Scalable Microservices Architecture for Face Recognition-Based Employee Attendance Systems

1st Ridwan Setiawan*
Department of Computer Science
Institut Teknologi Garut
Garut, Indonesia
ridwan.setiawan@itg.ac.id

2nd Wawan Hermawan
Department of Computer Science
Institut Teknologi Garut
Garut, Indonesia
2106144@itg.ac.id

3rd Asep Trisna Setiawan
Department of Computer Science
Universitas Bandar Lampung
Bandar Lampung, Indonesia
Asep.iot@ubl.ac.id

*Corresponding author: ridwan.setiawan@itg.ac.id
Received: 2025-08-26; **Accepted:** 2025-10-21

***Abstract*—We present a face-recognition-based employee attendance system built on a microservices architecture and integrated with the external Worker AI (LSKK) inference API. The design separates camera I/O, verification, persistence, and presentation into independently deployable services, enabling targeted scaling and resilient operation through asynchronous queues. The system was developed using Rapid Application Development (RAD) and evaluated via black-box testing that covered authentication, camera and AI data views, filtering and pagination, reporting, and employee CRUD. The results show conformance to specifications: the interface renders the expected outputs, and the cooldown policy effectively prevents duplicate entries, while the separation of history (raw) and history_ai (verified) supports traceability and clean reporting. These findings indicate that combining microservices with API-based face recognition offers a practical and maintainable alternative to RFID-based workflows with fewer operational frictions. Limitations include the use of an external inference API (model configuration and thresholds are outside our control) and testing within a single organizational setting. Future work will focus on operational measurements of the deployed pipeline, particularly end-to-end latency under load spikes and queue formation, as well as monitoring misread/error rates to inform model improvements.**

Keywords— Black-Box Testing; Employee Attendance System; Face Recognition; Microservices Architecture; Rapid Application Development

I. Introduction

Information and communication technology has made significant advancements, particularly in managing employee attendance. People are increasingly using web-based system innovations, artificial intelligence, and biometric technologies like facial recognition to accurately and efficiently record attendance [1], [2]. Microservices architecture is a modern approach in software development. In contrast to monolithic systems that combine all features in a single application, microservices divide the system into separate services that can run independently but are connected to each other via an Application Programming Interface (API) [3].

Microservices architecture has advantages in scalability, modularity, and ease of system maintenance [4], [5]. Meanwhile, face recognition technology is widely used in security and attendance systems because it is able to identify individuals automatically without requiring physical contact based on facial characteristics. This technology can reduce the potential for fraud and increase the efficiency of attendance recording [6], [7].

Based on observations at PT LSKK, the attendance system used still uses Radio Frequency Identification (RFID) cards. Although it is considered modern, this system has several shortcomings, such as the possibility of employees forgetting to bring their cards, the potential for absenteeism, and development constraints because every feature change can affect the entire system. This study shows the limitations of the system in handling the ever-growing needs of the company [8], [9].

Previous research shows that microservices architecture can make the system flexible and keep it running even if parts of the service are updated or repaired [10], [11], [12]. Face recognition technology, utilizing various algorithms like Eigenfaces, Fisherfaces, and LBPH, has demonstrated accuracy in face detection [1].

This research uses Rapid Application Development (RAD), which is considered effective because it allows rapid iteration, prototyping, and direct user involvement during the development process [13], [14]. The attendance system developed utilizes the advantages of microservices architecture and face recognition technology as employee identification methods. With this approach, the attendance system that is built is expected to be able to provide a more flexible solution that is easy to develop for the needs of modern industry.

II. Methods

This research adopts the Rapid Application Development (RAD) method, an incremental (multi-level) software development technique. The RAD model is a life-cycle strategy intended to deliver software faster and with better quality than conventional methods [15], [16]. According to [14], the RAD model has three main stages: requirement planning, design workshop, and implementation. The first version of the system is developed iteratively by dividing the overall project into several incremental releases. Fig. 1 provides an overview of the proposed system architecture.

A. *Requirement Planning*

This stage identifies problems where the author analyzes the attendance system currently used at PT LSKK, which is based on Radio Frequency Identification (RFID) cards, through direct observation of the attendance process and interviews with related parties. Based on the results of observations and discussions with PT LSKK, the author then identified system requirements, which were grouped into functional and non-functional requirements.

