

DEVELOPMENT OF A BOOK-TRACKING SYSTEM IN THE LIBRARY USING MICROCONTROLLER AND RFID

NUR FAZIMA BINTI CHE ADNAN



UNIVERSITI TEKNIKAL MALAYSIA MELAKA

DEVELOPMENT OF A BOOK-TRACKING SYSTEM IN THE LIBRARY USING A MICROCONTROLLER AND RFID

NUR FAZIMA BINTI CHE ADNAN

**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**

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2025

BORANG PENGESAHAN STATUS LAPORAN
PROJEK SARJANA MUDA II

Tajuk Projek : DEVELOPMENT OF A BOOK-TRACKING SYSTEM
IN THE LIBRARY USING A MICROCONTROLLER
AND RFID

Sesi Pengajian : 2024/2025

Saya NUR FAZIMA BINTI CHE ADNAN 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 : 8 February 2025

Tarikh : 08 Februari 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.



اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

DECLARATION

I declare that this project report entitled “Development of a Book Tracking System in a Library Using a Microcontroller and RFID” 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

:

NUR FAZIMA BINTI CHE ADNAN

Date

:

8 February 2025

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

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 : Ts. IMRAN BIN HINDUSTAN

Date : 8 Februari 2025

Signature :

Co-Supervisor :

Name (if any)

Date :

DEDICATION

This project entitled Development of a Book Tracking System in the Library Using a Microcontroller and RFID is dedicated to the following people and companies, who gave strong support and inspiration for the development of this thesis:

To my parents, for their unconditional love, support, and encouragement in all my endeavors. Your belief in me has been my greatest motivation. To my sisters and brother, thank you for your continuous encouragement and mental/daily support, and financial support, without which, the completion of this work would not be realized. I do thank you from the very bottom of my heart.

To my supervisor, Ts. Imran Bin Hindustan, thanks a lot for the contribution, moral and encouragement that helped me to complete this study successfully. Your insights, patience, and teachings have significantly contributed to my academic and personal growth.

Lastly, I am very thankful to my friends and housemates, who have been with me, giving me moral support and continuing their journey. Your help and encouragement have propelled me to complete this challenging journey without any hitches. I find this experience very enjoyable and fulfilling because with all of your supports, I am able to go out of the comfort zone.

ABSTRACT

Libraries play a crucial role in managing extensive collections of books and resources, but traditional book-tracking systems are often inefficient, error-prone, and labor-intensive. This project addresses these challenges by integrating RFID technology and the NodeMCU ESP8266 microcontroller to develop an automated and cost-effective book-tracking system. The system uses RFID tags attached to books, which are detected by RFID readers (MFRC522 modules). These readers communicate with the NodeMCU ESP8266, a microcontroller chosen for its built-in Wi-Fi capabilities, enabling real-time data transmission. The collected data is processed and stored in a MySQL database, hosted on a local server set up using XAMPP. A dynamic web platform, developed with PHP and managed through phpMyAdmin, allows librarians and users to track books' locations and access relevant details such as title, author, and timestamp.

The project's methodology involved not only hardware integration but also the development of software tools for efficient communication between components. Tools like the Arduino IDE were used to program the NodeMCU, while Visual Studio Code facilitated web development and debugging. Extensive testing was conducted to evaluate the system's performance, including tests for detection range, tag orientation, and reader interference. The results showed the system effectively tracks books, providing real-time updates on their locations. By automating inventory management, reducing manual effort, and integrating both hardware and software seamlessly, the project delivers a scalable and efficient solution to modernize library operations and enhance user satisfaction.

ABSTRAK

Perpustakaan memainkan peranan penting dalam menguruskan koleksi buku dan sumber yang besar, tetapi sistem penjejakan buku tradisional sering kali tidak cekap, terdedah kepada kesilapan, dan memerlukan banyak tenaga kerja. Projek ini menangani cabaran ini dengan mengintegrasikan teknologi RFID dan mikropengawal NodeMCU ESP8266 untuk membangunkan sistem penjejakan buku yang automatik dan kos efektif. Sistem ini menggunakan tag RFID yang dilekatkan pada buku, yang dikesan oleh pembaca RFID (modul MFRC522). Pembaca ini berkomunikasi dengan NodeMCU ESP8266, sebuah mikropengawal yang dipilih kerana keupayaan Wi-Fi terbina dalamnya, membolehkan penghantaran data masa nyata. Data yang dikumpulkan diproses dan disimpan dalam pangkalan data MySQL, yang dihoskan pada pelayan tempatan yang disediakan menggunakan XAMPP. Platform web dinamik, yang dibangunkan dengan PHP dan diuruskan melalui phpMyAdmin, membolehkan pustakawan dan pengguna menjejaki lokasi buku dan mengakses maklumat penting seperti tajuk, pengarang, dan cap masa.

Metodologi projek ini melibatkan bukan sahaja integrasi perkakasan tetapi juga pembangunan alat perisian untuk komunikasi yang cekap antara komponen. Alat seperti Arduino IDE digunakan untuk memprogram NodeMCU, manakala Visual Studio Code digunakan untuk pembangunan dan penyahpejatan laman web. Ujian yang komprehensif dijalankan untuk menilai prestasi sistem, termasuk ujian jarak pengesanan, orientasi tag, dan gangguan pembaca. Hasilnya menunjukkan sistem ini berfungsi dengan cekap untuk menjejaki buku dan menyediakan kemas kini lokasi masa nyata. Dengan mengautomatiskan pengurusan inventori, mengurangkan usaha manual, dan mengintegrasikan perkakasan dan

perisian dengan lancar, projek ini menawarkan penyelesaian yang skalabel dan cekap untuk memodenkan operasi perpustakaan serta meningkatkan kepuasan pengguna.



ACKNOWLEDGEMENTS

I am indebted to many people for seeing to the completion of my project, Book Tracking System Development in Library Using Microcontroller and RFID System.

Above all, I would like to take this opportunity to thank my supervisor, Ts. Imran Bin Hindustan, for his guidance, expertise, and patience during my whole journey of study, from which I benefited greatly. The insight, encouragement, and motivation you have given me throughout my research period have greatly contributed to my increase in academic knowledge and personal development.

Words would fail to express my deepest and sincere appreciation for the sacrifice of my parents. Your love, encouragement, and not giving up on me during this hard period have been a source of inspiration and motivation. Sisters and brother, whom I will always treasure, thank you for encouraging me always, mentally, and sometimes offering me that financial support during this project. It has been very fruitful for this work.

Special thanks go to my friends and housemates, who have been the most incredulous people for great support and cooperation all through the way. Your readiness to give your time and awake with me late at night during the testing sessions has remained instrumental. But for your company and encouragement, the road would have been harder to tread.

TABLE OF CONTENTS

| | PAGE |
|--|--------------|
| DECLARATION | |
| APPROVAL | |
| DEDICATIONS | |
| ABSTRACT | i |
| ABSTRAK | ii |
| ACKNOWLEDGEMENTS | iv |
| TABLE OF CONTENTS | v |
| LIST OF TABLES | viii |
| LIST OF FIGURES | ix |
| LIST OF SYMBOLS | xii |
| LIST OF ABBREVIATIONS | xiii |
| LIST OF APPENDICES | xiv |
| CHAPTER 1 INTRODUCTION | 1 |
| 1.1 Background | 1 |
| 1.2 Problem Statement | 2 |
| 1.3 Project Objective | 2 |
| 1.4 Scope of Project | 3 |
| CHAPTER 2 LITERATURE REVIEW | 4 |
| 2.1 Introduction | 4 |
| 2.2 Previous of Related Projects | 5 |
| 2.2.1 RFID Function | 5 |
| 2.2.2 Logic Structure Design | 6 |
| 2.2.3 Flowchart of the System | 7 |
| 2.2.4 Reader Module | 8 |
| 2.2.5 Maintaining Database and Notification | 9 |
| 2.2.6 Existing Model and Propose System Design | 10 |
| 2.2.7 Nodemcu and MFRC522 Wiring Diagram | 11 |
| 2.2.8 Comparison between Barcode, QR Code and RFID | 12 |
| 2.2.9 Block Diagram | 13 |
| 2.2.10 Overview of Library Automation | 15 |
| 2.3 Comparison of Previous Related Project | 16 |
| 2.4 Summary | 18 |

| | | |
|------------------|--|-----------|
| CHAPTER 3 | METHODOLOGY | 19 |
| 3.1 | Introduction | 19 |
| 3.2 | Selecting and Evaluating Tools for a Sustainable Development | 19 |
| 3.3 | Methodology | 21 |
| 3.3.1 | Project Planning | 22 |
| 3.3.2 | Elaboration of System Flow | 23 |
| 3.3.3 | Schedule | 25 |
| 3.4 | Equipment | 26 |
| 3.4.1 | Hardware Equipment | 26 |
| 3.4.1.1 | NFC Sticker | 26 |
| 3.4.1.2 | RFID Reader | 27 |
| 3.4.1.3 | NodeMCU ESP8266 | 28 |
| 3.4.1.4 | Wire Jumper | 29 |
| 3.4.1.5 | Breadboard | 30 |
| 3.4.2 | Software Requirement | 31 |
| 3.4.2.1 | Arduino IDE | 31 |
| 3.4.2.2 | Visual Studio Code | 32 |
| 3.4.2.3 | Phpmyadmin | 34 |
| 3.4.2.4 | Xampp | 35 |
| 3.4.2.5 | MySQL | 36 |
| 3.4.3 | Bill of Material (BOM) | 38 |
| 3.5 | Hardware Setup | 39 |
| 3.6 | Test Method | 41 |
| 3.6.1 | Wi-fi Connectivity Test | 41 |
| 3.6.2 | Server Communication Test | 41 |
| 3.6.3 | Edge Case Handling Test | 41 |
| 3.6.4 | Distance Test | 42 |
| 3.6.5 | Reader Overlap and Interference Test | 42 |
| 3.6.6 | Orientation Test | 43 |
| 3.7 | Summary | 45 |
| CHAPTER 4 | RESULTS AND DISCUSSIONS | 47 |
| 4.1 | Introduction | 47 |
| 4.2 | Results | 48 |
| 4.2.1 | Prototype | 48 |
| 4.2.2 | Software Development | 50 |
| 4.2.2.1 | Home Page | 50 |
| 4.2.2.2 | Registration | 51 |
| 4.2.2.3 | Admin Record | 52 |
| 4.2.2.4 | User Data Edit | 53 |
| 4.2.2.5 | User Data Delete | 53 |
| 4.2.2.6 | Book Display | 54 |
| 4.2.2.7 | Book Record | 55 |
| 4.2.3 | Database | 56 |
| 4.2.3.1 | table_nodemcu_rfidrc522_mysql | 56 |
| 4.2.3.2 | page_data | 57 |
| 4.2.4 | Coding | 58 |

| | | |
|-------------------|---|-----------|
| 4.2.4.1 | Arduino IDE | 58 |
| 4.2.4.2 | PHP Script | 59 |
| 4.2.6.1 | Serial Monitor Arduino IDE | 77 |
| 4.3 | Analysis | 78 |
| 4.3.1 | Data Testing | 78 |
| 4.3.1.1 | Wi-Fi Connectivity | 78 |
| 4.3.1.2 | Server Communication Test | 79 |
| 4.3.1.3 | Edge Case Handling Test | 80 |
| 4.3.1.4 | Distance Test between RFID Reader and NFC Sticker | 81 |
| 4.3.1.5 | Reader Overlap and Interference Test | 83 |
| 4.3.1.6 | Orientation Test | 85 |
| 4.3.2 | Challenge in Completing the Project | 87 |
| 4.3.2.1 | Arduino IDE Coding | 87 |
| 4.3.2.2 | Php Script | 88 |
| 4.4 | Summary | 88 |
| CHAPTER 5 | CONCLUSION AND RECOMMENDATIONS | 90 |
| 5.1 | Conclusion | 90 |
| 5.2 | Potential for Commercialization | 91 |
| 5.3 | Future Works | 91 |
| REFERENCES | | 93 |
| APPENDICES | | 96 |

LIST OF TABLES

| TABLE | TITLE | PAGE |
|--------------|---|-------------|
| Table 2.1 | Comparison between barcode, QR code and RFID. [8] | 12 |
| Table 2.2 | Comparison Table. | 16 |
| Table 3.1 | Gantt Chart of the Project | 25 |
| Table 3.2 | Bill of Material table | 38 |
| Table 3.3 | Hardware Connection | 40 |
| Table 4.1 | Wi-fi Test | 78 |
| Table 4.2 | Server Test | 79 |
| Table 4.3 | edge Test | 80 |
| Table 4.4 | Distance Test | 81 |
| Table 4.5 | Reader Overlap and Interference Test | 83 |
| Table 4.6 | Orientation Test | 85 |

LIST OF FIGURES

| FIGURE | TITLE | PAGE |
|---------------|---|-------------|
| Figure 1.1 | Block Diagram of the Project | 3 |
| Figure 2.1 | The Smart Library Management System [1] | 5 |
| Figure 2.2 | Entity Relational (E-R) diagram [2] | 6 |
| Figure 2.3 | Flow Diagram of the Proposed System [3] | 7 |
| Figure 2.4 | Circuit of Communication Interface [4] | 8 |
| Figure 2.5 | Flowchart for Maintaining Database and Notification [5] | 9 |
| Figure 2.6 | RFID Based Library Books Management Model and System Flow [6] | 10 |
| Figure 2.7 | Nodemcu and RFID Module Wiring Diagram [7] | 11 |
| Figure 2.8 | Smart Library Management System [9] | 13 |
| Figure 2.9 | Circuit Diagram [10] | 15 |
| Figure 3.1 | Project Planning | 22 |
| Figure 3.2 | Book Tracking Flowchart | 24 |
| Figure 3.3 | NFC Sticker | 26 |
| Figure 3.4 | RFID Reader | 27 |
| Figure 3.5 | ESP8266 | 28 |
| Figure 3.6 | Wire Jumper | 29 |
| Figure 3.7 | Breadboard | 30 |
| Figure 3.8 | Arduino IDE | 31 |
| Figure 3.9 | Visual Studio Code | 32 |
| Figure 3.10 | phpMyAdmin | 34 |
| Figure 3.11 | Xampp Control Panel | 35 |
| Figure 3.12 | MySQL 8.0 Command Line Client | 36 |

| | |
|--|----|
| Figure 3.13 Hardware Connection | 39 |
| Figure 3.14 0° orientation test | 43 |
| Figure 3.15 30° orientation test | 43 |
| Figure 3.16 60° orientation test | 44 |
| Figure 3.17 90° orientation test | 44 |
| Figure 4.2 Prototype | 48 |
| Figure 4.3 Inside Prototype | 49 |
| Figure 4.4 NFC Sticker Attached to the Book | 49 |
| Figure 4.5 Home Page | 50 |
| Figure 4.6 Registration | 51 |
| Figure 4.7 Book Data Table | 52 |
| Figure 4.8 Edit Book Data | 53 |
| Figure 4.9 Remove Book | 53 |
| Figure 4.10 Book Display | 54 |
| Figure 4.11 Book Record | 55 |
| Figure 4.12 Database table_nodemcu_rfidrc522_mysql | 56 |
| Figure 4.13 Database page_data | 57 |
| Figure 4.14 Book Detected at Shelf A | 70 |
| Figure 4.15 Registration Book | 71 |
| Figure 4.16 Book Data Table | 71 |
| Figure 4.17 Book Detected at Shelf B | 72 |
| Figure 4.18 Registration Book | 72 |
| Figure 4.19 Book Data Table | 73 |
| Figure 4.20 Book Data Table | 73 |
| Figure 4.21 Book Display | 74 |

| | |
|---|----|
| Figure 4.22 Book Display | 75 |
| Figure 4.23 Book Data Table | 75 |
| Figure 4.24 Book Data Table | 76 |
| Figure 4.25 Serial Monitor of Arduino Ide | 77 |
| Figure 4.26 Data Chart | 81 |
| Figure 4.27 Data Chart | 84 |
| Figure 4.28 Data Chart | 86 |



LIST OF SYMBOLS

◦ - Degree



LIST OF ABBREVIATIONS

| | | |
|----|---|------------|
| V | - | Voltage |
| Cm | - | Centimetre |



LIST OF APPENDICES

| APPENDIX | TITLE | PAGE |
|------------|---------------------------|------|
| Appendix A | Arduino IDE | 96 |
| Appendix B | Arduino IDE | 103 |
| Appendix C | Home.php | 105 |
| Appendix D | User data.php | 107 |
| Appendix E | User data edit page.php | 110 |
| Appendix F | User data edit tb.php | 112 |
| Appendix G | User data delete page.php | 113 |
| Appendix H | Registration.php | 115 |
| Appendix I | Read tag.php | 118 |
| Appendix J | Read tag user data.php | 123 |
| Appendix K | Book record.php | 125 |
| Appendix L | Database.php | 128 |
| Appendix M | getUID.php | 130 |
| Appendix N | insertDB.php | 130 |
| Appendix O | UIDContainer.php | 131 |
| Appendix P | readerContainer.php | 131 |
| Appendix Q | ESP8266 pintout | 132 |
| Appendix R | RFID reader pinout | 132 |
| Appendix S | Turnitin Report | 133 |

CHAPTER 1

INTRODUCTION

1.1 Background

Libraries are the custodians of the dissemination of knowledge. They possess great diversity in different books and resources. It becomes really overwhelming for large libraries to manage extensive book collections and keep an eye on them. The libraries also cannot manage laborious traditional systems for book tracking. Traditional book tracking systems in place are highly ineffective, full of inherent errors, and very labor-intensive. There is a high probability of books getting misplaced in the libraries; every so often, frustrated librarians wonder how the book managed to get out of their sight.

Digital technology has presented a tempting solution to the situation. Libraries can enhance their inventory management capacity by integrating Radio Frequency Identification (RFID) technology with modern microcontrollers. Tags, containing embedded RFID tags, when pasted on books could be sensed through RFID readers placed all over the library. On sensing an RFID, the reader passes the information to a microcontroller, like NodeMCU ESP8266, which then updates a web-based inventory system in real-time. The system will help precisely track the locations of books, save manual effort, and increase customer satisfaction.

The implementation of this system in a nutshell includes steps, such as tagging the books with RFID tags, getting the RFID readers up and running, programming the microcontroller, setting up the web-platform, and comprehensive testing to make sure the system is robust and accurate. The expected outcomes in terms of increased efficiency in inventory

management, real-time update on book locations, reduced time for both employees and users in book location, improved system robustness, and man-hour manual management relief.

1.2 Problem Statement

The problem statement of this project revolves around the lack of a book tracking mechanism, especially in large libraries. With hundreds or even thousands of books spread across various shelves and sections, keeping track of each book's location becomes a daunting task. This leads to situations where the librarian has to manually search the library to find misplaced books, often relying on memory or incomplete records. Additionally, the whereabouts of books are not recorded in the library system, causing visitors difficulties in accessing the materials they need, leading to frustration, wasted time, and decreased user satisfaction.

1.3 Project Objective

The objective of this project is to develop a cost-effective and efficient library management system that uses RFID technology and NodeMCU ESP8266 to:

- a) Construct a book tracking system consisting of an RFID tag, RFID reader, microcontroller and web-platform
- b) To validate the data acquisition and data recording
- c) To analyze the performance of the project.

