IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Mathematical Modelling of Engineering Problems
Vol. 13, No. 3, March, 2026, pp. 483-492
Journal homepage: http://iieta.org/journals/mmep

CNN-Based Mobile Application for Banana Ripeness Classification: Development and Performance Evaluation

R. Rizal Isnanto[1], Rahmat Gernowo[1], Bellia Dwi Cahya Putri[2], Fathin Zhafira Fauzi[2], Agustiyar[1*]

[1] Doctoral Program of Information Systems, Postgraduate School, Universitas Diponegoro, Semarang 50241, Indonesia
[2] Department of Computer Engineering, Faculty of Engineering, Universitas Diponegoro, Semarang 50241, Indonesia

Corresponding Author Email: agustiyar@students.undip.ac.id

https://doi.org/10.18280/mmep.130303

Received: 11 November 2025
Revised: 2 January 2026
Accepted: 10 January 2026
Available online: 10 April 2026

Keywords:
banana ripeness classification, convolutional neural network, mobile vision application, agricultural image analysis, system performance evaluation

ABSTRACT

This study develops and evaluates a mobile application for banana ripeness classification based on a convolutional neural network (CNN). The proposed system enables users to capture banana images using a smartphone camera and automatically determine ripeness stages through image-based inference. A CNN model was trained using a curated dataset consisting of 1,200 labeled images representing five ripeness categories. The system was further assessed through functionality, compatibility, and load performance testing to examine its operational stability under varying user demands. Experimental results show that under moderate workloads (≤ 100 concurrent users), the application-maintained response times ranging from 10.5 to 17.3 ms with no observed errors. Under stress conditions (500 concurrent users), the response time increased to over 110 ms and the error rate reached 81.4%, indicating significant performance degradation at extreme loads. Overall, the results demonstrate that the proposed system provides reliable performance under typical usage conditions and offers a practical, non-destructive approach for rapid banana ripeness assessment in agricultural and post-harvest management applications.

1. INTRODUCTION

Bananas (Musa spp.) are among the most widely consumed fruits worldwide, both in fresh and processed forms, because of their high nutritional value [1-3]. Their nutritional composition varies across cultivars, including differences in carbohydrate, calorie, vitamin, and water content. For instance, Ambon bananas have been reported to contain higher carbohydrate levels than several other local varieties [4-6].

Banana composition is also influenced by the ripening stage. In unripe fruit, carbohydrates are mainly present as resistant starch, whereas during ripening, starch is converted into simple sugars such as glucose, fructose, and sucrose [7-9]. These compositional changes affect the taste, texture, and nutritional properties of bananas.

Numerous studies have shown that banana ripeness is closely associated with biochemical transformations, particularly the enzymatic conversion of starch into soluble sugars such as glucose, fructose, and sucrose. These changes affect the sweetness, texture, and caloric value of the fruit as ripening progresses. Increasing sugar content during ripening is also accompanied by observable external changes, especially in peel color and texture [10].

In practice, conventional ripeness assessment often depends on destructive laboratory analyses or spectrophotometric methods, which are not suitable for routine consumer use or mobile implementation. Recent advances in non-destructive techniques, including hyperspectral imaging, RGB image analysis, and electronic sensing, have enabled visual features to be linked with internal quality indicators, including sugar-related changes [11]. Building on this foundation, the present study proposes a convolutional neural network (CNN)-based mobile application for classifying banana ripeness from peel images.

Accurate ripeness assessment is crucial for evaluating banana quality, as fruit maturity has been widely recognized as a determining factor in harvesting decisions and overall fruit quality in modern agricultural systems [12]. However, traditional methods, such as manual inspection based on peel color and texture, are subjective, error-prone, and may cause mechanical damage to the fruit. These limitations highlight the need for non-destructive and automated alternatives that can provide more consistent and reliable results.

In response to this need, the Ba-Nanas! application was developed as a mobile-based solution for automated banana ripeness classification. By integrating a CNN model with a mobile phone camera, the application enables users to capture banana images and obtain ripeness analysis results directly through their mobile devices. The system was implemented using the React Native framework to support cross-platform accessibility, while the back-end Application Programming Interface (API) connected the trained CNN model with the banana image dataset.

In mobile application development, system reliability is determined not only by the correctness of functionalities but also by the ability to maintain stable performance under

diverse operating conditions. Therefore, testing constitutes a critical stage in ensuring software quality [13-16]. Functional testing verifies that all features operate according to specification, whereas non-functional testing evaluates performance and compatibility aspects. Performance evaluation commonly includes load and stress testing to measure throughput, error rate, and response time under concurrent user requests. Compatibility testing further examines application operability across devices with different specifications.

Although previous studies have explored CNN-based image classification for agricultural applications, limited attention has been given to the integration of banana ripeness analysis with systematic mobile performance evaluation. In particular, the combined use of load testing, stress testing, and compatibility testing in a CNN-based agricultural mobile application remains limited. This integration enables a more comprehensive assessment of both algorithmic capability and application robustness.

Non-destructive quality assessment of agricultural produce has advanced considerably through imaging and spectral analysis techniques. Hyperspectral imaging and related spectral methods, such as Visible Spectroscopy/Near-Infrared Spectroscopy (VIS/NIR) and Fourier Transform Infrared Spectroscopy (FTIR), have been widely reported as effective tools for assessing banana ripeness and internal quality attributes without damaging the fruit [17, 18]. However, these approaches typically require specialized and costly equipment, which limits their scalability and practical adoption, particularly among smallholder farmers. In contrast, the present study adopts an image-based approach using widely accessible mobile devices and CNN-based RGB image classification, positioning the proposed system as a more practical and scalable alternative for non-destructive banana ripeness assessment.

The proposed system does not infer biochemical quantities, such as glucose or caloric values, directly from RGB images. Instead, the CNN is used solely to classify banana ripeness into predefined categories based on visual characteristics of the peel.

This study aims to evaluate the performance and compatibility of the Ba-Nanas! application in terms of reliability, responsiveness, and stability. Specifically, the contributions of this study are threefold: (i) the design and implementation of a CNN-based mobile system for banana ripeness classification, (ii) systematic evaluation of functionality, performance, and compatibility under different user load conditions, and (iii) assessment of system responsiveness and stability across multiple mobile devices. Overall, the study is expected to demonstrate the feasibility of AI-assisted mobile tools for practical banana ripeness assessment in agricultural and post-harvest contexts.