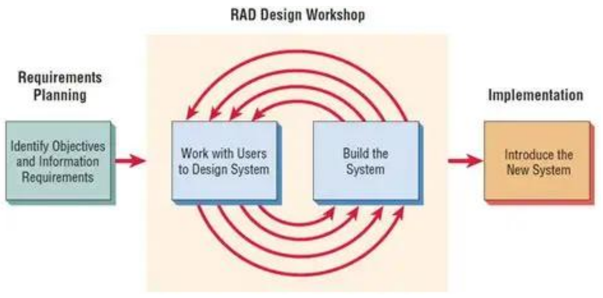


Fig. 1. Rapid Application Development Cycle

B. *Workshop Design*

This stage aims to identify solutions and choose the best one. Next, develop a business process design and a programming design for the collected data, utilizing the Unified Modeling Language (UML).

C. *Implementation*

This stage builds a microservices-based employee attendance system using NestJS for backend services and ReactJS for the frontend interface and integrates face recognition services into the system. After the system was completed, the author conducted functional testing using the black box testing method to ensure that each function ran according to the established specifications, without checking the internal structure of the program code [17].

III. Results and Discussion

The employee attendance system was developed using the Rapid Application Development (RAD) approach and a microservices architecture to ensure components could be scaled and maintained independently. Initial analysis with PT LSKK showed limitations of the existing RFID-based system—reliance on physical cards (lost/exchanged), potential for buddy punching (use of cards by others), vulnerability to UID duplication/cloning, which reduces data trust, and administrative burden for card issuance and blocking [18], [19]. The system workflow is shown in Fig. 2: employees swipe their RFID cards, the system verifies against the database, and attendance records are written if the data is valid.

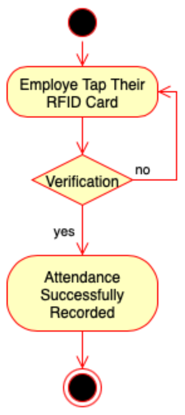


Fig. 2. Previous system flow

To overcome these limitations, we propose a face recognition-based system. Fig. 3 shows the use case diagram with two external actors, Employee and Admin. The core use case, Perform Attendance, requires two internal processes—Detect Employee by Camera and Face Classification—before attendance recording is authorized, thus reducing the likelihood of incorrect or duplicate entries within a short interval. On the governance side, the admin runs the Login, View Dashboard, View Camera Data, Manage Employee Data, View AI Data, and View Report use cases for monitoring and reporting. The face classification component is provided as an external Worker AI (LSKK) API accessed through an inference endpoint; the payload contains an encoded image and device metadata, while the response returns candidate identities and verification scores. The cooldown policy per employee-device pair is implemented to prevent duplicate entries, and message queue configuration is used to limit stale frames. Functional requirements include facial image capture, metadata transmission via RabbitMQ, raw data logging in history, identity verification via Worker AI, and history presentation through a web interface; non-functional requirements encompass device specifications (CPU/RAM/camera) and the software stack (TypeScript/NestJS, ReactJS, MongoDB, RabbitMQ).

