M.A.T.A: A Facial Recognition Smart Door Lock with Emergency Alerts for Guardless Residences

Nor Haslinda Binti Ismail[1*], Muhammad Shaffuan Amir Abd Muthalib[2]

[1]School of Technical Foundation and Diploma Studies (STEP);

[2]Faculty of Information and Communication Technology (FTMK) Universiti Teknikal Malaysia Melaka (UTeM), Durian Tunggal, Melaka, 76100, Malaysia

[*]Corresponding Author

DOI: https://doi.org/10.47772/IJRISS.2026.101000772

Received: 27 October 2026; Accepted: 02 November 2026; Published: 17 November 2026

ABSTRACT

The M.A.T.A: Monitoring & Authentication Tracking Application is developed to improve residential security and access management efficiency for modern households. Traditional physical locks and standard security systems often lack real-time monitoring capabilities, making them vulnerable to unauthorized breaches and failing to keep homeowners actively informed of who is accessing their premises. These challenges lead to compromised safety, untracked user activities, and delayed responses during critical emergencies. To address these issues, the proposed system modernizes home security by integrating AI-driven facial recognition technology to securely authenticate users before granting system access. The mobile application utilizes cloud-based infrastructure to automate real-time tracking of user activities, instantly logging events such as biometric updates, profile modifications, and system unlocks. Additionally, the system provides a centralized interactive SOS emergency protocol that proactively alerts all registered household members while granting rapid-dial access to emergency hotlines and the primary homeowner. Visual notification alerts, role-based access controls, and a dedicated secure mode feature are also included, empowering homeowners to instantly lock or unlock overall system access to mitigate potential threats. Overall, this project aims to deliver a robust, biometric-secured monitoring solution that enhances residential safety, reduces unauthorized access risks, and promotes rapid emergency management. By transforming traditional security measures into an intelligent, centralized mobile application, M.A.T.A supports smarter household monitoring and aligns with modern advancements in smart home technology and biometric automation.

Keywords— Smart Door Lock, Internet of Things, Facial Recognition, Firebase, Emergency S.O.S

INTRODUCTION

Traditional physical door locks have long been the standard for residential security; however, they present significant vulnerabilities, primarily because physical keys are easily misplaced, duplicated or stolen. These conventional systems offer no remote access control or real-time monitoring, creating a critical security gap for modern households. The limitation is particularly concerning in guardless residential areas, such as Taman Bukit Tambun Perdana in Melaka, where houses are frequently left unoccupied during working hours or inhabited solely by elderly family members. Under these circumstances, homeowners lack a centralized platform to monitor remotely who is accessing their property or to respond rapidly to indoor emergencies.

To address these vulnerabilities, the Monitoring & Authentication Tracking Application (M.A.T.A) was developed as a comprehensive, IoT-based smart door lock ecosystem. M.A.T.A replaces key-based entry with biometric authentication and centralized mobile software control. The mobile application, developed using the flutter framework, acts as the primary control hub for the homeowner. At the physical entryway, an ESP32-CAM module performs real-time facial scanning and grants automated access to registered household members

through a secure face unlock mechanism. The application also provides a remote button unlock feature, while firebase is used to log entry activities and to alert the user's smartphone when an unrecognized visitor is detected.

Beyond access control, M.A.T.A functions as a proactive safety management tool. It features a centralized In-App Emergency S.O.S protocol that enables vulnerable occupants, such as the elderly, to trigger a virtual panic button instantly in the event of a medical emergency or a potential break-in. This feature dispatches push notifications to remote family members and provides rapid-dial access to emergency services. A dedicated secure mode further allows homeowners to lock down or control overall system access to mitigate potential threats. Together, these features deliver a reliable, secure, and proactive monitoring solution tailored to the safety needs of modern, guardless communities.

A. Problem Statement

There are three problems motivate this work; (i) a high risk of unauthorized entry in guardless residential areas owing to the reliance on physical key;s (ii) the lack of a mobile platform for remote access control and real-time visitor monitoring and; (iii) the absence of a quick emergency response mechanism when a household member collapses or feels threatened inside the house.

B. Objectives

The study pursues four objectives. First, to design and develop a flutter-based smart home mobile application that serves as the main control center, allowing the homeowner to manage registered family members, view activity logs and unlock the door remotely. Second, to integrate an ESP32-CAM microcontroller with a solenoid door lock so that the camera scans faces at the front door and triggers the lock automatically when a registered family member is recognized. Third, to implement a centralized In-App Emergency S.O.S feature that keeps occupants safe, especially the elderly, by sending an instant push notification to the homeowner and enabling a quick call to the police or an ambulance. Fourth, to implement advanced security controls, including a secure mode that lets the homeowner lock down the system from a phone and block physical access at the door, which is useful when the house is left empty for a long time or a threat is detected outside.

C. Scope and Significance

The target users are homeowners, who monitor the front door and control access while away; elderly occupants, who are often alone during the day and need the quick and accessible S.O.S tool; and registered family members, who use facial recognition to unlock the door. Technically, the system comprises three parts (Fig.1) a flutter mobile application (remote button unlock, live visitor alerts, an S.O.S dashboard, secure mode, and access history logs); a hardware layer in which an ESP32-CAM performs real-time facial scanning and triggers a solenoid door lock for recognized faces; and a cloud layer in which firebase realtime database and cloud storage synchronize data, store user profiles securely, and handle push notifications between the lock and the application.