2. LITERATURE REVIEW

2.1 Mobile development frameworks

React Native is a cross-platform framework that enables mobile application development using the JavaScript programming language. It offers simplicity, reusable components, and a live reload feature that accelerates the development process [19]. In agricultural informatics, cross-platform development is essential to reach diverse stakeholders such as farmers, distributors, and consumers who often use heterogeneous mobile devices. Several studies have demonstrated the feasibility of React Native in agricultural applications, ensuring accessibility and usability across Android and iOS platforms [20]. TypeScript, a superset of JavaScript, is frequently employed to enhance code reliability and maintainability during application development [21].

2.2 Programming languages

Python is among the most widely adopted programming languages in artificial intelligence and computer vision research. Its rich ecosystem of libraries, such as TensorFlow, Keras, and PyTorch, makes it highly suitable for agricultural applications, including fruit quality assessment, ripeness classification, and yield prediction. For example, Tapia-Mendez et al. [22] used CNNs to classify the ripeness of fruits and vegetables based on image data, while Ashfaq et al. [23] applied CNN for crop yield prediction using environmental and remote sensing data. Similarly, CNN-based approaches have also been successfully employed to classify agricultural products based on visual quality attributes, such as maturity or postharvest condition, using RGB image features [24]. In the Ba-Nanas system, Python is utilized for model training and back-end services, enabling efficient integration of CNN-based classification with mobile interfaces.

2.3 System integration with Representational State Transfer Application Programming Interface

Representational State Transfer (RESTful) API provides a lightweight and scalable architecture for communication between mobile clients and servers. Using Hypertext Transfer Protocol (HTTP) protocols and JavaScript Object Notation (JSON) formatting, RESTful APIs allow seamless integration of mobile applications with machine learning models hosted in the cloud [25]. In the agricultural domain, web service APIs including RESTful approaches have been increasingly applied to support the integration of field data, image repositories, and AI models, enabling real-time accessibility and interoperability in smart farming systems [26].

2.4 Performance evaluation tools

Performance testing is critical to ensure that an application can maintain reliability under expected workloads. Apache JMeter is a widely used open-source tool designed to simulate concurrent user requests and evaluate server responses [27]. In the context of IoT-based agricultural applications, performance evaluation is increasingly emphasized to guarantee responsiveness and stability. Recent reviews have highlighted JMeter among the commonly employed tools for application-layer testing in IoT systems, particularly to analyze throughput, error rates, and response latency [28]. For example, Minani et al. [29] evaluated mobile applications for estimating soil properties and emphasized the importance of response time and stability under various usage scenarios, demonstrating the relevance of performance testing in agricultural mobile solutions.

2.5 Software testing approaches

Testing plays a central role in software development to ensure both functionality and robustness [30]. Automated testing frameworks for mobile applications have been systematically reviewed to help development teams maintain

quality and reduce defects, particularly in usability and system reliability. In the agricultural mobile-technology domain, recent trend surveys underline the increasing importance of mobile app quality, reliability, and performance as key factors in technology adoption among farmers and stakeholders [31].

3. RESEARCH METHODS

3.1 Research process

This study evaluates the Ba-Nanas! application through a comprehensive software testing approach that focuses on functionality, performance, and compatibility. The research process consists of system requirements identification, user interface (UI/UX) design, system architecture design, front-end and back-end implementation, API deployment, and application testing. The testing phase includes black box testing, load testing, stress testing, and compatibility testing to ensure the reliability and robustness of the system under various operational conditions.

3.2 Identify system requirements

System requirements identification covered both functional and non-functional requirements. Functional requirements were defined to describe the main operations supported by the application, whereas non-functional requirements were specified to determine performance and compatibility expectations. The functional requirements included image acquisition through the camera or gallery, banana ripeness classification using a CNN model, display of prediction results across five predefined ripeness categories, storage of previous analysis records, and deletion of historical analysis data.

The non-functional requirements focused on performance and compatibility. In terms of performance, the application was expected to provide responsive analysis under normal operating conditions. In terms of compatibility, the application was intended to operate across mobile devices with different hardware specifications and screen characteristics.

3.3 User interface design

Application interface design begins with the creation of a wireframe as a preliminary design that illustrates the layout of key elements such as buttons, menus, and images before moving on to the final design stage. The wireframe serves to organize the display structure so that the development process is more focused, and then it is developed into a color design with the addition of icons and more attractive and interactive visual elements. Each page has a different function, including the Splashscreen page as the opening display with the application logo, the Home page containing the main menu and feature navigation, the Scan page for scanning or selecting images from the gallery, the Analysis Results page that displays the scan results, and the History page that stores and displays previous analysis results.

3.4 System design

The CNN model used in this study consists of four convolutional layers, each followed by Rectified Linear Unit (ReLU) activation and max-pooling operations. The model ends with two fully connected layers and was trained on a dataset of labeled banana images representing five ripeness levels. The CNN was used solely to classify banana ripeness into predefined categories based on visual characteristics of the peel.

Figure 1 illustrates the use-case structure of the Ba-Nanas! application and the interactions between the user and the system. The main features are accessible through the Home, Scan, and History pages. The Home page displays introductory information and provides navigation to the main functions. The Scan page allows users to capture or upload banana images for ripeness analysis, whereas the History page stores previous analysis results and supports record deletion.

To support system development, the application workflow was modeled using an activity diagram, as shown in Figure 2. The workflow begins when the user opens the application and accesses the Home page, where introductory information and recent analysis history are displayed. From this page, the user may delete selected history records or navigate to the Scan or History page. On the Scan page, the user can either capture a photo using the camera or import an image from the gallery. The system then validates the selected image. If the image contains a banana, it is processed by the CNN model and the predicted ripeness result is displayed and stored in the History page. Otherwise, the application displays a notification indicating that the selected image is not a banana. The user may then reload the analysis or navigate to the History page to review previous scan results.

From an implementation perspective, the mobile application was developed using React Native to support cross-platform accessibility. The backend was implemented through an API that connected the application interface, the trained CNN model, and the database for storing prediction history. This architecture enabled image submission, ripeness inference, result display, and history management within a unified mobile environment.

After designing the system, the next step is to implement the system in code based on the identified needs and the established system design. In terms of appearance, this application has several main pages that have been designed based on user needs. The Home page displays animated information and analysis history, and allows users to delete historical data. The Scan page allows users to take pictures or upload them from the gallery, as well as analyze the ripeness of bananas. However, if the uploaded image is not a banana, the system cannot process the image.

