Easy Parcel: IoT-Based Parcel Storage System for University Villages

by

Muhammad Afiq bin Zakaria

21001351

Dissertation submitted in partial fulfilment of
the requirements for the

Bachelor of Computer Science (Hons)

(BCS)

FYP II September 2025

Universiti Teknologi PETRONAS

32610, Bandar Seri Iskandar

Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

Easy Parcel: IoT-Based Parcel Storage System for University Villages

by

Muhammad Afiq bin Zakaria
21001351

A project dissertation submitted to the
Computer Science Programme
Universiti Teknologi PETRONAS
in partial fulfillment of the requirement for the
BACHELOR OF COMPUTER SCIENCE (Hons)
(COMPUTER SCIENCE)

Approved by,

_____

(Dr Siti Nurlaili bt Karim)

UNIVERSITI TEKNOLOGI PETRONAS
BANDAR SERI ISKANDAR, PERAK

September 2025

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project. The original work is entirely my own except as specified in the references and acknowledgements. I further declare that this work has not been submitted to any other institution or person and has not been undertaken or completed by any unspecified sources or individuals.

_____

(MUHAMMAD AFIQ BIN ZAKARIA)

ABSTRACT

The rapid growth of e-commerce has significantly increased parcel delivery volumes, leading to new logistical challenges related to security, efficiency, and timely collection, particularly in high-density living environments. University residential areas are especially affected, as students often face difficulties retrieving delivered parcels due to limited mailroom operating hours, security risks, and inconvenient collection points. To address these issues, this study presents an IoT-based smart parcel storage system, Easy Parcel, designed specifically for university village environments to improve parcel accessibility, enhance security, and streamline delivery workflows. The proposed solution integrates an ESP32 microcontroller, a Flutter mobile application, and a Supabase cloud database to enable secure parcel registration, OTP-based authentication, barcode scanning, and real-time synchronization between the smart locker and user application. A fully functional prototype was developed and evaluated through system testing, demonstrating successful user authentication, real-time database communication, and accurate locker actuation for parcel release. The results confirm that the system effectively meets its objectives and provides a scalable foundation for future enhancements, including expanded locker modules and advanced authentication methods, supporting the transition toward smarter and more efficient campus parcel management systems.

ACKNOWLEDGEMENT

First and foremost, I would like to express my deepest gratitude to Allah SWT for granting me the strength, patience, and guidance to complete this Final Year Project. Without His blessings, this journey would not have been possible.

I would also like to sincerely thank Universiti Teknologi PETRONAS (UTP) for providing the opportunity and platform to learn, explore, and grow throughout my years of study. My appreciation also goes to the Department of Computer and Information Sciences for their continuous support and for creating an environment that encourages learning and innovation.

My heartfelt thanks go to my supervisor, Dr. Siti Nurlaili Bt Karim, for her guidance, kindness, and continuous support throughout this project. Her advice and encouragement have helped me stay focused and motivated from the beginning until the end.

I am also deeply grateful to my mother, Norhijah binti Mustaffa, for her endless love, prayers, and sacrifices. Her belief in me has always been my greatest source of strength and motivation.

To my friends and seniors, thank you for always being there with your advice, encouragement, and support. You have made this journey more meaningful and enjoyable through every step of the process.

Lastly, I would like to thank myself for staying strong and not giving up despite the challenges faced. This project has been a valuable journey of learning and growth, and I am proud of the effort and perseverance that brought me to this point.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background of Study

Over the past decade, Malaysia has experienced remarkable growth in digital transformation, particularly within the e-commerce sector. The increasing use of the internet, mobile devices, and digital payment platforms has greatly influenced how Malaysians shop and interact with businesses. According to the Department of Statistics Malaysia (2024), the country's e-commerce income reached RM918.2 billion in the first nine months of 2024, marking a 4.0% year-on-year growth (Mahidin M, U, 2024). This reflects Malaysia's strong commitment to digitalization and its continuous effort to strengthen the nation's digital economy through innovation and technology adoption (Salleh et al., 2023).

Among this expanding population of online consumers, university students represent a particularly active and influential segment (Kaur, 2024). They are typically tech-savvy, highly connected through mobile and internet technologies, and rely heavily on digital platforms for both academic and personal needs. With their busy schedules and limited transportation options, students often prefer the flexibility and ease offered by online shopping. Factors such as price comparison tools, promotional deals, student discounts, and free delivery services further enhance the attractiveness of e-commerce to this demographic, making it an essential part of their daily lifestyle.

However, the increasing volume of online shopping among students has led to growing challenges in parcel collection and management within university environments. Most higher learning institutions, including Universiti Teknologi PETRONAS (UTP), rely on centralized parcel collection hubs such as the ParcelHub system. While this model allows efficient handling of large numbers of deliveries, it often creates accessibility issues for students living far from the collection center. Limited operating hours, long queues during peak delivery times, and additional

collection fees have made the process inconvenient for many. Similar challenges have been observed in other universities where traditional parcel systems struggle to cope with high delivery volumes and the need for around-the-clock accessibility (Yusoff, 2023).

These ongoing issues highlight the need for a more student-friendly, secure, and technologically advanced parcel collection system. In recent years, smart locker systems have gained attention as an efficient and automated alternative for last-mile delivery management. Studies suggest that integrating Internet of Things (IoT) technologies into parcel lockers can enhance security, improve accessibility, and reduce human dependency in parcel handling (Nguyen et al., 2022). Building upon these advancements, this project focuses on developing an IoT-based Smart Locker System tailored for university environments. The system aims to improve the convenience and reliability of parcel collection for students by incorporating features such as password-protected access, camera monitoring, and a mobile application for both couriers and users. By doing so, the project seeks to enhance the overall parcel management experience within campus residential areas while supporting Malaysia's ongoing digital innovation initiatives.

1.2 Problem Statement

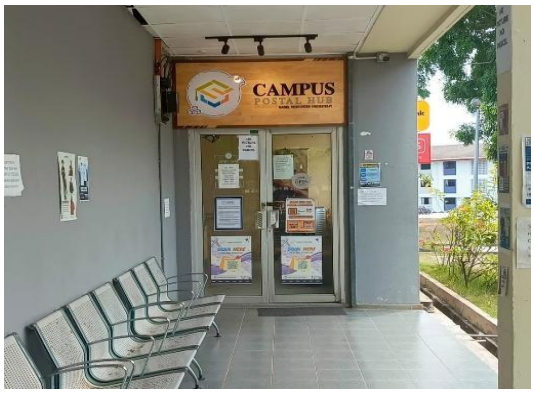


Figure 1.1: UTP ParcelHub.

Despite the introduction of centralized parcel management systems such as ParcelHub at universities like Universiti Teknologi PETRONAS (UTP), several

challenges continue to affect student convenience and overall system efficiency. One of the main issues is the location of the ParcelHub, which is situated far from most student residential villages, causing accessibility difficulties, particularly for students without personal transportation. Many are required to walk long distances or depend on others for rides, which becomes especially inconvenient during unfavorable weather conditions, tight academic schedules, or at night. Additionally, during peak delivery periods such as after semester breaks, festive seasons, or major online sales, students often face long queues and overcrowding at the hub, leading to wasted time and unnecessary stress. The situation is further aggravated by limited parking spaces near the ParcelHub, resulting in campus congestion and safety concerns, especially during high-traffic hours. Moreover, the additional handling or service fees imposed for parcel collection can accumulate over time, placing a financial strain on students who frequently shop online. These persistent issues highlight the limitations of the current centralized system in fully accommodating student needs and call for improvements to make parcel collection more convenient and student friendly.

1.3 Objectives

This project consists of four main objectives:

1. To design and develop an IoT-based smart locker system for student villages to store small parcels securely.

2. To build a mobile application for both students and couriers to manage locker access and track deliveries.

3. To implement a random password lock system and barcode scanning for secure identification of parcel senders (couriers) and receivers (students).

4. To evaluate the usability and effectiveness of the Easy Parcel system

through user testing and feedback from students and couriers.

1.4 Scope of study

The scope of this study focuses on the design and development of an Internet of Things (IoT) based smart locker system integrated with a mobile application to improve parcel management efficiency within a university environment. The study emphasizes the implementation of both hardware and software components using the Design and Development Research (DDR) methodology, which involves iterative design, prototyping, integration, and testing.

The scope of the hardware development is limited to the construction of a single smart locker prototype equipped with an ESP32 microcontroller, servo motor, keypad module, buzzer, light-emitting diodes (LEDs), and an I2C LCD display. These components were selected to demonstrate the locker's core functionality, including parcel storage, one-time password (OTP) verification, and door control. The hardware prototype was designed to represent one functional unit of a larger locker system that could be expanded in future implementation phases.

The software development scope covers the creation of a mobile application using Flutter, integrated with Supabase as the backend service. The application includes two main user interfaces, one for students and another for couriers. Core software functions implemented within this scope include user registration and authentication, parcel data management, barcode and QR code scanning, OTP generation and validation, and preliminary notification design. The integration between the mobile application, IoT hardware, and Supabase database was planned and partially implemented to validate system communication and data synchronization.

The study focuses primarily on functional validation rather than large-scale deployment. Testing was conducted on a prototype level to ensure that key operations, such as parcel registration, locker unlocking, and data retrieval, worked

as intended.

This study does not include the development of multiple locker units, large-scale server hosting, or commercial deployment. Security testing such as penetration testing, user scalability analysis, and advanced encryption techniques are also excluded from the current scope due to time and resource limitations.

Overall, this research focuses on demonstrating the feasibility and effectiveness of an integrated IoT-based smart locker system for campus parcel management. The scope ensures that all core functionalities necessary for secure and efficient parcel handling are achieved within the prototype, providing a solid foundation for future development and expansion.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter presents a review of relevant literature that provides the theoretical and empirical foundation for the development of the Easy Parcel System, an Internet of Things (IoT)-based smart locker solution for automated parcel management. The purpose of this literature review is to examine past studies, technologies, and research findings related to IoT integration, smart delivery systems, mobile applications, and cloud-based data management. By reviewing these works, the chapter aims to establish the context of the current study, identify existing technological approaches, and determine the research gaps that the Easy Parcel System seeks to address.

A literature review is an essential component of any research project as it demonstrates an understanding of existing knowledge, identifies relevant methodologies, and highlights limitations in current systems. It allows researchers to position their work within the broader field, showing how it contributes to technological advancement or practical application. In the context of this study, reviewing previous works helps to define the core concepts behind IoT-based automation, communication between mobile and hardware components, and authentication mechanisms such as one-time passwords (OTPs) that ensure secure access to digital systems.

The review process involves analyzing prior research on smart locker technologies, IoT-based communication models, mobile application integration, and cloud database frameworks. These studies provide valuable insights into how similar systems have been designed, the challenges they faced, and the technological solutions they implemented. Through this review, the project validates the choice of technologies used, including the ESP32 microcontroller, Supabase backend, and Flutter mobile framework, while also highlighting how the proposed system differs

from or improves upon existing solutions.

This chapter is organized into several sections. Section 2.2 presents the theoretical framework underlying the system, focusing on key technological concepts such as IoT architecture and cloud-based communication. Section 2.3 discusses related works in the domains of smart lockers, mobile applications, database management, and security authentication. Section 2.4 identifies research gaps that remain unaddressed in existing studies and explains how the Easy Parcel System addresses these shortcomings. Finally, Section 2.5 provides a summary of the key findings from the reviewed literature and their relevance to the overall research objectives.

2.2 Theoretical Framework

The theoretical framework of this study integrates several modern technological domains that collectively enable the operation of the Easy Parcel System: the Internet of Things (IoT), cloud computing, client–server communication, and one-time-password (OTP) authentication. Together, these elements provide the structural and logical basis for developing a secure, responsive, and scalable smart-locker solution.

The Internet of Things forms the central foundation of this project. IoT refers to the networking of interconnected physical devices that collect, exchange, and process data autonomously through the Internet (Khurshid et al., 2023). These devices, equipped with sensors and microcontrollers, support real-time monitoring, automation, and decision-making in diverse domains such as logistics, smart cities, and education environments. Recent studies highlight the importance of IoT in improving operational efficiency and digital transformation across industries (Alotaibi, 2023). Within this project, the ESP32 microcontroller operates as the IoT node that enables real-time communication between the physical lockers, the Supabase database, and the Flutter mobile application.

Cloud computing provides the backbone for data management and system scalability. It allows applications to store, retrieve, and process information over distributed servers, reducing the need for on-premise hardware. The integration of IoT and cloud computing, often called “Cloud-IoT”, enhances performance, reliability, and accessibility of real-time systems (Ahmad et al., 2021). Supabase, used in this project, functions as the cloud backend offering authentication, database storage, and real-time synchronization features essential for maintaining seamless operations between users and devices.

Client–server architecture underpins the data flow of the system. In this model, the Flutter mobile application serves as the client sending structured HTTP requests, while Supabase and the ESP32 hardware respond as servers processing and executing those requests. RESTful design principles provide the standardized framework for these interactions, ensuring scalability and compatibility between diverse components (Ari Keränen et al., 2023). This structure allows the system to exchange data efficiently while remaining flexible for future upgrades.

Security and authentication mechanisms complete the theoretical framework. OTP authentication is commonly applied in IoT environments to strengthen user verification and prevent unauthorized access. It offers time-sensitive credentials for single-use verification, mitigating replay or interception risks (Alotaibi, 2023). In the Easy Parcel System, this mechanism ensures that only verified students can unlock the locker associated with their parcel, thus maintaining integrity and user trust.

In summary, the Easy Parcel System is built upon an integrated theoretical foundation combining IoT connectivity, cloud computing, RESTful communication, and OTP-based authentication. These technologies collectively enable a secure, scalable, and efficient digital infrastructure for automated parcel handling within university environments.

2.3 Related Works

2.3.1 IoT in Smart Systems

The Internet of Things (IoT) has become the cornerstone of modern automation systems, enabling seamless communication between physical devices and cloud services. Recent studies have emphasized its potential in improving efficiency, connectivity, and decision-making in various sectors, including logistics, transportation, and smart campuses. According to Soori et al. (2023) highlight that IoT in smart factories involves interconnected devices monitoring production, optimizing maintenance, energy, safety, and supply chain processes, collectively forming intelligent, automated ecosystems that enhance operational efficiency. In the context of smart environments, Manvendra Kunwar (2024) demonstrated that IoT-based automation improves the accuracy processes while reducing human intervention.

These studies collectively show that IoT serves as the enabling foundation for connected infrastructures. The proposed Easy Parcel system leverages this foundation by combining IoT hardware (ESP32 microcontroller) with software components (mobile application and cloud backend) to facilitate automated parcel delivery and retrieval in a university environment.

