QR Code and Face Recognition-Based Vehicle Key Validation System in SIPAKAD Application

Adi Setiawan[1)], Vincensius Arga Yoda[2)], dan Yudhi Darmawan[3)]
[1), 2), 3)]Prodi Teknik Telekomunikasi Militer. Politeknik Angkatan Darat Jl.Raya Anggrek No.1 Junrejo, Batu, Indonesia
E-mail: 22Setiawan.adi@gmail.com[1)], vincentsius@gmail.com[2)], dan yudhidharmawan@polektad.ac.id[3)]

Abstract: *Vehicle management in the military, particularly the Army, requires a secure, fast, and efficient system to support daily operations. Manual processes for driver identity verification and vehicle access have proven to be ineffective and prone to errors. This study aims to design and implement a vehicle key pickup and return validation system based on QR Code technology and face recognition through a Progressive Web App (PWA) called SIPAKAD. The system is developed to enhance security, efficiency, and transparency in the management of official vehicles at Poltekad. QR Code technology is used for vehicle identification, while face recognition serves as biometric authentication for drivers before accessing vehicle keys. The system is built using PHP, MySQL, Flask (Python), integrated with Firebase Cloud Messaging for real-time notifications, and utilizes the MQTT protocol for communication between IoT devices. Testing conducted with the Raspberry Pi 3B showed that the highest validation accuracy was achieved under straight face conditions with bright lighting at 64.12%, with an average total accuracy of 43.99% across four conditions, and an average response time of 5.2 seconds. The implementation results demonstrate that this system reduces identification time, speeds up validation processes, minimizes manual errors, and supports more modern and documented vehicle data management.*
Keywords: Face recognition; IoT (Internet of Things); MQTT (Message Queuing Telemetry Transport); QR Code; Vehicle management system

Abstrak: *Sistem ini dikembangkan untuk meningkatkan keamanan, efisiensi, dan transparansi dalam pengelolaan kendaraan dinas di Politeknik Angkatan Darat (Poltekad). Teknologi QR Code digunakan untuk identifikasi kendaraan, sedangkan pengenalan wajah berfungsi sebagai autentikasi biometrik bagi pengemudi sebelum mengakses kunci kendaraan. Sistem dibangun menggunakan PHP, MySQL, dan Flask (Python), terintegrasi dengan Firebase Cloud Messaging untuk notifikasi real-time, serta memanfaatkan protokol MQTT sebagai media komunikasi antar perangkat IoT. Pengujian yang dilakukan menggunakan Raspberry Pi 3B menunjukkan bahwa akurasi validasi tertinggi diperoleh pada kondisi wajah tegak dengan pencahayaan terang sebesar 64,12%, dengan rata-rata akurasi keseluruhan sebesar 43,99% pada empat kondisi pengujian, serta rata-rata waktu respons sebesar 5,2 detik. Hasil implementasi menunjukkan bahwa sistem ini mampu mengurangi waktu identifikasi, mempercepat proses validasi, meminimalkan kesalahan manual, serta mendukung pengelolaan data kendaraan yang lebih modern dan terdokumentasi dengan baik.*
*Keywords: Face recognition; IoT (Internet of Things); MQTT (Message Queuing Telemetry Transport); QR Code; Sistem manajemen kendaraan***.**

INTRODUCTION

The management of official vehicles within military institutions, particularly the Army, is an important aspect in supporting daily operations. The efficient and transparent management of official vehicles is a challenge faced by many government agencies (Firdaus, 2024). Currently, vehicle access management and driver verification still rely on manual processes that are time-consuming, prone to human error, and ineffective in preventing unauthorized access. One alternative is the implementation of QR Code technology for controlling official vehicle security (Muhaimin & Nurhidayati, 2024).

With the development of technology, the application of systems based on QR Code, face recognition, and integration with Firebase Cloud Messaging offers a modern and more reliable solution. QR Code technology enables rapid vehicle identification by scanning unique codes integrated with vehicle and driver data. Meanwhile, face recognition technology is used to ensure that the driver accessing the vehicle matches the authenticated data registered when retrieving keys from the key box. Face recognition technology is considered more secure than conventional security methods and more affordable in terms of cost (Diantoro et al., 2023). QR Codes have the advantage of a very low rate of information retrieval failure because they can be read from any direction (Purnomo & Alijoyo, 2024). Face recognition works by matching the unique facial characteristics of an individual using Dlib as a library for facial detection based on facial landmark methods (Mardhiyyah et al., 2022). The use of Progressive Web Apps (PWAs) on Android devices provides flexibility in accessing the vehicle validation system with features such as QR Code scanning, face recognition, offline mode, and a native app-like experience (Novia et al., 2023).

