Biometric Based Equipment Access and Monitoring System

Piyush Mallappa Munje[1], Swapnil Rajaram Koli[2], Zaheer Abdul Mujawar[3]
[1,2,3]*Department of Computer Science and Engineering, Sharad Institute of Technology College of Engineering, Yadrav (Ichalkaranji), Maharashtra, India*

***Abstract*—This paper presents a Biometric-Based Equipment Access System designed to provide secure and authorized control of equipment using fingerprint authentication. The system uses an ESP32 microcontroller to verify registered users and control equipment through a relay mechanism. It also maintains access logs and provides web monitoring through an admin dashboard for managing users and equipment permissions. The proposed system helps prevent unauthorized equipment usage, improves accountability, and provides an efficient approach to equipment access management in laboratories, workshops, and industrial environments.**

***Index Terms*—Biometric Authentication, Equipment Access Control, ESP32, Fingerprint Recognition, Internet of Things (IoT), Security System.**

I. INTRODUCTION

Security and proper management of laboratory equipment are important aspects of educational institutions and technical organizations. Laboratories contain various electronic devices, instruments, and other valuable resources that require controlled access. Traditional equipment management systems generally depend on physical keys or PIN-based locking mechanisms. However, these methods may create problems such as misplaced keys, shared passwords, unauthorized access, and difficulty in maintaining records of equipment usage. These limitations highlight the need for a more secure and systematic equipment access management solution.

The Biometric-Based Equipment Access System is developed to overcome these challenges by introducing fingerprint-based authentication for equipment access. The proposed system uses an ESP32 microcontroller integrated with an R307 optical fingerprint sensor to identify registered users. After successful fingerprint verification and authorization, the system activates a relay-controlled solenoid lock to provide access to the equipment locker. This approach helps ensure that equipment can only be accessed by users who have been registered and granted the required permissions.

The system also includes a web-based administrative dashboard developed using React and Node.js to simplify equipment and user management. Administrators can register users, manage fingerprint enrollment, assign access permissions, and monitor equipment activities through the dashboard. An access logging mechanism maintains records of access events, helping improve transparency and accountability in laboratory operations. The system also supports communication between the hardware and web application to provide updated equipment status and access information.

Another important feature of the proposed system is its locally hosted architecture, which allows operation through a local area network (LAN) without depending on external cloud services. This design makes the system suitable for laboratory environments where local control, equipment security, and reliable monitoring are important requirements.

The main aim of this project is to develop an efficient and user-friendly solution that combines biometric authentication, electronic locking, and centralized equipment management. By integrating these technologies, the proposed system provides a practical approach to reducing unauthorized equipment access, improving monitoring, and maintaining better control over laboratory resources.

II. LITERATURE REVIEW

Biometric authentication, embedded controllers, and Internet of Things (IoT) technologies have been widely explored for improving access control and security systems. Traditional access methods, such as physical keys and passwords, can create difficulties in environments where equipment is shared among multiple users. Biometric systems provide an alternative by using individual characteristics, such as fingerprints, to verify a person's identity. The following review discusses the technologies and approaches related to the proposed Biometric-Based Equipment Access and Monitoring System.

A. Biometric Authentication for Security Applications

Biometric authentication is an important approach to identity verification because it uses physical or behavioral characteristics of individuals. Fingerprint recognition is commonly used in access control applications because it provides a convenient way to identify registered users. A review by researchers on biometrics for IoT security discusses how biometric techniques can improve authentication in connected devices. It also highlights challenges related to biometric data protection, privacy, and security when biometric systems are integrated with IoT environments [1].

B. IoT-Based Access Control Systems

The integration of IoT technology with access control systems enables connected devices to exchange information and support remote monitoring. Trabelsi et al. reviewed access control approaches for IoT and discussed centralized and distributed architectures, along with security challenges associated with connected devices [2]. Similarly, Trnka et al. examined authentication and authorization techniques used in IoT applications and highlighted the importance of identity management and access policies [3]. These studies indicate that an effective access control system must not only identify users but also determine which resources they are permitted to access.

C. Fingerprint-Based Smart Lock Systems

Several research projects have explored fingerprint-based electronic locking systems using microcontrollers and relay-controlled locks. A 2025 IEEE conference paper describes an IoT-based security system that uses an ESP32, fingerprint sensor, keypad, LCD, and solenoid lock. It also includes IoT-based monitoring and reports experimental results for authentication response and accuracy [4]. Another study published in the International Journal of Advanced Computer Science and Applications in 2025 presents an ESP32-based biometric smart lock with fingerprint authentication, mobile monitoring, and alerts following repeated unsuccessful attempts [5]. These systems demonstrate the practical use of biometric verification and electronic locking mechanisms in security applications.