2.3.2 Smart Locker Systems and Parcel Automation

Smart locker systems have recently gained attention as efficient, contactless solutions for parcel storage and delivery. Alzhrani et al. (2024) developed an IoT-integrated smart locker system featuring dual authentication through both fingerprint and one-time password verification. Their findings indicated a high level of reliability and security, demonstrating the feasibility of integrating multi-factor authentication in locker systems. Similarly, Prima Atmaja et al. (2022) introduced a low-cost IoT-based smart locker for online shopping packages using barcode verification and servo-controlled compartments, which significantly reduced parcel loss and manual handling errors.

In the Malaysian context, Mohd Yusoff et al. (2023) analyzed the adoption of

parcel locker systems, concluding that perceived ease of use and system reliability were critical factors influencing user acceptance. These findings are consistent with Zainuddin et al. (2023), who demonstrated that IoT-based parcel alert systems with QR code verification can enhance delivery reliability and reduce uncollected parcels. The Easy Parcel system aligns with these advancements by automating the delivery-to-collection workflow using OTP verification, barcode validation, and real-time system feedback, specifically tailored to the university environment.

2.3.3 Mobile Applications for IoT Integration

Mobile applications play an important role in bridging the interaction between users and IoT devices. Current research highlights the importance of user-centric mobile interfaces in controlling, monitoring, and managing IoT systems remotely. For example, Li et al. (2025) demonstrated an IoT mobile application for healthcare that enabled real-time monitoring and management, significantly improving user convenience and remote control capabilities. Similarly, Tazi et al. (2023) evaluated the accessibility of IoT Android companion apps, emphasizing the need for inclusive design to ensure effective user interaction with IoT devices via mobile platforms.

In the context of locker systems, Guan et al. (2024) proposed an accessibility-enhanced mobile interface for parcel lockers, which used deep-learning-based audio and visual guidance to assist visually impaired users. These studies underscore the importance of mobile integration in ensuring usability, accessibility, and real-time interaction. The Easy Parcel system extends this concept by integrating a Flutter-based application that enables couriers and students to interact seamlessly with the IoT locker through real-time OTP generation, parcel tracking, and status monitoring.

2.3.4 Cloud Databases and Real-Time Communication

The integration of cloud computing with IoT devices has enabled real-time data synchronization, scalability, and reliability. According to Alkateeb & Abdullah (2024), cloud computing provides the scalable infrastructure necessary for

processing, storing, and analyzing the massive data generated by IoT devices, supporting continuous communication between devices and users. Recent developments such as Supabase and Firebase have been widely adopted for real-time IoT systems due to their support for authentication, RESTful APIs, and event-driven data updates. For instance, Meiryani et al. (2023) demonstrated a cloud-based IoT platform for smart living environments, achieving low latency and high reliability in real-time data monitoring and device control.

These architectures offer a framework for developing interactive systems where data from IoT hardware can be reflected instantly on user interfaces. In the Easy Parcel system, Supabase serves as the central backend for authentication, database management, and real-time communication between the Flutter application and ESP32 hardware, ensuring instant synchronization of OTP, parcel status, and locker responses.

2.3.5 Security and Authentication Mechanisms in IoT Systems

Security remains one of the most significant challenges in IoT systems due to the continuous exchange of sensitive data between devices and cloud services. Recent studies have explored encryption, token-based authentication, and biometric verification to safeguard data integrity. Kavianpour et al. (2023) emphasized that secure IoT design must incorporate lightweight yet robust authentication mechanisms suitable for resource-constrained devices. Moreover, Kelaniki & Komninos (2025) proposed dynamic authentication frameworks that enhance protection against attacks such as replay in IoT communication.

Other works, such as Alzhrani et al. (2024), demonstrated that combining OTP with secondary authentication factors significantly enhances system resilience to unauthorized access. These insights informed the Easy Parcel system's adoption of OTP-based access control as its primary security mechanism, ensuring that only authorized users can unlock lockers and retrieve parcels while maintaining operational simplicity and user convenience.

2.4 Research Gap

Although IoT-based parcel locker solutions have been explored in previous studies, several gaps remain, particularly in relation to low-cost campus deployment and student-focused accessibility. Existing smart locker systems often prioritize advanced security mechanisms such as facial recognition or biometric scans, which enhance authentication reliability but significantly increase hardware complexity and cost. For instance, Alzhrani et al. (2024) implemented facial recognition and OTP verification for smart lockers, demonstrating high security performance while noting the necessity of robust computing resources and cameras, making such systems less feasible for budget-constrained university settings. This highlights the need for a more affordable IoT solution suitable for student residential environments.

Similarly, Prima Atmaja et al. (2022) developed a low-cost IoT smart locker system using barcode verification, proving the feasibility of economical hardware components. However, their work primarily targeted household parcel management rather than multi-user, high-volume university residential areas. This indicates a lack of research addressing scalable solutions specifically designed to manage large numbers of student users living in dispersed housing clusters.

Furthermore, studies on parcel locker adoption in Malaysia emphasise that convenience, accessibility, and system usability are key factors influencing user acceptance, yet current implementations remain centralized and often require users to travel to dedicated pickup hubs (Mohd Yusoff et al., 2023). This presents a challenge in university settings such as residential villages, where students frequently face transportation limitations and time constraints. Despite this, limited studies have designed decentralized smart locker systems positioned closer to student accommodations to improve accessibility and reduce collection delays.

In addition, many prior studies rely on proprietary or complex cloud infrastructures, whereas there is limited exploration of integrating open-source backend platforms such as Supabase for real-time data synchronisation and secure

OTP distribution. Addressing this gap is crucial as open-source technologies provide cost-effective alternatives that support scalability and independence from commercial vendors.

Therefore, this research contributes by developing a low-cost, decentralized IoT smart locker prototype using ESP32, Supabase, and Flutter, specifically tailored for university residential areas. By focusing on affordability, decentralized placement, real-time OTP authentication, and student usability, this study addresses gaps in accessibility, cost efficiency, and context-specific deployment. Ultimately, bridging these gaps is essential to improving campus logistics, reducing student burden in parcel retrieval, and supporting Malaysia's drive toward smart campus ecosystems.

2.5 Summary

This chapter reviewed existing literature related to IoT technology, smart parcel locker systems, mobile application integration, cloud database communication, and security authentication mechanisms. Prior studies demonstrated that IoT-based automation can enhance operational efficiency and security in parcel delivery environments, particularly through real-time monitoring, sensor-based control, and digital authentication. Research on smart parcel lockers highlighted the importance of secure access systems, barcode or biometric verification, and integration of cloud technologies to ensure reliable parcel tracking and user authentication. In addition, studies on mobile-IoT integration emphasized the need for user-friendly interfaces and seamless communication between mobile applications and hardware systems to support efficient parcel handling. Cloud-supported systems such as Supabase and Firebase were identified as crucial in enabling real-time data storage, retrieval, and user authentication, ensuring consistent and secure system performance.

Despite these advancements, the chapter identified significant gaps in existing research, particularly concerning low-cost solutions designed for university

environments, scalable decentralized locker placement near student residences, and streamlined OTP-based authentication using open-source platforms. While commercial systems and prior academic prototypes have focused on advanced security features or home-based implementations, limited attention has been given to affordable, campus-specific smart locker solutions that address accessibility challenges faced by students in residential villages. These gaps highlight the need to tailor IoT-based smart locker systems to the unique context of university communities, ensuring practicality, cost-efficiency, and student-centric usability.

Overall, the literature review establishes a strong foundation for the development of the Easy Parcel System by demonstrating the relevance of IoT, cloud computing, and mobile integration in improving parcel collection workflows. It also highlights the importance of addressing accessibility, affordability, and system usability within university settings. The insights gained from the reviewed studies guide the design considerations and justify the significance of this research in contributing to smarter, more accessible, and scalable parcel management solutions on campus.

CHAPTER 3

METHODOLOGY

3.1 Research Methodology

This project adopts the Design and Development Research (DDR) methodology, which is suitable for engineering-based solutions that aim to address real-world problems through systematic design and evaluation. The DDR approach emphasizes iterative development, where both hardware and software components are continuously designed, implemented, tested, and refined based on ongoing feedback and performance evaluation.

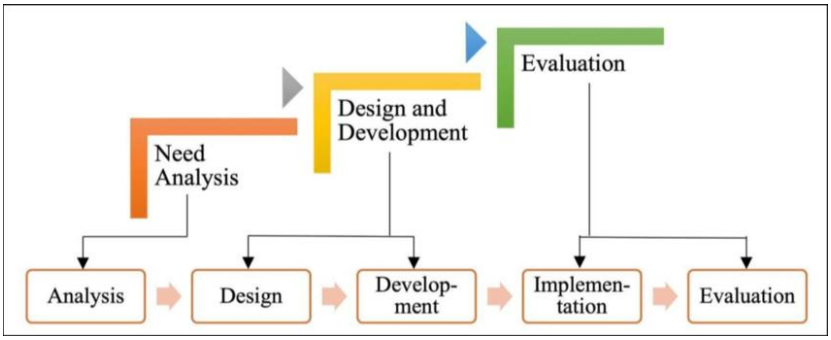


Figure 3.1:DDR Framework

In this study, the DDR methodology guides the development of an IoT-based smart locker system integrated with a mobile application to enhance parcel management efficiency within a university environment. The methodology covers the entire process from problem identification to prototype evaluation, ensuring that the system is both functional and user oriented. The research begins with the design of the smart locker hardware, focusing on the selection of materials, circuit design, and microcontroller programming. Simultaneously, the software development process involves the creation of a mobile application and backend system that enable parcel management, user notifications, and secure access control.

Following the design and development stages, the system undergoes integration and testing to ensure seamless communication between all components. A

cloud-based database is implemented to facilitate real-time data synchronization and ensure secure storage of parcel and user information. The iterative nature of the DDR methodology ensures that both hardware and software aspects are refined throughout the development process, leading to an efficient and reliable solution for on-campus parcel management.

3.2 System Development Overview

The system development process for this project follows the Design and Development Research (DDR) methodology, which emphasizes iterative design, prototyping, and testing to ensure both functionality and reliability of the proposed system. The development is divided into two main components: the hardware system for the IoT-based smart locker and the software system that includes the mobile application, backend integration, and database management.

The development process begins with requirement analysis, where the functional and non-functional requirements of the system are identified based on the challenges observed in the current parcel collection process. This stage focuses on defining the objectives, such as improving accessibility and ensuring parcel security. Once the requirements are finalized, the design phase commences, where both hardware and software architectures are planned. For the hardware, the locker layout, electronic components, and wiring configurations are determined, while for the software, the application interface, system flow, and data handling structure are designed.

During the development phase, the hardware components such as sensors, actuators, and microcontrollers are assembled and programmed to perform the necessary operations for parcel storage and retrieval. Concurrently, the software development focuses on creating a mobile application that enables students to manage parcel collection and authenticate locker access. The backend system is integrated to ensure real-time data exchange between the application and the database.

After development, the system enters the integration and testing phase, where all hardware and software components are combined into a functional prototype. This phase involves debugging, performance testing, and validation to ensure that each feature operates as intended. Any detected issues are refined through several iterations until the system achieves stable performance and reliability. The overall methodology ensures that the final prototype meets user expectations and successfully addresses the limitations identified in the existing parcel management system.

3.3 Hardware Development

3.3.1 Materials and Components

The selection of hardware components was made carefully based on functionality, reliability, and cost-effectiveness to ensure the system operates efficiently. The main control unit of the system is the ESP32 microcontroller, which was chosen for its built-in Wi-Fi and Bluetooth capabilities. These features make it ideal for Internet of Things (IoT) applications, allowing seamless communication between the smart locker and the mobile application. The ESP32 also offers sufficient processing power and GPIO pins to control multiple peripheral devices simultaneously.

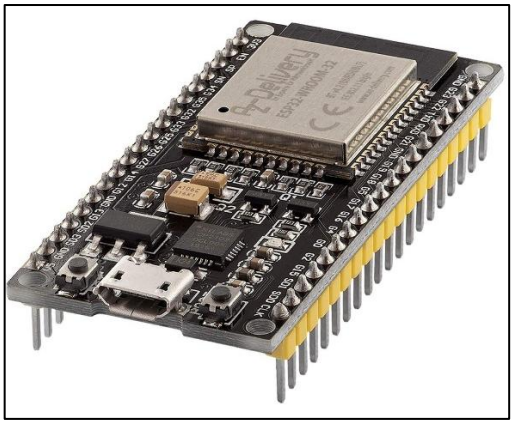


Figure 3.2: ESP32

The servo motor serves as the core actuator responsible for locking and

unlocking the locker door. When the correct one-time password (OTP) is entered through the keypad, the servo motor rotates 90 degrees to unlock the locker. Once the session is completed or the locker is re-secured, the motor rotates back to its original position, effectively locking the door again. This provides a simple yet secure mechanical control mechanism.

Figure 3.3: Servo motor

The keypad module functions as the main input device for users to enter their OTP. It allows direct interaction between the user and the locker without requiring mobile application access for verification. The system reads the input from the keypad and compares it with stored credentials before triggering the servo motor to unlock.

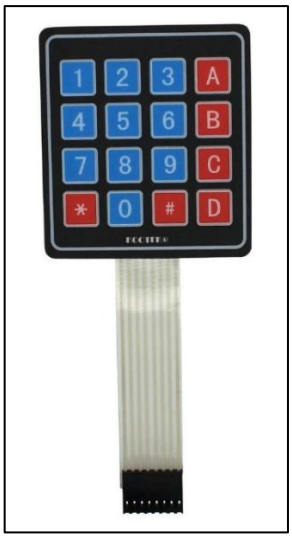


Figure 3.4: Keypad

A buzzer is included in the system to provide audible feedback to users during operation. It alerts the user when an incorrect OTP is entered, confirms successful operations such as locker unlocking, and can also serve as an alarm in case of unauthorized attempts.

Figure 3.5: Buzzer

The LED indicators are used to display system status and operational feedback. For instance, a green LED lights up when the locker is successfully unlocked, while a red LED indicates an incorrect OTP or error. These visual cues improve user experience by making the system more intuitive and responsive.

Figure 3.6: LEDs

The I2C LCD display is used to show messages such as “Enter OTP” or “Access Granted.” It provides real-time information to guide users through each step of the locker interaction. The I2C interface minimizes wiring complexity, allowing cleaner and more organized circuit connections.

