Jurnal Teknik Informatika Vol. 18 No. 2, October 2025 (236-247)
ISSN: p-ISSN 1979-9160 (Print)| e-ISSN 2549-7901 (Online)
DOI: https://doi.org/10.15408/jti.v18i2.46450

JURNAL TEKNIK INFORMATIKA
Homepage : http://journal.uinjkt.ac.id/index.php/ti

Back-End Development of an Interactive Dashboard with Real-Time API Integration for Chili Plant Monitoring in Precision Agriculture

Azwar Farrel Wirasena[1], Hanif Fakhrurroja[2], Dita Pramesti[3*]

[1,2,3]Information System Study Program, School of Industrial Engineering, Telkom University
[1,2,3]Main Campus (Bandung Campus), Jl. Telekomunikasi No.1, Bandung 40257, West Java, Indonesia

Article:
Accepted: June 17, 2025
Revised: February 21, 2025
Issued: October 30, 2025

***Correspondence Address:**
ditapramesti@telkomuniversity.ac.id

ABSTRACT

This research focuses on the development of an interactive web-based dashboard to support a precision agriculture system for chili plants. The primary focus of this research is on the back-end development of the system. The system integrates several internal and external APIs, including the Flask API (internal) for plant disease classification and growth prediction, and the Google Gemini API for the AI-powered chatbot that provides consultation to farmers (external). These features allow farmers to receive automatic disease diagnosis and growth predictions, improving decision-making and crop management. The dashboard also presents weather information, environmental data, and nanobubble data, along with Echarts gauge charts for seven essential metrics: Electrical Conductivity (EC), temperature, humidity, pH, nitrogen, phosphorus, and potassium. Data for the environmental and nanobubble data is retrieved from the ThingSpeak API (external), while weather information is fetched from the OpenWeatherMap API (external). The system was thoroughly tested using Postman to ensure all API endpoints function correctly. The results confirmed that all endpoints responded with status code 200 OK, indicating stable back-end performance. Performance testing showed response times stabilizing at 2000 ms after initial 4500 ms peaks, confirming efficient handling, reliable endpoints, and deployment readiness.

Keywords : *AI-powered chatbot; back-end development; disease classification; growth prediction; interactive dashboard; precision agriculture.*

1. INTRODUCTION

Precision agriculture is a modern agricultural approach that utilizes advanced technologies such as sensors, data analysis, and automation to optimize farming practices and improve crop yields. This approach is particularly beneficial for chili plant cultivation, which requires precise monitoring of environmental factors such as soil moisture, temperature, humidity, and plant health [1], [2]. In this research, the focus is placed primarily on the back-end development of an interactive web-based dashboard that supports precision agriculture for chili plants [3]. Application Programming Interfaces (APIs) play a crucial role in enabling seamless communication between the back-end systems and external services, providing the flexibility to access real-time data from various sources, integrate machine learning models, and deliver valuable insights [4]. APIs facilitate efficient data exchange between the front-end and back-end systems, ensuring that all data required for analysis and predictions is properly fetched, processed, and served to the user interface.

The primary objective of this research is to create a robust back-end system that integrates various internal and external APIs to automate disease classification, growth prediction, and provide real-time environmental monitoring for chili plants [5], [6]. The title clearly reflects the focus on back-end development. The system's core functionalities rely heavily on back-end processing to aggregate data from multiple sources and deliver it to the front-end in real-time. The system integrates several internal and external APIs, including the Flask API (internal) for plant disease classification and growth prediction, and the Google Gemini API (external) for an AI-powered chatbot that provides consultation to farmers. Additionally, the ThingSpeak API (external) is used to retrieve environmental data, such as soil moisture and other key metrics from sensor channels, while the OpenWeatherMap API (external) provides weather data, including temperature and humidity, which are crucial for chili plant care. These APIs collectively enable the system to provide accurate, real-time data, empowering farmers to make informed decisions and manage their crops effectively.

In the field of precision agriculture, particularly chili plant cultivation, numerous studies have focused on leveraging IoT sensors, machine learning, and real-time data systems for disease classification, growth prediction, and environmental monitoring. However, few have integrated advanced features such as AI-powered chatbot consultations and real-time data visualization with Echarts gauges. This research aims to fill that gap by combining these innovative features, enabling farmers to make more informed, data-driven decisions. A Systematic Literature Review (SLR) was conducted to identify the tools, technologies, and features employed in previous studies, highlighting the contributions of AI chatbot integration and Echarts gauges. The review followed the PRISMA framework [7], and Figure 1 illustrates the selection and evaluation process of relevant studies.