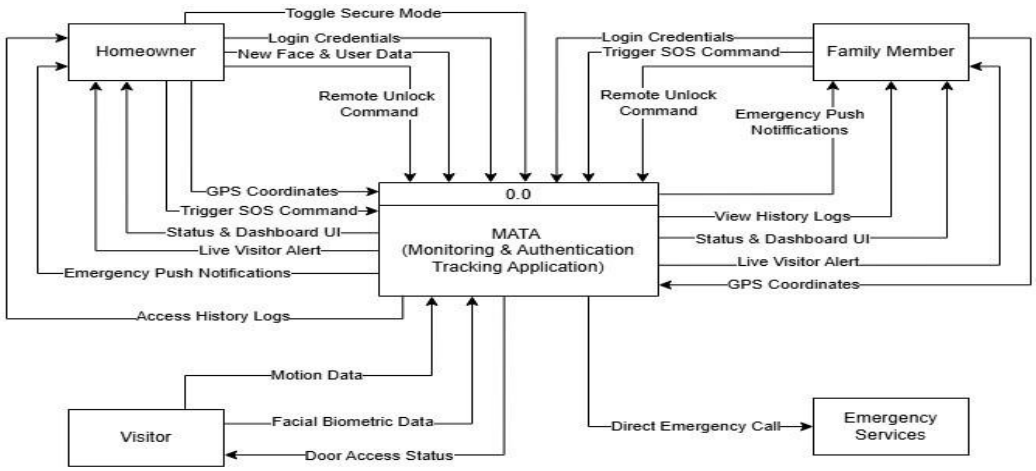


Fig. 1. M.A.T.A Context Diagram

The project benefits homeowners, who gain peace of mind because they can monitor the front door, track who enters the house, and lock down the system through secure mode, elderly and vulnerable occupants, who gain a safety net in the form of a simple S.O.S button that alerts family or the authorities in case of illness, a fall or a break-in and the local community, which gains a proactive approach to neighborhood security in which facial recognition actively prevents unauthorized entry. The expected output is a fully functional smart door lock prototype comprising the flutter application, a working ESP32-CAM and solenoid lock hardware prototype and a configured real-time firebase database that synchronizes activity logs, image captures and emergency push notifications without delay.

Related Work

The primary domain of this work is the Internet of Things (IoT) applied to smart home security, which upgrades traditional mechanical hardware into internet-connected devices that users can control and monitor remotely (Gubbi et al., 2013). Within this domain, the project focuses on the sub-domain of automated access control and mobile-based emergency safety tools for houses in guardless residential areas.

The chosen approach, which combines real-time facial recognition at the door with a remote-control mobile application, is supported by past research. Bello et al. (2024) showed that an ESP32-CAM can capture and analyze facial images to identify authorized individuals without physical contact, and their experiments demonstrated high recognition accuracy and practicality in real-world scenarios, granting access to authorized users while denying unauthorized people. Alsayaydeh et al. (2025) evaluated a biometric IoT-based smart lock with real-time monitoring and alert mechanisms and confirmed that combining biometric authentication with internet connectivity enables the rapid data transfer that is crucial for real-time alerts and remote access control. Permana et al. (2024) likewise built an IoT smart door lock that used firebase for real-time data communication with an Android application. These findings justify employing facial recognition on the ESP32-CAM, which simplifies the overall design, reduces hardware cost, and provides hands-free entry (Bello et al., 2024), and integrating it with a centralized flutter application so that homeowners receive instant security alerts and keep remote control. At the same time, Ho et al. (2016) showed that flaws in commercial smart locks can be exploited to gain unauthorized home access, which underlines the need for authenticated cloud access and strict database rules in the proposed system. The hardware and software selected on this basis are the ESP32-CAM and a solenoid door lock, the Flutter framework, and Firebase for the real-time database and cloud synchronization.

Two alternatives were considered and rejected. Fingerprint scanning is a popular biometric technique for smart locks (Jain et al., 2004), but it requires physical contact with the scanner; facial recognition was preferred because it is contactless and hands-free, which is more convenient and hygienic when a resident is carrying groceries or holding a child. A Raspberry Pi offers the processing power of a mini computer, but it is expensive, bulky, and power-hungry; the ESP32-CAM was selected because it is a compact, versatile, and cost-effective solution for real-time image capture in access control applications (Bello et al., 2024).

METHODOLOGY

The Agile Software Development Life Cycle (SDLC) was selected (Beck et al., 2001). Unlike linear models in which testing occurs only at the end, agile permits continuous testing and improvement, which is essential for an IoT project in which the ESP32-CAM hardware and the flutter application must be tested together and communicate reliably through firebase. development followed five phases (Fig. 2).

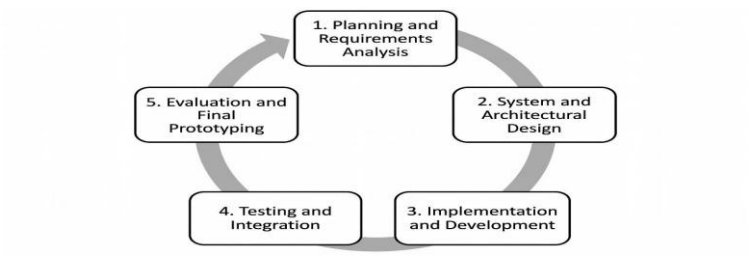


Fig. 2. Agile Methodology

Phase 1 – Planning and requirements analysis. The core problems of traditional locks were identified and the requirements gathered, including the selection of the hardware (ESP32-CAM, solenoid lock) and software platforms (Flutter, Firebase) for a secure, contactless entry system with an S.O.S feature. Phase 2 – System and architectural design. A circuit diagram was planned to connect the ESP32-CAM safely to the solenoid lock and power supply, and the application interface was designed so that the dashboard is easy to navigate, particularly for elderly users who need quick access to the S.O.S button. Phase 3 – Implementation and development. The ESP32-CAM was programmed in C++ through the Arduino IDE to handle real-time facial recognition and trigger the lock, while the mobile application was coded in flutter and integrated with the Firebase cloud database for user profiles, activity logs, and push notifications. Phase 4 – Testing and integration. Testing was continuous and covered facial recognition accuracy under different lighting conditions, the responsiveness of the remote button unlock, and the delivery of the In-App S.O.S notification to the homeowner's phone without delay; bugs were fixed immediately. Phase 5 – Evaluation and final prototyping. The completed hardware and software were integrated into a working door prototype and reviewed against the initial objectives.

The tools required are summarized in Table I. In addition to these, the project required a stable Wi-Fi network, because the ESP32-CAM and the mobile phone communicate through the Firebase cloud; a hardware workbench for soldering and assembly; and a miniature wooden door model on which the lock and camera are mounted for the final evaluation. The project schedule was organized into five stages (initiation and planning; literature review and system design; core development; system integration and testing; and final prototyping and documentation), with buffer time reserved for troubleshooting.

TABLE I. HARDWARE AND SOFTWARE COMPONENTS

Component	Role in the System
ESP32-CAM (OV2640)	Captures faces and connects to Wi-Fi
12V solenoid door lock	Electronic lock mechanism
1-channel relay module	Switches the high-power lock using low-power signals
Buck converter	Steps 12V down to 5V for the ESP32-CAM
FTDI programmer, 12V supply	Code upload and power
Flutter SDK, VS Code / Android Studio	Cross-platform mobile application
Firebase	Realtime database, storage, push notifications
Arduino IDE, GitHub, Visily	ESP32-CAM firmware, version control, UI wireframes

System Analysis

Most residential homes (Fig. 3), especially in guardless neighborhoods, rely on standard mechanical pin-tumbler locks, and the operational flow is entirely manual: the homeowner must search for the key, insert it, and rotate the mechanism, and no one can grant access remotely to a family member or guest. In emergencies, such as a break-in or a medical crisis involving an elderly resident, the resident must step away from the door, find and unlock a phone, dial a contact, and explain the situation, which consumes critical time.