Python is a high-level programming language that is easy to learn and use to run programs with various complex functions with minimal code, but it has drawbacks such as slow performance and limited support on mobile platforms (Dermawan et al., 2023). Flask is a lightweight Python framework used to efficiently build the backend of the vehicle validation system (Wijayanto & Susetyo, 2022). MySQL is an efficient relational database platform for storing, managing, and accessing data based on the RDBMS concept (Noviana, 2022). OpenCV is an open-source library for image processing and computer vision supporting various functions from image reading to object detection (Mahmod et al., 2023). Raspberry Pi is a small single-board computer module with display ports and USB connections, distinguishing it from regular microcontroller boards (Aryapranata, 2020).

This study aims to design a QR Code and face recognition-based official vehicle access system for validating key pickup and return, analyzing the accuracy and speed of the system using Raspberry Pi 3B and MQTT, and developing modern and structured vehicle and driver data integration with the Flask Framework. The system is supported by MQTT (Message Queuing Telemetry Transport), a machine-to-machine (M2M) connectivity protocol with the advantage of lightweight bandwidth data transmission and low latency (Sadewo et al., 2024). This system is expected to serve as a modern and adaptive vehicle management solution tailored to military environment needs.

RESEARCH METHOD

This research was conducted at the Poltekad Kodiklatad Telecommunications Workshop from November 2024 to July 2025. A quantitative approach and software engineering method were used to design and test the driver identity validation system. The research involved system design, software implementation, and system performance testing through accuracy and response time measurements. The research began with data collection and analysis, followed by design, construction, simulation, and testing. If the equipment did not meet standards, improvements were made until it met the requirements, after which it was implemented and conclusions were drawn.

Main Block Diagram

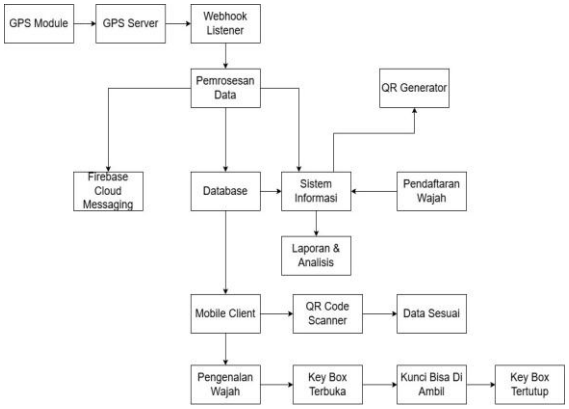

Figure 1. Main Block Diagram

The process begins with the GPS Module sending location data to the GPS Server, which is then forwarded to the Webhook Listener as the entry point for initial processing. The data is then processed in the Data Processing module and stored in the Database as the central storage location. This system is connected to the Information System, which interacts with modules such as Face Registration for biometric authentication and QR Generator for QR code generation. For real-time information delivery, Firebase Cloud Messaging is used. All collected data is analyzed through the Reporting and Analysis module.

Program Block Diagram

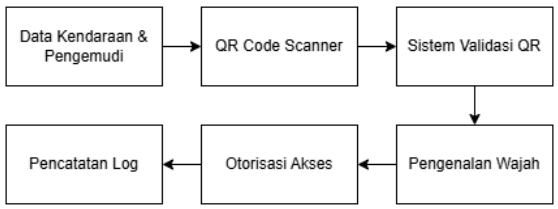

Figure 2. Program Block Diagram

This system uses vehicle and driver data stored in a centralized database for access validation. Vehicle identification is performed using an Android-based QR Code Scanner that reads unique codes and sends them to the server for verification. The Mobile Client application serves as the main user interface. The process begins with QR code scanning for authentication, followed by facial verification to unlock the Key Box where the keys are stored.