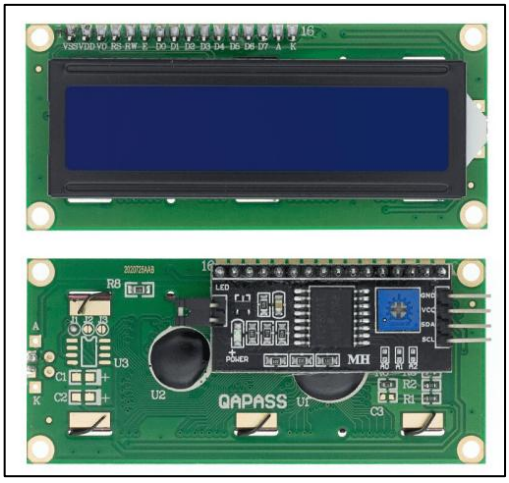


Figure 3.7: I2C LCD

Lastly, supporting components such as resistors, breadboards, jumper wires, and a regulated power supply module are used to ensure stable and safe operation. These components help distribute power correctly across all devices and maintain consistent communication between the ESP32 and peripheral modules. Each element was chosen to create a balanced design that prioritizes efficiency, scalability, and cost-effectiveness.

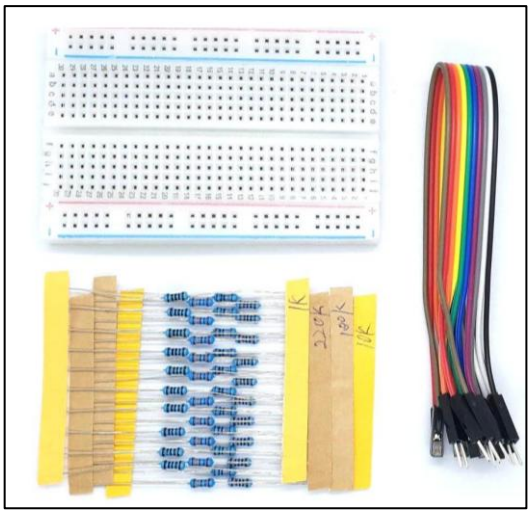


Figure 3.8: Breadboard, Resistors, Jumper wires

3.3.2 Circuit Design and Hardware Assembly

The circuit design process began with careful planning, prioritizing safety and connection integrity. The initial circuit was first designed and simulated using Tinkercad. This platform was chosen to create a virtual prototype and confirm all component connections; however, Tinkercad does not provide an ESP32 component in its library. Therefore, an Arduino UNO was used as the controller for this initial simulation as shown in Figure 3.9. This virtual testing phase was crucial for validating the core circuit logic, ensuring components like the keypad , LCD , servo , buzzer, and LEDs interacted correctly, and verifying the connections safely before assembling the physical hardware.

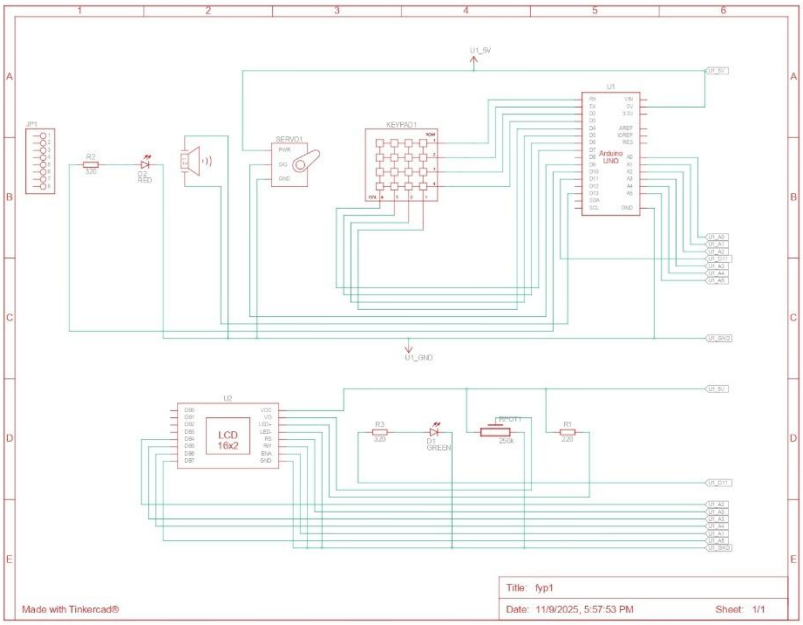


Figure 3.9: Hardware Schematic Diagram

Once the component interactions were successfully validated in the simulation, the design was migrated to the physical ESP32 microcontroller. The ESP32 was selected as the final controller for the project due to its built-in Wi-Fi and Bluetooth capabilities , which are essential for the IoT-based communication with the Supabase backend and Flutter mobile application.

In the final assembly, the ESP32 microcontroller acted as the central control unit, managing communication between all the connected components. The servo motor was connected through a pulse-width modulation (PWM) control pin, enabling precise rotation to lock and unlock the locker when the correct OTP was entered. The keypad was connected to several digital input pins to allow the microcontroller to detect user inputs. The I2C LCD display was connected using its communication interface, simplifying wiring and reducing the number of required connections compared to traditional LCD setups.

To improve usability, a buzzer and two LEDs were integrated into the circuit

as feedback indicators. The buzzer produced short tones to confirm successful actions or longer tones to indicate errors. The green LED signaled a successful OTP entry and an unlocked state, while the red LED indicated incorrect inputs or restricted access.

The final circuit was first assembled on a breadboard to allow flexible testing and troubleshooting during early development. In addition to the breadboard, an ESP32 extension board was used to provide stable power delivery and organized pin connections, reducing the risk of loose or misplaced wires. This approach improved the overall safety and reliability of the circuit, especially during repeated testing phases.

Each hardware component was tested individually to confirm correct operation before integrating the full system. The servo motor was tested to ensure smooth rotation, while the keypad and LCD were verified for accurate input and display output. This structured assembly and verification process ensured that the hardware system was fully functional and ready for integration with the programmed microcontroller.

3.3.3 Microcontroller Programming

The ESP32 microcontroller was programmed using the Arduino IDE, with the necessary libraries installed to support Wi-Fi connectivity, Over-the-Air (OTA) updates, LCD display control, keypad input, servo motor operation, and web server communication. The programming was carried out in the C++ language, and the compiled code was uploaded via USB during the testing phase.

The program begins by initializing all hardware components and setting up communication between the ESP32 and peripherals through the defined libraries. Once initialization is complete, the system enters an active state, continuously monitoring keypad inputs for OTP entry. The main logic sequence ensures that when a valid OTP is entered, the servo motor activates to unlock the locker, and visual and

audio feedback are provided through the LEDs and buzzer. This integration ensures synchronized operation and smooth user interaction between hardware and software.

Wi-Fi functionality is established early in the setup process, where the ESP32 connects to a specified wireless network using predefined credentials. The connection enables the microcontroller to communicate with the Flutter-based mobile application through an HTTP web server. Once connected, the system displays the assigned IP address on the LCD screen to confirm successful configuration.

A lightweight web server is implemented on the ESP32 to handle communication with the mobile application. It manages several endpoints, such as receiving OTPs from the app, checking system status, and confirming the current OTP value. The system employs JavaScript Object Notation (JSON) for structured data exchange and Cross-Origin Resource Sharing (CORS) headers to enable communication between different platforms securely. When a valid OTP is received from the app, it is temporarily stored and awaits verification through the keypad input.

The core system logic is organized using a state-based approach, where different system conditions are defined, including startup, idle, waiting for OTP, authentication, and servo movement. Each state determines the system's behavior, allowing smooth transitions between user input, verification, and locker operation. Once a user inputs an OTP via the keypad, the program compares it to the received value from the server. A correct match activates the servo motor to unlock the locker, accompanied by a green LED and success beep. Incorrect OTP attempts trigger the red LED and an error tone, ensuring user feedback and system clarity.

An important feature included in the program is the Over-the-Air (OTA) update mechanism, which allows new firmware versions to be uploaded wirelessly without requiring a physical USB connection. This significantly improves system

maintainability, as updates or bug fixes can be deployed directly over the network. During an OTA process, the LCD notifies users of the update progress, completion, or any errors encountered.

Additionally, the program includes a scrolling text feature on the LCD to display instructions while the locker is open, enhancing user guidance. The system also supports optional debugging features through the serial monitor for performance monitoring and troubleshooting.

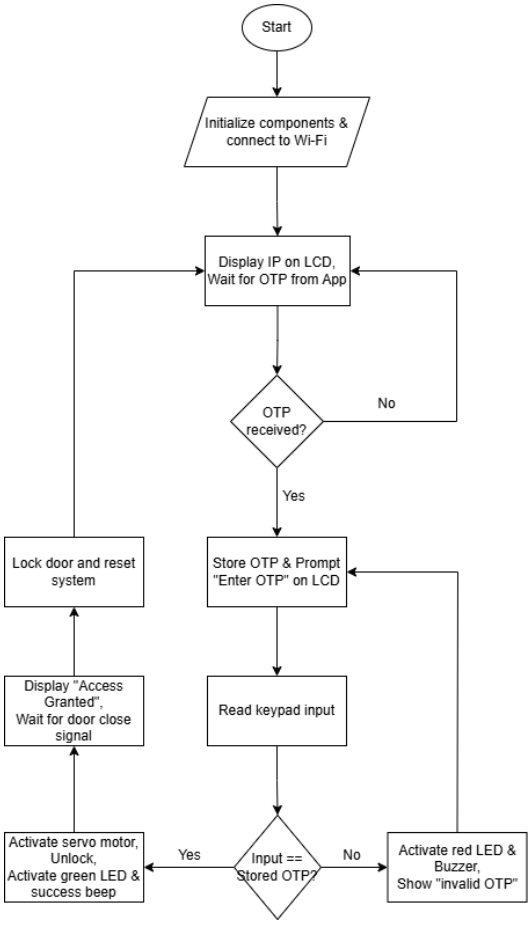


Figure 3.10: Microcontroller Program Flowchart

All program modules were developed with modularity and readability in mind to allow future improvements, such as integration with a camera module or enhanced security features. The complete program flow is illustrated in Figure 3.10, and the full source code is provided in Appendix A for reference.

3.4 Software Development

The software development of the IoT-based smart locker system involved the creation of a mobile application, backend integration, and system automation features. The software was designed to enable users to receive parcel notifications, access OTP codes securely, and interact seamlessly with the locker hardware through the ESP32 microcontroller. The development process was carried out using an iterative approach, where each feature was implemented, tested, and refined to ensure smooth functionality and user experience.

3.4.1 Mobile Application Development

The mobile application was developed using Flutter, a cross-platform framework that supports deployment on both Android and iOS devices. It serves as the main communication channel between users and the smart locker system, offering a user-friendly interface that ensures smooth interaction and efficient parcel management. The design emphasized simplicity, accessibility, and responsiveness, allowing both students and couriers to easily navigate the system.

The application was developed to cater to two main user groups: students and couriers. For students, the application allows them to log in securely, view their parcel delivery status, and retrieve a one-time password (OTP) to unlock their designated locker. The interface displays key information such as locker number, parcel arrival time, and collection status, helping students manage their deliveries conveniently and independently.

For couriers, the mobile application provides a dedicated interface that enables them to deposit parcels efficiently. Couriers can log in through a verified account, select the locker assigned to a specific recipient, and input the student's details into the system. Once the parcel is stored, the system automatically generates an OTP and send it to the intended student's account. This process helps reduce manual coordination and ensures that all transactions are recorded securely in the system.

During the early stage of software development, a mock database was implemented using a file named ‘mock_database.dart’ in Visual Studio Code. This allowed testing of both student and courier functionalities, including parcel registration, OTP generation, and user interface validation, without requiring a live backend connection. After successful validation, the mock database was replaced with Supabase, a cloud-based backend service that manages user authentication, parcel records, and data communication between the mobile application and the IoT locker system.

The complete source code, user interface design, and implementation logic for both student and courier modules are included in the Appendices for reference.

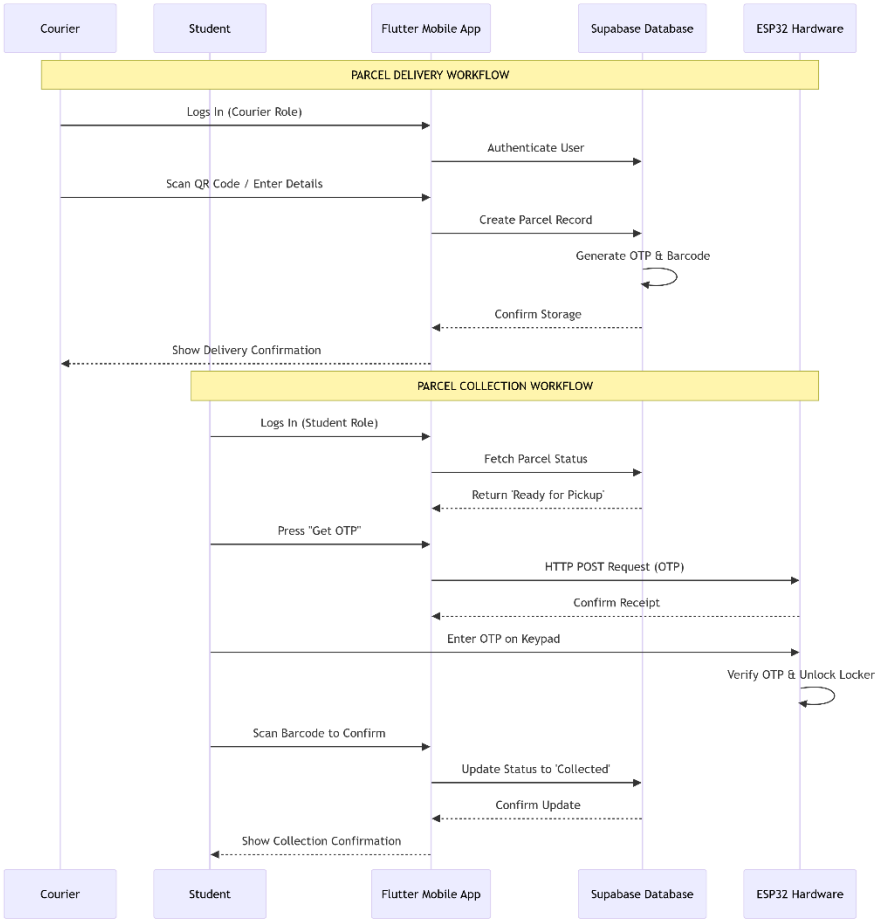


Figure 3.11: Mobile App Architecture

3.4.2 Barcode/QR Code Scanner Implementation

The barcode and QR code scanner feature was implemented in the mobile application to support parcel verification and streamline the collection process for both couriers and students. This functionality was developed using the 'mobile_scanner' package in Flutter, which enables real-time scanning through the mobile device's camera with high accuracy and performance.

For couriers, the scanner was designed to simplify the process of recording parcel information during delivery. By scanning the barcode or QR code attached to each parcel, the application automatically retrieves and displays the encoded details, such as tracking number and recipient information. This minimizes manual data entry errors and improves operational efficiency. Once the system is fully connected to the database, the scanned information will be directly stored and synchronized with Supabase to ensure all parcel records are properly updated.