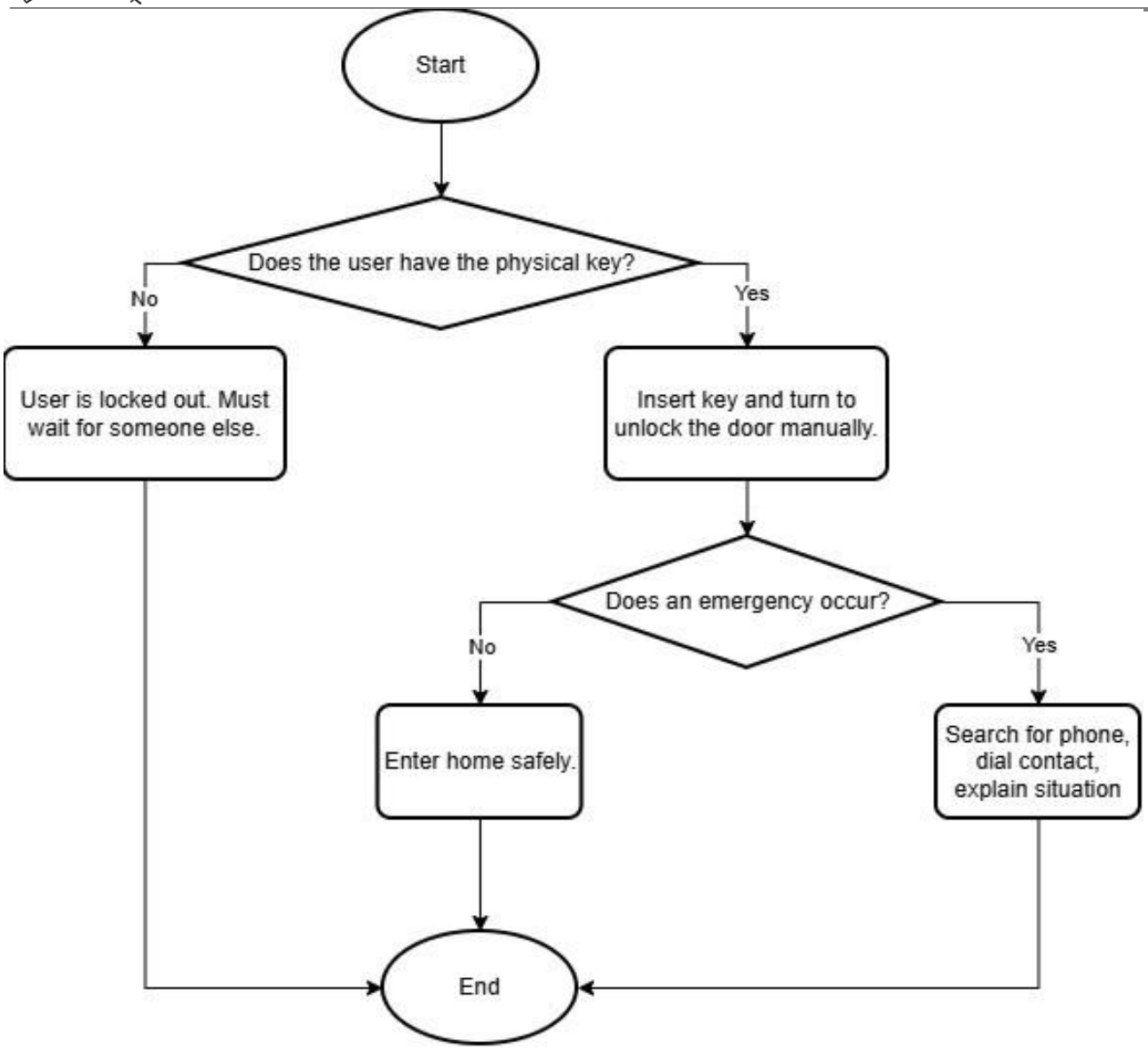


Fig. 3. Flowchart of a Traditional Mechanical Lock System

Four problems were analyzed in detail. First, standard locks are easy to break: intruders can bypass them with lock-picking tools or duplicated keys, and because the lock is offline the homeowner learns of a break-in only on returning home. Second, the door cannot be checked or controlled from afar, so a forgotten lock cannot be verified and a family member who forgets a key cannot be admitted remotely. Third, emergency response is slow because there is no instant, one-click way to alert people outside. Fourth, keys are easily lost, left inside, or stolen, and even fingerprint-based smart locks require the resident to put down groceries or a child to touch the scanner.

The data requirements are as follows (Fig. 4). The primary input is the live facial image captured by the ESP32-CAM; secondary inputs are the unlock door, Secure Mode, and S.O.S Emergency buttons in the application. The physical output is the electronic signal that triggers the relay and solenoid lock, and the digital outputs are push notifications such as “Door Unlocked” (by button or face), “Intruders Alert”, and “S.O.S Alert”. Firebase stores user profiles (names, registered face IDs, and contact details) and access logs (the time and the user for each unlock). The functional requirements are facial recognition and access control, remote override from any location, emergency alerting of all connected family members through a high-priority S.O.S notification, and activity logging with a timestamp for every access event. The non-functional requirements are a performance target of 5–8 seconds for face identification and unlocking and less than 5 seconds for app commands; high accuracy that prevents false positives and minimizes false negatives; availability of 24/7 operation without overheating or crashing; and storage of at least 30 days of access logs without slowing the application.

