APPLIED INFORMATION TECHNOLOGY AND COMPUTER SCIENCE
e-ISSN: 2773-5141

AITCS

Vol. 6 No. 2 (2025) 1644-1663
https://publisher.uthm.edu.my/periodicals/index.php/aitcs

IoT- Integrated Built-In Doorbell and Monitoring System

Jeviena Rani Sanmuga Raja[1], Azizul Azhar Ramli[1]*

[1] *Fakulti Sains Komputer dan Teknologi Maklumat*
Universiti Tun Hussein Onn Malaysia, Parit Raja, Batu Pahat, 86400, MALAYSIA

*Corresponding Author: azizulr@uthm.edu.my
DOI: https://doi.org/10.30880/aitcs.2025.06.02.088

Article Info

Received: 12 June 2025
Accepted: 3 November 2025
Available online: 30 November 2025

Keywords

IoT-Based Security, Smart Doorbell, Monitoring System, Real-Time Notifications, Raspberry Pi.

Abstract

The rapid evolution of security needs has outpaced traditional systems, which often lack real-time notifications, remote accessibility, and integration, creating vulnerabilities that demand smarter, connected solutions. VisionDoor: Built-in Doorbell and Monitoring System leverages IoT technologies to bridge these gaps, offering an efficient and comprehensive smart doorbell solution. This project focuses to desig, developing and testing VisionDoor to enhance security through real-time monitoring, facial recognition, motion detection, and remote access. By prioritizing user-friendliness and cost-effectiveness, it caters small-scale applications and modern security challenges. A Prototyping-Oriented Software Development Life Cycle is chosen to ensure iterative development, enabling continuous refinement through real-world testing. The system is developed using Raspberry Pi, the OpenCV library, and a Flutter-based mobile app with Firebase for real-time data processing. VisionDoor demonstrates improved security management through features like facial recognition, real-time notifications, efficient motion detection, and remote access, offering enhanced user control over premises and reducing response times. By combining advanced features such as automated notifications and real-time monitoring, VisionDoor empowers users to maintain robust security remotely. Its dual modes Sentry and Normal, offer flexibility for continuous monitoring or targeted alerts, meeting the needs of residential and small business users alike. The findings underscore the transformative impact of IoT on security, paving the way for smarter, more connected systems.

1. Introduction

The Internet of Things (IoT) has transformed the security landscape by integrating smart, connected systems with real-time capabilities to enable data sharing and advanced functionality [1]. However, traditional CCTV and standalone doorbells still face issues such as delayed notifications, limited remote access, and vulnerability to tampering [2]. Even wireless doorbells, though easier to install, often lack integration with broader monitoring systems. These limitations highlight the need for a more secure and unified solution. In response to these challenges, the objective of this project is to develop a comprehensive IoT-based doorbell and monitoring system that merges essential smart security functions into a single platform. This includes implementing facial recognition, motion detection, cloud-based data management, and real-time mobile notifications. Compared to conventional security systems, the proposed solution offers several clear advantages. Automation through facial recognition reduces the need for manual verification or physical keys. Real-time alerts and remote access allow users to respond promptly to security events regardless of their location. The system also improves data efficiency

by capturing and storing only relevant events, reducing unnecessary storage usage [3]. Additionally, the integration of all components into a tamper-resistant design enhances physical security and reduces the likelihood of unauthorized interference. These combined features make the system more robust, responsive, and convenient than existing solutions. To meet these goals, the VisionDoor: IoT-Integrated Built-In Doorbell and Monitoring System was developed as a unified and intelligent security platform [4]. The system operates in two primary modes: Sentry Mode, which enables continuous surveillance through motion detection and facial recognition, and Normal Mode, which is triggered by doorbell activation. Users receive real-time notifications via a mobile application and can remotely control door locks, lighting modes, and view live camera feeds. The facial recognition module distinguishes between registered and unknown individuals, providing smart access control. VisionDoor's design is tamper-resistant, with components embedded within the door structure to minimize visibility and protect against sabotage. Powered by a Raspberry Pi and integrated with sensors, cameras, and Firebase cloud services, the system delivers a reliable, efficient, and scalable solution that redefines smart home and small business security.

2. Related work

Security and monitoring systems have evolved significantly with the integration of technologies like IoT, facial recognition, and mobile connectivity [5]. These advancements aim to enhance user convenience, functionality, and reliability [6]. This section reviews related works, highlights limitations of current solutions, and sets the foundation for the VisionDoor system by comparing it with notable systems such as Ring, ADT, and TimeTec.

2.1 Internet of Things (IoT)

Traditional security systems, such as Closed-Circuit Television (CCTV), often fall short of addressing modern challenges due to the lack of advanced features. While they offer basic surveillance, they typically do not support real-time notifications, facial recognition, or seamless mobile app integration. Likewise, standalone doorbells and motion sensors provide limited functionality and lack the flexibility and responsiveness needed for effective security management, underscoring the growing need for more intelligent and integrated systems. Recent technological advancements have introduced IoT-enabled solutions that combine motion detection, facial recognition, and mobile connectivity. These systems create highly efficient and responsive environments by enabling seamless communication between interconnected devices and empowering users with remote, real-time control capabilities [7]. Together, these features form the backbone of modern smart security solutions, offering robust and user-friendly functionality for homeowners and businesses alike. The VisionDoor system builds on this foundation by leveraging a monitoring framework that includes cameras, motion sensors, and a Raspberry Pi for data processing. Its interconnected IoT components enable real-time notifications, live video feeds, and remote access via a mobile application. By ensuring seamless communication and automation across all devices, VisionDoor delivers an efficient, responsive, and modern security solution tailored to both residential and business environments.

3. Facial Recognition

Facial recognition plays a critical role in enhancing the security of IoT-enabled systems by enabling identity verification and restricting unauthorized access [8]. Facial recognition adds an additional layer of security by enabling identity verification, while mobile applications offer users the ability to receive instant notifications and interact directly with their systems [8]. In traditional systems, users often rely on keypads or access cards, which are vulnerable to theft or misuse. Facial recognition addresses this by adding a biometric layer to access control. The VisionDoor system incorporates facial recognition using the OpenCV library and LBPH algorithm, allowing it to identify both known and unknown individuals in real time [9]. When motion is detected, the system captures facial data and compares it against a registered dataset, granting or denying access accordingly. This feature enhances security by reducing reliance on physical access methods and providing personalized authentication for trusted users.

3.1 Mobile Application (VisionDoor)

Mobile applications are central to the functionality of modern IoT-based security systems. They allow users to receive real-time notifications, control system components remotely, and access live video streams. Systems without robust mobile integration often suffer from poor user engagement and delayed responses during security events. The VisionDoor mobile application, developed using Flutter SDK and optimized with Android Studio, provides cross-platform compatibility and ensures a responsive user experience [10]. Through the app, users can lock or unlock doors, control entryway lighting, view live camera feeds, switch between Normal and Sentry modes, and receive door status updates instantly through cloud-based architecture for real-time data processing. VisionDoor ensures constant monitoring and improved security management for homes and businesses [11].

3.2 Comparison with the Existing Systems

Several existing systems such as Ring, ADT, and TimeTec incorporate elements of modern security technology, but each has its limitations. Ring offers video streaming without facial recognition, TimeTec enables keyless entry but lacks HD video and biometrics, while ADT provides HD surveillance but not biometric access. These partial solutions highlight the need for a more integrated approach. A detailed comparison is presented in Table 1.

Table 1 *System Comparison*

Features \ System	TimeTec Security	ADT Smart Home	Ring Video Doorbell	VisionDoor
User Authentication	User ID and password	Fingerprint	User ID and password	User ID and password
Visitor Management	Tracks visitor entry/exit	X	Basic visitor alerts	Custom visitor alerts
Motion Detection	Mobile notifications	Customized sensitivity	Mobile notifications	Adjustable sensitivity
Remote Access Control	Mobile notifications	Mobile app integration	Mobile notifications	Dedicated mobile app
High-Definition Video	X	HD video surveillance	HD streaming	HD capture
Two-Way Communication	X	X	Two-way audio	Optional two-way audio
Emergency Alerts	X	Instant emergency alert	X	Optional custom alerts
Visitor Management	Tracks visitor entry/exit	X	Basic visitor alerts	Custom visitor alerts
Motion Detection	Mobile notifications	Customized sensitivity	Mobile notifications	Adjustable sensitivity

The proposed VisionDoor fills this gap by combining key features such as facial recognition, dual security modes, HD video, motion detection, and full mobile control into a unified, user-friendly system for comprehensive home and business security. VisionDoor provides a seamless and efficient security experience that empowers homeowners and businesses to monitor and manage their security systems remotely [10].

