FACULTY OF ELECTRONICS AND COMPUTER TECHNOLOGY AND ENGINEERING

DEVELOPMENT OF INTELLIGENT BASED ON IOT FOR SECURITY LOCK USING QR CODE SPECIALIZE FOR HOMESTAY

MUHD ARIFF BIN ZULKIFLI TEOH

BACHELOR OF COMPUTER ENGINEERING TECHNOLOGY (COMPUTER SYSTEMS) WITH HONOURS

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2025

DEVELOPMENT OF INTELLIGENT BASED ON IOT FOR SECURITY LOCK KEY USING QR CODE SPECIALIZE FOR HOMESTAY

MUHD ARIFF BIN ZULKIFLI TEOH

This report is submitted in partial fulfilment of the requirements for the degree of Bachelor of Computer Engineering Technology (Computer Systems) with Honours

Faculty of Electronics and Computer Technology and Engineering
Universiti Teknikal Malaysia Melaka

2025

UNIVERSITI TEKNIKAL MALAYSIA MELAKA
FAKULTI TEKNOLOGI DAN KEJURUTERAAN ELEKTRONIK DAN KOMPUTER

BORANG PENGESAHAN STATUS LAPORAN
PROJEK SARJANA MUDA II

Tajuk Projek : Development of Intelligent Based on IoT for Security Lock Key using QR Code Specialize for Homestay

Sesi Pengajian : 2024/2025

Saya MUHD ARIFF BIN ZULKIFLI TEOH mengaku membenarkan laporan Projek Sarjana Muda ini disimpan di Perpustakaan dengan syarat-syarat kegunaan seperti berikut:

1. Laporan adalah hakmilik Universiti Teknikal Malaysia Melaka.
2. Perpustakaan dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan laporan ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. Sila tandakan (✓):

☐	**SULIT***	(Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972)
☐	**TERHAD***	(Mengandungi maklumat terhad yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan.)
/	**TIDAK TERHAD**	

Disahkan oleh:

(TANDATANGAN PENULIS) (COP DAN TANDATANGAN PENYELIA)

Alamat Tetap:

Tarikh : 10 Januari 2025 Tarikh : 20 Januari 2025

*CATATAN: Jika laporan ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa/organisasi berkenaan dengan menyatakan sekali tempoh laporan ini perlu dikelaskan sebagai SULIT atau TERHAD.

DECLARATION

I declare that this project report entitled “Development of Intelligent Based on IoT for Security Lock Key using QR Code Specialize for Homestay” is the result of my own research except as cited in the references. The project report has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Signature :

Student Name : MUHD ARIFF BIN ZULKIFLI TEOH

Date : 10/1/2025

.....

APPROVAL

I hereby declare that I have checked this project report and in my opinion, this project report is adequate in terms of scope and quality for the award of the degree of Bachelor of Computer Engineering Technology (Computer Systems) with Honours.

Signature :

Supervisor Name : MA TIEN CHOON

Date : 20/1/2025

Signature :

Co-Supervisor :

Name (if any)

Date :

DEDICATION

To my beloved mother, Normala Binti Mohamad, and father, Zulkifli Teoh Bin Abdullah,, who have been the source of inspiration and strength when I'm feeling down and giving up, who continually provide their moral, spiritual, emotional, and financial support

And

To Nur Ain Teoh, Nurshafiqah, Muhd Aliff, Iena Zalina, Mohd Hafizi,Aiman, Batrisyah, Delisya, Aisar, Raisya, Faizal Ali, Adzlan Dannish, Daniyal Iskandar and Aisy who shared their words of advice

And encouragement to finish this project

And

To my friends, Fidauddin, Asyrafudin, Mus'ab, Iwan,Taufiq, Bobok, Bob, NoMercyGang, Syazwan, Hamizan, Darwisy, Khaliff, Fakrullah, Amir, Irfan, Faiz,Ng Chin Huan, Roobakanthan, Qayyum , Zaim, Lan, Ammar, Haziq, Haikal Hafiz, Nazrul and Saifuddin whose has contributed and guide me in completing my work, giving their thoughts, idea and solution that greatly help me in solving all the problems that occur in my journey to finish this project.

And lastly,

To the Allah S.W.T, thank you for the guidance, strength, power of mind, protection and skills and for giving me a healthy life. It is because of your mighty power, I was able to finish this project successfully

ABSTRACT

This project presents the development of an IoT-based smart lock system that uses QR codes as digital keys, specifically designed for homestays. The system integrates IoT components, including the ESP32 microcontroller, QR code scanner, and Firebase database, to provide secure and contactless access control. A smartphone application allows property owners to generate unique QR codes for guests, enabling seamless check-ins without the need for physical interaction. The system's design emphasizes real-time synchronization with Firebase, ensuring efficient data validation and granting access only to authorized users. Unlike traditional lock-and-key mechanisms or RFID cards, QR code authentication eliminates risks such as key duplication, loss, or unauthorized sharing of credentials. The solution also incorporates features like real-time door status updates, user-friendly application interfaces, and customizable access duration. This project demonstrates a practical application of IoT to enhance the security and operational efficiency of homestays, addressing the demand for cost-effective and scalable solutions in the hospitality industry. The proposed system not only provides a robust security framework but also enhances the user experience for property owners and guests, paving the way for broader adoption of smart technologies in property management.

ABSTRAK

Projek ini membentangkan pembangunan sistem kunci pintar berasaskan IoT yang menggunakan kod QR sebagai kunci digital, yang direka khusus untuk *homestay*. Sistem ini menyepadukan komponen IoT, termasuk mikropengawal ESP32, pengimbas kod QR dan pangkalan data Firebase, untuk menyediakan kawalan akses yang selamat dan tanpa sentuhan. Aplikasi telefon pintar membolehkan pemilik hartanah menjana kod QR unik untuk tetamu, membolehkan daftar masuk yang lancar tanpa memerlukan interaksi fizikal. Reka bentuk sistem menekankan penyegerakan masa nyata dengan Firebase, memastikan pengesahan data yang cekap dan memberikan akses hanya kepada pengguna yang dibenarkan. Tidak seperti mekanisme kunci dan kunci tradisional atau kad RFID, pengesahan kod QR menghapuskan risiko seperti penduaan kunci, kehilangan atau perkongsian bukti kelayakan yang tidak dibenarkan. Penyelesaian ini juga menggabungkan ciri-ciri seperti kemas kini status pintu masa nyata, antara muka aplikasi mesra pengguna dan tempoh akses yang boleh disesuaikan. Projek ini menunjukkan aplikasi praktikal IoT untuk meningkatkan keselamatan dan kecekapan operasi inap desa, menangani permintaan untuk penyelesaian kos efektif dan berskala dalam industri hospitaliti. Sistem yang dicadangkan bukan sahaja menyediakan rangka kerja keselamatan yang teguh tetapi juga meningkatkan pengalaman pengguna untuk pemilik dan tetamu hartanah, membuka jalan kepada penggunaan teknologi pintar yang lebih meluas dalam pengurusan hartanah..

ACKNOWLEDGEMENTS

First and foremost, I would like to express my gratitude to my supervisor, Ma Tien Choon for his precious guidance, words of wisdom and patient throughout this project.

I am also indebted to Universiti Teknikal Malaysia Melaka (UTeM) and my mother Normala Binti Mohamad, my father Zulkifli Teoh bin Abdullah and my whole family for the financial support through PSM 1 and PSM 2 which enables me to accomplish the project. Not forgetting my fellow colleague and long time friend, Fidauddin, Amirhalizan, Irfan, Asyrafudin, Mus'ab, Iwan, Syazwan, Hamizan, Darwisy, Khaliff, Fakrullah, Faiz, Ng Chin Huan, Roobakanthan, Qayyum, Lan, Ammar, Haziq, Hafiz, Nazrul and Saifuddin for the willingness of sharing their thoughts and ideas regarding the project.

My highest appreciation goes to my parents, and family members for their love and prayer during the period of my study.

Finally, I would like to thank all the fellow colleagues and classmates, the Faculty members, as well as other individuals who are not listed here for being co-operative and helpful.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

V = Voltage angle

% = Percentage

LIST OF ABBREVIATIONS

V	-	Voltage

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Background

In the realm of residential security, ensuring the safety of occupants and safeguarding property remains a top priority. Technological advancements continue to offer innovative solutions, particularly in response to emergencies. One such innovation is the Homestay Security Code Using QR code. Traditionally, accessing a homestay or rental property involves physical keys or passcodes, which can be cumbersome to manage and potentially insecure. However, with the advent of QR code technology, a streamlined and secure access solution emerges. Much like how a semi-outdoor heat-driven fire extinguishing system enhances response times in emergencies, the Homestay Security Code Using QR code revolutionizes the way homeowners grant access to their properties. With the idea of making an application for key access using a QR code, the homeowner can generate QR code from the said application which works as a digital key. For example, if the homeowner has made a deal with the client regarding renting, the homeowner can generate a newly create QR code on their phone which later the QR code will share through any communication platform or application such as email, WhatsApp, Telegram and many more. Once the customer arrives at the homestay, he may now scan the given QR code on the scanner available at the door. This idea not only applies to the advantage of existing technology, but it also improves the security measure of the homestay.

1.2 Problem Statements

In modern times, there are emerging concerns for homestay owners about property security and the facilitation of guest access. Traditional lock-and-key systems are quite dated; the risks involve unauthorized duplication, loss of keys, and the efficiency of having to replace locks for every guest. Electronic keycards enhance the level of security yet face their own challenges like theft, loss, and expensive installation. Such systems also involve physical contact, which, during the pandemic, had been inconvenient and risky for health reasons. Homestay owners need an inexpensive, safe, and easy-to-operate solution that reduces operational hassles and allows guests frictionless access. This means there is a need for a new kind of system that provides safety with convenience for both hosts and guests.

1.3 Project Objective

The purpose is one of the most crucial things that need to be thought out and acknowledged effectively as it shall serve as a guideline in designing a project. For this project, the major purpose is to present a systematic and practical methodology to enhance security and convenience in homestays utilizing IoT technology. Specifically, the objectives are as follows:

- To create a security door sytem using QR code as a key to avoid key duplication
- Creating a homestay project that doesn't require owner and client to meet face-to-face.

1.4 Scope of Project

This project focuses on developing a smartphone app that generates QR codes as digital keys for rental properties. Homeowners can create QR codes to grant access to family, friends, or clients.The system is secure because it verifies the scanned QR code for validity to avoid

unauthorized access. It also allows for time-limited access, whereby a homeowner can set the expiration time of the QR code according to the duration of the rental.Additionally, the results of the project will contribute to the improvement of the system to maintain its relevance; therefore, it will be a reliable and secure solution for property access.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

The availability of the Internet of Things also known as IoT has help improved the standard of living in daily life in this modern era that surrounded with digital innovation. Out all of existing technology, stands security system domain has seen drastic improvement and approached of protecting residences, real estate, and also possessions. This drastic change had help increase a couple number of advanced development yet approachable solutions that suited to today's security requirments. In this case, the usage of QR code technology seems to be the most potential path of improving security protocols especially in the business such as hotel and homestay.The QR code technology is used as a key access. This development guarantee the strong security measure for both homeowner and client. Through the utilization of QR codes and Internet of Things connectivity, this novel solution has the potential to completely transform access control within the hotel industry. The goal of this research is to address contemporary security concerns by leveraging the inventive momentum of solar energy technologies, which have experienced a revolutionary evolution as recorded in previous literature. It is influenced by these swift advances in technology. The need for intelligent security system development is growing as society transitions to a digitally connected future. Living spaces will be safer, more efficient, and equipped with cutting-edge technology as a result.

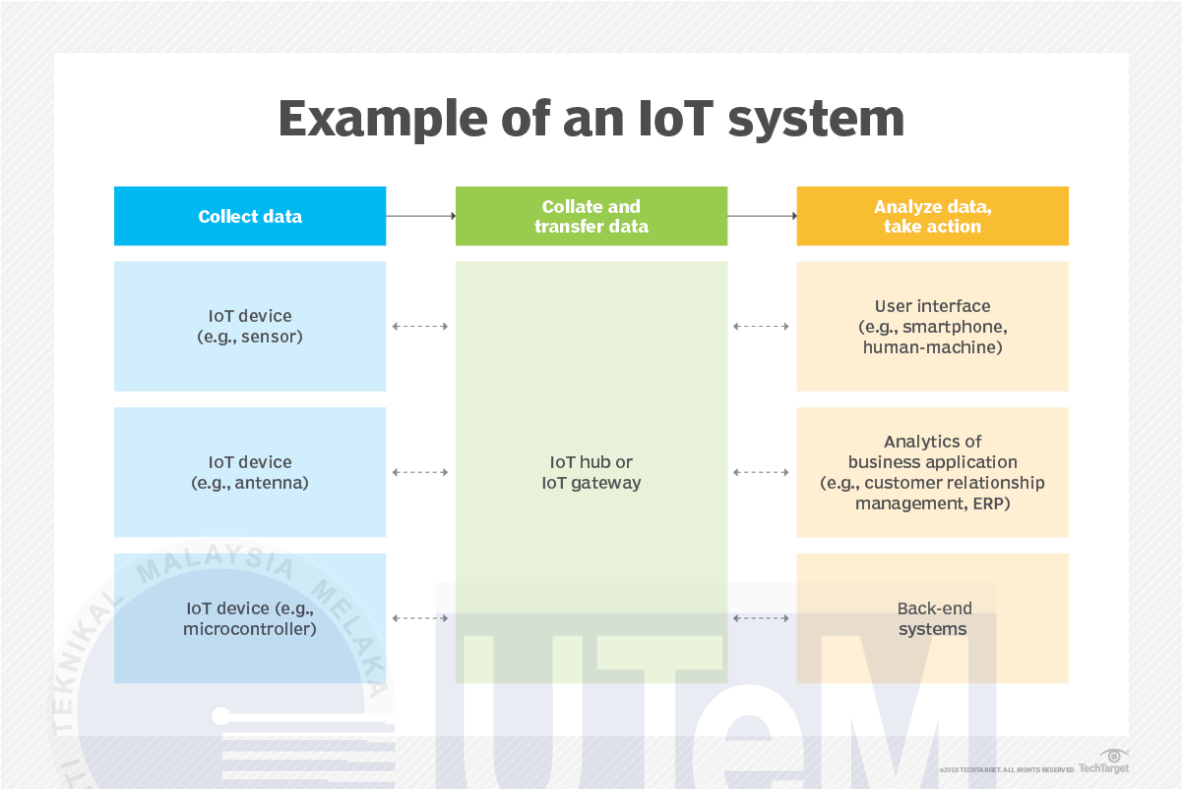

Figure 2.1.1 Example of an IoT System [1]

Figure 2.1.1 shows An IoT system collects data from sensors installed in IoT devices and transfers that data through an IoT gateway for it to be analyzed by an application or back-end system

2.2 QR Code (Quick Response Code)

Quick Response codes, or QR codes for short, are a kind of two-dimensional matrix barcode that were created in 1994 by the Japanese business Denso Wave [2]. A mobile phone can swiftly scan or recognize a QR code. Unlike a traditional bar code, which only has one data orientation, it offers information in both horizontal and vertical directions.

Figure 2.1: QR Code

Figure 2.3.1 show a pattern of QR code. With 7,089 characters for numeric data, 4,296 characters for alphanumeric data, 2,953 bytes for binary (8 bits), and 1,817 characters for Kanji / Kana symbols in Japan, a QR code can store a far bigger volume of data than an ID barcode[3].

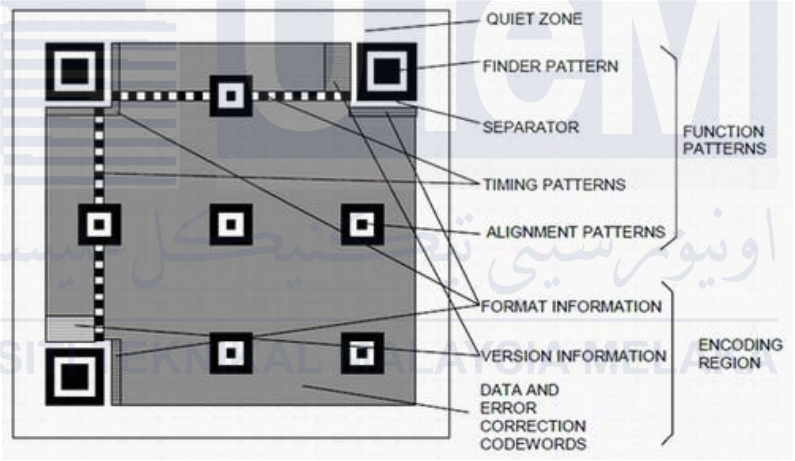

Figure 2.2: The Structure of QR Code

2.2.1 The Application of QR code in IOT System

One technology that has emerged as a beacon of safety in the constantly changing world of IoT security is QR codes. In the IoT Security Solutions sector, QR codes quickly and efficiently verify login credentials and authenticate user accounts [4] . These codes, which include network information such as SSID, password, and encryption type, also make Wi-Fi connecting easier by streamlining connections. Deployments of QR codes increased

by 23% in 2023. Furthermore, a majority of US consumers (67%)have scanned QR codes, demonstratng the broad adoption and familiarity with this technology.

2.2.2 Key Use Cases of QR Codes for IoT Security Solution

It was discovered that there are 5 key use cases of QR Codes in IoT Secuirty solution [4]. Starting from product authentication and anti-counterfeiting, secure device pairing and provisioning, maintenance tracking, access control and authentication and lastly securing firmware/software update.

2.2.3 QR Codes Safeguarding Product Authenticity and Combating Counterfeiting

By including these codes, producers give customers and companies a trustworthy way to confirm the authenticity of their items, hence lowering the possibility that fake goods will enter the market[4]. It's interesting to note that 70% of IoT device manufacturers now include QR code-based authentication in their products, indicating the industry's confidence in this technology to enhance security protocols. Moreover, research indicates that this methodology might effectively curtail instances of counterfeit goods by up to 50%, signifying a noteworthy advancement in guaranteeing the genuineness of the product.

2.2.4 QR Codes Improving Device Pairing and Provisioning

A use case centered on using QR codes to improve equipment monitoring and maintenance in IoT contexts is Optimizing Equipment Maintenance monitoring with QR Codes for IoT Security Solutions. This technique greatly enhances security standards by making maintenance schedule more precise and effective. Remarkably, QR code-based technologies have been included by 75% of industrial facilities to improve their maintenance

protocols and guarantee the safe and effective operation of their equipment[4] . Additionally, using QR codes to track maintenance has resulted in a 40% decrease in downtime, demonstrating a significant improvement in operational dependability and efficiency.

2.2.5 QR Codes contribution in Equipment Maintenance Tracking

Using QR codes to strengthen access control and authentication procedures is the main goal of the use case "Strengthening IoT Device Security: Access Control and Authentication with QR Codes for IoT Security Solutions." This application aims to make IoT devices more secure. With 85% of IoT security solution vendors utilizing QR code solutions to strengthen device security, this tactic is quickly gaining support[4] . Furthermore, implementing QR code-based authentication techniques has shown to be successful in reducing instances of unwanted access by up to 60%, thus raising the bar for IoT device security.

