JINAV: Journal of Information and Visualization *ISSN: 2746-1440 (online)*
Vol. 6 No. 1 (2025)
https://doi.org/10.35877/454RI.jinav4160

DigiField: Real-Time Smart Farming Monitoring System Using React.js and Express.js for Agricultural Digitalization in Indonesia

Basuki Rahim Setya Permana*, Sigit Auliana, & Rizqi Kevin Octavian

Universitas Bina Bangsa, Serang, Banten, Indonesia

Abstract

The agricultural sector remains a cornerstone of Indonesia's economy, yet it faces persistent challenges such as climate change, land degradation, and limited adoption of digital technology among smallholder farmers. One critical issue is the lack of real-time monitoring systems accessible to farmers for tracking essential environmental parameters such as soil moisture and temperature. Traditional manual methods are still widely used, leading to inefficiencies and suboptimal decision-making. This study proposes the development of DigiField, a smart farming monitoring system designed to bridge the technological gap in rural agriculture by leveraging modern web technologies. The system is developed using the React.js framework for the frontend, enabling a dynamic, responsive, and mobile-friendly user interface, and Express.js for the backend to facilitate real-time data processing from sensor devices. The research adopts a technology-based development approach, involving needs analysis from field observations and interviews with local farmers. The DigiField prototype is designed to monitor soil moisture and temperature using sensor data that is transmitted and visualized in real-time through a web interface. The system aims to support data-driven agricultural decision-making and enhance productivity through accessible, low-cost technology. Preliminary testing and user feedback indicate that DigiField is effective in addressing local agricultural monitoring needs, with a user interface that is intuitive for farmers and a backend infrastructure that supports real-time updates. This study contributes to both academic discourse in the field of agricultural informatics and practical efforts to promote digital transformation in Indonesia's agricultural sector.

Keywords: Smart Farming, React.js, Web-based Monitoring System, Soil Moisture, Real-Time Data, DigiField, Agricultural Technology.

Received: 21 February 2025 Revised: 30 April 2025 Accepted: 6 June 2025

1. Introduction

The agricultural sector plays a vital role in Indonesia's economy, both as a source of employment for millions and as a significant contributor to the national Gross Domestic Product (GDP). Despite being known as an agrarian country, Indonesia continues to face complex challenges such as climate change, land degradation, and low adoption of modern agricultural technology (Ministry of Agriculture, 2020).

One of the main problems is the lack of a real-time land monitoring system accessible to farmers. Most farmers in Indonesia still rely on manual methods to monitor temperature, soil moisture, and weather conditions. These methods are not only inaccurate but also hinder timely decision-making in daily farming practices (Rachmawati, 2020). Leonardo (2021) also emphasized that integrating web-based systems can significantly enhance the efficiency of land condition monitoring.

To address this issue, the concept of smart farming has emerged by utilizing technologies such as the Internet of Things (IoT), wireless sensors, and web-based systems to support precision agriculture (Febrianti et al., 2021). The Indonesian government has also introduced initiatives such as Smart Farming 4.0, promoting the use of drones, automated irrigation systems, and digital soil sensors (Ministry of Agriculture, 2021).

* Corresponding author.
E-mail address: basukirakhim@gmail.com

While smart farming offers numerous advantages, its adoption among farmers remains low. Factors such as limited digital literacy, high initial investment costs, inadequate infrastructure, and perceptions that such technologies are difficult to use or unsuitable for local conditions continue to hinder its implementation.

The author's experience during the 2024 Impact Hackathon held in Leuwimalang Village, Bogor, revealed real challenges faced by local farmers, particularly regarding accurate and efficient land condition monitoring. During this event, the author and their team developed a prototype system called DigiField—a web-based smart farming solution designed to monitor temperature and soil moisture using sensors and modern technology. The prototype was well-received for effectively addressing farmers' needs through a simple yet practical technological approach.

DigiField was developed using the React.js framework for the frontend and Express.js for the backend. React.js enables the creation of responsive, dynamic user interfaces accessible across devices such as smartphones, while Express.js provides a flexible server-side structure for real-time sensor data integration and processing.

Previous research by Renando (2024) on the Agri-Intelligence Control System (ICS) showed that using React.js significantly improved the efficiency of agricultural monitoring interfaces. Rachmawati (2020) also highlighted the importance of digital technology adoption in agriculture to enhance productivity.