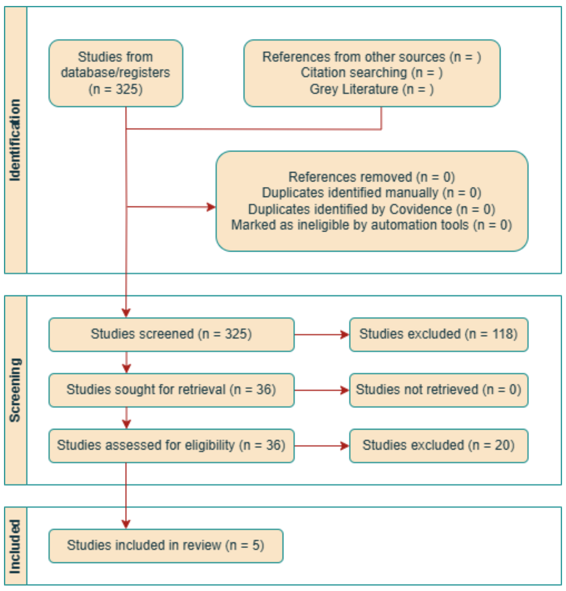
Identification
Studies from database/registers (n = 325)
References from other sources (n =)
Citation searching (n =)
Grey Literature (n =)
References removed (n = 0)
Duplicates identified manually (n = 0)
Duplicates identified by Covidence (n = 0)
Marked as ineligible by automation tools (n = 0)
Screening
Studies screened (n = 325)
Studies excluded (n = 118)
Studies sought for retrieval (n = 36)
Studies not retrieved (n = 0)
Studies assessed for eligibility (n = 36)
Studies excluded (n = 20)
Included
Studies included in review (n = 5)

Figure 1. *PRISMA diagram for systematic literature review*

Following the PRISMA diagram, Table 1 summarizes the key findings from the Systematic Literature Review (SLR).

Table 1. *Systematic literature review*

Citation	Features	App	Tools	API	Key Findings
[8]	Disease Classification, Growth Prediction, Environmental Monitoring, Real-time Data Logging	Web	Laravel, YOLOv8, Python, OpenCV	-	Utilized deep learning for image recognition in hydroponic environments, improving disease detection and growth stage monitoring.
[9]	Disease Classification, Growth Prediction, Environmental Monitoring, Decision Support	Mobile	TensorFlow, AI Models	-	Focused on AI's application in agriculture, particularly for disease classification and crop health predictions.
[10]	Soil Monitoring, Temperature Control, Automated Irrigation, Data Logging	Web	IoT Sensors, Hybrid Machine Learning, MySQL	ThingSpeak API	IoT-based soil monitoring with hybrid machine learning for real-time predictions of tomato crop diseases, enhancing pest control efficiency.
[11]	Weather Data, Environmental Data, Growth Prediction, Decision Support	Mobile	Big Data Analytics, AI Models, IoT Sensors	OpenWeatherMap API	Applied AI and big data to manage weather data in precision agriculture, improving decision support for crop management.
[12]	Disease Classification, Visual Monitoring, Real-time Disease Detection	Web	Deep Learning, Computer Vision, TensorFlow	-	Comprehensive review of deep learning and computer vision applications in plant disease detection and its integration in precision agriculture.
Our Research	Weather Information, Environmental Data, Torn Nanobubble, Echarts Gauge Graphs, Disease Classification, Growth Prediction, AI Chatbot	Web	Laravel, PHP, MySQL, Postman	ThingSpeak API, OpenWeatherMap API, Google Gemini API, Flask API	Introduced AI-powered chatbot for real-time consultation, and used Echarts gauges to display 7 metrics (EC, Temperature, Humidity, pH, Nitrogen, Phosphorus, Potassium) for each sensor channel.

Table 1 presents a summary of the key features and innovations from previous studies in the field of precision agriculture, particularly in chili plant monitoring systems. Most prior research has focused on fundamental aspects such as disease classification, growth prediction, and environmental monitoring. While relevant, these features have been widely implemented in various studies. As a distinguishing factor, this study introduces two main innovations, namely the AI Chatbot and Echarts gauges. The AI Chatbot enables direct interaction between users and the system, providing real-time, personalized responses based on current environmental data, thereby supporting faster and more accurate decision-making. Meanwhile, the Echarts gauges offer visual representations of seven essential parameters (EC, temperature, humidity, pH, nitrogen, phosphorus, and potassium) from each sensor channel, enhancing data readability and overall user experience. These features offer a more interactive, adaptive, and data-driven approach to the implementation of precision agriculture systems, representing added value not found in previous research.

2. METHODS

The Prototyping Method is widely used in software development due to its ability to quickly generate a working prototype, facilitating rapid testing and gathering user feedback. This method is advantageous for validating system functionality early in the development cycle, enabling prompt identification and correction of shortcomings [13]. In this research, the Prototyping Method allowed iterative testing and refinement to ensure that the system's integration of various internal and external APIs met required standards and provided necessary functionalities. Furthermore, this approach helped address potential challenges in real-time data processing and system performance before final deployment, resulting in a reliable and efficient system.

Several tools were selected based on their ability to support scalable and efficient back-end development. Laravel 12, a popular PHP framework, was chosen for its flexibility in building robust RESTful APIs with features such as routing, authentication, and database management. PHP 8 was utilized to handle multiple API requests and efficiently process large datasets in real-time, meeting the demands of a high-load system [14]. MySQL was chosen as the relational database to manage data related to user information, sensor data, and prediction results, allowing for efficient storage and retrieval of structured data [15]. Postman was essential for API testing, allowing for the validation of each endpoint to ensure the system's responses were accurate and met the requirements of the project [16]. The integration of APIs, including the ThingSpeak API for environmental monitoring, OpenWeatherMap for weather data, and Google Gemini for chatbot functionality, provided real-time insights critical for improving decision-making in precision agriculture.