2.2.6 QR Codes in access control and Authentication

Safeguarding Unauthorized Changes during Firmware/Software Updates for IoT Devices: Using QR Codes for IoT Security Solutions is a use case that focuses on using QR codes to secure IoT device firmware and software update processes. This approach is extensively used; to guarantee that updates are deployed and sent safely, 90% of IoT security solution providers use QR code-based methods[4] . This strategy has been demonstrated to dramatically minimize update-related vulnerabilities by up to 70%, strengthening the IoT device security architecture as a whole.

2.2.7 QR Codes ensuring Secure Firmware/Software Updates

By streamlining the procedure, this method makes it simpler to oversee every aspect remotely. Use QR codes to unlock the potential of indoor farming technology's remote monitoring, allowing for easy supervision and increased output. Through resource allocation optimization and downtime minimization, QR code-enabled remote monitoring may yield an astounding 80% boost in operational efficiency[4] .

2.3 Microcontroller

A microcontroller is a small yet powerful computer system with integrated circuit that contains a processor core, memory, and input/output peripherals[5] . It serves as the brain of embedded systems, capable of executing programmed instructions to control various electronic devices and systems. Microcontrollers are commonly used in a wide range of applications, from consumer electronics to industrial automation, due to their compact size, low cost, and versatility. Microcontroller plays a crucial role in implementing the intelligent security lock key system.

2.3.1 Hardware

2.3.1.1 ESP 32 Microcontroller

The ESP32 is a powerful microcontroller that has gained widespread popularity in the IoT and embedded systems community due to its advanced features and capabilities [6], [7]. Developed by Espressif Systems, the ESP32 is based on a dual-core Xtensa LX6 processor with integrated Wi-Fi and Bluetooth connectivity, making it well-suited for a variety of applications requiring wireless communication and internet connectivity.

Figure 2.3: ESP 32

2.3.1.2 Arduino Uno Microcontroller

The Arduino Uno is an open-source microcontroller board based on the Microchip ATmega328P microcontroller (MCU) and developed by Arduino.cc and initially released in 2010 [8]. The microcontroller board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The board has 14 digital I/O pins (six capable of PWM output), 6 analog I/O pins, and is programmable with the Arduino IDE (Integrated Development Environment), via a type B USB cable[9]. It can be powered by a USB cable or a barrel connector that accepts voltages between 7 and 20 volts, such as a rectangular 9-volt battery.

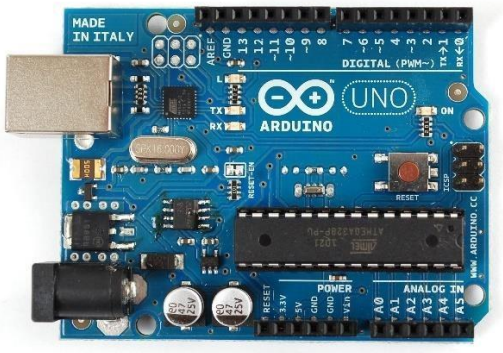

Figure 2.4: Arduino Uno[10]

2.3.1.3 Raspberry Pi Controller

The Raspberry Pi is a small card sized computer, function almost as similar as the regular computer people know, which run on Linux operating system. The Raspberry Pi has a borad com BCM 2835 SoC, which comprises of an advanced RISC Machine 76JZF-S 700 MHz processor, video core IV GPU, and was originally distributed with 256 megabytes of RAM, and which the upgraded version to 512 MB (Model B & Model B+) [11]. It doesn't come with a hard disk or solid-state drive, but it uses an SD card for booting and data storage.

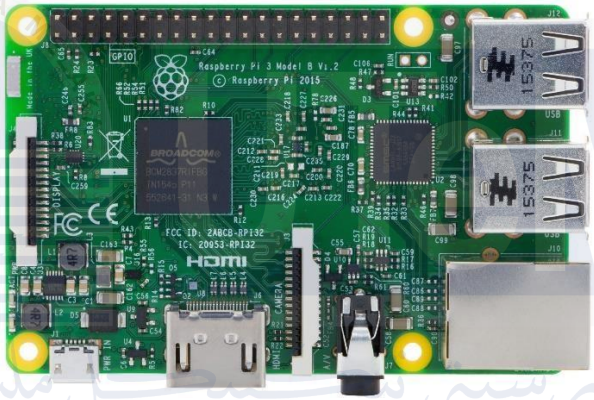

Figure 2.5: Raspberry Pi

2.3.2 Microcontroller Software

2.3.2.1 Arduino IDE

Arduino IDE is an open-source platform for prototyping built on user-friendly hardware and software [12]. Arduino boards have the ability to take inputs, such as a light from a sensor, a finger pressing a button, or a message from Twitter, and convert them into outputs, such as starting a motor, turning on an LED, or posting content to the internet. By delivering a set of instructions to the microcontroller on the board, you can instruct your board on what to do. You use the Arduino Software (IDE), based on Processing, and the

Arduino programming language, which is based on Wiring, to accomplish this. ESP 32 is also compatible with Arduino IDE.

Figure 2.6: The Arduino IDE logo [12]

Figure 2.7: Arduino IDE Interface

2.3.2.2 Raspberry Pi

The Raspberry Pi software ecosystem is diverse and robust, facilitating a broad range of uses, including professional growth, DIY projects, teaching, and prototyping [13].Raspberry Pi is quite different compared to Arduino and ESP 32 as it development environment is wide. Several type of IDE's that can execute the Raspberry Pi is Thonny, Geanny, Visual Studio Code, and PyCharm .

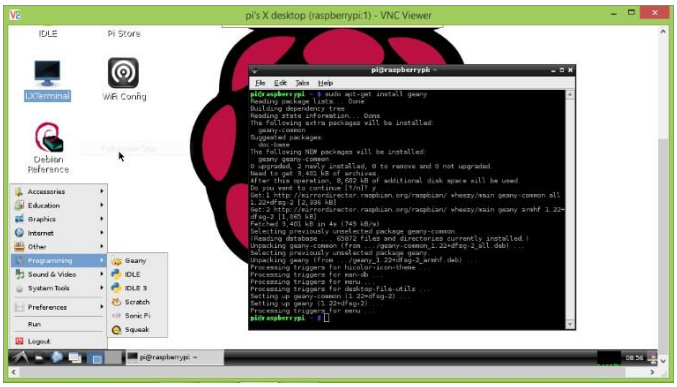

Figure 2.8: Running Raspberry Pi program using Geany[14]

2.3.3 Comparison between microcontrollers

Table 2.1 The comparison of Arduino Uno, ESP 32 and Raspberry Pi

Parameter	Arduino Uno (R1 to R3)	ESP 32	Raspberry Pi
Microcontroller/ Processor	Microchip Atmega328P	Tensilica Xtensa LX6 dual-core	Broadcom BCM2711, Quad-core Cortex-A72
Operating Voltage	5V	3.3V	5V
InputVoltage (Recommended)	7-12V	5V (via USB)	5V/3A DC via USB-C
Digital I/O Pins	14 (6 PWM)	34	40

Analog Input Pins	14	18	Requires additional hardware
PWM Channels	6	16	Yes
Flash Memroy	32 KB	4 MB	Depends on MicroSD card
SRAM	2 KB	520 KB	2GB, 4GB, or 8GB LPDDR4
EEPROM	1 KB	-	-
Clock Speed	16 MHz	160 MHz (up to 240 MHz)	1.5 GHz

Wi-Fi	No	802.11 b/g/n	2.4Ghz and 5GHz IEEE 802.11 b/g/n/ac
Bluetooth	No	V4.2 BR/EDR and BLE	Bluetooth 5.0, BLE
USB ports	1 x USB 2.0	Varies by board	2 x USB 3.0, 2 x USB 2.0
Networking	No	Wi-Fi, Bluetooth	Gigabit Ethernet, Wi-Fi, Bluetooth
GPIO	14	34	40
Video Output	No	No	2 x micro-HDMI (up to 4Kp60)

Audio Output	No	No	4-pole stereo audio and composite video port
Power Consumption	Low	Medium	High
Dimensions	68.6 mm x 53.4 mm	Varies by board (e.g, 18 mm x 25.5 mm)	85.6 mm x 56.5 mm
Weight	25 g	Varies by board	46 g

2.4 Past project

2.4.1 "Automated Barrier Gate for Housing Estate Security System Using QR Code Based on Android Application"

Based on the [15], the authors conduct research on automatic barrier gates that use an Android-based QR Code that is only accessible by the housing unit's homeowners in order to enhance the security system. Subsequently, the portal will be connected to the QR Code on the Android app and the database's current data. Every enrolled member of the home owner's account will generate a unique QR Code, which will be distinct from one another. The ATmega2S60, which is used by the portal control system center, receives data from the QR Code scanner and compares it with the database. By employing the absolute difference method, the system checks the visitors with data matching. The GM66 QR Code Scanner's functionality test result shows an error value of 7.14% and a system running time of 13.6 seconds overall.

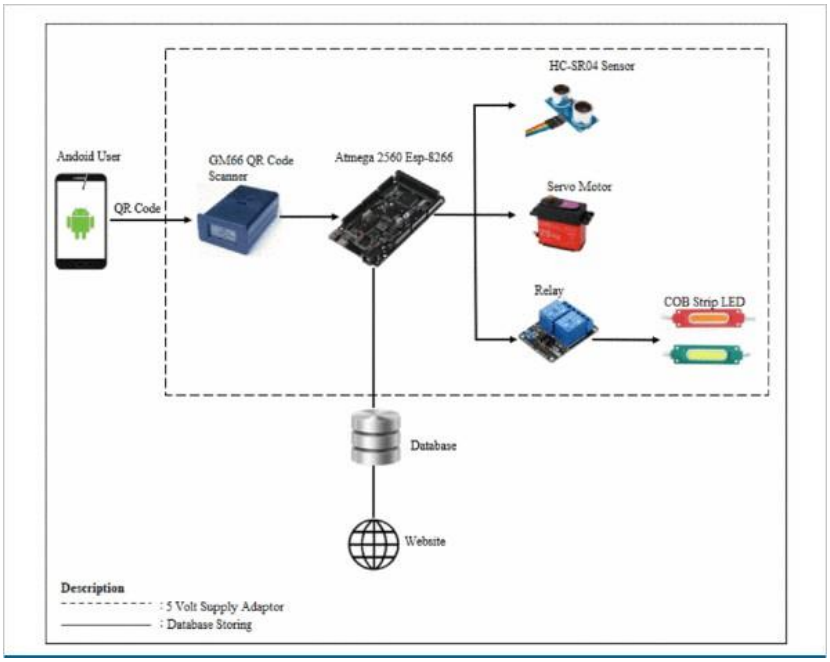

Figure 2.9 System Architecture and Design

2.4.2 Development of Web-Based Smart Security Door Using QR Code System

The work of this author [16] This work demonstrates the construction of a secure door lock system that allows authorized personnel to monitor who has access to the university laboratory or classroom. The system makes use of Quick Response (QR) technology and a Raspberry Pi processor. The purpose of the data recording system is to monitor all incoming and outgoing activity that occurs when a user logs in to the security door from the web server. In a similar spirit, the only people who will be granted entry to the doors are the approved ones. This is preliminary work to test the functionality of the system for usage in various assets and facilities, including offices, labs, and classrooms. The testbed development has been successful as an early research to give a smart security door employing QR code technology, based on the results and data collecting.

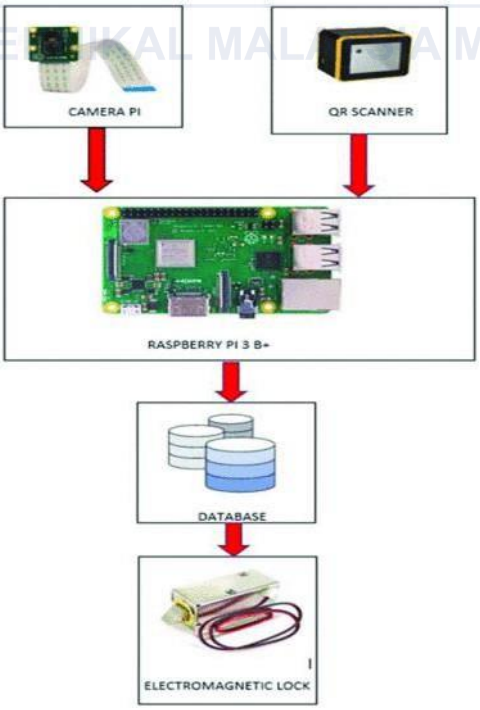

Figure 2.10: The component used for Development of Web-Based Smart Security Door Using QR Code System

2.4.3 Smart door access control system based on QR Code

This work uses Python, Arduino, and QR code technology to build an automated access control system that uses codes. The QR scanner at the entry gathers and compares the user's unique identification (UID) with the UID entered into the system after it scans a QR code [17] . The outcomes demonstrate that this system is able to promptly, effectively, and dependably give or prohibit entry to a secured area. Security systems keep unauthorized people out of a place, protecting both intellectual and physical property. Many door locks, including electronic and mechanical ones, were developed to satisfy fundamental security requirements, but they also aid in organizing data files listing the individuals who are permitted.

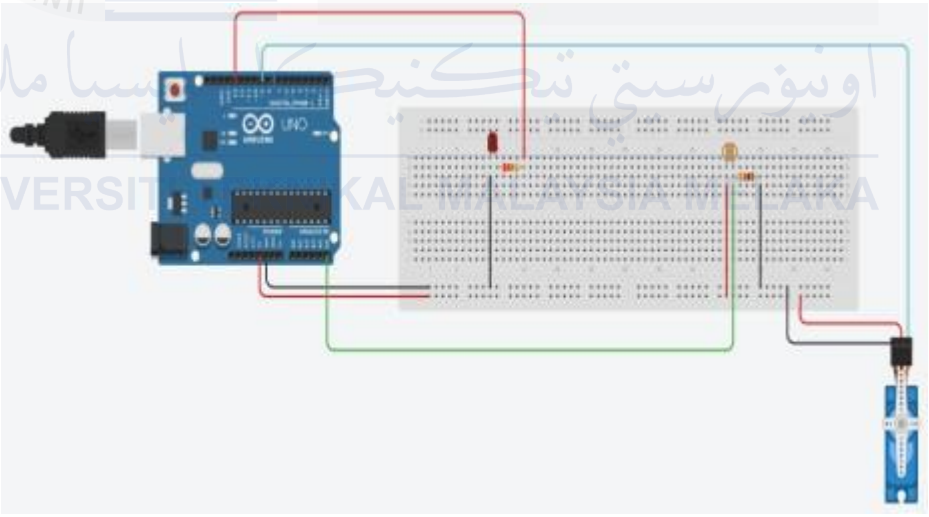

Figure 2.11: The simulation of a project with Arduino Uno

2.4.4 QR Code Based Door Opening System

The writer proposed a project for accessing a gate using QR code. The writer also mentioned that in his home country, India has a huge population[18]. By having a huge population India faced a problem in traffic congression. The writer mentioned, by designing a control system for traffic by utilizing IoT idea through QR code which generated in the webpage for the respective person can help solved the congestion problem.

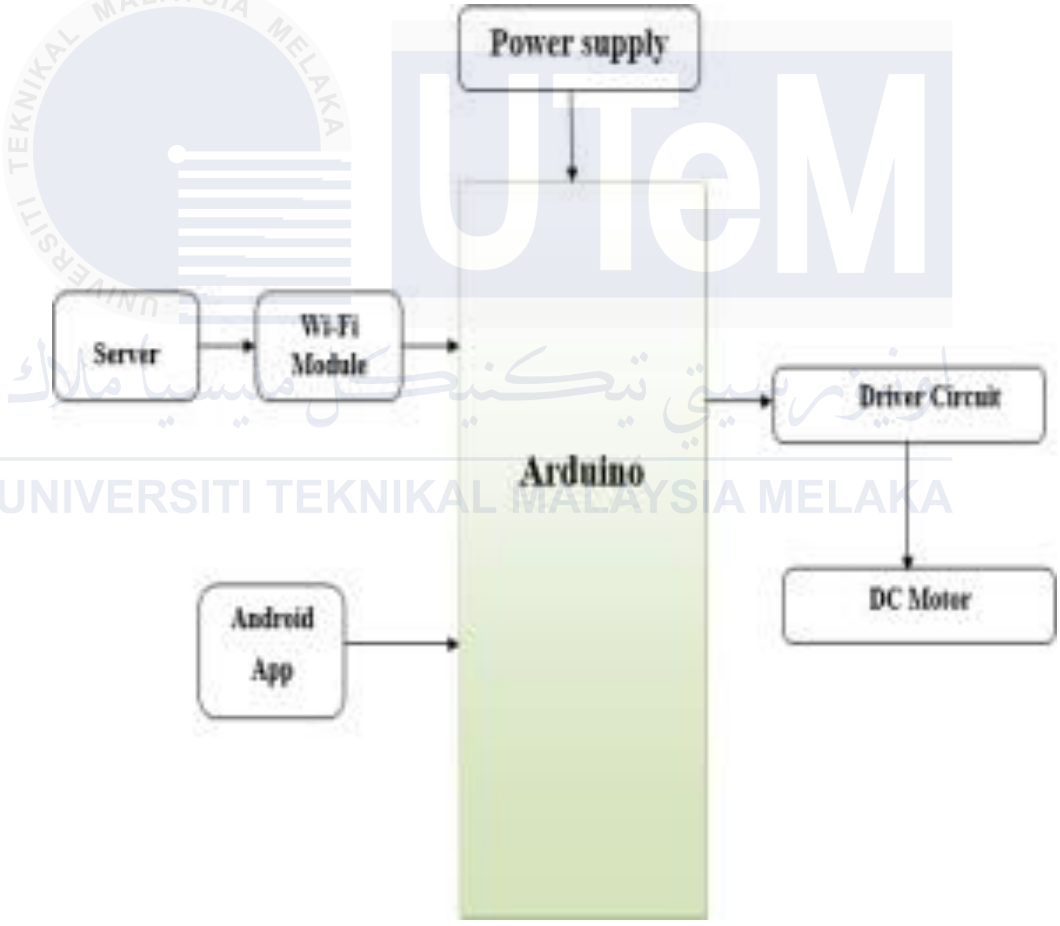

Figure 2.12: Block Diagram of QR Code Based Door Opening System

2.4.5 Design and development of smart lock system based QR Code for library's locker at Faculty of Engineering

The writer of this project designed of an intelligent lock system in the Library of the Faculty of Engineering, Universitas Riau, that uses QR code scanning as a locker key is covered in this paper [19] . The device system comprises a single ESP8266 Node MCU module, a single 12V Power Switch Adapter, a single AMS 1117 module with 3.3V and 5V output voltages, three 5V relay parts, and three 12V solenoid components. In comparison to the current security systems, the security system has been upgraded. Features like email notifications, level login to treat theft notifications, and personal data verification have all been added. This study also examined the device system's delay. The average delay obtained, based on measurement data, is 1.66 seconds.