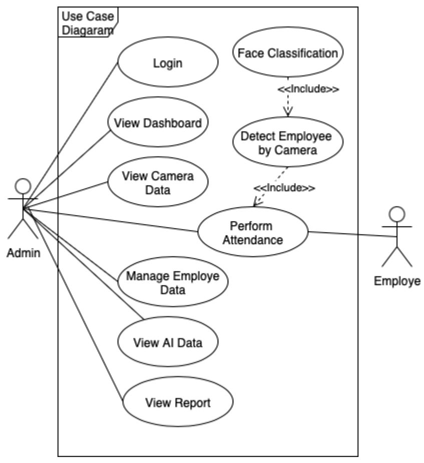


Fig. 3. Use Case Diagram

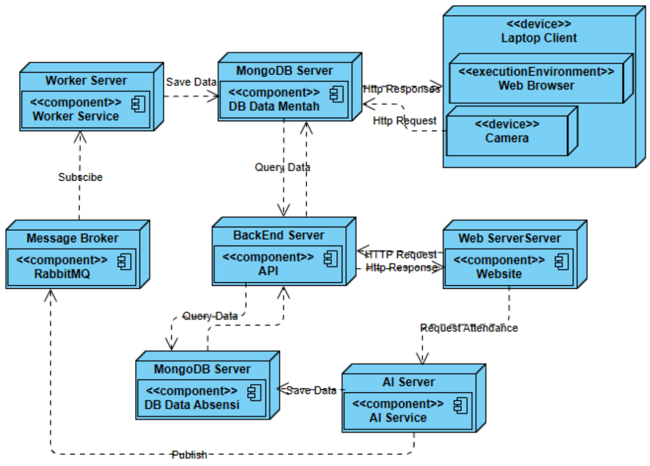


Fig. 4. Microservices System Architecture

Fig. 4 illustrates the microservices architecture connecting the client side, application services, message broker, AI Worker (LSKK), and two separate databases.

From the client laptop (browser with camera access), the face image and metadata are sent to the web server. The request is then forwarded to the backend server (API) for validation and payload formatting. For asynchronous processing, the API publishes events to RabbitMQ. The Worker Service subscribes to the queue, stores the raw data in MongoDB (Raw Data DB), and then calls the AI Worker (LSKK) via the inference endpoint for identity verification. Valid classification results are published back to the service path and recorded in MongoDB (Attendance Data DB). The web server takes the summary and presents it to the user through the web interface. This system is designed using a microservices architecture, which allows each service to be developed independently using different technologies, programming languages, and databases according to the needs of each component [20].

This design separates the camera I/O path from the inference process so that peak loads can be handled without degrading the user experience. HTTP request/response communication is used for synchronous interaction between the browser, web server, backend server, and AI worker (LSKK). The publish/subscribe pattern in RabbitMQ handles presence flows that require buffering and retries. Separating the two databases—Raw Data for the audit trail and Attendance Data for verified records—makes it easier to track verification failures while maintaining operational query performance.

Fig. 5 illustrates the system class diagram, which includes utility classes `DB_data_mentah` and `DB_absensi` for database connection, along with business classes like `Data_kamera`, `Data_AI`, and `Employees` to store their respective data. Additionally, there's an interface that displays a list and details of camera data, AI data, attendance reports, and employee information. All of these components work together to manage and display data within the system.

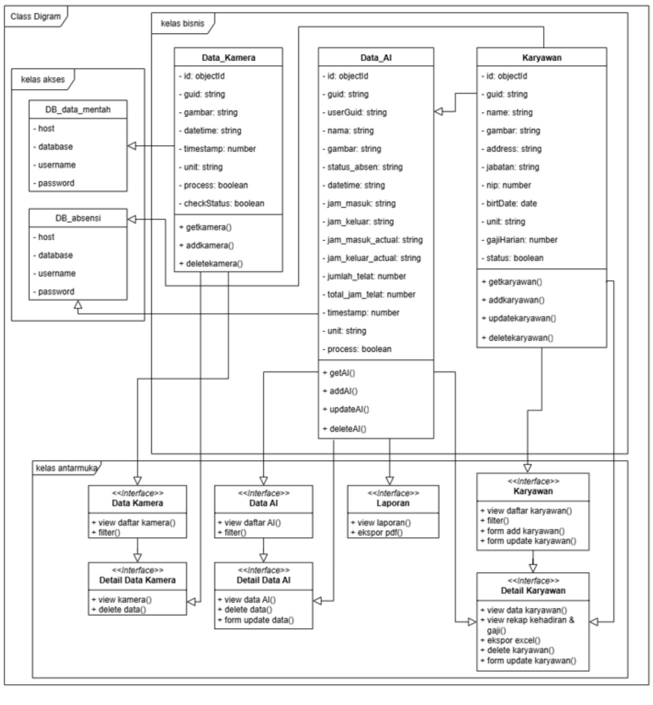


Fig. 5. System Class Diagram

Code 1 is the implementation of the API endpoint recognize_frame on the Worker AI (LSKK) service for face detection and verification. This endpoint receives a POST request containing a base64-encoded image, decodes it into an image format, and then analyzes it to generate candidate identities along with their scores. Each result is verified with a cooldown policy to prevent duplicate entries within a short time interval; only results that pass the cooldown are processed as attendance and sent back to the application in JSON format so that the backend module can record them in verified storage.