2.1. Prototyping Method

The development method employed in this research was the Prototyping Method with a single iteration approach, selected for its ability to rapidly develop and test an initial prototype followed by refinement based on feedback. This approach suits back-end development well, enabling smooth integration of internal APIs (such as the Flask API for disease classification and growth prediction) and external APIs (including ThingSpeak API for environmental data, OpenWeatherMap API for weather data, and Google Gemini API for the AI-powered chatbot). This method ensures that real-time data processing, accurate disease classification, and growth prediction features are effectively implemented. Moreover, the method's flexibility allows for continuous improvements through early-stage evaluations, ensuring that the system satisfies user needs, as shown in Figure 2.

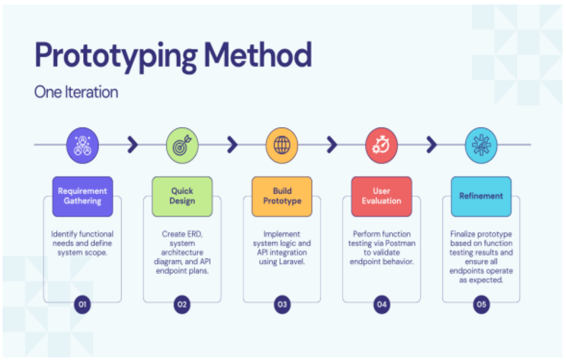
Prototyping Method
One Iteration
Requirement Gathering
Identify functional needs and define system scope.
01
Quick Design
Create ERD, system architecture diagram, and API endpoint plans.
02
Build Prototype
Implement system logic and API integration using Laravel.
03
User Evaluation
Perform function testing via Postman to validate endpoint behavior.
04
Refinement
Finalize prototype based on function testing results and ensure all endpoints operate as expected.
05

Figure 2. *Prototyping method (one iteration)*

Figure 2 illustrates the five stages of the prototyping method applied in this research. First, Requirement Gathering identified key features including disease classification, growth prediction, and real-time environmental monitoring. This was followed by Quick Design, where design artifacts such as the ERD, system architecture diagram, and API endpoint specifications were prepared. In the Build Prototype stage, system logic was implemented using the Laravel framework, integrating both internal and external APIs to enable real-time data processing. The User Evaluation phase involved functional testing with Postman to validate the behavior of the system's API endpoints, ensuring all components functioned as expected with real-time data. Finally, in the Refinement phase, the prototype was optimized based on feedback and testing results, resolving any issues and making necessary adjustments. This iterative process ensured that the system was ready for further integration and eventual deployment.

2.2. Tools and Development Environment

In the development of the chili plant precision agriculture system, several tools and technologies were utilized to ensure an efficient and scalable back-end system. Laravel 12 was chosen for building RESTful APIs, offering modern features and support for API development, including Laravel Sanctum for authentication. The back-end was developed using PHP 8 to handle multiple API requests and process large datasets efficiently. MySQL was used to manage system data, including user information, disease classifications, and sensor data from external APIs. Postman was employed for API testing to ensure system responses were accurate. Both internal and external APIs were integrated to provide real-time data: the ThingSpeak API for environmental data, OpenWeatherMap API for weather data, Google Gemini API for the AI-powered chatbot, and Flask API for disease classification and growth prediction. These integrations enabled real-time insights, assisting farmers in making informed decisions for improved crop management.

As shown in Table 2, the tools and technologies used in the development of the back-end system for this research.

Jurnal Teknik Informatika Vol. 18 No. 2, October 2025 (236-247)
ISSN: p-ISSN 1979-9160 (Print)| e-ISSN 2549-7901 (Online)
DOI: https://doi.org/10.15408/jti.v18i2.46450

Table 2. *Tools and development environment*

Component	Tools / Framework
Back-End Framework	Laravel 12
Programming Language	PHP 8
Database	MySQL
API Testing	Postman
Authentication	Laravel Sanctum
AI Chatbot Integration	Google Gemini API
Weather API	OpenWeatherMap API
Environmental Sensor API	ThingSpeak API
Disease Model	CNN (.h5 with Keras)
Growth Prediction Model	SVM (.pkl with XGBoost)

3. RESULTS AND DISCUSSION

The system developed for chili plant precision agriculture monitoring has successfully integrated key features that support efficient data processing, analysis, and decision-making. The system includes four main features: weather and environmental monitoring, disease classification, growth prediction, and an AI-powered chatbot. One of the primary indicators of the system's success is the effective integration of internal and external APIs. For weather and environmental monitoring, the system uses ThingSpeak API (external) to retrieve environmental sensor data and OpenWeatherMap API (external) to fetch real-time weather information. These APIs communicate seamlessly with the backend system, ensuring that real-time data on temperature, humidity, pH, nitrogen levels, and other metrics are consistently delivered to the dashboard. This allows farmers to make informed decisions based on the current environmental conditions, ensuring optimal growth for their crops.

Another critical feature is the AI-powered chatbot, which provides real-time consultation to farmers. This chatbot is powered by Google Gemini API (external), allowing farmers to receive immediate, personalized responses based on current environmental data and plant health. In addition, the system integrates Flask API (internal) for disease classification and growth prediction. The disease classification model, based on Convolutional Neural Networks (CNN) with Keras, classifies plant diseases by analyzing images, while the growth prediction model, utilizing Support Vector Machine (SVM) with XGBoost, predicts plant growth based on environmental data. These predictions help farmers take proactive actions to manage their crops efficiently. Furthermore, the Echarts gauges visualize key environmental metrics such as EC, temperature, humidity, nitrogen, phosphorus, potassium, and pH, giving farmers a clear, interactive view of their crops' status.