Developing a web-based monitoring system like DigiField is essential for bridging the technological gap between farmers and digital solutions. The system not only provides real-time data to farmers but also supports better decision-making based on accurate information. Designed to be affordable and user-friendly, this solution is especially suitable for small and medium-sized farming operations.

Beyond the technical relevance, DigiField's development aligns with national digital transformation efforts in agriculture. The Ministry of Agriculture has stressed the need to adopt digital technologies to support sustainable farming systems and boost productivity through accurate, timely data (Ministry of Agriculture, 2021). Thus, this research contributes not only to the field of information system development but also supports public policy implementation in the agricultural sector.

Academically, this research is expected to enrich the literature in the field of information systems and web-based technology applications in agriculture. Practically, the resulting system has the potential to be adopted by farmers in various regions to improve the quality and efficiency of their farming activities.

However, the implementation of such systems also faces challenges, including the need for farmer training, provision of sensor devices and networks, and building awareness about the importance of data-driven farming. Therefore, collaboration among researchers, technology developers, the government, and farming communities is essential for the successful widespread deployment of DigiField.

There remains a significant gap between the potential of agricultural technology and its real-world application in Indonesia. DigiField is presented as a solution offering real-time land monitoring using a React.js-based interface and an Express.js-powered backend. Designed with local needs and usability in mind, the system aims to improve efficiency, productivity, and sustainability in agriculture.

Based on this background, the author is motivated to conduct a study entitled " DigiField: Real-Time Smart Farming Monitoring System Using React.js and Express.js for Agricultural Digitalization", which aims to design a real-time agricultural monitoring system that supports farmers in making data-driven decisions for day-to-day farming operations.

2. Literature Review

he application of digital technology in agriculture has evolved rapidly in recent years, especially with the emergence of smart farming—a data-driven approach that leverages advanced technologies such as the Internet of Things (IoT), wireless sensor networks (WSNs), and web-based platforms to optimize land management and farming operations (Wolfert et al., 2017; Zhang et al., 2017). These technologies provide real-time data that enable farmers to make informed decisions, reduce input waste, and increase yield efficiency (Kamilaris et al., 2018).

Despite its advantages, many smallholder farmers still rely on manual methods to monitor field conditions like soil temperature and humidity. These traditional practices are not only time-consuming but also prone to human error and delayed response, leading to suboptimal productivity (Jayaraman et al., 2016). In this context, web-based monitoring systems offer a promising solution due to their accessibility, ease of use, and capability to deliver real-time environmental feedback (Patel & Patel, 2016).

From a technical perspective, the React.js framework has gained popularity for front-end development because of its modular structure, responsiveness, and efficiency in building interactive user interfaces. It is especially suitable for applications that require real-time data visualization and dynamic content updates (Gackenheimer, 2015). Previous research by Kiani et al. (2022) demonstrated that React.js-based interfaces significantly improve user experience and system usability in web-based environmental monitoring applications.

On the backend, Express.js is widely recognized for its lightweight architecture and effective integration with databases and RESTful APIs. It supports real-time communication through WebSockets and can handle multiple concurrent requests efficiently—critical features for IoT-based monitoring systems (Kumar et al., 2021; Liang et al., 2020). The combination of React.js and Express.js has become a standard stack in building modern, full-stack applications capable of processing sensor data in real time.

Regarding hardware integration, ESP32 microcontrollers paired with sensors such as DHT11 or capacitive soil moisture sensors have been widely adopted in smart farming prototypes due to their low cost, power efficiency, and Wi-Fi capabilities (Srinivas et al., 2019). However, many implementations have relied on proprietary cloud platforms like Blynk, which limit system customization and scalability. Open-source approaches are increasingly preferred for enabling localized development and community-driven adaptation (Bharambe et al., 2021; Sharma et al., 2020).

User-centered design is another crucial factor in the adoption of smart farming tools. Systems that provide simple, responsive, and intuitive interfaces, especially with visual aids such as charts, alerts, and color-coded indicators, have shown better acceptance among non-technical users, including rural farmers (Amin et al., 2022; Ahmad et al., 2021). These features help bridge the gap between complex data and actionable decisions, especially in low-literacy or low-connectivity settings.