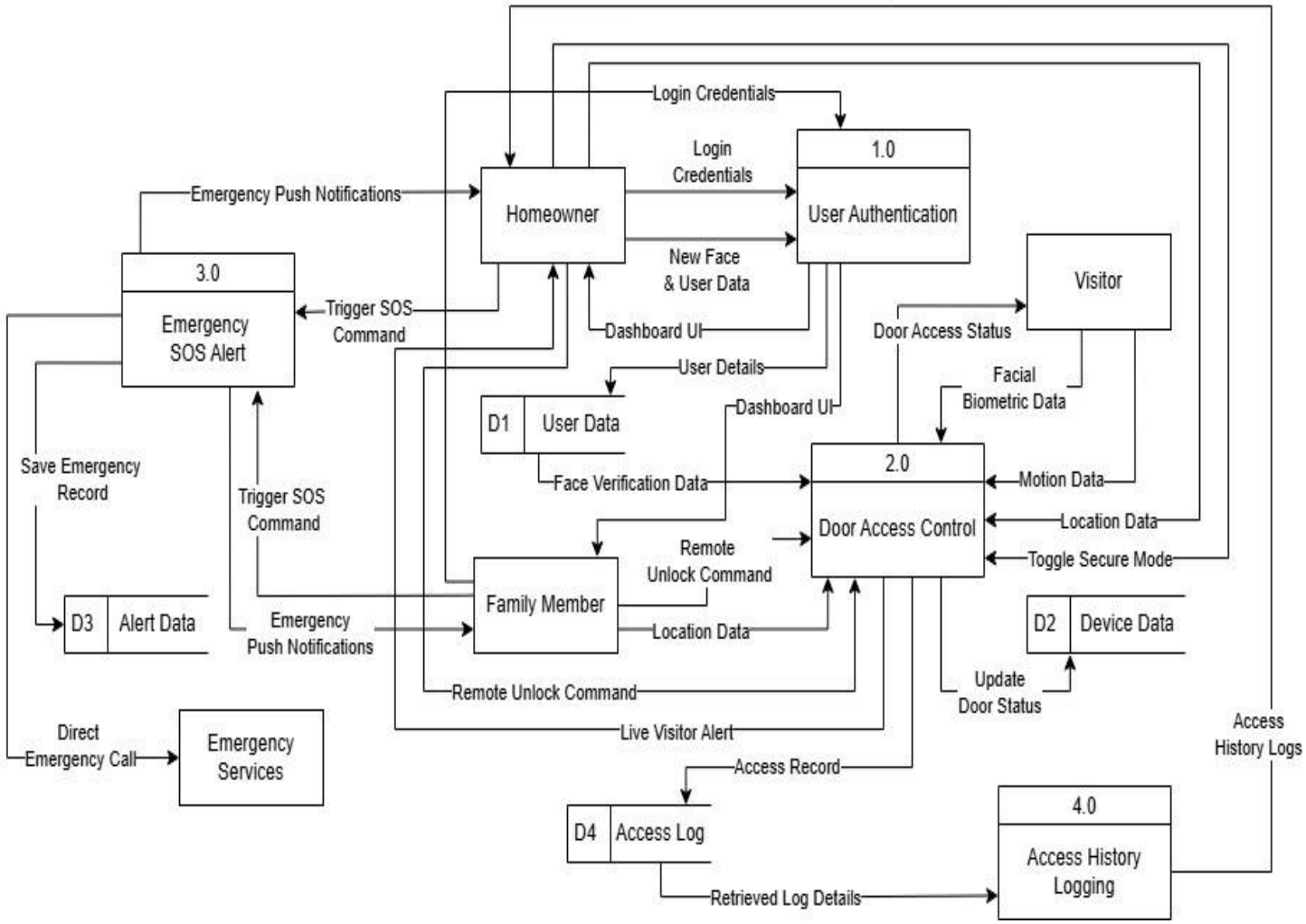


Fig. 4. M.A.T.A Data Flow Diagram Level 1

The technology choices follow from these requirements. Flutter builds the application for android and iOS simultaneously, which saves development time; Firebase offers real-time database capabilities suited to instant IoT communication; and visibly was used to map the interface clearly for elderly users. The ESP32-CAM was chosen over a raspberry pi because it is compact, cost-effective and has built-in Wi-Fi; the 12V solenoid lock provides a heavy-duty barrier against forced entry; and the buck converter is strictly required because excess voltage would permanently damage the ESP32-CAM.

System Design

A. System Architecture

The M.A.T.A system follows a four-layer architecture (Fig. 5) to ensure modularity, scalability and ease of maintenance. The user layer supports the homeowner and family member roles, each with different privileges for managing profiles, controlling remote access, and monitoring security activity. The presentation layer is the flutter application, which handles secure login and profile management, dashboard navigation, remote unlocking, access history, secure mode, and the S.O.S trigger; user actions generate API requests to the layer below. The application logic layer integrates real-time facial scanning through the ESP32-CAM, access control logic that verifies authorized users, intruder alerts during secure mode, S.O.S alert processing, and hardware-to-cloud synchronization. The backend layer is powered by firebase: Firebase authentication manages secure login and role-based access control, the firestore database stores user profiles, pointers to registered biometric data, remote unlock logs, and access history, and firebase cloud messaging (FCM) delivers push notifications for real-time door access, intruder alerts, and S.O.S warnings. The layered design allows future enhancements with minimal impact on existing components.