Finally, the successful performance of the system was validated through rigorous API testing using Postman, with all endpoints responding with a 200 OK status, confirming the backend system's reliability. The integration of disease classification, growth prediction, and environmental data monitoring has ensured that the system offers farmers a comprehensive solution for precision agriculture. The AI-powered chatbot and interactive Echarts gauges have further enhanced the system's usability, enabling farmers to monitor, assess, and manage their crops with greater efficiency. The system's ability to handle real-time data from multiple internal and external APIs and maintain stable performance under testing conditions proves its effectiveness in providing real-time, actionable insights to farmers.

3.1. Entity Relationship Diagram (ERD)

The Entity Relationship Diagram (ERD) provides a visual representation of the database structure used in the chili plant precision agriculture system, as shown in Figure 3. The users table acts as the central entity, linking to other tables via the user_id foreign key to ensure data is personalized and traceable. The chat_sessions and chat_histories tables manage interactions with the AI chatbot sessions group conversations by topic or time, while histories log detailed exchanges between users and the system. The crop_images table stores uploaded plant images along with predicted results, confidence scores, and descriptions for disease classification and growth stage analysis. Meanwhile, the growth_predictions table records structured data such as plant height, diameter, leaf count, week number, predicted growth grade, confidence level, and optional notes, enabling continuous monitoring of chili plant development.

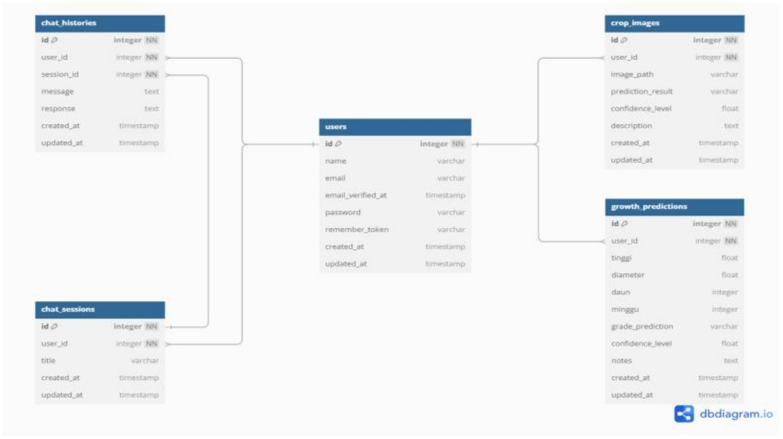

Figure 3. *Entity relationship diagram*

Based on Figure 3, this relational design ensures that all data ranging from chatbot interactions to sensor-based growth predictions is stored systematically and linked to specific users. This structure supports efficient data retrieval, accurate data attribution, and smooth integration with both internal and external APIs, including IoT-based data and machine learning outputs. Overall, the ERD supports critical system features such as real-time environmental monitoring, disease diagnosis, growth forecasting, and intelligent user interaction, ultimately empowering farmers with timely, data-driven insights.

3.2. System Architecture Diagram

The System Architecture Diagram illustrates the overall structure of the Precision Agriculture Dashboard, emphasizing how various components interact to deliver a seamless experience for the user. The diagram depicts the flow of data between the User (Farmer), Web Browser (Client Side), Laravel Web Server (Back-End), Internal and External APIs, and MySQL Database. The User interacts with the system through the Web Browser, which sends requests for weather data, plant health, and other queries to the Laravel Web Server. The server processes these requests and retrieves data from internal and external APIs, including the ThingSpeak API, which collects environmental sensor data via the IoT ESP-32 device, and the OpenWeatherMap API, which provides real-time weather data. Additionally, the Flask API powers disease classification and growth prediction models, while the Google Gemini API enables real-time, personalized consultations via the AI-powered chatbot. All relevant data is stored and managed within the MySQL Database, ensuring organized, efficient retrieval and supporting farmers with real-time, actionable insights for better farming decisions. This architecture is designed to optimize data flow, making the system a comprehensive and effective tool for precision agriculture, as shown in Figure 4.