1.4 Scope of Project

This project, "Development of a Book Tracking System in the Library Using a Microcontroller and RFID" addresses the challenges of book tracking in a large library setting. This project presents the design, development, and implementation of a system that uses RFID technologies and an NodeMCU ESP8266 for efficient library operations. The scopes of the project include the following:

- a) RFID tags. In this project, 4 RFID tags that will attach to the front of the page of books. These tags will act as a sensor that will communicate with the RFID reader
- b) RFID reader. 2 RFID reader that will place separately. This reader will receive the signal from RFID tags and update the data on the web page.
- c) NodeMCU. Will act as interface between RFID reader and webpage. Displaying the book status and display the where the misplace book location
- d) Website. This website will display updated book receive from RFID reader, helping the librarian and visitors to keep track of the book location efficiently.

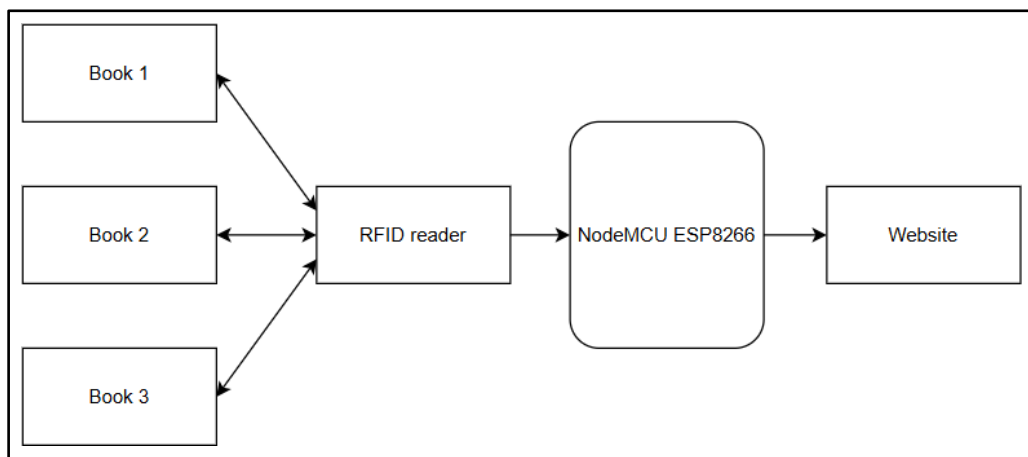


Figure 1.1 Block Diagram of the Project

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Today, in the digital age, traditional library systems need to be upgraded by integrating modern technologies. Libraries, being knowledge repositories, cannot escape this development. The mainstream library management systems currently in place, whose most organizational operations are either manual or through barcode technology, grapple with time-consuming inventory management, inaccuracy in book tracking, inefficiency in user service delivery, etc. This study first seeks to highlight the existing research in this area of RFID and microcontroller advancement, paying strict heed to its application in library management systems. Through a critical analysis of the benefits and deficiencies of current systems, and the technological needs for a devised comprehensive book tracking solution, the project lays the foundation for the development of an innovative solution that will cater to the changing needs of growing modern libraries.

2.2 Previous of Related Projects

2.2.1 RFID Function

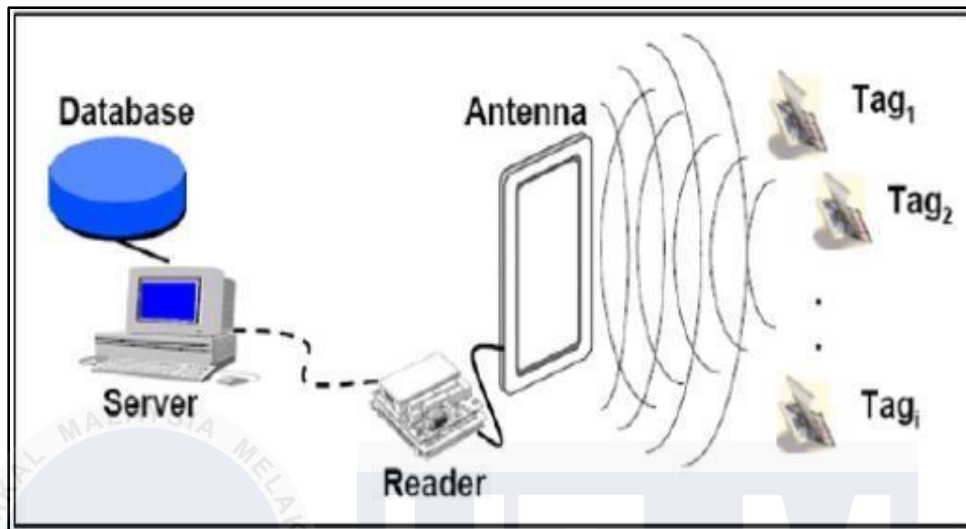


Figure 2.1 The Smart Library Management System [1]

According to [1], An RFID system consists of a reader containing a transmitter, receiver, and an antenna and a transponder tag containing the antenna. The transmitter sends a radio signal that the tag, if in its range receives and responds to by sending a response. The receiver receives the response. The modern RFID tags are usually batteryless and derive power from the radio signal of the transmitter, making them economical and environmentally friendly. These tags can be put in different materials including books, and each of them has a unique code that talks in the circulation database of the library to inform on the location of the materials. The library can then locate misfiled items by scanning the shelves with hand-held readers that use radio pulses. A chip is powered whenever it passes a reading station by the radio field, allowing the reading of information on the chips of the materials tagged for the library.

2.2.2 Logic Structure Design

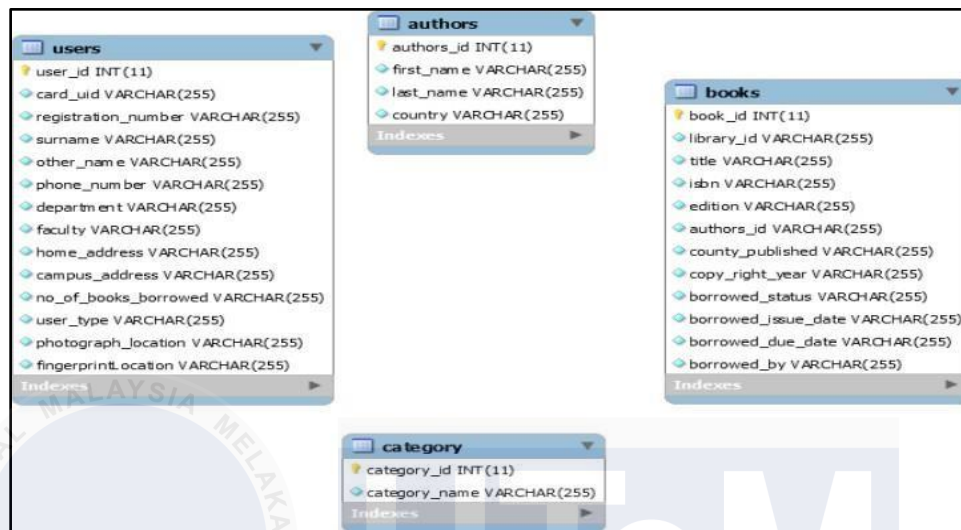


Figure 2.2 Entity Relational (E-R) diagram [2]

The Entity Relational Model (ERM) from [2] is designed to ensure order and consistency within the database of the library, avoiding any related chaos. In this model, the book in the library is uniquely identified and categorized based on its ISBN. These books have Attributes such as Title, Copyright Year, and Edition, whereas categories are identified by Category IDs and Category Names. Authors, on the other hand, are uniquely identified by Author IDs, and their details include First Name, Last Name, and Country. Users in the library are basically of two types: Student and Staff. These two types of members are registered and also have attributes such as RFID Card ID and Mobile Phone. The access rights are identified in terms of Privilege IDs and names, and they are used to control user access. Further, there is a specific time period allocated based on the type of user, that is, whether the user is a Student or Staff. Borrowing is also carefully designed in the form of associating entities identified as "Borrowed," which also details some of the key Identifiers such as Issued Date

and Due Date. The status of the book is decided by Status IDs and some of its attributes, such as Status Name and RFID Tag Number.

2.2.3 Flowchart of the System

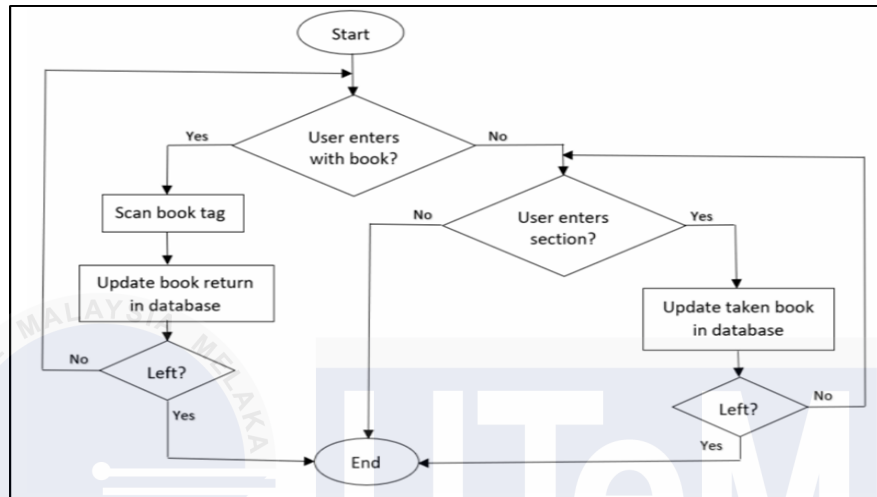


Figure 2.3 Flow Diagram of the Proposed System [3]

Dharani Devi, P., Mirudhula, S., & Devi, A. [3] emphasize the crucial need to encourage the habit of reading in students. They point out how habitual reading not only promotes understanding among students in other subjects but also improves vocabulary and enhances overall learning. Moreover, reading literature from established authors helps students gain experience as well as expertise. Libraries have maintained their existence since ancient times, when clay tablet archives were in use about 4,000 years ago. The British Museum Library and the Bodleian Library are among such libraries. The invention of digital libraries has not necessarily curtailed the significance of the library of materials for those seeking self-learning. It still serves as a place for exploring oneself and spending free time productively. It is laborious to maintain a library, as the number of inventory and transaction records is humongous, and librarians are required to maintain a catalog and records. Though smart systems and software for libraries are developed for this purpose, the need for human input does exist, as challenges such as the failure of electronic devices do occur.

2.2.4 Reader Module

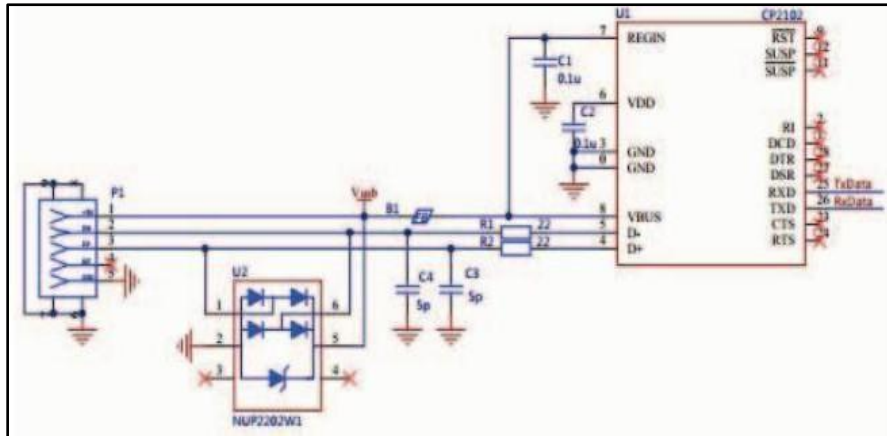


Figure 2.4 Circuit of Communication Interface [4]

According to He He, Tao Liu and Ershen Wang [4], the Reader Module is designed for positioning books and statistical applications. The RFID chip PR9000 is adopted in this module. This chip is characterized by its fast reading and writing capabilities and high speed in line with the ISO18000-6C standard. This chip implements a high- performance RF section, a sufficient memory, a baseband processor, and an improved 8052 microcontroller, which greatly reduces the peripheral component. This module uses the CP2102 chip for the PC communication. The CP2102 chip is a USB to UART bridge. The device not only converts USB to RS232 but also integrates the USB 2.0 full-speed function controller and transceiver crystal, EEPROM, and UART. It does not require adding any USB transceiver outside to support full modem signal. Considering some shortcomings of the CP2102 chip, the current limiting resistor R1 and R2 are added to the design. The communication interface circuit is as shown in figure 2.4.

2.2.5 Maintaining Database and Notification

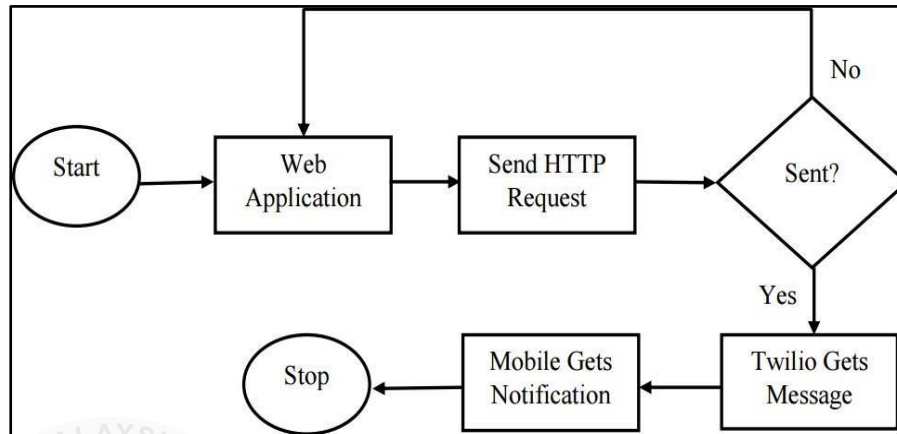


Figure 2.5 Flowchart for Maintaining Database and Notification [5]

The Reader Module functions within the framework of a Library Management System (LMS) by Rujan Khadgi, Subin Dangol, Sujun Lamsal and Suman Shrestha [5], maintaining a database that enables users and administrators to communicate through mobile devices by sending and receiving text notifications. This system utilizes Twilio's REST API platform, which serves as a communication bridge between the web application and mobile devices. When the web application sends an HTTP request to Twilio, Twilio responds and sends the requested message to the user. A REST API (Representational State Transfer Application Programming Interface) facilitates consistent data and functionality exchange over the internet, typically accessed via HTTP protocol at predefined URLs.

2.2.6 Existing Model and Propose System Design

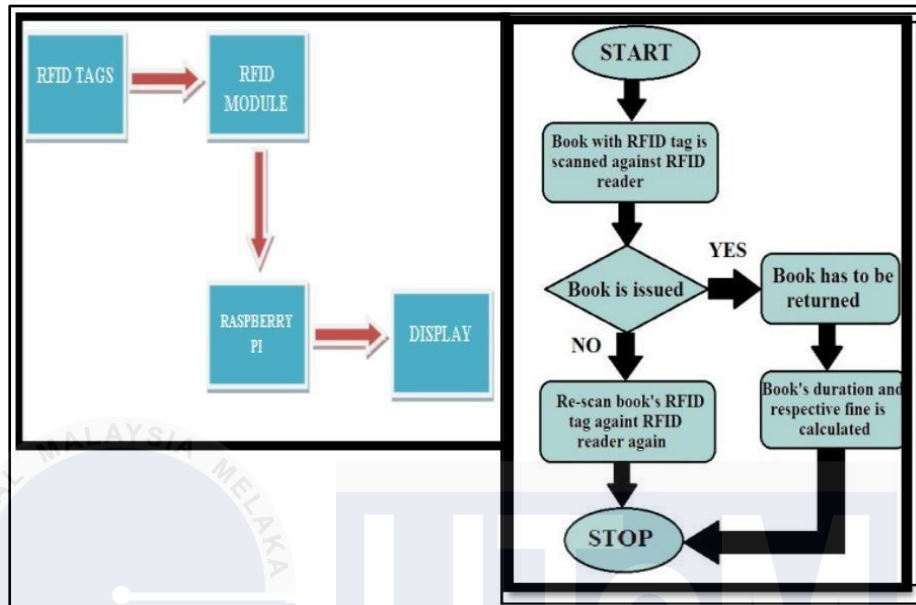


Figure 2.6 RFID Based Library Books Management Model and System Flow [6]

RFID technology has replaced bar-coding in libraries, using RFID tags, readers, and a Raspberry Pi. Each book has an RFID tag with unique information, which the reader scans and displays on screens. The Raspberry Pi handles borrowing and returning books, updating the main database in Firebase efficiently. The system scans books, updates their status, and notifies staff about fines and notifications. It analyzes data to provide insights into book circulation and inventory, predicts which books to remove based on usage, and organizes book placements to avoid redundancy. Additional features include tracking book availability, managing checkouts and returns, user registration via a web page, predicting book popularity, and simplifying fine calculations, greatly improving library management.

2.2.7 Nodemcu and MFRC522 Wiring Diagram

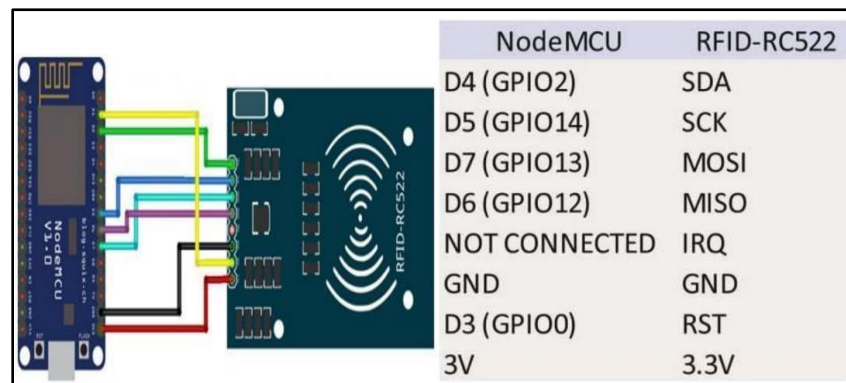


Figure 2.7 Nodemcu and RFID Module Wiring Diagram [7]

According to [7] NodeMCU and MFRC522 wiring diagram, the VCC pin is responsible for supplying power to the module, which operates within a voltage range of 2.5 to 3.3 volts. It can be connected directly to the 3.3V output pin of the NodeMCU microcontroller. The RST pin serves as an input for resetting and powering down the module. When this pin is set to low, it enables power-down mode, turning off all internal current sinks and disconnecting the module's input pins from external connections. The GND pin is the module's ground connection and should be linked to the ground pin of the NodeMCU microcontroller. The IRQ pin functions as an interrupt, alerting the microcontroller to wake up the module when an RFID tag enters its range, thus enabling the module to enter sleep mode to conserve power.

Depending on the type of the user interface, the MISO/SCL/Tx pin has a SPI role of Master-In-Slave-Out, an I2C serial clock, and a UART serial data output. MOSI allows communication from Master to Slave in the SPI mode. SCK is the Serial Clock pin and receives the clock pulse from the SPI bus Master, which is the NodeMCU in this case. SS/SDA/RX is again a triple-role pin: during SPI communication, the Arduino will consider

this pin as a Serial Input (SS), during I2C it will be an SDA line, and during UART it will be Rx. Types of data are transmitted using General Purpose Input/Output (GPIO) pins; in this case, those are the D3 to D7 pins on the NodeMCU.