D. Web-Based Monitoring and Access Management

Web-based monitoring can make access control systems easier to manage by providing administrators with information about users, device status, and access activities. IoT access control research also identifies authorization policies and identity management as important parts of a secure connected system [2], [3]. A centralized dashboard can therefore support administrative activities such as user registration, permission management, and access-history review. These functions are particularly relevant in laboratories and institutions where equipment is shared by users with different access permissions.

E. Research Gap

The reviewed studies establish the usefulness of fingerprint authentication, microcontroller-based electronic locks, and IoT-enabled monitoring. However, many of the discussed implementations focus primarily on controlling access to a door or a restricted area. Equipment management in laboratories involves additional requirements, including assigning permissions to different users, maintaining equipment access records, managing user roles, and monitoring equipment or locker status. The proposed system addresses this application need by combining fingerprint-based authentication using the R307 sensor, ESP32-based lock control, and a web dashboard developed using React and Node.js. It provides user and permission management, access logging, and real-time status updates over a local network. Unlike systems that depend on an external

cloud platform, the proposed implementation is designed to operate within a local network. Its objective is to provide an integrated approach to biometric equipment access and locker management in laboratory environments.

III. PROBLEM STATEMENT AND PROPOSED SYSTEM

A. Problem Statement

In educational institutions and laboratory environments, various types of equipment are shared by students, faculty members, and laboratory staff. Managing access to these resources is important for maintaining security and ensuring that equipment is used only by authorized individuals. However, conventional access management methods often depend on physical keys or shared passwords, which can create difficulties in controlling access and identifying the person responsible for using a particular resource.

Physical keys may be misplaced, duplicated, or transferred between users without proper records. Similarly, shared passwords do not always provide reliable individual identification. These methods also offer limited support for maintaining access histories and monitoring equipment status remotely. As a result, laboratory administrators may face difficulties in managing user permissions, tracking access activities, and maintaining accountability.

To address these challenges, a system is required that can verify individual users, control equipment access electronically, and maintain records of access activities. It should also provide administrators with a convenient interface for managing registered users and monitoring system operations. The proposed project focuses on developing such a solution using fingerprint authentication and an ESP32-based electronic locking mechanism.

B. Proposed System

The proposed system, named BioLocker Pro, is an intelligent biometric equipment access and locker management solution developed for laboratory environments. It combines fingerprint-based identity verification, embedded hardware control, and a web-based management interface.

The hardware unit uses an ESP32 microcontroller connected to an R307 fingerprint sensor. During an access attempt, the sensor captures the user's fingerprint and compares it with the templates registered in its internal storage. When a fingerprint is successfully identified and the corresponding user has the required permission, the ESP32 activates a relay module connected to a solenoid lock. The system provides visual and audio feedback through LEDs and a buzzer to indicate whether access has been granted or denied.

A web application developed using React and Node.js provides the administrative interface. It supports user registration, fingerprint enrollment and deletion, permission management, and access-history monitoring. The application also displays system information and receives real-time updates through WebSocket communication. Communication between the hardware and backend is handled through HTTP-based requests.

The system is designed to operate over a local network, allowing the hardware and web application to communicate without depending on an external cloud platform. This approach supports monitoring and management within the institution's network. The current hardware implementation controls a physical lock through a single relay, while the software dashboard is designed with scope for managing multiple locker entries as the system is expanded.

C. Objectives of the Proposed System

The major objectives of the proposed system are:

1. To develop a biometric authentication mechanism that identifies users through fingerprint verification.
2. To control equipment or locker access electronically using an ESP32 microcontroller and a relay-operated solenoid lock.
3. To provide a web-based interface for registering users and managing fingerprint enrollment.
4. To implement permission management so that access can be restricted to authorized users.
5. To maintain access records for reviewing successful and unsuccessful access attempts.
6. To provide real-time system updates and monitoring through a web dashboard.
7. To establish communication between the embedded hardware and the web application over a local network.