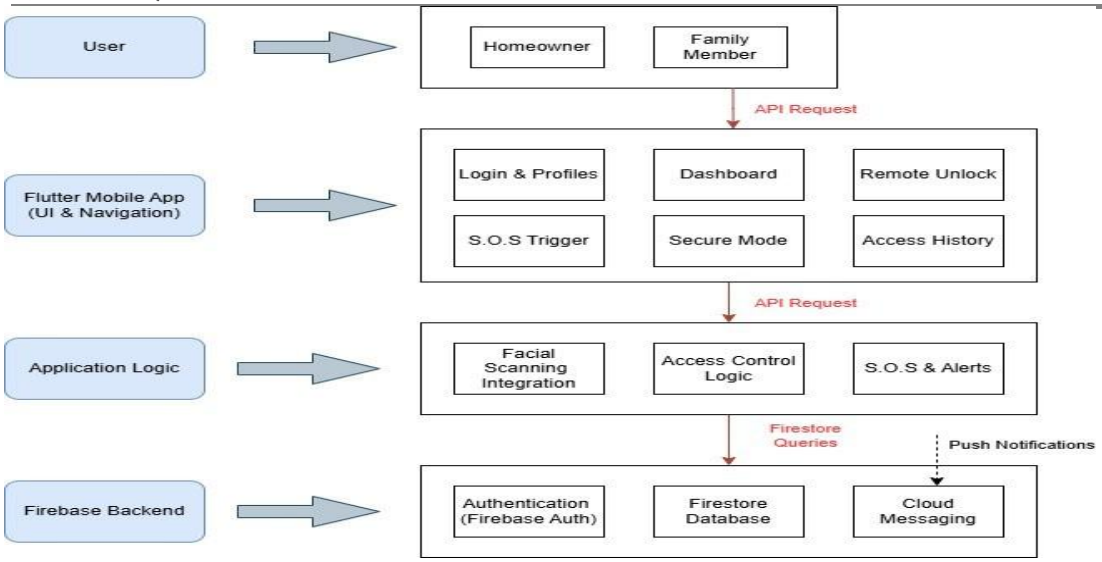


Fig. 5. System Architecture

B. User Interface Design

Because M.A.T.A is both a daily access tool and an emergency system, the interface prioritizes simplicity, speed, and accessibility, so that a resident does not waste time in cluttered menus during a breach or medical emergency. It uses a high-contrast dark theme with distinct, recognizable buttons for primary functions, such as the S.O.S trigger and remote unlock. Color coding is applied strategically: urgent red tones signify security breaches and alarms, whereas calming blue and green tones indicate standard, secure access. The designed screens (Fig. 6) cover the secure login, face unlock, remote unlock, S.O.S trigger, secure ode, event logging with image capture, the notification centre, and member registration.

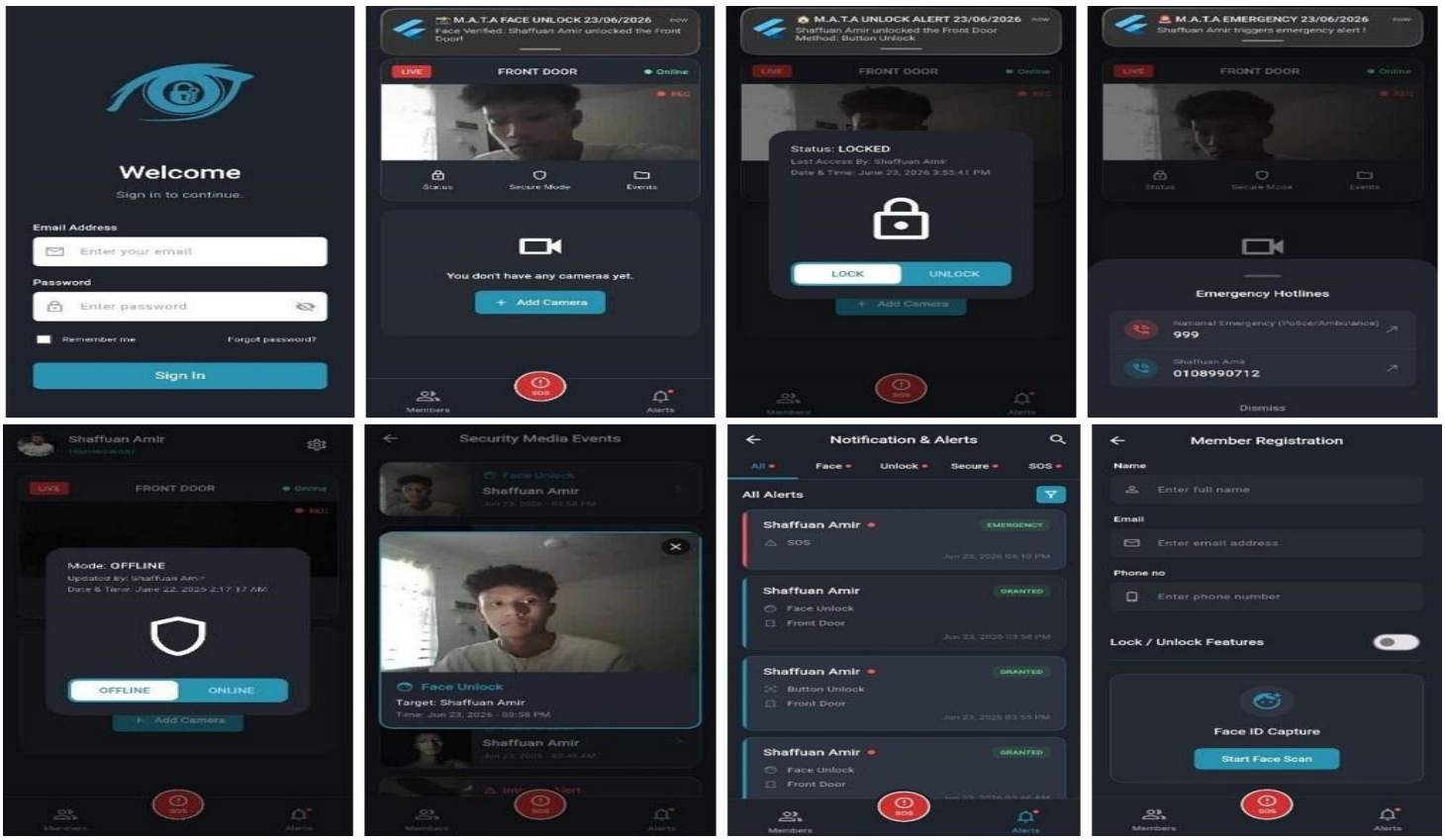


Fig. 6. User Interface of M.A.T.A: Login, Face Unlock, Remote Unlock, S.O.S, Secure Mode, Event Logging, Notification Centre, and Member Registration

C. Database Design

The database is described through an Entity Relationship Diagram (Fig. 7), business rules, and a data dictionary. USER stores profile and authentication data, including the biometric identifier (FaceData); ROLE defines hierarchical permission levels (Homeowner, Family Member); DEVICE represents the ESP32-CAM and lock deployed at a location; ACCESS is an associative entity that resolves permissions between users and devices, including button permissions; LOG is the audit trail that records the timestamp, method (Face Unlock or App Unlock), and image proof through MediaURL; EMERGENCY records S.O.S events; and ALERT manages the push notifications dispatched to the application, tracking whether each is notified or read.