In summary, current literature highlights that a synergistic integration of IoT, modern web technologies (React.js and Express.js), and user-centered interface design is essential to build effective and scalable agricultural monitoring systems. However, many existing implementations are either too generic or not tailored to the specific contexts of developing countries such as Indonesia. To address this gap, this study introduces DigiField, a smart farming platform designed to support smallholder farmers with real-time environmental data through an accessible and customizable web interface. The solution aims to empower farmers with accurate, timely, and actionable information to enhance productivity and resilience in the agricultural sector.

3. Methods

In this study, the development of the DigiField Smart Farming system adopts the Waterfall model, a classic software development methodology that is sequential and systematic. Each phase in this model is completed in order before moving on to the next. The Waterfall model is chosen because it provides a clear workflow structure and is well-suited for projects with well-defined requirements from the outset. The stages of the Waterfall model applied in this system development are as follows:

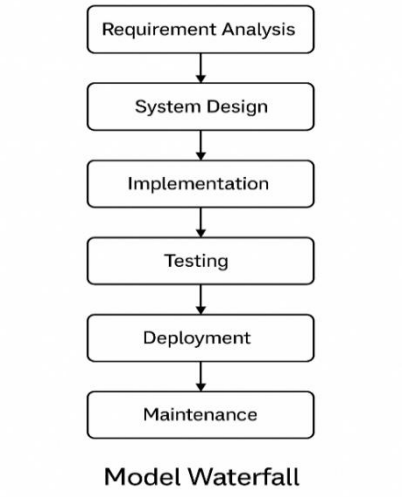

Figure 1. System Development Method Flowchart

3.1. Requirement Analysis

This phase aims to identify and define the system requirements based on observations, interviews, and literature studies. The system requirements include, Functional requirements (e.g., real-time monitoring of temperature and humidity), Non-functional requirements (e.g., system speed and ease of use). The output of this phase is the requirement specification document, which serves as the basis for the design phase.

Based on field observations, interviews with farmers in Leuwimalang Village – Bogor, and reflections on previous experience using the Blynk platform in the early version of DigiField, several issues and system needs were identified. Key problems included dependency on a closed platform, limited interface customization, lack of scalability, absence of historical data integration, and restricted backend control. To address these, the new system requires flexible, self-managed data handling, a responsive and user-friendly interface tailored to local farmers, scalable modular architecture, real-time sensor-to-dashboard communication, interactive historical data visualization, and the use of open-source technologies. This research aims to meet these needs through a modern technological approach using React.js and Express.js, with a user-centered design focused on farmers.

3.2. System Design

Based on the requirement analysis, this phase focuses on designing the system architecture, user interface, and selecting the technologies to be used. Activities in this phase include, Frontend design using React.js and Tailwind CSS, Backend design using Express.js and MongoDB, Communication design between sensors (ESP32, DHT22, Soil Moisture Sensor) and the server using HTTP/MQTT protocols, Creation of diagrams such as use case, activity diagram, and class diagram.

3.2.1. Use Case Diagram

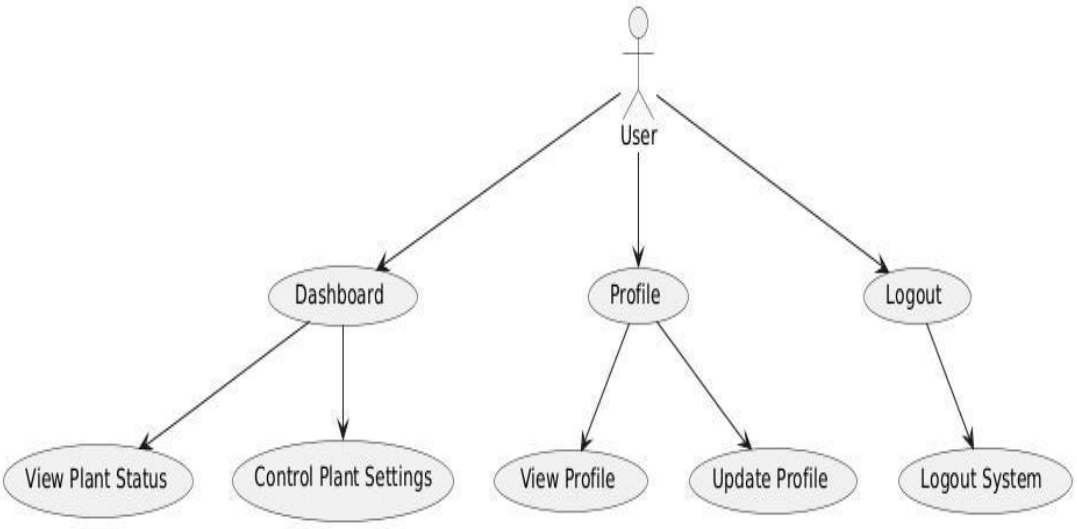

Figure 2. Use Case Diagram

Based on Figure 3.2, the Use Case Diagram illustrates the interaction between the actor (User) and the web-based plant monitoring system. This diagram shows the main features accessible by users within the system and the relationships between different functions.