Code 1. Facial Recognition Process

```
def recognize_frame():
    data = request.get_json()
    image_data_b64 = data['image'].split(',')[1]
    img_bytes = base64.b64decode
      (image_data_b64)
    img = cv2.imdecode(np.frombuffer
      (img_bytes, np.uint8), cv2.IMREAD_COLOR)

    results = face_analyzer.analyze_image
      (img,user_details_map)
    for person in results:
        guid = person.get('guid')
        cooldown_ok = _check_cooldown(guid)
        if cooldown_ok:
            person['presence_sent'] =
      _process_detection(person, img, lat, lon)
    return jsonify({"results": results})
```

The integration of this flow is shown in Fig. 6, which displays the face recognition results when the attendance system is operated. The camera captures the employee's face, the application calls the recognize_frame API, and the response containing the GUID is mapped to the employee's name in the attendance module. The name was then displayed directly on the face area outlined in a green box, indicating successful detection. When verification is valid, an attendance record is created; if it doesn't meet the verification threshold or is blocked by a cooldown, the system doesn't add a new entry, keeping the history clean of duplicates.

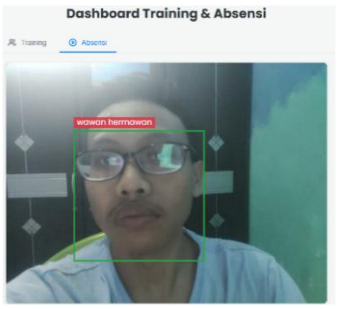
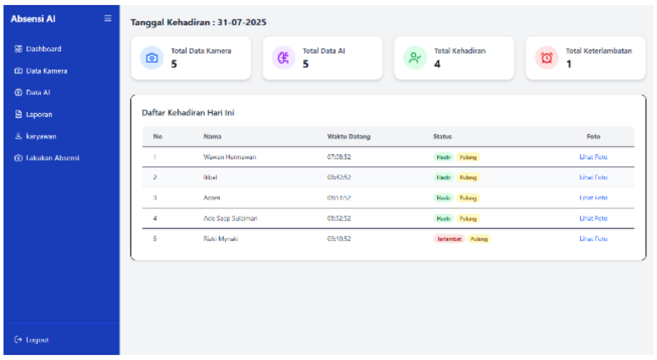


Fig. 6. Face recognition integration

The implementation results interface shows the integration of all components after integration with Worker AI (LSKK). Fig. 7 shows a dashboard that summarizes daily operational conditions concisely but informatively: the number of entries received from the cameras, the number of verification results, and the check-in and check-out history for the current day. The presence of this summary is important not merely as a display but as proof that the processing pipeline from image capture to inference calling to attendance record writing runs consistently within the system. At the same time, operational indicators (e.g., total successful and rejected verifications) provide early warning signals about the health of the service and the quality of the data being processed.



Tanggal Kehadiran : 31-07-2025

Total Data Kamera	Total Data AI	Total Kehadiran	Total Keterlambatan
5	5	4	1

Daftar Kehadiran Hari Ini

No	Nama	Waktu Datang	Status	Foto
1	Wawan Hermawan	07:08:52	Hadir Pulang	Lihat Foto
2	Ikbal	08:42:52	Hadir Pulang	Lihat Foto
3	Adam	08:51:52	Hadir Pulang	Lihat Foto
4	Ade Saep Sulaiman	08:52:52	Hadir Pulang	Lihat Foto
5	Rizki Mynaki	09:10:52	Terlambat Pulang	Lihat Foto

Fig. 7. Dashboard View

Fig. 8 illustrates how the separation of storage space supports data consistency and traceability. Here, the raw entries from the camera device are presented as a complete audit trail, including device identity and timestamps. This practice aligns with a database design that separates history (raw data before verification) from history AI (verified records for daily operations). With this separation, the cause of any discrepancies is traced; if an entry fails verification, the raw data is still available for incident analysis without contaminating the operational summary.