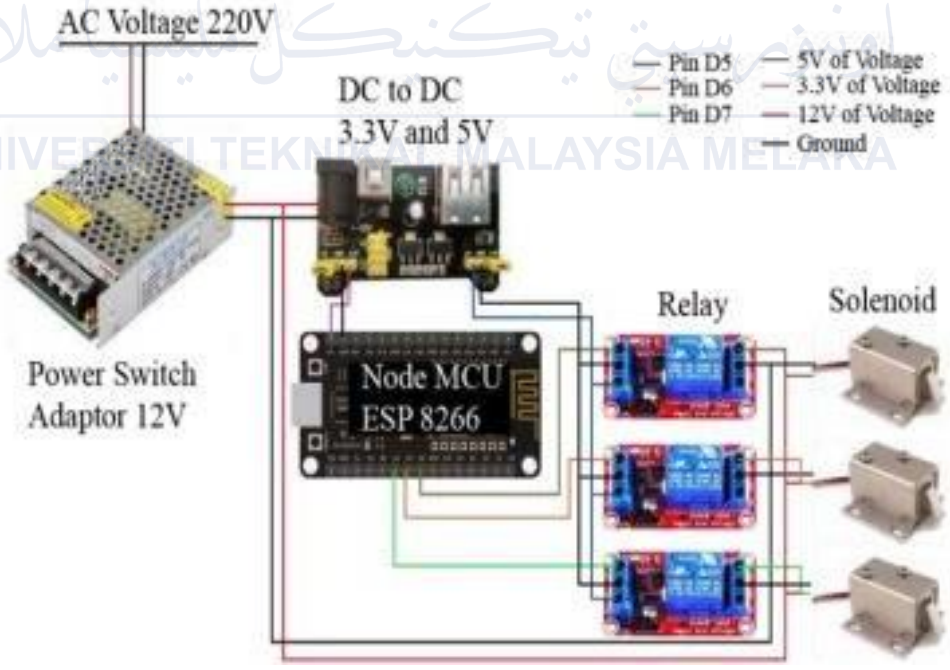

Figure 2.13: The design of smart lock system based QR Code for library's locker at Faculty of Engineering

2.4.6 QR Code DOOR Project: Access Control Application using QR Code Image

The research suggests a new approach to access management that involves attaching a WebCam to an electric door lock and developing an embedded key for smartphones that uses two different forms of cryptography to create a QR Code image [20]. This paper presents the architectural model of the access device, the encryption procedure, the elaboration of the QR Code reading device using a Raspberry Pi 2 CPU, and the generation of a QR Code image using encrypted users' data.

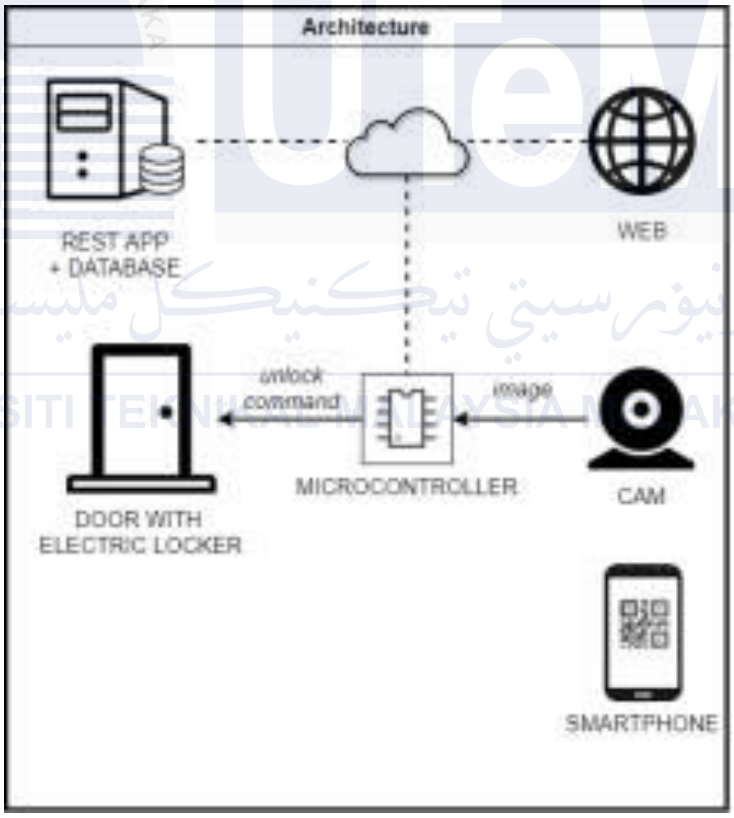

Figure 2.14: The architecture QR Code DOOR Project

2.4.7 Door Lock Security System Using Raspberry Pi & QR Code

The writer [21] stated that the device designed in this project can be installed in the main entrance of a house. With the use of a USB webcam and a PIR sensor, it can detect any movement from a visitor and begin taking pictures.The photographs are uploaded to the Google Cloud and sent as an email notice to the home owner after being momentarily stored on the Raspberry Pi. Thus, the user receives the visitor's photographs by email right away, which he can view on his smartphone. Through the TCP-IP stack, the Raspberry Pi and Google Cloud are connected. One of the IoT boards that has an on-board TCP/IP stack is the Raspberry Pi 3, which makes connecting to an IoT network easy. The Pi utilizes the OpenCV library to take pictures with the webcam and transmit them to the user's registered email account.

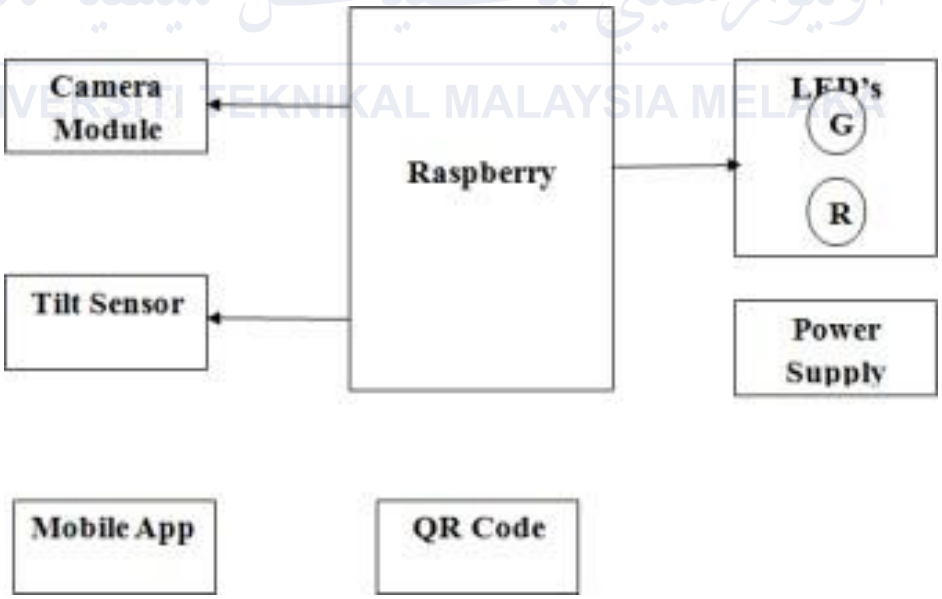

Figure 2.15: The Block diagram Door Lock Security System using Raspberry Pi & QR code

2.5 Project Comparison

Table 2.2 The comparison of project of the past

NO	Author	Project Name	Microcontroller/processor Used	Additional component	Output
1.	F. Istiqomah, D. K. Nuurul Izza, J. Susila, B. A. Kindhi, E. Indasyah and F. I. Adhim	Automated Barrier Gate for Housing Estate Security System Using QR Code Based on Android Application	Atmega 2560 ESP-8266	GM66 QR Code Scanner, HC-SR04 Sensor, Servo motor, relay and COB Strip LED	- The Red LED will lights up if the system reject the QR code, Green LED will light up if the QR code valid.
2.	A. F. M. Fauzi, N. N. Mohamed, H.	Development of Web-Based Smart Security Door Using QR Code System	Raspberry Pi 3 B+	Camera Pi, QR scanner,Electromagnetic Lock	- The Lock will activate and unlock once it receive the valid QR code

	Hashim and M. A. Saleh				
3.	Jain, Agrim & Panwar, Abhinav & Azam, Mohd & Khanam, Ruqaiya	Smart door access control system based on QR code	Arduino UNO	Servo motor, QR code Scanner, Light dependent resistor (LDR), Light emitting diode (LED)	- The voltage value changing depending on the brightness of surrounding light - Servo motor rotates to 90 degree to left once it receeive the correct QR code.
4.	Nethrasri P, Nethrasri P	QR Code Based Door Opening System	Arduino UNO	ESP8266EX, L293D (moto driver),DC motor	- When the QR code being scanned match that of the server, the DC motor will run thus unlocking the door

5.	Yusnita Rahayu, Luthfi Afif , Ping Jack Soh	Design and development of smart lock system based QR Code for library's locker at Faculty of Engineering, Universitas Riau	Node MCU ESP 8266	AC voltage 220V, DC-DCstep down (module AMS1117) with output of 3.3V, 5V relay,12V Solenoid	- The selectinve QR code will triggered a specific solenoid given there are three of them. The solenoid will activate the indicating that the door has been unlock
6.	Luis Antonio Pereira	QRCode DOOR Project: Access Control Application using QR Code Image	Raspberry Pi 2	LED, Digital camera with 5 megapixels Resolution connected via USB	- LED will lights on indicating that the door has been open
7.	Dr. Badugu Suresh	Door Lock Security System Using Raspberry Pi & QR Code	Raspberry Pi Camera	Tilt sensor, Red LED, Green LED	- Red LED light on when QR being scanned is rejected, Green Light on

					when QR being scanned accepted.

2.6 Summary

Most of the completed project, been done by referring to projects of the past. Reviewing the past project is very important as a lesson can be learnt from past attempts, which may spark the development of fresh concepts or improvements for later undertakings. Even though every project is different and has its own features and designs, it is important to recognize and value the contributions of earlier work because cooperation and the exchange of knowledge frequently lead to success. To further foresee and lessen potential roadblocks in the development process, in-depth study and comprehension of pertinent theories, concepts, ideas, and technologies are essential. This thorough evaluation guarantees that the project draws on prior expertise and understanding to produce a reliable and efficient security solution designed for homestay settings.

CHAPTER 3

METHODOLOGY

3.1 Introduction

In general, accuracy and effectiveness are among the most crucial elements in developing a project, especially one that involves security and access control. To address these issues and improve the general security and convenience of the homestay environment, the project that is executed must include all necessary characteristics. The on-going evaluation made toward the project being build help ensure that the system, database, hardware, and part produce is suitable and match a certain standards. The information need to be absord quickly, program the continuing duty, and continue to funciton for certain amount of time. Each stage is important in order to build and implement an advanced yet user-frinedly system that consisting the usage of QR code that based on IoT technology specialized for homestay.

3.2 Selecting and Evaluating Tools for a Sustainable Development

A careful tool selection is necessary for the integration of IoT technologies and QR codes in the building of a security lock key system for homestays. While advanced tools have advantages, they can also add complexity and waste resources if features are mismatched. On the other hand, excessively basic tools could require regular updates, which could cause downtime. Finding the right balance in tool selection is essential, ensuring that functionality is neither compromised or overshadowed. Every tool is carefully examined for

support for QR codes, IoT integration, and secure data transmission. Scalability and compatibility are crucial. The project minimizes maintenance while ensuring optimal performance through thorough tool selection and evaluation. This strategy ensures that homestays will have a solid, safe, and effective security solution that lasts over time.

3.3 Methodology

This documentation outlines an innovative approach to improving homestay security by using an IoT-based smart lock system that uses QR codes. The goal of this system is to offer safe, contactless entry control, guaranteeing the comfort and safety of both visitors and property owners. The procedure for registering new visitors and how the system validates QR codes before granting access are shown in the flowchart and comprehensive procedures that follow.

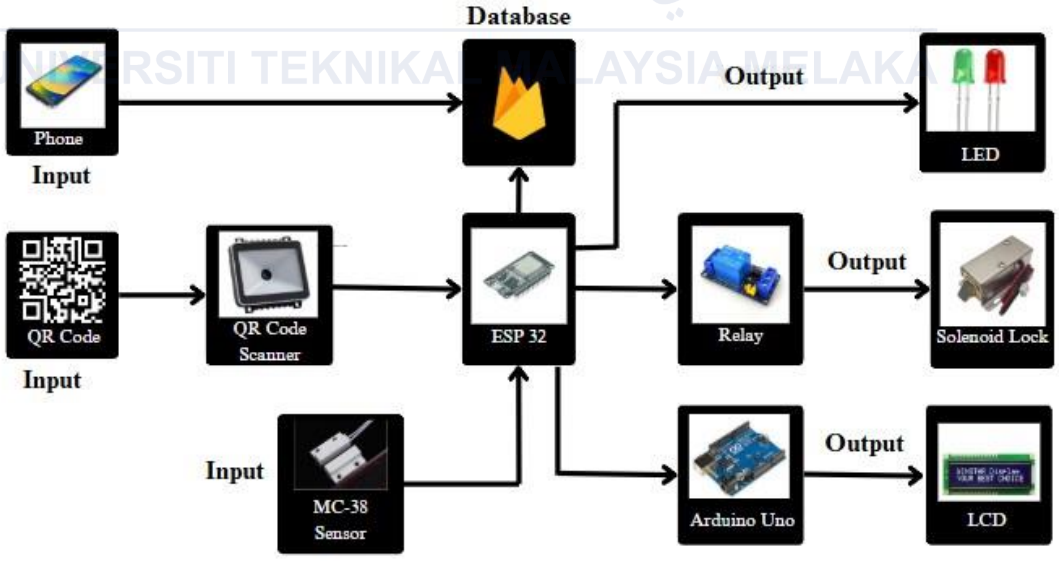

Figure 3.1: Block Diagram of the project

The block diagram above shows a QR code security door system using ESP32, Arduino Uno microcontroller. The system include QR code scanner to decode data from

scanned QR code and MC-38 sensor to indicate if the door is open or close. These component help collect and send data to ESP32 to interact with database and will activate the LED, relay to trigger solenoid and Arduino Uno to help LCD display the output in real-time. The Arduino Uno is used as a memory extension due to the limitation of ESP32 flash memory to help LCD functioning.

3.3.1 Workflow

This system uses QR codes for secure access control. The process starts with the homeowner generating a QR code through a smartphone application. The code is saved on a database server for validation. When the QR code is scanned, the scanner sends the data to a microcontroller, which forwards it to the server to check its validity. If the server confirms the QR code is valid, the microcontroller activates a relay that unlocks the solenoid lock, granting access. If the QR code is invalid, the door remains locked. This method provides secure, automated access control, ensuring only authorized users can unlock the door.

3.4 Flowchart

The flowchart will outline the development process. This ensures that the project is organized and executed according to the plan. The flowchart will demonstrate how the experiment was carried out and highlight significant features.

3.4.1 Project Flowchart

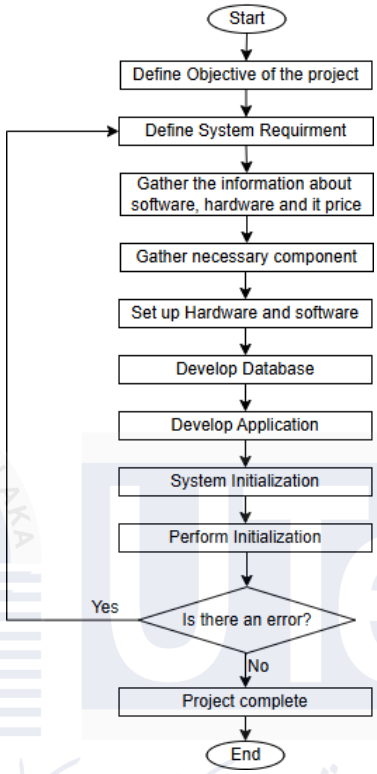

Figure 3.2: Project flowchart

Figure 3.2 shows the flowchart on how this project were build. Starting from understanding the objective to perform initialization. If there's an error, back to the defining system requirement until there's no error. Only then the project is consider complete.

3.4.2 Application Flowchart

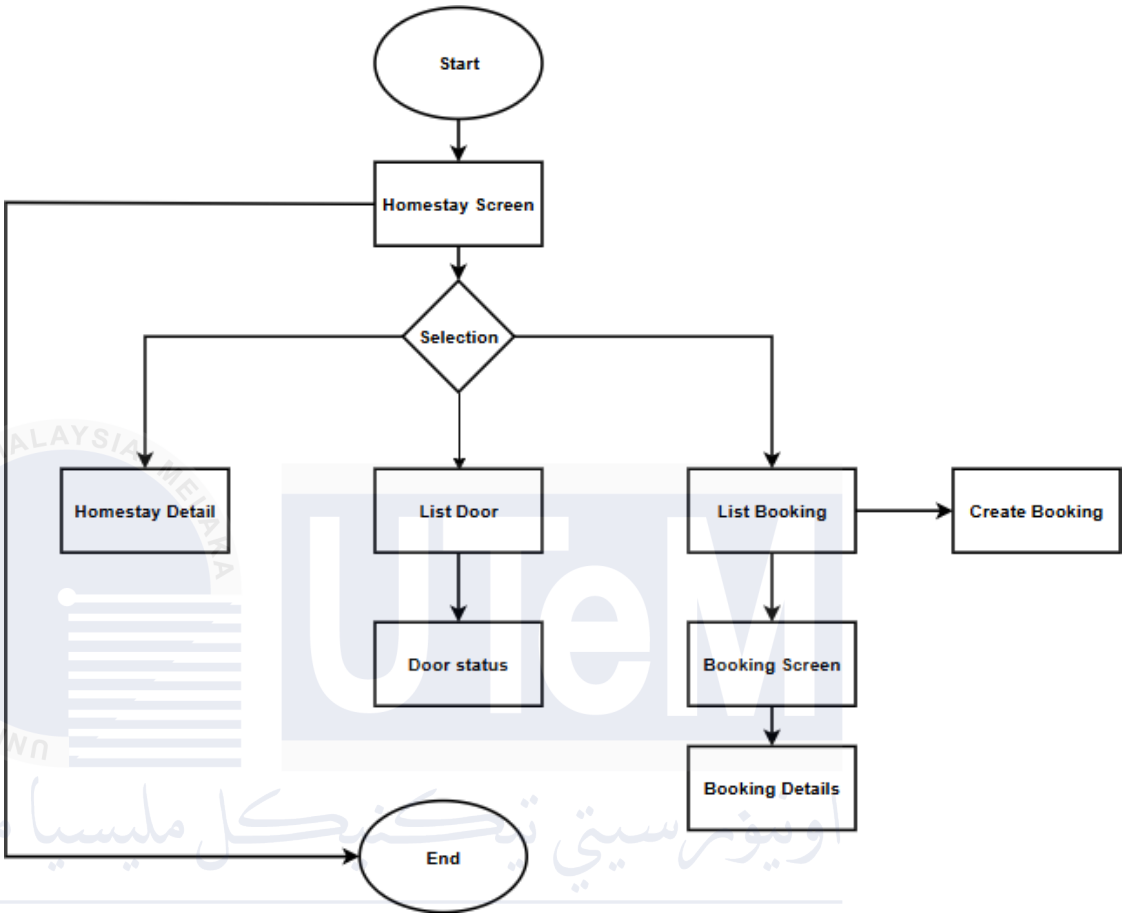

Figure 3.3: The application flowchart

On Figure 3.3 shows the UI (User Interface) of the application on the User phone. On the selection part, there are 3 options which is Homestay detail, List door and list booking. The homestay detail basically contain the detail of the address such as address. For List door, it shows a status of door either it is locked or not. For list booking, it contain all the detail regarding the booking such as creating a booking for new customer or existing one. The homeowner can also terminate the existing QR code as well as share the QR code up to his liking.