For students, the scanner serves as a confirmation tool during parcel collection. When collecting a parcel from the smart locker, students can scan the barcode or QR code printed on the parcel label to verify and confirm that they have collected the parcel. The data contained in the QR code follows a structured format such as {"key1": "value1", "key2": "value2", "key3": "value3"}, allowing the system to easily parse and interpret the encoded information. At this development stage, since the live database had not yet been adopted, the scanner was primarily used to read and display the QR code content rather than storing it in the database.

The scanner feature was tested to ensure smooth functionality across multiple Android devices and consistent detection accuracy under varying lighting conditions. Detailed implementation can be reviewed in the Appendices, where the relevant code segments are provided for reference.

3.5 Database Integration

3.5.1 Database Setup

The database integration of the smart locker system was developed using Supabase, an open-source backend platform that provides a PostgreSQL database with real-time capabilities. Supabase was chosen due to its compatibility with Flutter, its ability to handle authentication and data management seamlessly, and its ease of integration with IoT and mobile systems. The database was designed to store and manage critical data such as user accounts, parcel details, and locker information efficiently and securely.

The database integration has been fully implemented. The structure of the database was designed with relational tables for users, parcels, and lockers. Each table contains key attributes such as user identification, email, phone number, parcel ID, status, locker number, barcode, and one-time password (OTP). These relationships allow the system to link each parcel to a specific student and courier, enabling accurate tracking and status updates throughout the delivery and collection process.

The database schema and entity relationships were carefully planned to ensure smooth future scalability and real-time synchronization. Supabase's built-in authentication service was used to register and manage user credentials, while parcel and locker data were handled through structured queries within the Supabase client. The schema design and table structure are provided in the Appendices for reference.

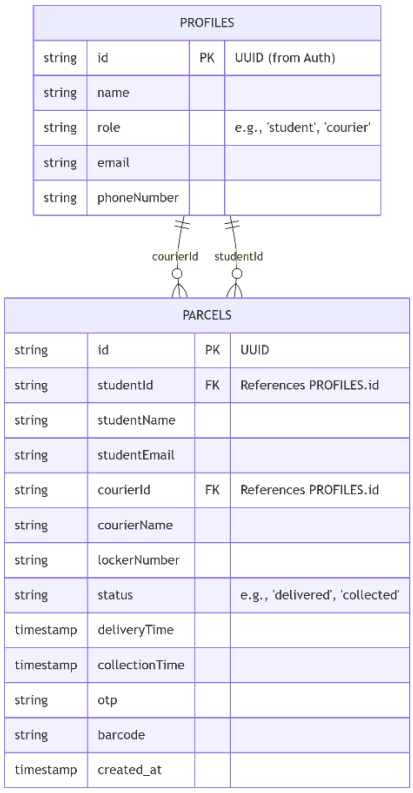


Figure 3.12: Database Schema Diagram

3.5.2 Communication Between Database and System

The communication between the mobile application and the Supabase database was established using the Supabase Flutter client. This integration allowed the system to perform various functions such as user registration, login authentication, parcel creation, and real-time data streaming for both couriers and students. All communication occurred over a secure RESTful API connection, ensuring that sensitive data such as user credentials and parcel details were transmitted safely.

When a courier creates a parcel entry, the system automatically generates a unique barcode and OTP using time-based functions within the ‘supabase_service.dart’ file. This information is then stored in the Supabase database under the corresponding parcel record. Students can later retrieve their assigned parcel data from the database through the mobile application. Real-time streaming was also implemented to allow both students and couriers to receive instant updates

when a parcel's status changes, such as when it is delivered or collected.

For verification, the system includes a method that allows barcode validation when a student scans their parcel during collection. This process compares the scanned code with the record stored in the Supabase database. If the verification is successful, the parcel status is updated to "collected," and the timestamp of the collection is stored in the database. The full implementation code for the database communication is provided in the Appendices for reference.

3.6 System Integration and Testing

System integration and testing are crucial phases to ensure that all components of the smart locker system operate together as intended. The integration process involved connecting the IoT hardware, mobile application, and the Supabase database into a single functional system. Each module, including locker control, user authentication, parcel management, and notification, was first tested individually before being combined into the complete system. The testing phase then evaluated the overall functionality, reliability, and performance of the integrated system.

3.6.1 Integration Process

The integration process was conducted in several stages. At the beginning, the IoT hardware, which consists of the ESP32 microcontroller, servo motor, and keypad module, was tested independently to verify its responsiveness and accuracy. The ESP32 was programmed to communicate with the Supabase database through the mobile application. The mobile application, developed using Flutter, acted as the main interface that connected users to both the database and the IoT device.

The integration between the mobile application and Supabase was achieved through the Supabase Flutter client, which enabled data synchronization such as parcel registration, status updates, and authentication. Although the complete connection between the IoT hardware and Supabase was not yet finalized, the system

architecture was designed to support smooth data exchange once the final setup is implemented. During this phase, mock data and simulated responses were used to verify that the mobile application could correctly retrieve and display parcel information from the database.

The overall system architecture was designed to be modular, allowing each component to be tested separately. This modular approach simplified troubleshooting and made it easier to identify and resolve issues during integration. Once the mobile application and database communication were verified, the IoT hardware was connected through the local network to simulate real-time locker operations.

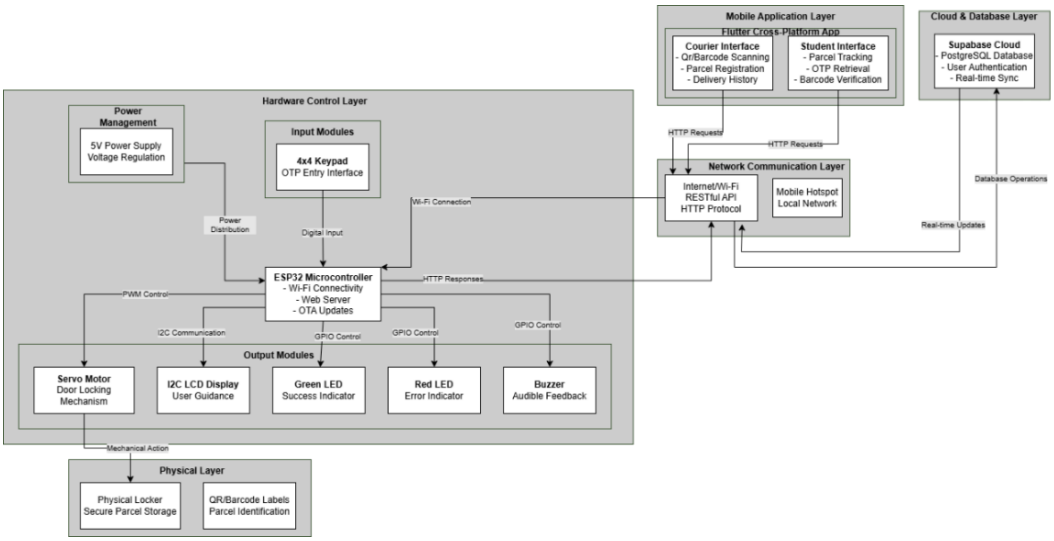


Figure 3.13: IoT System Block Diagram

3.6.2 Testing Methodology

A combination of unit testing, integration testing, and user acceptance testing was conducted to evaluate system performance and stability.

Unit testing was carried out on individual functions within the mobile application, including barcode generation, one-time password validation, and database queries. Each function was tested using sample inputs to ensure that the output produced was accurate and consistent.

Integration testing focused on verifying the communication between the mobile application, Supabase, and IoT hardware. For instance, when a courier registered a parcel in the application, the system was tested to confirm that the parcel details were correctly stored in the database and were accessible through the student's account.

User acceptance testing was conducted with a small group of users to evaluate usability, layout design, and functional accuracy. Users were asked to perform essential operations such as creating an account, viewing parcel details, and simulating parcel collection. Feedback gathered from this session was used to improve the user interface flow and resolve minor data synchronization inconsistencies.

3.6.3 Testing Results

The testing results showed that most components of the system functioned as expected. The mobile application successfully connected to the Supabase database for user registration, login authentication, and parcel data retrieval. The IoT hardware responded accurately to lock and unlock commands during simulation tests. The barcode and one-time password features were verified to work correctly, ensuring a secure verification process.

The real-time synchronization between the IoT hardware and Supabase was also successfully implemented, allowing seamless data exchange and immediate system response during parcel registration and OTP verification. The hardware was able to retrieve updated parcel credentials from the database and respond according to real-time user actions recorded in the application. This confirms that the communication framework is stable and reliable, demonstrating the system's readiness for deployment in a live environment.

The test results indicate that the system design meets its objectives, providing a structured and scalable foundation for future enhancement. Detailed test cases,

screenshots, and sample outputs are included in the Appendices for reference.

CHAPTER 4

RESULT AND DISCUSSION

4.1 System Implementation Results

4.1.1 Hardware Prototype Output

The hardware prototype of the Easy Parcel Smart Locker was successfully assembled and programmed to perform all intended operational functions, including one-time password verification, door actuation, and user feedback through visual and audio indicators. During testing, the system demonstrated stable performance with consistent component responsiveness, validating the design and programming carried out during the development phase.

Upon initialization, the liquid crystal display presented the startup screen and connection status, confirming that all peripheral modules were properly configured. The microcontroller established a connection with the local wireless network within a few seconds, after which the system displayed its internet protocol address to indicate readiness. This confirmed the reliability of the microcontroller's wireless integration and the proper execution of the startup sequence.

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

*Figure 4.1: LCD Startup State **(a)** Connnecting to Wi-Fi **(b)** Wi-Fi connected and IP Address displayed **(c)** Screen Initialization State*

During the operational test, a one-time password generated from the mobile application was transmitted to the microcontroller through the local network. Once received, the locker entered the active state and prompted the user to enter the corresponding code using the keypad. The system responded to each key input with an audible tone and accurate masking on the display, ensuring both responsiveness and confidentiality during the input process.

Figure 4.2: OTP Input

When the correct code was entered, the servo motor rotated smoothly to unlock the door while the green indicator light turned on to signal successful authentication. The display simultaneously presented confirmation messages to guide the user through the parcel retrieval process. The transition between states, from verification to motor activation, was executed without noticeable delay, showing that the microcontroller processed instructions efficiently and managed all peripherals in synchronization.

Figure 4.3: LCD State after enter right OTP

The locker remained open until the user pressed the designated command to close it. Once activated, the servo motor returned to its original position and the system displayed a “Thank You” message before resetting to standby mode. This automatic return to the idle state ensured that the system could continue operating without manual resets.

Figure 4.4: LCD state after close locker

In cases where an incorrect code was entered, the system displayed an error message and provided immediate feedback through the red indicator light and a longer buzzer tone. The short delay before allowing another attempt helped to prevent continuous retries and reinforced the system’s reliability and security.

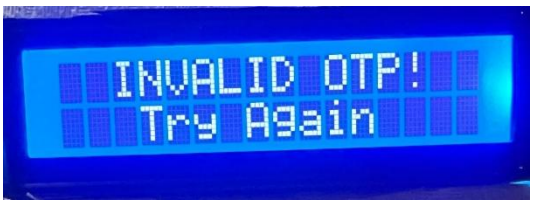


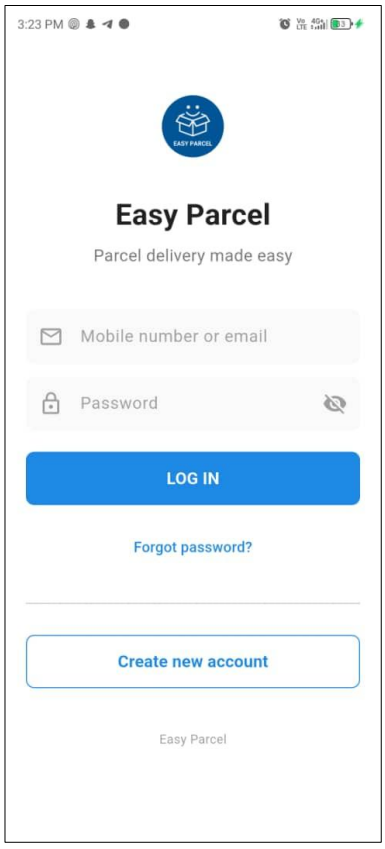
Figure 4.5: Invalid OTP LCD state

Overall, the testing confirmed that the hardware prototype performed as intended. Each component, including the display, keypad, servo motor, indicator lights, and buzzer, operated in proper coordination throughout the tests. The servo produced consistent movement during repeated open and close cycles, and the input modules maintained stable signal transmission. These results verified that the hardware and control logic worked effectively to support secure and user-friendly parcel operations, fulfilling the functional objectives of the Easy Parcel system.

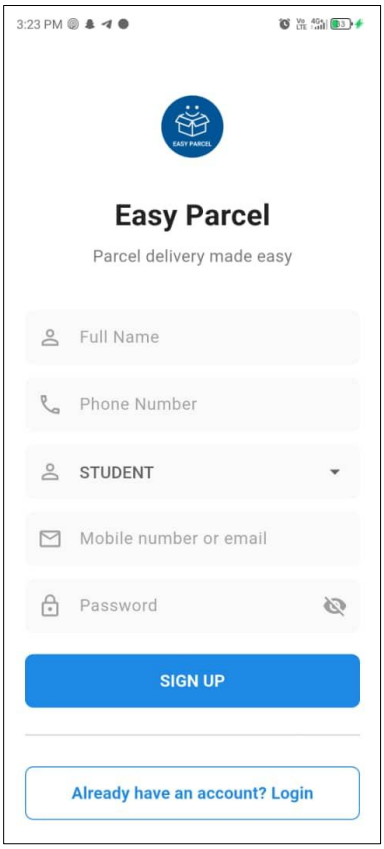
4.1.2 Mobile Application Output

The mobile application of the Easy Parcel System was successfully developed and tested using the Flutter framework, integrated with the Supabase backend and the ESP32 locker hardware. The application functioned as the central interface for both students and couriers, enabling efficient parcel management, authentication, and communication with the locker system. The testing phase confirmed that the mobile interface operated as expected across all key functions, providing a seamless and responsive user experience.

During the authentication tests, both courier and student users were able to register, log in, and access their respective dashboards according to their assigned roles. The Supabase authentication module correctly managed session states, allowing returning users to log in automatically without re-entering their credentials. This verified the successful implementation of secure session persistence and role-based navigation within the system.