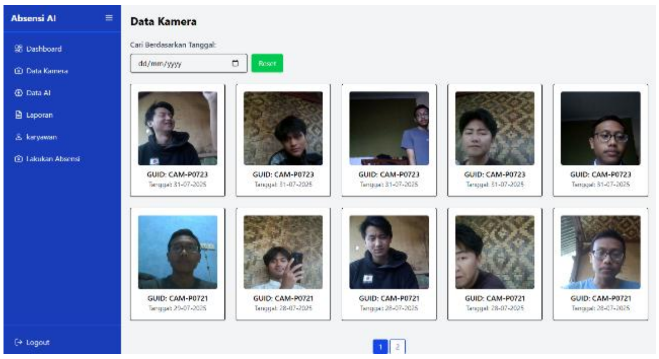


Fig. 8. Camera Data Display

The impact of integrating the classification service is shown in Fig. 9. The employee's name is displayed directly on the blue-boxed area of the face, which serves as a valid detection marker. This visual interpretation is not merely cosmetic but a direct representation of the inference endpoint output that has passed verification thresholds and cooldown policies. Thus, only truly unique and valid attempts are recorded as legitimate attendance; repeated attempts within short intervals do not add new entries, thereby maintaining data quality. This mechanism addresses the weaknesses of the physical card approach, particularly the potential for misuse, without adding administrative burden on the user side.

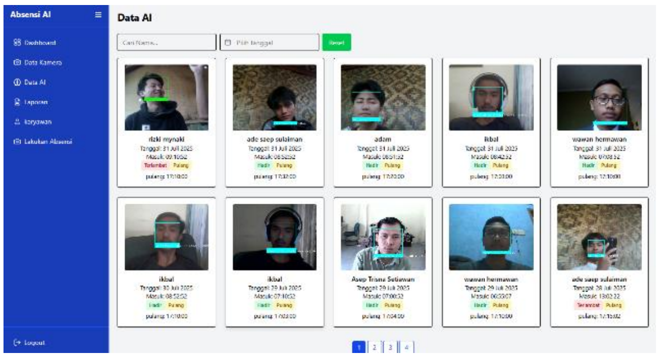


Fig. 9. AI Data Display

The accumulation of verified records then forms the basis for reporting in Fig. 10. Attendance records can be filtered by period and presented in a format ready for administrative action. Its practical value lies in the direct connection between the reporting interface and the history_AI data source, eliminating the need for separate manual reconciliation. This also shows that the microservices design not only improves the technical processing path but also facilitates governance functions at the organizational level.

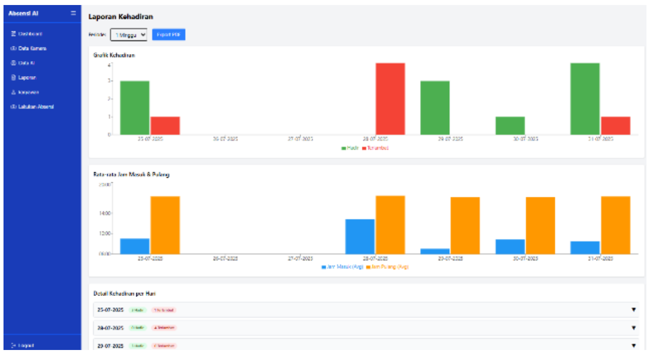
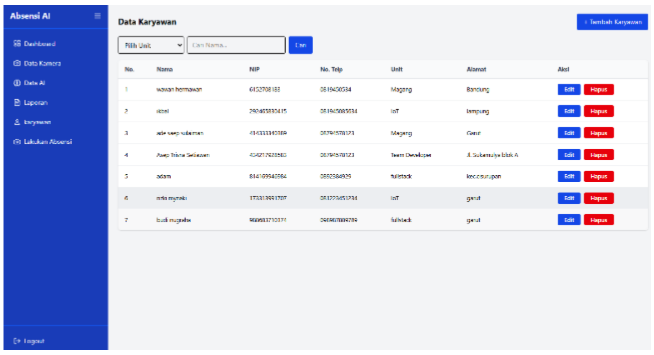


Fig. 10. Report View

Finally, Fig. 11 shows the employee data management that supports identity consistency in face recognition. The availability of profile searching, filtering, and updating ensures that the references used by Worker AI (LSKK) are always up-to-date. In practice, this layer acts as a quality controller, preventing accuracy degradation due to changes in user attributes, and serves as the cornerstone for operational sustainability—both for onboarding new employees and deactivating inactive accounts.