3.2.2. Activity Diagram

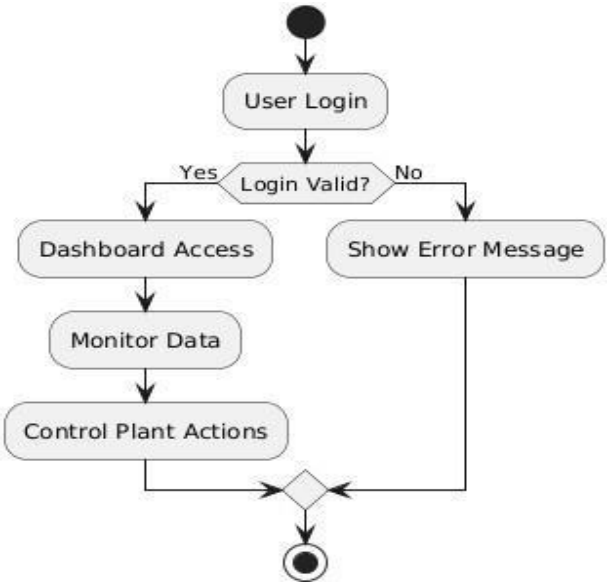

Figure 3. Activity Diagram for User Login and System Control

This diagram illustrates the main activities performed by the user in the monitoring system, starting from login to controlling plant conditions.

3.2.3. Sequence Diagram

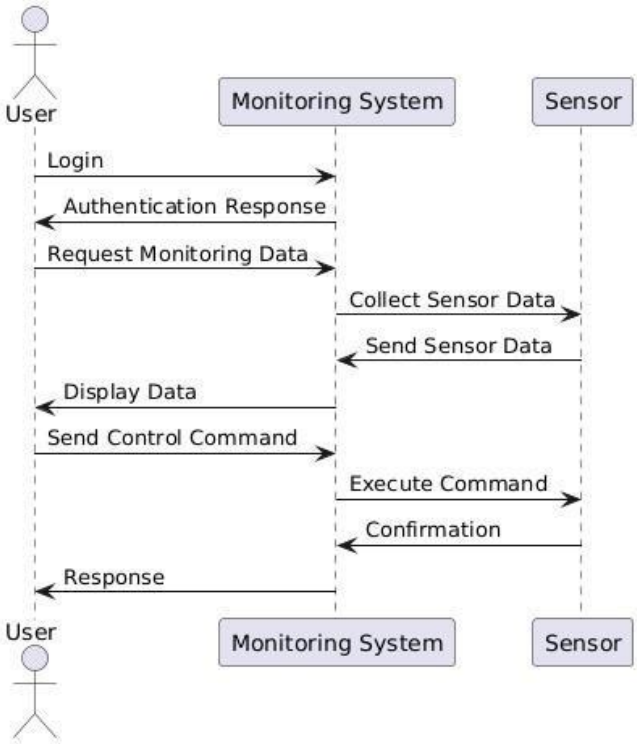

Figure 4. Sequence Diagram for Monitoring Interaction

This diagram depicts the chronological sequence of communication between the User, Monitoring System, and Sensor.

3.2.4. Class Diagram

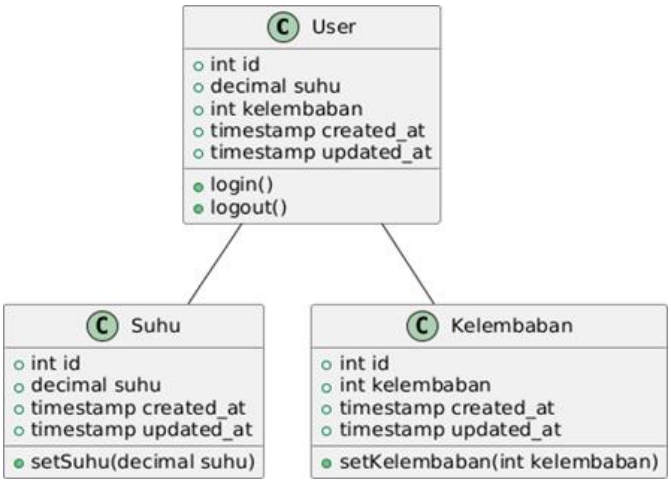

Figure 5. Class Diagram of the Monitoring System

This diagram explains the data structure of objects used in the plant monitoring system. It consists of three main classes: User, Temperature, and Humidity.