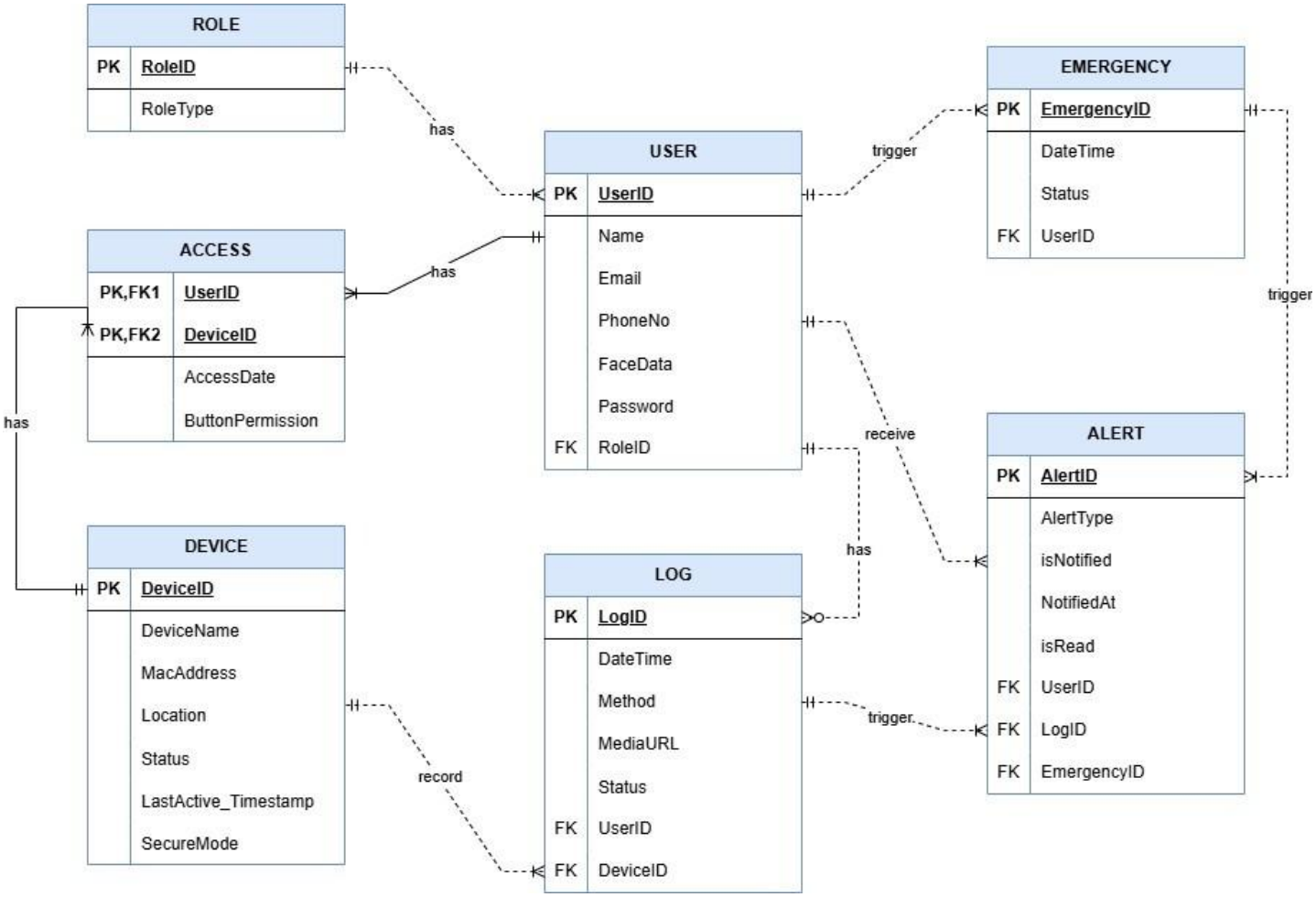


Fig. 7. Entity Relationship Diagram (ERD)

The business rules enforce data integrity. A Role can be assigned to many Users, but each User has exactly one Role, which enforces strict role-based access control through the RoleID foreign key. A User can trigger zero, one, or many Emergency events, and each Emergency belongs to exactly one User, which gives an audit trail for critical situations. A Device can record many Logs, and each Log belongs to one Device and one User, so history can be filtered by person and door. A Log or an Emergency can trigger zero or many Alerts, since not every unlock requires a push notification, and each Alert is dispatched to exactly one User, so that high-priority warnings reach the registered user and whether they have been read can be tracked.

Because Firebase Firestore is a cloud-based NoSQL database, the logical tables become collections and the rows become JSON documents, and constraints are enforced through Firestore Security Rules rather than SQL DDL. For example, the rules for the USER collection allow only authenticated users to read and write and validate that Name and Email are strings and that RoleID is either Homeowner or Family. Core text data such as profiles, timestamps, and lock statuses sit in Firestore documents, whereas heavy binary files, namely the photographic evidence captured by the ESP32-CAM, are stored in Firebase Cloud Storage and referenced by a lightweight URL, keeping the database small and retrieval fast. Firebase creates single-field indexes automatically, and a

composite index is defined in the Firebase Console to support complex queries, such as listing all logs with a Denied status sorted by most recent date.

D. Software Design

Each major feature was documented as a program specification that lists its purpose, input, output, and screen (Table II). Secure Mode arms the ESP32-CAM's motion detection so that any unverified movement at the door triggers an Intruder Alert. To save cloud storage, the event logger uploads photographic evidence only during a Face Unlock attempt or when unauthorized motion is detected in Secure Mode. Member registration strictly enforces role-based access control so that only the Homeowner can create profiles, assign biometric FaceData, and grant device access.

TABLE II. PROGRAM SPECIFICATIONS OF THE CORE MODULES

Module	Input	Output
Secure login (dual authentication)	Email and password or local face scan	Home page access or error message
Face verification (ESP32-CAM)	Live image, registered FaceData	Unlock signal, access log, notification
Remote app unlock	Button tap, UserID, timestamp	Unlock via cloud update, access log
Emergency S.O.S trigger	Button tap, UserID, timestamp	Emergency record, FCM high-priority alerts
Secure Mode activation	Toggle, UserID, target status	Status update, camera armed, confirmation
Event logging and image capture	Motion or face trigger, image frame, mode	Image URL in Storage, log document
Notification centre	UserID, ALERT records	Alert cards, isRead update
Member registration	Homeowner token, member details, FaceData	New profile, USER and ACCESS documents

Implementation

The hardware prototype (Fig. 8) uses the ESP32-CAM to manage real-time facial recognition. A buck converter steps the 12V supply down to 5V for the microcontroller and a 1-channel relay module switches the 12V solenoid door lock, allowing a safe low-power signal to control the lock.