4. Methodology

The Prototyping Methodology was employed in the development of the VisionDoor: Built-In Doorbell and Monitoring System, given its effectiveness in supporting iterative design and continuous user engagement. This methodology emphasizes the early development of functional prototypes, enabling stakeholders to visualize system functionality and provide timely feedback. Such an approach ensures that the final product is closely aligned with user requirements and expectations. The sequential phases of planning, designing, building, testing, refining, and finalizing the prototype is depicted in Fig. 1.

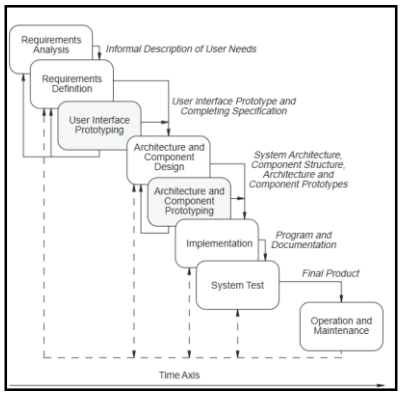


Fig. 1 Prototyping - Oriented Software Life Cycle

Applied Information Technology and Computer Science Vol. 6 No. 2 (2025) p. 1644-1663

Prototyping methodology emphasizes user-centric design through continuous feedback. For VisionDoor, it was vital in addressing integration challenges of IoT, facial recognition, and security features. Stakeholder involvement allowed iterative refinement and resolution of issues like motion sensitivity, notifications, and hardware-software integration. Table 2 outlines the tasks conducted using this approach.

Table 2 *Software development activities*

Phase	Task	Output
Requirement Analysis	Conduct user interviews, surveys, and market research to gather insights and identify opportunities for improvement.	Gantt Chart, Project proposal
Requirement Definition	Define functional and non-functional requirements and validate them with stakeholders.	System requirements, UML Diagram, Activity Diagram, Sequence Diagram, Class Diagram and Flowchart
User Interface Prototyping	Create low-fidelity wireframes and gather user feedback for iterative refinement.	System architecture, Database schema and data dictionaries, Wireframes or mock-ups.
Architecture and Component Design	Specify system architecture, component interactions, and technical requirements.	Component Specification
Architecture and Component Prototyping	Develop initial prototypes and conduct preliminary testing for functionality and integration.	Functional prototypes, Prototype test results.
Implementation	Develop the mobile app using Flutter, integrate Firebase, and connect hardware components.	Fully developed mobile app, Integrated system (hardware + software)
System Test	Conduct unit, integration, and user acceptance testing to ensure system functionality and user satisfaction.	Unit testing results, Integration testing results, User acceptance testing results.

Table 2 outlines the software development phases for VisionDoor, guided by an object-oriented and user-centered approach. It begins with requirement analysis through user research, followed by defining system specifications using UML diagrams. Interface prototyping and architectural design lead to initial functional components. The mobile app is then developed using Flutter, integrated with Firebase, and connected to hardware. Final system testing ensures functionality and user satisfaction through unit, integration, and acceptance testing.

4.1 System Requirements

This section outlines the system requirements that include functional, non-functional, user and technical requirements. The main features of the system. The system's functional modules include user registration and login, set mode (Sentry/Normal), light control, door control, motion monitoring, facial recognition, notification and alert, chatroom and reporting. Table 3 and 4 show the functional and non-functional requirements for the proposed system. Table 3 below shows the functional requirement modules of the VisionDoor system.

Table 3 *Functional Requirements*

Modules	Functionality
User Registration and Login Module	- The system should allow the new users to register a new account before logging in. - The system should allow the existing users to login with valid email and password - The system should notify users of any invalid input such as invalid email or incorrect password. - The system should redirect the valid users to dashboard once successful login. - The system should allow users to reset password through verification and reset link sent to registered email.

Table 3: (cont.)

Modules	Functionality
Set Mode (Sentry/ Normal) Module	- Sentry Mode: - The system should allow the user to activate a monitoring mode when mode is set to sentry for continuous observation of the surroundings to detect motion. - The system should enable the detection and recognition of both known and unknown faces. - The system should unlock the door if it is a recognized face, else door remains locked. - A triggers camera for facial recognition id motion is detected. Normal Mode: - The system should activate monitoring when the doorbell is pressed. - The system should allow user to lock and unlock door manually and view live stream.
Light Control Module	- The system should allow users to remotely control the lighting around the entryway through the mobile app. - The system should offer multiple lighting modes such as turning the lights on or off, activating night mode, and adjusting the brightness to suit user preferences and security needs.
Door Control Module	- The system should allow users to lock and unlock the door remotely through the mobile app. - The system should enable premise owners to conveniently manage door access.
Motion Monitoring Module	- The system should detect motion in the entryway, particularly while operating in Sentry Mode. - The system should send notifications to the user for movements, especially during odd hours.
Notification and Alert Module	- The system should notify users in real-time for door status - "Door is unlocked" and "Door is locked"
Chatroom and Reporting	- The system should allow users to view, store, and manage messages through an intuitive interface resembling a chatroom. - The system should enable users maintain detailed logs on door activity of all interactions for later review

Table 3 outlines the functional requirement modules of the VisionDoor system, highlighting their purposes and features.

Table 4 *Non-Functional Requirements*

Requirements	Description
Performance	- The system should always be usable. - The system should be able to detect motion. - The system should be able to recognize known and unknown faces.
Operational	- The system should be user-friendly. - The loading time required for the app is no more than 1 minute.
Security	- The system should only allow users to access their own account with email and password.
Cultural and political	- The system should be able to work on any Android 8.0 (Android API 26) and above devices.
Performance	- The system should always be usable. - The system should be able to detect motion. - The system should be able to recognize known and unknown faces.

Table 4 outlines the non-functional requirements of the VisionDoor system, covering performance, operational, security, and platform compatibility aspects.

Table 5 *User Requirements*

No.	Requirement
1.	All users must have an account with a valid email and password to access the system.
2.	Premise owners should be able to switch between Sentry mode and Normal mode for monitoring activities.
3.	Premise owners should be able to control and manage door access, including locking and unlocking the door remotely.
4.	Premise owners should be able to adjust light modes
5.	Premise owners should be able to view notifications on door activity through the mobile app.
6.	Premise owners should be able to review logs door activity, including for later reference.
7.	Premise owners can add notes and messages in the chatroom to be reviewed when necessary.

Table 5 outlines the seven user requirements for the VisionDoor System.

4.2 System Analysis

The VisionDoor: Built-In Doorbell and Monitoring System is represented through various diagrams that illustrate its functionality, design, and workflows. These include the flowchart, UML diagrams, class diagram, sequence diagram, and activity diagram. Each of these diagrams provides a detailed view of the system's operations, interactions, and architecture, offering a comprehensive understanding of the VisionDoor system. The use case diagram visually represents the interactions between users and the VisionDoor system, identifying the premise owner, visitors, and the smart door as primary actors. It outlines how the system supports key functionalities, ensuring its usability and alignment with user needs. Appendix A, B and C depict these diagrams for the developed system.

Appendix A illustrates the use case diagram for the VisionDoor: IoT-Integrated Built-In Doorbell and Monitoring System, showing interactions between the Smart Door, Premise Owner, and Visitors. The Premise Owner, as the main user, can log in or register, reset passwords, control door locks, switch between Sentry and Normal modes, adjust lighting, and review activity logs. Additional features include viewing the chatroom, adding notes, and setting brightness or light modes. The Smart Door performs key tasks like motion detection, facial recognition, data storage, and activity logging to ensure continuous monitoring. Visitors interact by pressing the doorbell or triggering motion sensors, which may activate facial recognition or voice message capture for secure, automated access.

Appendix B presents the flowchart of the VisionDoor: IoT-Integrated Built-In Doorbell and Monitoring System, outlining the system's operation in both Normal and Sentry modes. The process begins with system initialization in Default Mode (Normal), where it passively waits for user interaction. When switched to Sentry Mode, the system actively monitors for motion, detects faces, and performs facial recognition. Recognized users are granted access, while unrecognized faces trigger denial to enter and the system continues to detect known faces. In Normal Mode, the system waits for the doorbell press. Once triggered, the system evaluates the visitor's face and either grants access or prompts the owner for a decision. All interactions including door status changes are logged and notified, with lighting adjustments applied accordingly. This workflow ensures automated, secure, and flexible access control. Fig. 3 is shown in Appendix B.