2.2.8 Comparison between Barcode, QR Code and RFID

| Items | Barcode | QR Code | RFID |
|-----------------------|----------------------|----------------------|--|
| Technology | Laser scanner | Optical scanner | Radio frequency |
| Range | Few inches to a foot | Few inches to a foot | 30 feet or more (passive) 60 to 300 feet (active) |
| Cost | Low | Low | Low (passive) High (active) |
| Accuracy | Accurate | Accurate | Accurate |
| Line of sight | Required | Required | Not required |
| Information Collected | Fast | Fast | Extremely fast |
| Capability to scan | One item | One item | Multiple items |
| Automation | Human operator | Human operator | Only fixed reader |

Table 2.1 Comparison between barcode, QR code and RFID. [8]

According to table 1 in [8], The comparison of technologies such as Barcode, QR code, and RFID for library systems demonstrates that RFID is the most efficient option, especially passive RFID tags, which are cost-effective, compact, and battery-free. These tags are particularly suitable for library applications as they enable scanning multiple books simultaneously with a fixed reader. Tools like Node-RED further enhance the functionality of such systems. Node-RED, an open-source IoT platform, is widely used for visual programming, data analytics, and dashboard integration. It supports various applications, including library management, by providing features such as data monitoring, alarming, and remote access via web browsers. Additionally, Fusion360, a CAD software, is utilized in system design for modeling, simulation, and optimization, adding value in areas like robotics and manufacturing.

In Malaysia, research on library management systems highlights the need for integrated hardware and software solutions but also reveals challenges such as cost, limited tracking capabilities, and portability. For instance, RFID-based systems have been proposed for theft prevention and book tracking but face high implementation costs. QR-based systems focus on book borrowing but are limited to single-book scanning without data tracking capabilities. Shelf management systems using RFID are practical but lack GUIs and portability, while barcode-based systems primarily focus on computerizing manual processes and are limited to software-only solutions. These studies suggest that leveraging RFID alongside advanced tools like Node-RED can lead to more comprehensive and user-friendly library systems.

2.2.9 Block Diagram

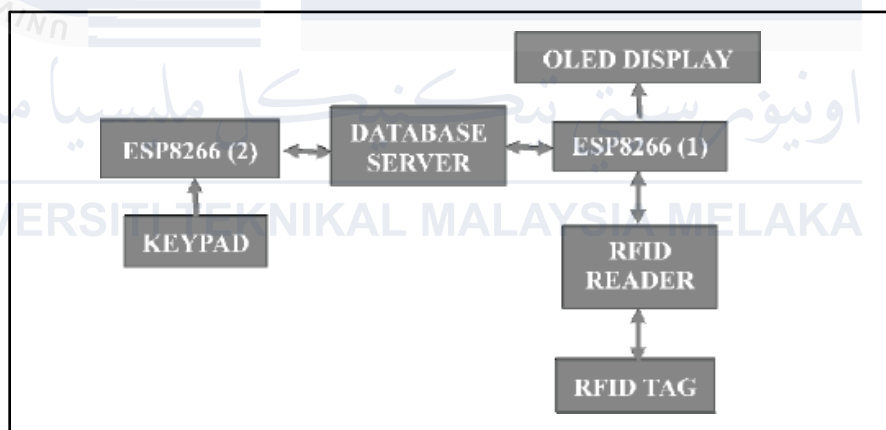


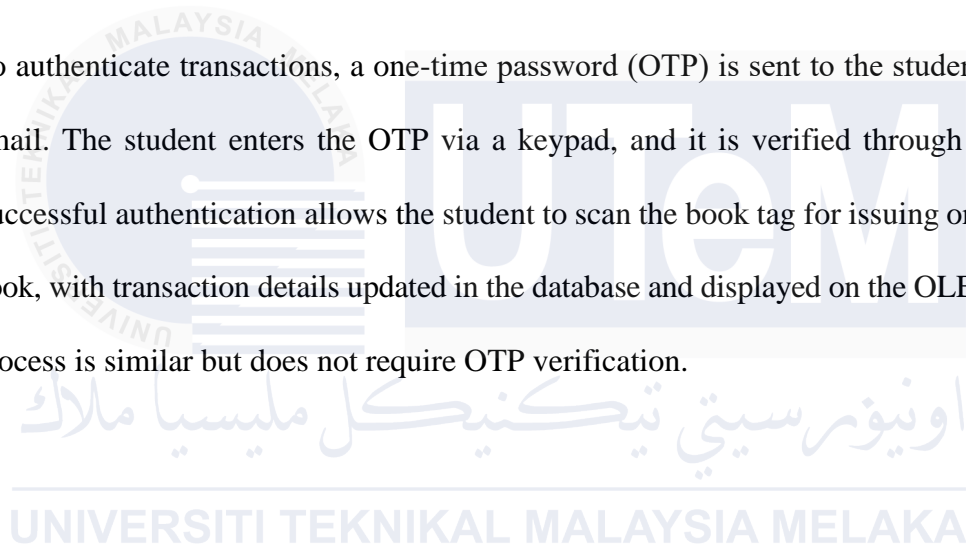
Figure 2.8 Smart Library Management System [9]

The block diagram from [9] illustrates the smart RFID library management system, which comprises two ESP8266 microcontrollers, a keyboard, an OLED display, an RFID reader, and RFID tags. The system is designed to manage library operations, including storing book details, issuing and returning books, and enabling internet access using individual user IDs. The ESP8266 serves as both the microcontroller and the Wi-Fi module, while a MySQL

database server manages data storage and updates, organized into tables like books, students, admin, issue, return, and fine, with SQL facilitating database operations.

The system incorporates a high-resolution OLED display, which initially shows the NIT Puducherry logo when powered on. The RFID reader detects unique tag IDs for student cards and book labels. When a student tag is scanned, details such as name, roll number, and email are retrieved and sent to the ESP8266, which forwards them to the MySQL database for verification. The student's name and roll number are then displayed on the OLED.

To authenticate transactions, a one-time password (OTP) is sent to the student's registered email. The student enters the OTP via a keypad, and it is verified through the database. Successful authentication allows the student to scan the book tag for issuing or returning the book, with transaction details updated in the database and displayed on the OLED. The return process is similar but does not require OTP verification.



2.2.10 Overview of Library Automation

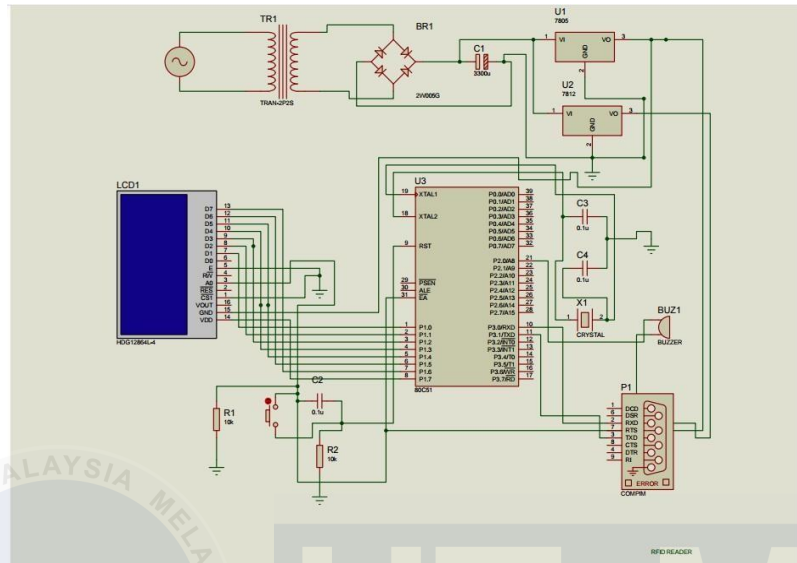


Figure 2.9 Circuit Diagram [10]

To issue a book from the library, a student inserts their smart card into a Smart Card Reader. The system creates by [10], using a microcontroller, verifies the student's membership through serial communication. If the card is valid, the student can proceed; if not, a buzzer sound. The RFID-tagged book is scanned, and its details are captured by the microcontroller, displayed on an LCD, and stored in EEPROM. When returning the book, the RFID reader verifies its validity, with a buzzer sounding if there is an issue.

The system has three main sections: the microcontroller section, the power supply section, and the DC regulated power supply section. The microcontroller section includes an 89C51 microcontroller and an 11.0592 MHz crystal oscillator, operating on a +5V DC power supply. The RFID module connects to the microcontroller via an RS232 serial port, facilitated by a MAX232 driver. The LCD interfaced with the microcontroller displays information about issued books.

2.3 Comparison of Previous Related Project

Table 2.2 Comparison Table.

| No | Reference | Similarity | Difference | Remark |
|----|-----------|--|---|---------------------------------|
| 1 | [1] | Use RFID for book-tracking and using web-based interface for interaction | Include borrowing and returning books and using IoT | Easy to use since using IoT |
| 2 | [2] | Use RFID for book-tracking and using web-based interface for interaction | This project used biometric technology, buzzer and IoT | Enhance security to avoid theft |
| 3 | [3] | Use RFID for book-tracking and using web-based interface for interaction | This article uses RFID tags attached to the identity card of every individual user including borrowing and returning the book using IoT | Easy to use since using IoT |
| 4 | [4] | Use RFID for book-tracking and using web-based interface for interaction | user-searching books using inventory system | User friendly |
| 5 | [5] | Use RFID for book-tracking and using web-based interface for interaction | Use an alarm system to avoid theft | Enhance security to avoid theft |

| | | | | |
|----|------|--|---|---|
| 6 | [6] | Use RFID for book-tracking and using web-based interface for interaction | Use Raspberry Pi and including borrowing and returning books in the system | The component use is expensive |
| 7 | [7] | Use RFID for book-tracking and using web-based interface for interaction | Include borrowing and returning books for user and searching books using inventory system | User-friendly |
| 8 | [8] | Use RFID for book tracking, real-time update to database that also use web-based interface for interaction. This reference also used centralized database, MySQL | Use Node-RED for IoT integration, emphasizes reader angle optimization for performance, portable RFID reader and complex IoT-based architecture | Both seek to reduce manual effort and provide accurate and real time book location data |
| 9 | [9] | Use RFID for book tracking, real time update to a database, web-based interface for interaction and used centralized database, MySQL | Use feature-rich web portal with feedback, fine calculation, and trend analysis, OTP verification for security and emphasizes IoT-based automation and data visualization with google chart | Both provide web-base access to manage book |
| 10 | [10] | Use RFID to manage the book and inventory system | Use an alarm system to avoid theft | Complex circuit |

2.4 Summary

From this chapter, it is evident that integrating RFID technology and microcontroller advancements into modern library management systems addresses key challenges such as inefficiency in inventory management, inaccuracies in book tracking, and slow service delivery. RFID systems, utilizing tags and readers, enable precise book tracking, simultaneous scanning, and real-time data updates, significantly enhancing library operations.

The systems reviewed incorporate components like ESP8266 microcontrollers and MySQL databases to facilitate efficient book issuing, returning, and fine management while maintaining portability and cost-effectiveness. Comparative analyses confirm RFID's superiority over barcodes and QR codes in terms of efficiency, scalability, and automation. Additionally, the chapter highlights IoT-enabled solutions, such as Node-RED integration and web-based portals, which improve user interaction and introduce advanced security measures like OTP verification. Despite challenges such as high implementation costs and hardware limitations, the adoption of IoT, predictive analytics, and sophisticated software tools demonstrates the potential to transform libraries into fully automated, user-friendly systems that align with the demands of the digital age.

CHAPTER 3

METHODOLOGY

3.1 Introduction

The methodology section of the project, "Development of a Book Tracking System in the Library Using a Microcontroller and RFID" provides a guideline for the implementation of the proposed solution. It details the procedures involved in the creation and implementation of a book tracking system. The general objective of the methodology section is to link the project's objectives with a concise manner in which library goals can be systematically achieved using microcontroller technology and RFID. The methodology has been orchestrated to provide the project team with a clear roadmap that navigates them through the theoretical research, prototyping, testing, and iterative refinement processes, ultimately leading to the production of a solution relevant to the user requirement in current library management systems.

3.2 Selecting and Evaluating Tools for a Sustainable Development

When implementing a library book tracking system using a microcontroller and RFID, careful consideration is necessary to ensure sustainability, efficiency, and reliability. The choice of tools is guided by their purpose, compatibility, affordability, maintainability, scalability, and ease of use. RFID readers and tags are ideal for this system as they uniquely identify books using distinct frequencies, making them reliable for handling a variety of book volumes and transactions.

The NodeMCU ESP8266 was selected for this project instead of Arduino due to its built-in Wi-Fi capabilities, making it more suitable for IoT-based applications. It is highly cost-effective, readily available, and widely supported by an active user community, which provides robust documentation and resources. This microcontroller allows seamless integration with the RFID RC522 reader, enabling communication between the RFID reader and the library database.

For programming and development, Visual Studio Code was used due to its flexibility, extensive library support, and compatibility with the ESP8266 platform. The database and web application were developed using MySQL and XAMPP, which provide a reliable environment for database management and hosting. These tools work in tandem to create a dynamic website that displays book details such as BookID, Title, Author, Date, Time, and Location.

The NodeMCU ESP8266 reads the UID from the RFID RC522 reader and appends metadata such as timestamps and locations before transmitting this data to the web application. The website, developed with tools like PHP and hosted on XAMPP, updates in real time to reflect any changes in a book's location or status. This system ensures efficient inventory management, scalability, and ease of use, meeting the library's operational requirements and goals.

3.3 Methodology

The steps used in the creation of a book tracking system in a library using RFID and a microcontroller can be highlighted as follows. The equipment used in the system includes RFID tags affixed to the books, RFID readers placed at strategic points within the library, NodeMCU ESP8266 analyzing the received information, and the web-based inventory tracking platform. This includes placing tags on the books, installing the RFID readers, writing firmware for the microcontroller, designing the web interface, and conducting extensive testing. It should be noted that this system aims to increase the speed of book tracking, minimize the need for employee intervention, and enhance library's visitor satisfaction by providing detailed information about the exact location of books. The system comprises RFID tags and readers, the NodeMCU ESP8266, computers, and networks for communication. The inventory platform is developed using web development tools, while the microcontroller firmware is programmed specifically for the NodeMCU ESP8266 to handle real-time communication between the RFID readers and the web-based system. This methodology simplifies library operations and enhances service delivery to library patrons.

3.3.1 Project Planning

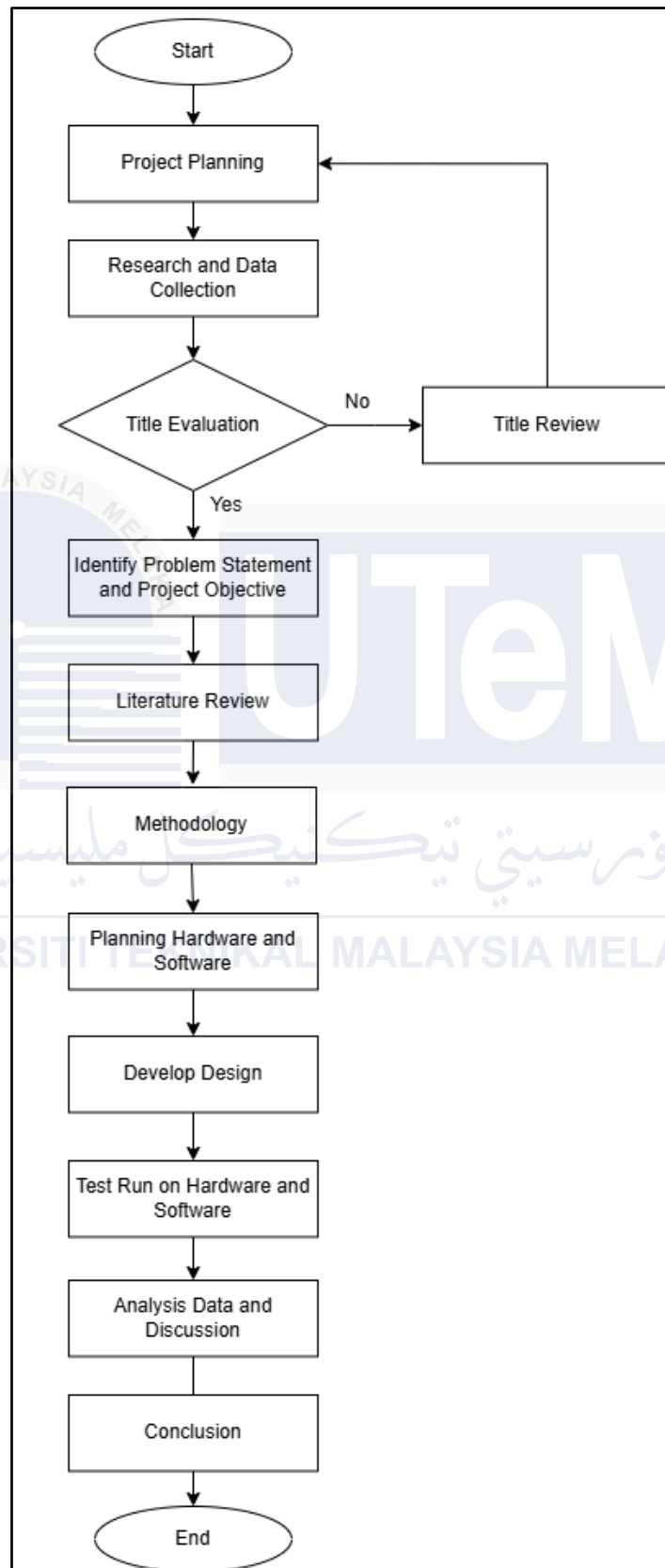


Figure 3.1 Project Planning

Figure 3.1 illustrates the overall project planning framework, outlining the sequential steps to estimate and guide the project's progression within the project environment.

3.3.2 Elaboration of System Flow

The flowchart helps to assist the developer find some part of project that can be improve and determine what and where the modification can be done to enhance the performance of the system. Base on figure 3.2 below, the flowchart shows how the overall system intended for tracing the book. The RFID tag will be place on the book. Inside RFID tag, there are small chip that store information and antenna that help tag to communicate with RFID reader. Then the RFID reader that is connected to microcontroller, emit radio signal. When the RFID tag enters the signal range, it gets power up. The RFID tag send its stored information to RFID reader. After that the RFID reader will decode it and send to system for use. The

microcontroller will receive the UID from RFID reader and determine the timestamp and location of the book and send this information to database.