3.3. Implementation

This phase involves translating the system design into actual code. The implementation is carried out in two main parts, Frontend, Developing the dashboard interface using React.js to display real-time sensor data in charts and tables. Backend. Developing RESTful APIs using Express.js to receive, store, and serve sensor data to the frontend.

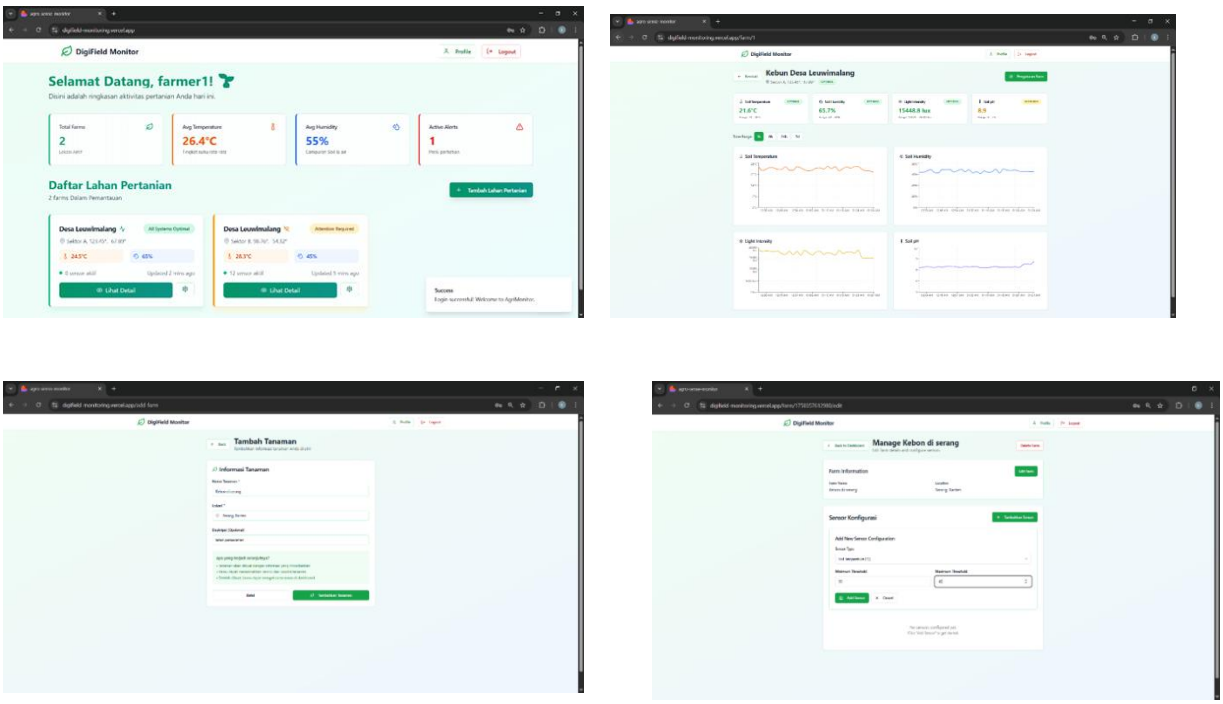

Figure 6. Implementation

After successfully logging in, users are directed to the DigiField Main Dashboard, which serves as the central control hub and summary of agricultural monitoring activities. The dashboard displays several key information cards, including the total number of monitored farms, average temperature from all active sensors, average humidity (combining soil

and air moisture), and the number of active alerts that require user attention. Below this summary, a list of monitored farms is shown, with each entry displaying the location, sensor status (such as "All Systems Optimal" or "Attention Required"), the number of active sensors, and buttons to view details or manage configurations. By clicking "View Details," users are taken to a specific farm's detailed monitoring page, which presents real-time sensor data on soil temperature (with optimal status), soil humidity, light intensity (in lux), and soil pH levels. Each parameter is equipped with a status indicator that changes color (green for Optimal, yellow for Warning) based on defined thresholds. Users can also select a time range (1 hour, 6 hours, 24 hours, or 7 days) to analyze data trends over time. To add a new farm, users can click the "Add Farm" button on the dashboard, which leads to a form where they can enter the plant/farm name, location (e.g., district, province), and an optional description about the crop or soil condition. After submitting the form, the system processes the input and displays a success notification. The newly added farm will then appear in the dashboard list for monitoring.