8. To develop a modular design that can be extended to support additional lockers and equipment.

D. Scope of the System

The proposed implementation focuses on biometric access management for shared equipment and lockers in educational laboratory environments. It includes fingerprint-based authentication, electronic lock control, user and permission management, access logging, and web-based monitoring.

The current prototype demonstrates the operation of one physical locking mechanism. The software architecture provides a foundation for extending the solution to additional lockers and equipment. Further development may include integration with institutional databases, advanced reporting facilities, and additional hardware units.

IV. SYSTEM ARCHITECTURE AND DESIGN

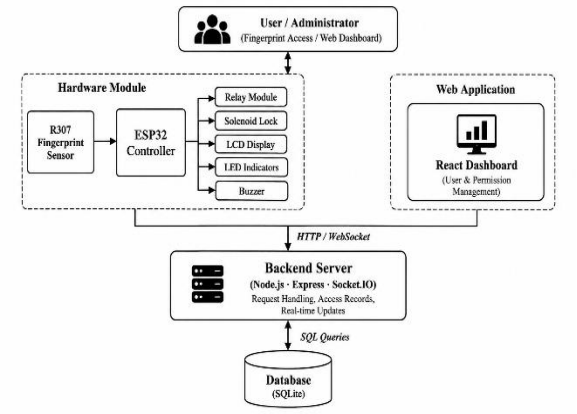


Fig. 1. System Architecture of BioLocker Pro

This section describes the overall architecture of BioLocker Pro and explains how its hardware, backend, and web application work together. The system is organized into three main parts: the biometric and locking hardware, the backend communication layer, and the web-based management interface.

The design follows a modular approach in which fingerprint verification and physical lock control are handled by the ESP32, while user administration and access monitoring are provided through the web application. This separation allows the hardware and software components to perform their respective tasks while exchanging information over the local network.

A. Overall System Architecture

BioLocker Pro follows a client-server architecture connected to an embedded hardware unit. The ESP32 acts as the main controller and manages communication with the fingerprint sensor and electronic locking circuit. The backend receives information from the hardware and makes the relevant data available to the web application.

The architecture consists of the following functional layers:

1. Biometric authentication layer: Captures fingerprint information and verifies it against registered fingerprint templates.
2. Embedded control layer: Processes authentication results and operates the relay, solenoid lock, LEDs, buzzer, and display.
3. Backend communication layer: Handles requests from the ESP32, processes application data, and distributes system events.
4. Web application layer: Provides an interface for administrators to manage users, permissions, locker information, and access records.

The ESP32 is suitable for this role because it provides wireless networking and peripheral interfaces, including UART and GPIO, for connecting external hardware modules [1].

B. Hardware Architecture

The hardware section is responsible for identifying users and controlling the physical locking mechanism. The ESP32 DevKit V1 is used as the central processing unit. It receives fingerprint-related information from the R307 sensor through UART communication and controls the output devices through its GPIO connections.

The R307 fingerprint module captures the fingerprint and performs the matching operation using the templates stored in its internal memory. When a user places a finger on the sensor, the module checks the captured fingerprint against the enrolled templates and provides the matching result to the ESP32.

After receiving the result, the ESP32 applies the access decision based on the available authorization information. If access is permitted, it activates the relay module, which controls the 12 V solenoid lock. The LCD provides status information, while the

LEDs and buzzer communicate the access result to the user.

The major hardware components are:

- ESP32 DevKit V1: Main controller responsible for processing and communication.
- R307 fingerprint sensor: Used for fingerprint enrollment and authentication.
- Single-channel relay module: Acts as an electrically controlled switch for the locking mechanism.
- 12 V solenoid lock: Provides the physical locking and unlocking action.
- 16×2 I2C LCD: Displays system messages and authentication status.
- Green and red LEDs: Indicate successful and unsuccessful access attempts.
- Buzzer: Produces audio indications during system operation.

C. Software Architecture

The software architecture is divided into the frontend and backend components. Both parts work together to provide administrative control and monitoring facilities.

1. Frontend application

The frontend is developed using React. It provides the graphical interface through which administrators can interact with the system. The dashboard supports user-related operations, fingerprint enrollment management, permission configuration, access-history viewing, and locker-status monitoring.

2. Backend application

The backend is developed using Node.js. It acts as the communication and processing layer between the hardware and the frontend. It receives HTTP requests from the ESP32, processes application operations, and provides the required information to the dashboard.