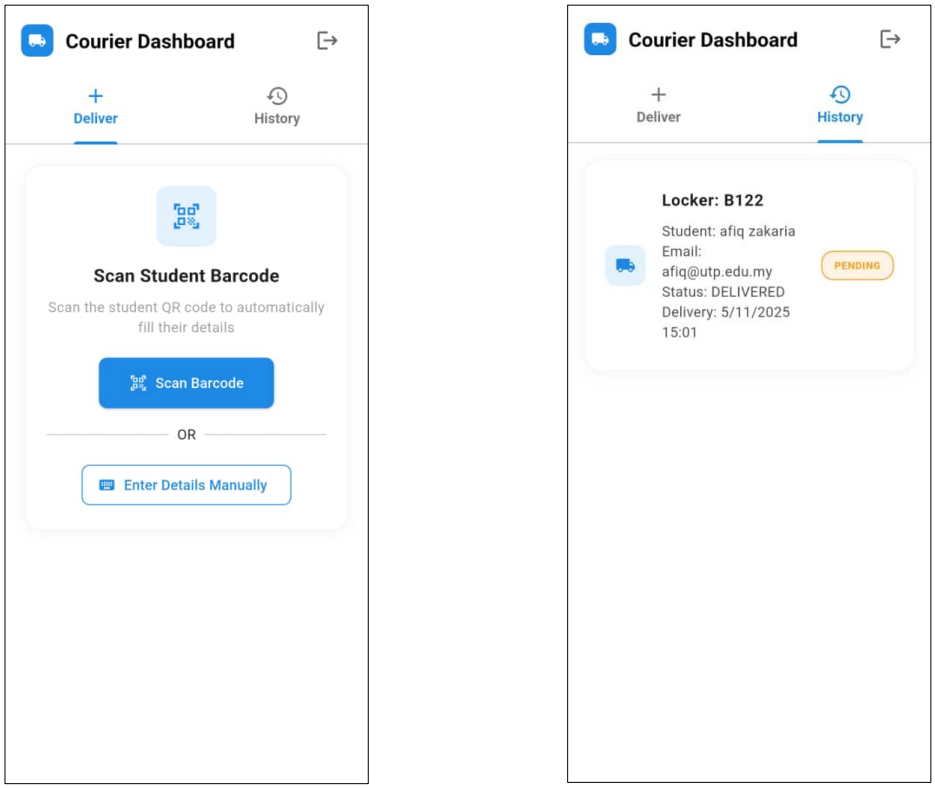
(a)



(b)

Figure 4.6: The interface of (a) Login screen (b) Registeration screen

For courier operations, the delivery workflow was tested through both scanning and manual entry methods. When a courier scanned a student's QR code or entered delivery details manually, the system generated a unique one-time password and barcode for each parcel. These values were stored in the Supabase database and displayed on the courier's interface in real time. The newly created parcel also appeared in the courier's delivery history, confirming that database synchronization and real-time data streaming were functioning correctly.

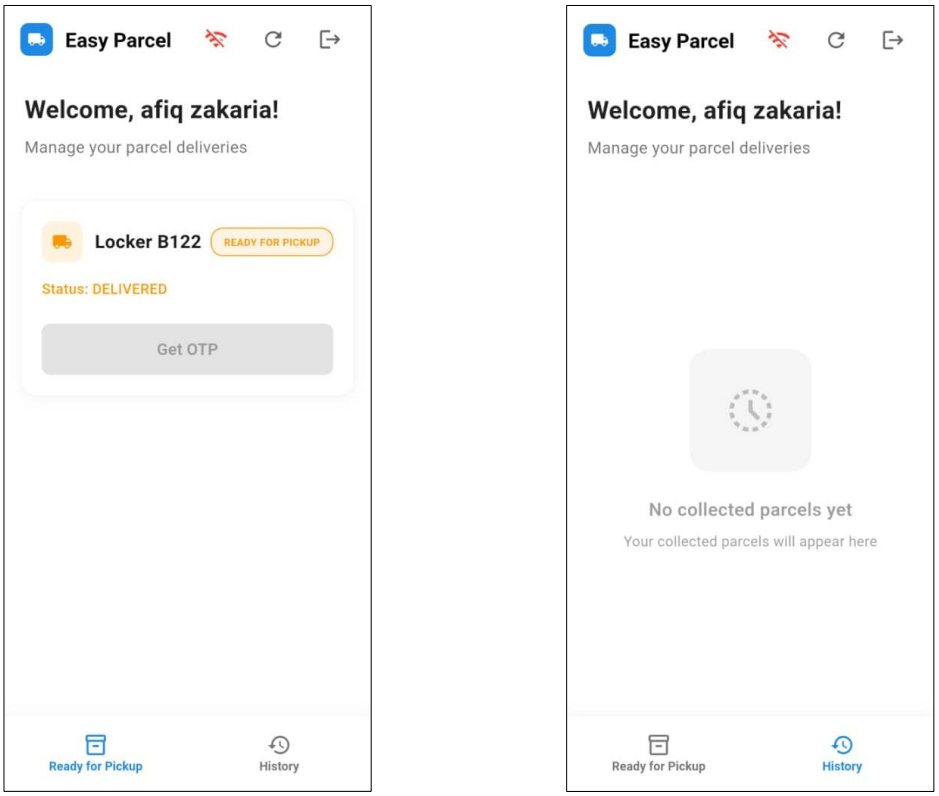


(a) **(b)**

*Figure 4.7: The interface of courier home screen **(a)** Deliver tab **(b)** History tab*

On the student side, the application effectively displayed parcels categorized under "Ready for Pickup" and "History". When a parcel reached the "Delivered" state, students were able to retrieve the associated one-time password directly from the application. Upon pressing the "Get OTP" button, the system communicated with the ESP32 hardware via an HTTP request to transmit the code to the locker. During

testing, this interaction consistently resulted in successful message delivery and confirmation responses from the hardware, indicating stable connectivity between the mobile interface and the locker unit.



(a) **(b)**

*Figure 4.8: The interface of the student home screen **(a)** Ready for Pickup tab **(b)** History tab.*

After retrieving the one-time password, students proceeded to the locker to collect their parcels. Once the parcel was collected, the barcode scanner feature was used to validate the delivery. The barcode scanning process accurately detected and verified parcel information using the mobile device's camera. Upon successful verification, the parcel status in Supabase was automatically updated from "Delivered" to "Collected," and the collection time was recorded. This confirmed that the end-to-end process, from delivery registration to parcel collection, was executed without data inconsistency or synchronization delay.

The user interface demonstrated clear feedback at each interaction point, displaying status updates such as “Locker Online,” “OTP Sent,” or “Parcel Collected.” Error handling mechanisms also worked as intended. For instance, when the locker was temporarily offline or a barcode scan failed, the system displayed appropriate notifications, allowing users to retry without losing session data.

Overall, the mobile application achieved its intended functionality and maintained reliable communication with both the Supabase backend and the ESP32 hardware. The results indicate that the system effectively supports two distinct user roles while maintaining real-time synchronization and data integrity. The testing confirmed that the Easy Parcel mobile application provides a stable, intuitive, and efficient interface for managing parcel delivery and collection in a campus environment.

4.2 Integration Results

4.2.1 Communication Between Hardware and Application

The communication between the mobile application and the ESP32 hardware was tested to evaluate the reliability of data transmission and synchronization during real operation. The connection between both components was established through a local wireless network using the mobile device’s hotspot. The ESP32 successfully connected to the network, and its assigned internet protocol address was displayed on the mobile application interface, indicating that the device was online and ready for data exchange.

During testing, the transmission of the one-time password from the mobile application to the ESP32 was consistently successful. The application sent the code using a structured HTTP POST request, and the ESP32 received and processed it immediately without noticeable delay. Each request resulted in an appropriate confirmation response, verifying that the message format and communication protocol were functioning correctly.

The status monitoring feature also performed as expected when the connection was stable. When both the ESP32 and the mobile device were connected to the same network, the application consistently displayed the locker as online. However, variations were observed when using mobile data while maintaining the hotspot connection. In such cases, the connection status occasionally fluctuated between online and offline, suggesting that network switching on the mobile device influenced the consistency of communication. Despite this, when both the hotspot and the wireless network remained active, the ESP32 maintained a stable and continuous connection throughout the test.

When the wireless connection was temporarily interrupted, the mobile application automatically attempted to refresh the connection until communication was restored. If the disconnection occurred after the one-time password had been sent, the system displayed a temporary error message at the bottom of the application interface, but the existing session remained active. Once the connection was re-established, the application resumed normal operation without requiring a restart.

Throughout the testing process, the communication between the application and the hardware remained stable and responsive under standard conditions. The data exchange between the two components was verified to be accurate, with no failed transmissions or corrupted requests detected during the trial period.

4.2.2 Database Functionality

The database functionality of the Easy Parcel System was implemented using Supabase, which provided both authentication and real-time data management features for the mobile application. During testing, the database successfully handled data transactions between the Flutter application and the backend, specifically for parcel registration, delivery tracking, and collection processes. The database tables were structured to store user profiles and parcel-related information, allowing both couriers and students to perform their respective operations without conflict or data redundancy.

The ‘public.parcels’ table was designed as the core dataset for managing parcel information. Each record contained details such as ‘id’, ‘studentId’, ‘studentName’, ‘studentEmail’, courierId, courierName, lockerNumber, status, ‘deliveryTime’, ‘collectionTime’, ‘otp’, ‘barcode’, and ‘created_at’. During system operation, new entries were automatically inserted when couriers created parcel deliveries through the application. The initial status for each parcel was recorded as delivered, and it was designed to update to collected once the student scanned the corresponding barcode through the application interface.

The ‘public.profiles’ table stored user information including id, name, role, email, and ‘phoneNumber’, which allowed the application to differentiate between couriers and students upon login. Role-based access control ensured that users only accessed the functionalities assigned to their specific roles.

Throughout testing, database synchronization between the application and Supabase was consistent and stable. The application successfully fetched and displayed updated parcel records without noticeable delay. Real-time synchronization was maintained during delivery creation and OTP transmission processes. No data inconsistency, missing records, or synchronization failures were observed during the evaluation phase. The results demonstrated that Supabase provided a reliable and responsive backend service for the Easy Parcel System, ensuring accurate data management and smooth communication with the Flutter application.

4.3 System Testing and Validation

System testing and validation were conducted to ensure that the Easy Parcel System fulfilled its intended objectives in terms of functionality, usability, and reliability. This process was designed to assess whether the system operated as expected and provided a smooth experience for both students and couriers. The testing and validation phase directly addressed the fourth project objective, which aimed to evaluate the usability and effectiveness of the Easy Parcel System through

user testing and feedback from students and couriers.

The testing process consisted of two main components, namely technical testing and user evaluation. Technical testing was performed to verify that all features integrated within the system operated correctly and consistently under normal conditions. This included validating the communication between the Flutter mobile application, the Supabase backend, and the ESP32 hardware prototype. Functions such as parcel creation, OTP generation, locker unlocking, and parcel status updating were carefully examined to confirm that they performed accurately and efficiently.

In addition to technical testing, user evaluation was carried out to measure the system's usability and effectiveness from the perspective of its end users. This was accomplished through structured survey forms distributed to both students and couriers who interacted with the Easy Parcel System. The survey was divided into three main sections that focused on usability, effectiveness, and overall satisfaction. Each question was rated using a five-point Likert scale, where one indicated strong disagreement and five indicated strong agreement. The questions for students evaluated the clarity of the interface, the ease of OTP entry, and the convenience of parcel collection, while those for couriers focused on delivery accuracy, task efficiency, and overall system reliability. Additional open-ended questions allowed respondents to share specific feedback and suggestions for improvement.

The data collected from the user testing served as an essential validation measure to determine the system's performance and user acceptance. The feedback provided by students and couriers helped to confirm that the Easy Parcel System was capable of achieving its intended objectives by simplifying parcel management and ensuring secure delivery and collection processes. All data of responses from survey are shown in Appendix B The following subsections present the outcomes of functional and user interface testing, which together provide a comprehensive evaluation of the system's performance.

4.3.1 Functional Testing

Functional testing was carried out to verify that each feature of the Easy Parcel System performed according to its design requirements. The objective of this testing was to ensure that all modules, including the Flutter mobile application, Supabase database, and ESP32 hardware prototype, operated together seamlessly to deliver an efficient and reliable parcel management process. This stage of testing focused on validating the overall workflow of the system, starting from parcel registration by the courier to parcel collection by the student through OTP verification at the locker.

During testing, the courier module within the mobile application was used to register parcels by entering the recipient's details such as name, email address, and locker number. Upon submission, the system automatically generated a unique barcode and a six-digit one-time password (OTP) that were stored in the Supabase database. The parcel entry was then reflected in the database with the status marked as "delivered." The communication between the application and the backend database was observed to be stable, with no significant delays or data transmission errors recorded during testing.

The student module was subsequently tested to confirm the accuracy of parcel retrieval and OTP verification. When the student selected a parcel for collection, the application successfully transmitted the corresponding OTP to the ESP32 microcontroller through an HTTP POST request. The ESP32 received the OTP, displayed confirmation on the serial monitor, and prompted the user to enter the same OTP on the keypad attached to the locker. Upon successful entry, the locker door was unlocked through the servo motor mechanism, and the status indicator on the LCD screen changed accordingly. This confirmed that the OTP verification and servo actuation functions were operating correctly.

Further testing verified that the LED indicators and buzzer provided accurate visual and auditory feedback during each stage of operation. The green LED

illuminated to indicate successful OTP verification, while the red LED remained active during idle or invalid entry states. The buzzer also emitted distinct tones to signal success or error events, thereby improving user interaction and confirmation during the collection process.

Throughout the functional testing, all system components responded consistently and accurately. The communication between the application, database, and hardware maintained stable connectivity, and all operations were executed without unexpected errors or interruptions. These results validated that the Easy Parcel System successfully achieved its intended functionality, supporting both courier and student operations through reliable and integrated features.

4.3.2 User Interface Testing

User interface testing was conducted to evaluate the overall usability, accessibility, and clarity of the Easy Parcel System from the perspective of both students and couriers. This testing aimed to determine whether users could navigate the application easily, perform their intended tasks without confusion, and complete parcel delivery or collection efficiently. The evaluation process also served as a means to validate the fourth project objective, which emphasized assessing the usability and effectiveness of the system through direct user feedback.

During this phase, selected participants comprising both couriers and students were invited to use the mobile application and interact with its primary features. Couriers were instructed to simulate parcel delivery by creating parcel records through the application, while students were asked to retrieve parcels using the one-time password generated and sent to the locker hardware. After completing their respective tasks, each participant was asked to complete a structured survey form designed to capture their feedback on various aspects of usability, effectiveness, and satisfaction.

The survey employed a five-point Likert scale ranging from strongly disagree

to strongly agree. Questions focused on the ease of navigation, clarity of the interface layout, responsiveness of the application, and the overall satisfaction of completing parcel-related tasks. Couriers evaluated the simplicity of parcel creation, the accuracy of OTP generation, and the system's ability to streamline their workflow, whereas students assessed the convenience of viewing parcel status, the clarity of OTP entry instructions, and the reliability of parcel collection.

