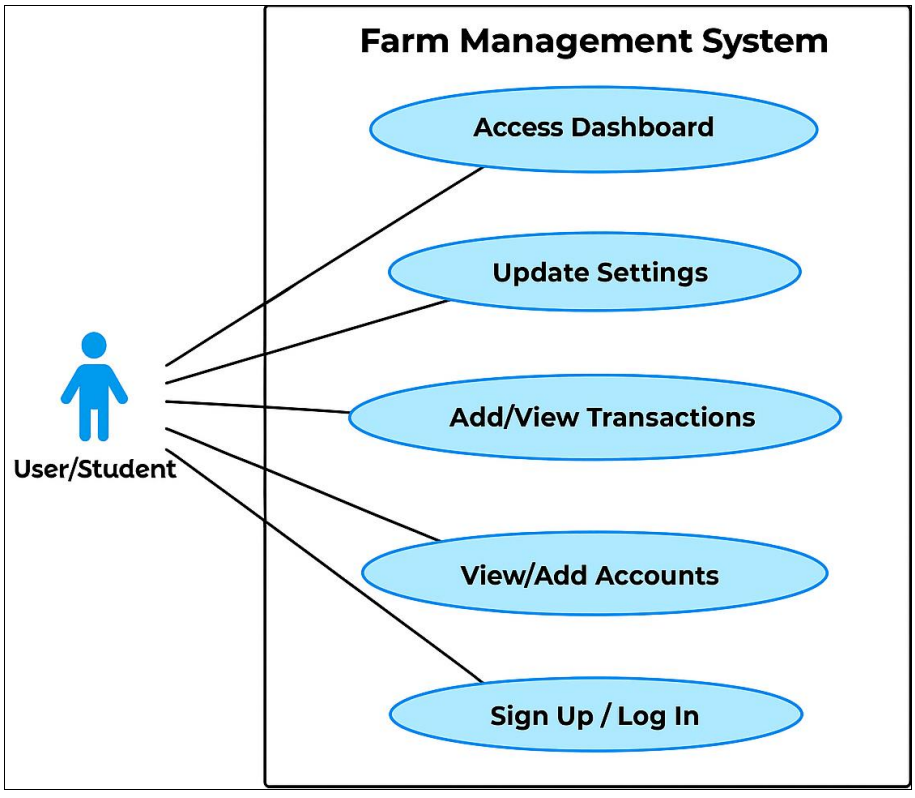

Fig 2: General Use Case Diagram.

System Implementation

This chapter details the comprehensive process involved in the implementation of the Web-Based Farm Management System, a platform developed to enhance agricultural operations, record-keeping, and data-driven decision-making for small- to medium-scale farms. The system is particularly tailored to meet the needs of academic institutions and training farms, where efficient management of farming activities, data collection, and performance analysis are essential.

1. Implementation Procedures

The implementation of the Web-Based Farm Management System followed the Prototyping Methodology, which involved building a functional prototype early in the development process, gathering user feedback, and making continuous improvements. This approach allowed for flexible adaptation to changing requirements and incremental feature development.

The development process included the following key steps

1. **Environment Setup:** The development environment was initialized using Node.js/Express.js for the backend and React.js (with Vite) for the frontend. This setup ensured fast server responses and efficient code bundling.

2. **Frontend Development:** The user interface was developed using React.js, a component-based JavaScript library. Vite was used as the frontend build tool and development server, offering fast hot reloads and efficient development workflows. Pages such as login, dashboard, and task manager were created with modular components for maintainability.

3. **State Management:** Application state was managed using React's built-in useState hook. This provided a simple and effective means of handling dynamic data and user interactions within components, suitable for the project's medium scale.

4. **API Development:** The backend logic was developed using Express.js, a lightweight web framework for Node.js. RESTful APIs were created to handle user login, task management, transaction logging, and data retrieval.

5. **Database Integration:** Google Sheets was used as the database for storing and managing structured data such as user records, tasks, and transaction logs. Integration was achieved using Google Apps Script and the Google Sheets API, enabling CRUD (Create, Read, Update, Delete) operations directly from the backend. This setup provided a lightweight and cost-effective solution suitable for small to medium-scale applications and prototyping.

6. **Integration:** Axios was used to connect frontend components to backend APIs, enabling asynchronous data transfer between the user interface and server. This ensured smooth interactions and real-time updates without full page reloads.