3.4.3 System Flowchart

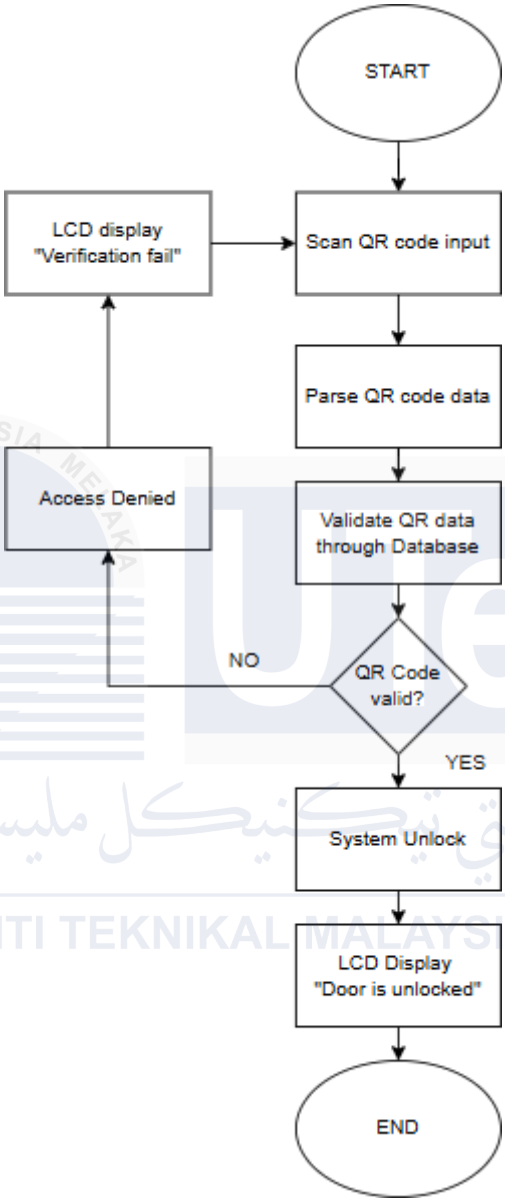

Figure 3.4: System Flowchart

Figure 3.4 shows the flowchard of the system. It shows that when QR code is scanned, the system will decode QR code data and will parse the data through database to server to check check in-check out timing and it validity. If the QR code is valid, then the door will be open. Otherwise, the LCD will display that the QR code is invalid and the door will remained lock.

3.5 Experimental Setup

The ESP32 is the main microcontroller utilized in this project. Its functions include processing data from QR codes, interacting with the database, and managing system operations. The ESP32 can process data entries from the QR code scanner, record them in real-time, and communicate with the database to validate the scanned codes thanks to algorithm codes that can be implemented on the Arduino IDE.

A Mini QR code scanner module is used for scanning QR codes. This module sends the data to the ESP32 for processing after scanning the QR codes that visitors present. After scanning, the device obtains the necessary information from Firebase, a real-time cloud database that is used to hold visitor data and the QR codes that correspond with it. Firebase is a database server that help to communicate and update with the ESP 32 for confirmation and matching the QR code so the system may or may not grant acsess.

A solenoid locks is the lock mechanism used in and work it function in accordance to the system outcomes. The Solenoid door will be activate thus unlok the door once it is confirm by the system that the scanned QR code is valid. The door will simply remain as it is if the QR code scanned is invalid.

The software used in this project consist of Arduino IDE to programme the ESP 32 while syncing it to the real-time connection with Firebase. Furthermore, an application that was develop by Android Studio that help homeowner generate QR codes which will be shared to the guest. This application also have a real-time connection with Firebase and the data saved will be accessed by the ESP 32 to grant access of the validity of QR code. The QR code ensure the integration between mobile platform and the IoT system.

3.6 Equipment Use

3.6.1 ESP 32

Because of its built-in Wi-Fi and Bluetooth capabilities, the ESP32 microcontroller is perfect for an Internet of Things (IoT)-based security lock door system that uses QR codes for homestays. These features allow for seamless real-time interaction with cloud databases like Firebase. It can handle duties like data processing and QR code scanning with efficiency because to its dual-core processor and plenty of memory. It is reasonable to use the ESP32 to create a dependable, safe, and easy-to-use access control system for homestays because of its affordable price and sophisticated capabilities.

Figure 3.5: ESP 32

3.6.2 Arduino Uno

Arduino Uno Boards are based on the ATmega328P chip, designed for simplicity, ease of use, and versatility in various electronic projects. As part of the open-source Arduino platform, the boards are equipped with 14 digital input/output pins, six analog inputs, a USB connection for programming, and a power jack for an external power supply. The Arduino Uno Boards operate at a clock speed of 16MHz, making it suitable for handling various sensors, motors, LEDs, and communication modules. The Arduino Uno supports digital and analog interfacing, enabling applications ranging from basic LED blinking to complex

robotics and Internet of Things (IoT) systems. Its simplicity and broad community support make it a favored choice among hobbyists, educators, and professionals for prototyping and developing embedded systems, interactive devices, automation solutions, and more.

Figure 3.6: Arduino Uno

3.6.3 Mini QR code scanner

QR scanners are highly suitable for Internet of Things (IoT)-based security lock systems for guest houses due to their high-speed decoding, compact size, and durability. Their compatibility with various interfaces, such as USB, UART, and others, simplifies integration with microcontrollers like the ESP32. Additionally, their low power consumption ensures reduced maintenance requirements and extended operational life. These characteristics make QR scanners an ideal choice for reliable and efficient access control in homestay and similar settings, offering a seamless and secure experience for users.

Figure 3.7: Mini QR scanner

3.6.4 MC-38 Door Sensor

MC-38 Magnetic Contact Switch Sensor can be used as a door or window security system. It produces a signal when moved away from each other which can be fed to the microcontroller (e.g Arduino) to perform the desired action as per requirement. This sensor is suitable to use for trigger alarm or ON/OFF light inside a cupboard sliding door. This wired sensor is triggered by the magnet. When the magnet is closed by, the circuit is closed or open if the magnet is far from the sensor.

Figure 3.8: MC-38

3.6.5 Relay

In this project, the high-power door locking mechanism is controlled by the relay. By acting as a switch, it enables the low-power ESP32 to securely regulate the higher voltage/current required for door locking and unlocking. As shown in Figure 3.7.

Figure 3.9: Relay

3.6.6 Solenoid Lock

A solenoid lock is ideal for this project due to its reliability, ease of integration, and secure locking mechanism. When powered, it offers strong, rapid locking and unlocking, in line with the system's requirement for quick access control. Smooth integration is ensured by the solenoid lock's simple operation and compatibility with microcontrollers such as the ESP32. Furthermore, it is appropriate for the frequent use anticipated in homestays due to its longevity and low maintenance requirements, which improves overall security and user convenience. As shown in Figure 3.8.

Figure 3.10: Solenoid Lock

3.6.7 LCD 16 x 2

A common kind of display module used in many electronic projects and embedded systems is a 16x2 LCD (Liquid Crystal Display). The display can show up to 32 characters at a time since it contains 16 columns and 2 rows, as indicated by the "16x2" designation. In this project, this is used to show the output whenever the QR code scanned to indicate if the QR code received is valid or invalid. As shown in Figure 3.9

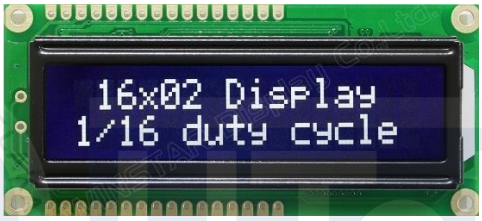

Figure 3.11: LCD 16x2

3.7 Software Use

3.7.1 Android Studio

An extensive set of tools designed especially for Android app development is provided by Android Studio. It offers a feature-rich Integrated Development Environment (IDE) with tools for debugging, testing, and code editing that are all tailored to make Android applications run smoothly. Android Studio benefits from a vast developer community and extensive documentation. This ecosystem provides access to libraries, APIs, and support forums, facilitating faster development cycles and easier troubleshooting. Developers can leverage these resources to implement advanced features, ensure app stability, and optimize performance for the specific requirements of a homestay security system.

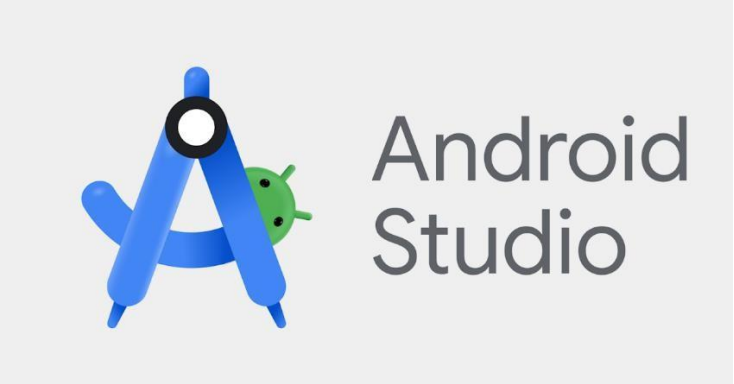

Figure 3.12: Android studio logo

3.7.2 Flutter

Flutter is Google's portable UI toolkit for crafting beautiful, natively compiled applications for mobile, web, and desktop from a single codebase. Flutter works with existing code, is used by developers and organizations around the world, and is free and open source. Flutter also use a coding language called Dart language. Though it name is quiet unknown to most, it easy to learn. You can build apps with Flutter using any text editor or integrated development environment (IDE) combined with Flutter's command-line tools. The Flutter team recommends using an editor that supports a Flutter extension or plugin, like VS Code and Android Studio. These plugins provide code completion, syntax highlighting, widget editing assists, run & debug support, and more. The Flutter plugin only works with Android Studio.

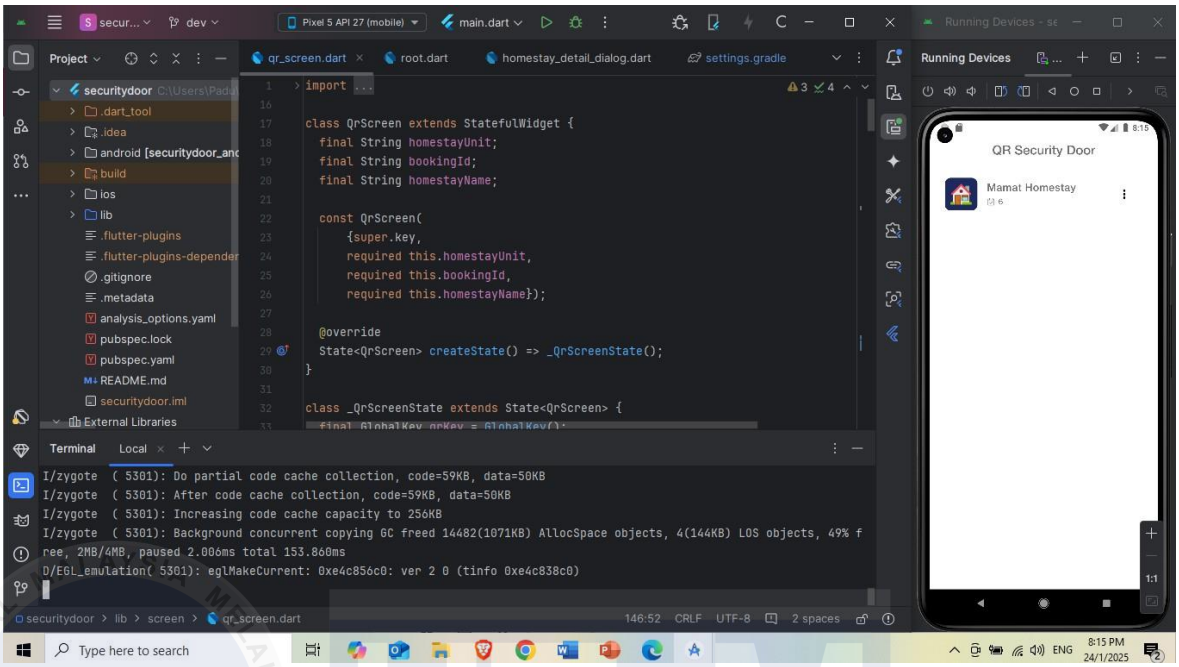

Figure 3.13: Fluter on Android Studio

3.7.3 Arduino IDE

Arduino (IDE) is an application for the platform (Windows, macOS, Linux) written in the Java programming language. It is used to write and upload programs to Arduino boards. The source code for the IDE is released under the GNU General Public License, version 2. Arduino supports the C and C++ languages using special rules for structuring code. Arduino provides software libraries from the Wiring project, which offer many common input and output procedures. User-written code only requires two basic functions to initialize the sketch and the main program loop. These are compiled and linked with the main program included in the executable cycle, which can be executed with GNU tools, also included with the IDE distribution.

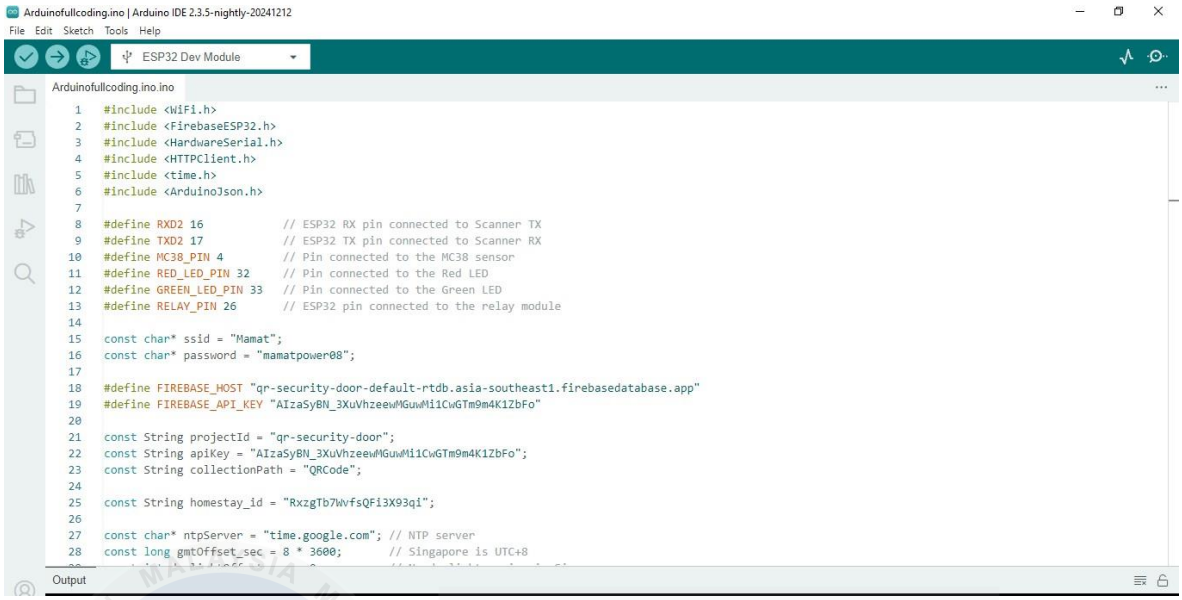

Figure 3.14: Coding in Arduino IDE

3.7.4 Firebase

Firebase is a great choice for the database server because of its scalability, real-time capabilities, and ease of integration. With Firebase's real-time synchronization feature, devices may instantly get updates anytime new QR codes are created or viewed. This feature improves responsiveness and security by ensuring that decisions about access control are rapidly relayed to the security system. The mobile application was developed using Android Studio, which is one of the platforms and programming environments that Firebase seamlessly integrates with. Its simple integration SDKs and APIs cut down on development time and effort by making the deployment of data store and retrieval operations easier. The QR code that been produce by the android studio application will directly link up with Firebase sauthentication and access control mechanisms, ensuring that guest QR code data remains protected from unauthorized access.

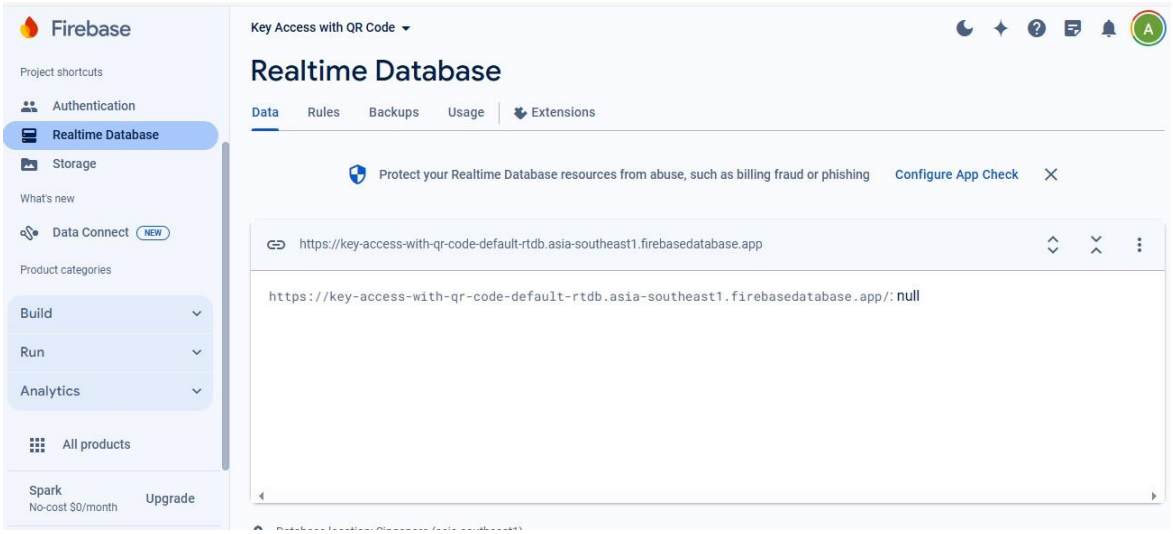

Figure 3.15: The interface of Firebase in the website

3.8 Summary

This chapter outlines the methodology adopted for the development of an IoT-based security lock door system using QR codes specialized for homestays. It describes how to choose and integrate key parts such the Firebase database, Mini QR code scanner, and ESP32 microcontroller. The process entails analyzing system requirements, choosing components based on compatibility and performance standards, and creating a comprehensive system design to guarantee smooth functionality and integration. Limitations are also covered, including the need for consistent internet connectivity and possible problems with QR code readability. Through effective access control based on QR codes, the initiative seeks to improve homestay security while guaranteeing user-friendliness and dependability for both hosts and visitors.