The responses indicated that users found the interface to be straightforward and visually organized. Both groups reported that the layout was easy to understand and that the icons, text, and color schemes helped guide them through each process without difficulty. The feedback also reflected satisfaction with the response time of the application, as actions such as generating OTPs, updating parcel statuses, and communicating with the hardware were completed without noticeable delay. Users noted that the process of parcel delivery and collection was faster and more systematic compared to conventional manual methods.

Overall, the user interface testing confirmed that the Easy Parcel System provided a clear, user-friendly, and efficient interaction experience. The design of the Flutter application successfully supported both courier and student roles by minimizing complexity and improving operational flow. These findings validated that the system achieved its usability goals and demonstrated the potential for wider adoption in a real parcel delivery environment, such as within university residential areas.

4.4 Discussion

4.4.1 Achievement of Project Objectives

The Easy Parcel System was developed based on four main objectives, all of which were achieved through the implementation and testing process. The first objective was to design and develop an Internet of Things (IoT)-based smart locker system for parcel collection and delivery. This was successfully accomplished through the integration of the ESP32 microcontroller, keypad, servo motor, and I2C

LCD display, which together enabled parcel access control using one-time passwords. The hardware prototype operated as intended, allowing the locker door to open and close according to user authentication, while the LCD provided real-time feedback to guide user actions.

The second objective was to implement a mobile application that supports parcel tracking and verification features. This goal was fulfilled through the development of the Flutter-based application connected to the Supabase backend. The application enabled both couriers and students to perform essential tasks, including parcel creation, OTP generation, parcel status tracking, and barcode verification. The interface was successfully designed to support role-based navigation, providing separate functionalities for couriers and students in an organized manner.

The third objective was to establish seamless communication between the mobile application and the hardware through internet-based protocols. This objective was met through the use of RESTful communication between the ESP32 microcontroller and the application. Testing confirmed that the system could transmit OTP data reliably to the hardware and receive confirmation responses without noticeable delay or data loss, provided a stable Wi-Fi connection was maintained.

The fourth objective was to evaluate the usability and effectiveness of the Easy Parcel System through user testing and feedback from students and couriers. This objective was achieved by conducting survey-based evaluation and user interface testing. The responses indicated that users found the system to be practical, easy to use, and effective in managing parcel delivery and collection. The feedback also confirmed that the system successfully minimized manual handling and provided a structured and efficient process for both couriers and students.

4.4.2 Comparison with Existing Systems

In comparison with existing systems, the Easy Parcel System offers several

advantages in terms of accessibility, integration, and user engagement. Traditional parcel handling methods, particularly in university residential areas, typically rely on manual distribution processes in which students must collect parcels from staff at designated counters. Such approaches often lead to time inefficiency, misplacement risks, and limited collection flexibility. In contrast, the Easy Parcel System automates the process by enabling secure, self-service parcel collection using OTP authentication. This reduces staff involvement, minimizes waiting time, and enhances security by ensuring that only the intended recipient can access the locker.

When compared conceptually with existing commercial smart locker systems such as those used by postal and courier services, the Easy Parcel System demonstrates similar core functionalities, including secure access control and status tracking. However, its distinguishing feature lies in the integration of an open-source backend (Supabase) with a custom-built Flutter mobile application, offering flexibility, scalability, and cost efficiency. While commercial systems are typically proprietary and expensive to implement, the Easy Parcel System represents a lightweight and adaptable alternative suitable for smaller-scale environments, such as university residential areas or office buildings.

The comparison indicates that the Easy Parcel System combines the advantages of commercial smart lockers with the simplicity and affordability required for localized deployment. It therefore provides an efficient and practical solution for improving parcel handling processes within targeted user communities.

4.4.3 Contribution to Sustainable Development Goals (SDGs)

This project aligns with the principles of the United Nations Sustainable Development Goal (SDG) 11, which focuses on Sustainable Cities and Communities. SDG 11 emphasizes the need to make cities and human settlements inclusive, safe, resilient, and sustainable. Within the context of a university environment, the development of the IoT-based Smart Locker System contributes to creating a smarter and more efficient campus community through digital innovation.

The system enhances sustainability by streamlining the parcel delivery and collection process for students and couriers. Traditionally, manual parcel handling can lead to inefficiencies such as misplaced items, long waiting times, and unnecessary human contact. By automating this process, the smart locker system reduces these challenges, promotes better time management, and improves overall operational efficiency. This reflects the SDG 11 target of improving urban planning and management through the use of technology.

In addition, the project contributes to the creation of a safer and more organized environment within the university's residential areas. Features such as one-time password (OTP) access verification, barcode scanning, and real-time parcel tracking enhance the accountability and security of parcel collection. These technological implementations ensure that users can access their parcels conveniently while minimizing the risk of unauthorized access or loss.

Overall, the project demonstrates how sustainable technology can be integrated into everyday systems to improve the quality of life within a community. By aligning with SDG 11, the IoT-based Smart Locker System promotes inclusivity, efficiency, and safety, reflecting a step forward toward smarter and more sustainable campus operations.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

The development of the Easy Parcel System successfully demonstrated the potential of integrating Internet of Things (IoT) technology with mobile application platforms to enhance parcel management efficiency within a university environment. The system's primary objective was to provide a secure, automated, and user-friendly solution for parcel delivery and collection. Through the integration of the ESP32 microcontroller, Supabase backend, and Flutter mobile application, the project achieved reliable communication between hardware and software components, resulting in a seamless and responsive parcel management process.

The functional and user interface testing confirmed that all features of the system operated as intended. The mobile application effectively supported parcel registration, OTP generation, and parcel tracking, while the hardware prototype accurately executed authentication and access control processes. User testing involving students and couriers further validated the system's usability and effectiveness, with feedback indicating that the interface was easy to navigate and that the automated process significantly improved convenience and reliability compared to traditional manual handling methods.

The Easy Parcel System also contributed to sustainable digital transformation by reducing human dependency in parcel distribution and improving operational efficiency. The project aligns with Sustainable Development Goal 11, emphasizing smart and sustainable community development through the application of innovative technology. Overall, the system met all four project objectives, confirming its practicality and potential for further development and large-scale deployment within university settings and beyond.

5.2 Recommendations

Although the Easy Parcel System achieved its objectives, several improvements can be made to enhance its performance, scalability, and user experience in future implementations. One key recommendation is to strengthen the system's network stability by integrating a more reliable Wi-Fi module or supporting alternative connectivity options such as Ethernet or GSM modules. This would reduce the system's dependency on mobile hotspot connections and ensure continuous operation in various environments.

Future versions of the Easy Parcel System could also include administrative features for system monitoring, such as real-time locker usage analytics and maintenance alerts. From a hardware perspective, upgrading the prototype into a multi-locker configuration with individual compartment control would make the system more practical for large-scale deployment.

Finally, usability testing could be expanded to a larger group of users across different demographic backgrounds to obtain more diverse feedback. This would provide stronger evidence of the system's effectiveness and reveal new areas for refinement. By incorporating these enhancements, the Easy Parcel System could evolve into a comprehensive and scalable smart parcel management platform suitable for broader institutional and commercial applications.

REFERENCES

Ahmad, W., Rasool, A., Javed, A. R., Baker, T., & Jalil, Z. (2021). Cyber Security in IoT-Based Cloud Computing: a Comprehensive Survey. Electronics, 11(1), 16. MDPI. https://doi.org/10.3390/electronics11010016

Alkateeb, Z., & Abdullah, D. (2024). Unlocking the Potential: Synergizing IoT, Cloud Computing, and Big Data for a Bright Future. Iraqi Journal for Computer Science and Mathematics, 5(3), 1–13. https://doi.org/10.52866/ijcsm.2024.05.03.001

Alotaibi, B. (2023). A Survey on Industrial Internet of Things Security: Requirements, Attacks, AI-Based Solutions, and Edge Computing Opportunities. Sensors, 23(17), 7470. https://doi.org/10.3390/s23177470

Alzhrani, A. A., Balfaqih, M., Fadi Alsenani, Mohemmed Alharthi, Alshehri, A., & Zain Balfagih. (2024). Design and Implementation of an IoT-Integrated Smart Locker System utilizing Facial Recognition Technology. Engineering Technology & Applied Science Research, 14(4), 16000–16010. https://doi.org/10.48084/etasr.7737

Ari Keränen, Matthias Kovatsch, & Hartke, K. (2023, January 11). Guidance on RESTful Design for Internet of Things Systems. Ietf.org. https://www.ietf.org/archive/id/draft-irtf-t2trg-rest-iot-11.html

Guan, Z., Xiong, Z., & Fan, M. (2024). FetchAid: Making Parcel Lockers More Accessible to Blind and Low Vision People With Deep-learning Enhanced Touchscreen Guidance, Error-Recovery Mechanism, and AR-based Search Support. ArXiv.org. https://arxiv.org/abs/2402.15723

Kaur, N. (2024, January 24). *A Statistical Analysis of E-Commerce in Malaysia*. Wecorporate. https://wecorporate.com.my/blog/statistical-analysis-ecommerce-in- malaysia/

Kavianpour, S., Razaq, A., & Hales, G. (2023). A Secure Lightweight Authentication Mechanism for IoT Devices in Generic Domain. Abertay Research Portal (Abertay University), 1–6. https://doi.org/10.1109/iceccme57830.2023.10253392

Kelaniki, S. M., & Komninos, N. (2025). A Study on IoT Device Authentication Using Artificial Intelligence. Sensors, 25(18), 5809–5809. https://doi.org/10.3390/s25185809

Khurshid, K., Danish, A., Salim, M. U., Bayram, M., Ozbakkaloglu, T., & Mosaberpanah, M. A. (2023). An In-Depth Survey Demystifying the Internet of Things (IoT) in the Construction Industry: Unfolding New Dimensions. Sustainability, 15(2), 1275. https://doi.org/10.3390/su15021275

Li, J., Me, R. C., Ahmad, F. A., & Zhu, Q. (2025). Investigating the application of IoT mobile app and healthcare services for diabetic elderly: A systematic review. PLOS ONE, 20(4), e0321090. https://doi.org/10.1371/journal.pone.0321090

Mahidin M, U. (2024, December 19). *Malaysia's e-commerce income reaches RM918.2bil in first nine months of 2024*. The Star. https://www.thestar.com.my/business/business-news/2024/12/19/malaysia039s-e-commerce-income-reaches-rm9182bil-in-first-nine-months-of2024

Manvendra Kunwar. (2024, November 28). IoT in Industrial Automation: Transforming Industry Efficiency. Hashstudioz.com. https://www.hashstudioz.com/blog/iot-in-industrial-automation-enhancing-operational-efficiency-across-industries/

Meiryani, Jasmine, Fahlevi, M., & Purnomo, A. (2023, May 1). The Integration of Internet of Things (IoT) And Cloud Computing in Finance and Accounting: Systematic Literature Review. IEEE Xplore. https://doi.org/10.1109/ICBIR57571.2023.10147688

Mohd Yusoff, F. A., Mohamad*, F., Muhamad Tamyez, P. F., & Panatik, S. A. (2023). Adoption of Parcel Locker in Malaysia: Literature Review and Research Agenda. An International Journal, 15(2s). http://www.gbmrjournal.com/pdf/v15n2s/V15N2s-1.pdf

Prima Atmaja, A., Dwi Setia, L., Fajar, M. S., & Ismar, MH. R. (2022). Low Cost IoT Based Home Smart Locker to Receive Online Shopping Packages. Andalasian International Journal of Applied Science, Engineering and Technology, 2(03), 126–132. https://doi.org/10.25077/aijaset.v2i03.57

Quan, N. H., Binh, N. T., & Ly, B. T. (2022). Impact of smart locker use on customer satisfaction of online shoppers in Vietnam. *Humanities and Social Sciences Communications*, *9*(1), 1–11. https://doi.org/10.1057/s41599-022-01428-6

Salleh, N., Stegmann, H., Ong, C.-F., Shen Ow, Y., Charanya, T., Segaran, R.,

Cheah, A., Ming, W., & Hashim, A. (2023). *Harnessing the Power of Technology: Building a Strong Digital Economy for Malaysia's Future*. https://platform.mdec.com.my/cmscdn/v1.aspx?GUID=b7aed424-3c73-44ef-8e64-370d7bd38f65&file=Harnessing%20the%20Power%20of%20Technology%20-Building%20a%20Strong%20Digital%20Economy%20for%20Malaysia%E2%80%99s%20Future.pdf

Soori, M., Arezoo, B., & Dastres, R. (2023). Internet of Things for Smart Factories in Industry 4.0, a Review. Internet of Things and Cyber-Physical Systems, 3(1), 192–204. ScienceDirect. https://doi.org/10.1016/j.iotcps.2023.04.006

Tazi, F., Saka, S., Opp, G., Shradha Neupane, Das, S., Lorenzo De Carli, & Ray, I. (2023). Accessibility Evaluation of IoT Android Mobile Companion Apps. Accessibility Evaluation of IoT Android Mobile Companion Apps. https://doi.org/10.1145/3544549.3585652

Yusoff, F. A. M., Mohamad, F., Tamyez, P. F., & Panatik, S. A. (2023). The Future of Deliveries: Parcel Locker Intentions. In N. M. Suki, A. R. Mazlan, R. Azmi, N. A. Abdul Rahman, Z. Adnan, N. Hanafi, & R. Truell (Eds.), Strengthening Governance, Enhancing Integrity and Navigating Communication for Future Resilient Growth, vol 132. European Proceedings of Social and Behavioural Sciences (pp. 70-82). European Publisher. https://doi.org/10.15405/epsbs.2023.11.02.6

Zainuddin, A. A. ., Mansor, H. ., Badrulhisham, N. I. ., Zulkifli, N. N. ., Mohd Ridzal, A. A. ., & Ghazalli, N. . (2023). Simulating the Effectiveness of an IoT Parcel Alert System for Enhancing Delivery Efficiency and Safety During Covid-19. Malaysian Journal of Science and Advanced Technology, 3(1), 28–36. https://doi.org/10.56532/mjsat.v3i1.145