Face Recognition Block Diagram

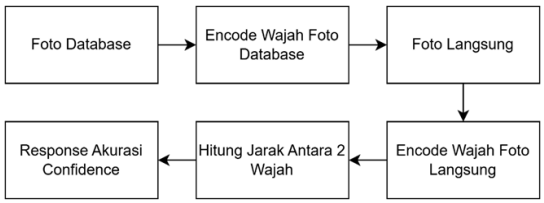

Figure 3. Face Recognition Block Diagram

The facial verification system works by comparing a reference photo stored in the database with the latest photo taken when the driver wants to retrieve the vehicle key. Both photos are processed using face recognition technology that converts the image into a numerical vector using facial encoding techniques. The system then calculates the similarity distance using the Euclidean formula between the two vectors. If the match is confirmed, the system automatically unlocks the key box for key retrieval.

Mobile Application Flowchart

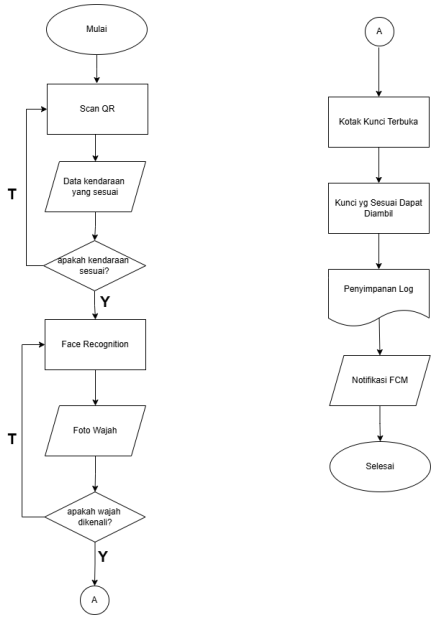

Figure 4. Mobile Application Flowchart

The process begins when an officer scans the vehicle's QR code using an Android app to match the data with the database. If the vehicle data matches, the driver proceeds. Next, facial verification is performed by taking a photo of the driver and matching it with stored data. If the face is recognized, the key

box automatically opens. All validated data, including QR code, facial photo, and verification status, is stored in the database for tracking. The system also sends automatic notifications via Firebase to the officer or administrator.

RESULT AND DISCUSSION

SIPAKAD is a web-based and mobile information system created to support the management of official vehicles at POLTEKAD. The system aims to improve transparency and efficiency, particularly in driver verification when picking up keys and vehicle authorization at guard posts. SIPAKAD Mobile has two main interfaces tailored to user roles: drivers and guard post officers. Operational data is recorded digitally and can be exported in Excel format.

User Interface

The initial screen displays a login page with the Indonesian Army logo as the institutional identity. There are two input fields for NRP/Email and Password, and a blue "Sign In" button. After logging in, users are taken to the home page displaying the vehicle key status and shortcuts to travel features. The Key Box page is designed to process driver identity through three stages: QR code scanning, face capture, and face comparison, with MQTT support to send verification results.