3.4. Testing

System testing is a crucial step in the software development process to ensure that all functionalities operate according to requirements and that there are no execution errors within the program. In the DigiField Monitor system, testing is carried out using two approaches—White Box Testing and Black Box Testing—to guarantee the reliability of both internal logic and user interface functionality.

3.4.1. White Box Testing

White Box Testing is a software testing method that involves examining the internal structure, logic flow, and source code of the program directly. The goal of this testing is to ensure that each function, condition, and code branch operates as intended.

In the DigiField Monitor system, White Box Testing is focused on several core backend functions (built with Express.js), including authentication, sensor data processing, and threshold logic. The testing is performed manually and supported by unit testing tools such as Jest to ensure that all blocks of code (e.g., if, else, switch, loop) are adequately covered during the testing process.

Table 1. White Box Testing

No	Module	Flow Tested	Expected Result
1	Login API	Email and password validation	Success
2	Login API	Empty email or password	Login failed, error message displayed
3	Sensor Data Handler	Incomplete input (e.g., missing max value)	Error with status 400
4	Sensor Data Handler	minThreshold ≥ maxThreshold	Error: Invalid threshold
5	Sensor Data Handler	All inputs are valid	Input accepted and processed
6	Farm Configurator	Save threshold configuration	Successfully saved to the database
7	Farm Configurator	Save configuration with null input	Save failed, validation error displayed
8	Alert Generator	Sensor value < min threshold	Status: Alert
9	Alert Generator	Sensor value > max threshold	Status: Alert
10	Alert Generator	Sensor value within threshold range	Status: Normal
11	Dashboard Display	Display total number of farms and active sensors	Data displayed correctly from database
12	Monitoring Charts	Display dynamic sensor charts with real-time data	Charts displayed and updated automatically
13	User Registration	Register with complete data	Account successfully created

3.4.2. Black Box Testing

Black Box Testing is a software testing method that focuses on the functional aspects of the application without considering its internal structure or source code. Testing is conducted based on system inputs and expected outputs. The primary objective of this testing is to verify that the system operates according to user requirements defined during the design phase. Black Box Testing was applied to the user interface of the DigiField Monitor application, particularly on the login page, dashboard, input forms, and sensor configuration features.

Table 2. Black Box Testing

No	Feature Tested	Test Scenario	Expected Result	Status
1	Login	Enter correct email and password	User successfully redirected to dashboard	Passed
2	Registration	Input username, email, password, and confirm password	New account created and user can log in	Passed
3	Add Farm	Input farm name, location, and description	New farm appears on the dashboard	Passed
4	View Sensor Details	Click "View Details" button on an active farm	Sensor data displayed in graphical format	Passed
5	Add Sensor & Threshold	Input sensor type, minimum and maximum threshold	Sensor is saved and monitored	Passed
6	Alert Status Generator	Sensor value below minimum threshold	System displays a warning alert	Passed
7	Page Responsiveness	Test display on both mobile and desktop screen sizes	Interface remains consistent and usable	Passed

4. Result and Discussion

The development and implementation of the DigiField Smart Farming monitoring system have successfully addressed several core challenges in traditional agricultural monitoring, particularly in rural farming environments. This section presents the outcomes of the system development, supported by testing results and discussions on its practical implications and performance.

4.1. System Implementation Result

The final DigiField system consists of a full-stack web application developed using React.js for the frontend and Express.js for the backend, with data stored in a MongoDB database. The system was deployed successfully using Vercel for the frontend and Render for the backend, enabling real-time data access from various devices. Key functionalities implemented include:

1) User Authentication and Registration: Ensuring secure access to the dashboard.
2) Real-time Monitoring: Displaying updated sensor data (temperature, humidity, pH, and light intensity) in interactive charts.
3) Farm Management: Adding, editing, and configuring multiple farms and sensors.
4) Alert System: Triggering visual alerts when sensor data exceeds defined thresholds.
5) Data Visualization: Supporting various time ranges (hourly, daily, weekly) for historical analysis.