Appendix C depicts the class diagram of the VisionDoor system, showcasing its key entities, attributes, and relationships. The diagram highlights how components such as smart door, user accounts, and notifications interact to deliver system functionality. The User class stores core user data such as ID, name, email, and password, with methods for registration, login, and password reset. The MobileApp class functions as the user interface, enabling users to receive notifications, adjust lighting modes, trigger facial capture, and access activity logs. The SmartDoorLock class controls the physical locking mechanism, offering lock, unlock, and status-check functionalities. The AccessControl class manages access permissions, motion detection, facial recognition, and logging of access attempts. Additionally, the LightControl class handles entryway lighting based on brightness, night mode, and power status. Together, these classes define the system architecture, enabling modular development and effective integration of IoT, real-time monitoring, and cloud-connected features.

4.3 System Design

This section outlines the system design for the VisionDoor system, focusing on its architecture, class diagram, and interface design.

4.3.1 System architecture

Appendix D illustrates the architecture of the VisionDoor system, which integrates hardware, cloud services, and a mobile application for smart security. A Raspberry Pi 3B+ controls key components including the doorbell, PIR sensor, camera, LCD, LED, and electromagnetic lock. When motion is detected or the doorbell is pressed, an image is captured and stored locally, then processed using OpenCV for facial recognition. Event data is synced via WiFi to Firebase for real-time notifications, authentication, and cloud storage. The Flutter-based mobile app allows users to remotely monitor, receive alerts, and manage access, ensuring seamless interaction between devices, cloud, and users.

The schema table defines the structure of the VisionDoor system's database by outlining the main collections and their attributes. It ensures the collections maintain an organized data structure and establishes clear relationships between the key components of the system. The following collections have been designed and extracted from the class diagram.

i. user (UID, email, address, dateJoined, fullName, isNewUser, mobile, nickname)
ii. alertMode(mode)
iii. doorLockUnlockHistory (activationDate, isLocked)
iv. doorMode (lockState)
v. LightActivation (state, bightness, isNightMode, isSwitchedOn)
vi. logs (event, status, timestamp)
vii. facialCapture(imageID, time, label)

5. Results and Discussion

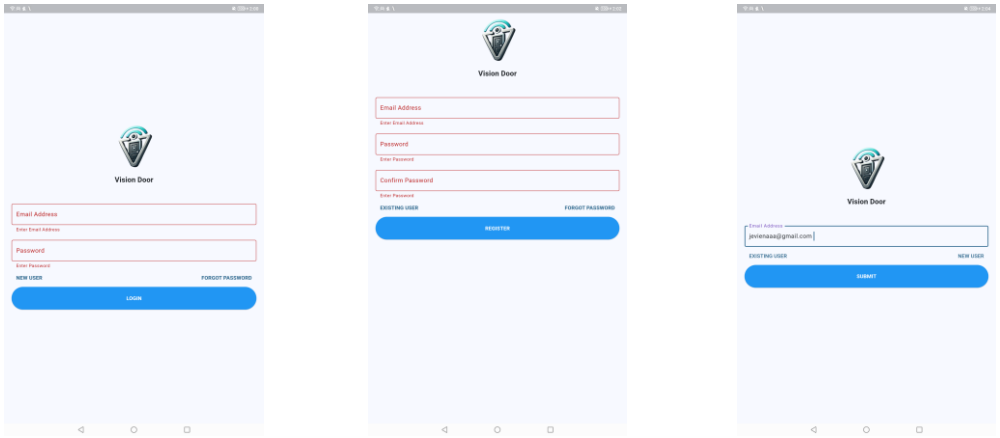
The VisionDoor Smart Security System was developed as an IoT-based solution combining facial recognition, motion detection, and intelligent door control. The back-end processing is handled by Python and Firebase, while the cross-platform mobile application was built using Flutter and Dart. Integration with Firebase enables real-time notifications, authentication, and cloud-based data logging, offering a complete ecosystem for smart access management. Development and testing were carried out using Geany, Android Studio, and Firebase Console, ensuring seamless communication between hardware and software modules. VisionDoor consists of critical modules such as User Registration & Login, Facial Recognition, Door & Light Control, Notification & History Logs, and Dashboard Management. These modules were designed to enhance home and small business security, automate entry processes, and provide remote management capabilities.

5.1 Interface Design

Fig.s 2(a) to 2(c) illustrate the VisionDoor IoT-Integrated Built-In Doorbell and Monitoring System, showcasing a mobile application with the essential screens, each designed to provide distinct functionalities.

Fig. 2(a) and 2(b) shows The Login and Registration Interface that allows users to securely access the system through Firebase Authentication. The login screen includes input fields for email and password, with additional options for password recovery, while the registration page enables new users to sign up by entering their email and password. This ensures secure access by requiring users to input their email and password, with a prominently displayed "Log in" button for straightforward navigation. Supporting links for "New User" and "Forgot Password" are strategically placed for user convenience. Fig. 2(c) shows the password reset interface that sends a secure link via email, allowing users to update their credentials without administrative assistance. Once logged in, users are directed to the main dashboard shown in Fig. 3 . The clean and minimal design focuses on essential login elements without unnecessary clutter.

Applied Information Technology and Computer Science Vol. 6 No. 2 (2025) p. 1644-1663



(a) **(b)** **(c)**

Fig.2 *VisionDoor Interface (a) Login; (b) New user registration; (c) Forgot password*

Fig. 3 shows the the main dashboard serves as the central hub for system management, organized in a structured grid layout. It includes functionalities such as Door Control, Mode Selection, a real-time camera View, Light Control, Chatroom, and Logs. The top panel displays user information such as email and status, while the center panel attempts to load a live video stream from the Raspberry Pi. This centralization enhances overall usability and ensures that all core functionalities are accessible from a single screen.

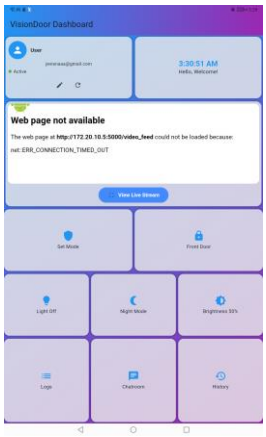
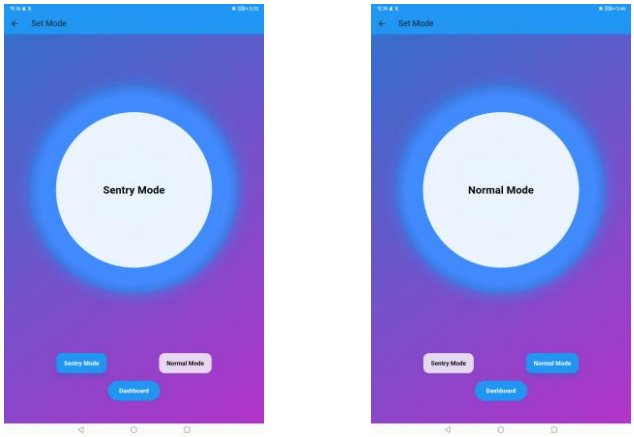


Fig. 3 VisionDoor Dashboard

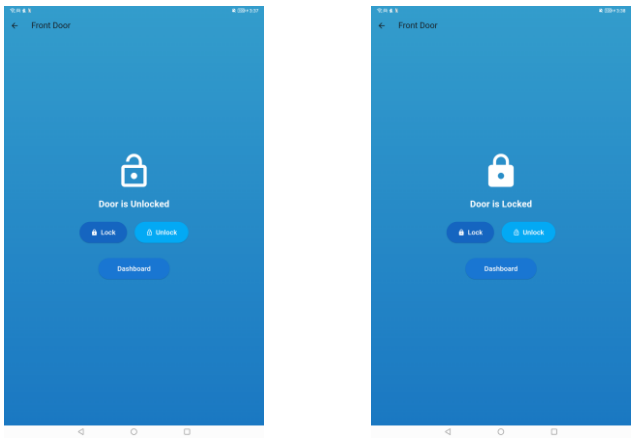
Fig. 4(a) shows the user profile update interface. Fig. 4(b) shows the mode selection interface allows users to switch between "Sentry Mode" and "Normal Mode" using a large circular toggle button. In Sentry Mode, the system actively performs motion detection and facial recognition, while in Normal Mode, these features remain idle unless manually triggered.