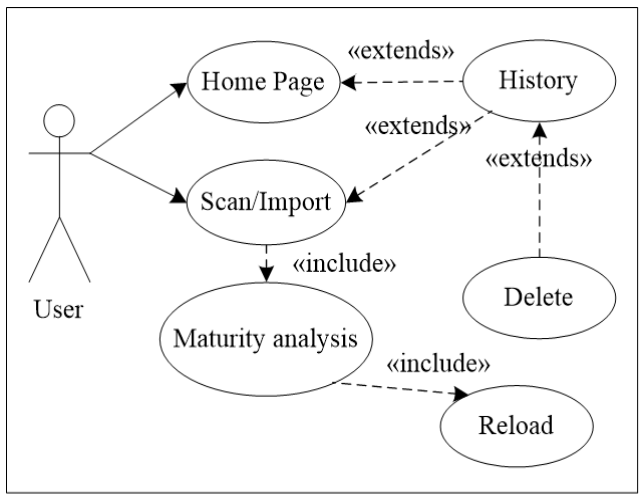

Figure 1. Use case diagram of Ba-Nanas! application

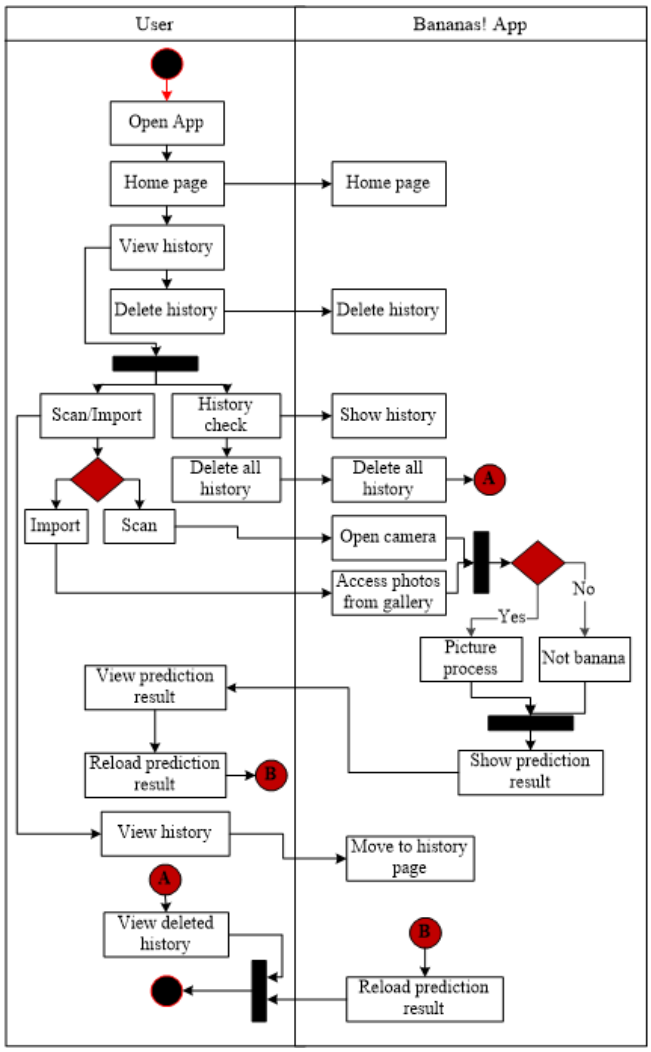

Figure 2. Ba-Nanas! application activity diagram

The History page is designed to store and display analysis results in the form of banana type, ripeness category, and ripeness value. The next step is back-end design. At this stage, the model that has been trained using the CNN algorithm will be integrated using the Flask API. The goal is to enable mobile devices to send banana images uploaded by users to the server. The server will process the images and return the predicted ripeness of the bananas in JSON format. The back-end implementation in this system utilizes Flask to build a RESTful API that enables communication between the front-end and back-end via the HTTPS protocol.

The CNN model is stored in h5 format and loaded into Flask using the TensorFlow library. Uploaded banana images will be analyzed to determine ripeness based on predetermined categories, namely unripe, half-ripe, ripe, overripe, and almost rotten. The analysis results are stored in the Firebase Realtime Database for later access. The server can handle POST, GET, and DELETE commands sent by the application. After the design phase is complete, the Flask API is hosted using CloudVPS Rumah Web so that it can be accessed online.

3.5 Testing scenarios

This study focuses on evaluating system-level performance metrics, including response time and error rate, under varying user loads. It should be emphasized that the reported error rate represents system-level request failures rather than CNN misclassification, as the primary objective of this study is system performance evaluation.

3.5.1 Black box testing scenario

Black box testing was conducted to verify whether all functional features of the Ba-Nanas! application operate according to the specified requirements. The testing focused on core application features, including image acquisition, ripeness analysis, result visualization, history management, and error handling. Each test case evaluated expected outputs without considering internal code structure, ensuring that the system functions correctly from the end-user perspective.

3.5.2 Load testing scenario

Load testing was conducted to simulate realistic usage patterns of the Ba-Nanas! application under expected operational conditions. Three scenarios were defined to represent light, moderate, and upper-bound user concurrency levels, using 25, 50, and 100 simulated users, respectively, in line with common benchmarking practices for mobile systems [32]. Each scenario applied a 10-second ramp-up period to emulate gradual user access and avoid artificial traffic spikes, thereby providing a more realistic representation of application usage. Performance evaluation was carried out using Apache JMeter, and the recorded metrics included average response time, throughput, and error rate. These metrics were selected to capture the responsiveness, processing capacity, and reliability of the system under increasing workloads. This approach also allowed the system to be evaluated progressively across different levels of concurrent demand. Overall, the testing framework was designed to assess system stability and performance under normal and anticipated usage conditions, consistent with prior studies on mobile application benchmarking [33]. The load testing scenarios used in this study are summarized in Table 1.

Table 1. Load testing scenarios

Scenario	Number of Users	Ramp-up Period (seconds)	Loop Count	Purpose
Scenario 1	25	10	1	Light expected load
Scenario 2	50	10	1	Moderate expected load
Scenario 3	100	10	1	Upper-bound expected load

3.5.3 Stress test scenario

Stress testing was conducted to evaluate the system's behavior when subjected to workloads exceeding its expected operational capacity. This test aims to identify performance degradation, failure points, and system limitations.

In this scenario, as shown in Table 2, 500 concurrent users were generated with a rapid ramp-up period of 5 seconds, intentionally creating a traffic surge that exceeds normal operating conditions. This configuration simulates sudden spikes in user requests and allows observation of throughput collapse, error escalation, and response time saturation.