These features collectively support data-driven decision-making for farmers and provide a scalable foundation for further system expansion.

4.2. Testing Results

System reliability was validated through White Box and Black Box Testing. White Box Testing was conducted on backend functions such as user login, sensor data handling, and alert generation. All logical flows, conditions, and edge cases were covered using manual inspection and Jest unit tests. The test results confirmed that all backend modules performed as expected without critical logic errors. Black Box Testing focused on the frontend interface, including login, registration, dashboard interaction, sensor display, and form submission. Each feature was tested from the user perspective, and all functions responded correctly based on predefined input scenarios. Responsive design testing also showed that the system performed well on both mobile and desktop devices.

4.3. Discussion

The successful development of DigiField demonstrates the effectiveness of modern web technologies in solving practical challenges in the agricultural sector. Compared to the initial prototype using the Blynk platform, the new

system offers enhanced flexibility, local data control, improved scalability, and a customizable user experience tailored to the needs of local farmers. Several advantages were observed:

1) Flexibility, Open-source technologies allow for easier customization and community-based development.
2) Real-time Monitoring, Enables immediate response to environmental changes, improving decision accuracy.
3) User-Centered Design, The intuitive interface increases usability for non-technical users such as farmers.

However, the study also highlights several areas for future improvement. These include the integration of mobile app versions, offline data caching for low-connectivity areas, and advanced analytics (e.g., prediction models using machine learning).

5. Conclusion

In conclusion, the development of the DigiField Smart Farming monitoring system using React.js and Express.js has proven effective in addressing key agricultural challenges by enabling real-time monitoring, user-friendly interfaces, and scalable system architecture tailored for local farmers. The transition from a closed platform to an open-source web-based solution has enhanced flexibility, autonomy, and data management, while successful testing confirmed the system's functional reliability and usability. DigiField not only supports informed decision-making through accurate sensor data but also contributes to Indonesia's broader digital transformation in agriculture, offering a sustainable and practical tool to improve productivity and precision in farming practices.

References

Ahmed, H., Traore, I., & Saad, S. (2018). Detecting opinion spams and fake news using text classification. Security and Privacy, 1(1), e9. https://doi.org/10.1002/spy2.9

Alsmadi, I., & O'Brien, A. (2020). A Web-Based Framework for Fake News Detection. International Journal of Human–Computer Interaction, 36(12), 1128–1139. https://doi.org/10.1080/10447318.2020.1726106

Bondielli, A., & Marcelloni, F. (2019). A survey on fake news and rumour detection techniques. Information Sciences, 497, 38–55. https://doi.org/10.1016/j.ins.2019.05.035

Conroy, N. J., Rubin, V. L., & Chen, Y. (2015). Automatic deception detection: Methods for finding fake news. Proceedings of the Association for Information Science and Technology, 52(1), 1–4. https://doi.org/10.1002/pra2.2015.145052010082

Gilda, S. (2017). Evaluating machine learning algorithms for fake news detection. 2017 IEEE 15th Student Conference on Research and Development (SCOReD), 110–115. https://doi.org/10.1109/SCORED.2017.8305411

Joachims, T. (1998). Text categorization with Support Vector Machines: Learning with many relevant features. European Conference on Machine Learning, 137–142. https://doi.org/10.1007/BFb0026683

Kaliyar, R. K., Goswami, A., & Narang, P. (2021). FakeBERT: Fake news detection in social media with a BERT-based deep learning approach. Multimedia Tools and Applications, 80, 11765–11788. https://doi.org/10.1007/s11042-020-10183-2

Kowsari, K., Meimandi, K. J., Heidarysafa, M., Mendu, S., Barnes, L. E., & Brown, D. E. (2019). Text classification algorithms: A survey. Information, 10(4), 150. https://doi.org/10.3390/info10040150

Lazer, D. M., Baum, M. A., Benkler, Y., Berinsky, A. J., Greenhill, K. M., Menczer, F., ... & Zittrain, J. L. (2018). The science of fake news. Science, 359(6380), 1094–1096. https://doi.org/10.1126/science.aao2998

Pérez-Rosas, V., Kleinberg, B., Lefevre, A., & Mihalcea, R. (2018). Automatic detection of fake news. Proceedings of COLING 2018, 3391–3401. https://aclanthology.org/C18-1287