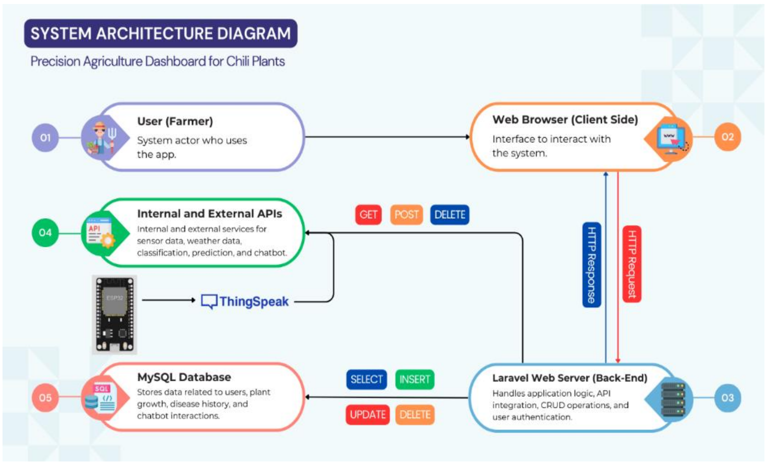
SYSTEM ARCHITECTURE DIAGRAM
Precision Agriculture Dashboard for Chili Plants
01
User (Farmer)
System actor who uses the app.
02
Web Browser (Client Side)
Interface to interact with the system.
04
Internal and External APIs
Internal and external services for sensor data, weather data, classification, prediction, and chatbot.
GET
POST
DELETE
HTTP Response
HTTP Request
ThingSpeak
05
MySQL Database
Stores data related to users, plant growth, disease history, and chatbot interactions.
SELECT
INSERT
UPDATE
DELETE
03
Laravel Web Server (Back-End)
Handles application logic, API integration, CRUD operations, and user authentication.

Figure 4. *System architecture diagram*

Based on Figure 4, this section discusses each of the components involved in the system's architecture and their relationships:

a. User (Farmer)

The User (Farmer) represents the main actor in the system, who interacts with the system via a web browser. The farmer uses the dashboard to access environmental information, interact with the AI-powered chatbot, and review plant health and growth predictions. This interaction is critical for providing farmers with the data needed for informed decision-making, ensuring their actions support crop health and optimal farming practices.

b. Web Browser (Client Side)

The Web Browser (Client Side) serves as the interface for farmers to interact with the system. It communicates with the back-end via HTTP requests, sending user inputs (e.g., weather queries, chatbot messages) to the server. The browser displays the server's responses, such as real-time environmental data, disease classification results, and growth predictions, to the user in a readable format, enhancing usability.

c. Laravel Web Server (Back-End)

The Laravel Web Server is the backbone of the system's back-end. It is responsible for handling all application logic, user authentication, and API integrations. The server processes incoming requests from the client (e.g., weather data, plant health queries) and fetches real-time data through APIs like OpenWeatherMap and ThingSpeak. It also handles CRUD operations with the MySQL Database, ensuring that user data and predictions are properly stored and updated.

d. Internal and External APIs

Internal and external APIs play a crucial role in providing the system with real-time data. The ThingSpeak API collects environmental sensor data such as temperature, humidity, pH, and other key metrics, which is provided by the IoT ESP-32 device. The IoT ESP-32, an Internet of Things (IoT) device, collects data from physical sensors such as temperature and humidity and sends this information to ThingSpeak API, which the system then uses for monitoring environmental conditions in real time. Additionally, the OpenWeatherMap API provides up-to-date weather information, helping farmers monitor external environmental conditions. The Google Gemini API powers the AI-powered chatbot, enabling real-time communication and personalized feedback for farmers. Furthermore, the Flask API is utilized for disease classification and growth prediction, integrating machine learning models to process images and environmental data for accurate results. Together, these internal and external APIs ensure that the system provides a comprehensive, real-time solution for precision agriculture.

e. MySQL Database

The MySQL Database stores all relevant data within the system, including user information, sensor data, disease classification, growth predictions, and chatbot interactions. It supports relational data management, ensuring efficient data organization and seamless integration across system components. This allows farmers to track and analyze data over time, including chatbot conversations, helping to support decision-making with both real-time and historical insights.

3.3. API Endpoint Response Testing

In this section, we present the results of the API endpoint response testing performed using Postman. This testing aimed to ensure that all API endpoints in the system are functioning correctly, providing the expected responses, and handling requests as intended. The endpoints tested include both public and protected APIs, which provide various functionalities such as logging in, retrieving environmental data, analyzing crop images, predicting plant growth, and interacting with the AI-powered chatbot. The system integrates both internal APIs, such as the Flask API for disease classification and growth prediction, and external APIs, including the Google Gemini API for the AI-powered chatbot, ThingSpeak API for environmental data, and OpenWeatherMap API for weather information. All tested endpoints returned a 200 OK status, confirming that the system's back-end is stable, reliable, and capable of processing real-time data accurately. The successful API response testing validates the robustness of the system and assures that it can be effectively used in a real-world farming environment. The following table summarizes the results of this testing, detailing each endpoint, its request type, expected response, and status code.

Table 3. *API endpoint response testing*

Endpoint URL	Request Type	Expected Response	Status Code
/api/login	POST	JSON	200 OK
/api/ weather	GET	JSON	200 OK
/api/metrics	GET	JSON	200 OK
/api/crop-images/ analyze	POST	JSON	200 OK
/api/growth/predict	POST	JSON	200 OK
/api/chatbot/send	POST	JSON	200 OK
/api/logout	POST	JSON	200 OK

Table 3 summarizes all the API endpoints tested during the validation process. Each endpoint's status code of 200 OK indicates that the system is functioning properly. This includes endpoints for various critical features such as user login, logout, retrieving weather and environmental data, analyzing crop images, predicting plant growth, and interacting with the AI-powered chatbot. The successful testing of these endpoints confirms that the backend system is stable and responsive, capable of handling real-time data and providing reliable information to farmers for decision-making. The consistent performance across all API endpoints ensures the system's readiness for deployment in real-world farming operations. For a more detailed analysis of each endpoint's functionality and response, please refer to the corresponding section on Response Testing for Each Endpoint.