The stress testing results are used to determine the current operational limits of the Ba-Nanas! application and to identify areas requiring backend optimization and scalability improvements.

Table 2. Stress test scenario

Scenario	Number of Users	Ramp-up Period (seconds)	Loop Count	Purpose
Stress Scenario	500	5	1	Beyond-capacity stress condition

3.5.4 Compatibility testing scenario

Compatibility testing was performed to ensure that the Ba-Nanas! application operates correctly across mobile devices with different hardware specifications, screen resolutions, and Android versions. The application was tested on smartphones with low, medium, and high specifications to evaluate interface consistency, responsiveness, and functional stability.

This testing aims to confirm that the application can be reliably used across heterogeneous mobile environments without crashes, functional errors, or significant performance degradation.

4. RESULTS AND DISCUSSION

4.1 Black box testing

Black box testing was conducted to verify that all core features of the Ba-Nanas! application operated according to the specified requirements. The black box testing results confirmed that all core features of the Ba-Nanas! application operated according to their expected behavior.

As shown in Table 3, all functionalities on the Home page were executed successfully, including the automatic animation of banana-related information, navigation between tabs, and deletion of history items. These results indicate that the interface responded correctly to user input.

Table 3. Homepage testing

Scenario	Expected Results	Conclusion
The user opens the Application.	Can display a Home page with automatic animation containing fun facts about bananas and displaying a scan history list.	Success
Users see animated information, and they can manually scroll through the animation.	Animation moves to the next item.	Success
Users view scan history.	Banana data information is displayed correctly (banana type, category, ripeness value, and ripeness prediction).	Success
The user presses the delete button on the History page.	Related history items removed from the list.	Success
The user accesses another tab, namely scan.	Navigation successfully directed to the appropriate page.	Success
Users can access another tab, namely History.	Navigation successfully directed to the appropriate page.	Success

Table 4 summarizes the testing results for the Scan page, including image capture, gallery upload, ripeness analysis, result display, and invalid-input handling. All scenarios produced the expected outputs, indicating that the interaction between the front-end interface, the CNN model, and the API functioned correctly. The system also handled non-banana images appropriately by displaying relevant error notifications.

Table 5 presents the testing results for the History page. All scenarios were completed successfully, including viewing and deleting previous analysis records.

Overall, the black box testing results indicate that the main functions of the Ba-Nanas! application operated consistently across all primary pages and were suitable for subsequent performance and compatibility evaluation.

Table 4. Scanned page testing

Scenario	Expected Results	Conclusion
Users can select images from the gallery.	The application opens the gallery and allows users to select images.	Success
Users can take pictures using the camera.	The application opens the camera to take a new picture.	Success
Users cannot perform image analysis without selecting an image.	A notification appears stating that the user must select an image first.	Success
Users can press the maturity analysis button after selecting an image.	The application sends images to the API and displays the analysis results.	Success
Users can view prediction results.	The application identifies images, and the prediction results displayed include banana type, category, ripeness value, and prediction.	Success
The user presses the “reload” button after analysis.	The prediction results are deleted, and the application display returns to its initial state before the analysis (images and prediction results are gone).	Success
The user presses the “View History” button.	The application displays a list of analysis results.	Success
Users take or upload images other than bananas.	A notification appears saying, “Sorry, the image you uploaded is not a banana. Please try again.”	Success

Table 5. History page testing

Scenario	Expected Results	Conclusion
The user presses the “history” button if there is historical data.	The application displays a list of analysis results history.	Success
Users press the “view history” button if there is no history data.	Displaying the message “No history available”.	Success
The user deleted the history list.	A notification appears stating that “History data has been successfully deleted.”	Success

4.2 Load testing

Three scenarios were run in load testing. The first scenario was conducted under light load conditions with 5 trials. The second scenario was conducted under moderate load conditions with 5 trials. The third scenario was conducted under upper-bound load conditions with 5 trials.

4.2.1 Scenario 1

Scenario 1 was conducted under light load conditions with 5 trials and produced the following results. Figure 3 shows that the throughput value fluctuates slightly with an upward trend from 10.05 kB/s in the first test to 13.81 kB/s in the fifth test. This indicates that the system is able to process user requests more efficiently after several tests, signifying good server stability in handling light loads.

Figure 4 shows that the error rate remained at 0% throughout the testing, meaning that there were no delivery or request failures during the process. This indicates that the system ran stably without network disruptions or server errors under light load conditions.

Meanwhile, Figure 5 shows response times varying between 6.103 ms and 9.015 ms. The highest value occurred in the second test (9.015 ms), while the lowest was in the fourth test (6.103 ms). Despite slight fluctuations, the overall response time is still relatively fast and consistent, indicating that the application performs well and does not experience a significant decline in responding to user requests.