Data Karyawan

No.	Nama	NIP	No. Telp	Unit	Alamat	Aksi
1	wawan hermawan	6152708188	0819450534	Magang	Bandung	Edit Hapus
2	ikbal	292465830415	081945085834	IoT	lampung	Edit Hapus
3	ade saep sulaiman	414333140169	08794578123	Magang	Garut	Edit Hapus
4	Asep Trisna Setiawan	434217628560	08794578123	Team Developer	Jl. Sukamulya blok A	Edit Hapus
5	adam	814169946984	0892384929	fullstack	kec.cisurupan	Edit Hapus
6	rizki mynaki	173313991707	081223451234	IoT	garut	Edit Hapus
7	budi nugraha	969683710374	089807889789	fullstack	garut	Edit Hapus

Fig. 11. Employee View

Black-box testing is used to assess the functionality of a system without looking at the source code or internal structure [21]. This approach is appropriate because face recognition capabilities are provided through the Worker AI API (LSKK), and several services are run as independent microservices; what is evaluated is the observable behavior at the interface (HTTP endpoints, message queues, and database output), not the implementation method. Black-box testing also aligns with the goals of acceptance and specification compliance: test oracles are derived directly from use cases and written specifications, while test cases are designed using equivalence partitioning, boundary value analysis, and both normal and exception flow scenarios (including cooldown and unknown cases). Compliance with specifications is assessed by providing input and evaluating output—including status codes, JSON payloads, and stored notes—while monitoring non-functional aspects such as end-to-end latency. Table 1 presents the system testing results.

TABLE I. BLACK BOX TESTING

No.	Test Scenario	Expected Result	Observed Outcome
1	Login with an invalid username and password	An error message “Incorrect username or password” is displayed; access is denied.	Error message shown: access denied.
2	Login with a valid username and password	The user is authenticated and redirected to the dashboard.	Login succeeds; the dashboard is rendered.
3	Open the Camera Data page	Camera entries are listed with images and metadata.	Camera data loads with timestamps and device IDs.
4	Apply a date filter, then click “Reset”	Records are filtered by the selected date; “Reset” restores the default range.	Filtering works as specified; reset restores defaults.
5	Use pagination to navigate to page 2 (Camera Data)	Page-2 camera records are displayed in the correct order without omissions.	Page-2 entries appear in order; no gaps or duplicates.
6	Open the AI Data page	Recognized and unknown faces are listed with attendance information.	AI Data loads; recognized/unknown labels are shown with attendance fields.
7	Apply a name filter (AI Data)	Results are restricted to records matching the entered name.	Records correctly filtered by name.
8	Apply a date filter (AI Data)	Results are restricted to the selected date.	Records correctly filtered by date.
9	Use pagination to page 2 (AI Data)	Page-2 AI records are displayed in sequence.	Page-2 AI entries appear in order.
10	Open the Reports page	The attendance summary table and statistical charts are displayed.	The summary table and charts render as expected.
11	Click the Download PDF/Excel button	A PDF/Excel file is generated and downloaded; its contents match the on-screen report.	File downloaded; contents match the web view.
12	Open the Employees page	Employee records are displayed.	The Employee list loads successfully.
13	Open an employee's details page	Detailed information for the selected employee is displayed.	Detail view renders with complete fields.
14	Add a new employee record	The record was created and visible in subsequent queries.	Record created and appears in the list.
15	Edit an existing employee record	The record is updated, and the changes are visible.	Record updated; changes reflected in the list.
16	Delete an employee record	The record is removed and no longer returned in queries.	Record deleted; no longer present in results.

All 16 test scenarios produced outputs consistent with the oracle in the Expected Result column. For authentication scenarios, the system rejected invalid credentials and redirected valid users to the dashboard. In the camera data and AI module, filtering (name/date) and pagination display the correct and sequential results; specifically for AI Data, the recognized/unknown tagging corresponds to the output of the recognize_frame endpoint. In the report module, the period summary can be exported to PDF/Excel with content identical to the web view. CRUD operations on employee data are working normally (add, edit, delete), with changes reflected immediately in search results. No deviations were found that required failure handling.