(a) **(b)**

Fig. 4 *VisionDoor Interface (a) Sentry mode; (b) Normal mode*

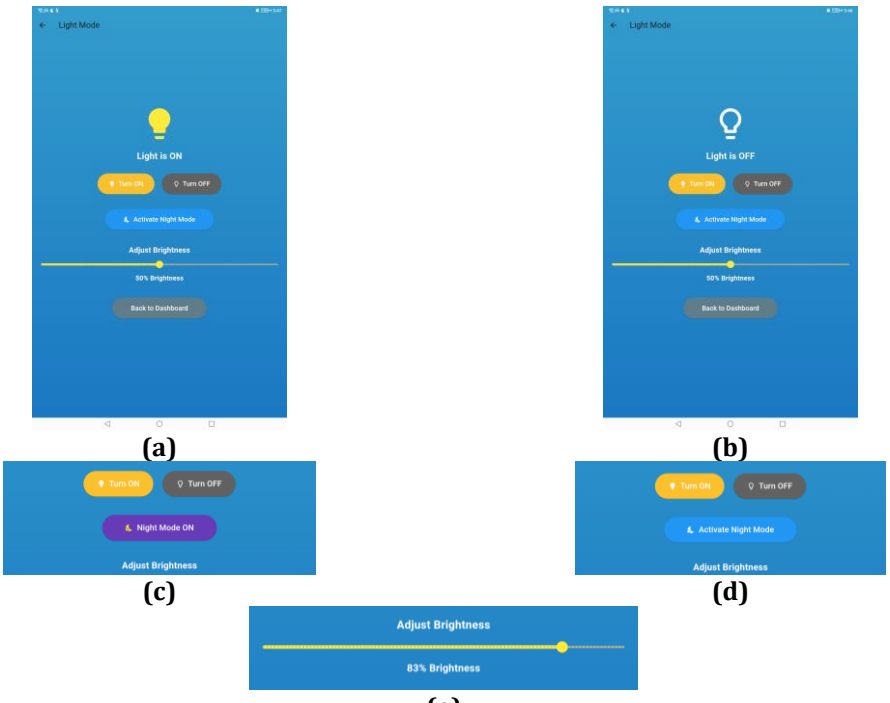
Fig. 5(a) and 5(b) presents the door control allowing users to remotely lock or unlock their front door through the mobile application. The current status of the door is prominently displayed with large lock/unlock icons, supported by dedicated buttons labeled “Lock” and “Unlock.” When a user initiates a command, the interface communicates with Firebase in real-time to trigger the appropriate GPIO pin on the Raspberry Pi, controlling the electronic lock mechanism. This provides secure and convenient access management, especially when the user is not physically present.



(a) **(b)**

Fig. 5 *VisionDoor Interface (a) Door Unlocked; (b) Door Locked*

Fig. 6(a) to 6(e) show the Light Control interface, where users can turn lights ON or OFF, activate Night Mode, or adjust brightness using a slider. The interface reflects the real-time state of the light through icon changes and dynamic feedback. Fig. 6(a) shows the light turned ON, while Fig. 6(b) shows it OFF. Fig. 6(c) and (d) demonstrate Night Mode ON and OFF, providing dim lighting for low visibility conditions. Fig. 6(e) illustrates the brightness slider, allowing users to fine-tune light intensity in real time. This control enhances both the system’s functionality and the user’s comfort.

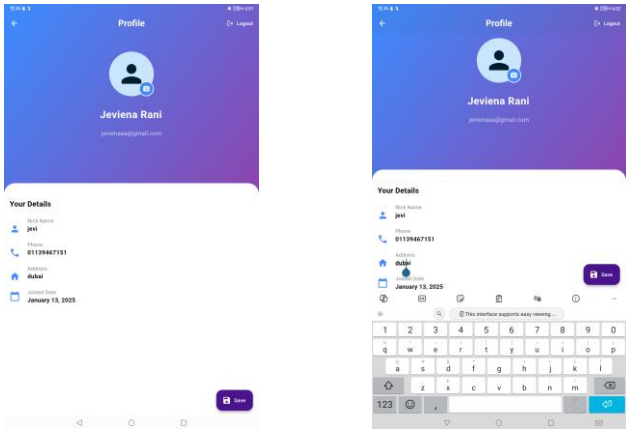


(a) **(b)** **(c)** **(d)** **(e)**

Fig. 6 *VisionDoor Interface (a) Light “On”; (b) Light “Off”; (c) Night mode “On”; (d) Night mode “Off”; (e) Adjust brightness*

Fig. 7(a) and 7(b) display the User Profile module, which shows personal details such as name, nickname, email, phone number, and address and the interface to edit the details respectively. The information is organized in a clean layout for easy viewing. Users can update their profile through the Edit Profile screen, where input fields

become editable. Once changes are confirmed, the data is updated in Firebase, ensuring that user records remain current and personalized.



(a) (b)

Fig. 7 *VisionDoor Interface (a) User profile; (b) Edit profile*

Fig. 8 demonstrates the History Log module, where a chronological list of door access events such locked or unlocked is displayed. Each log entry includes a timestamp, and events are visually differentiated using color-coded icons: green for “Unlocked” and red for “Locked.” This feature promotes transparency, security tracking, and accountability.input fields become editable. Once changes are confirmed, the data is updated in Firebase, ensuring that user records remain current and personalized.

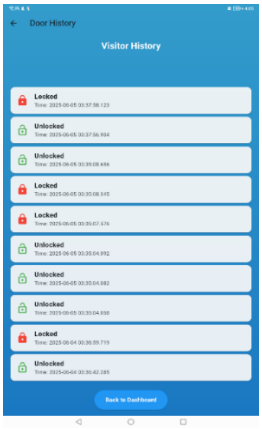
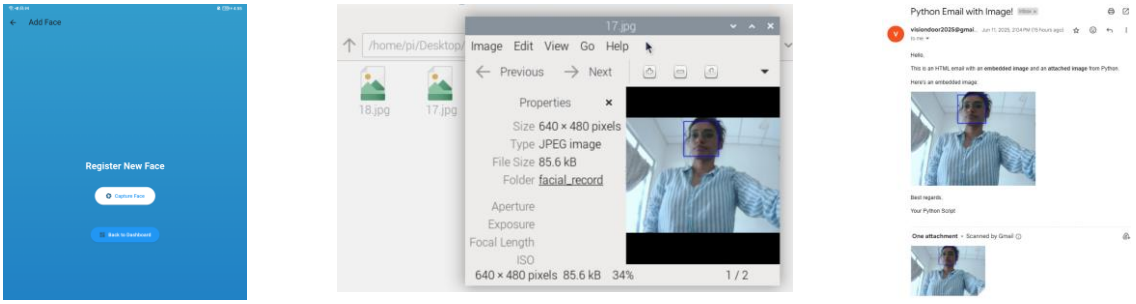


Fig. 8 *VisionDoor Interface*

Fig. 9(a) shows add face interface in flutter application, where users can register their face to the system for future recognition. This interface includes a "Capture Face" button that triggeres the camera to collect facial images and to store it locally in raspberry pi. Fig. 9(b) shows the image is then labelled with an incrementing ID and locally stored in a folder. Fig. 9(c) shows the captured image is embedded to an email sent to owners mailbox via gmail SMTP.

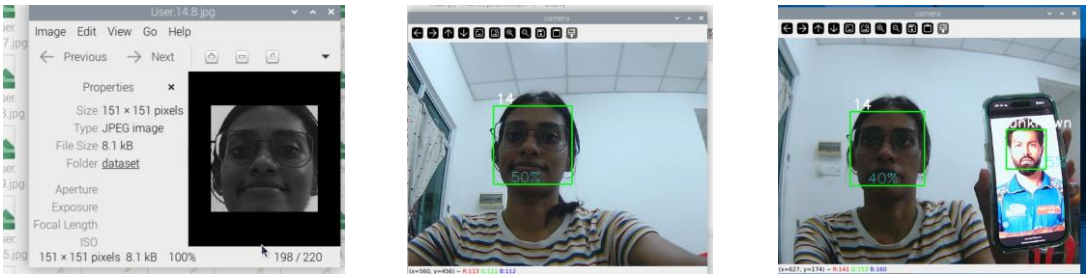


(a) (b) (c)

Fig. 9 *Facial capture (a) Add face interface; (b) Image label; (c) SMTP email*

Fig. 10(a) and 10(b) how the Facial Recognition module in action. When triggered by motion in Sentry Mode, the system captures a face image using the Pi Camera and compares it to stored training data. If a match is found, the recognized user's name and confidence score are displayed. Unknown users are flagged, and their images are sent to Firebase Storage. The interface shows bounding boxes, name labels, and percentages for real-time recognition feedback. It also distinguishes multiple faces, handling both recognized and unknown individuals simultaneously.

Fig. 10(a) shows the face capture dataset, where multiple facial images are collected for training using the Pi Camera. Fig. 10(b) presents a recognized face, with the system successfully identifying the user and displaying their ID with a confidence score. Fig. 10(c) shows an unrecognized face, where the system labels the individual as "Unknown" and sends an alert to the app.