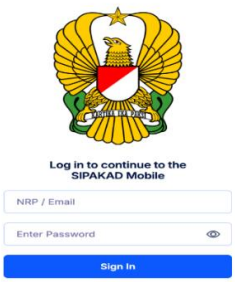

Figure 5. Login Page

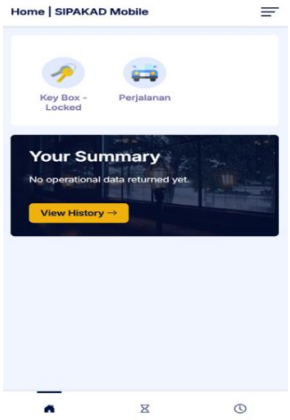

Figure 6. Home Page

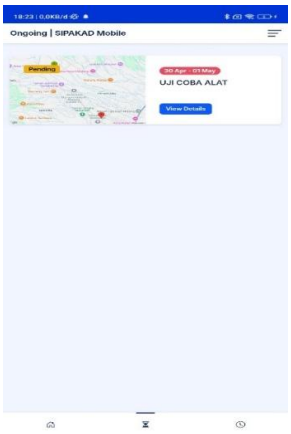

Figure 7. Ongoing Page

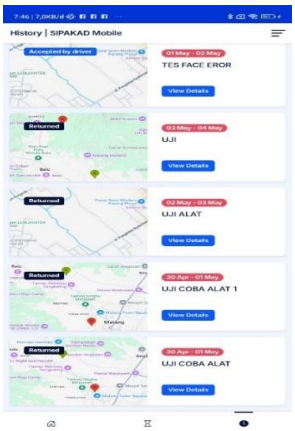

Figure 8. History Page

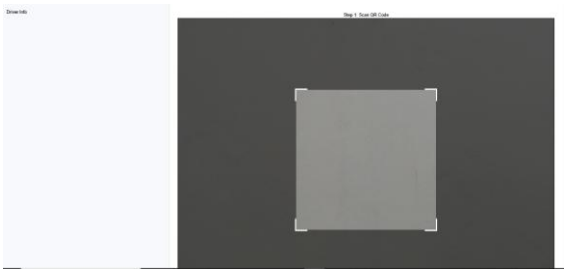

Figure 9. Key Box QR Code Screen Page

Vehicle Management System Testing

This test shows that the SIPAKAD Website and SIPAKAD Mobile have been well integrated. When the admin starts a new trip, users with the driver role who have logged into the SIPAKAD Mobile application will receive automatic notifications. The driver must read the task details containing departure time, return time, requirements, and task instructions. After understanding and agreeing to the task, the driver obtains permission to open the key box through QR Code scanning and face verification.

Figure 10. Task Notification

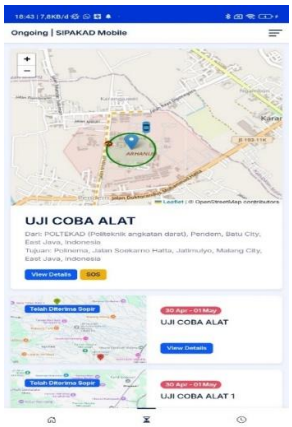

Figure 11. On Going Page with Task Pending

Figure 12. Details Page Before Driver Approval

Figure 13. Detail Page After Accepted by Driver

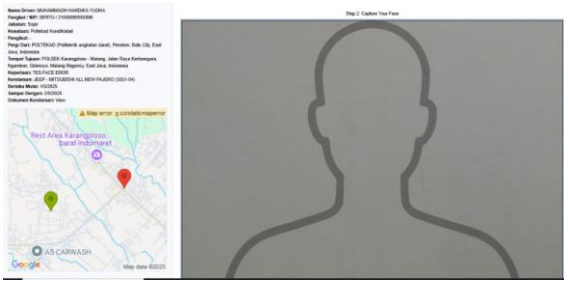

Figure 14. Key Box Face Recognition Screen Page

After successful facial verification, the key box door opens automatically and the driver can retrieve the vehicle keys. The driver then closes the key box door using the button on the SIPAKAD Mobile app, then proceeds to meet the guard post officer. The officer scans the QR Code to view the driver's task details. After validation, the officer presses the "Depart Now" button, providing the driver with location information and a route map generated using the Dijkstra algorithm. Drivers can send SOS messages including images and coordinates, which are received by administrators.

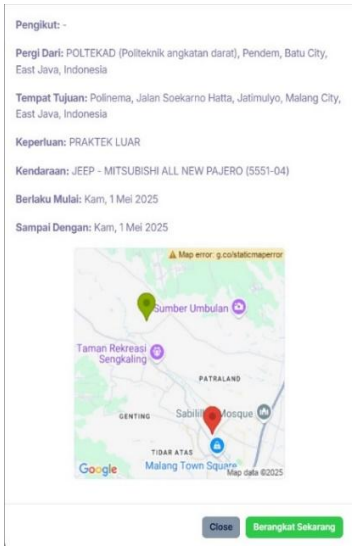

Figure 15. Details Ongoing Driver Has Departed

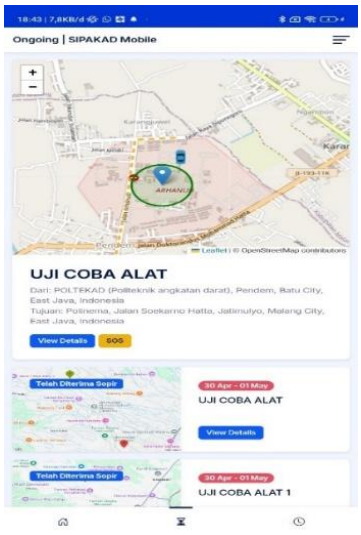

Figure 16. SOS Notification Map Details

Figure 17. Driver Tasks Upon Arrival at Destination

Figure 18. Driver Tasks Upon Return

System Validation and Speed Testing

Validation and speed testing of the QR Code scanning and face recognition systems on the SIPAKAD Mobile application was conducted to ensure the accuracy and performance of the driver identity verification feature. The system uses the face_recognition library to generate vector representations of each face, which are then compared using the face_distance() function. The system was tested using a Raspberry Pi 3B as the hardware controller and the MQTT protocol for data communication.