a. Response Testing Endpoint: /api/login

The /api/login endpoint is responsible for authenticating the user and providing a bearer token for subsequent requests to protected endpoints. Upon a successful login attempt, the system generates a token that validates the user session and grants secure access to the system's features. The response confirms that the login was successful, returning a 200 OK status and the message "Login Berhasil!", along with the user data and the generated token. This token is essential for making authenticated requests to the system. Here is the relevant part of the response JSON received during testing:

```
{
  "message": "Login Berhasil!",
  "token":
"1|Mw74i8aEGD4g54PokZB4d3I4mLfwoTu36"R99a
b"c3"1b693",
  "user": {
    "id": 1,
    "name": "Petani Cabai",
    "email": "petani@cabai.com"
  }
}
```

This response verifies that the login functionality is working correctly and that the user can proceed with authenticated requests using the bearer token for accessing protected features.

b. Response Testing Endpoint: /api/weather

The /api/weather endpoint retrieves real-time weather data, which is critical for monitoring environmental conditions impacting chili plant growth. Upon successful testing, the system returns a 200 OK status and provides weather information such as temperature, humidity, and wind speed. This data is essential for farmers to adjust farming practices based on current weather conditions. The relevant weather data received during testing are summarized in Table 4.

Table 4. *API endpoint response testing - /api/weather*

Field	Value
Success	true
Weather Description	Awan Mendung
Weather Main	Clouds
Temperature (°C)	27.13
Feels Like (°C)	28.96
Humidity (%)	66
Wind Speed (m/s)	1.13
Country	ID (Indonesia)
Sunrise Time	1745967030 (Epoch)
City Name	Kota Bandung

This response shows that the weather data retrieval functionality works as expected, providing real-time information about temperature and humidity for the farmers to take appropriate action.

c. Response Testing Endpoint: /api/metrics

The /api/metrics endpoint provides key environmental metrics, such as Electrical Conductivity (EC), temperature, humidity, pH, nitrogen, phosphorus, and potassium levels. The successful response, with a status code of 200 OK, confirms that the system is processing and returning the required environmental data accurately. Below is the relevant data in table format, as presented in Table 5, which was received during testing:

Table 5. *API endpoint response testing - /api/metrics*

Sensor ID	EC	Temperature (°C)	Humidity (%)	pH	Nitrogen (ppm)	Phosphorus (ppm)	Potassiu m (ppm)
2573088	686	22.7	26	6.5	38	135	128
2573089	600	23.0	39.6	5.6	135	358	353
2573090	455	23.4	24.6	5.6	30	115	108

This data is important for providing real-time environmental information to the farmers, enabling them to monitor conditions like soil health, moisture, and essential nutrients to make timely adjustments for better crop care.

d. Response Testing Endpoint: /api/crop images/analyze

The /api/crop-images/analyze endpoint is used to analyze uploaded crop images and predict potential plant diseases. When a crop image is uploaded, the system uses machine learning models to classify the disease affecting the plant. The response from this endpoint includes a prediction of the disease (in this case, "Keriting Mozaik") along with a confidence level that indicates the accuracy of the prediction. In this case, the model predicts the disease with a high confidence level of 93.38%. The system uses this information to alert farmers to the possible disease affecting their plants, allowing them to take prompt action to manage plant health. Here is the relevant part of the response JSON received during testing:

```
{
    "success": true,
    "prediction": "Keriting Mozaik",
    "confidence": 93.37526559829712,
    "description": null
}
```

This response indicates that the crop image was successfully analyzed, and the disease prediction model has provided a reliable result that can assist farmers in making informed decisions about plant health.

e. Response Testing Endpoint: /api/growth/predict

The /api/growth/predict endpoint is used to predict the growth stage of chili plants based on environmental factors. This endpoint generates a growth prediction grade, which helps farmers understand the expected development of their crops. In this instance, the system predicts a growth grade of "A," indicating very good plant health. The confidence level of 99.92% reinforces the reliability of the prediction. Additionally, the "minggu" (week) field indicates the plant's growth stage, which in this case is week 5. The response also includes a note, which can provide additional context, such as "Percobaan prediksi" for testing purposes. Here is the relevant part of the response JSON:

```
{
    "success": true,
    "grade_prediction": "A",
    "confidence_level":
99.92234490394592,
    "minggu": 5,
    "notes": "Percobaan prediksi"
}
```

This response indicates that the system has successfully predicted the growth of the chili plants, and farmers can use this information to make informed decisions about plant care based on the growth stage.

f. Response Testing Endpoint: /api/chatbot/send

The /api/chatbot/send endpoint is used to send a message to the AI-powered chatbot, allowing the user (farmer) to receive a response based on their query. The chatbot is designed to provide real-time consultation on various aspects of chili plant farming, such as planting, care, pest control, fertilization, harvesting, post-harvest, and marketing. The response from the chatbot is designed to be informative, with detailed knowledge of chili farming. Upon receiving the query, the chatbot responds with personalized advice. Here is the relevant part of the response JSON received during testing:

```
{
    "success": true,
    "response": "Saya adalah chatbot
untuk pertanian cabai. Silakan ajukan
pertanyaan Anda tentang tanaman cabai."
"session_id": 1
}
```