(a) **(b)** **(c)**

Fig. 10 *VisionDoor Facial recognition (a) Face capture dataset; (b) Recognised face; (c) Unrecognised face*

Fig. 11 shows the facial recognition code in python using OpenCV's LBPH algorithm.

```
# Load recognizer
recognizer = cv2.face.LBPHFaceRecognizer_create()

# If no model found, train once
if not os.path.exists("trainer/trainer.yml"):
    print("Model not found. Training now...")
    trainer = FacialTrainer()
    trainer.TrainDataSets()

recognizer.read("trainer/trainer.yml")

# Load Haar Cascade
face_cascade = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")

# Confidence threshold (lower confidence = better match)
CONFIDENCE_THRESHOLD = 50

# Start camera
cam = cv2.VideoCapture(0)

while True:
    ret, img = cam.read()
    if not ret:
        break

    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.2, minNeighbors=5)

    for (x, y, w, h) in faces:
        roi_gray = gray[y:y+h, x:x+w]
        label_id, confidence = recognizer.predict(roi_gray)

        match_percent = round(100 - confidence)
        if confidence < CONFIDENCE_THRESHOLD:
            label_display = str(label_id)
        else:
            label_display = "Unknown"

        # Draw label and confidence
        cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)
        cv2.putText(img, f"{label_display}", (x+5, y-10), cv2.FONT_HERSHEY_SIMPLEX, 1, (255,255,255), 2)
        cv2.putText(img, f"{match_percent}%", (x+5, y+h+25), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0,255,255), 2)

    cv2.imshow("Face Recognition", img)

    # Exit on pressing 'q'
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

Fig. 11 *Facial recognition python code*

This algorithm loads a pre-trained model (trainer.yml) and detects faces from camera frames using a Haar cascade classifier.

5.2 Functional Testing

Functional testing focused on the core components of the system, such as facial recognition, door control, lighting, login, and mode switching. Each feature was tested independently and in integration to ensure real-time communication with the Firebase backend and proper hardware response from the Raspberry Pi. The results of these are presented in detailed tables.

Table 6 *Functional Test Cases for Login and Registration Module*

Test Case	Description	Expected Results	Actual	Results
1	Register user with email and password.	Account created and redirected to dashboard.	User successfully registered.	Pass
2	Login with correct credentials.	Login successful.	User successfully logged in.	Pass
3	Login redirects to dashboard.	Dashboard appears with user info.	Redirected correctly.	Pass
4	Reset password via email.	Password reset email sent and applied.	Reset link received, password changed.	Pass

Applied Information Technology and Computer Science Vol. 6 No. 2 (2025) p. 1644-1663

Table 6: (cont.)

Test Case	Description	Expected Results	Actual	Results
5	Login with invalid credentials.	Access denied.	Login restricted.	Pass
6	Auto-login persistence (session management).	User stays logged in unless manually logged out.	Session persisted as expected.	Pass
7	Register user with already used email.	Error message shown that email is already registered.	Duplicate email blocked	Pass
8	Register user with invalid email format	Error message shown: "Enter a valid email"	Proper validation triggered	Pass
9	Password mismatch when registering new account	Error message shown to enter correct password to match	Password match confirmation	Pass
10	Logout from dashboard	User returned to login screen	Logout successful	Pass

Table 6 outlines the login and registration process handle account creation, credential validation, and session state properly under various input conditions

Table 7 *Functional Test Cases for Dashboard Module*

Test Case	Description	Expected Results	Actual	Results
1	View dashboard after login	User profile and system status shown	Dashboard loaded correctly	Pass
2	Dashboard shows live camera feed in normal mode	Real-time feed and system status visible	Live feed and status displayed	Pass

Table 7 outlines the that the dashboard successfully loads relevant data and live feeds immediately after user authentication.

Table 8 *Functional Test Cases for Set Mode Module*

Test Case	Description	Expected Results	Actual	Results
1	Switch to Sentry Mode	System enters active surveillance mode	Sentry Mode enabled	Pass
2	Switch to Normal Mode	System becomes idle until triggered	Normal Mode set	Pass
3	System is in default mode (Normal)	System automatically changes to normal mode by default	System in normal mode by default after every session	Pass

Table 8 outlines the mode toggling behaves as designed, including fallback to normal mode by default.

Table 9 *Functional Test Cases for Door Control Module*

Test Case	Description	Expected Results	Actual	Results
1	Unlock door remotely	Door unlocks, notification sent	Door unlocks triggered and notified	Pass
2	Lock door via app	Door locks and status updates	Door locks and status updated	Pass

Table 9 outlines the core functionalities of door locking and unlocking respond to app controls and notify users appropriately.

Table 10 *Functional Test Cases for Light Control Module*