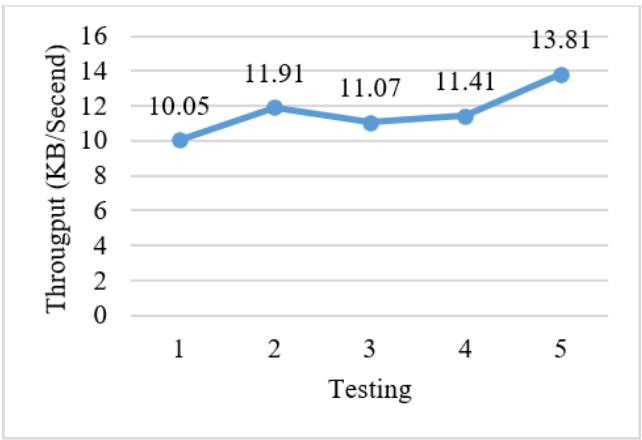

Figure 3. Throughput at load testing first scenario

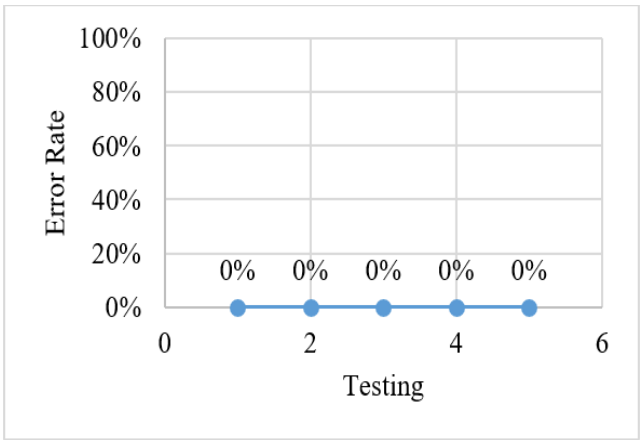

Figure 4. Error rate at load testing first scenario

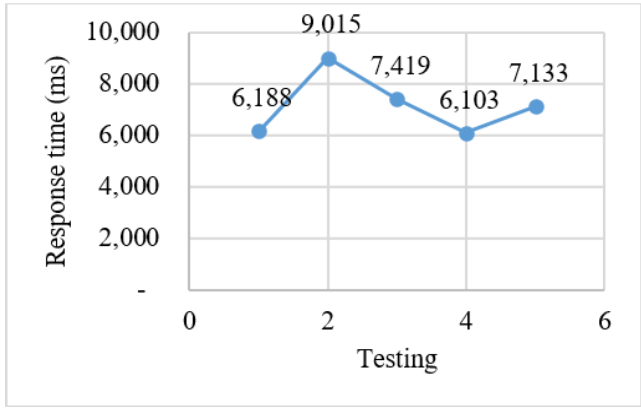

Figure 5. Response time at load testing first scenario

From the results of this test, it can be concluded that under a light load (25 users), the Ba-Nanas! application showed stable performance with increased throughput, zero error rate, and relatively low and consistent response times. This indicates that the system is already optimized in handling a limited number of users without experiencing disruptions or performance degradation.

4.2.2 Scenario 2

Scenario 2 was conducted under moderate load conditions with 5 trials. Figure 6 shows that throughput values increased consistently from 10.05 kB/s in the first test to 13.81 kB/s in the fifth test. This trend indicates that the application was still able to process requests well, even though the number of users doubled compared to the first scenario.

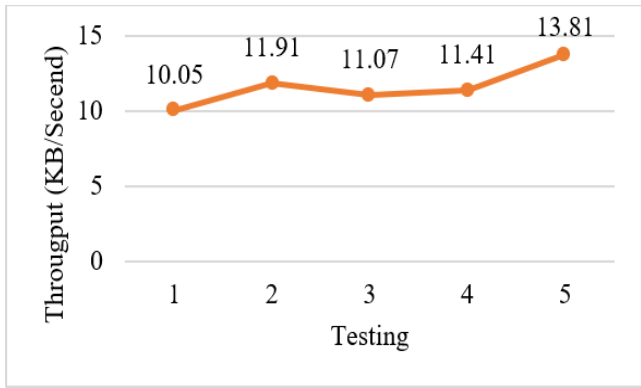

Figure 6. Throughput at load testing the second scenario

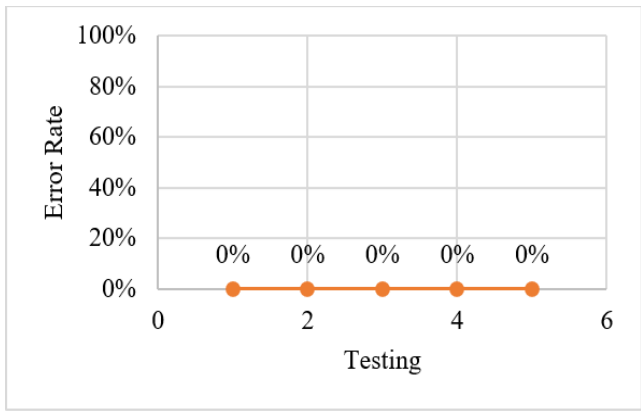

Figure 7. Error rate at load testing the second scenario

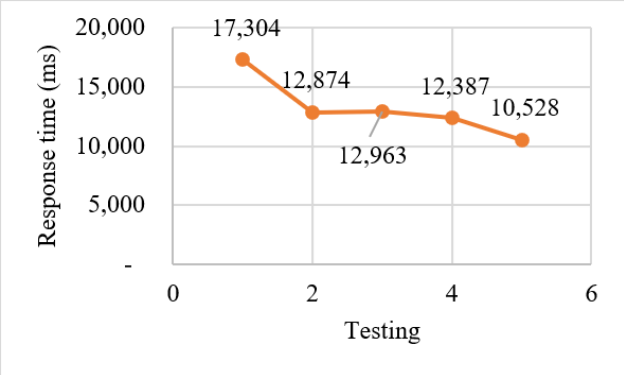

Figure 8. Response time at load testing the second scenario

Figure 7 shows that the error rate remained at 0% throughout the testing, indicating no request failures or system disruptions during the process. This indicates good server stability in handling medium user loads.

Meanwhile, Figure 8 shows fluctuations in response time, which tended to decrease from 17.304 ms in the first test to 10.528 ms in the fifth test. This decrease illustrates an increase in system efficiency, possibly due to cache optimization and connection management during repeated testing.

From the results of this test, it can be concluded that under moderate load (50 users), the Ba-Nanas! application continues to show stable performance with increased throughput, zero error rate, and significantly reduced response time. These results indicate that the system has good adaptability and efficiency in handling an increase in the number of users without a decline in performance, making it suitable for use in operational conditions with moderate user load.

4.2.3 Scenario 3

The third scenario was conducted under upper-bound load conditions with 5 trials. Figure 9 shows that the throughput value fluctuated slightly but remained within a stable range, between 11.65 kB/s and 13.01 kB/s. The lowest value occurred in the fourth test (11.65 kB/s), while the highest value occurred in the second test (13.01 kB/s). These fluctuations indicate that the system is still capable of maintaining its data processing capabilities even under high pressure.

Figure 10 shows that the error rate is generally at 0%, except for the fourth test, which experienced a small increase of 1%. This indicates that most requests were processed successfully, and the errors that occurred were relatively minor and did not interfere with the overall stability of the system.

Meanwhile, Figure 11 shows a significant increase in response time, which peaked at 24.028 ms in the fourth test. The lowest response time occurred in the second test (21.393 ms), and decreased again to 21.195 ms in the fifth test. This pattern indicates that when the number of users reaches the maximum limit, the system begins to show signs of resource saturation, such as request queues and increased latency.

From the results of this test, it can be concluded that under upper-bound load conditions (100 users), the Ba-Nanas application can still operate with a fairly good level of stability, although performance begins to decline compared to light and medium loads. Throughput values remain relatively stable, the error rate only increases marginally, but response time shows a significant spike, indicating that the system is beginning to reach its optimal capacity limit. Therefore, an upgrade in server specifications or optimization on the back-end is necessary to maintain performance when the number of users increases substantially.