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Introduction

This chapter presents the results, analysis, and evaluation of the Development Of Intelligent Based On IoT For Security Lock Key using QR Code Specialize For Homestay. The system integrates components such as the ESP32 and Arduino Uno microcontroller, Mini QR code scanner, and Firebase database to facilitate secure access control through QR code verification. The system's performance is assessed in terms of its capacity to scan QR codes quickly, the dependability of its access control systems, and its interface with Firebase for real-time data synchronization. The usefulness of the system in boosting homestay security is discussed, along with any implementation-related issues and suggestions for enhancements for further editions. The outcomes demonstrate how the system can manage guest and host access conveniently and securely, all the while maintaining dependability and user-friendliness.

4.2 Result and Analysis

IoT-based QR code security system projects have great potential for changing the face of access management in property. This therefore proposed a project that embedded the QR code system within a homestay management application that could allow homeowners to create QR codes for secure and easy access. These are actually digital keys whose idea prevents the use of real keys or even electronic keycards, which might be costly and less practical.The system adopts the policy of security and verifies the QR code through the Firebase database. Provided the QR code matches against the booking details and is within the valid check-in and check-out times, the system will unlock the door through an IoT-connected solenoid lock. Advanced IoT technologies ensure reliability in real-time monitoring and control for enhanced convenience and security to users.The project deals with some of the major challenges in managing access to homestays by automating the access, including remote control, timestamp validation, and database synchronization. This ensures a more scalable, user-friendly, and secure solution to the changing needs of property owners and their guests.

4.2.1 Initial Hardware

By referring Figure 4.1 below, it shows hardware configuration required initially consists of an ESP32 module, Arduino Uno, relay module, a pair of 9V battery for powering other Solenoid lock, and few connecting wires. Inprevious approaches, a breadboard was used to create the connections of ESP32 withotherequipment. For this project setting, the connection of a hardware is without the usage of breadboard as the ESP32 extension board is good enough. It contains ar elay module to provide control over high-powered devices and has the ESP32 as its central processing unit, which is capable of doing the processing of logics and also establishes communication. This kind of refined hardware opens up opportunities in IoT for various applications such as the QR code security door for prototyping to deployment without fail and in an efficient way

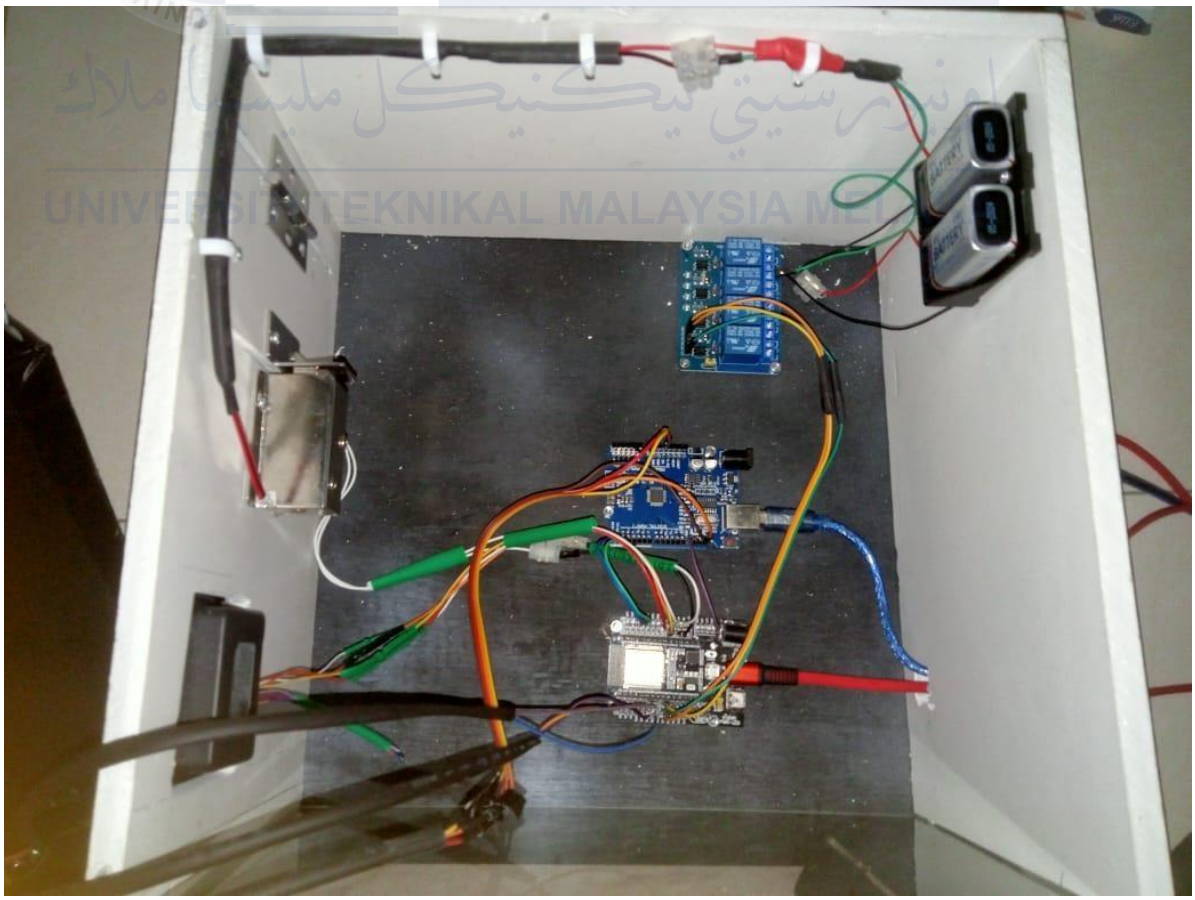

Figure 4.1: Initial project hardware

4.2.2 Project Prototype

Figure 4.2: Final result of project

Figure 4.2 illustrates the outcome of the QR code-based smart security door system. This project represents an advanced IoT-enabled solution that integrates cutting-edge technology with enhanced security features. As shown in the Figure above, the prototype contains QR scanner beside of door. That's where the QR code will be scanned. On top of the prototype, there's a red and green LED and LCD.

4.2.3 Project Application

The interface of the application is shown in the figure below starting from Figure 4.3 to Figure 4.6. Figure 4.3 is the homepage of the application consisting of navigation to Homestay detail, List booking and list door. Figure 4.4 show interface of homestay detail, Figure 4.5 shows the booking detail page and figure 4.6 shows the status of the door.

Figure 4.3: Application homepage

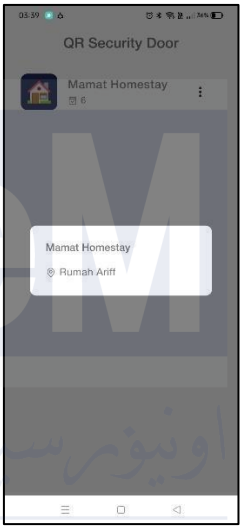

Figure 4.4: Homestay Detail

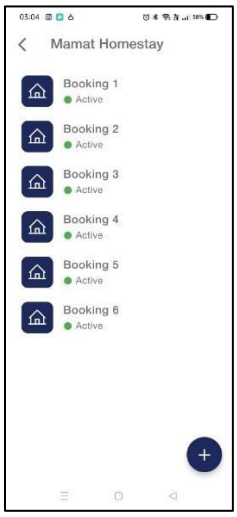

Figure 4.5: List Booking

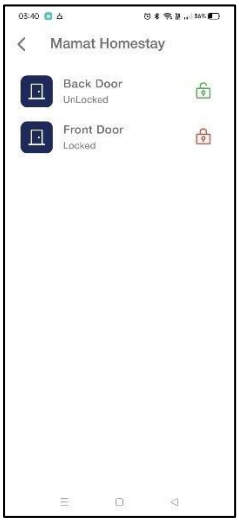

Figure 4.6: List door

4.2.4 Project Database

This project required the usage of database to ensure application interact with the hardware even in long distances. We use Firebase as the database platform where the QR code data is stored. The hardware will interact with the database using ESP32 with internet connection.

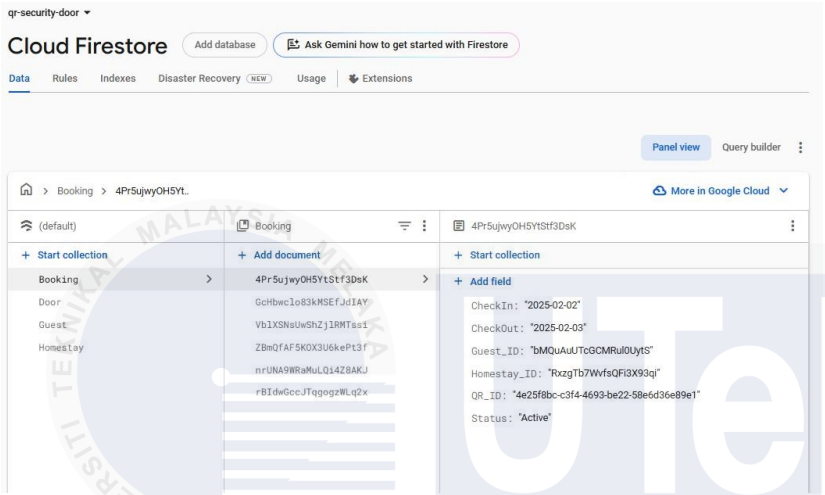

Figure 4.7: Firebase's Firestore

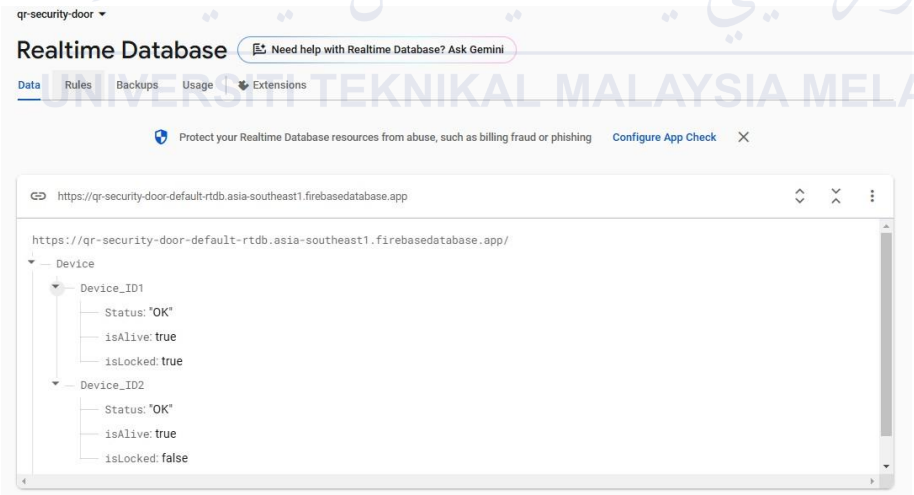

Figure 4.8: Firebase's reatime database

Figure 4.7 shows the Firebase's Firestore that stored QR code data that were generated on applications. This is where the hardware will parse the QR code scanned on QR code scanner. In figure 4.8 is the real-time database of the Firebase. This part is interacted with

List door page on the application. It shows the status of the door either it is close or open. This data is updated in real-time.

4.2.5 Generating new QR code for added user

Figure 4.3 – Figure 4.6 shows the flow of QR code authentication in QR code security lock system ensures secure and reliable access control. When a new user wants to gain access, the homeowner generates a unique QR code on the application.

Figure 4.9: User added image partition from left to right in sequence

In Figure 4.9 1st partition, shows the page of list booking. If you look at the bottom corner, there's a plus button. The button when pressed will pop up the booking log as shown in 2nd partition. The booking log is where we add a client data like name, IC number, phone number, and check-in and check-out. Once submit, the system will generate QR data shown in third partition. In the last partition, it shows the total booking is added as there's a total of 7 compared to the 1st image partition.

4.2.6 Removing the existing booking.

This feature allows homeowners to remove specific QR codes and their associated booking records from the Firebase database without affecting other active codes. This function is particularly useful when access for a specific user needs to be revoked due to expired bookings or changes in access permissions. By deleting only the required QR code, the system maintains flexibility and avoids the need for a full reset. This selective deletion ensures that access remains secure and organized while allowing the homeowner to manage credentials. The Figure 4.10 shows a sequence of deleting the existing QR code. When the selecting data booking is pressed, it will navigate to the booking detail. There the user can terminate the QR code and thus the number of booking will decrease as shown.

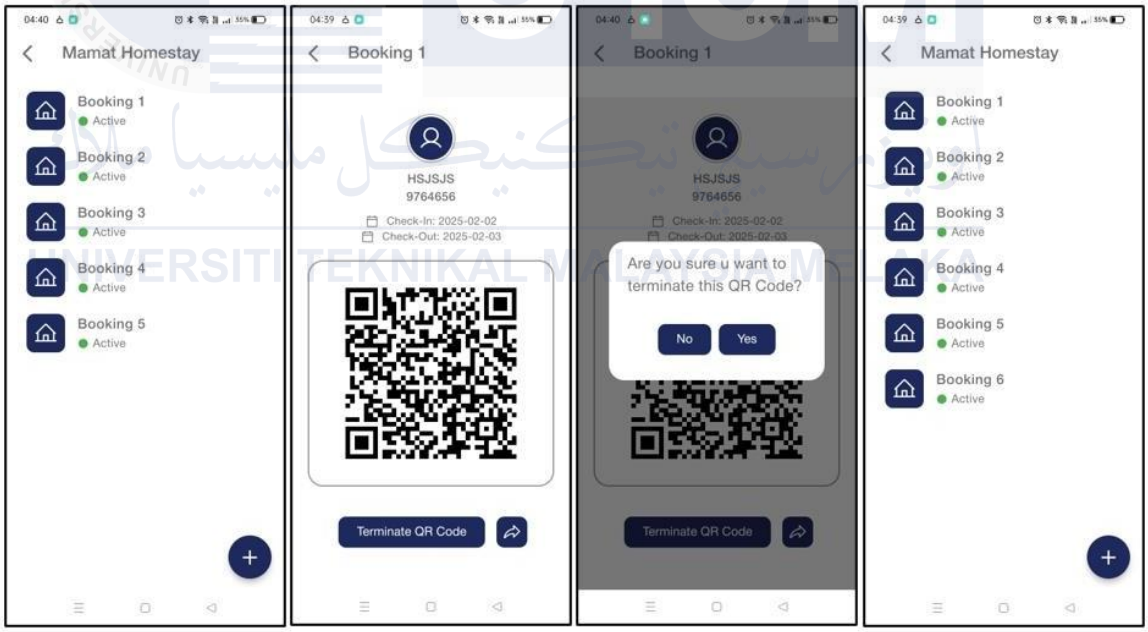

Figure 4.10: QR code data deletion image partition from left to right in sequence

4.2.7 Accessing the homestay with the correct QR code

Figure 4.11 shows an authorized user accessing the QR code security lock key system with a valid QR code in demonstrating effectiveness and reliability concerning the QR based security door system. Once the QR is scanned, the information is cross checked in real time with the Firebase database. Once the QR code has been verified to be correct, it triggers the turning on of the solenoid lock and the user may enter. It will display on the LCD, upon successful authentication, "Door is UNLOCKED". This approach provides evidence of the security and convenience in QR code authentication since it ensures that only pre-approved users with valid QR codes unlock the door, preventing unauthorized entry and increasing overall security.

Figure 4.11: Authorized user gaining access with a correct QR code

4.2.8 Accessing the homestay with incorrect QR code

Figure 4.12 Shows unauthorized user trying to access the system with an invalid/unregistered QR code. The system immediately checks the QR code data by comparing it to the records present on Firebase once it has been scanned. It sends negative logic to the solenoid lock if there is no matching entry in the Firebase. The lock will remain securely locked.

Figure 4.12: Unauthorized user attempt to access with a invalid QR code

4.2.9 Real-time update on door status

Since the system is compatible with the database, User can see the status of the door. This is because the system will update the door status in real-time. You can see the door status locked on Figure 4.13 while in Figure 4.14 shows the door status unlocked.

Figure 4.13: Door status locked

Figure 4.14: Door status unlocked

4.3 Data Analysis

For the data analysis, the table will explain about the functionality of QR code scanner and the relay of the solenoid lock. It will also explain about the limitation of the QR code scanner with fetching error table.

4.3.1 Distance vs Time taken

The table below summarize the delation of the relay once the QR code is scanned based on the distance. It's basically a time taken for the relay before activating the solenoid Lock. This table shows the factor play by the QR scanner limitation on scanning the QR code from certain distance. Therefore, the delation of the relay can be related with the distance when the QR code is scanned.