Test Case	Description	Expected Results	Actual	Results
1	Turn light ON	Light is turned on and reflected in UI	Light "ON" confirmed	Pass
2	Turn light OFF	Light is turned off	Light "OFF" confirmed	Pass
3	Enable Night Mode	Dim lighting activated	Night Mode enables	Pass
4	Adjust brightness	LED brightness changes	Brightness updated	Pass

Table 10 outlines the that all light control functions work correctly through the mobile interface.

Table 11 *Functional Test Cases for User Profile Module*

Test Case	Description	Expected Results	Actual	Results
1	View user profile	Displays user info (email, phone, address)	Profile displayed	Pass
2	Edit user profile	Updated info saved	Data updated successfully	Pass

Table 11 outlines the user profile data can be viewed and edited with accurate updates.

Table 12 *Functional Test Cases for Door History Log Module*

Test Case	Description	Expected Results	Actual	Results
1	View door lock/unlock logs	Chronological events with timestamps shown	Logs displayed	Pass
2	View logs with color-coded indicators	"Locked" in red, "Unlocked" in green	Color - coded icons visible	Pass

Table 12 outlines the indicates the log system reliably stores and displays historical records with visual cues.

Table 13 *Functional Test Cases for Facial Recognition Module*

Test Case	Description	Expected Results	Actual	Results
1	Trigger face capture from app	Firestore updated, RPi begins image capture	Image captured and stored in RPi	Pass
2	Detect face in real-time	Face detected and ID recognized	Recognition successful	Pass
3	Unregistered face in sentry mode	Motion is constantly detected until recognised face is detected	Recognition in loop until recognised faces is detected	Pass
4	Facial data saved with label	Images saved with correct IDs	Stored correctly in dataset	Pass
5	Train dataset	LBPH model file generated with no error	Model trained successfully	Pass
6	Face recognized from dataset	User ID and confidence score returned	Match found, and door unlocked	Partial Pass
7	Face not recognized from dataset	Labeled as "Unknown", access denied	Door stays locked	Pass
8	New registered face and email sent	Email sent with image ID	Email with embedded image is sent	Pass
9	Recognition in low light	Face should be detected	Recognition failed occasionally	Pass
10	Trigger face capture from app	Firestore updated, RPi begins image capture	Image captured and stored in RPi	Fail

Table 13 outlines the shows that facial recognition features function as expected in normal conditions. However, low-light environments reduce reliability as indicated.

Table 14 *Functional Test Cases for Notification Module*

Test Case	Description	Expected Results	Actual	Results
1	Send real-time notification	Immediate alert via Firebase	Notification sent	Pass
2	Log access attempts	Log created in Firebase	Log saved and retrievable	Pass

Table 14 validates that lock/unlock logs and related alerts are functional, though delays may occur under weak internet conditions.

5.3 Integration Testing

Integration testing was conducted to ensure that all functional modules of the VisionDoor system such as facial recognition, door control, lighting, login, and mode switching work seamlessly together in a real-world environment.

Table 15 *Integration Testing for each functional modules.*

Test Case	Description	Expected Results	Actual	Results
1	User logs in, dashboard loads user info and live feed.	User authenticated, dashboard shows user info and real-time feed.	Dashboard displayed with correct user info and camera feed.	Pass
2	User sets mode to "Sentry" from app, RPi activates motion and recognition.	Mode changed in Firestore, RPi activates motion detection and facial module.	Firebase updated, RPi detects motion, recognition started.	Pass
3	Motion triggers camera, face captured and compared with dataset	Camera captures image, compares with trained model, access granted or denied	Face captured, matched, door unlocked, and notification sent	Pass
4	User presses lock button, RPi activates door lock relay	Firestore updated, relay pin activated to lock door	Door relay triggered via GPIO, lock status confirmed	Pass
5	User turns ON night light via app	Firestore updates, RGB LED dims to night mode level	Night mode activated and brightness adjusted via PWM	Pass
6	User updates profile, dashboard reflects changes on reload	Changes saved to Firestore and updated UI elements on next load	New profile info correctly saved and displayed	Pass
7	Unrecognized face in Sentry Mode triggers alert + log	Alert sent to app, access denied, log created in Firestore	App received alert, "Unknown" tag displayed, event logged	Pass
8	User captures new face via app and receives confirmation email	Capture flag set in Firestore, RPi stores images and sends email	Face captured, image stored, email with attachment sent	Pass
9	Brightness slider adjusted from app	Brightness level updated and applied on RGB LED in real-time	Slider input reflected instantly in light brightness	Pass
10	Door unlocked remotely, log entry created	Door unlocks and timestamped log entry saved	Lock status changed and event visible in history logs	Pass