The face recognition accuracy is calculated using the following formula:

$$Akurasi_{\text{avg}} = \frac{\sum_{i=1}^{n} Akurasi_i}{n} \quad \text{..... (1)}$$

Where *d* is the Euclidean distance between the camera face encoding and the database encoding.

The average accuracy and average response time are calculated using the following formulas (Pancono et al., 2024):

$$Waktu_{\text{avg}} = \frac{\sum_{i=1}^{n} Waktu_i}{n} \quad \text{..... (2)}$$

Where $Accuracy_i$ = Accuracy in each experiment, n = Total number of tests, $Accuracy_{avg}$ = Average accuracy of all tests.

Table 1. Straight Face and Bright Location Scenario

No	Sample	Matching Accuracy QR Code + Face	Response Time	Description

1	Personnel 1	64.08%	6930 ms	Detected
2		57.09%	5214 ms	Detected
3		63.70%	4172 ms	Detected
4		62.32%	4948 ms	Detected
5		66.01%	4829 ms	Detected
6	Personnel 2	68.02%	5037 ms	Detected
7		63.03%	6640 ms	Detected
8		59.98%	5030 ms	Detected
9		66.35%	5434 ms	Detected
10		70.65%	4587 ms	Detected
	Average	**64.12%**	**5282.1 ms**	

**) Source: Authors, 2025*

Table 2. Tilted Face and Bright Location Scenario

No	Sample	Matching Accuracy QR Code + Face	Response Time	Description
1	Personnel 1	61.73%	5596 ms	Detected
2		57.98%	4784 ms	Detected
3		42.97%	4360 ms	Detected
4		57.81%	4656 ms	Detected
5		24.69%	5937 ms	Detected
6	Personnel 2	29.38%	5919 ms	Detected
7		54.33%	4198 ms	Detected
8		45.78%	5012 ms	Detected
9		42.73%	6273 ms	Detected
10		24.72%	5971 ms	Detected
	Average	**44.212%**	**5270.6 ms**	

**) Source: Authors, 2025*

Table 3. Straight Face and Dark Location Scenario

No	Sample	Matching	Response Time	Description
		Accuracy QR Code + Face		
1	Personnel 1	60.48%	6352 ms	Detected
2		60.26%	4723 ms	Detected
3		46.66%	3947 ms	Detected
4		43.16%	4393 ms	Detected
5		16.28%	5093 ms	Detected
6	Personnel 2	58.08%	4222 ms	Detected
7		24.00%	5205 ms	Detected
8		56.18%	4024 ms	Detected
9		57.46%	4315 ms	Detected
10		19.29%	5800 ms	Detected
	Average	**44.185%**	**4807.4 ms**	

**) Source: Authors, 2025*

Table 4. Tilted Face and Dark Location Scenario

No	Sample	Matching Accuracy QR Code + Face	Response Time	Description
1	Personnel 1	25.00%	5655 ms	Detected
2		22.37%	5588 ms	Detected
3		25.45%	5884 ms	Detected
4		18.08%	6436 ms	Detected
5		25.82%	5161 ms	Detected
6	Personnel 2	22.65%	5325 ms	Detected
7		19.87%	6227 ms	Detected
8		27.43%	5875 ms	Detected
9		20.41%	5496 ms	Detected
10		27.30%	5906 ms	Detected
	Average	**23.438%**	**5755.3 ms**	

**) Source: Authors, 2025*

Testing of the vehicle and driver data verification system using Raspberry Pi 3B and MQTT under four lighting conditions and face positions showed varying results. With a straight face under bright lighting, accuracy reached 64.12% with a response time of 5282.1 ms. With a tilted face under bright lighting, accuracy dropped to 44.21% with a response time of 5270.6 ms. A straight face under dim lighting yielded an accuracy of 44.19% and a response time of 4807.4 ms, and the worst condition, a tilted face under dim lighting, showed an accuracy of only 23.44% with a response time of 5755.3 ms. The average total accuracy for all four conditions is 43.99%, with an average response time of 5278.85 ms.