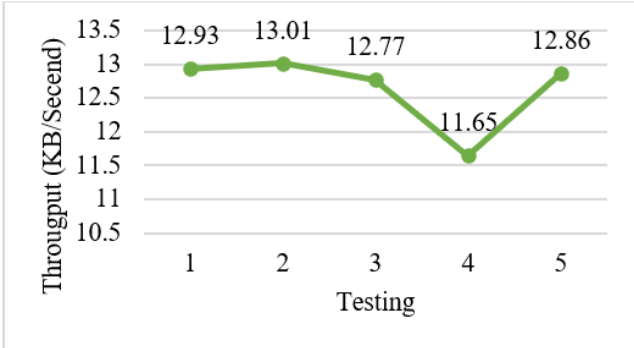

Figure 9. Throughput at load testing the third scenario

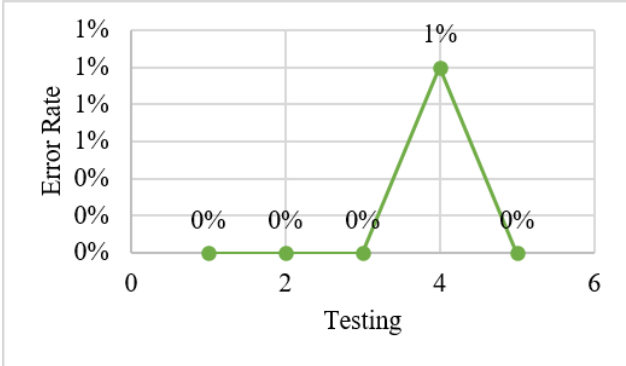

Figure 10. Error rate at load testing the third scenario

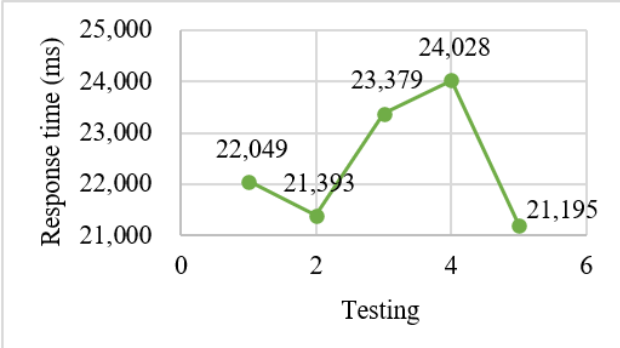

Figure 11. Response time at load testing the third scenario

Compared with similar mobile agricultural applications, such as the Estimating Soil Properties app [27], the Campeche Soils App [34], and mobile plant disease detection systems [35], the Ba-Nanas! system demonstrates comparable performance stability under moderate loads. However, unlike those systems, Ba-Nanas! performs computationally intensive CNN inference on image data, which results in slightly higher latency during stress testing. This outcome highlights the trade-off between model complexity and real-time responsiveness in AI-based agricultural mobile applications.

4.3 Stress testing

In the stress testing scenario, only one scenario was used with a capacity of 500 users, conducted with 5 trials. Figure 12 shows relatively stable throughput values in the first four tests, ranging from 1,637 to 1,757 kB/s, but a sharp decline to 1,135 kB/s in the fifth test. This decline indicates that when the system load reaches the saturation point, the application's ability to process user requests decreases significantly due to resource limitations.

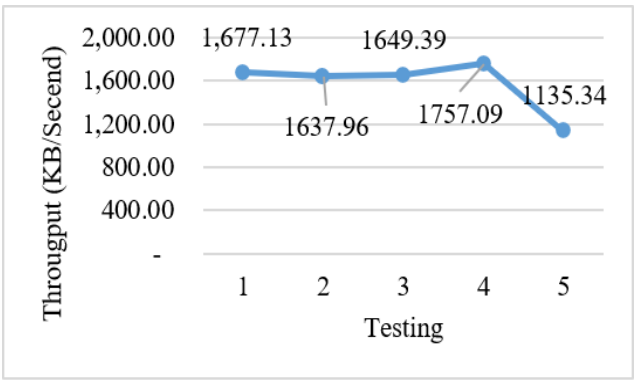

Figure 12. Throughput at stress testing

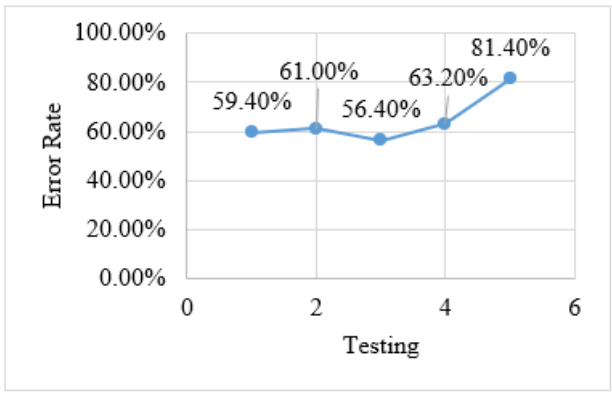

Figure 13. Error rate at stress testing

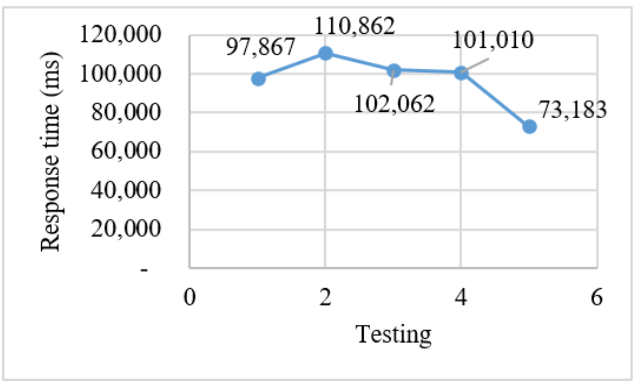

Figure 14. Response time at stress testing

Figure 13 shows a high error rate throughout the tests, starting at 59.40% and increasing to 81.40% in the fifth test. This increase indicates that the heavier the system load, the more requests fail to be processed. This condition indicates significant system degradation when the maximum capacity of the application is exceeded.

Meanwhile, Figure 14 shows very high response time values, ranging from 73.183 ms to 110.862 ms. Response time increased sharply in the first two tests, then decreased slightly in the final test. These fluctuations indicate that the system attempted to stabilize performance under pressure, but still experienced significant delays compared to normal conditions.