3. Communication mechanism

HTTP REST communication is used for exchanging requests and responses between the hardware and backend. WebSocket communication through Socket.IO is used to deliver real-time events to the dashboard. This allows changes in access activity and system status to be reflected without repeatedly refreshing the web page.

D. Working Flow of the System

The system follows a sequence of operations whenever a person attempts to access a locker:

1. The user places a finger on the R307 fingerprint sensor.
2. The sensor captures the fingerprint and checks it against the enrolled templates.
3. The authentication result is communicated to the ESP32.
4. The controller checks whether the identified user is authorized to access the equipment.
5. If authorization is successful, the ESP32 activates the relay and unlocks the solenoid lock.
6. The green LED and buzzer provide an indication of successful access.
7. If authentication or authorization fails, the locking mechanism remains inactive, and the red LED and buzzer indicate rejection.
8. The access event is communicated to the backend through an HTTP request.
9. The backend processes the event and updates the relevant access record.
10. The dashboard receives the corresponding update through WebSocket communication.

E. System Design Considerations

The architecture is designed around modularity, centralized administration, and local-network communication. The separation of hardware control from web-based management makes it easier to maintain individual components and extend the system in the future.

The use of a local network reduces the system's dependency on external cloud services. However, its monitoring and management features depend on the availability of the local network and backend server. The current prototype demonstrates one physical locking unit, while support for additional lockers would require corresponding hardware connections and configuration.

V. METHODOLOGY AND IMPLEMENTATION

A. Hardware Setup and Controller Initialization

The hardware prototype was developed around the ESP32 DevKit V1, which functions as the main processing and communication unit. The R307 fingerprint sensor is connected to the controller

through a UART interface. The relay module, LCD display, LEDs, and buzzer are connected through the required input/output pins.

When the system is powered on, the ESP32 initializes its serial communication interfaces and configures the connected output devices. The relay, LEDs, and buzzer are initially placed in their inactive states to prevent an unintended locking operation during startup. The LCD is initialized through the I2C interface and displays a startup message.

The fingerprint sensor is then initialized using its serial communication interface. The firmware checks whether the sensor is responding correctly before allowing normal operation. If initialization is unsuccessful, an error message is displayed. Once the sensor is available, the controller proceeds with network initialization.

B. Fingerprint Enrollment Process

Fingerprint enrollment is performed through the web-based management interface. An administrator initiates the registration process by selecting the required user and submitting an enrollment request.

After receiving the request, the backend communicates the enrollment command to the ESP32. The controller starts the fingerprint registration sequence and guides the user through the scanning procedure. The fingerprint is captured in two scanning steps so that the sensor can create a usable fingerprint template.

Following successful registration, the enrollment result is communicated to the backend. The associated fingerprint information is linked with the relevant user and locker details in the application database. The dashboard receives the enrollment status and displays the result to the administrator.

The system also supports fingerprint removal and management through the administrative interface. This allows the administrator to update registered fingerprint information when required.

C. Fingerprint Authentication and Access Verification

The authentication process begins when a user places a finger on the R307 sensor. The sensor captures the fingerprint and compares it with the templates available in its internal memory.

When a matching fingerprint is identified, the sensor provides the corresponding template information to the ESP32. The controller then uses the available user and authorization information to determine whether access should be allowed.

If the identified user has permission to use the associated locker, the access request is accepted. If no matching fingerprint is found or the user does not have the required authorization, the request is rejected.

This process separates identity verification from permission checking. A registered fingerprint establishes the identity associated with a template, while the authorization stage determines whether that user is permitted to access the requested resource.

D. Electronic Lock Control and User Feedback

The electronic locking mechanism is operated through a relay module connected to the ESP32. The relay acts as an interface between the controller's output signal and the solenoid lock's power circuit.

When the access decision is positive, the controller activates the relay, allowing the solenoid lock to operate. The green LED and buzzer provide feedback indicating that access has been granted. When authentication or authorization is unsuccessful, the relay remains inactive, the red LED is activated, and the buzzer produces a rejection indication.

The LCD is used to display messages related to system initialization, fingerprint scanning, and access status. These indicators help users understand the result of their interaction with the device without requiring access to the administrative dashboard.

E. Firmware Development and Network Communication

The ESP32 firmware manages sensor communication, lock operation, display updates, and interaction with the backend. After completing hardware initialization, the controller attempts to establish a Wi-Fi connection.

The firmware includes a limited number of connection attempts. If the network becomes available, the LCD displays the connection information, including the assigned IP address. If the connection cannot be established, the device displays an offline status and continues its local hardware operation.