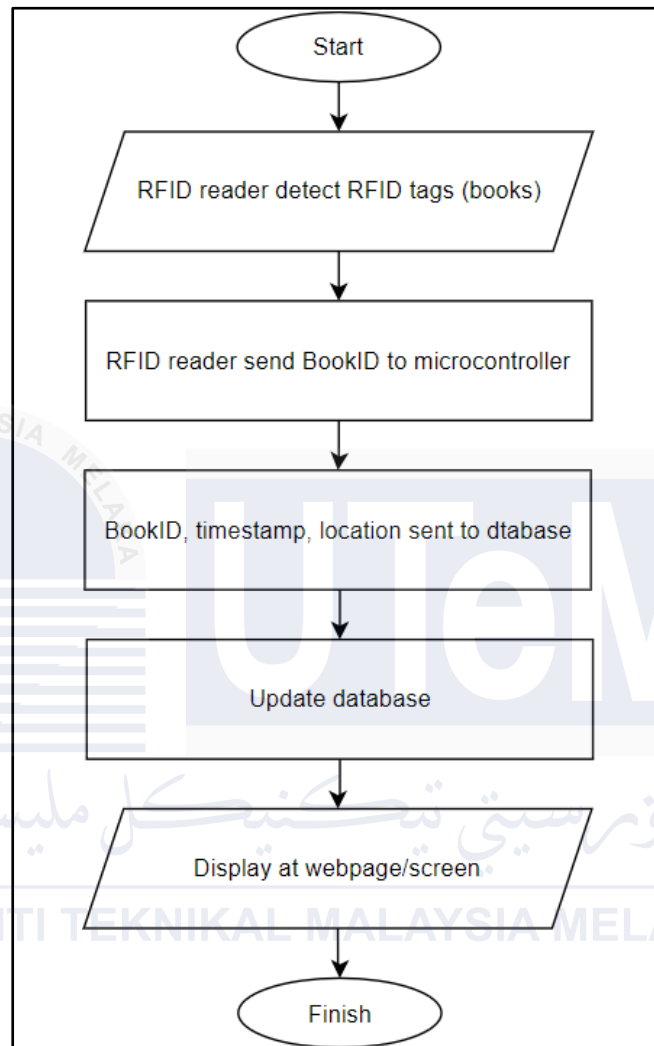


Figure 3.2 Book Tracking Flowchart

3.3.3 Schedule

| Activities | Project Planning PSM2 Week | | | | | | | | | | | | | | |
|---------------------------------------|-------------------------------|---|---|---|---|---|---|---|---|----|----|----|----|----|----|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| Final year project briefing by JK PSM | | | | | | | | | | | | | | | |
| Planning and Research | | | | | | | | | | | | | | | |
| Define Requirement and Feature | | | | | | | | | | | | | | | |
| Research Component | | | | | | | | | | | | | | | |
| Circuit Assembly | | | | | | | | | | | | | | | |
| Verify Circuit | | | | | | | | | | | | | | | |
| Implement project | | | | | | | | | | | | | | | |
| Develop Webpage | | | | | | | | | | | | | | | |
| Integrating Hardware and Software | | | | | | | | | | | | | | | |
| Functional Testing | | | | | | | | | | | | | | | |
| Preliminary Result | | | | | | | | | | | | | | | |
| Performance Testing | | | | | | | | | | | | | | | |
| Data Collection and Analysis | | | | | | | | | | | | | | | |
| Update Report | | | | | | | | | | | | | | | |
| Submit report to panel | | | | | | | | | | | | | | | |
| BDP presentation | | | | | | | | | | | | | | | |
| Submit full report | | | | | | | | | | | | | | | |

Table 3.1 Gantt Chart of the Project

The Gantt chart is utilized to allocate an appropriate time frame for all project workflow tasks. Table 3.1 presents the Gantt chart for the project. Proper scheduling of activities is

crucial to prevent delays and ensure timely project completion. Furthermore, effective planning and scheduling can enhance overall productivity.

3.4 Equipment

3.4.1 Hardware Equipment

3.4.1.1 NFC Sticker

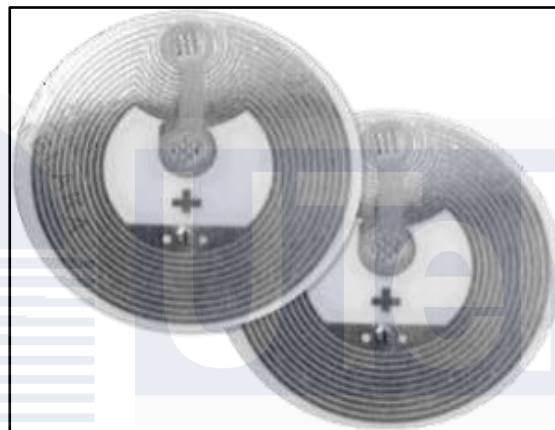


Figure 3.3 NFC Sticker

An NFC sticker, Figure 3.3 is a small, adhesive-backed tag embedded with a Near Field Communication (NFC) chip and an antenna, enabling wireless communication when brought close to an NFC-enabled device, such as a smartphone or an RFID reader. These stickers are thin and flexible, making them easy to attach to various surfaces like books, cards, posters, or packaging. The embedded NFC chip stores data, which can include unique identifiers (UIDs), URLs, text, or commands, and this data can be read or written by compatible devices within a short range (typically 1-4 cm).

NFC stickers are powered passively, meaning they do not require a battery, instead, they draw power from the electromagnetic field generated by the reader device during interaction. This makes them cost-effective, maintenance-free, and ideal for numerous applications.

Common uses include contactless payment systems, access control, inventory tracking, and enhancing product interactivity through smart labels. Their compact design and programmability make NFC stickers versatile tools for enabling seamless and innovative digital interactions.

3.4.1.2 RFID Reader



Figure 3.4 RFID Reader

An RFID reader, Figure 3.4 is an electronic device designed to communicate with RFID (Radio Frequency Identification) tags to read or write data stored on them. It consists of an antenna and a transceiver that emits radio waves to power and interact with RFID tags within its range. When an RFID tag enters the reader's electromagnetic field, the tag is activated and transmits its unique identifier (UID) or other stored information back to the reader. RFID readers can operate at different frequencies, including low frequency (LF), high frequency (HF), and ultra-high frequency (UHF), depending on the application and range requirements.

RFID readers come in various forms, including handheld devices, fixed-mounted units, and embedded modules for integration into larger systems. They are widely used in applications such as access control, inventory management, supply chain tracking, and contactless payment systems. Some RFID readers, like the MFRC522, are compact and suitable for DIY projects and small-scale implementations. These readers are often interfaced with

microcontrollers like Arduino or ESP8266 to build custom solutions. Overall, RFID readers provide a reliable, fast, and efficient way to identify and track objects wirelessly, making them essential in modern identification and automation systems.

3.4.1.3 NodeMCU ESP8266

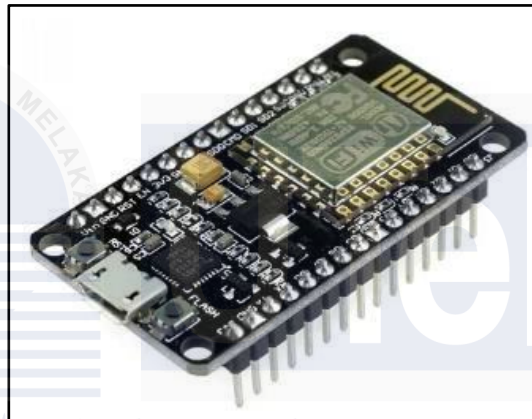


Figure 3.5 ESP8266

The NodeMCU ESP8266, Figure 3.5 is a compact, low-cost microcontroller board based on the ESP8266 Wi-Fi module, designed for IoT (Internet of Things) applications. It combines a powerful 32-bit microcontroller with integrated Wi-Fi connectivity, making it an ideal choice for building connected devices. The board features GPIO pins that support digital and analog input/output, enabling the connection of sensors, actuators, and other peripherals. Its compact size and built-in Wi-Fi allow it to communicate wirelessly with other devices, access cloud services, or act as a web server.

The NodeMCU ESP8266 is programmed using the Arduino IDE, Lua script, or other platforms, offering flexibility and ease of use for beginners and professionals alike. It includes essential components like a USB-to-serial converter for easy programming, a 3.3V

voltage regulator, and reset/flash buttons. Its versatility makes it popular in applications like home automation, weather monitoring, smart appliances, and more. With its affordability and robust feature set, the NodeMCU ESP8266 is a go-to solution for developing IoT projects.

3.4.1.4 Wire Jumper



Figure 3.6 Wire Jumper

Jumper wires, Figure 3.6 are insulated electrical wires with connectors or pins at each end, designed to create temporary or permanent connections between components on a breadboard, microcontroller, or other electronic devices. They come in three types: male-to-male, male-to-female, and female-to-female, allowing versatile connections between pins, headers, or terminals. Jumper wires are typically made of thin, flexible copper conductors encased in plastic insulation, ensuring durability and ease of use in prototyping.

These wires are essential tools in electronics projects, enabling the transfer of signals or power without the need for soldering. Their compatibility with breadboards and development boards like Arduino and NodeMCU makes them invaluable for testing and debugging circuits. Jumper wires are color-coded to help identify connections easily, simplifying the process of designing and troubleshooting complex circuits. Their reusability

and simplicity make them a fundamental component in electronics prototyping and experimentation.

3.4.1.5 Breadboard

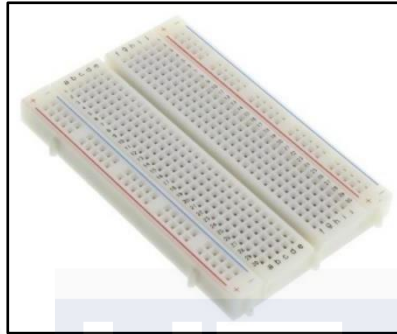


Figure 3.7 Breadboard

A breadboard, Figure 3.7 is a reusable platform used for prototyping and testing electronic circuits without soldering. It features a grid of interconnected holes that allow components such as resistors, capacitors, ICs, and jumper wires to be easily inserted and connected. The breadboard is divided into sections: power rails, which run horizontally along the edges and are used for distributing power and ground, and terminal strips, which consist of vertical rows of connected holes where components are placed. A central gap separates the terminal strips, providing space to insert integrated circuits (ICs) without shorting their pins.

The breadboard's design allows users to quickly assemble, modify, and test circuits, making it ideal for learning electronics, debugging designs, and building temporary prototypes. Its versatility and compatibility with various components and microcontroller boards, such as Arduino or NodeMCU, make it an essential tool for hobbyists, students, and engineers. By eliminating the need for soldering, a breadboard provides a clean and efficient way to experiment with electronic circuits.

3.4.2 Software Requirement

3.4.2.1 Arduino IDE

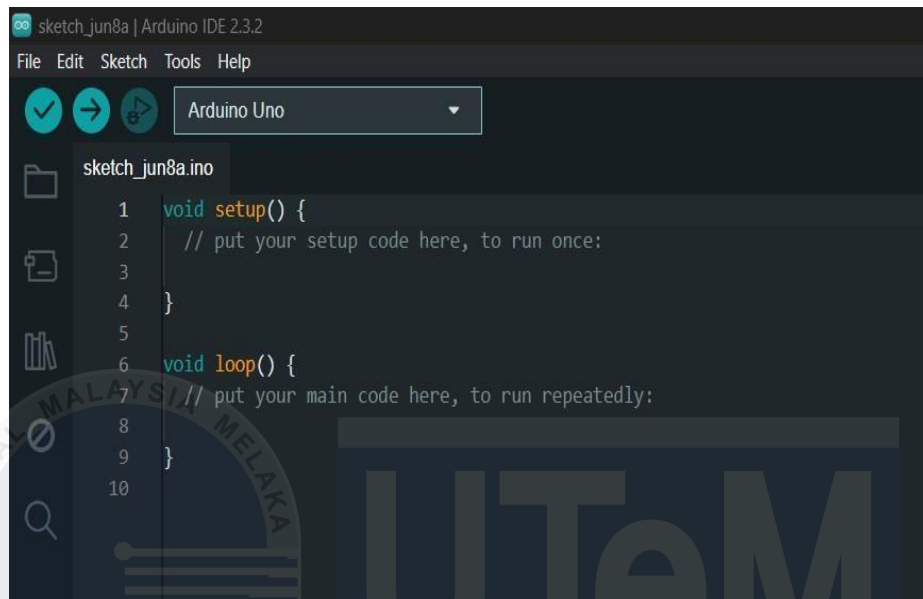


Figure 3.8 Arduino IDE

The Arduino Integrated Development Environment (IDE), Figure 3.8 is a software tool used to program, compile, and upload code onto Arduino microcontroller boards. It provides a user-friendly interface for scripting Arduino projects, making it suitable for both professionals and beginners. The Arduino IDE includes a text editor for writing code, a compiler to translate programs into machine code, and a bootloader to transfer the compiled program to an Arduino board via a computer's USB or serial port. Additionally, it offers various libraries and examples to simplify project development for specific applications. Overall, the Arduino IDE is an indispensable tool for creating interactive electronics projects.

In this project, the Arduino IDE is used to configure an ESP8266 microcontroller to operate with two MFRC522 RFID readers and connect to a Wi-Fi network for transmitting detected card information to a server. The code initializes SPI communication for the RFID readers

and Wi-Fi for network connectivity. During execution, the system continuously checks each RFID reader for new cards. When a card is detected, its unique identifier (UID) is read, converted to a string, and sent to the server via an HTTP POST request. The system logs relevant details to the serial monitor, such as initialization status, detected UIDs, HTTP response codes, and any errors during communication. Error handling ensures that issues like failed HTTP requests or Wi-Fi disconnections are managed smoothly. The process repeats in a loop with a short delay to avoid rapid polling.

3.4.2.2 Visual Studio Code

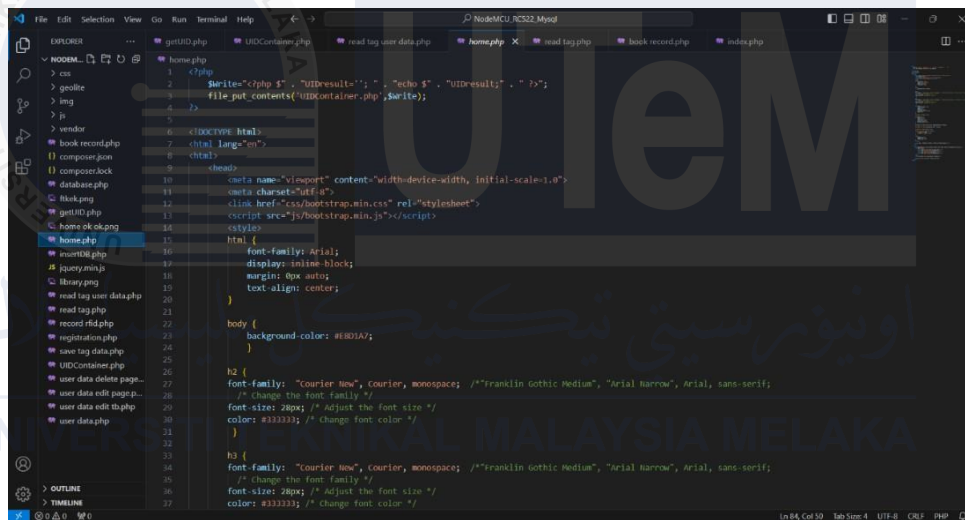


Figure 3.9 Visual Studio Code

Visual Studio Code (VS Code), Figure 3.9 is a feature-rich, lightweight code editor that serves as an excellent tool for web development, including creating dynamic PHP-based websites. It provides a user-friendly interface for writing, editing, and managing PHP scripts, as well as other web technologies like HTML, CSS, and JavaScript. One of the standout features of VS Code is its intelligent coding assistance, which includes syntax highlighting, code autocompletion, and error detection, making it easier to write and debug PHP code efficiently. For PHP development, VS Code supports powerful extensions such as PHP IntelliSense, which provides smart autocompletion for PHP functions and variables, and

PHP Intelephense, which enhances the development experience with improved code analysis and error-checking. Debugging is made seamless with the integration of Xdebug, allowing developers to set breakpoints, step through code, and inspect variables in real-time. Additionally, the editor supports live server extensions that enable developers to preview their PHP applications in a browser and see changes instantly.

The built-in Git integration is another crucial feature, allowing developers to track changes, manage repositories, and collaborate with others directly within the editor. The integrated terminal simplifies running PHP scripts, managing dependencies with tools like Composer, or executing database commands, eliminating the need to switch between multiple tools. VS Code's highly customizable interface lets developers tailor the workspace with themes, keyboard shortcuts, and custom layouts to suit their workflow. Its multi-file and multi-language support makes it easy to work on complex projects involving interconnected PHP scripts and front-end technologies. Furthermore, features like IntelliSense for parameter hints, bracket matching, and snippet generation enhance productivity and reduce coding errors.

In my project, I use VS Code to write and manage PHP scripts for server-side processing, interact with MySQL and Xampp database, and integrate back-end functionality with the front-end design. The extensions and tools provided by VS Code streamline tasks such as coding, testing, debugging, and version control, making it an indispensable tool for PHP development and ensuring a smooth and efficient workflow.

3.4.2.3 Phpmyadmin

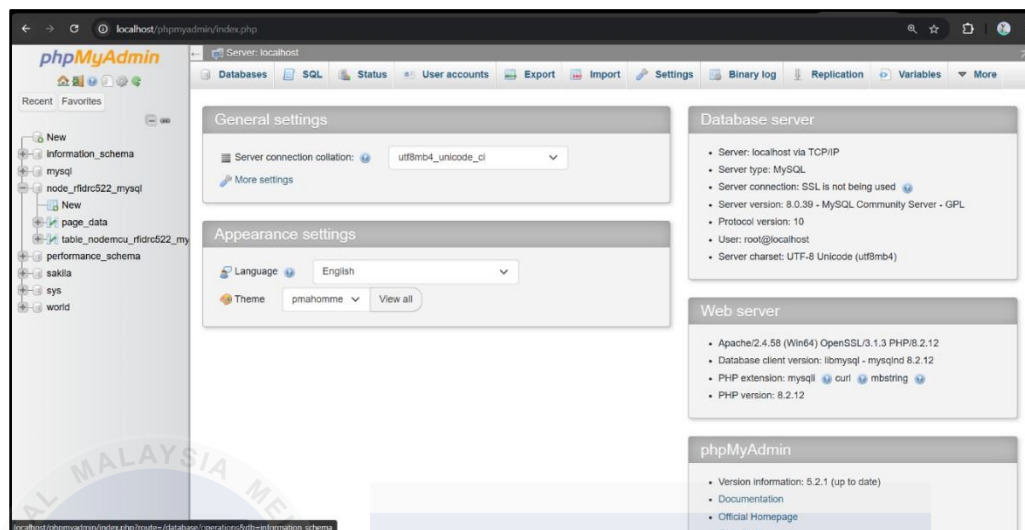


Figure 3.10 phpMyAdmin

phpMyAdmin from Figure 3.10 is a web-based tool designed for managing MySQL and MariaDB databases through an intuitive graphical interface. It simplifies database administration by allowing users to create, modify, and delete databases and manage their associated tables. With phpMyAdmin, users can easily handle table structures by adding, editing, or removing columns and perform data operations such as inserting, updating, and deleting records. The tool also provides a platform for executing custom SQL queries, offering flexibility for advanced database tasks. Additionally, phpMyAdmin facilitates user and privilege management, enabling administrators to create users, assign roles, and control access to databases and tables.

Backup and restoration are streamlined with its export and import features, supporting various formats like SQL, CSV, and XML for database migrations or data recovery. It includes robust search functionality for locating specific data within tables or across databases, as well as tools for managing indexes and keys to optimize query performance. Visual representations of table relationships, such as foreign keys, help users understand and

navigate database schemas. Moreover, phpMyAdmin offers maintenance tools for analyzing database performance, repairing corrupted tables, and optimizing database structures.