Integrating Systems with Vehicle and Driver Data Using the Flask Framework

The system was successfully built on the Flask framework, which manages communication between mobile applications, servers, and IoT devices in an integrated manner. All vehicle data, driver data, trip status, and face verification results are stored in a MySQL database accessed by Flask through an API. The Flask backend also handles user authentication, session management, and notification delivery through Firebase Cloud Messaging.

Implementing Server Code Using the Flask Framework

The app.py file is the core logic of the SIPAKAD Mobile application, built using the Flask framework. As the backend server, this file manages communication between users, the database, authentication, trip validation, trip status management, and integration with third-party services such as Firebase Cloud Messaging and Google Maps API. Users can log in via the /login endpoint using their NRP or email and password, which are encrypted with bcrypt.

The facial verification process is performed at the /compare endpoint, where the driver's real-time facial photo is compared with the stored photo on the server using a face recognition library, resulting in an accuracy score that classifies the match as "Very high", "Possibly a match", or "Not a match". Travel status management is done through the /ongoing/<id>/accept endpoint, which changes status from pending, on the way, arrived, to returned, with time recording and notification delivery to the admin via Firebase Cloud Messaging. The SOS feature via the /sos endpoint allows drivers to send coordinates, descriptions, and photos of emergency conditions.

Code Implementation of IoT (main.py)

In the SIPAKAD Mobile system, the physical security of official vehicles is integrated with IoT devices controlled by the Raspberry Pi through the main script main.py. This script regulates physical access to vehicles using servo motors and relays controlled remotely, connected to the main server via the MQTT protocol. The system uses TLS to secure data communication and configures the Raspberry Pi's GPIO on two pins: one for the servo motor (GPIO14) and one for the relay (GPIO4) controlling the solenoid door lock. The servo is controlled using a PWM signal at 50Hz frequency through the set_servo_angle(angle) function.

Figure 19. Key Box Display for Key Retrieval Process

The IoT system focuses on two MQTT callbacks: on_connect() and on_message(). on_connect() subscribes to the sipakad/compare/result topic. When the server sends the "face_compare_success" event, the servo opens the key box door and the relay deactivates the solenoid lock. Conversely, the "close" event locks the door. After each action, the system sends a response to the mobile/reply topic with the lock status and timestamp. The cleanup() function ensures safe device shutdown.

Table 5. IoT System Function Data Collection

No	Iteration	Raspberry Pi Receives Message	Solenoid & Servo
1	Trial 1	face_compare_success	Active → Inactive
2	Trial 2	close	Inactive → Active
3	Trial 3	face_compare_success	Active → Inactive
4	Trial 4	close	Inactive → Active
5	**Trial 5**	**face_compare_success**	**Active → Inactive**

**) Source: Authors, 2025*

Based on five trials, the system responds to the face_compare_success message by opening access and responds to the close message by locking access again. This pattern shows the system runs synchronously and functions without issues.

DISCUSSION

The system was designed with a two-layer validation approach: QR Code scanning and driver face verification. The QR Code displays vehicle data as a unique identifier linked directly to the vehicle database. The mobile application reads the QR Code and sends the data to the server for verification, followed by face image capture compared with the reference photo using the face recognition algorithm. Testing on two personnel showed an average validation time of 4.7 seconds with all attempts successfully detected.

Face recognition testing across four different conditions showed an average total accuracy of 43.99%. The highest accuracy of 64.12% was achieved under straight face conditions with bright lighting, while the lowest accuracy of 23.44% occurred under tilted face conditions with dim lighting. The average system response time was 5278.85 ms.

System Evaluation and Limitations

The average total accuracy of 43.99% indicates that the system fails to correctly recognize faces more often than it succeeds. For a security system, this figure does not meet adequate standards. Several factors contributing to the low accuracy were identified as follows.