A local HTTP server is configured on the ESP32 to provide routes for operations such as unlocking, fingerprint enrollment, and fingerprint deletion. The

main firmware loop continuously checks for incoming requests and handles authentication-related operations.

The controller also sends periodic heartbeat messages to the backend. These messages allow the software to determine whether the hardware device is responding. If heartbeat messages are not received within the defined monitoring period, the backend changes the device status to offline and updates connected dashboard clients.

F. Backend Development

The backend was implemented using Node.js with the Express framework. It acts as the central communication layer between the ESP32 hardware and the React-based frontend.

The backend provides REST API endpoints for application operations, including user management, locker management, fingerprint records, inventory information, and access logs. It also handles login requests for administrative users.

Socket.IO is integrated into the backend to support real-time communication with connected dashboard clients. When an important event occurs, such as a locker state change, fingerprint enrollment result, access event, or hardware status update, the backend distributes the corresponding information to the frontend.

A hardware monitoring mechanism uses the heartbeat information received from the ESP32. When the device stops reporting for the configured timeout period, its connection status is updated and the dashboard is notified.

G. Database Design and Management

SQLite is used as the database technology for storing application information. The backend accesses the database through the better-sqlite3 library.

The database is organized into five main tables:

1. Lockers: Stores locker identifiers, names, current status, and information about the materials available in each locker.
2. Users: Contains user details, assigned roles, dashboard login information, and locker assignments.
3. Fingerprints: Maintains fingerprint record identifiers, associated users, locker references, and descriptive labels.
4. AccessLogs: Records access-related events, including the user, locker, action, and event timestamp.
5. Inventory: Stores equipment names, quantities, associated lockers, and stock conditions.

Relationships between these tables help maintain consistency between users, fingerprints, lockers, inventory items, and access records. The database initialization process also includes table creation checks and provisions for preparing initial application records when the database is empty.

H. Frontend Development and Dashboard Implementation

The frontend was developed using React and Vite. It provides a single-page application through which administrators can operate and monitor the system.

The dashboard is divided into functional sections. The overview section presents general system information, while the locker section displays locker states and provides remote control options. The user management section supports creating users, assigning lockers, and viewing fingerprint assignments.

The access-log section displays recorded activities with their corresponding timestamps. The analytics section presents information about access patterns, including activity over time and access distribution. The settings section provides fingerprint enrollment, renaming, and deletion operations.

The frontend uses the Socket.IO client to receive updates from the backend. This allows changes in device status, locker status, enrollment progress, and access activity to be reflected in the interface as events occur.

I. Integration of Hardware and Software

After the individual hardware and software modules were developed, they were integrated to establish complete system operation. The ESP32 communicates with the backend through HTTP requests, while the backend manages application records and sends real-time updates to the frontend.

During a normal access operation, the fingerprint sensor supplies the identification result to the controller. The controller processes the authorization outcome, operates the locking mechanism when permitted, and sends the event information to the

backend. The backend updates the access record and notifies the dashboard.
The same communication structure is used for administrative operations such as fingerprint enrollment and deletion. This integration connects physical access operations with centralized user and equipment management.

J. Implementation Summary
The completed prototype combines fingerprint-based identification, electronic lock control, local-network communication, database-backed administration, and web-based monitoring. The hardware handles the physical authentication and locking operations, whereas the backend and frontend provide administrative functions and access information.
The implementation establishes a modular foundation for laboratory equipment management. Although the current prototype demonstrates one physical locking mechanism, its software organization provides scope for extending the system to additional lockers and equipment in future development.

VI. RESULTS AND DISCUSSION

A. Fingerprint Authentication Results
The developed system demonstrates fingerprint-based access verification using the R307 optical fingerprint sensor and ESP32 controller. During operation, the sensor checks the user's fingerprint against the templates stored in its internal memory. When a registered fingerprint is recognized and the associated user has the required permission, the system proceeds with the unlocking operation.
When the fingerprint is not recognized, the access request is rejected. The system provides a visual indication through the red LED and an audible alert through the buzzer. This process establishes an individual authentication mechanism and avoids relying entirely on shared keys or passwords for physical equipment access.

The project implementation also supports two-stage fingerprint enrollment. During registration, the user scans the same finger twice so that the sensor can generate and store the corresponding template. Enrollment progress and the final outcome are communicated to the dashboard.