7. **Testing and Debugging:** Functional and integration testing were performed using Postman and browser developer tools. These tests verified the correct behavior of API endpoints, data processing logic, and UI responsiveness. Errors identified during testing were corrected to ensure system reliability.

8. **Deployment:** The application was deployed using cloud hosting platforms. The backend was hosted on Render, while the frontend was deployed on Vercel. These platforms support continuous deployment and version control, ensuring the system remains accessible and scalable for remote users.

System Modules Description

The Web-Based Farm Management System is designed using a modular architecture to ensure clarity, maintainability, and functionality across different areas of farm operations. Each module addresses a distinct aspect of the farming process, from daily task logging to inventory control and performance monitoring. The modular approach ensures scalability and enables the system to be adapted or extended based on future needs. Below is a detailed description of the key system modules implemented

1. **Sign-up Module:** Authentication and sign in

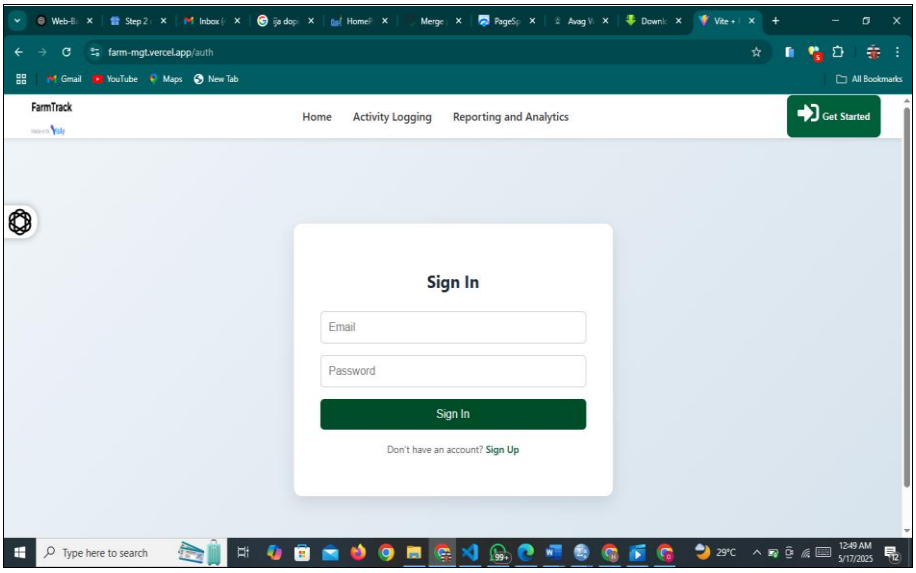

Fig 3: Sign-up Module

2. **Dashboard Module:** The Dashboard Module serves as the central interface that presents users with a summarized view of essential farm data. It provides real-time updates and visual indicators of current activities, scheduled tasks, and inventory levels.

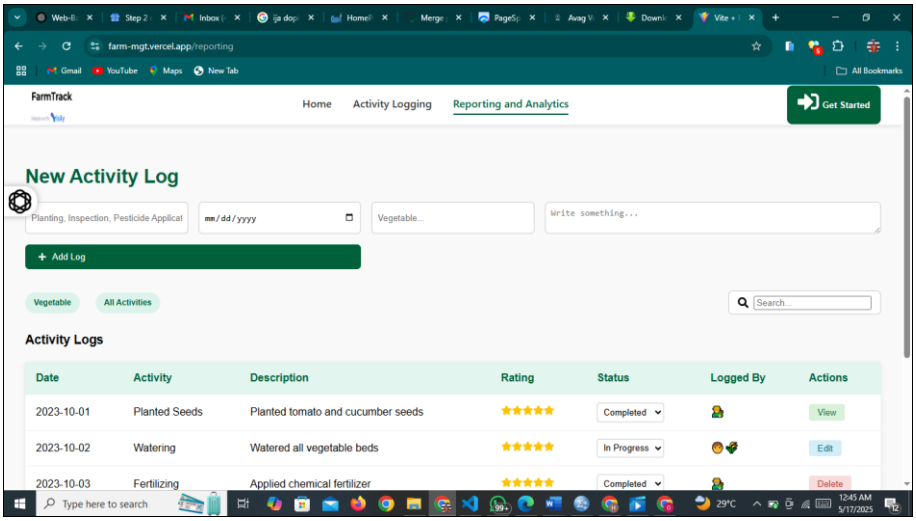

Fig 4: Dashboard Module

3. **Farm Activity Logger:** This module enables users to record all daily farming operations in a structured and time-stamped format. Activities such as land preparation, sowing, irrigation, weeding, fertilization, pest control, and harvesting can be logged systematically.