In this project, I used phpMyAdmin to create a database for storing all relevant information and to retrieve this data for display on the website. This made it possible to efficiently manage and utilize data collected from the system, ensuring smooth integration between the back-end database and the website interface.

3.4.2.4 Xampp

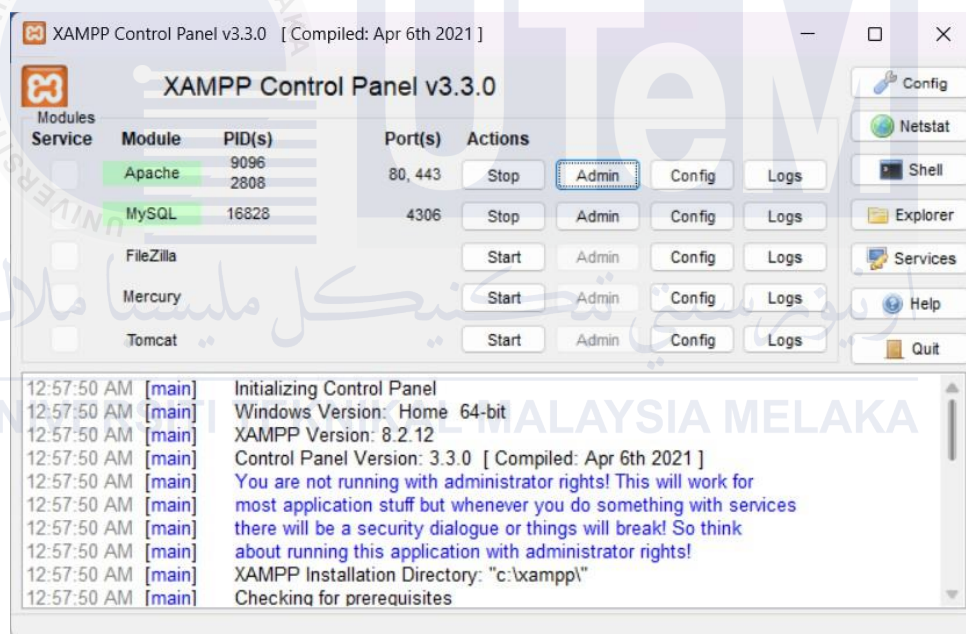


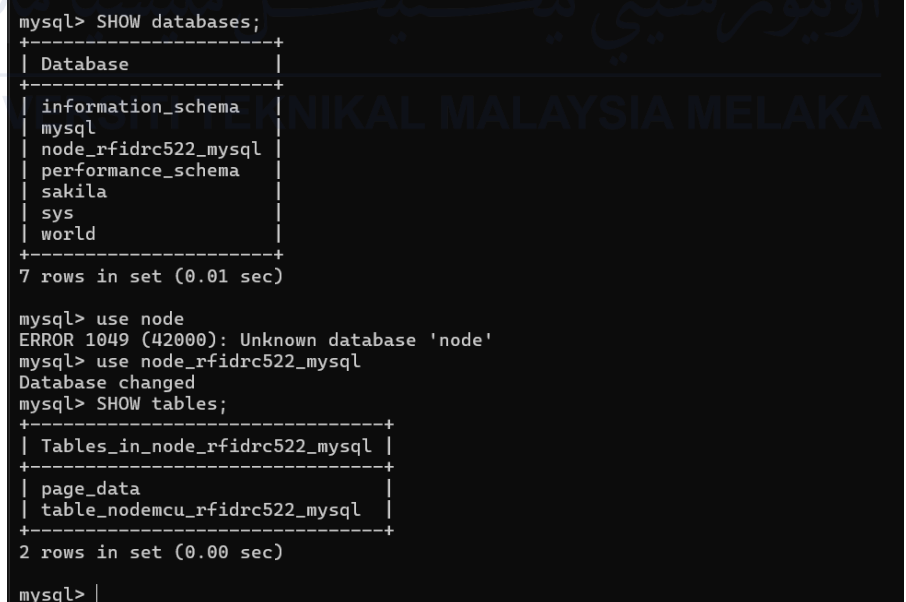
Figure 3.11 Xampp Control Panel

XAMPP, refer at Figure 3.11 is a free and open-source software package that provides a complete local development environment for building and testing web applications. It includes key components such as Apache, a web server for hosting websites locally; MySQL or MariaDB, a database server for managing and storing data; and PHP and Perl, programming languages used for server-side scripting. XAMPP enables developers to simulate a real server environment on their local machines, allowing them to test websites

or applications before deploying them to a live server. It also includes additional tools like phpMyAdmin for database management and FileZilla for FTP functionality.

XAMPP's user-friendly control panel simplifies the process of starting, stopping, and configuring its various components. Developers can use it to host and run dynamic websites, test PHP scripts, and manage databases without needing an active internet connection. Its cross-platform compatibility makes it accessible on Windows, macOS, and Linux systems, catering to a wide range of users. In this project, I use XAMPP to run my PHP scripts and design my website, leveraging its comprehensive features for local web development and testing. Overall, XAMPP is an essential tool for developers, offering a convenient and efficient solution for creating and managing web applications locally.

3.4.2.5 MySQL



```
mysql> SHOW databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| node_rfidrc522_mysql |
| performance_schema |
| sakila |
| sys |
| world |
+-----+
7 rows in set (0.01 sec)

mysql> use node
ERROR 1049 (42000): Unknown database 'node'
mysql> use node_rfidrc522_mysql
Database changed
mysql> SHOW tables;
+-----+
| Tables_in_node_rfidrc522_mysql |
+-----+
| page_data |
| table_nodemcu_rfidrc522_mysql |
+-----+
2 rows in set (0.00 sec)

mysql> |
```

Figure 3.12 MySQL 8.0 Command Line Client

From Figure 3.12, MySQL is a powerful and widely used open-source relational database management system (RDBMS) that is essential for storing, managing, and retrieving data in

structured formats. It allows developers to create databases, define tables, and establish relationships between data entities using SQL (Structured Query Language). MySQL is known for its speed, scalability, and reliability, making it suitable for small-scale projects as well as large-scale enterprise applications. It supports various data types and indexing mechanisms to optimize query performance and ensures data integrity through primary keys, foreign keys, and constraints.

MySQL provides robust features for managing user permissions, enabling secure access control and multi-user collaboration. It also offers powerful querying capabilities for extracting and manipulating data, as well as advanced features like stored procedures, triggers, and views to automate and simplify complex tasks. MySQL is compatible with many programming languages, including PHP, making it an integral part of web application development. Its compatibility with popular tools like phpMyAdmin and seamless integration with platforms like XAMPP further enhances its usability.

In this project, I used MySQL to store timestamp data whenever an RFID tag is detected by the reader. This timestamp information is linked to the respective book data and displayed to the librarians and visitor, allowing them to track the time and date associated with each book's activity. MySQL's reliability and flexibility made it an ideal choice for managing this time-sensitive data, ensuring efficient storage and retrieval for dynamic web applications.

3.4.3 Bill of Material (BOM)

Table 3.2 show all the necessary component used in this project and the price for each component. This component will be purchased before the implementation process is begin.

| No | Component | Description | Quantity | Unit cost | Cost |
|-----------------|-------------------------------------|---|----------|------------|---------|
| 1 | NFC sticker | Adhesive tags with NFC chips for wireless communication | 4 | RM1.19 | RM4.79 |
| 2 | RFID reader | MFRC522 module for reading/writing RFID/NFC tags | 2 | RM4.90 | RM9.80 |
| 3 | NodeMCU (Node MicroController Unit) | Microcontroller board with built-in Wi-Fi | 1 | RM28.00 | RM28.00 |
| 4 | Wire jumper | Insulated wires for connections | 20 | RM0.14 | RM2.80 |
| 5 | Breadboard | Prototyping board for assembling circuits | 1 | RM2.50 | RM2.50 |
| Total component | | | 28 | Total cost | RM47.89 |

Table 3.2 Bill of Material table

3.5 Hardware Setup

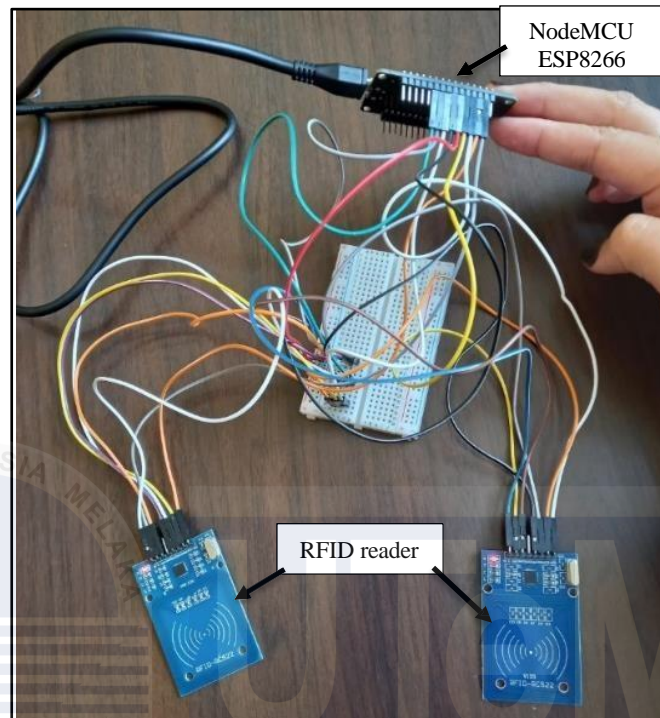


Figure 3.13 Hardware Connection

The Figure 3.13 shows the hardware setup of an ESP8266 microcontroller connected to two MFRC522 RFID readers via a breadboard. This configuration enables simultaneous communication with both RFID readers for an RFID-based book tracking system. The connections for Reader 1 and Reader 2 are designed to share the SPI communication lines (SCK, MOSI, MISO) while using separate pins for SDA (SS) and RST to allow individual identification and operation. Reader 1 connects to the ESP8266 with SDA (SS) wired to D2, RST to D1, SCK to D5, MOSI to D7, MISO to D6, GND to GND, and 3.3V to 3.3V, ensuring safe voltage levels. Similarly, Reader 2 is connected with SDA (SS) to D3, RST to D4, and the shared SPI lines (SCK to D5, MOSI to D7, and MISO to D6). Both readers share the same GND and 3.3V pins from the ESP8266. The setup is powered via the ESP8266's USB interface, ensuring sufficient power for the microcontroller and both RFID readers. The

breadboard provides an organized and modular platform for the wiring, facilitating easy debugging and system testing.

Table 3.3 below show the summarize ESP8266 Pin connections for Two MFRC522 RFID Readers. The pinout reference of ESP8266 can be refer at Appendix Q and the pinout refference of RFID reader can be refer at Appendix R

| Reader 1 | Reader 2 |
|---------------|---------------|
| SDA (SS) → D2 | SDA (SS) → D3 |
| SCK → D5 | SCK → D5 |
| MOSI → D7 | MOSI → D7 |
| MISO → D6 | MISO → D6 |
| RST → D1 | RST → D4 |
| GND → GND | GND → GND |
| 3.3V → 3.3V | 3.3V → 3.3V |

Table 3.3 Hardware Connection

3.6 Test Method

3.6.1 Wi-fi Connectivity Test

The Wi-Fi connectivity test was conducted to evaluate the system's ability to automatically reconnect to a Wi-Fi network. For this test, the system's Wi-Fi connection was intentionally turned off. During this time, the system's behavior was observed to determine how it responded to the disconnection. After that, the Wi-Fi was turned back on, and the system's response to the restored connection was recorded.

3.6.2 Server Communication Test

The Server Communication Test was conducted to evaluate the system's response when communicating with a server under different conditions. First, the system was configured to use a correct URL, and its behavior during the connection attempt was observed. Next, an incorrect URL was deliberately introduced to simulate a faulty server communication scenario. The system's response and behavior under this condition were observed and compared to its behavior when using the correct URL.

3.6.3 Edge Case Handling Test

The Edge Case Handling Test was conducted to evaluate the system's ability to detect NFC stickers under different conditions. The first test involved tapping the same book on different RFID readers to observe the system's behavior and data logging for each reader. The second test involved repeatedly tapping different books on each RFID reader to examine how the system handled multiple scans. In both cases, the system's output was carefully observed and recorded for analysis.

3.6.4 Distance Test

The test evaluates the maximum distance at which an RFID reader can reliably detect an NFC sticker. The procedure involves placing the NFC sticker at varying distances from the RFID reader, starting from 1 cm and incrementally increasing to 2.5 cm. The test stops when the reader can no longer detect the NFC sticker, defining its operational range. This method helps determine the effective detection distance and highlights the reader's performance under varying proximity conditions.

3.6.5 Reader Overlap and Interference Test

The test evaluates the interference between two RFID readers placed at varying distances. The setup starts with the readers positioned 3 cm apart, and the distance is incrementally increased to 6 cm, 9 cm, and finally 12 cm. At each distance, both readers attempt to detect NFC tags. The number of successful and missed detections for each reader is recorded to assess how proximity affects performance, likely due to overlapping electromagnetic fields. This method identifies the extent of interference and its impact on reader performance at different distances.

3.6.6 Orientation Test

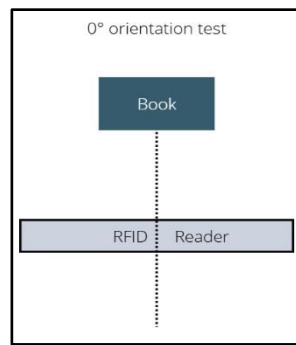


Figure 3.14 0° orientation test

Figure 3.13 show position of RFID reader and the book during 0° Orientation Test

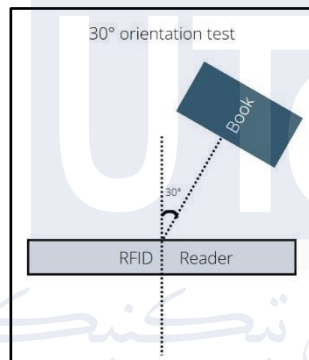


Figure 3.15 30° orientation test

Figure 3.14 show position of RFID reader and the book during 30° Orientation Test

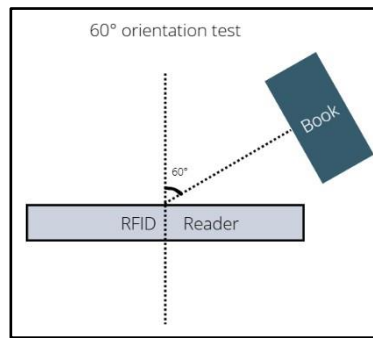


Figure 3.16 60° orientation test

Figure 3.15 show position of RFID reader and the book during 60° Orientation Test

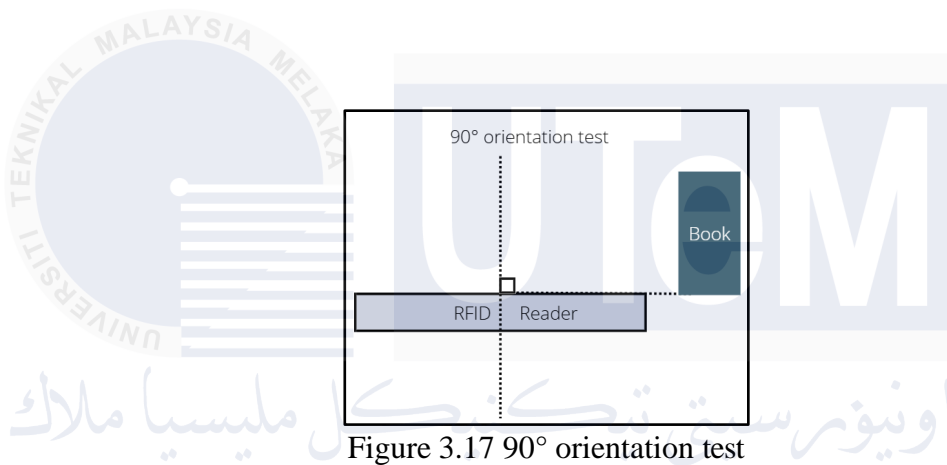


Figure 3.17 90° orientation test

Figure 3.16 show position of RFID reader and the book during 90° Orientation Test

The orientation test was conducted to determine how the angle between the RFID reader and an NFC tag (attached to a book) affects detection performance. The setup involved positioning the NFC tag at specific angles relative to the reader: 0° (aligned flat with the reader), 30°, 60°, and 90° (perpendicular to the reader). This test evaluates the reader's performance at different orientations and identifies the optimal alignment for reliable tag detection.

3.7 Summary

This chapter outlines the comprehensive development methodology for a library book tracking system that leverages RFID technology and a microcontroller to enhance efficiency, accuracy, and user satisfaction in library management. The primary objective is to create a system that automates the tracking of books, minimizes manual intervention, and provides real-time updates on book locations. This innovative solution incorporates a combination of hardware and software tools to achieve a streamlined and scalable system. At the hardware level, the project employs NFC stickers attached to books for identification, RFID RC522 readers for detecting tags, and the NodeMCU ESP8266 microcontroller to process and transmit data. The NodeMCU ESP8266 was chosen for its built-in Wi-Fi capabilities, cost-effectiveness, and ease of integration with IoT-based applications. Other hardware components, such as jumper wires, breadboards, and power sources, were utilized for prototyping and testing the circuitry. The RFID readers are strategically positioned within the library to detect tags as books move through designated areas, while the NodeMCU collects the data, adds metadata such as timestamps and location information, and transmits it to a web-based system.

On the software side, tools like Visual Studio Code, XAMPP, phpMyAdmin, and MySQL were employed to develop the web application and manage the database. Visual Studio Code was used for writing PHP scripts and designing the web interface, while XAMPP provided a local server environment for hosting and testing the system. phpMyAdmin facilitated database management, enabling efficient storage and retrieval of book-related data. The Arduino IDE was used to program the NodeMCU for communication with the RFID readers and for handling HTTP requests to update the server with detected data. The methodology

emphasizes meticulous planning, starting with theoretical research and prototyping, followed by hardware-software integration, extensive testing, and iterative refinement.

The project employed various tests to ensure system reliability and accuracy, such as Wi-Fi connectivity tests to assess the system's ability to reconnect to the network after intentional disconnections and server communication tests to validate error-handling capabilities under correct and incorrect configurations. Edge case handling tests investigated the system's response to repetitive and simultaneous scans, while distance tests determined the effective range of the RFID readers. Reader overlap and interference tests analyzed the impact of electromagnetic interference between RFID readers at varying distances, and orientation tests evaluated how the angle between the NFC tag and RFID reader affected detection accuracy.

The system was designed to meet the library's operational needs, such as real-time book tracking, scalability, and ease of use. By automating inventory management, it reduces manual labor, enhances accuracy, and improves the overall user experience. This chapter provides a thorough roadmap for developing a robust RFID-based library book tracking system, emphasizing sustainability, efficiency, and scalability to meet both current and future requirements of modern library operations.

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Introduction

This chapter presents the findings and analyses derived from the implementation and testing of the book tracking system using a microcontroller and RFID technology. The results are organized to demonstrate the system's functionality, accuracy, and performance in tracking book movements within a library environment. Key aspects such as RFID detection, timestamp data, and database updates are evaluated to ensure alignment with the project objectives. The analysis focuses on the system's ability to reliably detect RFID tags, record timestamps, and display book information on the web interface. Furthermore, this chapter discusses the effectiveness of the system in achieving real-time data updates and minimizing manual effort. Through detailed evaluation, the results validate the system's capability to streamline library operations and improve inventory management.