This response confirms that the chatbot is providing valuable information on chili plant farming and is functioning as expected. The session ID confirms that this particular conversation was initiated and can be tracked for further interactions.

g. Response Testing Endpoint: /api/logout

The /api/logout endpoint is responsible for logging the user out of the system, ensuring that the user session is terminated securely. Upon a successful logout request, the system returns a confirmation message indicating that the user has successfully logged out and is no longer authenticated. The response confirms that the logout process was completed successfully with a 200 OK status and includes the message "Logout Berhasil!." This endpoint ensures that the user's session is properly invalidated, preventing unauthorized access to the system. Here is the relevant part of the response JSON received during testing:

```
{
    "message": "Logout Berhasil!”
}
```

This response verifies that the logout functionality works correctly, confirming that the user can successfully log out and that the system appropriately handles session termination.

Following the successful testing of the /api/logout endpoint, further performance testing was conducted for all the API endpoints using a load test to assess their performance under typical user interactions. During the performance test, a total of 10 virtual users (VUs) were used over a test duration of 5 minutes, with a fixed load profile to simulate consistent usage. The performance tests confirmed that all endpoints performed excellently, with no error rates observed, demonstrating the system's stability and its ability to handle simultaneous user requests effectively.

First, let's discuss the response time performance. As shown in Figure 5, the response time trends during the test period demonstrate that the average response time remained consistently low, with peaks observed during the initial user interactions. However, as the test continued, the response time stabilized, showcasing the system's ability to handle multiple requests efficiently over time. The 90th, 95th, and 99th percentiles also indicated that the response times remained within acceptable limits for real-time applications.

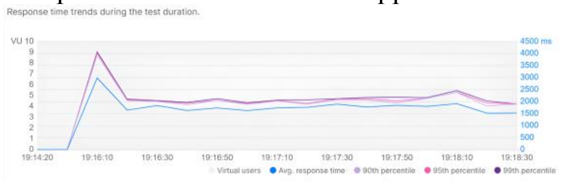

Figure 5. *Response time trends*

Next, we examine the throughput performance. As shown in Figure 6, the throughput during the test remained steady, with slight fluctuations in the rate of requests per second. These fluctuations were due to varying user interactions, but overall, the system maintained a high throughput rate, ensuring that it could handle a significant number of requests per second without any degradation in performance. This is a critical factor in ensuring that the system can scale to meet the demands of real-world usage.

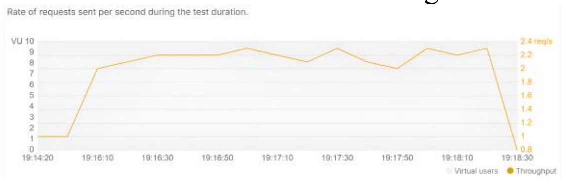

Figure 6. *Throughput trends*

These results confirm that the system's backend is highly responsive, capable of efficiently processing multiple concurrent API requests, and ready for deployment in production environments.

In addition, preliminary usability testing of the dashboard's front-end was conducted by actual farmers using Maze. Feedback from these end-users indicates that the dashboard offers an intuitive, responsive, and interactive user interface, facilitating effective interaction with real-time data and AI-powered features. The dashboard effectively supports farmers in making data-driven decisions by providing real-time environmental information and AI-powered insights, thereby enhancing crop management and productivity. While the results are encouraging, further analysis and refinement are ongoing to continuously improve the overall user experience.

CONCLUSION

In conclusion, this research has successfully developed a precision agriculture system for chili plant monitoring by integrating various internal and external APIs into a robust back-end system. The system enables real-time data processing and provides crucial functionalities such as disease classification, growth prediction, environmental monitoring, and AI-powered chatbot consultation. The integration of external APIs like ThingSpeak, OpenWeatherMap, and Google Gemini, alongside internal APIs such as Flask, has enabled the system to offer real-time insights, aiding farmers in making data-driven decisions for better crop management. The use of the Prototyping Method allowed for quick development and iterative testing, ensuring that the final system met the needs of the users. The back-end system's scalability, reliability, and ability to handle large datasets in real-time were demonstrated through performance testing. The system successfully processed multiple API requests, with the average response time for the API endpoints stabilizing around 2000 ms after an initial peak of 4500 ms during the first interactions. This indicates the system's efficiency in handling requests even under load, ensuring timely delivery of critical information to farmers. Additionally, all API endpoints consistently returned a 200 OK status during the tests, confirming the system's stability and reliability. However, this research has some limitations. The current system is specifically tailored for chili plant monitoring and may require adjustments to scale effectively to other crops or broader agricultural applications. Furthermore, while the back-end system is robust, the front-end usability and user experience aspects need further exploration to enhance accessibility and ease of use for farmers with varying levels of technical skills.

For future work, efforts will focus on improving the prediction models by expanding the range of diseases detected and incorporating additional environmental factors to refine growth predictions and accuracy. Moreover, the development of a mobile application will be prioritized to increase accessibility for farmers in the field. The integration of real-time alert systems to notify users of critical conditions or disease outbreaks is also planned, aiming to provide timely interventions and improve crop management outcomes.