The results of this stress testing confirm that the Ba-Nanas application has a certain capacity limit in handling spikes in demand. When the number of users or requests exceeds the threshold, throughput decreases dramatically, the error rate increases significantly, and response time becomes very high. This indicates that the system still needs to be optimized in terms of server scalability, thread pool management, and API efficiency in order to remain stable under extreme loads and not experience performance degradation when facing high traffic.

4.4 Compatibility testing

Compatibility testing was conducted to evaluate whether the Ba-Nanas! application could operate properly across devices with different hardware specifications, screen resolutions, and Android versions. This evaluation also examined interface responsiveness and layout consistency across different screen sizes. The tested device specifications are presented in Table 6.

Table 6. Testing device specifications

Device Type	Screen Dimensions	Screen Resolution	Screen Size	Android Version
SamsungA 12	164 × 75.8 × 8.9 mm	720 × 1600 (HD+)	IPS 6.5-inch	13
SamsungM 21	159 × 75.1 × 8.9 mm	2340 × 1080 (FHD+)	6.4-inch	12
Vivo v21	161.24 × 74.37 × 7.38 mm	2400 × 1080 (FHD+)	6.44-inch	11

Note: HD+: High Definition Plus; FHD+: Full High Definition Plus; IPS: In-Plane Switching.

The compatibility test results showed several visual differences across devices due to variations in screen resolution and aspect ratio. Nevertheless, the Ba-Nanas! application remained functional on all tested devices without crashes or major performance issues. These findings indicate acceptable compatibility across the evaluated devices, although further refinement is still needed to improve visual consistency across different screen sizes.

5. CONCLUSION

This study developed and evaluated the Ba-Nanas! application as a CNN-based mobile system for banana ripeness classification. The performance evaluation showed that the application functioned reliably under light to moderate load conditions, whereas noticeable degradation occurred under upper-bound and stress scenarios, reflecting limitations in the current backend configuration. These findings indicate that the application is suitable for small-scale deployment or pilot use, but it still requires backend optimization and scalability improvements before being applied in high-demand environments. Thus, the present results should be viewed as indicators of the system's current operational limits rather than as confirmation of production readiness.

Future improvements should focus on optimizing the CNN model using lightweight deployment frameworks, enhancing backend scalability through more capable cloud infrastructure, and refining interface consistency across devices with different screen resolutions. Further research should also examine CNN classification accuracy under more diverse imaging conditions to complement the system-level performance evaluation reported in this study.

ACKNOWLEDGMENT

This research was financially supported by The Postgraduate School, Universitas Diponegoro, Indonesia, through the "RKAT Universitas Diponegoro Tahun Anggaran 2025" scheme.

REFERENCES

[1] Vu, N.D., Hang, N.T.T.N., Mi, K.D.Q., Anh, N.H.T., Pham, B.A. (2025). Exploring the nutritional composition, physicochemical properties, and biological characteristics of mature banana varieties (Musaceae). Food Chemistry: X, 28: 102594. https://doi.org/10.1016/j.fochx.2025.102594

[2] Choudhury, N., Nickhil, C., Deka, S.C. (2023). Comprehensive review on the nutritional and therapeutic value of banana by-products and their applications in food and non-food sectors. Food Bioscience, 56: 103416. https://doi.org/10.1016/j.fbio.2023.103416

[3] Zou, F., Tan, C., Zhang, B., Wu, W., Shang, N. (2022). The valorization of banana by-products: Nutritional composition, bioactivities, applications, and future development. Foods, 11(20): 3170. https://doi.org/10.3390/foods11203170

[4] Rai, I.N., Nyoman, N., Mayadewi, A., Wiraatmaja, I.W., Astiari, N.K.A. (2023). Fruit morphology and nutritional composition of different genome groups of six bananas cultivars from Bali Island. Caraka Tani: Journal of Sustainable Agriculture, 38(2): 421-432. https://doi.org/10.20961/carakatani.v38i2.74941

[5] Susan, A., Agustina, A. (2023). Utilization of Ambon banana (Musa acuminata) peel flour as a prebiotic in the Nile Tilapia (Oreochromis Niloticus). Egyptian Journal of Aquatic Biology and Fisheries, 27(5): 973-986. https://doi.org/10.21608/ejabf.2023.323545

[6] Eko Mulyo, G.P., Sukowati, S.A., Sulaeman, A., Rahmat, M., Saleky, Y.W., Syarief, O., Fauziyah, R.N. (2022). Analysis of liability and protein content of soybean biscuits with Ambon banana as an alternative to emergency food for the elderly. Slovak Journal of Food Sciences, 16: 568-578. https://doi.org/10.5219/1753

[7] Cheng, Y., Huang, P., Chan, Y., Chiang, P., et al. (2024). Investigate the composition and physicochemical properties attributes of banana starch and flour during ripening. Carbohydrate Polymer Technologies and Applications, 7: 100446. https://doi.org/10.1016/j.carpta.2024.100446

[8] Huang, P.H., Cheng, Y.T., Lu, W.C., Chiang, P.Y., et al. (2024). Changes in nutrient content and physicochemical properties of cavendish bananas var. pei chiao during ripening. Horticulturae, 10(4): 384. https://doi.org/10.3390/horticulturae10040384

[9] Phillips, K.M., McGinty, R.C., Couture, G., Pehrsson, P.R., McKillop, K., Fukagawa, N.K. (2021). Dietary fiber, starch, and sugars in bananas at different stages of ripeness in the retail market. PLoS One, 16(7): e0253366. https://doi.org/10.1371/journal.pone.0253366

[10] Cordenunsi-Lysenko, B.R., Nascimento, J.R.O., Castro-Alves, V.C., Purgatto, E., Fabi, J.P., Peroni-Okyta, F.H.G. (2019). The starch is (not) just another brick in the wall: The primary metabolism of sugars during banana ripening. Frontiers in Plant Science, 10: 391. https://doi.org/10.3389/fpls.2019.00391

[11] Maduwanthi, S.D.T., Marapana, R.A.U.J. (2017). Biochemical changes during ripening of banana: A review. International Journal of Food Science and Nutrition, 2(5): 166-169.