B. Electronic Locking Mechanism
The integration of the ESP32, relay module, and solenoid lock successfully establishes an electronic method of controlling physical access. When an authorized access operation is initiated, the controller activates the relay and operates the solenoid locking mechanism. The relay is configured to return to its inactive condition after the specified unlocking interval, allowing the lock to return to its secured state.
For rejected access attempts, the relay is not activated. The red LED and buzzer provide an indication of the unsuccessful operation. The normally closed locking arrangement also provides a default locked condition when power is unavailable, according to the hardware configuration described in the project.
These functions demonstrate how biometric identification can be connected to a physical security mechanism through an embedded controller.

C. Web-Based User and Locker Management
The React-based dashboard provides a centralized interface for managing system users and locker-related information. The implemented application supports four user categories: Super Admin, Admin, Staff, and Student. User profiles can be created with their respective roles, login PINs, and locker assignments.
The dashboard also provides a locker management section that displays the status of up to five locker entries. Administrators can view locker information and initiate remote unlocking operations through the interface. However, the current physical prototype uses one relay-controlled locking mechanism. Therefore, the five-locker dashboard represents the software management capability and should not be interpreted as five independently connected physical locks.
This distinction is important when evaluating the current implementation and its possible expansion.

D. Access Logging and Activity Monitoring
The system maintains access records through the backend and SQLite database. Each recorded access event includes relevant information such as the user identity, associated locker, action, and timestamp. Both granted and denied access

activities can be represented in the access-history section.

The dashboard provides a view of the latest 100 access events, arranged with the most recent records first. When an access event is received by the backend, the corresponding information is stored and distributed to connected dashboard clients.

This feature improves activity traceability because administrators can review previous access operations rather than depending only on physical observation. It also provides a basis for identifying repeated unsuccessful access attempts and examining equipment usage patterns.

E. Real-Time Communication and Hardware Monitoring

The integration of HTTP communication and Socket.IO enables information exchange between the hardware, backend, and frontend. HTTP requests are used for operations such as access-event reporting, fingerprint enrollment, and hardware communication. Socket.IO distributes relevant updates to active dashboard clients.

The ESP32 periodically sends heartbeat messages to the backend. These messages contain device information, including its local IP address. The backend uses the received heartbeat information to determine whether the hardware is responding. When communication is interrupted beyond the configured monitoring interval, the device is marked as offline and the dashboard receives a status update.

This mechanism provides administrators with information about hardware connectivity without requiring them to inspect the physical controller continuously.

F. Inventory Management and Analytical Features

The implemented dashboard includes an inventory management module for maintaining information about equipment stored in lockers. Administrators can add items, modify quantities, and review their stock conditions.

The application classifies inventory according to the available quantity. Items with zero quantity are categorized as out of stock, quantities below three are marked as low stock, and quantities of three or more are considered in stock.

An analytics section provides graphical summaries of access activities. It includes a weekly access trend, a chart showing frequently occurring access hours, and a comparison between granted and denied access events. These visual summaries help administrators understand usage patterns through information stored in the access-log database.

When the database does not contain actual activity records, the application can display sample information for demonstration. Such sample values must be distinguished from results obtained during real system operation.

G. Offline Operation and Network Dependency

One important characteristic of the developed hardware is its ability to continue local fingerprint scanning and access decisions when Wi-Fi connectivity is interrupted. The fingerprint templates are maintained in the sensor's internal memory, allowing the hardware to perform the matching operation without requiring continuous communication with the backend.

During a network interruption, access events can be transmitted when connectivity becomes available again, as described in the project implementation. However, web-based administration and real-time dashboard updates depend on communication with the backend. Consequently, offline hardware operation should not be confused with complete offline availability of every software feature.

This arrangement reduces dependence on external cloud services while retaining local-network monitoring when the connection is available.

H. Discussion of Overall System Performance

The implemented prototype demonstrates the integration of biometric verification, electronic lock control, user administration, access logging, and web-based monitoring within one application. The separation of hardware operations and administrative software makes it possible to manage physical access and related information through different but connected modules.

The R307 sensor performs fingerprint matching, the ESP32 controls the locking mechanism, and the backend maintains communication between the embedded unit and the web application. The dashboard adds administrative functions that extend beyond basic fingerprint-controlled locking,

including role-based user records, locker information, inventory management, and access analytics.