Table 15 validates that integration tests confirmed successful communication between the mobile application, Firebase backend, and Raspberry Pi hardware.

5.4 User Acceptance Testing (UAT)

UAT was conducted to determine how real users interacted with the VisionDoor system and whether it fulfilled their practical needs. This test focuses on real-world usability, feature satisfaction, and overall system performance as experienced by actual users to evaluate VisionDoor system meets stakeholders' expectations and functional requirements. A total 6 staffs from GTE Solutions Sdn Bhd/ First CDK Sdn Bhd participated, to test the full VisionDoor system, including its mobile application and hardware functions, and subsequently provided structured feedback via Google Forms. Quantitative data were collected using Likert scales, and qualitative feedback was gathered from open-ended questions in the Google Form as attached in Appendix E. The following sections present the feedback and findings based on structured evaluations.

Overall feedback was highly positive. Most users (66.7%) found the setup very easy, while some (33.3%) noted minor difficulties, suggesting a need for clearer setup guidance. All users rated the app's interface, navigation, and user-friendliness with a perfect score, confirming the system is intuitive even for first-time users. The visual design was also praised, with minor suggestions for layout or contrast improvements. System responsiveness received unanimous top ratings, with users confirming fast reactions to commands like door control and lighting. Core features such as login, dashboard access, and notifications worked smoothly. Some users found face registration slightly challenging, especially in low-light conditions, highlighting an area for visual feedback improvement. Despite this, automation features like door lock, light control, and facial recognition were praised for reliability and integration. Most users (83.3%) said they would recommend VisionDoor, and overall satisfaction was high, confirming the system as a secure, user-friendly solution with only minor refinements needed.

6. Conclusion

The VisionDoor IoT-Integrated Built-In Doorbell and Monitoring System provides an innovative and secure solution for modern home monitoring and access control. By integrating real-time motion detection, facial recognition, remote door management, and cloud-based data syncronization, the system ensures enhanced security and convenience for homeowners. Throughout the development process, VisionDoor successfully met its objectives by combining Python, OpenCV, Firebase, and Flutter into a robust and scalable system. The intuitive mobile application, equipped with features such as lighting control, door logs, and real-time notifications, streamlines user interaction and monitoring. This comprehensive approach effectively addresses the limitations of traditional security systems, offering a smart, user-friendly, and efficient solution tailored to the needs of modern households. The system depends on stable internet connectivity for real-time features, and the facial recognition module based on the LBPH algorithm shows reduced accuracy in low-light conditions or with obstructed faces. Similar facial features among users may also affect recognition confidence. Future improvements will focus on enhancing recognition accuracy using advanced AI models, enabling offline functionality through edge computing to facilitate facial recogntion in various environments. Other potential enhancements include multi-factor authentication, role-based access control, and smart home integration such as Google Home and Alexa, making VisionDoor even more intelligent, adaptable, and secure. Overall, VisionDoor represents a meaningful step toward smarter, more secure living environments.

Acknowledgement

The authors would like to thank the Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia for its support.

Author Contribution

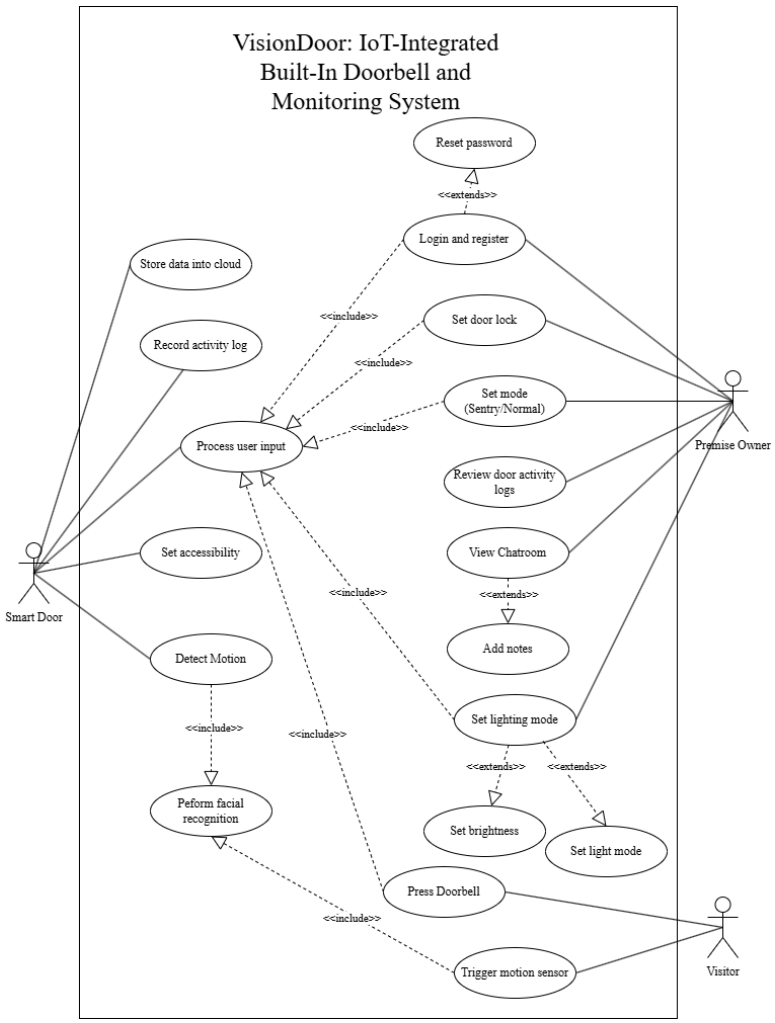
The authors confirm contribution to the paper as follows: **study conception and design:** Jeviena Rani A/P Sanmuga Raja, Azizul Azhar Ramli; **data collection:** Jeviena Rani A/P Sanmuga Raja; **analysis and interpretation of results:** Jeviena Rani A/P Sanmuga Raja, Azizul Azhar Ramli; **draft manuscript preparation:** Jeviena Rani A/P Sanmuga Raja, Azizul Azhar Ramli. All authors reviewed the results and approved the final version of the manuscript.

References

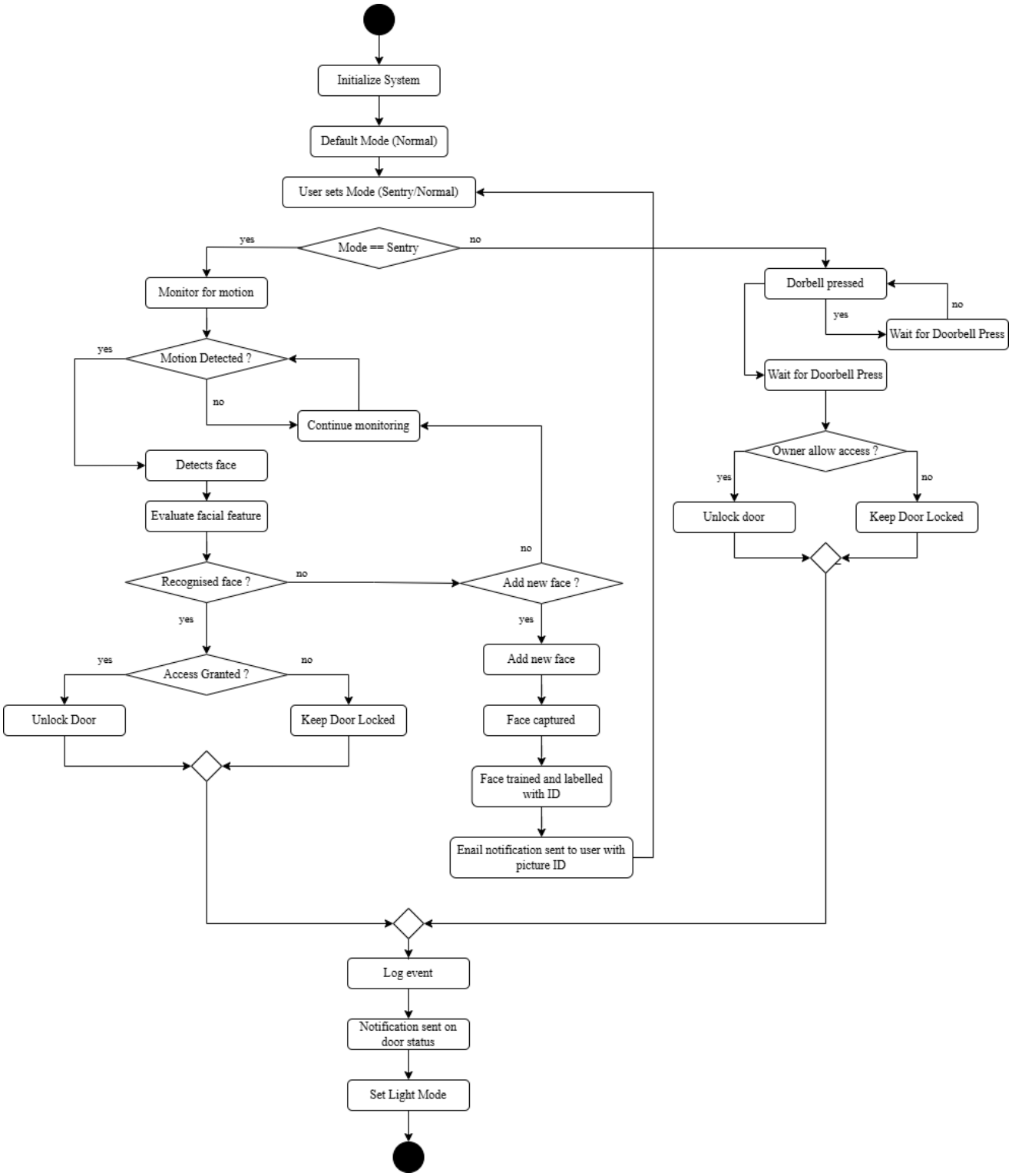
[1] Gupta, A. K., and Johari, R., *IoT Based Electrical Device Surveillance and Control System*. IEEE, 2019. doi: 10.1109/iot-siu.2019.8777342.

[2] Patel, J., Anand, S., and Luthra, R., "Image-Based Smart Surveillance and Remote Door Lock Switching System for Homes," in *Procedia Computer Science*, Elsevier B.V., 2019, pp. 624–630. doi: 10.1016/j.procs.2020.01.056.
[3] Chaudhari, U., Gilbile, S., Bhosale, G., Chavan, N., and Wakhare, P., "Smart Doorbell Security System Using IoT," *EasyChair Preprint*, 2020.
[4] Upadhyay, J., Rida, P., Gupta, S., Siddique, N., and Professor, A., "Smart Doorbell System Based on Face Recognition," 2017. [Online]. Available: www.irjet.net.