Ramesh, M., & Siddaiah, P. (2021). Comparative analysis of machine learning algorithms for fake news detection. Materials Today: Proceedings, 47, 5214–5219. https://doi.org/10.1016/j.matpr.2021.05.325

Ruchansky, N., Seo, S., & Liu, Y. (2017). CSI: A hybrid deep model for fake news detection. Proceedings of the 2017 ACM CIKM, 797–806. https://doi.org/10.1145/3132847.3132877

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. Information Processing & Management, 24(5), 513–523. https://doi.org/10.1016/0306-4573(88)90021-0

Shu, K., Sliva, A., Wang, S., Tang, J., & Liu, H. (2017). Fake news detection on social media: A data mining perspective. ACM SIGKDD Explorations Newsletter, 19(1), 22–36. https://doi.org/10.1145/3137597.3137600

Vosoughi, S., Roy, D., & Aral, S. (2018). The spread of true and false news online. Science, 359(6380), 1146–1151. https://doi.org/10.1126/science.aap9559

Ahmad, S., Arshad, M., Hussain, M., & Raza, M. (2021). Smart agriculture monitoring system using IoT. Sustainability, 13(6), 3295. https://doi.org/10.3390/su13063295

Amin, M. Y., Shahid, A. R., Habib, M. A., & Habib, U. (2022). IoT-based smart agriculture: A review of recent trends. Journal of Ambient Intelligence and Humanized Computing, 13, 3219–3245. https://doi.org/10.1007/s12652-021-03490-5

Bharambe, A., Shahane, A., & Patil, V. (2021). Design and implementation of smart irrigation system using open-source IoT platform. Materials Today: Proceedings, 45, 1226–1231. https://doi.org/10.1016/j.matpr.2020.11.796

Gackenheimer, C. (2015). Introduction to React. Apress. https://doi.org/10.1007/978-1-4842-1083-1

Jayaraman, P. P., Yavari, A., Georgakopoulos, D., Morshed, A., & Zaslavsky, A. (2016). Internet of Things platform for smart farming: Experiences and lessons learnt. Sensors, 16(11), 1884. https://doi.org/10.3390/s16111884

Kamilaris, A., Kartakoullis, A., & Prenafeta-Boldú, F. X. (2018). A review on the practice of big data analysis in agriculture. Computers and Electronics in Agriculture, 143, 23–37. https://doi.org/10.1016/j.compag.2017.09.037

Kiani, A., Niazi, M., & Rauf, A. (2022). Improving user experience in web applications with React.js: A usability evaluation. Journal of Web Engineering, 21(4), 971–990.

Kumar, R., Singh, M., & Mohan, B. (2021). Real-time smart agriculture monitoring system using Node.js and Express.js. Procedia Computer Science, 185, 618–625. https://doi.org/10.1016/j.procs.2021.05.064

Liang, X., Shetty, S., Tosh, D., & Kamhoua, C. (2020). Web technologies for IoT application development: An empirical evaluation. IEEE Access, 8, 23479–23490. https://doi.org/10.1109/ACCESS.2020.2970271

Patel, K. K., & Patel, S. M. (2016). Internet of Things-IOT: Definition, characteristics, architecture, enabling technologies, application & future challenges. International Journal of Engineering Science and Computing, 6(5), 6122–6131.

Sharma, N., Rana, D., & Gautam, A. (2020). Implementation of IoT based real-time soil monitoring system. Materials Today: Proceedings, 33, 4346–4351. https://doi.org/10.1016/j.matpr.2020.03.483

Srinivas, K., Venu, M., & Naik, G. R. (2019). Low-cost soil moisture monitoring system using ESP32. International Journal of Recent Technology and Engineering, 8(2S11), 487–491.

Wolfert, S., Ge, L., Verdouw, C., & Bogaardt, M. J. (2017). Big data in smart farming–a review. Agricultural Systems, 153, 69–80. https://doi.org/10.1016/j.agsy.2017.01.023

Zhang, Y., Wang, G., & Wang, X. (2017). Precision agriculture: An overview. Science China Technological Sciences, 60(6), 1093–1106. https://doi.org/10.1007/s11431-017-9076-1

Zhang, C., & Kovacs, J. M. (2012). The application of small unmanned aerial systems for precision agriculture: A review. Precision Agriculture, 13, 693–712. https://doi.org/10.1007/s11119-012-9274-5