APPENDICES

Appendix A

```
#include <WiFi.h>
#include <ESPmDNS.h>
#include <WiFiUdp.h>
#include <ArduinoOTA.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <ESP32Servo.h>
#include <Keypad.h>
#include <WebServer.h>
#include <ArduinoJson.h>

// WiFi credentials
const char* ssid = " ";
const char* password = " ";

// Web server
WebServer server(80);

// I2C LCD configuration
LiquidCrystal_I2C lcd(0x27, 16, 2);

// Servo configuration
Servo myServo;
const int servoPin = 13;
int servoPosition = 0;

// LED configuration
const int redLedPin = 25;
const int greenLedPin = 26;

// Buzzer configuration
const int buzzerPin = 27;

// Keypad configuration
const byte ROWS = 4;
const byte COLS = 4;
char keys[ROWS][COLS] = {
  {'3', '2', '1', 'A'},
  {'6', '5', '4', 'B'},
  {'9', '8', '7', 'C'},
  {'#', '0', '*', 'D'}
};

byte rowPins[ROWS] = {19, 18, 5, 17};
```

```
byte colPins[COLS] = {2, 4, 16, 15};

Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

// OTP configuration
String inputPassword = "";
String currentOTP = "";  // This will be set by the Flutter app
const int passwordLength = 6;

// System states
enum SystemState {
  STATE_STARTUP,
  STATE_IDLE,
  STATE_WAITING_FOR_OTP,
  STATE_AUTHENTICATED,
  STATE_MOVING_SERVO,
  STATE_DOOR_OPEN,
  STATE_THANK_YOU
};
SystemState currentState = STATE_STARTUP;

// Timing variables
unsigned long previousMillis = 0;
const long interval = 1000;

// Password display timing
unsigned long keyPressTime = 0;
bool showActualKey = false;
int currentKeyPosition = -1;
const long showKeyDuration = 1000;

// Startup display timing
unsigned long startupTime = 0;
const long startupDelay = 2000;

// Scrolling text variables
unsigned long scrollTime = 0;
const long scrollDelay = 400;
int scrollPosition = 0;
String scrollText = "Press A to close";
String displayText = "";

// CORS headers function - ADD THIS FUNCTION
void addCORSHeaders() {
  server.sendHeader("Access-Control-Allow-Origin", "*");
  server.sendHeader("Access-Control-Allow-Methods", "GET, POST, PUT,
OPTIONS");
  server.sendHeader("Access-Control-Allow-Headers", "Content-Type,
Authorization");
```

```
}

void setup() {
  Serial.begin(115200);
  delay(1000);

  // Initialize LEDs
  pinMode(redLedPin, OUTPUT);
  pinMode(greenLedPin, OUTPUT);
  digitalWrite(redLedPin, HIGH);    // Red LED ON initially
  digitalWrite(greenLedPin, LOW);   // Green LED OFF initially

  // Initialize Buzzer
  pinMode(buzzerPin, OUTPUT);
  digitalWrite(buzzerPin, LOW);

  // Initialize I2C LCD
  Wire.begin();
  lcd.begin(16, 2);
  lcd.backlight();

  // Show Easy Parcel startup screen
  lcd.clear();
  printCentered("Easy Parcel", 0);
  printCentered("", 1);
  currentState = STATE_STARTUP;
  startupTime = millis();

  // Initialize servo
  ESP32PWM::allocateTimer(0);
  myServo.setPeriodHertz(50);
  myServo.attach(servoPin, 500, 2400);
  myServo.write(0);

  // Connect to WiFi
  connectToWiFi();

  // Setup web server routes
  setupWebServer();

  // Setup OTA
  setupOTA();
}

void setupWebServer() {
  // Handle OTP reception from Flutter app - MODIFIED WITH CORS
  server.on("/otp", HTTP_POST, []() {
    addCORSHeaders();
    if (server.hasArg("plain")) {
```

```
      String body = server.arg("plain");
      Serial.println("Received OTP request: " + body);

      DynamicJsonDocument doc(1024);
      DeserializationError error = deserializeJson(doc, body);

      if (error) {
        server.send(400, "application/json",
"{\"status\":\"error\",\"message\":\"Invalid JSON\"}");
        return;
      }

      String otp = doc["otp"];
      String locker = doc["locker"];
      String action = doc["action"];

      if (otp.length() == 6 && action == "unlock") {
        currentOTP = otp;
        currentState = STATE_WAITING_FOR_OTP;

        lcd.clear();
        printCentered("OTP Received!", 0);
        printCentered("Enter OTP: ______", 1);

        inputPassword = "";

        // Send success response
        DynamicJsonDocument responseDoc(1024);
        responseDoc["status"] = "success";
        responseDoc["message"] = "OTP received";
        responseDoc["locker"] = locker;
        responseDoc["otp"] = otp;

        String response;
        serializeJson(responseDoc, response);
        server.send(200, "application/json", response);

        Serial.println("OTP received: " + otp + " for locker: " +
locker);
        beepSuccess();
      } else {
        server.send(400, "application/json",
"{\"status\":\"error\",\"message\":\"Invalid OTP or action\"}");
      }
    } else {
      server.send(400, "application/json",
"{\"status\":\"error\",\"message\":\"No data received\"}");
    }
  });
```

```
  // Handle status check - MODIFIED WITH CORS
  server.on("/status", HTTP_GET, []() {
    addCORSHeaders();
    DynamicJsonDocument doc(512);
    doc["status"] = "online";
    doc["system_state"] = String(currentState);
    doc["has_otp"] = (currentOTP != "");
    doc["ip_address"] = WiFi.localIP().toString();

    String response;
    serializeJson(doc, response);
    server.send(200, "application/json", response);
  });

  // Handle current OTP check - MODIFIED WITH CORS
  server.on("/current_otp", HTTP_GET, []() {
    addCORSHeaders();
    DynamicJsonDocument doc(512);
    doc["current_otp"] = currentOTP;
    doc["system_state"] = String(currentState);

    String response;
    serializeJson(doc, response);
    server.send(200, "application/json", response);
  });

  // Handle root - MODIFIED WITH CORS
  server.on("/", HTTP_GET, []() {
    addCORSHeaders();
    server.send(200, "text/plain", "Easy Parcel Locker System - Ready
for OTP");
  });

  // Add OPTIONS handler for CORS preflight requests - NEW
  server.on("/otp", HTTP_OPTIONS, []() {
    addCORSHeaders();
    server.send(200, "text/plain", "");
  });

  server.onNotFound([]() {
    addCORSHeaders();
    server.send(404, "text/plain", "Endpoint not found");
  });

  server.begin();
  Serial.println("HTTP server started with CORS support");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
```

```
}

// Function to print centered text
void printCentered(String text, int line) {
  lcd.setCursor(0, line);
  lcd.print("                "); // Clear the line
  int spaces = (16 - text.length()) / 2;
  lcd.setCursor(spaces, line);
  lcd.print(text);
}

void connectToWiFi() {
  Serial.println("Connecting to WiFi...");
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  lcd.clear();
  printCentered("Connecting WiFi", 0);

  int attempts = 0;
  while (WiFi.status() != WL_CONNECTED && attempts < 20) {
    delay(500);
    Serial.print(".");
    attempts++;
  }

  if (WiFi.status() == WL_CONNECTED) {
    Serial.println("\nWiFi connected!");
    Serial.print("IP address: ");
    Serial.println(WiFi.localIP());

    lcd.clear();
    printCentered("WiFi Connected!", 0);
    printCentered(WiFi.localIP().toString(), 1);
    delay(2000);
  } else {
    Serial.println("\nFailed to connect to WiFi");
    lcd.clear();
    printCentered("WiFi Failed!", 0);
    printCentered("Check Credentials", 1);
    delay(3000);
  }
}

void setupOTA() {
  ArduinoOTA.setPort(3232);
  ArduinoOTA.setHostname("easy-parcel-locker");

  ArduinoOTA
```

```
    .onStart([]() {
      lcd.clear();
      printCentered("OTA Update Start", 0);
      printCentered("Please wait...", 1);
    })
    .onEnd([]() {
      lcd.clear();
      printCentered("OTA Complete!", 0);
      printCentered("Restarting...", 1);
    })
    .onProgress([](unsigned int progress, unsigned int total) {
      int percent = (progress / (total / 100));
      String progressText = "Progress: " + String(percent) + "%";
      printCentered(progressText, 1);
    })
    .onError([](ota_error_t error) {
      lcd.clear();
      String errorText = "OTA Error: " + String(error);
      printCentered(errorText, 0);
    });

  ArduinoOTA.begin();
}

void beep(int duration = 100) {
  digitalWrite(buzzerPin, HIGH);
  delay(duration);
  digitalWrite(buzzerPin, LOW);
}

void beepSuccess() {
  beep(100);
  delay(50);
  beep(100);
}

void beepError() {
  beep(300);
  delay(100);
  beep(300);
}

void loop() {
  // Handle web server
  server.handleClient();

  // Handle OTA updates
  ArduinoOTA.handle();
```

```
  unsigned long currentMillis = millis();

  // Handle startup screen timing
  if (currentState == STATE_STARTUP && (currentMillis - startupTime >
startupDelay)) {
    resetToMainMenu();
  }

  // Handle keypad input
  handleKeypad();

  // Handle password display timing
  if (showActualKey && (currentMillis - keyPressTime >
showKeyDuration)) {
    showActualKey = false;
    currentKeyPosition = -1;
    updatePasswordDisplay();
  }

  // Handle servo movement
  if (currentState == STATE_MOVING_SERVO) {
    if (currentMillis - previousMillis >= 20) {
      previousMillis = currentMillis;
      moveServoToOpen();
    }
  }

  if (currentState == STATE_THANK_YOU) {
    if (currentMillis - previousMillis >= 20) {
      previousMillis = currentMillis;
      moveServoToClose();
    }
  }

  // Handle scrolling text
  if (currentState == STATE_DOOR_OPEN) {
    if (currentMillis - scrollTime > scrollDelay) {
      scrollTime = currentMillis;
      updateScrollingText();
    }
  }
}

void updateScrollingText() {
  displayText = "";
  String paddedText = "    " + scrollText + "    ";

  for (int i = 0; i < 16; i++) {
    int textIndex = (scrollPosition + i) % paddedText.length();
```

```
    displayText += paddedText.charAt(textIndex);
  }

  lcd.setCursor(0, 1);
  lcd.print(displayText);
  scrollPosition = (scrollPosition + 1) % paddedText.length();
}

void handleKeypad() {
  char key = keypad.getKey();

  if (key) {
    beep(50);

    // Handle A key to close door
    if (key == 'A' && currentState == STATE_DOOR_OPEN) {
      closeDoor();
      return;
    }

    // Handle B key as back button (only in OTP entry)
    if (key == 'B' && currentState == STATE_WAITING_FOR_OTP) {
      resetToMainMenu();
      return;
    }

    // Handle C key to show current OTP (for debugging)
    if (key == 'C' && currentState == STATE_IDLE) {
      lcd.clear();
      printCentered("Current OTP:", 0);
      printCentered(currentOTP, 1);
      delay(3000);
      resetToMainMenu();
      return;
    }

    switch (currentState) {
      case STATE_WAITING_FOR_OTP:
        handleOTPInput(key);
        break;
    }
  }
}

void handleOTPInput(char key) {
  if (key == '#') {
    checkOTP();
  } else if (key == '*') {
    resetPasswordInput();
```

```
  } else if (isDigit(key)) {
    if (inputPassword.length() < passwordLength) {
      inputPassword += key;
      showActualKey = true;
      currentKeyPosition = inputPassword.length() - 1;
      keyPressTime = millis();
      updatePasswordDisplayWithActualKey();
    }

    if (inputPassword.length() >= passwordLength) {
      checkOTP();
    }
  }
}

void updatePasswordDisplayWithActualKey() {
  String displayStr = "";
  for (int i = 0; i < passwordLength; i++) {
    if (i < inputPassword.length()) {
      if (showActualKey && i == currentKeyPosition) {
        displayStr += inputPassword.charAt(i);
      } else {
        displayStr += "*";
      }
    } else {
      displayStr += "_";
    }
  }

  lcd.clear();
  printCentered("Enter OTP:", 0);
  printCentered(displayStr, 1);
}

void updatePasswordDisplay() {
  String displayStr = "";
  for (int i = 0; i < passwordLength; i++) {
    if (i < inputPassword.length()) {
      displayStr += "*";
    } else {
      displayStr += "_";
    }
  }

  lcd.clear();
  printCentered("Enter OTP:", 0);
  printCentered(displayStr, 1);
}
```

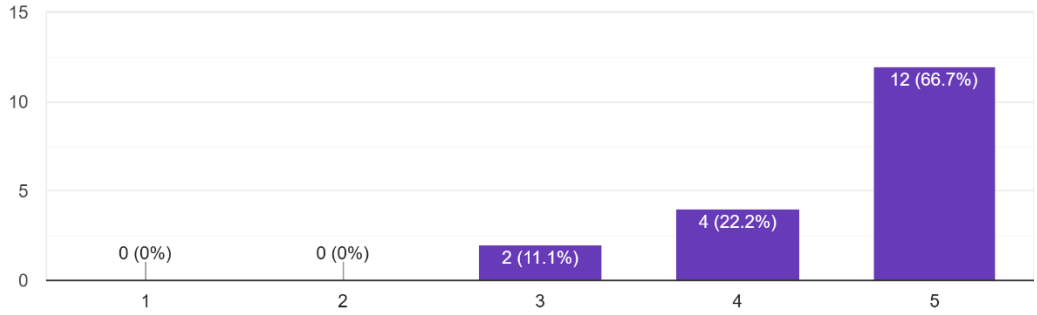
```
void resetToMainMenu() {
  inputPassword = "";
  currentState = STATE_IDLE;
  showActualKey = false;
  currentKeyPosition = -1;
  scrollPosition = 0;

  digitalWrite(redLedPin, HIGH);
  digitalWrite(greenLedPin, LOW);

  lcd.clear();
  printCentered("Easy Parcel", 0);
  printCentered("", 1);  // Empty second line
}

void resetPasswordInput() {
  inputPassword = "";
  updatePasswordDisplay();
}

void checkOTP() {
  if (inputPassword == currentOTP && currentOTP != "") {
    beepSuccess();
    currentState = STATE_AUTHENTICATED;

    digitalWrite(redLedPin, LOW);
    digitalWrite(greenLedPin, HIGH);

    lcd.clear();
    printCentered("OTP VERIFIED!", 0);
    printCentered("Door Opening...", 1);

    delay(1000);

    currentState = STATE_MOVING_SERVO;
    servoPosition = 0;

  } else {
    beepError();

    for(int i = 0; i < 3; i++) {
      digitalWrite(redLedPin, LOW);
      delay(200);
      digitalWrite(redLedPin, HIGH);
      delay(200);
    }

    lcd.clear();
    printCentered("INVALID OTP!", 0);
```

```
    printCentered("Try Again", 1);

    delay(2000);

    // Return to OTP entry state
    currentState = STATE_WAITING_FOR_OTP;
    inputPassword = "";
    lcd.clear();
    printCentered("Enter OTP:", 0);
    printCentered("______", 1);
  }
}

void moveServoToOpen() {
  if (servoPosition < 90) {
    servoPosition += 10;
    myServo.write(servoPosition);
  } else {
    currentState = STATE_DOOR_OPEN;
    scrollPosition = 0;
    lcd.clear();
    printCentered("Door Open!", 0);
  }
}

void closeDoor() {
  beepSuccess();
  currentState = STATE_THANK_YOU;
  lcd.clear();
  printCentered("Closing Door...", 0);
  printCentered("Please wait...", 1);
}

void moveServoToClose() {
  if (servoPosition > 0) {
    servoPosition -= 10;
    myServo.write(servoPosition);
  } else {
    lcd.clear();
    printCentered("Thank You!", 0);
    printCentered("", 1);

    delay(2000);
    resetToMainMenu();
  }
}
```

Appendix B

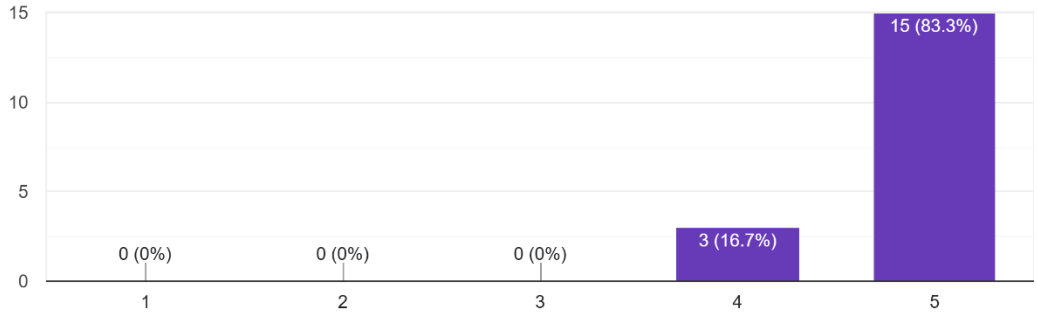
The application interface is easy to understand and navigate.