	Attempt (Relay Delation(s))					
Distance	1	2	3	4	5	Average
5	0	0	3.57	0	3.74	1.46
10	6.48	5.24	4.85	5.46	6.19	5.644
15	4.99	6.51	4.82	4.25	5.33	5.18
20	5.18	4.56	6.57	6.17	4.25	5.346
25	6.81	4.18	5.81	8.25	4.88	5.986

Table 4.1: Table for relay delation

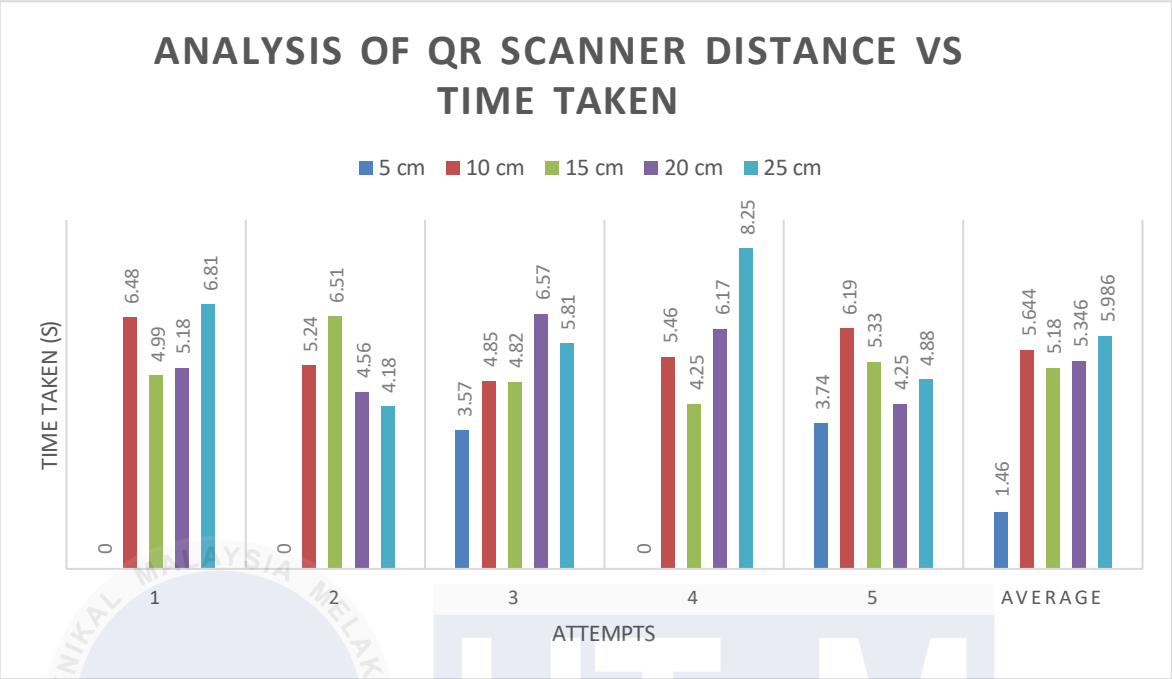

Figure 4.15 : Histogram for Distance vs Time taken

In Figure 4.15 shows that in the first and second attemps, for 5 cm distance is invalid when it comes to scanning the QR code. When the QR code is too closed with the QR scanner, It will become hard for it to read the data.

Figure 4.16 : Method use to record data

Figure 4.16 shows a method used to measure the distance from phone to QR code scanner using measuring tapes. The time taken is based on the relay reaction.

4.3.2 Fetching error analysis

Table 4.2 and Figure 4.16 shows some of the major contributors, based on error rate analysis for the QR code-based smart security system, are as follows: The most significant contributor is QR codes with a black background, accounting for 10 errors. This is primarily due to challenges in scanning caused by poor contrast between the QR code and its background. Slow Internet is responsible for 4 errors, leading to delays in QR code validation and access authorization. Distance-related issues contribute 3 errors, typically occurring when the QR code is scanned from an improper or excessive range. Environmental factors, such as poor lighting or cable management for 2 error. Finally, QR codes with a white background have the least impact, contributing to just 1 error. These findings highlight the need to improve QR code design, optimize environmental conditions, and ensure reliable internet connectivity to reduce errors and enhance system performance.

Error Type	Description	Frequency
Slow internet	The internet connection plays a huge part of the system as the hardware and software are interacting through database.	4
Distance	The QR code scanner distance limitation of scanning the QR code	3
QR code with black background	The QR code with the Black background is causing the big problem of the scanner to scan it	10
QR code with white background	QR code with white background are less likely to cause an error compared to the black background	1
Environment	The cable management can cause a multiple error if not connect correctly. If the Esp32 is damaged, it could lead for the same thing.	2

Table 4.2 : Fetching error table

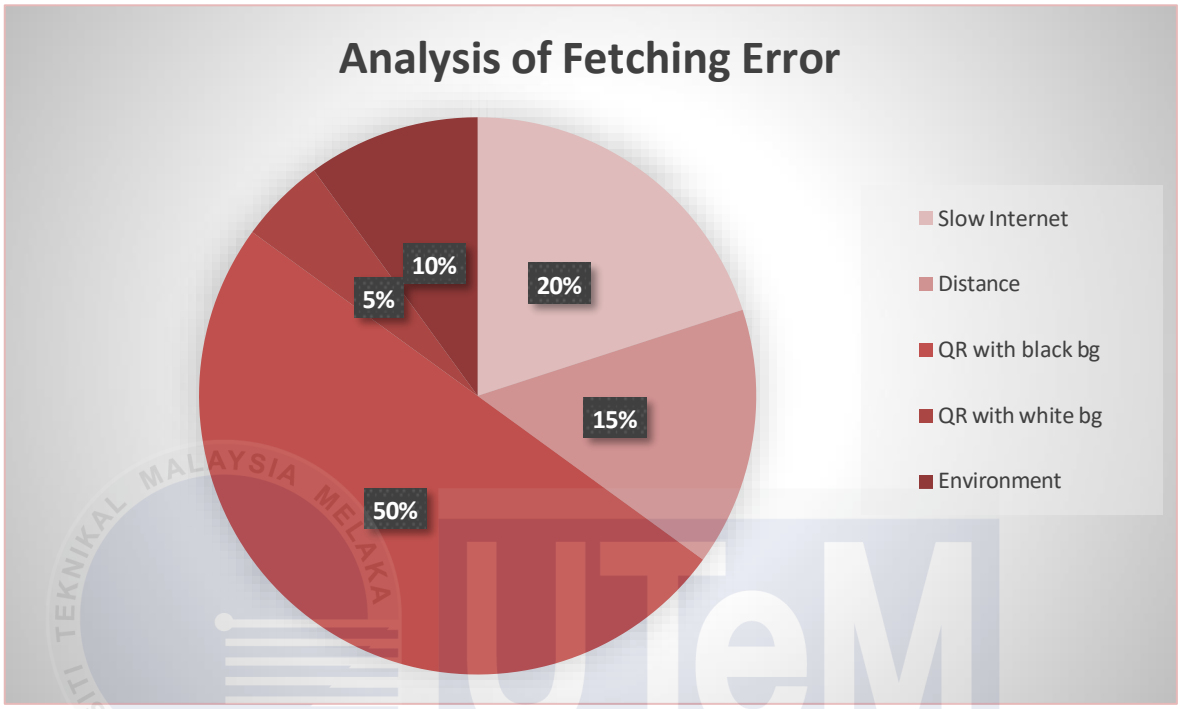

Figure 4.17 : Fetching error pie chart

Figure 4.17 shows a that QR code with a black background become a huge factor of the error. This is due to the lack of capability of the QR code scanner. Changing it with the expensive or better one might decrease the amount of error.

4.4 Summary

In summary, development of intelligent based on iot for security lock key using QR code specialize for homestay improved security and convenience. The system demonstrated it effectiveness in preventing the unauthorize user as shown in Figure 4.12. IoT integration allows me to monitor the status of the door as shown in Figure 4.13 and Figure 4.14. Lastly, the data analysis that i've collect shows even with the error, it still does not effect the safety of the homestay.In the data analysis also conclcude that in order to improve the system, a better QR code scanner or better internet can be implemented

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

In conclusion, the development of the IoT-based security lock system using QR codes for homestays has proven to be an innovative and effective solution for modern access control challenges. The project's first objective, creating a functional security system, was successfully achieved through the integration of IoT components such as the ESP32 microcontroller, QR code scanner, and Firebase database. This system provides secure, contactless access control, ensuring convenience for both guests and property owners.The second objective, evaluating the system's performance, was fulfilled through tests that demonstrated the reliability of QR code scanning, real-time data synchronization, and secure door locking mechanisms. Results indicate the system's robustness in preventing unauthorized access and its potential scalability to meet future needs. Overall, the project showcases the effective implementation of IoT technology to enhance security and operational efficiency in homestay settings.

5.2 Potential for commercialization

In the smart home security market, this project ability to generate, validate, and revoke QR codes for home access, this sytem can cater the growing demand for smart home solutions. It can be comercialize in property management and rentals too. The system is ideal for short-term homestay renting or rental platforms. Hosts can issue temporary QR code keys

to guests, ensuring secure and controlled access that expires automatically after the stay period.

5.3 Future Works

For future improvements, The development of Intelligent based IoT for security door using QR Code specialize for homestay could be enhace as follows:

a) Instead of only homeowner using the application, it would be great if the customer side also it's own interface of choosing the homestay that their like. Basically the same application but on the customer side.

b) The QR code being generate on the customer phone instead of the homeowner phone as long they were grant permission by the owner.

c) There can be multiple QR code produce that allows people to access to the house with their information shown to the homeowner.

d) The QR code produce on the customer will be valid only for a short time like 1 minute only to avoid duplication

e) There should also an option for the customer to make a payment so in case if there's a scammer, they can make a report and ask for a refunds.

REFERENCES

[1] "What is IoT (Internet of Things) and How Does it Work? | Definition from TechTarget." Accessed: Jun. 10, 2024. [Online]. Available: https://www.techtarget.com/iotagenda/definition/Internet-of-Things-IoT

[2] M. B. Hoy, "An introduction to QR codes: Linking libraries and mobile patrons," *Med Ref Serv Q*, vol. 30, no. 3, 2011, doi: 10.1080/02763869.2011.590423.

[3] M. H. Tseng and H. C. Wu, "A cloud medication safety support system using QR code and Web services for elderly outpatients," *Technol Health Care*, vol. 22, no. 1, pp. 99–113, 2014, doi: 10.3233/THC-140778.

[4] "QR Codes for IoT Security Solutions: Reinforce Safety | My QR Code." Accessed: Jun. 10, 2024. [Online]. Available: https://myqrcode.com/industry/iot-security-solutions

[5] "Microcontroller: Types, Functions, Uses, Challenges, and Solutions." Accessed: Jun. 10, 2024. [Online]. Available: https://www.shiksha.com/online-courses/articles/microcontroller-types-functions-uses-challenges-and-solutions-blogId-155711

[6] A. Maier, A. Sharp, and Y. Vagapov, "Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things," *2017 Internet Technologies and Applications, ITA 2017 - Proceedings of the 7th International Conference*, pp. 143–148, Nov. 2017, doi: 10.1109/ITECHA.2017.8101926.

[7] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010, doi: 10.1016/J.COMNET.2010.05.010.

[8] "What is an Arduino? - SparkFun Learn." Accessed: Jun. 10, 2024. [Online]. Available: https://learn.sparkfun.com/tutorials/what-is-an-arduino/all

[9] A. G. Smith, "Introduction to Arduino A piece of cake! Introduction to Arduino: A piece of cake!," 2011, Accessed: Jun. 10, 2024. [Online]. Available: http://www.amazon.com

[10] "Adafruit Learning System." Accessed: Jun. 10, 2024. [Online]. Available: https://learn.adafruit.com/assets/3199

[11] "Raspberry Pi based Smart Phone (Bluetooth) Controlled Home Automation." Accessed: Jun. 10, 2024. [Online]. Available: https://circuitdigest.com/microcontroller-projects/raspberry-pi-smart-phone-home-automation

[12] A. Modules, S. P. Micro, and A. Mega, "Arduino IDE : Introduction," *[online]. Disponible en: https://www.arduino.cc/en/Guide/Introduction*, 2015.

[13] "What is a Raspberry Pi? | Opensource.com." Accessed: Jun. 10, 2024. [Online]. Available: https://opensource.com/resources/raspberry-pi

[14] "Hello Raspberry Pi: Geany - small and fast Editor/IDE." Accessed: Jun. 10, 2024. [Online]. Available: https://helloraspberrypi.blogspot.com/2013/11/geany-small-and-fast-editoride.html

[15] F. Istiqomah, D. K. Nuurul Izza, J. Susila, B. Al Kindhi, E. Indasyah, and F. I. Adhim, "Automated Barrier Gate for Housing Estate Security System Using QR Code Based on Android Application," *2021 International Conference on Advanced Mechatronics, Intelligent Manufacture and Industrial Automation, ICAMIMIA 2021 - Proceeding*, pp. 293–297, 2021, doi: 10.1109/ICAMIMIA54022.2021.9809801.

[16] A. F. M. Fauzi, N. N. Mohamed, H. Hashim, and M. A. Saleh, "Development of Web-Based Smart Security Door Using QR Code System," *2020 IEEE International Conference on Automatic Control and Intelligent Systems, I2CACIS 2020 - Proceedings*, pp. 13–17, Jun. 2020, doi: 10.1109/I2CACIS49202.2020.9140200.

[17] A. Jain, A. Panwar, M. Azam, and R. Khanam, "Smart door access control system based on QR code," *International Journal of Informatics and Communication Technology*, vol. 12, no. 2, pp. 171–179, Aug. 2023, doi: 10.11591/IJICT.V12I2.PP171-179.

[18] Nethrasri P and A. Venkataramana, "QR code based door opening system," vol. 11, no. 11, 2020, Accessed: Jun. 10, 2024. [Online]. Available: www.jespublication.com

[19] Y. Rahayu, L. Afif, and P. J. Soh, "Design and development of smart lock system based QR-Code for library's locker at Faculty of Engineering, Universitas Riau," vol. 26, no. 3, pp. 379–384, 2022, doi: 10.22441/sinergi.2022.3.013.

[20] L. Antonio Pereira Neves, K. Santos Martins, W. Ricardo Santos Lima, and G. Antonio Giraldi, "QRCode DOOR Project: Access Control Application using QR Code Image".

[21] B. Suresh, A. S. Kalyan, B. Bharat, T. Raju, and M. Venkatesh, "Door Lock Security System Using Raspberry Pi & QR Code," *International Research Journal of Engineering and Technology*, 2021, Accessed: Jun. 10, 2024. [Online]. Available: www.irjet.net

APPENDICES

s	Week / Activity	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14
1	CHAPTER 1														
2	CHAPTER 2														
3	CHAPTER 3														
4	CHAPTER 4														
5	CHAPTER 5														
6	MEETING WITH SUPERVISOR														
7	DRAFT SUBMISSION														
8	REPORT SUBMISSION														
9	PRESENTATION														

PSM 1 Project progress by week

s	Week / Activity	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14
1	CHAPTER 3 & 4														
2	HARDWARE SETUP														
3	DATABASE SETUP														
4	APPLICATION DEVELOPMENT														
5	CREATING POSTER														
6	MEETING WITH SUPERVISOR														
7	DRAFT SUBMISSION														
8	REPORT SUBMISSION														
9	PRESENTATION														
10	IIDEX PRESENTATION														