First, the hardware limitations of the Raspberry Pi 3B, which has only 1GB of RAM and a 1.2 GHz quad-core processor, are a primary factor. The face_recognition library used is based on deep learning (dlib), which requires substantial computational capacity to produce accurate face encodings. The limited specifications of the Raspberry Pi 3B cause the encoding process to be suboptimal, especially when images have high resolution or non-ideal lighting conditions. This aligns with the findings of Ghael et al. (2021), who demonstrated that Raspberry Pi performance can decrease significantly on image-intensive processing tasks.

Second, the camera quality significantly affects accuracy. A standard-resolution webcam produces images lacking detail, especially under low lighting. Test results showed a drastic accuracy decrease from 64.12% (bright, straight) to 23.44% (dark, tilted), indicating the system's strong dependence on input image quality. Research by Kumaran et al. (2021) also showed that lighting variation and viewing angle are primary challenges in face recognition systems.

Third, the limited training dataset (using only photos input by the admin) prevents the model from optimally recognizing variations in facial expressions, viewing angles, and different lighting conditions. The system only compares one reference photo with a real-time photo, limiting its adaptability to varying capture conditions.

Nevertheless, the QR Code system achieved 100% reading accuracy across all tests, demonstrating that the vehicle identification component functions very well.

The average response time of 5.2 seconds is also within acceptable limits for field operational needs.

To improve future system performance, it is recommended to upgrade to Raspberry Pi 4 or 5 with higher specifications, use a higher-resolution camera, increase the face training dataset with various conditions, and consider using 3D face recognition methods employing depth sensors or 3D cameras for spatial face shape recognition, which provides better accuracy across various angles and lighting conditions.

The Flask framework integration successfully connected all system components including the mobile application, server, and IoT devices. The Flask backend handles user authentication, trip management, face verification, and real-time notification delivery. IoT device testing showed that the Raspberry Pi 3B responded correctly to MQTT commands across all five trials, indicating good synchronization between the device and server.

CONCLUSION

The use of QR Code and face recognition technology has been successfully implemented to validate driver identity before picking up and returning vehicle keys. System testing using a Raspberry Pi 3B and MQTT under four conditions demonstrated the highest accuracy of 64.12% when the face was facing directly forward under bright lighting, while the worst conditions occurred when the face was tilted under dim lighting with an accuracy of 23.44%. The average total accuracy across the four conditions was 43.99% with an average response time of 5.2 seconds. Although the overall accuracy is still considered low for security system standards, this is attributable to the hardware limitations of the Raspberry Pi 3B, camera quality, and limited training dataset. The system successfully demonstrated the concept of integrating QR Code, face recognition, and IoT technologies in military official vehicle management. The Flask framework backend integrates in real-time with the vehicle and driver database, supporting more modern, documented, and transparent vehicle management, including travel log recording, vehicle status tracking, and emergency condition reporting via the mobile application. Future research is recommended to improve accuracy through hardware upgrades, better camera quality, additional training datasets, and exploration of 3D face recognition methods.

REFERENCES

Aryapranata, A. (2020). Pengembangan Jaringan Komputer Lokal dengan Memanfaatkan Raspberry Pi Bagi Perusahaan Startup atau UMKM. Jurnal Esensi Infokom, 2(2), 47–53.

Dacipta, P. N., & Putra, R. E. (2022). Sistem Klasifikasi Limbah Menggunakan Metode Convolutional Neural Network (CNN) Pada Web Service Berbasis Framework Flask. Journal of Informatics and Computer Science, 03.

Dermawan, M. F. H., Witarsyah, D., & Fakhruroja, H. (2023). Penerapan Image Processing Untuk Mengetahui Tingkat Kematangan Kopi Menggunakan Algoritma K-Nearest Neighbor (KNN) pada Perkebunan Kopi Malabar Bandung. E-Proceeding of Engineering, 10(3), 3246–3252.

Diantoro, K., Rohman, A., Juwari, & Anjelis, R. (2023). Rancang Bangun Sistem Keamanan Rumah dengan Face Recognition Menggunakan ESP32-CAM. IBI Kosgoro, 4(2), 150–156. https://doi.org/10.55122/junsibi.v4i2.970

Fadli, F., & Munawir. (2019). Kontrol Mouse Menggunakan Webcam Berdasarkan Deteksi Warna. JTIM: Jurnal Teknologi Informasi dan Multimedia, 1(1), 73–77. https://doi.org/10.35746/jtim.v1i1.18