4.2 Results

4.2.1 Prototype



Figure 4.1 Prototype

Figure 4.2 illustrates the prototype designed for this project. The setup includes a laptop connected to a black box that serves as a physical representation of two distinct book locations, labeled as 'Shelf A' and 'Shelf B' using a marker for visual clarity. This division is essential for demonstrating the system's capability to differentiate between multiple locations.

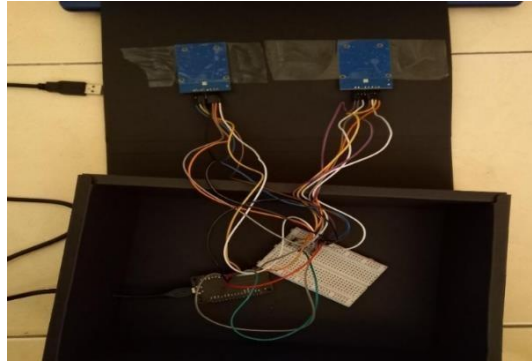


Figure 4.2 Inside Prototype

Inside the box from Figure 4.3, there are two RFID readers. 'Shelf A' is represented Reader 1 and 'Shelf B' is represent Reader 2. The wiring of this system connects with nodemcu ESP8266 which send data to the website.

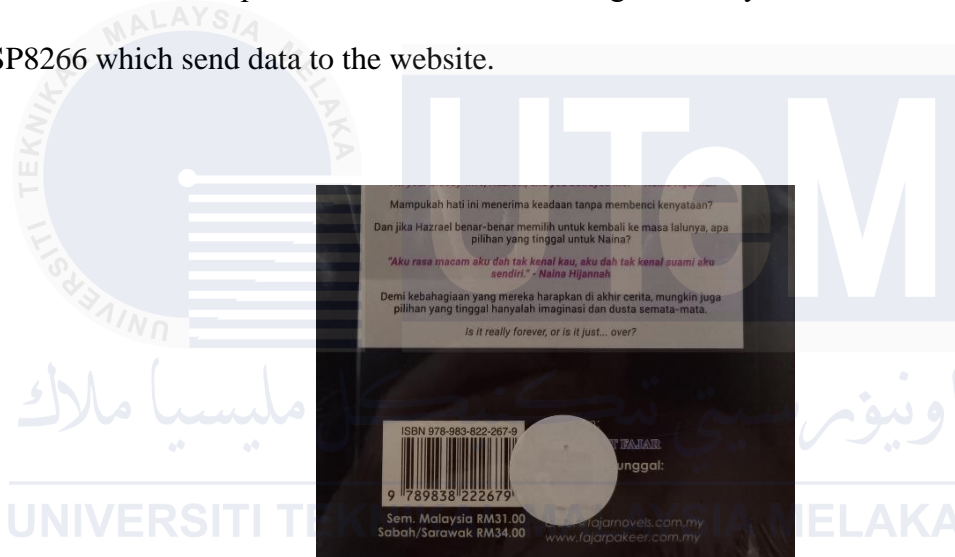


Figure 4.3 NFC Sticker Attached to the Book

The NFC sticker is place at the back of each book just like Figure 4.4. The purpose of the prototype is to demonstrate the functionality of the project, which aims to track and identify the location of books. By scanning the NFC sticker associated with books, the system updates their location as either 'Shelf A' or 'Shelf B,' showcasing its ability to distinguish and log book placements accurately.

4.2.2 Software Development

4.2.2.1 Home Page

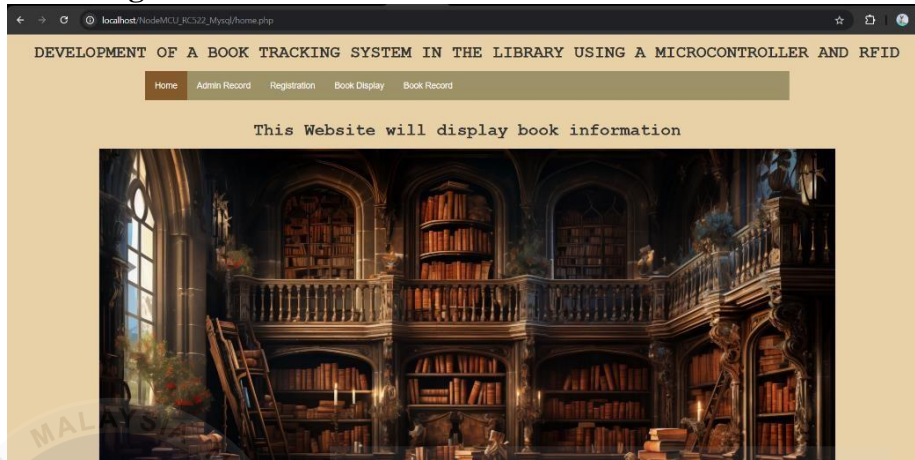


Figure 4.4 Home Page

Figure 4.5 illustrates the home page of the website for the RFID-based book tracking system, designed with a library-centric theme to align with its purpose. The website features a prominent header displaying the project's title, "DEVELOPMENT OF A BOOK TRACKING SYSTEM IN THE LIBRARY USING A MICROCONTROLLER AND RFID," which immediately conveys its objective. Below the header, a navigation menu offers clearly labeled options such as "Home," "Admin Record," "Registration," "Book Display," and "Book Record," ensuring easy access to the website's core functionalities for user. Each menu option is designed to cater to specific tasks, streamlining library management processes.

The main content section features a warm-toned library image with high shelves filled with books, evoking a classic library ambiance. This visually reinforces the website's connection to books and libraries. Above the image, the text "This Website will display book information" succinctly describes the primary purpose of the system. The overall color scheme, typography, and layout create a professional yet approachable interface, making it

intuitive and engaging for users. This homepage serves as a gateway to all features, offering both functionality and thematic coherence in its design.

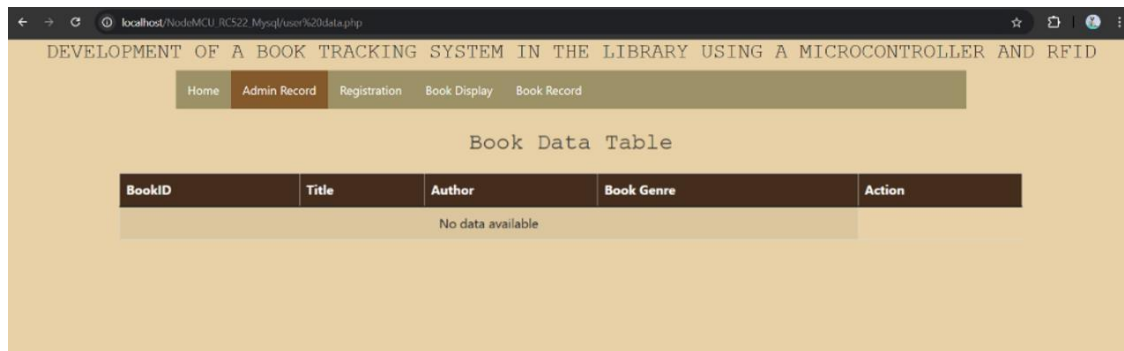
4.2.2.2 Registration

The image shows a web browser window displaying a registration form. The browser's address bar shows 'localhost/nodemcu_rc522_mysql/registration.php'. The page title is 'DEVELOPMENT OF A BOOK TRACKING SYSTEM IN THE LIBRARY USING A MICROCONTROLLER AND RFID'. A navigation bar at the top contains links: 'Home', 'Admin Record', 'Registration' (which is highlighted), 'Book Display', and 'Book Record'. The main content area features a 'Registration Form' with the following fields: 'BookID' (a dropdown menu with the placeholder text 'Please Tag your Card / Key'), 'Title' (a text input field), 'Author' (a text input field), and 'Book Genre' (a text input field). A green 'Save' button is located at the bottom of the form. A large, semi-transparent watermark of the Universiti Teknikal Malaysia Melaka logo is visible in the background.

Figure 4.5 Registration

The Registration menu, as shown in Figure 4.6, displays a form designed for registering books in a library tracking system. When an NFC tag is tapped on the RFID reader, the BookID field is automatically populated with the unique identifier retrieved from the NFC tag. The librarian is required to input additional details, including the Title, Author, and Book Genre of the book. Once all the fields are completed and the Save button is clicked, all the information is stored in the `table_nodemcu_rfidrc522_mysql` within the PHPMysql database, ensuring accurate tracking and management of book records.

4.2.2.3 Admin Record



| BookID | Title | Author | Book Genre | Action |
|-------------------|-------|--------|------------|--------|
| No data available | | | | |

Figure 4.6 Book Data Table

Figure 4.7 displays the "Admin Record" menu of the book tracking system's website, showcasing a table of registered books stored in the database. The table includes columns for essential details such as BookID, Title, Author, and Book Genre, allowing librarians to view all the registered book information in one organized interface. This menu serves as a centralized location where librarians can manage book records efficiently.

Additionally, the interface provides two action buttons, "Edit" and "Delete," for each book entry. The "Edit" function allows librarians to update or correct any errors in the book's details, ensuring the database remains accurate. The "Delete" option enables librarians to remove entries for books that are no longer available in the library, helping to keep the database clean and up to date. The visually distinct buttons, green for "Edit" and red for "Delete," improve usability and reduce the chance of errors during operation.

The clear and structured layout of the data table, combined with the straightforward functionality, ensures librarians can easily manage the library's book inventory while maintaining accuracy and efficiency.

4.2.2.4 User Data Edit

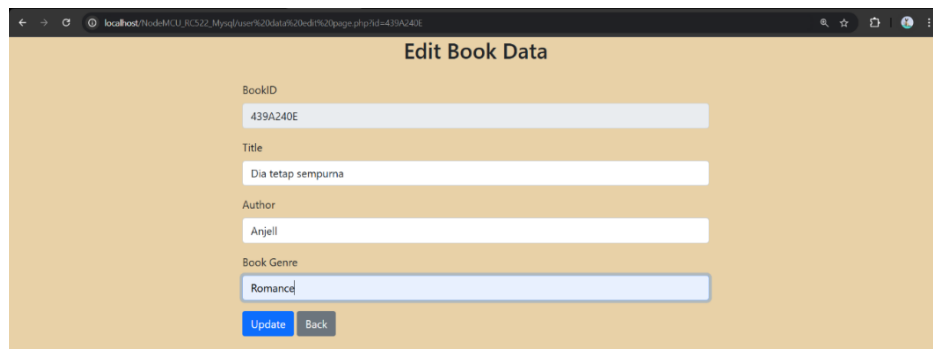


Figure 4.7 Edit Book Data

Based on Figure 4.8, the interface shown allows the librarian to edit the details of a registered book. The page is designed to correct any misinformation or update the book's data as needed. Fields such as `BookID`, `Title`, `Author`, and `Book Genre` are displayed in editable text boxes. While the `BookID` field is typically locked to maintain its uniqueness, other fields can be modified to ensure accurate and up-to-date information. This feature enhances the system's flexibility, allowing librarians to easily manage book records and rectify errors without complications.

4.2.2.5 User Data Delete

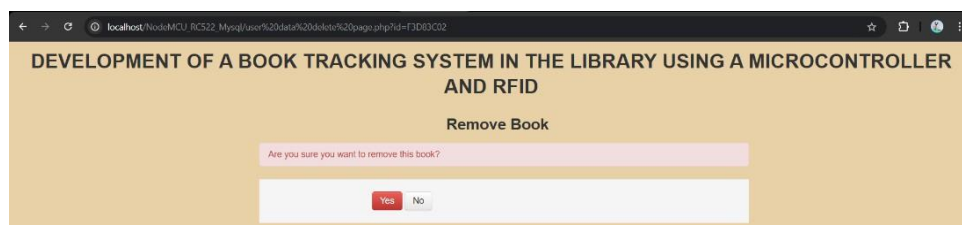


Figure 4.8 Remove Book

Based on Figure 4.9, the interface displayed allows librarians to delete book records that are no longer present in the library. The page provides a confirmation dialog to ensure that the action is intentional, displaying a message, "Are you sure you want to remove this book?"

along with two options: "Yes" and "No." This confirmation step minimizes accidental deletions and ensures that only unnecessary or obsolete book data is removed. By offering this functionality, the system helps maintain an accurate and up-to-date catalog of books available in the library.

4.2.2.6 Book Display

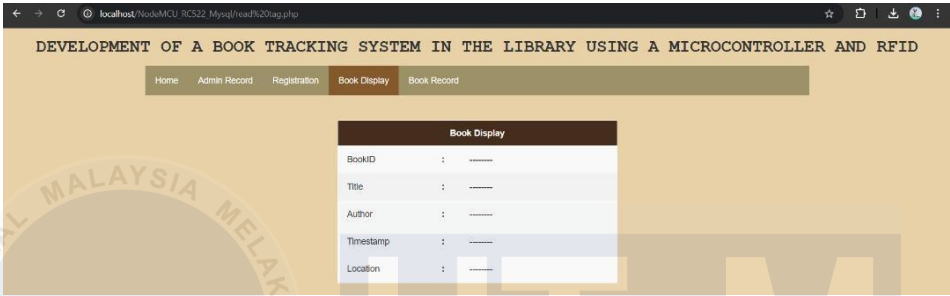
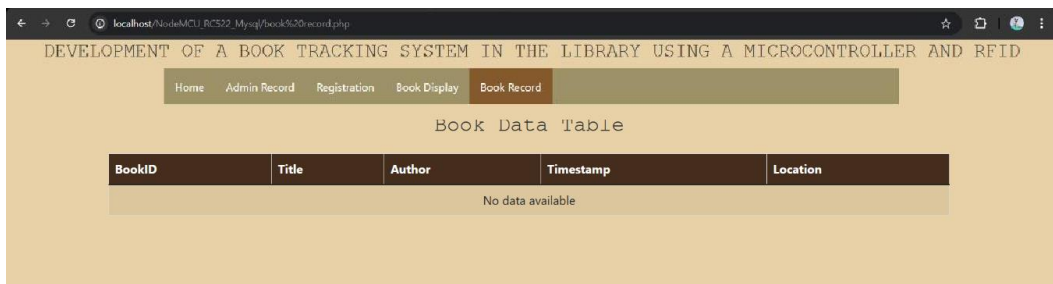


Figure 4.9 Book Display

Figure 4.10 illustrates the ‘Book Display’ menu, which prompts visitors with the book’s information being display here. Once the book is detected, the webpage displays key information, including BookID, Title, Author, Timestamp, and Location. The BookID, Title, and Author are fetched from the `table_nodemcu_rfidrc522_mysql` in the database, ensuring accurate retrieval of book-specific details. The Timestamp and Location are dynamically retrieved from the `page_data` table, providing real-time updates on when and where the book was scanned. This interface streamlines the tracking of books and enhances data accuracy in the library system.

4.2.2.7 Book Record



DEVELOPMENT OF A BOOK TRACKING SYSTEM IN THE LIBRARY USING A MICROCONTROLLER AND RFID

Home Admin Record Registration Book Display **Book Record**

Book Data Table


| BookID | Title | Author | Timestamp | Location |
|-------------------|-------|--------|-----------|----------|
| No data available | | | | |

Figure 4.10 Book Record

Figure 4.11 illustrates the Book Data Table webpage, which displays all the information of books that have been scanned through the ‘Book Display’ menu. The table provides an organized view of key details, including BookID, Title, Author, Timestamp, and Location. This webpage is particularly useful for visitors, as it allows them to monitor the exact time and location where each book was scanned. Since the Timestamp may change with each scan, and the Location of the book might change as visitors take books from the shelves, the table updates to reflect the most recent data. This ensures accurate and real-time tracking of book movements within the library system, enhancing overall book management and traceability.

4.2.3 Database

4.2.3.1 table_nodemcu_rfidrc522_mysql



The screenshot shows a database management interface with the following table structure:

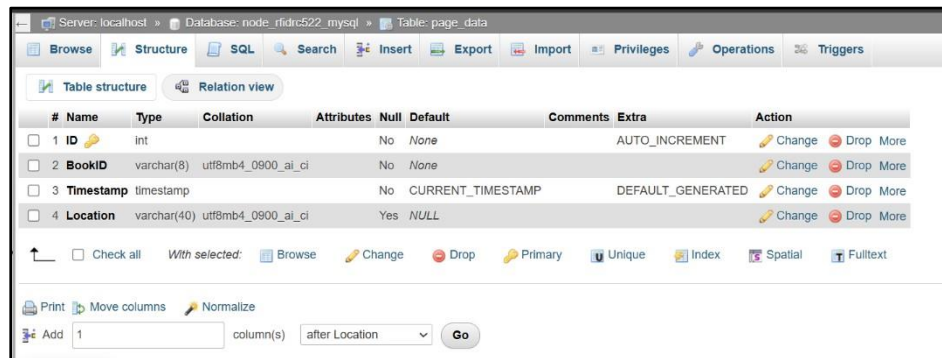
| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra | Action |
|---|-----------|--------------|--------------------|------------|------|---------|------------------------|-------|------------------|
| 1 | BookID | varchar(8) | utf8mb4_0900_ai_ci | | No | None | | | Change Drop More |
| 2 | Title | varchar(255) | utf8mb4_general_ci | | No | None | Title of the book | | Change Drop More |
| 3 | Author | varchar(255) | utf8mb4_general_ci | | No | None | The author of the book | | Change Drop More |
| 4 | BookGenre | varchar(255) | utf8mb4_general_ci | | No | None | | | Change Drop More |

Below the table structure, there are options to 'Check all', 'With selected', 'Browse', 'Change', 'Drop', 'Primary', 'Unique', 'Index', 'Spatial', and 'Fulltext'. At the bottom, there is a 'Print' button, 'Move columns', 'Normalize', and a 'Go' button with a dropdown menu set to 'after BookGenre'.

Figure 4.11 Database table_nodemcu_rfidrc522_mysql

Figure 4.12 illustrates the structure of the database table `nodemcu_rfidrc522_mysql`, designed to store book information. The table consists of four fields: BookID, Title, Author, and BookGenre. The BookID field serves as the primary key and is a unique identifier for each book, defined as `varchar(8)` and does not allow null values. This field is automatically populated when an RFID tag is scanned. The remaining fields like Title, Author, and BookGenre are defined as `varchar(255)` and are manually filled in by the librarian. The Title field stores the book's name, Author records the author's name, and BookGenre categorizes the book's genre. This database ensures that all book details are systematically recorded and securely stored.