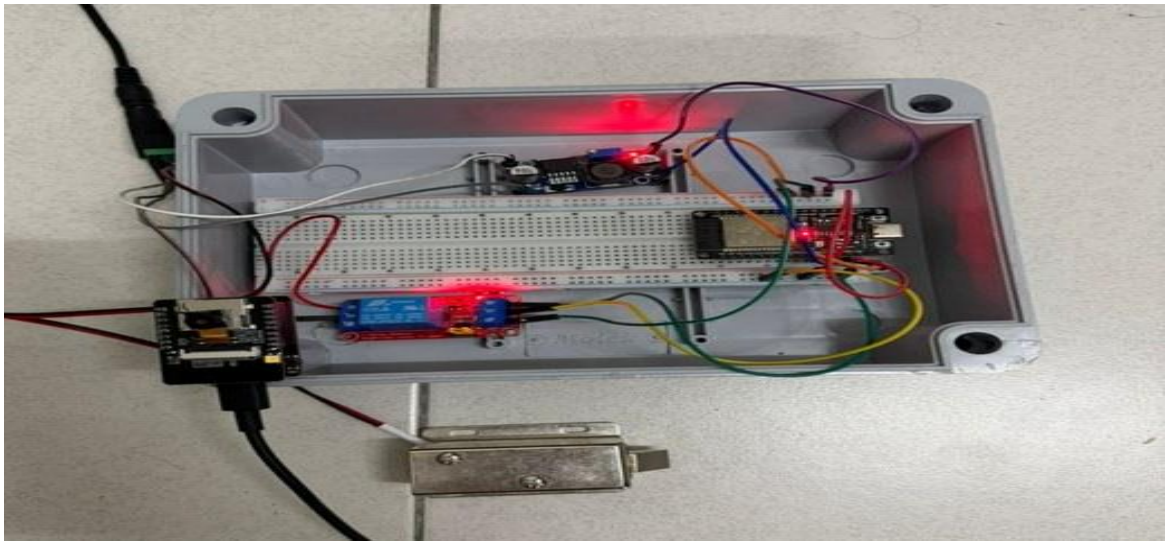


Fig. 8. Hardware Prototype Assembly of M.A.T.A

The development environment was configured with the Flutter SDK and Android Studio and Visual Studio Code with the Flutter, Dart, and Firebase extensions. A GitHub repository provided version control and synchronization across devices, and a Node.js server was set up on the local network to provide the facial recognition verification API that receives image data from the Flutter application. In the Firebase Console a project named "MATA" was created, and the Android and iOS versions of the application were registered by embedding the generated google-services.json and GoogleService-Info.plist files. Firebase Authentication was enabled with the Email or Password provider for standard logins and to support custom tokens for the facial recognition login flow.

Four root collections were initialized in Firestore: *users* (profiles, contact details, and biometric tracking URLs), *status* (single documents tracking the door as Locked or Unlocked and Secure Mode as Online or Offline), *notifications* (Button Unlock, Face Unlock, Intruder, and S.O.S alerts) and *logs* (metadata linking to photographic evidence). A separate Cloud Storage bucket holds profile pictures and the biometric image frames captured by the ESP32-CAM. The Security Rules were changed from test mode to strict authentication mode, so that reading and writing require a valid authenticated token, and users may update only their own profile document.

Because Firestore has no stored procedures or database-level triggers, application logic is programmed in asynchronous Dart functions in the Flutter application or in the Node.js server. For example, Secure Mode automation is implemented in Dart: when the user enables the smart timer, the security status document is updated with the user's name, the update time, and an optional automatic offline time; a notification document is created; any existing timer is cancelled; and a local timer is started that deactivates Secure Mode automatically when the scheduled time ends. The database is populated in real time through three interactions: manual registration of family members by the Homeowner (typed data go to Firestore and image bytes go to Firebase Storage, with the download link saved in the user document); hardware logs, in which the Node.js server uploads the captured frame to Storage and writes a log document with a timestamp and media link when a face scan succeeds or unauthorized motion is detected in Secure Mode; and dashboard buttons, such as S.O.S and Unlock, which write notification and status documents that are retrieved automatically on every family member's device.

TESTING AND RESULTS

Testing was organized around three roles: the system developer, who executed the unit, integration, and system tests; the project supervisor, who validated the prototype and results; and target users representing homeowners, family members, and elderly occupants, who participated in User Acceptance Testing (UAT). Testing took place in a controlled laboratory environment at UTeM, which allowed the Node.js terminal and Firebase logs to be monitored, and on the miniature wooden door model, which replicated a house entryway under varying lighting conditions. The ESP32-CAM was mounted at an optimal angle and wired to the relay, lock, and buck converter, its firmware was flashed through the Arduino IDE to handle live MJPEG streaming and HTTP request routing, and the Node.js verification server ran on a static local IP address, while the test smartphones ran the latest build of the Flutter application. A dedicated, stable Wi-Fi hotspot connected all components, old debugging data were cleared from Firestore, and the participants received a short orientation on the dark-themed dashboard, face registration, Secure Mode, and, for elderly participants, the red S.O.S trigger.

The 14-day testing phase was divided into three cycles: unit and component testing (4 days), system integration testing (6 days), and UAT (4 days). A bottom-up strategy was adopted in which the ESP32-CAM, the relay and lock, and the Node.js server were tested first, then the Firebase integration, and finally the Flutter application, so that communication errors are isolated before they reach the user dashboard. Black-box testing evaluated the system from the end user's perspective. The classes of tests were functionality (S.O.S broadcasts alerts, Button Unlock triggers the relay, Secure Mode changes the system status), security (an unregistered face cannot bypass recognition and Security Rules block unauthenticated access), and performance and stress (repeated Unlock presses and rapid successive face scans). Real face scans of the developer and participating family members and real Android and iOS smartphones were used as real-life data, whereas dummy profiles, mock email addresses, and placeholder pictures served as synthetic data.

Twenty-four test cases were executed across five modules (Table III), and all of them passed. They confirmed, among other behaviors, that mismatched faces and incorrect passwords are rejected without crashing the application, that role restrictions are enforced in member management and hardware control, that Secure Mode timers and motion alerts work as specified, and that notifications, logs, and profile functions behave as expected.