[12] Martínez-Mora, O., Capuñay-Uceda, O., Caucha-Morales, L., Sánchez-Ancajima, R., Ramírez-Morales, I., Córdova-Márquez, S., Cuenca-Mayorga, F. (2025). Artificial vision-based dual CNN classification of banana ripeness and quality attributes using RGB Images. Processes, 13(7): 1982. https://doi.org/10.3390/pr13071982

[13] Basir, M.S., Buckmaster, D., Raturi, A., Zhang, Y. (2024). From pen and paper to digital precision: A comprehensive review of on-farm recordkeeping. Precision Agriculture, 25(5): 2643-2682. https://doi.org/10.1007/s11119-024-10172-7

[14] Barbie, A., Hasselbring, W., Hansen, M. (2023). Enabling automated integration testing of smart farming applications via digital twin prototypes. In 2023 IEEE Smart World Congress (SWC), Portsmouth, United Kingdom, pp. 1-8. https://doi.org/10.1109/SWC57546.2023.10449240

[15] Antonelli, L., Camilleri, G., Torres, D., Zaraté, P. (2024). User acceptance test for software development in the agricultural domain using natural language processing. Journal of Decision Systems, 33(4): 913-936. https://doi.org/10.1080/12460125.2023.2229579

[16] Kumar, V., Sharma, K.V., Kedam, N., Patel, A., Kate, T.R., Rathnayake, U. (2024). A comprehensive review on smart and sustainable agriculture using IoT technologies. Smart Agricultural Technology, 8: 100487. https://doi.org/10.1016/j.atech.2024.100487

[17] Chu, X., Miao, P., Zhang, K., Wei, H., et al. (2022). Green banana maturity classification and quality evaluation using hyperspectral imaging. Agriculture, 12(4): 530. https://doi.org/10.3390/agriculture12040530

[18] Aline, U., Bhattacharya, T., Faqeerzada, M.A., Kim, M.S., Baek, I., Cho, B.K. (2023). Advancement of non-destructive spectral measurements for the quality of major tropical fruits and vegetables: A review. Frontiers in Plant Science, 14: 1240361. https://doi.org/10.3389/fpls.2023.1240361

[19] Duncan, B., Bulanon, D.M., Bulanon, J.I., Nelson, J. (2024). Development of a cross-platform mobile application for fruit yield estimation. AgriEngineering, 6(2): 1807-1826. https://doi.org/10.3390/agriengineering6020105

[20] Jošt, G., Taneski, V. (2025). State-of-the-art cross-platform mobile application development frameworks: A comparative study of market and developer trends. Informatics, 12(2): 45. https://doi.org/10.3390/informatics12020045

[21] Bogner, J., Merkel, M. (2022). To type or not to type? A systematic comparison of the software quality of JavaScript and typescript applications on GitHub. In Proceedings of the 19th International Conference on Mining Software Repositories, pp. 658-669. https://doi.org/10.1145/3524842.3528454

[22] Tapia-Mendez, E., Cruz-Albarran, I.A., Tovar-Arriaga, S., Morales-Hernandez, L.A. (2023). Deep learning-based method for classification and ripeness assessment of fruits and vegetables. Applied Sciences, 13(22): 12504. https://doi.org/10.3390/app132212504

[23] Ashfaq, M., Khan, I., Shah, D., Ali, S., Tahir, M. (2025). Predicting wheat yield using deep learning and multi-source environmental data. Scientific Reports, 15(1): 26446. https://doi.org/10.1038/s41598-025-11780-7

[24] Zhu, H.L., Huang, Y.W., An, Z.K., Zhang, H., Han, Y.Y., Zhao, Z.H., Li, F.F., Zhang, C., Hou, C.C. (2024). Assessing radiometric calibration methods for multispectral UAV imagery and the influence of illumination, flight altitude and flight time on reflectance, vegetation index and inversion of winter wheat AGB and LAI. Computers and Electronics in Agriculture, 219: 108821. https://doi.org/10.1016/j.compag.2024.108821

[25] Pautasso, C. (2014). RESTful web services: Principles, patterns, emerging technologies. Web Services Foundations, pp. 31-51. https://doi.org/10.1007/978-1-4614-7518-7_2

[26] Choudhary, V., Guha, P., Pau, G., Mishra, S. (2025). An overview of smart agriculture using internet of things (IoT) and web services. Environmental and Sustainability Indicators, 26: 100607. https://doi.org/10.1016/j.indic.2025.100607

[27] Sinclair, R., Nodi, S., Kabir, M.A. (2024). Evaluating mobile applications for estimating soil properties: Quality of current apps, limitations and future directions. Computers and Electronics in Agriculture, 216: 108527. https://doi.org/10.1016/j.compag.2023.108527

[28] Xu, J., Gu, B., Tian, G. (2022). Review of agricultural IoT technology. Artificial Intelligence in Agriculture, 6: 10-22. https://doi.org/10.1016/j.aiia.2022.01.001

[29] Minani, J.B., Sabir, F., Moha, N., Guéhéneuc, Y.G. (2024). A systematic review of IoT systems testing: Objectives, approaches, tools, and challenges. IEEE Transactions on Software Engineering, 50(4): 785-815. https://doi.org/10.1109/TSE.2024.3363611

[30] Berihun, N.G., Dongmo, C., Van der Poll, J.A. (2023). The applicability of automated testing frameworks for mobile application testing: A systematic literature review. Computers, 12(5): 97. https://doi.org/10.3390/computers12050097

[31] Huang, X., Zhao, Y., Guo, T.K., Mao, X.M. (2024). Enhancing SWAP simulation accuracy via assimilation of leaf area index and soil moisture under different irrigation, film mulching and maize varieties conditions. Computers and Electronics in Agriculture, 218: 108625. https://doi.org/10.1016/j.compag.2024.108625

[32] Liu, P., Li, Y. (2022). Response time evaluation of mobile applications combining network protocol analysis and information fusion. Information and Software Technology, 145: 106838. https://doi.org/10.1016/j.infsof.2022.106838

[33] Bolanowski, M., Ćmil, M., Starzec, A. (2024). New model for defining and implementing performance tests. Future Internet, 16(10): 366. https://doi.org/10.3390/fi16100366

[34] Gallegos, Á., Duran, J.A., Palma-López, D.J., Zavala-Cruz, J., López-Castañeda, A., Bautista, F. (2025). A mobile application for smartphones with soil information: Improving connectivity in terms of soil security. Soil Security, 19: 100187. https://doi.org/10.1016/j.soisec.2025.100187

[35] Siddiqua, A., Kabir, M.A., Ferdous, T., Ali, I.B., Weston, L.A. (2022). Evaluating plant disease detection mobile applications: Quality and limitations. Agronomy, 12(8): 1869. https://doi.org/10.3390/agronomy12081869