4.2.3.2 page_data



The screenshot shows the MySQL Table structure for the 'page_data' table. The table has four columns: ID (int, primary key, auto-increment), BookID (varchar(8), foreign key), Timestamp (timestamp, default CURRENT_TIMESTAMP), and Location (varchar(40)).

| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra | Action |
|---|-----------|-------------|--------------------|------------|------|-------------------|----------|-------------------|------------------|
| 1 | ID | int | | | No | None | | AUTO_INCREMENT | Change Drop More |
| 2 | BookID | varchar(8) | utf8mb4_0900_ai_ci | | No | None | | | Change Drop More |
| 3 | Timestamp | timestamp | | | No | CURRENT_TIMESTAMP | | DEFAULT_GENERATED | Change Drop More |
| 4 | Location | varchar(40) | utf8mb4_0900_ai_ci | | Yes | NULL | | | Change Drop More |

Figure 4.12 Database page_data

Figure 4.13 illustrates the structure and contents of the `page_data` table, which is connected to the `nodemcu_rfidrc522_mysql` table. This table is designed to store tracking data for books and consists of four fields examples ID, BookID, Timestamp, and Location. The ID field serves as the primary key, functioning as an auto-incrementing integer that uniquely identifies each record. The BookID field, defined as `varchar(8)`, references the unique identifier of a book from the `nodemcu_rfidrc522_mysql` table and does not allow null values, ensuring a strong connection between the two tables. The Timestamp field, defined with the `CURRENT_TIMESTAMP` attribute, automatically records the exact date and time of each interaction, as examples like `2024-12-03 14:43:32` and `2024-12-04 22:58:48`. The Location field, defined as `varchar(40)`, is currently unpopulated in the displayed records but allows null values, providing flexibility if location data is unavailable.

4.2.4 Coding

4.2.4.1 Arduino IDE

The code from Appendix A is for an ESP8266-based system that uses two RFID readers (MFRC522) to read NFC card UIDs and send them to a web server via HTTP POST requests. It starts by including necessary libraries for Wi-Fi, HTTP communication, SPI protocol, and RFID reader control. The RFID readers are connected to specific pins on the ESP8266, and their respective pins for reset and slave select (SS) are defined. The Wi-Fi credentials and the server URL are also specified, allowing the ESP8266 to send the UID data to the server.

In the `setup()` function, the system initializes serial communication and the SPI bus, then sets up each RFID reader. The version information of each reader is printed to the serial monitor for debugging. The ESP8266 connects to the Wi-Fi network and waits until it is successfully connected, displaying the IP address upon connection. In the `loop()` function, the code checks the Wi-Fi connection continuously, ensuring the device remains connected. It then loops through each RFID reader, checking if a card is present. If a card is detected, the UID is read and sent to the server. A debounce mechanism is in place to prevent multiple reads in quick succession by using a cooldown period of 1 second.

The `readCard()` function handles the card detection and UID extraction. It uses the `PICC_IsNewCardPresent()` and `PICC_ReadCardSerial()` functions to check for new cards and read their serial numbers. The UID is converted into a string and stored for further use. The `sendUIDToServer()` function is responsible for sending the detected UID to the server through an HTTP POST request. The UID, along with the reader's ID, is sent to the server as form data. The response from the server is printed to the serial monitor. Finally,

the `checkWiFi()` function ensures that if the Wi-Fi connection drops, the device will automatically attempt to reconnect.

This code allows the ESP8266 to interact with multiple RFID readers, read the UIDs from NFC stickers, and send the data to a web server, making it suitable for applications such as RFID-based tracking or access control.

4.2.4.2 PHP Script

4.2.5.2.1 Home.php

Appendix C provides the code for a basic web page used in a library book tracking system. The PHP section initializes a variable, `$UIDresult`, and writes it to a separate file, `UIDContainer.php`, which is likely used to store and display the UID (unique identifier) of RFID tags detected in the system. The HTML portion of the code constructs the user interface, styled with embedded CSS and Bootstrap for responsiveness across devices. The web page includes a title, a main heading introducing the project, and a navigation bar linking to key pages such as "Home," "Admin Record," "Registration," "Book Display," and "Book Record." This navigation bar adapts to smaller screens using media queries. Below the heading, the page displays a message indicating its purpose of showcasing book information, followed by a centered image (`library.png`) for visual enhancement. Overall, the code in Appendix B.1 provides a responsive interface and integrates dynamic backend functionality for RFID-based book tracking.

4.2.5.2.2 User data.php

Appendix D provides the code for displaying book data from a MySQL database in a structured table on a web page. The PHP section establishes a database connection using a

custom Database class and PDO for secure data handling. It fetches details such as BookID, Title, Author, and BookGenre from the `table_nodemcu_rfidrc522_mysql` table, explicitly excluding columns like Location and Date. The retrieved data is stored in an associative array, `$bookData`, which is dynamically used to populate the table in the HTML section.

The HTML portion creates the webpage, styled with Bootstrap and custom CSS for a clean and responsive interface. A navigation bar provides links to different sections of the application, with "Admin Record" set as the active page. The page includes a centered title introducing the project and a table displaying the book data. Each row features "Edit" and "Delete" buttons, linking to respective pages with the BookID passed as a query parameter. If no book data is available, the table displays a message indicating this.

The code uses Bootstrap components for styling and responsiveness, with additional custom CSS for table aesthetics and layout. Overall, Appendix B.2 provides a functional interface for managing book records in the library's RFID-based tracking system, allowing librarians to view, edit, or delete book entries efficiently.

4.2.5.2.3 User data edit page.php

Appendix E provides the code for editing book data in the library's RFID-based book tracking system. The PHP section initializes variables to store book details and checks whether a ``BookID`` is passed through the URL. If a valid ``BookID`` is provided, the script connects to the database using a custom ``Database`` class and retrieves the corresponding book's details (``Title``, ``Author``, and ``BookGenre``) from the table ``table_nodemcu_rfidrc522_mysql``. These details are populated into a form, allowing the librarians to update them. If the ``BookID`` is invalid or no matching data is found, the script redirects the user to the "User Data" page or displays an error message.

The script also handles form submissions for updating the book data. Upon submission, the updated `Title`, `Author`, and `BookGenre` values are sent via POST. The script reconnects to the database and executes an SQL `UPDATE` query to save the changes based on the `BookID`. After the update is complete, the user is redirected back to the "User Data" page to confirm the changes.

The HTML section creates a responsive form using Bootstrap, prefilled with the existing book details retrieved from the database. The form includes fields for `BookID` (read-only), `Title`, `Author`, and `BookGenre`, all marked as required. Buttons for "Update" and "Back" are provided for submitting changes or returning to the main user data page.

The page is styled with Bootstrap for consistent design and includes custom CSS for centering the form and applying a soft background color. This code in Appendix B.3 offers a user-friendly interface for administrators to edit book records efficiently, ensuring the data is updated accurately in the library's book tracking system.

4.2.5.2.4 User data edit tb.php

This code is used to handle the updating of book data in the library's RFID-based book tracking system. When a user navigates to this page, the `BookID` is passed as a query parameter via the URL. The script first retrieves the `BookID` using the `\$_GET` superglobal and ensures it is valid. Once the librarians submit the form with updated book details (`Title`, `Author`, and `BookGenre`), the script processes the data using the `\$_POST` superglobal.

The PHP script establishes a connection to the database using a custom `Database` class. It uses an SQL `UPDATE` query to modify the record associated with the provided `BookID`, ensuring that only the `Title`, `Author`, and `BookGenre` fields are updated, leaving other fields like `Date` and `Location` untouched. After the database update operation is completed, the connection is closed, and the user is redirected to the "User Data" page to view the updated records.

This script plays a critical role in maintaining up-to-date information about the library's book inventory and ensures data integrity during the update process. The functionality is crucial for administrators managing the book records efficiently. Refer to Appendix F for the full implementation of this code.

4.2.5.2.5 User data delete page.php

This PHP script facilitates the deletion of a book record from the library's database in the RFID-based book tracking system. The script starts by retrieving the `BookID` from the `GET` parameter in the URL to identify the specific record the librarian wants to delete. When the librarian submits the confirmation form, the `BookID` is retrieved via the `POST` method.

The script then connects to the database using a custom `Database` class and executes an SQL `DELETE` query to remove the record associated with the provided `BookID`. The `rowCount()` method is used to confirm whether the deletion was successful. If no record is found with the given ID, an appropriate message is displayed to the user. In case of successful deletion, the script redirects the user back to the "User Data" page.

The HTML portion of the script displays a confirmation dialog asking the user to confirm their intention to delete the selected book. It includes a form with a hidden input field containing the `BookID`. The form provides two options: a "Yes" button to confirm deletion and a "No" button to return to the "User Data" page without making any changes.

This functionality ensures that administrators can manage the book inventory by removing outdated or incorrect records. It incorporates user confirmation and error handling for a reliable deletion process. Refer to Appendix G for the full implementation of this code.

4.2.5.2.6 Registration.php

This PHP script serves as the foundation for a registration form in the RFID-based book tracking system. The primary function of the script is to dynamically retrieve and display a unique `BookID` whenever an NFC stickere is scanned. This functionality is achieved by writing an empty PHP variable, `\$UIDresult`, to a separate file, `UIDContainer.php`. This file is loaded and updated at regular intervals via AJAX using jQuery, ensuring the `BookID` is displayed in real time.

The HTML portion structures a user-friendly interface styled with Bootstrap, featuring a navigation bar for accessing different pages like "Home," "Admin Record," "Registration," "Book Display," and "Book Record." The form allows the librarian to input the book's details, such as `BookID`, `Title`, `Author`, and `Book Genre`. The `BookID` field is automatically populated with the unique identifier retrieved from the RFID scan. This is facilitated by the jQuery script, which fetches data from `UIDContainer.php` every 500 milliseconds, ensuring the system stays responsive to tag scans.

Additionally, the page layout includes responsive navigation and styling for compatibility across various devices. On form submission, the entered data is sent to `insertDB.php` to be stored in the database.

This script exemplifies real-time data fetching and updating in web applications, enhancing user interactivity and efficiency in managing library records. Refer to Appendix H for the complete implementation of this code.

4.2.5.2.7 Read tag.php

This PHP and HTML script implements a real-time RFID-based system that displays book data when an RFID tag is scanned. The PHP portion initializes a placeholder file, `UIDContainer.php`, containing a blank variable (`\$UIDresult`) that will later store the scanned UID. This file is dynamically updated via JavaScript using AJAX and jQuery.

The JavaScript code refreshes the content of `UIDContainer.php` every 500 milliseconds, ensuring that any new UID from an RFID scan is captured and displayed without requiring a manual page reload. The captured UID is then processed by another function, `showUser()`, which uses an asynchronous XMLHttpRequest to fetch associated book details (like Title, Author, Timestamp, and Location) from `read tag user data.php`. These details are displayed dynamically in a table on the webpage.

The HTML layout includes a responsive navigation bar styled with CSS, allowing visitors to switch between various system functionalities such as Home, Registration, Book Display, and Book Records. The page also includes a well-structured table that serves as a placeholder for displaying book information associated with a scanned RFID tag.

Additional JavaScript functionality includes monitoring changes in the scanned UID. If a new tag is detected, the system refreshes and retrieves updated data, maintaining synchronization with the database.

The script's primary objective is to ensure efficient and seamless interaction between the RFID system and the user interface, making it an integral part of the RFID-based library book tracking system. Refer to Appendix I for the detailed implementation of this code.

4.2.5.2.8 Read tag user data.php

This PHP code handles the retrieval and display of book details when an RFID tag is scanned. It integrates the `UID` (unique identifier of the RFID tag) and `readerID` (location identifier) to provide accurate book information, including its location in the library. The script starts by including necessary dependencies, such as the GeoIP2 library and database connection files, and initializes a session to manage the `readerID` variable. It also imports the UID and reader ID values from their respective PHP files.

When the RFID tag's `BookID` is passed as a URL parameter, the system determines the book's location based on the `readerID`. For example, `readerID = 1` corresponds to "Shelf A," while `readerID = 2` corresponds to "Shelf B." This data is then checked against the database to confirm whether the scanned book exists. If it does, the script inserts or updates its location in the `page_data` table.

Subsequently, the code retrieves the most recent record for the scanned book, including details such as the title, author, timestamp, and location. If the book does not exist in the database, a fallback message is displayed, and placeholders are used for the book's details.

The retrieved data is dynamically displayed in an HTML table format, ensuring user-friendly visualization.

This script is vital in the context of an RFID-based library book tracking system, where real-time updates and precise book locations are critical for efficient operation. Refer to Appendix J for the detailed implementation and integration of this code within the project.

4.2.5.2.9 Book record.php

This PHP code generates a webpage displaying book data retrieved from a MySQL database, which is part of an RFID-based book tracking system. The script begins by connecting to the database using a helper class (`database.php`) and executing an SQL query that joins two tables: `table_nodemcu_rfidrc522_mysql`, which contains book details such as `BookID`, `Title`, and `Author`, and `page_data`, which stores additional information such as `Timestamp` and `Location`. The `INNER JOIN` ensures that only records with matching `BookID` entries in both tables are fetched.

The fetched data is stored in an associative array (`$bookData`), which is then dynamically rendered into an HTML table using a `foreach` loop. If no data is retrieved, a placeholder message is displayed instead. After fetching the data, the database connection is closed to ensure proper resource management.

The HTML layout is styled using Bootstrap and custom CSS, ensuring a user-friendly interface. It includes a navigation menu for seamless access to other pages, such as "Home," "Registration," and "Book Display." The table is structured with columns for `BookID`, `Title`, `Author`, `Timestamp`, and `Location`. Each row corresponds to a book, with data populated from the database. If a `Location` value is not available, it defaults to "N/A."

This code plays a critical role in providing a clear overview of book information and their associated tracking details, making it essential for library management systems relying on RFID technology. Refer to Appendix K for further details on the database schema and the integration of this code within the project.

4.2.5.2.10 Database.php

This PHP code defines a class Database, which facilitates the connection and disconnection to a MySQL database using PDO (PHP Data Objects). The class follows the Singleton design pattern, ensuring that only one instance of the database connection is created throughout the script's execution. The class defines several private static properties: `$dbName`, `$dbHost`, `$dbUsername`, and `$dbUserPassword`, which store the database credentials. These are used to establish the connection to the database. The `$cont` property is a static variable that holds the PDO connection instance.

The constructor of the class is private, preventing direct instantiation of the class, which enforces the Singleton pattern. The `connect()` method checks if a connection already exists by verifying if `$cont` is null. If not, it attempts to create a new PDO connection to the MySQL database using the provided credentials and connection options (such as error handling and connection timeout). If the connection is successful, the method sets the character encoding to UTF-8 for the connection. If an exception occurs, an error message is logged, and a user-friendly message is displayed. The `disconnect()` method is used to close the database connection by setting the PDO instance (`$cont`) to null. Refer to Appendix L for further details on the database schema and the integration of this code within the project.

4.2.5.2.11 getUID.php

This PHP code snippet handles the processing of data sent via a POST request. It checks if both the ``UIDresult`` and ``readerID`` are set in the POST request using ``isset()``. If both are present, it assigns their values to the variables ``$UIDresult`` and ``$readerID``. The script then writes these values to two separate PHP files: ``UIDContainer.php`` and ``readerContainer.php``. For the ``UIDresult``, it generates a PHP file that defines a variable with the value of the UID and echoes it. Similarly, for the ``readerID``, it creates another PHP file that defines the reader ID and echoes it. The ``file_put_contents()`` function is used to write the dynamically created PHP code into these files.

If either ``UIDresult`` or ``readerID`` is missing in the POST request, the script outputs an error message, "Missing UIDresult or readerID!". This code ensures that the UID and reader ID are stored dynamically in PHP files for later use. For more detail of this code and its integration, refer to Appendix M.

4.2.5.2.12 insertDB.php

This PHP script handles the insertion of data into a MySQL database based on a POST request. It starts by requiring the ``database.php`` file, which contains the necessary database connection functionality. The script then checks if there is any data in the ``$_POST`` array, indicating that a form has been submitted. If data exists, it prints the contents of the ``$_POST`` array for debugging purposes.

The script retrieves form values such as ``bookID``, ``title``, ``author``, and ``BookGenre`` from the POST request, using ``isset()`` to prevent warnings in case any of the fields are not set. Next, it connects to the database using a ``Database::connect()`` call. It prepares a SQL query

to check if the provided `BookID` already exists in the database. If the `BookID` is found, the script outputs an error message indicating that the ID already exists and terminates further execution by calling `exit`.

If no duplicate `BookID` is found, the script proceeds to insert the new book data (BookID, Title, Author, and BookGenre) into the `table_nodemcu_rfidrc522_mysql` table. After the insertion, the database connection is closed with `Database::disconnect()`, and the user is redirected to the "user data.php" page.

For a more detailed understanding of the database connection process and the structure of the data, please refer to Appendix N.

4.2.5.2.13 UIDContainer.php

This PHP code defines a variable `\$UIDresult` and initializes it with an empty string (`""`). It then uses the `echo` statement to output the value of the `\$UIDresult` variable to the browser. Since `\$UIDresult` is set to an empty string, nothing will be displayed on the page when this code is executed. The code effectively serves as a placeholder for a variable that could be populated with data dynamically from other parts of the application. For more details on how this variable interacts with other components or its role in the broader system, refer to Appendix O.

4.2.5.2.14 readerContainer.php

This PHP code snippet defines a variable ``$readerID`` and assigns it an empty string (`""`). It then uses the ``echo`` statement to output the value of the ``$readerID`` variable to the browser. Since ``$readerID`` is initialized with an empty string, nothing will be displayed on the page when this code is executed. This code likely acts as a placeholder for the reader ID, which could be dynamically assigned or updated in the system depending on the user's actions or inputs. For a more detailed explanation of how these variable fits into the larger system and its usage, refer to Appendix P.

4.2.5 Data Collection



Figure 4.13 Book Detected at Shelf A

Figure 4.14 above show that when a book which have NFC sticker attached to it is place on top RFID reader label as 'Shelf A'. When RFID reader detect this NFC, it will read its unique identifier (UID).

The screenshot shows a web browser window with the URL `localhost/Nodemcu_RCS22_Mysql/registration.php`. The page title is "DEVELOPMENT OF A BOOK TRACKING SYSTEM IN THE LIBRARY USING A MICROCONTROLLER AND RFID". The navigation bar includes "Home", "Admin Record", "Registration", "Book Display", and "Book Record". The "Registration Form" is centered on the page with the following fields:

- BookID: 04515396
- Title: Internet Death Angel
- Author: Harleen
- Book Genre: Thriller
- A green "Save" button at the bottom.

Figure 4.14 Registration Book

Figure 4.15 above show that when 'TWIN'S TERRITORY' book is being tap, the BookID is automatically fill up. The other, 'Title', 'Author', and 'Book Genre' is need to fill up manually for register book

The screenshot shows the same web browser window, but the "Registration" tab is active, and the "Book Data Table" is displayed. The table contains the following data:

| BookID | Title | Author | Book Genre | Action |
|----------|----------------------|---------|------------|---|
| 04515396 | Internet Death Angel | Harleen | Thriller | Edit Delete |

Figure 4.15 Book Data Table

Figure 4.16 illustrates that the data of books already registered is stored and displayed on the webpage. The webpage also indicates which books are registered in the system and which are not.

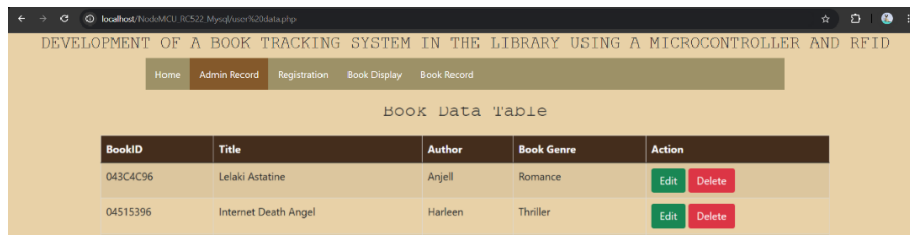


Figure 4.16 Book Detected at Shelf B

This Figure 4.17 show that when the 'LELAKI ASTATINE' is being tap at 'Shelf B'. This show that two readers can detect the UID of the NFC sticker effectively

Figure 4.17 Registration Book

Figure 4.18 show that when 'LELAKI ASTATINE' is being tap at 'Shelf B', the BookID is also automatically fill in. The other like 'Title', 'Author', and 'Book Genre' is need to fill up manually.