Implementation findings show that separating functions into small services allows for lightweight work units on the critical path (camera → Worker AI → logging), as each service only loads relevant dependencies and can be scaled independently according to the load pattern. As a result, the capture-verification process remains responsive even when the reporting/interface load increases, and non-critical work can be offloaded to asynchronous processing via queues. Architecturally, the characteristics of modularity, loose coupling, and independent deployment/scaling in microservices underpin this efficiency and have been consistently reported in multi-case studies and recent reviews of microservices [3], [11], [20], [22].

IV. CONCLUSION

This research indicates that the face recognition-based attendance system built on a microservices architecture and integrated with the Worker AI API (LSKK) functions as specified. Service separation—from camera I/O, verification, and storage to presentation—supports self-scaling and maintains responsiveness; effective cooldown policies suppress duplicate entries, while separating history (raw data) and history_ai (verified records) improves traceability and reporting quality. The limitations of this study lie in the consumption of the classification service as an external API, meaning the internal configuration of the model and its decision thresholds are beyond the authors' control; testing was also conducted within a single organizational environment with limited device and condition variations, and did not yet include large-scale robustness testing. Going forward, the focus of further work is to comprehensively measure system performance, including end-to-end latency, especially during load spikes or queue formation; additionally, the misreading error rate needs to be re-monitored to provide input for improving the face recognition model.

Acknowledgment

Thank you to the Institut Teknologi Garut for their support in funding this research. Awards were also presented to PT LSKK for providing Worker AI services for facial recognition inference, technical support during API integration, and facilitating system testing in an operational environment.

References

[1] R. Hasan and A. B. Sallow, "Face Detection and Recognition Using OpenCV," *Journal of Soft Computing and Data Mining*, vol. 2, no. 2, pp. 86–97, Oct. 2021, doi: 10.30880/jscdm.2021.02.02.008.

[2] J. Patel, S. Gandhi, V. Katheriya, P. Pataliya, and A. Majumdar, "Enhancing Classroom Attendance Systems with Face Recognition through CCTV using Deep Learning," *Procedia Comput Sci*, vol. 258, pp. 3031–3041, 2025, doi: 10.1016/j.procs.2025.04.561.

[3] Y. Abgaz *et al.*, "Decomposition of Monolith Applications Into Microservices Architectures: A Systematic Review," *IEEE Transactions on Software Engineering*, vol. 49, no. 8, pp. 4213–4242, Aug. 2023, doi: 10.1109/TSE.2023.3287297.

[4] I. Oumoussa and R. Saidi, "Evolution of Microservices Identification in Monolith Decomposition: A Systematic Review," *IEEE Access*, vol. 12, no. February, pp. 23389–23405, 2024, doi: 10.1109/ACCESS.2024.3365079.

[5] V. Abhilash, S. H. Venkat, S. Nishal, S. M. Rajagopal, and N. Panda, "E-commerce Evolution: Unleashing the Potential of Serverless Microservices," in *2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, IEEE, Jun. 2024, pp. 1–8. doi: 10.1109/ICCCNT61001.2024.10726037.

[6] S. Bussa, S. Bharuka, A. Mani, and S. Kaushik, "Smart Attendance System using OPENCV based on Facial Recognition," *2nd International Conference on Sustainable Computing and Smart Systems, ICSCSS 2024 - Proceedings*, vol. 9, no. 03, pp. 1529–1535, 2024, doi: 10.1109/ICSCSS60660.2024.10624932.

[7] A. S. Lateef and M. Y. Kamil, "Facial Recognition Technology-Based Attendance Management System Application in Smart Classroom," *Iraqi Journal for Computer Science and Mathematics*, pp. 136–158, Aug. 2023, doi: 10.52866/ijcsm.2023.02.03.012.

[8] F. Tapia, M. Á. Mora, W. Fuertes, H. Aules, E. Flores, and T. Toulkeridis, "From Monolithic Systems to Microservices: A Comparative Study of Performance," *Applied Sciences*, vol. 10, no. 17, p. 5797, Aug. 2020, doi: 10.3390/app10175797.