PSM 2 Project progress by week

Appendix C Coding

ESP32 coding

```
#include <WiFi.h>
#include <FirebaseESP32.h>
#include <HardwareSerial.h>
#include <HTTPClient.h>
#include <time.h>
#include <ArduinoJson.h>

#define RXD2 16          // ESP32 RX pin connected to Scanner TX
#define TXD2 17          // ESP32 TX pin connected to Scanner RX
#define MC38_PIN 4       // Pin connected to the MC38 sensor
#define RED_LED_PIN 32   // Pin connected to the Red LED
#define GREEN_LED_PIN 33 // Pin connected to the Green LED
#define RELAY_PIN 26     // ESP32 pin connected to the relay module

const char* ssid = "Mamat";
const char* password = "mamatpower08";

#define FIREBASE_HOST "qr-security-door-default-rtdb.asia-
southeast1.firebasedatabase.app"
#define FIREBASE_API_KEY
"AIzaSyBN_3XuVhzeewMGuwMi1CwGTm9m4K1ZbFo"

const String projectId = "qr-security-door";
const String apiKey = "AIzaSyBN_3XuVhzeewMGuwMi1CwGTm9m4K1ZbFo";
const String collectionPath = "QRCode";

const String homestay_id = "RxzgTb7WvfsQFi3X93qi";

const char* ntpServer = "time.google.com"; // NTP server
const long gmtOffset_sec = 8 * 3600;     // Singapore is UTC+8
const int daylightOffset_sec = 0;        // No daylight saving in Singapore

const char* deviceID = "Device_ID1";

FirebaseConfig config;
FirebaseAuth auth;
FirebaseData firebaseData;

HardwareSerial scannerSerial(2);  // Use UART2 for the scanner

bool qrApprove = false;         // QR code approved
bool mc38CycleComplete = false;  // Door open/close cycle
bool isDoorLocked = true;       // Door lock state

void setup() {
  Serial.begin(9600);
```

```
  scannerSerial.begin(9600, SERIAL_8N1, RXD2, TXD2);

  pinMode(MC38_PIN, INPUT_PULLUP); // MC38 sensor
  pinMode(RED_LED_PIN, OUTPUT);
  pinMode(GREEN_LED_PIN, OUTPUT);
  pinMode(RELAY_PIN, OUTPUT);

  digitalWrite(RED_LED_PIN, HIGH);   // Start with door locked
  digitalWrite(GREEN_LED_PIN, LOW);
  digitalWrite(RELAY_PIN, LOW);

  connectToWiFi();
  initializeFirebase();

  configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);
}

void loop() {
  int mc38State = digitalRead(MC38_PIN);

  if (scannerSerial.available()) {
    String qrData = readScannerData();
    processQRData(qrData);
  }

  if (qrApprove) {
    handleDoorCycle(mc38State);
  }

  updateSolenoidState();
  delay(100);
}

/**
 * Connects to the specified WiFi network.
 */
void connectToWiFi() {
  sendToLCD("Connecting to WiFi...");
  WiFi.begin(ssid, password);
  int attempts = 0;

  while (WiFi.status() != WL_CONNECTED && attempts < 20) {
    delay(1000);
    attempts++;
    sendToLCD("Attempting to connect...");
  }

  if (WiFi.status() == WL_CONNECTED) {
    sendToLCD("WiFi connected.");
  } else {
```

```
    sendToLCD("WiFi failed.");
    while (true);
  }
}

/**
 * Initializes the Firebase connection.
 */
void initializeFirebase() {
  sendToLCD("Initialized Firebase...");
  config.host = FIREBASE_HOST;
  config.api_key = FIREBASE_API_KEY;

  Firebase.begin(&config, &auth);
  Firebase.reconnectWiFi(true);

  if (Firebase.signUp(&config, &auth, "", "")) {
    sendToLCD("Firebase connected");
  } else {
    sendToLCD("Fail to connect Firebase");
    while (true);
  }
}

/**
 * Reads data from the QR code scanner.
 * @return The data read from the scanner.
 */
String readScannerData() {
  sendToLCD("Reading QR...");
  String qrData = "";
  unsigned long startTime = millis();
  while (millis() - startTime < 500) {
  while (scannerSerial.available()) {
      char incomingByte = scannerSerial.read();
      qrData += incomingByte;
      startTime = millis();
    }
  }
  return qrData;
}

/**
 * Processes the QR code data.
 * @param qrData The data read from the QR code.
 */
void processQRData(String qrData) {
  sendToLCD("Processing QR...");
  DynamicJsonDocument doc(2048);
  DeserializationError error = deserializeJson(doc, qrData);
```

```
  if (error) {
    sendToLCD("QR parse fail");
    return;
  }

  const char* qrId = doc["qr_id"];
  const char* bookId = doc["book_id"];

  if (qrId && bookId) {
    if (getBookingData(String(qrId), String(bookId))) {
      sendToLCD("Access granted");
      qrApprove = true;
      digitalWrite(GREEN_LED_PIN, HIGH);
      digitalWrite(RED_LED_PIN, LOW);
      isDoorLocked = true; // Door remains locked until opened and closed
    } else {
      sendToLCD("Validation fail");
    }
  } else {
    sendToLCD("Invalid QR data");
  }
}

/**
 * Fetches booking data from the Firebase database.
 * @param qrId The QR code ID.
 * @param bookId The booking ID.
 * @return True if the booking data is valid, false otherwise.
 */
bool getBookingData(String qrId, String bookId) {
  sendToLCD("Fetching data...");
  HTTPClient http;
  String url = "https://firestore.googleapis.com/v1/projects/" + projectId +
               "/databases/(default)/documents/Booking/" + bookId + "?key=" + apiKey;

  http.begin(url);
  int httpResponseCode = http.GET();

  if (httpResponseCode > 0) {
    String payload = http.getString();
    Serial.println("Response Payload: " + payload);
    DynamicJsonDocument doc(2048);
    DeserializationError error = deserializeJson(doc, payload);

    if (error || doc.containsKey("error")) {
      http.end();
      return false;
    }
```

```
    const char* homestay_id = doc["fields"]["Homestay_ID"]["stringValue"];
    const char* qr_id = doc["fields"]["QR_ID"]["stringValue"];
    const char* check_in = doc["fields"]["CheckIn"]["stringValue"];
    const char* check_out = doc["fields"]["CheckOut"]["stringValue"];
    const char* status = doc["fields"]["Status"]["stringValue"];

    http.end();
    if (String(qr_id) == qrId) {
      if (String(status) == "Active" && String(homestay_id) == homestay_id) {
        if (validateDate(String(check_in), String(check_out))) {
          unlockDoor();
          return true;
        }
      } else {
        sendToLCD("Wrong House!");
      }
    }
  } else {
    http.end();
  }
  return false;
}

/**
 * Handles the door open/close cycle based on the MC38 sensor state.
 * @param mc38State The current state of the MC38 sensor.
 */
void handleDoorCycle(int mc38State) {
  static bool waitingForOpen = true;
  static bool doorOpened = false;
  static bool waitingForClose = false;

  if (waitingForOpen && mc38State == LOW) {
    doorOpened = true;
    waitingForOpen = false;
    waitingForClose = true; // Start waiting for door to close
    sendToLCD("Door is Unlocked");
  }

  if (waitingForClose && mc38State == HIGH) {
    lockDoor();
    qrApprove = false;   // Reset QR approval status
    waitingForClose = false;
    waitingForOpen = true; // Reset and wait for the next open event
    doorOpened = false;
    sendToLCD("Door locked");
  }
}

/**
```

```
 * Locks the door and updates the database.
 */
void lockDoor() {
  Serial.println("Locking door...");
  digitalWrite(RED_LED_PIN, HIGH);
  digitalWrite(GREEN_LED_PIN, LOW);
  digitalWrite(RELAY_PIN, LOW);
  isDoorLocked = true;
  sendToLCD("Door Locked");

  updateDatabaseLockState(true);
}

/**
 * Unlocks the door and updates the database.
 */
void unlockDoor() {
  Serial.println("Unlocking door...");

unsigned long startUnlockTime = millis(); // Record the start time

  digitalWrite(RED_LED_PIN, LOW);
  digitalWrite(GREEN_LED_PIN, HIGH);
  digitalWrite(RELAY_PIN, HIGH);
  isDoorLocked = false;
  sendToLCD("Door Unlocked");

   unsigned long endUnlockTime = millis(); // Record the end time
  unsigned long unlockDuration = endUnlockTime - startUnlockTime;

  Serial.print("Time taken to unlock the door: ");
  Serial.print(unlockDuration);
  Serial.println(" ms");

  updateDatabaseLockState(false);
}

/**
 * Updates the lock state in the Firebase database.
 * @param isLocked The current lock state.
 */
void updateDatabaseLockState(bool isLocked) {
  Serial.println("Updating database lock state...");
  if (WiFi.status() == WL_CONNECTED) {
  HTTPClient http;
    String url = "https://" + String(FIREBASE_HOST) + "/Device/" +
String(deviceID) + ".json?auth=" + String(FIREBASE_API_KEY);

    http.begin(url);
    http.addHeader("Content-Type", "application/json");
```

```
    String jsonPayload = "{\"isLocked\":" + String(isLocked ? "true" : "false") +
"}";

    int httpResponseCode = http.PATCH(jsonPayload);

    if (httpResponseCode > 0) {
      String response = http.getString();
      sendToLCD("Database Update OK");
    } else {
      sendToLCD("Database Update Fail");
    }

    http.end(); // Free resources
  } else {
    sendToLCD("WiFi Disconnected");
  }
}

/**
 * Validates the current date against the check-in and check-out dates.
 * @param checkIn The check-in date.
 * @param checkOut The check-out date.
 * @return True if the current date is within the valid range, false otherwise.
 */
bool validateDate(String checkIn, String checkOut) {
  Serial.println("Validating date...");
  struct tm timeinfo;
  if (!getLocalTime(&timeinfo)) {
    sendToLCD("Time Fail");
    return false;
  }

  time_t now = mktime(&timeinfo);
  Serial.print("Current timestamp: ");
  Serial.println(now);

  time_t checkInTime = parseDate(checkIn);
  Serial.print("Check-in timestamp: ");
  Serial.println(checkInTime);

  time_t checkOutTime = parseDate(checkOut);
  Serial.print("Check-out timestamp: ");
  Serial.println(checkOutTime);

  if (now >= checkInTime && now <= checkOutTime) {
    Serial.println("Current time is within the valid range.");
    return true;
  } else {
```

```
    Serial.println("Current time is outside the valid range.");
    return false;
  }
}

/**
 * Parses a date string into a time_t object.
 * @param dateStr The date string to parse.
 * @return The parsed time_t object.
 */
time_t parseDate(String dateStr) {
  struct tm tm;
  memset(&tm, 0, sizeof(struct tm));
  tm.tm_year = dateStr.substring(0, 4).toInt() - 1900;
  tm.tm_mon = dateStr.substring(5, 7).toInt() - 1;
  tm.tm_mday = dateStr.substring(8, 10).toInt();
  if (dateStr.length() > 10) {
    tm.tm_hour = dateStr.substring(11, 13).toInt();
    tm.tm_min = dateStr.substring(14, 16).toInt();
    tm.tm_sec = dateStr.substring(17, 19).toInt();
  }
  return mktime(&tm);
}

/**
 * Updates the solenoid state based on the LED states.
 */
void updateSolenoidState() {
  if (digitalRead(RED_LED_PIN) == HIGH) {
    digitalWrite(RELAY_PIN, HIGH); // Lock door
  } else if (digitalRead(GREEN_LED_PIN) == HIGH) {
    digitalWrite(RELAY_PIN, LOW); // Unlock door
  }
}

/**
 * Sends a message to the Arduino Uno to be displayed on the LCD.
 * @param message The message to send.
 */
void sendToLCD(String message) {
  Serial.println(message); // Send message to Arduino Uno
}
```

Arduino Uno coding to display LCD

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// Set the LCD address to 0x27 for a 24 chars and 4 line display
LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup() {
  // Initialize serial communication
  Serial.begin(9600);

  // Initialize the LCD
  lcd.begin(16, 2);
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("Waiting for ESP32");
}

void loop() {
  // Check if there is data available on the serial port
  if (Serial.available()) {
    // Read a line of data from the serial port
    String message = Serial.readStringUntil('\n');

    // Clear the LCD and print the message
    lcd.clear();
    // Display the message on the LCD
    displayMessage(message);
  }
}

/**
 * Displays a message on the LCD, handling messages longer than 16 characters.
 */
void displayMessage(String message) {
  // If the message is longer than 16 characters, split it across lines
  if (message.length() > 16) {
    // Display the first 16 characters on the first line
    lcd.setCursor(0, 0);
    lcd.print(message.substring(0, 16));

    // Display the remaining characters on the second line
    lcd.setCursor(0, 1);
    lcd.print(message.substring(16));
  } else {
    // Display the message on the first line
    lcd.setCursor(0, 0);
    lcd.print(message);
  }
```

```
}
```

Flutter coding using Dart language on Android Studio.

Main.dart

```
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:securitydoor/screen/root.dart';

import 'constant/color.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized(); // Ensure bindings are initialized
  await Firebase.initializeApp(); // Initialize Firebase
  runApp(const QRSecurityDoorApp());
}

class QRSecurityDoorApp extends StatelessWidget {
  const QRSecurityDoorApp({super.key});

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    SystemChrome.setPreferredOrientations([
    DeviceOrientation.portraitUp,
    DeviceOrientation.portraitDown,
    ]);
    SystemChrome.setSystemUIOverlayStyle(
        const SystemUiOverlayStyle(statusBarBrightness: Brightness.dark));
    return ScreenUtilInit(
      designSize: const Size(360, 690),
      minTextAdapt: true,
      splitScreenMode: true,
      builder: (_, child) => MaterialApp(
        theme: ThemeData(
            fontFamily: 'HelveticaNeueRegular',
            colorScheme: ColorScheme.fromSwatch().copyWith(
                primary: secondaryColor,
                secondary: primaryColor,
                background: primaryColor,
                surfaceTint: primaryColor)),
        supportedLocales: const [
          Locale('en'),
        ],
        debugShowCheckedModeBanner: false,
        title: 'QR Security Door',
        home: child,
      ),
```

```
      child: const RootScreen(),
    );
  }
}
```

Firebase_service.dart

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:uuid/uuid.dart';

class FirebaseService {
  final FirebaseFirestore _firestore = FirebaseFirestore.instance;

  var uuid = Uuid();

  Future<Map<String, String>> createBooking({
    required String clientName,
    required String clientIcNo,
    required String clientTelNo,
    required String homestayId,
    required String checkInDate,
    required String checkOutDate,
  }) async {

    String generatedQRCode = uuid.v4();

    // process 1: Create a new guest document
    DocumentReference guestRef = await _firestore.collection('Guest').add({
      'Name': clientName,
      'IC_Number': clientIcNo,
      'Tel_No': clientTelNo,
    });

    // process 2: steCreate a new booking document
    DocumentReference bookingRef = await _firestore.collection('Booking').add({
      'Guest_ID': guestRef.id,
      'Homestay_ID': homestayId,
      'CheckIn': checkInDate,
      'CheckOut': checkOutDate,
      'QR_ID': generatedQRCode,
      'Status': 'Active',
    });
    // process 3: Create a new QRCode document


    return {
      'bookingId': bookingRef.id,
      'qrId': generatedQRCode,
```

```
    };
  }

  Future<String?> getBookingId(String qrId) async {
    DocumentSnapshot qrCodeDoc = await
_firestore.collection('QRCode').doc(qrId).get();
    String bookingId = qrCodeDoc['Book_ID'];
    return bookingId;
  }

  Future<List<Map<String, dynamic>>> fetchHomestays() async {
    List<Map<String, dynamic>> homestays = [];
    QuerySnapshot homestaySnapshot = await
_firestore.collection('Homestay').get();

    for (var homestayDoc in homestaySnapshot.docs) {
      String homestayId = homestayDoc.id;
      QuerySnapshot bookingSnapshot = await _firestore
          .collection('Booking')
          .where('Homestay_ID', isEqualTo: homestayId)
          .get();
      int totalBookings = bookingSnapshot.docs.length;

      homestays.add({
        'id': homestayDoc.id,
        'name': homestayDoc['Name'],
        'location': homestayDoc['Address'],
        'totalBookings': totalBookings,
      });
    }

    return homestays;
  }

  Future<Map<String, dynamic>?> fetchBookingDetails(String bookingId) async {
    DocumentSnapshot bookingDoc = await
_firestore.collection('Booking').doc(bookingId).get();

    if (bookingDoc.exists) {
      Map<String, dynamic> bookingData = bookingDoc.data() as Map<String,
dynamic>;
      String guestId = bookingData['Guest_ID'];

      DocumentSnapshot guestDoc = await
_firestore.collection('Guest').doc(guestId).get();

      if (guestDoc.exists) {
        Map<String, dynamic> guestData = guestDoc.data() as Map<String,
dynamic>;
        return {
```

```
        ...bookingData,
        'GuestDetails': guestData,
      };
    }
   }
   return null;
  }

  Future<String?> getQRCodeIdByBookingId(String bookID) async {
   DocumentSnapshot bookingDoc = await
_firestore.collection('Booking').doc(bookID).get();

   if (bookingDoc.exists) {
    return bookingDoc['QR_ID'];
   } else {
    return null;
   }
  }

  Future<void> terminateQRCodeAndBooking(String bookingId) async {
   await _firestore.collection('Booking').doc(bookingId).delete();
  }

  Future<List<Map<String, dynamic>>> fetchBookings(String homestayId) async {
   List<Map<String, dynamic>> bookings = [];
   QuerySnapshot bookingSnapshot = await _firestore
     .collection('Booking')
     .where('Homestay_ID', isEqualTo: homestayId)
     .get();

   for (var bookingDoc in bookingSnapshot.docs) {
    var bookingData = bookingDoc.data() as Map<String, dynamic>;
    bookingData['id'] = bookingDoc.id; // Add the bookingDoc.id to the map
    bookings.add(bookingData);
   }

   return bookings;
  }

}
```

Root.dart

```
import 'package:flutter/material.dart';

import 'homestay_screen.dart';

class RootScreen extends StatelessWidget {
  const RootScreen({super.key});

  @override
  Widget build(BuildContext context) {
    return const HomestayScreen();
  }
}
```

qr_screen.dart

```
import 'dart:convert';

import 'package:flutter/material.dart';
import 'package:phosphor_flutter/phosphor_flutter.dart';
import 'package:qr_flutter/qr_flutter.dart';
import 'package:securitydoor/constant/color.dart';
import 'package:securitydoor/constant/text.dart';
import 'package:securitydoor/constant/widget/avatar.dart';
import 'package:securitydoor/services/firebase_service.dart';

import '../constant/widget/custom_alert.dart';
import '../constant/widget/custom_appbar.dart';
import '../constant/widget/custom_button.dart';
import '../constant/widget/custom_icon_button.dart';
import '../controller/share_qr_controller.dart';

class QrScreen extends StatefulWidget {
  final String homestayUnit;
  final String bookingId;
  final String homestayName;

  const QrScreen(
      {super.key,
      required this.homestayUnit,
      required this.bookingId,
      required this.homestayName});

  @override
  State<QrScreen> createState() => _QrScreenState();
}

class _QrScreenState extends State<QrScreen> {
```

```
final GlobalKey qrKey = GlobalKey();
Map<String, dynamic>? bookingDetails;
final FirebaseService _firebaseService = FirebaseService();

Future<void> fetchBookingDetails() async {
  Map<String, dynamic>? details =
      await _firebaseService.fetchBookingDetails(widget.bookingId);
  setState(() {
    bookingDetails = details;
  });
}

@override
void initState() {
  super.initState();
  fetchBookingDetails();
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: CustomAppbar(title: widget.homestayUnit),
    body: Padding(
      padding: const EdgeInsets.all(20),
      child: FutureBuilder<String?>(
        future: _firebaseService.getQRCodeIdByBookingId(widget.bookingId),
        builder: (context, snapshot) {
          if (snapshot.connectionState == ConnectionState.waiting) {
            return const CircularProgressIndicator();
          } else if (snapshot.hasError) {
            return Text('Error: ${snapshot.error}');
          } else if (!snapshot.hasData || snapshot.data == null) {
            return const Text('No QR Code found for this booking');
          } else {
            String qrId = snapshot.data!;
            Map<String, String> qrDataMap = {
            "qr_id": qrId,
              "book_id": widget.bookingId,
            };

            String qrData = jsonEncode(qrDataMap);

            return Column(
              children: [
                const SizedBox(height: 30),
                const Avatar(),
                const SizedBox(height: 10),
                if (bookingDetails != null) ...[
                Text(
                  bookingDetails!['GuestDetails']['Name']?.toUpperCase() ??
```

```
          'Unknown Client',
      textAlign: TextAlign.center,
      style: textMediumSecondary.copyWith(fontSize: 16),
    ),
    Text(
      bookingDetails!['GuestDetails']['Tel_No'] ??
          'Unknown Tel No',
      style: textMediumSecondary.copyWith(fontSize: 16),
    ),
    const SizedBox(height: 10),
    Row(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        Icon(PhosphorIcons.calendarBlank(),
            size: 18, color: secondaryColor),
        const SizedBox(width: 10),
        Text(
          "Check-In: ${bookingDetails!['CheckIn']}",
          style: textRegularSecondary.copyWith(fontSize: 14),
        ),
      ],
    ),
    Row(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        Icon(PhosphorIcons.calendarBlank(),
            size: 18, color: secondaryColor),
        const SizedBox(width: 10),
        Text(
          "Check-Out: ${bookingDetails!['CheckOut']}",
          style: textRegularSecondary.copyWith(fontSize: 14),
        ),
      ],
    ),
  ] else ...[
    const CircularProgressIndicator(),
  ],
  const SizedBox(height: 20),
  Container(
    padding: const EdgeInsets.all(20.0),
    decoration: BoxDecoration(
      border: Border.all(color: Colors.grey, width: 2),
      borderRadius: BorderRadius.circular(20),
    ),
    child: Center(
      child: QrImageView(
        data: qrData,
        version: QrVersions.auto,
        size: 250.0,
      ),
```

```
          ),
        ),
        const SizedBox(height: 30),
        Row(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            CustomButton(
              text: "Terminate QR Code",
              onPressed: () {
                showDialog(
                  context: context,
                  builder: (BuildContext context) {
                    return CustomAlert(
                      text: "Are you sure u want to terminate this QR Code?",
                      onPressed: () async {
                        await _firebaseService
                            .terminateQRCodeAndBooking(
                                widget.bookingId);
                        Navigator.pop(context,
                            true); // Pass true to indicate success
                        Navigator.pop(context,
                            true); // Pass true to indicate success
                      },
                    );
                  },
                );
              },
            ),
            const SizedBox(width: 10),
            CustomIconButton(
              onTap: () async {
                String? qrId = await _firebaseService
                    .getQRCodeIdByBookingId(widget.bookingId);
                if (qrId != null) {
                  await ShareQrController.generateAndShareQRCode(
                      qrData, widget.homestayName);
                } else {
                  ScaffoldMessenger.of(context).showSnackBar(
                  const SnackBar(
                        content: Text(
                            'QR Code not found for this booking')),
                  );
                }
              },
              icon: PhosphorIcons.shareFat(),
            ),
          ],
        ),
      ],
    );
```

```
          }
        },
      ),
    ),
  );
 }
}
```