The implementation also has certain limitations. The current hardware arrangement demonstrates one physical locking unit, while the dashboard supports multiple locker records. The prototype's dashboard authentication stores PIN values in plaintext, and the project documentation identifies the absence of HTTPS/TLS as a security limitation. These aspects should be improved before deployment in an environment requiring stronger security protection.

Furthermore, a complete quantitative evaluation would require controlled experiments covering fingerprint recognition accuracy, false acceptance rate, false rejection rate, average authentication time, network response time, and reliability over repeated operations. These measurements are not presented as verified experimental results in this section.

Overall, the developed system demonstrates the practical feasibility of combining biometric access control with centralized equipment and locker management. Its implementation provides a foundation for further testing, security improvements, and expansion to multiple physical locking units.

VII. CONCLUSION AND FUTURE SCOPE

A. Conclusion

BioLocker Pro presents an intelligent approach to securing laboratory equipment and lockers through fingerprint-based authentication. The system integrates the R307 fingerprint sensor with the ESP32 microcontroller to identify registered users and control an electronic locking mechanism. Along with physical access control, the web-based dashboard provides facilities for user registration, permission management, fingerprint administration, access-history tracking, and inventory monitoring.

The integration of hardware and software allows administrators to monitor access activities and hardware status through a centralized interface. The system also supports local-network communication, reducing the need for external cloud services. The developed prototype demonstrates the practical use of biometric technology for improving equipment security, user accountability, and locker management in laboratory environments. Its modular design provides opportunities for further development and expansion.

B. Future Scope

- Connecting additional physical lockers to provide independent access control for multiple equipment units.
- Strengthening application security through encrypted communication and secure PIN storage.
- Developing a dedicated mobile application for convenient monitoring and administration.
- Connecting the system with institutional databases for easier student and staff management.
- Introducing RFID cards or OTP verification as additional authentication methods.
- Adding advanced reports to analyze equipment usage and user access patterns.
- Conducting detailed experiments to evaluate fingerprint accuracy, authentication time, and system reliability.
- Improving offline event storage and automatic synchronization when network connectivity is restored.
- Introducing notification facilities to alert administrators about unauthorized access attempts and hardware failures.
- Expanding inventory management to support more equipment categories and locker locations.

ACKNOWLEDGMENT

The authors would like to express their sincere gratitude to Mrs. Mariam T. Naik(Project Guide, CSE Department, Sharad Institute of Technology College of Engineering) for her continuous guidance, valuable suggestions, and academic support throughout the development of this project. Her mentorship significantly contributed to the successful completion of this research work.

Additionally, the authors acknowledge the support of the Department of Computer Science and Engineering, Sharad Institute of Technology College of Engineering, Ichalkaranji, for providing the

necessary resources and academic environment to carry out this study.

REFERENCES

[1]. W. Yang, S. Wang, N. M. Sahri, N. M. Karie, M. Ahmed, and C. Valli, “Biometrics for Internet-of-Things Security: A Review,” Sensors, vol. 21, no. 18, Art. no. 6163, Sep. 2021, doi: 10.3390/s21186163.

[2]. M. Trnka, A. S. Abdelfattah, A. Shrestha, M. Coffey, and T. Cerny, “Systematic Review of Authentication and Authorization Advancements for the Internet of Things,” Sensors, vol. 22, no. 4, Art. no. 1361, Feb. 2022, doi: 10.3390/s22041361.

[3]. R. Trabelsi, G. Fersi, and M. Jmaiel, “Access control in Internet of Things: A survey,” Computers & Security, vol. 135, Art. no. 103472, Dec. 2023, doi: 10.1016/j.cose.2023.103472.

[4]. Espressif Systems, “ESP32 Series Datasheet,” Espressif Systems. [Online]. Available: https://documentation.espressif.com/esp32_datasheet_en.html

[5]. Adafruit Industries, “Adafruit Fingerprint Sensor Library,” GitHub repository. [Online]. Available: https://github.com/adafruit/Adafruit-Fingerprint-Sensor-Library

[6]. React Team, “React Documentation.” [Online]. Available: https://react.dev/

[7]. OpenJS Foundation, “Node.js Documentation.” [Online]. Available: https://nodejs.org/docs/latest/api/

[8]. Socket.IO, “Socket.IO Documentation.” [Online]. Available: https://socket.io/docs/v4/

[9]. SQLite Consortium, “SQLite Documentation.” [Online]. Available: https://www.sqlite.org/docs.html