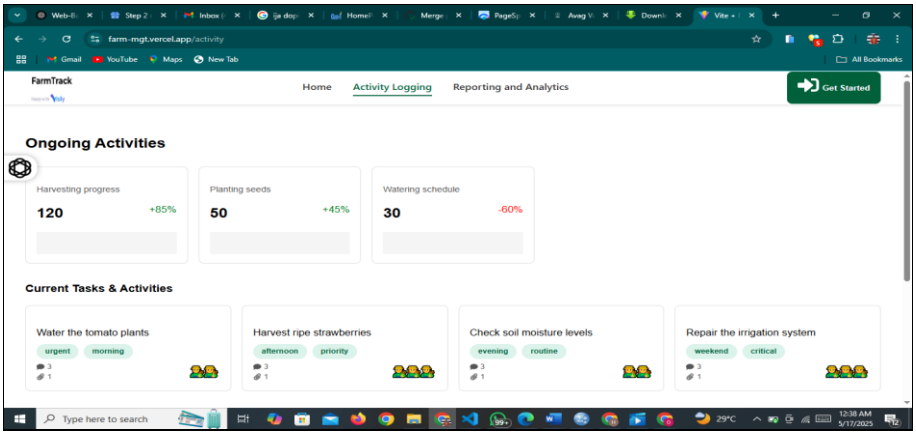

Fig 5: Activity Logger

3. Testing Strategies

To ensure the reliability, usability, and performance of the Web-Based Farm Management System, a multi-layered testing approach was adopted. This included unit testing, integration testing, user acceptance testing (UAT), and performance testing. Each testing strategy played a crucial role in identifying bugs, validating functionality, and ensuring the system met user expectations.

1. Unit Testing

Unit testing was conducted to verify the correctness of individual components and functions in both the frontend and backend. On the backend, each API endpoint built with Express.js was tested in isolation to ensure it handled requests and responses as expected, including edge cases such as invalid inputs or missing parameters. Tools like Postman and automated test scripts were used to simulate API calls and observe responses.

On the frontend, unit testing focused on React components. Core UI components such as form inputs, buttons, and display cards were tested independently to ensure they rendered correctly, responded to user interactions, and managed state effectively using React's useState. These tests helped ensure that the building blocks of the user interface functioned consistently across different parts of the application.

2. Integration Testing

Integration testing was performed to assess how well the frontend and backend systems worked together. This involved checking the complete data flow—from submitting a form on the frontend to processing it on the backend and saving or retrieving data from the PostgreSQL database.

For example, tests were conducted to verify that

- The Farm Activity Logger successfully posted data to the backend and stored it in the database.
- Inventory updates made on the frontend reflected correctly in the backend and were reloaded accurately on page refresh.
- Task assignments and updates triggered appropriate backend responses and UI feedback.

Axios, the HTTP client used in the project, was monitored during these tests to ensure all API interactions completed successfully, with proper error handling and data validation in place.

3. User Acceptance Testing (UAT)

User Acceptance Testing was conducted by inviting real users to interact with the system in a controlled environment. Participants included a farm manager and student testers who represented the target user base of the system.

Testers were asked to perform typical tasks such as

- Logging a new farm activity.
- Viewing and editing inventory items.
- Scheduling and updating a task.
- Interpreting data from the dashboard and reports.

Their feedback was recorded to identify usability issues, confusing workflows, or missing functionality. Adjustments were then made based on their suggestions, such as improving button placements, renaming unclear labels, and streamlining form inputs.

UAT played a critical role in aligning the system with real-world expectations and making it more intuitive and user-friendly.