[9] A. Bakhtin, X. Li, J. Soldani, A. Brogi, T. Cerny, and D. Taibi, "Tools Reconstructing Microservice Architecture: A Systematic Mapping Study," in *Software Architecture. ECSA 2023 Tracks, Workshops, and Doctoral Symposium*, B. Tekinerdoğan, R. Spalazzese, H. Sözer, S. Bonfanti, and D. Weyns, Eds., Cham: Springer Nature Switzerland, 2024, pp. 3–18.

[10] Mahender Singh, "Resilient Microservices Architecture with Embedded AI Observability for Financial Systems," *Journal of Electrical Systems*, vol. 20, no. 11s, pp. 4499–4510, Nov. 2024, doi: 10.52783/jes.8596.

[11] M. Niswar, R. Arisandy Safruddin, A. Bustamin, and I. Aswad, "Performance evaluation of microservices communication with REST, GraphQL, and gRPC," *International Journal of Electronics and Telecommunications*, pp. 429–436, Jun. 2024, doi: 10.24425/ijet.2024.149562.

[12] M. Waseem, P. Liang, M. Shahin, A. Di Salle, and G. Márquez, "Design, monitoring, and testing of microservices systems: The practitioners' perspective," *Journal of Systems and Software*, vol. 182, p. 111061, Dec. 2021, doi: 10.1016/j.jss.2021.111061.

[13] S. Newman, *Building microservices: designing fine-grained systems*. " O'Reilly Media, Inc.," 2021.

[14] R. S. Pressman, *Software engineePressman, R. S. (n.d.). Software engineering (2nd ed.). New York: McGraw-Hill Book Company.ring*, 7th ed. New York: Higher Education, 2010.

[15] F. Qudus Khan, S. Rasheed, M. Alsheshtawi, T. Mohamed Ahmed, and S. Jan, "A Comparative Analysis of RAD and Agile Technique for Management of Computing Graduation Projects," *Computers, Materials & Continua*, vol. 64, no. 2, pp. 777–796, 2020, doi: 10.32604/cmc.2020.010959.

[16] N. Singh and A. Hussain, "Rapid Application Development in Cloud Computing with IoT," in *IoT and AI Technologies for Sustainable Living*, Boca Raton: CRC Press, 2022, pp. 1–28. doi: 10.1201/9781003051022-1.

[17] Z. Aghababaeyan, M. Abdellatif, L. Briand, R. S, and M. Bagherzadeh, "Black-Box Testing of Deep Neural Networks through Test Case Diversity," *IEEE Transactions on Software Engineering*, vol. 49, no. 5, pp. 3182–3204, May 2023, doi: 10.1109/TSE.2023.3243522.

[18] I. El Gaabouri, M. Senhadji, M. Belkasmi, and B. El Bhiri, "A Systematic Literature Review on Authentication and Threat Challenges on RFID Based NFC Applications," *Future Internet*, vol. 15, no. 11, p. 354, Oct. 2023, doi: 10.3390/fi15110354.

[19] Y. Malabi, M. Hani'ah, Noprianto, V. N. Wijayaningrum, V. Al Hadid Firdaus, and A. Himawan, "Efficient Employee Attendance System Integrating RFID and Android-Based Face Recognition with Liveness Detection," in *2024 International Conference on Electrical and Information Technology (IEIT)*, IEEE, Sep. 2024, pp. 163–168. doi: 10.1109/IEIT64341.2024.10763296.

[20] M. Söylemez, B. Tekinerdogan, and A. K. Tarhan, "Microservice reference architecture design: A multi-case study," *Softw Pract Exp*, vol. 54, no. 1, pp. 58–84, Jan. 2024, doi: 10.1002/spe.3241.

[21] R. G. Kawi and Suprihadi, "Design of Website-Based Tourism Travel Information System (Case Study : Tenta Tour)," *International Journal Software Engineering and Computer Science (IJSECS)*, vol. 3, no. 3, pp. 317–323, Dec. 2023, doi: 10.35870/ijsecs.v3i3.1788.

[22] A. El Akhdar *et al.*, "Exploring the Potential of Microservices in Internet of Things: A Systematic Review of Security and Prospects," *Sensors*, vol. 24, no. 20, p. 6771, Oct. 2024, doi: 10.3390/s24206771.