Qr_dialog.dart

```
import 'dart:convert';

import 'package:flutter/material.dart';
import 'package:phosphor_flutter/phosphor_flutter.dart';
import 'package:qr_flutter/qr_flutter.dart';
import 'package:securitydoor/constant/strings.dart';

import '../constant/color.dart';
import '../constant/widget/custom_button.dart';
import '../constant/widget/custom_icon_button.dart';
import '../controller/share_qr_controller.dart';

class QRDialog extends StatefulWidget {
 final String qrId, bookingId, homestayName;

 const QRDialog(
   {super.key,
    required this.qrId,
    required this.bookingId,
    required this.homestayName});

 @override
 State<QRDialog> createState() => _QRDialogState();
}

class _QRDialogState extends State<QRDialog> {
 final GlobalKey qrKey = GlobalKey();

 @override
 Widget build(BuildContext context) {
  Map<String, String> qrDataMap = {
  "qr_id": widget.qrId,
   "book_id": widget.bookingId,
  };

  String qrData = jsonEncode(qrDataMap);

  return PopScope(
```

```
      canPop: false,
      child: AlertDialog(
        shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(10)),
        backgroundColor: primaryColor,
        content: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          mainAxisSize: MainAxisSize.min,
          children: [
            SizedBox(
            width: 200,
              height: 200,
              child: QrImageView(
                data: qrData,
                version: QrVersions.auto,
                size: 250.0, // Ensure size fits within the container
              ),
            ),
            Row(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                CustomButton(
                    text: "Done",
                    onPressed: () {
                      Navigator.pop(context);
                    }),
                const SizedBox(
                  width: 10,
                ),
                CustomIconButton(
                    onTap: () async {
                      await ShareQrController.generateAndShareQRCode(
                          qrData, widget.homestayName);
                    },
                    icon: PhosphorIcons.shareFat())
              ],
            ),
          ],
        ),
      ),
    );
  }
}
```

List_door_screen.dart

```
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'package:phosphor_flutter/phosphor_flutter.dart';
import 'package:securitydoor/constant/color.dart';
```

```
import 'package:securitydoor/constant/widget/door_tile.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_database/firebase_database.dart';

import '../constant/widget/custom_appbar.dart';

class ListDoorScreen extends StatefulWidget {
  final String homestayId;
  final String homestayName;

  const ListDoorScreen({
    super.key,
    required this.homestayId,
    required this.homestayName,
  });

  @override
  State<ListDoorScreen> createState() => _ListDoorScreenState();
}

class _ListDoorScreenState extends State<ListDoorScreen> {
  final FirebaseFirestore _firestore = FirebaseFirestore.instance;
  final FirebaseDatabase _database = FirebaseDatabase.instanceFor(
    app: Firebase.app(),
    databaseURL: 'https://qr-security-door-default-rtdb.asia-
southeast1.firebasedatabase.app',
  );

  List<Map<String, dynamic>> doors = [];
  bool isLoading = true;

  @override
  void initState() {
    super.initState();
    _fetchDoors();
  }

  Future<void> _fetchDoors() async {
    try {
      // Fetch door document IDs from Homestay collection
      DocumentSnapshot homestaySnapshot = await
_firestore.collection('Homestay').doc(widget.homestayId).get();
      List<dynamic> doorIds = homestaySnapshot['Door'];

      // Fetch door details from Door collection
      for (String doorId in doorIds) {
      DocumentSnapshot doorSnapshot = await
_firestore.collection('Door').doc(doorId).get();
        Map<String, dynamic> doorData = doorSnapshot.data() as Map<String,
dynamic>;
```

```
      // Fetch data from Realtime Database using Device_ID
      String deviceId = doorData['Device_ID'];
      DatabaseReference deviceRef = _database.ref().child('Device').child(deviceId);

      // Listen for changes in the Realtime Database
      deviceRef.onValue.listen((DatabaseEvent event) {
      setState(() {
         doorData['deviceData'] = event.snapshot.value;
       });
      });

      doors.add(doorData);
    }

    setState(() {
      isLoading = false;
    });
  } catch (e) {
    print('Error fetching doors: $e');
    setState(() {
      isLoading = false;
    });
  }
}

@override
Widget build(BuildContext context) {
 return Scaffold(
    appBar: CustomAppbar(title: widget.homestayName),
    body: isLoading
        ? Center(child: CircularProgressIndicator())
        : Padding(
            padding: const EdgeInsets.all(10),
            child: ListView.builder(
            itemCount: doors.length,
            itemBuilder: (context, index) {
               final deviceData = doors[index]['deviceData'];
               final isLocked = deviceData != null && deviceData['isLocked'] == true;
               return DoorTile(
                 name: doors[index]['Name'],
                 status: isLocked ? "Locked" : "UnLocked",
                 statusColor: isLocked ? lock : unlock,
                 trailing: isLocked
                     ? Icon(
                        PhosphorIcons.lockKey(),
                        color: lock,
                        size: 30,
                       )
                     : Icon(
```

```
                    PhosphorIcons.lockKeyOpen(),
                    color: unlock,
                    size: 30,
                  ),
                );
              },
            ),
          ),
    );
  }
}
```

List_booking_screen.dart

```
import 'package:flutter/material.dart';
import 'package:securitydoor/constant/color.dart';
import 'package:securitydoor/constant/strings.dart';
import 'package:securitydoor/screen/qr_screen.dart';
import 'package:securitydoor/services/firebase_service.dart';

import '../constant/widget/booking_tile.dart';
import '../constant/widget/custom_appbar.dart';
import '../constant/widget/floating_action_button.dart';
import 'booking_form_dialog.dart';

class ListBookingScreen extends StatefulWidget {
  final String homestayName;
  final String homestayId;

  const ListBookingScreen({
    super.key,
    required this.homestayName,
    required this.homestayId,
  });

  @override
  State<ListBookingScreen> createState() => _ListBookingScreenState();
}

class _ListBookingScreenState extends State<ListBookingScreen> {
  final FirebaseService _firebaseService = FirebaseService();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: CustomAppbar(title: widget.homestayName),
      body: FutureBuilder<List<Map<String, dynamic>>>(
```

```
    future: _firebaseService.fetchBookings(widget.homestayId),
    builder: (context, snapshot) {
     if (snapshot.connectionState == ConnectionState.waiting) {
      return const Center(child: CircularProgressIndicator());
     } else if (snapshot.hasError) {
      return Center(child: Text('Error: ${snapshot.error}'));
     } else if (!snapshot.hasData || snapshot.data!.isEmpty) {
      return Center(child: Text(noHomestay));
     } else {
      List<Map<String, dynamic>> bookings = snapshot.data!;
      return Padding(
       padding: const EdgeInsets.all(10),
       child: ListView.builder(
       itemCount: bookings.length,
       itemBuilder: (context, index) {
         var booking = bookings[index];
         return BookingTile(
          name: "Booking ${index + 1}",
          status: booking['Status'],
          onTap: booking['Status'] != 'Terminate'
            ? () async {
           bool? result = await Navigator.push(
            context,
            MaterialPageRoute(
             builder: (context) => QrScreen(
              homestayUnit: "Booking ${index + 1}",
              bookingId: booking['id'],
              homestayName: widget.homestayName,
             ),
            ),
           );
           if (result == true) {
            setState(() {}); // Refresh the booking list
           }
          }
            : null,
          statusColor: booking['Status'] == 'Active'
            ? active
            : booking['Status'] == 'Booked'
              ? booked
              : terminate,
         );
        },
       ),
      );
     }
    },
   ),
   floatingActionButton: FAB(onPressed: () async {
    bool? result = await showDialog(
```

```
        context: context,
        builder: (BuildContext context) {
          return BookingFormDialog(
          homestayId: widget.homestayId,
            homestayName: widget.homestayName,
          );
        },
      );
      if (result == true) {
        setState(() {}); // Refresh the booking list
      }
    }),
  );
 }
}
```

Homestay_screen.dart

```
import 'package:flutter/material.dart';
import 'package:securitydoor/screen/homestay_detail_dialog.dart';
import 'package:securitydoor/screen/list_door_screen.dart';
import 'package:securitydoor/services/firebase_service.dart';

import '../constant/strings.dart';
import '../constant/text.dart';
import '../constant/widget/homestay_tile.dart';
import 'list_booking_screen.dart';

class HomestayScreen extends StatefulWidget {
 const HomestayScreen({super.key});

 @override
 State<HomestayScreen> createState() => _HomestayScreenState();
}

class _HomestayScreenState extends State<HomestayScreen> {
 final FirebaseService _firebaseService = FirebaseService();

 @override
 Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text(appTitle, style: textMediumSecondary),
      centerTitle: true,
    ),
    body: FutureBuilder<List<Map<String, dynamic>>>(
      future: _firebaseService.fetchHomestays(),
      builder: (context, snapshot) {
        if (snapshot.connectionState == ConnectionState.waiting) {
```

```
  return const Center(child: CircularProgressIndicator());
 } else if (snapshot.hasError) {
  return Center(child: Text('Error: ${snapshot.error}'));
 } else if (!snapshot.hasData || snapshot.data!.isEmpty) {
  return Center(child: Text(noHomestay));
 } else {
  List<Map<String, dynamic>> homestays = snapshot.data!;
  return Padding(
   padding: const EdgeInsets.all(10),
   child: ListView.builder(
     itemCount: homestays.length,
     itemBuilder: (context, index) {
      var homestay = homestays[index];
      return HomestayTile(
       name: homestay['name'],
       totalbooking: homestay['totalBookings'].toString(),
       trailing: PopupMenuButton(
        onSelected: (value) {
         // your logic
        },
        itemBuilder: (BuildContext bc) {
         return [
          PopupMenuItem(
           value: '/detail',
           onTap: () {
           showDialog(
             context: context,
             builder: (BuildContext context) {
             return HomestayDetailDialog(
              homestayName: homestay['name'],
              homestayLoc: homestay['location'],
              );
             },
           );
          },
          child: Text(homestayDetail),
         ),
         PopupMenuItem(
          value: '/listbooking',
          onTap: () async {
           bool? result = await Navigator.push(
            context,
            MaterialPageRoute(
             builder: (context) => ListBookingScreen(
              homestayName: homestay['name'],
              homestayId: homestay['id'],
             ),
            ),
           );
           if (result == true) {
```

```
                        setState(() {});
                      }
                    },
                    child: Text(listBooking),
                  ),
                  PopupMenuItem(
                    value: '/listdoor',
                    onTap: () {
                    Navigator.push(
                        context,
                        MaterialPageRoute(
                          builder: (context) => ListDoorScreen(
                              homestayName:  homestay['name'],
                              homestayId: homestay['id'],
                            ),
                        ),
                      );
                    },
                    child: Text(listDoor),
                  )
                ];
              },
            ),
          );
        }),
      );
    }
  }));
 }
}
```

Homestay_detail_dialog.dart

```
import 'package:flutter/material.dart';
import 'package:phosphor_flutter/phosphor_flutter.dart';
import 'package:securitydoor/constant/text.dart';

import '../constant/color.dart';

class HomestayDetailDialog extends StatefulWidget {
  final String homestayName, homestayLoc;
  const HomestayDetailDialog(
      {super.key, required this.homestayName, required this.homestayLoc});

  @override
  State<HomestayDetailDialog> createState() => _HomestayDetailDialogState();
}
```

```
class _HomestayDetailDialogState extends State<HomestayDetailDialog> {
  @override
  Widget build(BuildContext context) {
    return PopScope(
      canPop: true,
      child: AlertDialog(
        shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(10)),
        backgroundColor: primaryColor,
        title: Text(
          widget.homestayName,
          style: textMediumSecondary.copyWith(fontSize: 16),
        ),
        content: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          mainAxisSize: MainAxisSize.min,
          children: [
            Row(
              children: [
                Icon(
                  PhosphorIcons.mapPin(),
                  size: 16,
                  color: secondaryColor,
                ),
                const SizedBox(
                  width: 5,
                ),
                Expanded(
                  child: Text(
                    widget.homestayLoc,
                    style: textRegularSecondary.copyWith(fontSize: 16),
                  ),
                ),
              ],
            ),
          ],
        ),
      ),
    );
  }
}
```

Booking_form_dialog.dart

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter/material.dart';
import 'package:phosphor_flutter/phosphor_flutter.dart';
import 'package:securitydoor/screen/qr_dialog.dart';

import '../constant/color.dart';
```

```
import '../constant/strings.dart';
import '../constant/text.dart';
import '../constant/widget/custom_button.dart';
import '../constant/widget/custom_textfield.dart';
import '../services/firebase_service.dart';

class BookingFormDialog extends StatefulWidget {
 final String homestayId, homestayName;

 const BookingFormDialog(
    {super.key, required this.homestayId, required this.homestayName});

 @override
 State<BookingFormDialog> createState() => _BookingFormDialogState();
}

class _BookingFormDialogState extends State<BookingFormDialog> {
 TextEditingController clientNameController = TextEditingController();
 TextEditingController clientIcNoController = TextEditingController();
 TextEditingController clientTelNoController = TextEditingController();
 TextEditingController checkInDateTimeController = TextEditingController();
 TextEditingController checkOutDateTimeController = TextEditingController();
 final _key = GlobalKey<FormState>();

 final FirebaseService _firebaseService = FirebaseService();

 Future<bool> _isDateAvailable(String homestayId, DateTime checkIn, DateTime
checkOut) async {
   QuerySnapshot bookingSnapshot = await FirebaseFirestore.instance
       .collection('Booking')
       .where('Homestay_ID', isEqualTo: homestayId)
       .get();

   for (var bookingDoc in bookingSnapshot.docs) {
     DateTime existingCheckIn = DateTime.parse(bookingDoc['CheckIn']);
     DateTime existingCheckOut = DateTime.parse(bookingDoc['CheckOut']);

     if (checkIn.isBefore(existingCheckOut) && checkOut.isAfter(existingCheckIn))
{
       return false;
     }
   }
   return true;
 }

 @override
 Widget build(BuildContext context) {
   return PopScope(
     canPop: false,
     child: AlertDialog(
```

```
shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(10)),
backgroundColor: primaryColor,
content: Form(
  key: _key,
  child: SingleChildScrollView(
    // Wrap the Column with SingleChildScrollView
    child: Container(
      padding: const EdgeInsets.all(10),
      child: Column(
        mainAxisSize: MainAxisSize.min,
        children: [
          Text(
            newBooking,
            style: textMediumSecondary.copyWith(fontSize: 20),
          ),
          const SizedBox(
            height: 20,
          ),
          CustomTextField(
            controller: clientNameController,
            hintText: clientName,
            keyboardType: TextInputType.text,
          ),
          CustomTextField(
            controller: clientIcNoController,
            hintText: icNumber,
            keyboardType: TextInputType.number,
          ),
          CustomTextField(
            controller: clientTelNoController,
            hintText: phoneNumber,
            keyboardType: TextInputType.phone,
          ),
          CustomTextField(
            controller: checkInDateTimeController,
            hintText: checkIn,
            readOnly: true,
            icon: PhosphorIcons.calendarBlank(),
            onPressedIconButton: () async {
              DateTime? pickedDate = await showDatePicker(
                context: context,
                initialDate: DateTime.now(),
                firstDate: DateTime(2000),
                lastDate: DateTime(2101),
              );
              if (pickedDate != null) {
                setState(() {
                  checkInDateTimeController.text =
                      "${pickedDate.toLocal()}".split(' ')[0];
                });
```

```
            }
          },
        ),
        CustomTextField(
          controller: checkOutDateTimeController,
          hintText: checkOut,
          readOnly: true,
          icon: PhosphorIcons.calendarBlank(),
          onPressedIconButton: () async {
            DateTime? pickedDate = await showDatePicker(
              context: context,
              initialDate: DateTime.now(),
              firstDate: DateTime(2000),
              lastDate: DateTime(2101),
            );
            if (pickedDate != null) {
              setState(() {
                checkOutDateTimeController.text =
                    "${pickedDate.toLocal()}".split(' ')[0];
              });
            }
          },
        ),
        const SizedBox(
          height: 30,
        ),
        Row(
          mainAxisSize: MainAxisSize.min,
          mainAxisAlignment: MainAxisAlignment.spaceBetween,
          children: [
            CustomButton(
              text: cancel,
              onPressed: () {
                Navigator.pop(context);
              }),
            CustomButton(
              text: submit,
              onPressed: () async {
                if (_key.currentState!.validate()) {
                  if (clientNameController.text.isNotEmpty &&
                      checkInDateTimeController.text.isNotEmpty &&
                      checkOutDateTimeController.text.isNotEmpty) {
                    DateTime checkInDate =
DateTime.parse(checkInDateTimeController.text);
                    DateTime checkOutDate =
DateTime.parse(checkOutDateTimeController.text);

                    bool isAvailable = await _isDateAvailable(widget.homestayId,
checkInDate, checkOutDate);
```

```
                    if (isAvailable) {
                      Map<String, String> bookingData = await
_firebaseService.createBooking(
                        clientName: clientNameController.text,
                        clientIcNo: clientIcNoController.text,
                        clientTelNo: clientTelNoController.text,
                        homestayId: widget.homestayId,
                        checkInDate: checkInDateTimeController.text,
                        checkOutDate: checkOutDateTimeController.text,
                      );

                      Navigator.pop(context, true);
                      showDialog(
                        context: context,
                        builder: (BuildContext context) {
                          return QRDialog(
                            qrId: bookingData['qrId']!,
                            bookingId: bookingData['bookingId']!,
                            homestayName: widget.homestayName,
                          );
                        },
                      );
                    } else {
                      ScaffoldMessenger.of(context).showSnackBar(
                        SnackBar(content: Text('Selected dates are not available.')),
                      );
                    }
                  }
                },
              ),
            ],
          )
        ],
      ),
    ),
  ),
 );
 }
}
```

Share_qr_controller.dart

```
import 'dart:io';
import 'dart:typed_data';

import 'package:flutter/material.dart';
import 'package:path_provider/path_provider.dart';
import 'package:qr_flutter/qr_flutter.dart';
import 'package:share_plus/share_plus.dart';

class ShareQrController {
  static Future<void> generateAndShareQRCode(String data, homestayName) async
{
    try {
      // Generate QR code as ByteData
      ByteData? qrBytes = await QrPainter(
      data: data,
       version: QrVersions.auto,
      gapless: true,
      emptyColor: Colors.white,
      ).toImageData(878);

      // Convert ByteData to Uint8List
      Uint8List pngBytes = qrBytes!.buffer.asUint8List();

      // Get temporary directory
      final tempDir = await getTemporaryDirectory();

      // Save the image as a temporary file
      final filePath = '${tempDir.path}/qr_code.png';
      final qrFile = File(filePath);
      await qrFile.writeAsBytes(pngBytes);

      // Share the file using ShareXFile
      Share.shareXFiles([XFile(filePath)], text: "$homestayName");
    } catch (e) {
      debugPrint("Error generating or sharing QR code: $e");
    }
  }
}
```

Booking_form_controller.dart

```
import 'package:uuid/uuid.dart';

class BookingFormController {
  static String generateRandomID() {
  var uuid = const Uuid();
    return uuid.v4();
```

```
  }
}
```

PSM2_B082110432

by Turnitin.my

Submission date: 13-Feb-2025 05:38PM (UTC+0900)
Submission ID: 2587429198
File name: PSM2_B082110432.pdf (2.69M)
Word count: 16158
Character count: 94504