[5] Al-Kuwari, M., et al., "Smart-Home Automation Using IoT-Based Sensing and Monitoring Platform."
[6] Wolniak, R., and Grebski, W., "The usage of smart doorbells in smart home," *Scientific Papers of Silesian University of Technology: Organization and Management Series*, vol. 2023, no. 190, pp. 237–247, 2023. doi: 10.29119/1641-3466.2023.190.16.
[7] Baron Sam, B., Purna Chander, K., Vinay, K., and Pio Sajin, A., "Doorbell System in Home Using IoT," in *IOP Conference Series: Materials Science and Engineering*, Institute of Physics Publishing, Oct. 2019. doi: 10.1088/1757-899X/590/1/012018.
[8] Shah, M., Shukla, D., and Pandya, D., "Smart Gate: Intelligent Security System Based on Face Recognition," 2020.
[9] Lakhwan, M., and Kumari, S., "Smart Door Bell System," *International Journal of Science and Research (IJSR)*, vol. 12, no. 4, pp. 1447–1450, Apr. 2023. doi: 10.21275/sr23422204050.
[10] Khrais, L. T., and Alghamdi, A. M., "The role of mobile application acceptance in shaping e-customer service," *Future Internet*, vol. 13, no. 3, 2021. doi: 10.3390/fi13030077.
[11] Jain, A., Lalwani, S., Jain, S., and Karandikar, V., "IoT-Based Smart Doorbell Using Raspberry Pi," in *Advances in Intelligent Systems and Computing*, Springer Verlag, 2019, pp. 175–181. doi: 10.1007/978-981-13-2673-8_20.

Appendix A: Use Case Diagram

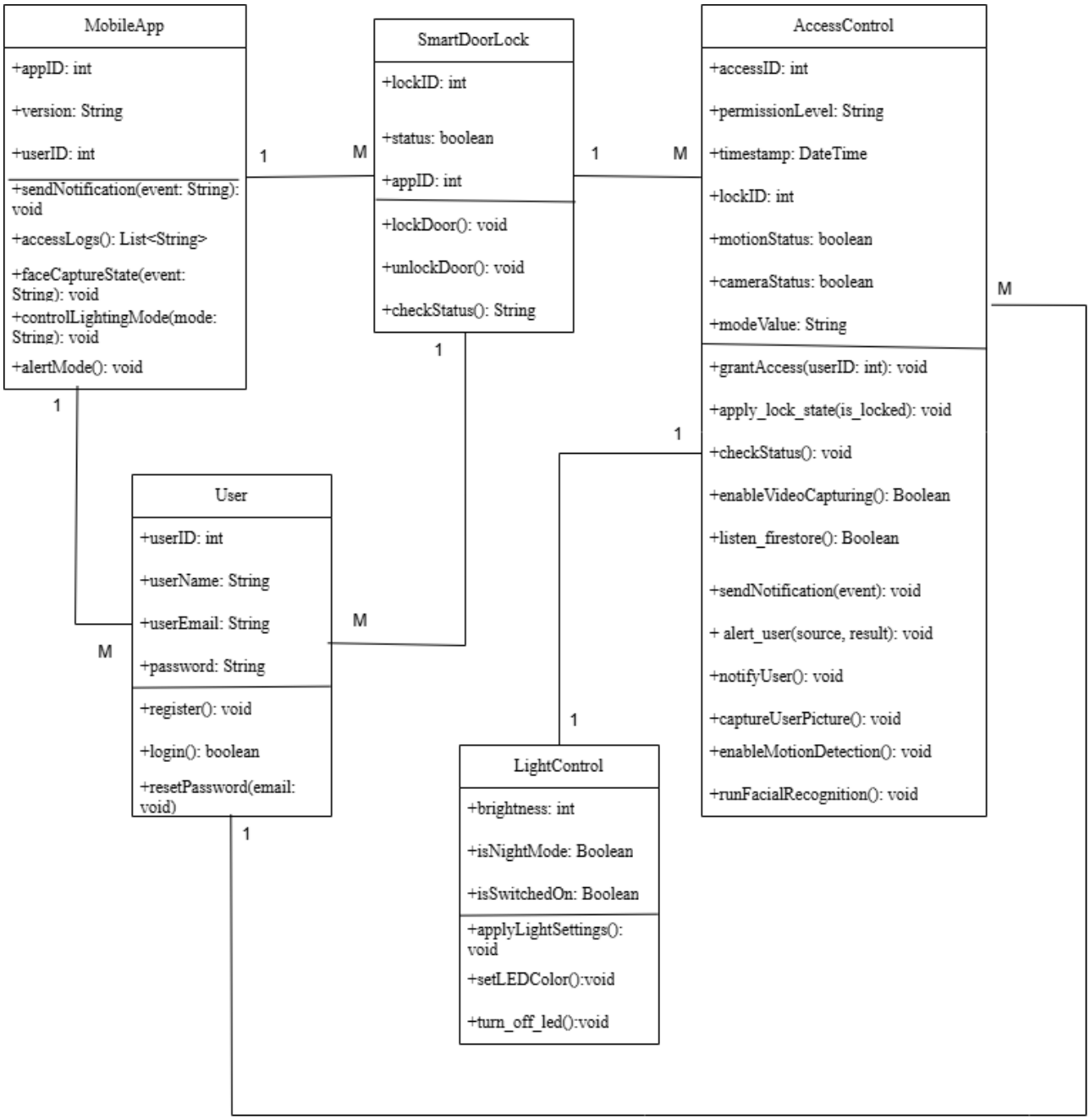


Appendix B: Flowchart

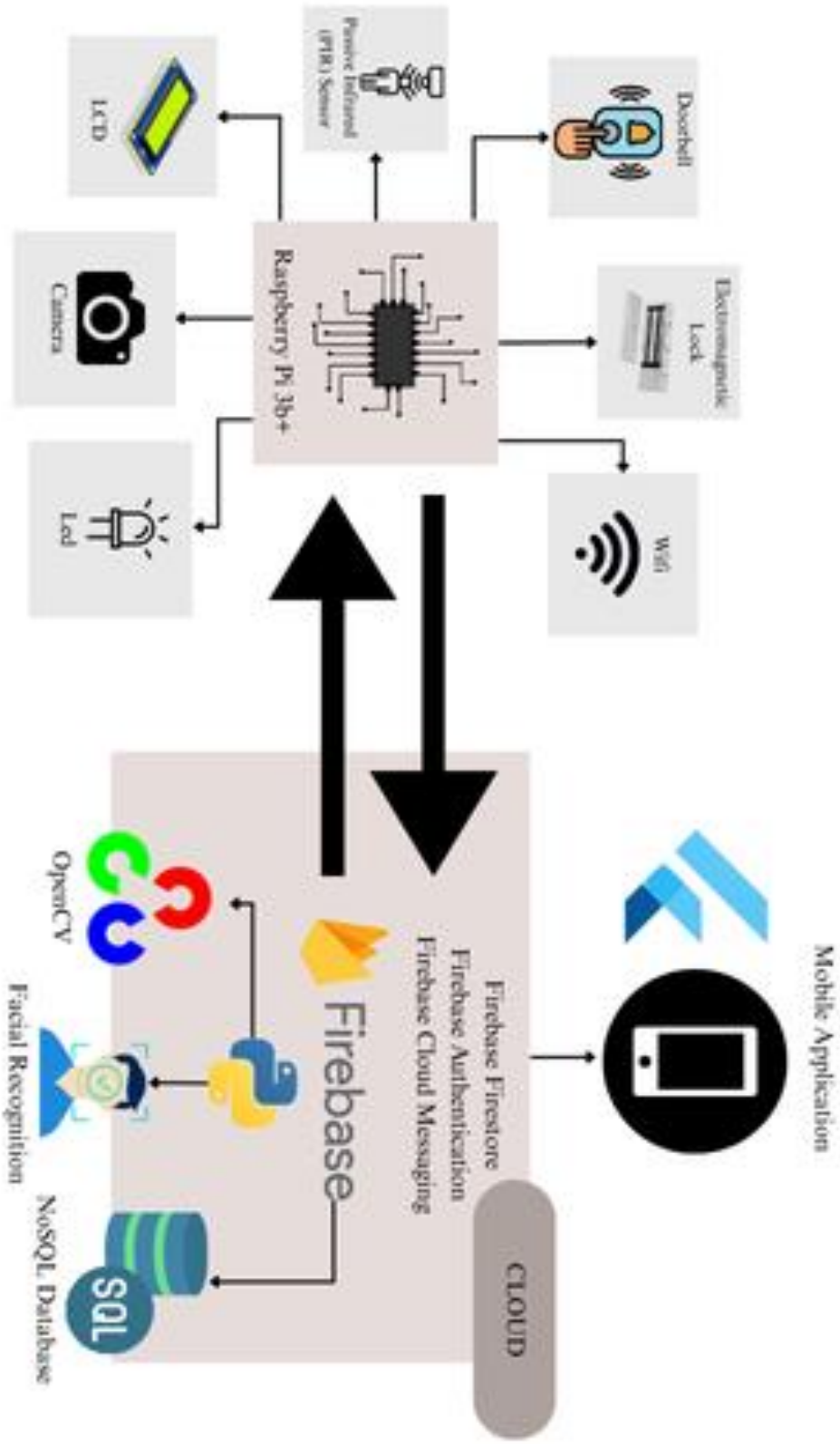


Applied Information Technology and Computer Science Vol. 6 No. 2 (2025) p. 1644-1663

Appendix C: UML Class Diagram



Appendix D: System Architecture

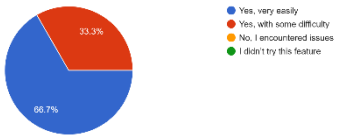


Applied Information Technology and Computer Science Vol. 6 No. 2 (2025) p. 1644-1663

Appendix E: UAT Feedback Charts

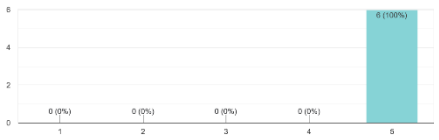
Were you able to set up the VisionDoor system and configure the mobile app?

6 responses



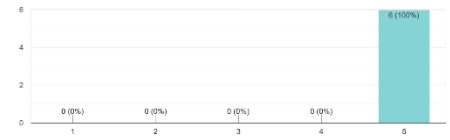
On a scale of 1 to 5, how user-friendly do you find the VisionDoor application ?

6 responses



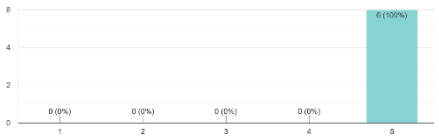
How easy was it to navigate the VisionDoor mobile app interface?

6 responses



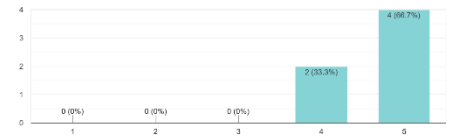
How intuitive was the system navigation for first-time users?

6 responses



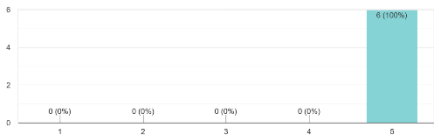
Is the visual appearance of the interface clean, uncluttered, and professional?

6 responses



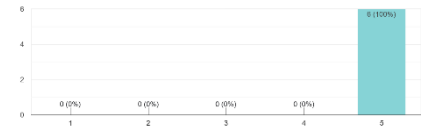
Did you clearly understand how to use each feature (e.g., set modes, control door, view live feed)?

6 responses

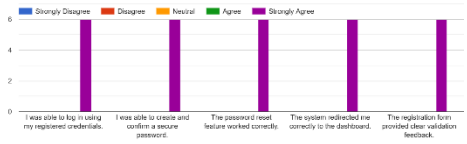


How fast did the system respond to your actions (e.g., unlocking door, switching modes, receiving alerts)?

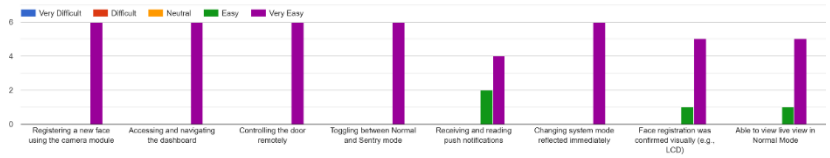
6 responses



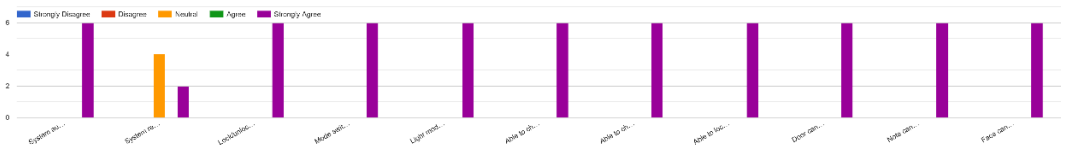
Please indicate your level of agreement with the following statements regarding the Login and Registration features of the VisionDoor system:



Please rate your experience performing the following tasks in VisionDoor system modules:



Please indicate your level of agreement with the following statements regarding system automation and integration:



What did you like most about the VisionDoor system?

6 responses

I loved how the app integrates both door and lighting control in one interface. Smooth switching between them.

The dual mode. Sentry mode is great as the system monitors for unknown faces.

Easy to log in and use. No unnecessary steps.

Real-time face recognition and automatic logging impressed me the most

I really liked how it sends real-time notifications when someone unlocks the door—great for security

The facial recognition feature was impressive and worked quickly in real-time. The dashboard was clean and user-friendly.

What features or functions did you find confusing or difficult to use?

6 responses

The profile edit and logout feature was a bit hidden. Took some time to locate.

Recognising face took awhile. Maybe because of bad lighting.

The notification did not respond immediately due to weak internet connection. Took awhile to recognise face

Sometimes it's hard to tell which mode the system is in—maybe add clearer labels

Face was taking longer to be recognised

The face adding as there was no display of my face being captured

Are there any features you would like to see added in future updates?

6 responses

Maybe a timeline view or graph of access events over time

Able to detect faces properly in dark lightning or different environment

A way to view captured faces in the history logs would be cool

To be able to add face through app itself.

I' like to see dark mode themse added to the app UI for better readability at night

Additional comments or suggestions:

6 responses

Smart design

Very impressive system. Could be useful in offices or shared homes too.

Keep it lightweight. Work better on facial recognition. Interface very plain, can add more elements

Should work without wifi connection. Overall impressive for a degree IT project.

Overall, it feels modern and secure. Good job!

It's convenient and easy to manage even for someone who isn't tech-savvy