4. Performance Testing

Performance testing was carried out to evaluate the system's responsiveness and load times under normal usage conditions. Key areas of focus included

- **Page Load Time:** Ensuring that major views such as the dashboard and farm activity pages load within 2–3 seconds on standard internet connections.
- **UI Responsiveness:** Testing whether form inputs, buttons, and tabs respond quickly to user actions without lag.
- **Data Fetching Efficiency:** Measuring the speed of API calls and checking for delays in rendering data retrieved from the backend.
- **Resource Optimization:** Verifying that assets such as images, scripts, and stylesheets were minified and properly cached using Vite's production build capabilities.

Testing tools such as Google Chrome DevTools and Lighthouse were used to audit performance metrics, including Time to First Byte (TTFB), First Contentful Paint (FCP), and overall performance scores.

Summary, Conclusion and Recommendations

The Farm Activity Logging and Reporting System is a web-based solution developed using the MERN (MongoDB, Express.js, React.js, and Node.js) stack. The system was designed to help farmers and agricultural stakeholders easily record, manage, and analyze daily farm operations such as planting, watering, and harvesting. It provides a centralized platform where users can log activities, monitor task progress, and generate performance insights for informed decision-making.

Conclusion

The Farm Activity Logging and Reporting System successfully meets its objective of providing a user-friendly digital solution for managing daily farm tasks. It supports accurate record-keeping, improves operational planning, and enhances productivity through informed decision-making. The use of modern full-stack technologies allowed for a responsive and scalable web application, adaptable to future agricultural innovations. Overall, the project was a success and lays the groundwork for broader application in real-world farm management scenarios.

Recommendation

Based on the effectiveness of the system, the following recommendations are proposed

1. Agricultural institutions or cooperatives should adopt the system to support farmers in record-keeping and performance tracking.
2. Future developers should expand the system with features like real-time collaboration, offline logging, or AI-driven crop suggestions.
3. A maintenance team should be assigned to ensure system updates, fix bugs, and handle user support.
4. Training programs or field workshops should be held to encourage farmers to embrace digital tools for modern agriculture.

References

1. FAO. Digital technologies in agriculture and rural areas: Briefing paper. Food and Agriculture Organization of the United Nations, 2019. http://www.fao.org/3/ca4887en/ca4887en.pdf
2. Kamilaris A, Kartakoullis A, Prenafeta-Boldú FX. A review on the practice of big data analysis in agriculture. Computers and Electronics in Agriculture,2017:143:23–37. https://doi.org/10.1016/j.compag.2017.09.037
3. Khan S, Ghosh S. Google Sheets as a cloud-based backend: Lightweight database model for rapid prototyping. International Journal of Web Applications,2019:11(3):65–72.
4. Köksal Ö, Tekinerdogan B. Architecture design approach for IoT-based farm management information systems. Precision Agriculture,2019:20(5):926–958. https://doi.org/10.1007/s11119-018-09614-w
5. Lamsal RR, Karthikeyan P, Otero P, Ariza A. Design and implementation of Internet of Things (IoT) platform targeted for smallholder farmers: From Nepal perspective. Agriculture,2023:13(10):1900. https://doi.org/10.3390/agriculture13101900
6. Manoharan R, Devi P. AgroSense: A decision support platform for agriculture using data analytics. International Journal of Scientific Technology Research,2020:9(1):2485–2489.
7. Ntirenganya A, Zhang Q. Cloud computing in precision agriculture: A review. Computers and Electronics in Agriculture,2021:191:106488. https://doi.org/10.1016/j.compag.2021.106488
8. Nwanekezie MN, Eze P. The use of ICT tools in Nigerian agriculture: Challenges and prospects. African Journal of Information Systems,2022:14(1):33–44.
9. Parameswaran G, Sivaprasath K. Arduino-based smart irrigation system using IoT. International Journal of

Engineering and Technology (IJET),2016:7(2):134–137.
10. Patel K, Shah H. A review on cloud computing for agriculture. International Journal of Computer Science and Mobile Computing,2021:10(5):45–50.
11. Praveen P, Kumar KS, Venkatesh A. IoT-based smart farming: Real-time monitoring of agriculture parameters. International Research Journal of Engineering and Technology (IRJET),2021:8(3):561–566.
12. Ramya M, Kumari AR. A web-based agriculture management system using open-source technology. International Journal of Scientific and Engineering Research,2023:14(2):153–159.
13. Roy S, Bhaduri S. MERN stack development: A comprehensive study. International Journal of Computer Applications,2020:176(21):1–5.