ACKNOWLEDGEMENT

The authors would like to express their gratitude for the RIIM-LPDP grant provided by the National Research and Innovation Agency of the Republic of Indonesia (BRIN), the Indonesia Endowment Fund for Education (LPDP). This support is outlined in the agreements numbered 37/II.7/HK/2023. The authors also express their gratitude to Telkom University for supporting the publication of this research.

REFERENCES

[1] G. Singh and S. Sharma, "Enhancing precision agriculture through cloud based transformative crop recommendation model," *Sci. Rep.*, vol. 15, no. 1, p. 9138, Mar. 2025, doi: 10.1038/s41598-025-93417-3.

[2] A. Monteiro, S. Santos, and P. Gonçalves, "Precision Agriculture for Crop and Livestock Farming—Brief Review," *Animals*, vol. 11, no. 8, p. 2345, Aug. 2021, doi: 10.3390/ani11082345.

[3] A. D. Putra Laksamana, H. Fakhrurroja, and D. Pramesti, "Developing a Labeled Dataset for Chili Plant Health Monitoring: A Multispectral Image Segmentation Approach with YOLOv8," in *2024 International Conference on Computer, Control, Informatics and its Applications (IC3INA)*, Oct. 2024, pp. 440–445. doi: 10.1109/IC3INA64086.2024.10732221.

[4] D. Khairani, T. Rosyadi, A. Arini, I. L. Rahmatullah, and F. F. Antoro, "Enhancing Speech-to-Text and Translation Capabilities for Developing Arabic Learning Games: Integration of Whisper OpenAI Model and Google API Translate," *J. Tek. Inform.*, vol. 17, no. 2, pp. 203–212, Oct. 2024, doi: 10.15408/jti.v17i2.41240.

[5] H. Fakhrurroja, A. Rofi Hidayatullah, D. Pramesti, and N. Ismail, "Classification of Diseases in Chili Plants Using the SVM Method: Development and Implementation," in *2024 12th International Conference on Information*

and Communication Technology (ICoICT), Aug. 2024, pp. 493–499. doi: 10.1109/ICoICT61617.2024.10698555.

[6] R. P. Pratama, H. Fakhurroja, H. Bangkit, and D. Pramesti, “Prediction of NPK Fertilizer in Chili Plants Using SARIMA Model,” in *2024 International Conference on ICT for Smart Society (ICISS)*, Sep. 2024, pp. 1–6. doi: 10.1109/ICISS62896.2024.10751445.

[7] M. J. Page *et al.*, “The PRISMA 2020 statement: an updated guideline for reporting systematic reviews,” *BMJ*, p. n71, Mar. 2021, doi: 10.1136/bmj.n71.

[8] F. Schneider, J. Swiatek, and M. Jelali, “Detection of Growth Stages of Chilli Plants in a Hydroponic Grower Using Machine Vision and YOLOv8 Deep Learning Algorithms,” Apr. 18, 2024, *Engineering*. doi: 10.20944/preprints202404.1243.v1.

[9] J. Arévalo-Royo, F.-J. Flor-Montalvo, J.-I. Latorre-Biel, R. Tino-Ramos, E. Martínez-Cámara, and J. Blanco-Fernández, “AI Algorithms in the Agrifood Industry: Application Potential in the Spanish Agrifood Context,” *Appl. Sci.*, vol. 15, no. 4, p. 2096, Feb. 2025, doi: 10.3390/app15042096.

[10] G. R. Babu, M. Gokuldhev, and P. S. Brahmanandam, “Integrating IoT for Soil Monitoring and Hybrid Machine Learning in Predicting Tomato Crop Disease in a Typical South India Station,” *Sensors*, vol. 24, no. 19, p. 6177, Sep. 2024, doi: 10.3390/s24196177.

[11] C. E. Hachimi, S. Belaqziz, S. Khabba, B. Sebbar, D. Dhiba, and A. Chehbouni, “Smart Weather Data Management Based on Artificial Intelligence and Big Data Analytics for Precision Agriculture,” *Agriculture*, vol. 13, no. 1, p. 95, Dec. 2022, doi: 10.3390/agriculture13010095.

[12] A. Upadhyay *et al.*, “Deep learning and computer vision in plant disease detection: a comprehensive review of techniques, models, and trends in precision agriculture,” *Artif. Intell. Rev.*, vol. 58, no. 3, p. 92, Jan. 2025, doi: 10.1007/s10462-024-11100-x.

[13] R. Real, C. Snider, M. Goudswaard, and B. Hicks, “Dimensions of Knowledge in Prototyping: A Review and Characterisation of Prototyping Methods and Their Contributions to Design Knowledge,” *Proc. Des. Soc.*, vol. 1, pp. 1303–1312, Aug. 2021, doi: 10.1017/pds.2021.130.

[14] Z. Subecz, “Web-development with Laravel framework,” *Gradus*, vol. 8, no. 1, pp. 211–218, 2021, doi: 10.47833/2021.1.CSC.006.

[15] I. Šušter and T. Ranisavljević, “Optimization of MySQL Database,” *J. Process Manag. New Technol.*, vol. 11, no. 1–2, pp. 141–151, 2023.

[16] P. P. Kore, M. J. Lohar, M. T. Surve, and S. Jadhav, “API Testing Using Postman Tool,” *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 10, no. 12, pp. 841–843, Dec. 2022, doi: 10.22214/ijraset.2022.48030.