Firdaus, M. I. (2024). Aplikasi Manajemen Kendaraan Berbasis Web Untuk Optimalisasi Efisiensi Operasional. Karimah Tauhid, 3(10), 11518–11527. https://doi.org/10.30997/karimahtauhid.v3i10.15263

Ghael, H. D., Solanki, L., & Sahu, G. (2021). A Review Paper on Raspberry Pi and its Applications. International Journal of Advances in Engineering and Management, 2(12), 10–13. https://doi.org/10.35629/5252-0212225227

Julius, F., Karnila, S., Safitri, E., Purwati, N., & Riza Nul Fikri, R. (2024). Implementasi Sistem Manajemen Parkir Menggunakan

Teknologi QR-Code Berbasis Web. IJCCS, x(No.x), 1–5.

Kumaran, I., Firmansyah, M. R., Fauziah, E., et al. (2021). Pengenalan Wajah Menggunakan Pendekatan Berbasis Pengukuran dan Metode Segmentasi dalam Berbagai Posisi dan Pencahayaan. Jurnal Teknik Elektro, 3(1), 5–8.

Mahmod, M. N. bin, Ramli, M. binti, & Yasin, S. N. T. M. (2023). QR Code Detection Using OpenCV Python with Tello Drone. Southeast Asian Journal of Technology and Science, 4(1), 22–27.

Manurung, R., Aspriyono, H., & Prasetio, E. (2022). Aplikasi Monitoring Gangguan Frekuensi Radio Berbasis Android Menggunakan Firebase Cloud Messaging (FCM). MEANS, 7(2).

Mardhiyyah, R., Hajar Puji Sejati, R., & Sekti Aji, A. (2022). Deteksi Wajah untuk Presensi Menggunakan Facial Landmark. Jurnal Teknologi Informasi, 5(2), 144–148.

Monita, M., & Hendri, H. (2021). Sistem Kontrol Rumah Pintar Menggunakan Kamera Berbasis IoT. JTEIN, 2(1), 107–112. https://doi.org/10.24036/jtein.v2i1.141

Muhaimin, A., & Nurhidayati. (2024). Implementation of QR Codes for Security Control of Official Vehicles at BPKAD Pekanbaru City. JOISSE.

Novia, A., Lestanti, S., & Budiman, S. N. (2023). Perancangan Aplikasi Automatic Cornell Note Berbasis Progressive Web App (PWA) Menggunakan Metode Extreme Programming. Jurnal Mahasiswa Teknik Informatika, 7(5), 3667–3675.

Noviana, R. (2022). Pembuatan Aplikasi Penjualan Berbasis Web Monja Store Menggunakan PHP dan MySQL. JTS, 1(2), 112–124.

Pamboro, L., & Setiawan, E. (2023). Sistem Pengaman Kendaraan Roda Dua dengan Pengenalan Wajah Menggunakan Principal Component Analysis. J-PTIIK, 7(5).

Pancono, S., Indroasyoko, N., & Asep Irfan Setiawan. (2024). Pemantauan dan Deteksi Penyakit Daun Tomat Berbasis IoT dan CNN dengan Aplikasi Android. Indonesian Journal of Computer Science, 13(3), 4692–4709.

Purnomo, S., & Alijoyo, F. A. (2024). Sistem Peminjaman Barang Menggunakan QR Code Berbasis Aplikasi Android. Jurnal Teknologi dan Sistem Informasi Bisnis, 6(2), 322–328.

Sadewo, L. D., Ramdhani, M., & Rahmawati, D. (2024). Analisis Komunikasi Data Menggunakan Internet of Things dengan Protokol MQTT pada Alat Swimming Lap Counter.

Savitri, C. E., & Paramytha, N. (2022). Sistem Monitoring Parkir Mobil Berbasis Mikrokontroller ESP32. Jurnal Ampere, 7(2). https://doi.org/10.31851/ampere

Wijayanto, C., & Susetyo, Y. A. (2022). Implementasi Flask Framework Pada Pembangunan Aplikasi Sistem Informasi Helpdesk (SIH). Jurnal Ilmiah Penelitian dan Pembelajaran Informatika, 7(3), 858–868.

Zaetun, Marhaeni, & Rosmawarni, N. (2020). Perancangan Sistem Informasi Parkir dengan QR-Code Berbasis Website pada Real Estate Indonesia Jakarta. Jurnal Rekayasa Informasi, 9(2).