TABLE III. TEST CASES AND EXPECTED RESULTS (ALL PASSED)

ID	Action / Input	Expected Result
AUTH-01	First launch after installation	Splash screen; "Let's Get Started!" opens login and remembers setup
AUTH-02	Click password field after valid registered email	Camera scanner opens to verify the face
AUTH-03	Face scan does not match the account	Access denied with error; user stays on login screen
AUTH-04	Incorrect password	"Incorrect password" warning; no crash
MEM-01	New member details without face scan	Registration blocked; face scan required
MEM-02	Successful member registration	New member added to the system
MEM-03	Delete a family member profile	Confirmation dialog; member removed on confirmation
MEM-04	View member details to edit	Email field locked
MEM-05	Standard member views member list	Delete button hidden
HW-01	Authorized user taps unlock	Door unlocks; app shows door open
HW-02	User without lock permission taps unlock	Action blocked; "Access Denied" dialog
HW-03	Timed Secure Mode (e.g., 10 minutes)	Countdown; Secure Mode turns off automatically
HW-04	Secure Mode on; movement detected	Instant alert; photo saved as evidence
HW-05	Camera loses Wi-Fi connection	"Camera Offline" message; no freeze or crash
HW-06	Manual unlock via status dialog	Icon shows open; reverts to Locked after 3 s
NOT-01	S.O.S or intruder alert received	Card highlighted red with warning icon
NOT-02	Tap an unread notification	Detail popup; unread dot removed; marked read
NOT-03	Type member name in search bar	List filters to that member's logs
NOT-04	Select date range filter	Logs and photos outside the range filtered out
PRO-01	Open profile; make no changes	Save Changes stays disabled
PRO-02	Change password	Live face scan required first
PRO-03	Select new profile picture	Crop tool forces square shape before upload
PRO-04	Tap sign-out	Confirmation popup before logout
PRO-05	Click download in analytics	PDF report of security logs generated

Most test cases passed on the first iteration, verifying that real-time facial recognition through the ESP32-CAM, remote overrides, and Firebase synchronization work together smoothly. Edge cases, such as incorrect passwords, network drops on the camera stream, and unauthenticated database access, were handled by the system's exception handling without crashes, and minor interface feedback issues identified during testing were patched in the Flutter codebase. The testing therefore indicates that the prototype meets the functional and non-functional requirements and is stable, secure, and operational.

DISCUSSION

The principal strength of the system is the contactless and seamless entry provided by ESP32-CAM facial recognition, which is convenient for residents carrying groceries or who often misplace keys. The centralized S.O.S protocol offers vulnerable occupants, especially the elderly, an instant way to seek help without dialing a number in a panic, and the Flutter application gives homeowners full remote management through access log monitoring, manual door control, and a Secure Mode timer. Feedback from target users representing family members was highly positive: they felt significantly safer in a guardless residential area because the live visitor alert and emergency features are easily accessible on their smartphones.

Two limitations were identified. The system depends strictly on the home's Wi-Fi and electricity, so a power outage or router failure stops cloud synchronization and makes the remote features temporarily inaccessible. Besides, because the ESP32-CAM uses a standard OV2640 optical lens, recognition accuracy decreases slightly at night or in poorly lit porch environments. Accordingly, three enhancements are proposed: an uninterruptible power supply or a 12V backup battery together with a SIM-based GSM module that sends SMS alerts for S.O.S and intruder warnings when Wi-Fi fails; an ESP32-CAM with infrared sensors and an IR-cut lens so that the Node.js backend can verify faces in complete darkness; and a Guest Pass feature in which the homeowner generates a temporary, time-limited QR code for a visiting relative or delivery driver that the ESP32-CAM reads, to eliminate the requirement for permanent guests registration. The project benefits the community by replacing insecure traditional locks with a proactive system that ensures elderly residents stay connected during emergencies.

CONCLUSION

This study addressed the vulnerability of key-based door locks in guardless residential areas, where homes are often left unoccupied or occupied only by elderly residents and homeowners cannot monitor or control access remotely. To this end, M.A.T.A was designed and developed as an IoT-based smart door lock that combines contactless facial recognition, remote control, and emergency response in a single system. All four objectives were achieved. The ESP32-CAM transformed a standard electronic solenoid lock into an intelligent entryway that grants access to registered household members; the Flutter application serves as the main control center for remote unlocking, access log monitoring, Secure Mode, and emergency S.O.S alerts; and Firebase synchronizes access logs, image evidence, and push notifications between the lock and the application in real time.

The prototype nevertheless has limitations: it depends on Wi-Fi and mains power, and recognition accuracy decreases slightly in poorly lit conditions. Future work will therefore add a backup power supply with a GSM module for SMS alerts, an infrared-capable camera for recognition in darkness, and a time-limited QR guest pass for visitors. By combining biometric technology with cloud infrastructure in this way, M.A.T.A offers a practical path toward a smarter and safer living environment for guardless communities.

REFERENCES

1. Alsayaydeh, J. A. J., Yusof, M. F., Mamchenko, S., Hamzah, R. A., & Herawan, S. G. (2025). Design and evaluation of a biometric IoT-based smart lock system with real-time monitoring and alert mechanisms. International Journal of Advanced Computer Science and Applications, 16(7). https://thesai.org/Downloads/Volume16No7/Paper_37-Design_and_Evaluation_of_a_Biometric_IoT.pdf
2. Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J.,

Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2001). Manifesto for agile software development. https://agilemanifesto.org

3. Bello, O. O., Obasanya, T. D., & Ajayi-Smart, E. A. (2024). ESP32-CAM module facial recognition door lock security system. Journal of Engineering Technology and Innovation (Special Issue for IEC 2024), 1–11.
4. Google. (n.d.). Firebase documentation. Retrieved October 21, 2026, from https://firebase.google.com/docs
5. Google. (n.d.). Flutter documentation. Retrieved October 21, 2026, from https://docs.flutter.dev
6. Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. Future Generation Computer Systems, 29(7), 1645–1660. https://doi.org/10.1016/j.future.2013.01.010
7. Ho, G., Leung, D., Mishra, P., Hosseini, A., Song, D., & Wagner, D. (2016). Smart locks: Lessons for securing commodity Internet of Things devices. In Proceedings of the 11th ACM Asia Conference on Computer and Communications Security (pp. 461–472). ACM. https://doi.org/10.1145/2897845.2897886
8. Jain, A. K., Ross, A., & Prabhakar, S. (2004). An introduction to biometric recognition. IEEE Transactions on Circuits and Systems for Video Technology, 14(1), 4–20. https://doi.org/10.1109/TCSVT.2003.818349
9. Permana, K. A. K., Piarsa, I. N., & Wiranatha, A. A. K. A. C. (2024). IoT-based smart door lock system with fingerprint and keypad access. Journal of Information Systems and Informatics, 6(3), 2086–2098. https://doi.org/10.51519/journalisi.v6i3.844