18 responses



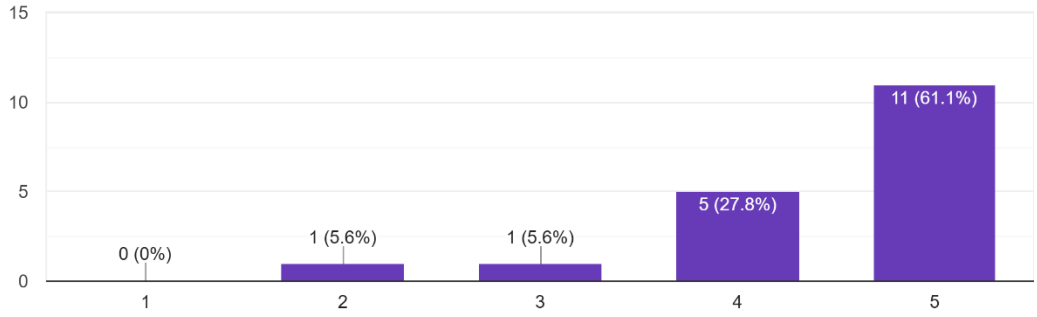
The system layout and design are clear and visually appealing.

18 responses



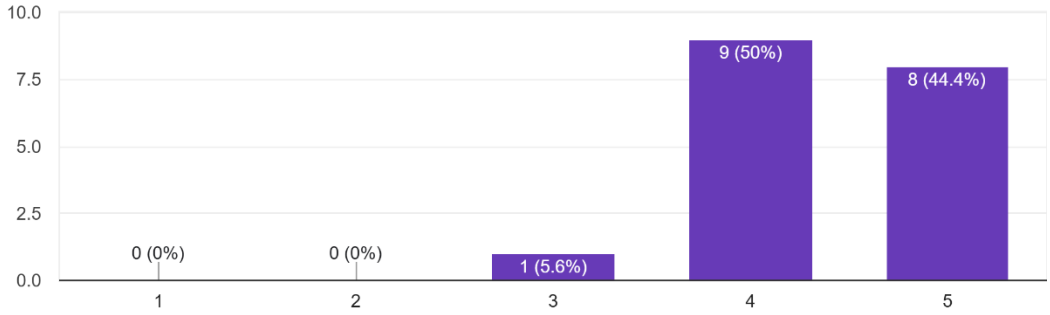
The instructions for each function (e.g., parcel creation, OTP verification) are easy to follow.

18 responses



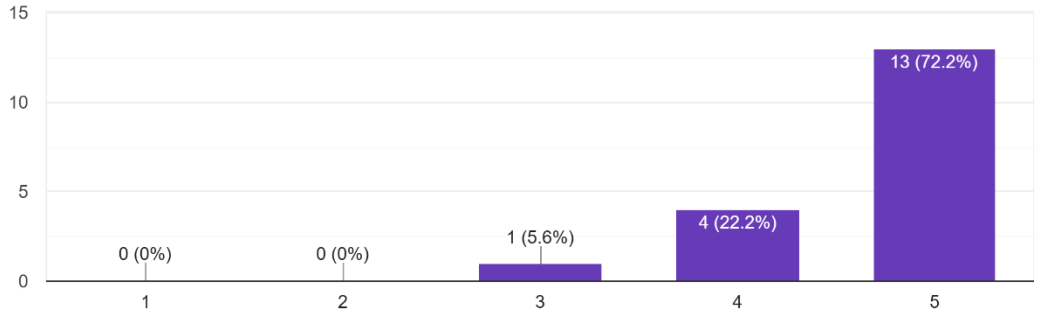
I can complete my tasks (delivery or collection) efficiently using the system.

18 responses



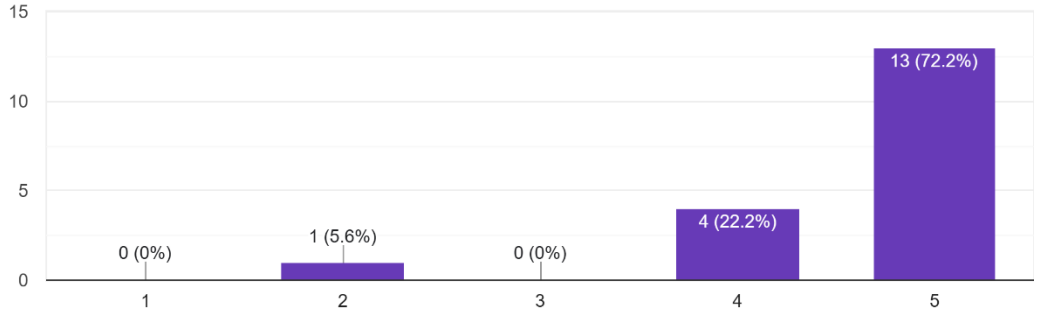
The response time of the application is fast and reliable.

18 responses



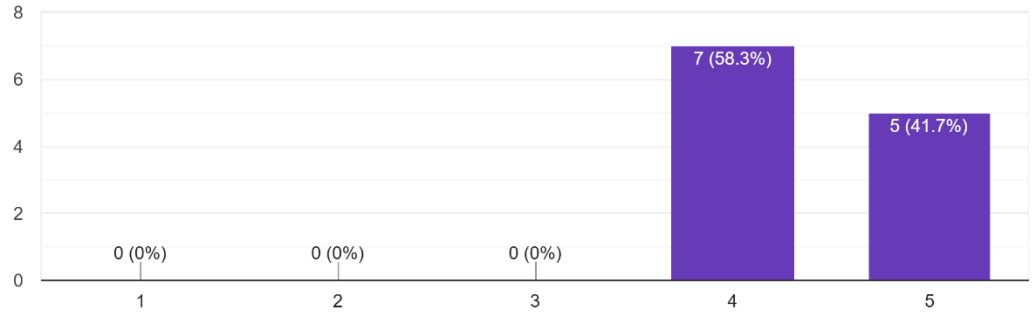
I did not experience any confusion or errors while using the system.

18 responses



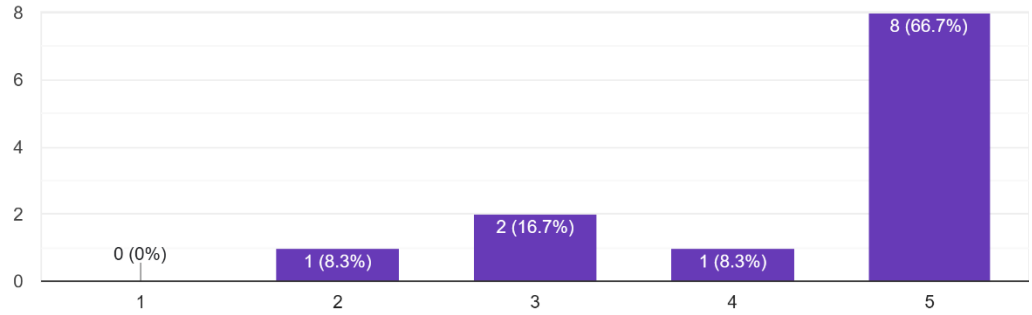
The OTP entry process on the locker is straightforward.

12 responses



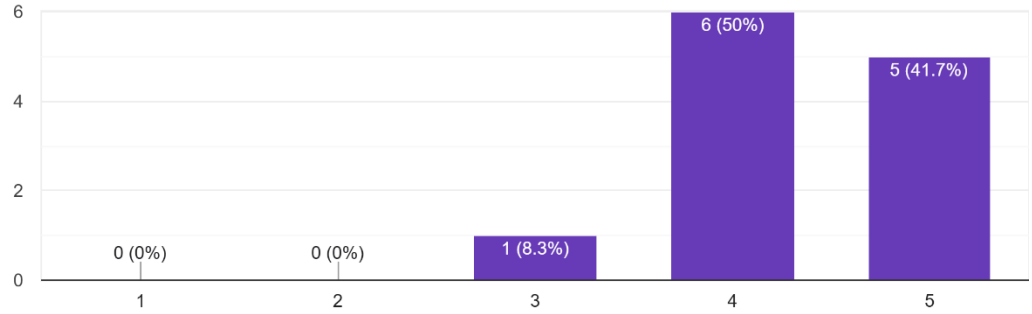
The system ensures that my parcel collection is secure and private.

12 responses



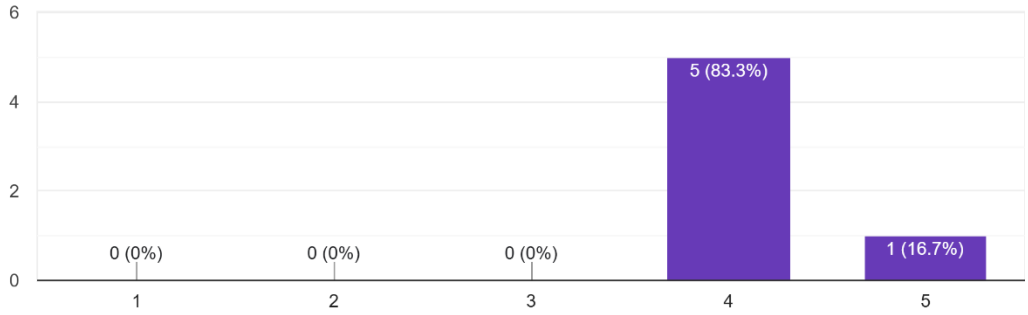
I find it convenient to view my parcel status in the app.

12 responses



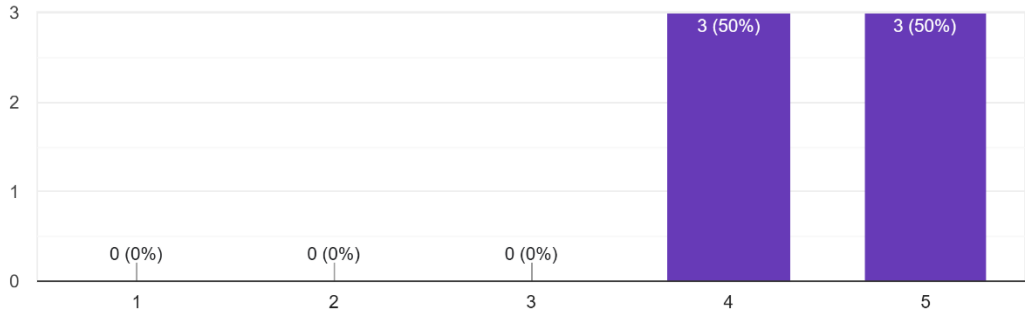
The system makes parcel delivery more organized and traceable.

6 responses



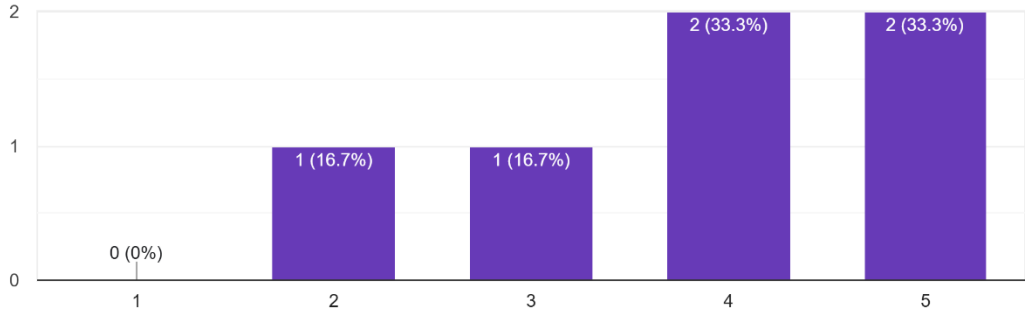
The process of generating OTPs and assigning lockers is simple and accurate.

6 responses



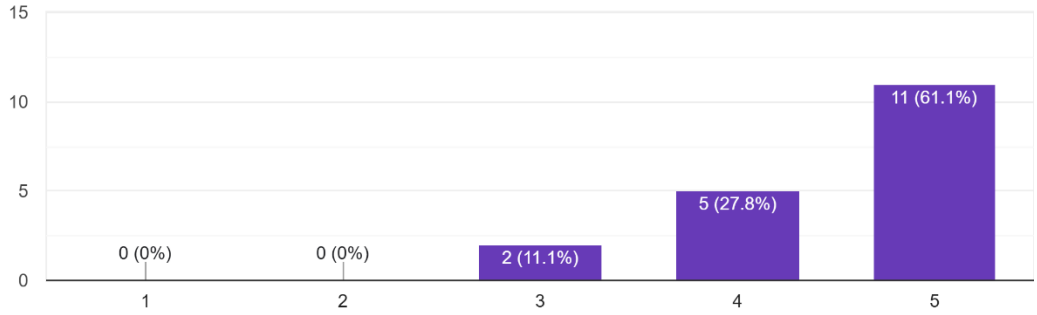
The system reduces manual work compared to the conventional delivery process.

6 responses



Overall, I am satisfied with the performance of the Easy Parcel System.

18 responses



I would recommend this system to be implemented for real parcel delivery in university residential areas.

18 responses