DEVELOPMENT OF A BOOK TRACKING SYSTEM IN THE LIBRARY USING A MICROCONTROLLER AND RFID

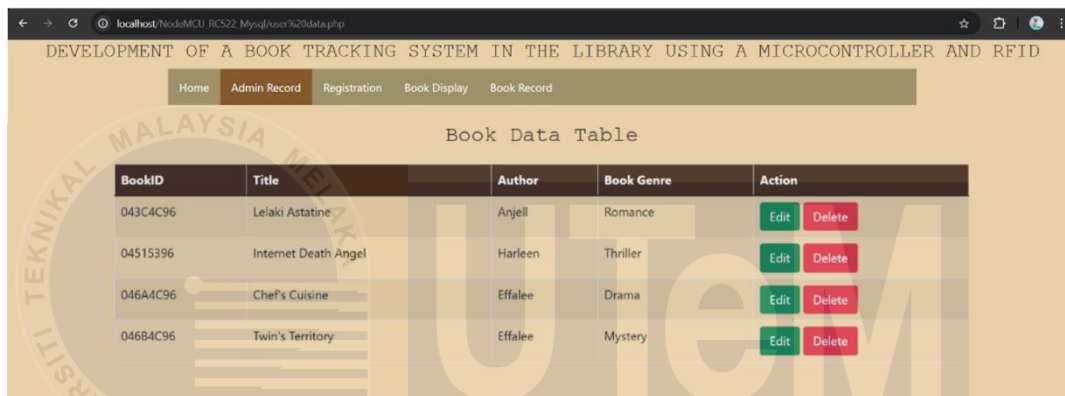
Home Admin Record Registration Book Display Book Record

Book Data Table

| BookID | Title | Author | Book Genre | Action |
|----------|----------------------|---------|------------|---|
| 043C4C96 | Lelaki Astatine | Anjell | Romance | Edit Delete |
| 04515396 | Internet Death Angel | Harleen | Thriller | Edit Delete |

Figure 4.18 Book Data Table

Figure 4.19 show that the 'LELAKI ASTATINE' is successfully being register. All the book's information is stored and display here.



DEVELOPMENT OF A BOOK TRACKING SYSTEM IN THE LIBRARY USING A MICROCONTROLLER AND RFID

Home Admin Record Registration Book Display Book Record

Book Data Table

| BookID | Title | Author | Book Genre | Action |
|----------|----------------------|---------|------------|---|
| 043C4C96 | Lelaki Astatine | Anjell | Romance | Edit Delete |
| 04515396 | Internet Death Angel | Harleen | Thriller | Edit Delete |
| 046A4C96 | Chef's Cuisine | Effalee | Drama | Edit Delete |
| 046B4C96 | Twin's Territory | Effalee | Mystery | Edit Delete |

Figure 4.19 Book Data Table

This Figure 4.20 show when I register the other two books into database and being display here.

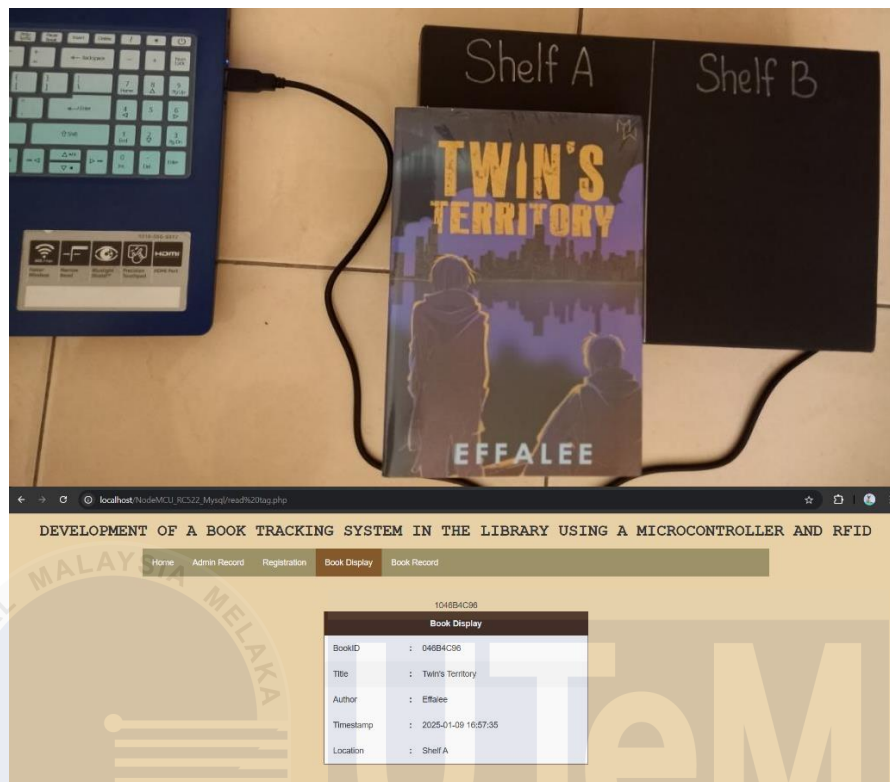


Figure 4.20 Book Display

This Figure 4.21 webpage shows that when 'TWIN'S TERRITORY' book is being tap at 'Shelf A'. This webpage displays the book's timestamp to show when the book is being detected at 'Shelf A'



Figure 4.21 Book Display

The Figure 4.22 show another book, 'INTERNET DEATH ANGEL' when the book is being tap at 'Shelf B'. This webpage displays the book's timestamp to show when is the book is detected at 'Shelf B'.

| BookID | Title | Author | Timestamp | Location |
|----------|----------------------|---------|---------------------|----------|
| 046B4C96 | Twin's Territory | Effalee | 2025-01-09 09:15:22 | Shelf A |
| 04515396 | Internet Death Angel | Harleen | 2025-01-09 09:16:53 | Shelf B |

Figure 4.22 Book Data Table

The Figure 4.23 show the Book Data table that display all the book that being tap is recorded here. We can see all the book record like 'BookID', 'Title', 'Author' and 'Timestamp' and 'Location' here.

DEVELOPMENT OF A BOOK TRACKING SYSTEM IN THE LIBRARY USING A MICROCONTROLLER AND RFID

Home Admin Record Registration Book Display **Book Record**

Book Data Table

| BookID | Title | Author | Timestamp | Location |
|----------|----------------------|---------|---------------------|----------|
| 046B4C96 | Twin's Territory | Effalee | 2025-01-09 09:15:22 | Shelf A |
| 04515396 | Internet Death Angel | Harleen | 2025-01-09 09:16:53 | Shelf B |
| 043C4C96 | Lelaki Astatine | Anjell | 2025-01-09 09:24:43 | Shelf B |
| 046A4C96 | Chef's Cuisine | Effalee | 2025-01-09 09:56:41 | Shelf B |
| 046B4C96 | Twin's Territory | Effalee | 2025-01-09 09:58:09 | Shelf A |
| 046A4C96 | Chef's Cuisine | Effalee | 2025-01-09 10:07:06 | Shelf B |
| 046B4C96 | Twin's Territory | Effalee | 2025-01-09 10:07:14 | Shelf A |
| 046A4C96 | Chef's Cuisine | Effalee | 2025-01-09 11:34:00 | Shelf A |
| 04515396 | Internet Death Angel | Harleen | 2025-01-09 11:34:34 | Shelf A |
| 043C4C96 | Lelaki Astatine | Anjell | 2025-01-09 11:35:18 | Shelf B |
| 046A4C96 | Chef's Cuisine | Effalee | 2025-01-09 16:39:27 | Shelf B |
| 04515396 | Internet Death Angel | Harleen | 2025-01-09 16:39:50 | Shelf A |
| 043C4C96 | Lelaki Astatine | Anjell | 2025-01-11 01:49:58 | Shelf A |
| 046B4C96 | Twin's Territory | Effalee | 2025-01-11 01:50:07 | Shelf B |
| 04515396 | Internet Death Angel | Harleen | 2025-01-11 01:52:12 | Shelf A |
| 046A4C96 | Chef's Cuisine | Effalee | 2025-01-11 01:52:33 | Shelf B |
| 04515396 | Internet Death Angel | Harleen | 2025-01-11 01:53:27 | Shelf B |
| 046B4C96 | Twin's Territory | Effalee | 2025-01-11 01:53:45 | Shelf A |
| 043C4C96 | Lelaki Astatine | Anjell | 2025-01-11 01:54:08 | Shelf B |

Figure 4.23 Book Data Table

Figure 4.24 above show book's record of the other book that being tap and display at 'User Display' menu. Here we can see the different date, timestamp and location for each book.

4.2.6.1 Serial Monitor Arduino IDE

```
Connecting to Wi-Fi.....  
Connected to Wi-Fi!  
IP Address: 192.168.242.103  
Reader 1 detected a card: 043C4C96  
HTTP Response code: 200  
Server response:  
Reader 2 detected a card: 046B4C96  
HTTP Response code: 200
```

Figure 4.24 Serial Monitor of Arduino Ide

Figure 4.25 shows the output on the Serial Monitor of the Arduino IDE, where the HTTP Response Code 200 confirms successful communication with the server. The number "200" is a standardized HTTP status code indicating an "OK" response, meaning the server successfully received, processed, and responded to the HTTP request sent by the ESP8266. In this project, the ESP8266 sent a POST request containing the UID and readerID to the server endpoint `getUID.php`. The server successfully received the data, executed its logic such as logging the data, saving it to the database, and displaying it on a webpage and then sent a response back to the ESP8266. The HTTP 200 status code indicates that the request was processed without issues.

HTTP status codes are a standardized system used in the HTTP protocol to communicate the result of a request between a client, such as the ESP8266, and a server. These three-digit codes are categorized based on the first digit. Codes in the 1xx range (Informational) indicate that the request was received and is being processed. The 2xx range (Success) confirms that the request was successfully received, understood, and processed, with 200 (OK) being a common example. Codes in the 3xx range (Redirection) suggest that further action is required to complete the request, while those in the 4xx range (Client Error) indicate issues

such as bad syntax or unauthorized access. Lastly, the 5xx range (Server Error) reflects a failure on the server's side to process a valid request.

In this project, receiving a 200-status code demonstrates that the communication between the ESP8266 and the server was reliable and executed without errors, highlighting the system's functionality.

4.3 Analysis

4.3.1 Data Testing

4.3.1.1 Wi-Fi Connectivity

| Wi-fi Status | Expected result | Actual result |
|---|--|-------------------|
| Reconnecting after 5 minutes of disconnection | System automatically reconnects and resume | Able to reconnect |

Table 4.1 Wi-fi Test

The Wi-Fi Connectivity Test examines the system's ability to automatically reconnect to a Wi-Fi network after a brief period of disconnection. Specifically, it simulates a scenario where the system is disconnected from Wi-Fi for five minutes. Based on Table 4.1, the expected behaviour is that the system should automatically reconnect to the network and resume its functionality without user intervention. The actual result of the test indicates that the system successfully reconnected as expected, demonstrating robust network recovery capabilities under such conditions.

4.3.1.2 Server Communication Test

| Server URL | Expected result | Actual result |
|---------------|--------------------------|--------------------|
| Correct URL | HTTP Response code: 200. | Behave as expected |
| Incorrect URL | HTTP request failed | Failed as expected |

Table 4.2 Server Test

The Server Communication Test evaluates how the system interacts with a server under two distinct conditions: using a correct URL and an incorrect URL. From Table 4.2, the first case uses correct URL. So, the system is expected to establish a successful connection and return an HTTP response code of 200, confirming that the communication was successful. The test result aligns with this expectation, as the system returned the expected HTTP 200 response code. Conversely, when an incorrect URL is used, the system is anticipated to fail the HTTP request, as the server cannot be reached. The test outcome shows that the system handled the invalid URL appropriately by failing the request, demonstrating effective error management in server communication.

4.3.1.3 Edge Case Handling Test

| Steps | Expected result | Actual result |
|---|--|------------------------------|
| Scan same book at different reader | System logs separate entries with timestamp and location | Failed to behave as expected |
| Scan different book at each reader repeatedly | System logs separate entries with timestamp and location | Behave as expected |

Table 4.3 edge Test

The Edge Case Handling Test focuses on how the system deals with specific edge cases related to scanning books using different readers. Based on Table 4.3, the first scenario involves scanning the same book using different readers. The expected behaviour is for the system to log separate entries for each scan, with distinct timestamps and location for each reader. However, the actual result indicates a failure in meeting this requirement, suggesting that the system does not differentiate between scans of the same book at different readers. This failure occurs because the system overwrites the previous location of the book with the new one rather than creating separate entries for each scan. Additionally, the system's design does not treat the `readerID` as a unique component when logging entries, causing all scans of the same book to be treated as a single record associated with the most recent scan's location and timestamp.

The second scenario tests the system's ability to handle repeated scans of different books at various readers. In this case, the expected behaviour is for the system to log each scan as a separate entry, including the associated timestamp and location details. The actual result confirms that the system behaved as expected in this scenario, accurately logging the entries. These findings highlight both strengths and areas for improvement in the system's ability to manage complex data logging scenarios.

4.3.1.4 Distance Test between RFID Reader and NFC Sticker

| Distance (cm) | Number of Test | Reader 1 Detection | Reader 2 Detection |
|---------------|----------------|-----------------------|-----------------------|
| 1 cm | 5 | 5 | 5 |
| 1.5 cm | 5 | 5 | 5 |
| 1.7 cm | 5 | 5 | 5 |
| 1.9 cm | 5 | 5 | 5 |
| 2.0 cm | 5 | 0 | 0 |
| 2.5 cm | 5 | 0 | 0 |

Table 4.4 Distance Test

Table 4.4 presents the detection range of an RFID reader for an NFC sticker at different distances. The results show successful detection at 1 cm, 1.5 cm, 1.7 cm and 1.9 cm while no detection occurs at 2 cm or beyond. This suggests the RFID reader's effective range is limited to 1.9 cm, highlighting its short detection capability.

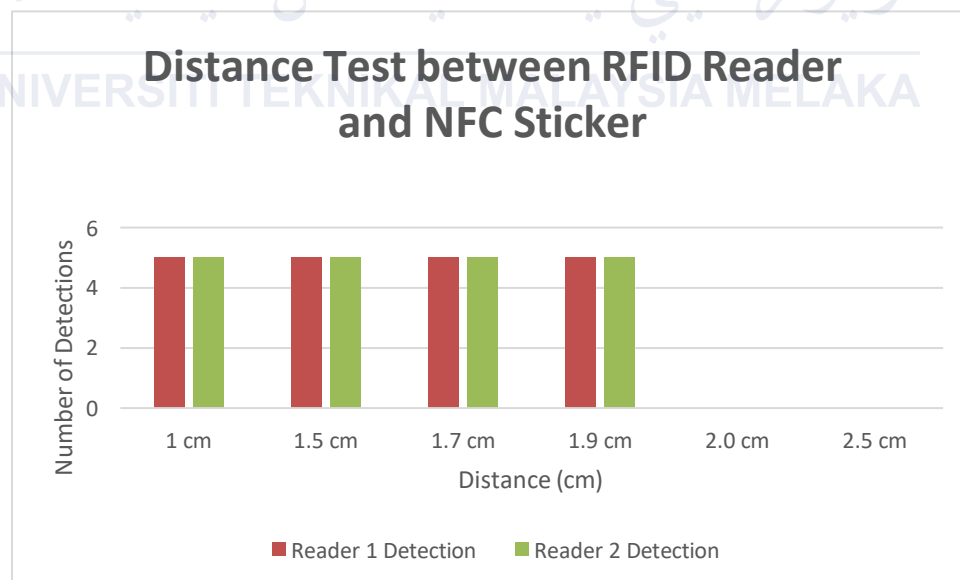


Figure 4.25 Data Chart

The Figure 4.26 bar chart illustrates the detection performance of two RFID readers (Reader 1 and Reader 2) when interacting with an NFC sticker at varying distances, ranging from 1

cm to 2.5 cm. The number of detections is used as the measure of performance across these distances.

Both RFID readers exhibit consistent and reliable detection performance at shorter distances. At 1 cm, both Reader 1 and Reader 2 achieve the maximum detection count, demonstrating their optimal functionality at close proximity. This trend continues as the distance increases incrementally to 1.5 cm, 1.7 cm, and 1.9 cm, where both readers still maintain a high level of detection reliability, albeit with a slight variation in detection counts.

However, beyond 1.9 cm, neither Reader 1 nor Reader 2 is able to detect the NFC sticker. This sharp decline in performance indicates a defined operational range for the RFID readers, effectively limiting their ability to detect NFC stickers beyond this threshold. At distances of 2 cm and 2.5 cm, the detection count drops to zero for both readers, signifying the complete loss of detection capability.

The results highlight that the effective detection range for both RFID readers is constrained to a maximum distance of 1.9 cm. Within this range, the readers demonstrate consistent and reliable performance, which likely reflects the intended design specifications of the system. This restricted operational range may be influenced by factors such as the power output of the readers, the sensitivity of the NFC sticker, or the electromagnetic field strength generated by the readers.

From an operational perspective, the findings underscore the importance of maintaining NFC stickers within the specified range to ensure consistent detection. While the design parameters may aim to minimize external interference and optimize performance within a controlled range, the limited detection distance could pose challenges in applications requiring extended operational ranges. These observations suggest potential areas for

improvement, such as enhancing reader hardware, increasing signal strength, or exploring alternative technologies to support greater detection distances while maintaining reliability.

4.3.1.5 Reader Overlap and Interference Test

| Distance between Reader (cm) | Number of Tests | Reader 1 Detection | Reader 2 Detection | Missed Detection (Reader 1) | Missed Detection (Reader 2) |
|------------------------------|-----------------|--------------------|--------------------|-----------------------------|-----------------------------|
| 3 cm | 5 | 3 | 2 | 2 | 3 |
| 6 cm | 5 | 2 | 3 | 3 | 2 |
| 9 cm | 5 | 4 | 4 | 1 | 1 |
| 12 cm | 5 | 5 | 5 | 0 | 0 |

Table 4.5 Reader Overlap and Interference Test

Table 4.5 presents the results of the Reader Overlap and Interference Test, which examines how the proximity of two RFID readers impacts their detection performance. The test was conducted at four distances: 3 cm, 6 cm, 9 cm, and 12 cm.

At the closest distance of 3 cm, significant interference was observed, resulting in lower detection rates for both readers, with Reader 1 detecting 3 out of 5 tests and Reader 2 detecting 2 out of 5 tests. As the distance increased to 6 cm, the performance remained inconsistent, with Reader 1 detecting 2 tests and Reader 2 detecting 3 tests, showing continued interference.

At 9 cm, both readers demonstrated improved detection rates, with each detecting 4 out of 5 tests and showing fewer missed detections. By 12 cm, the interference appeared to be minimal, as both readers achieved perfect detection rates with no missed detections.

Overall, the results suggest that interference between the readers diminishes as the distance between them increases, with optimal performance achieved at 12 cm.

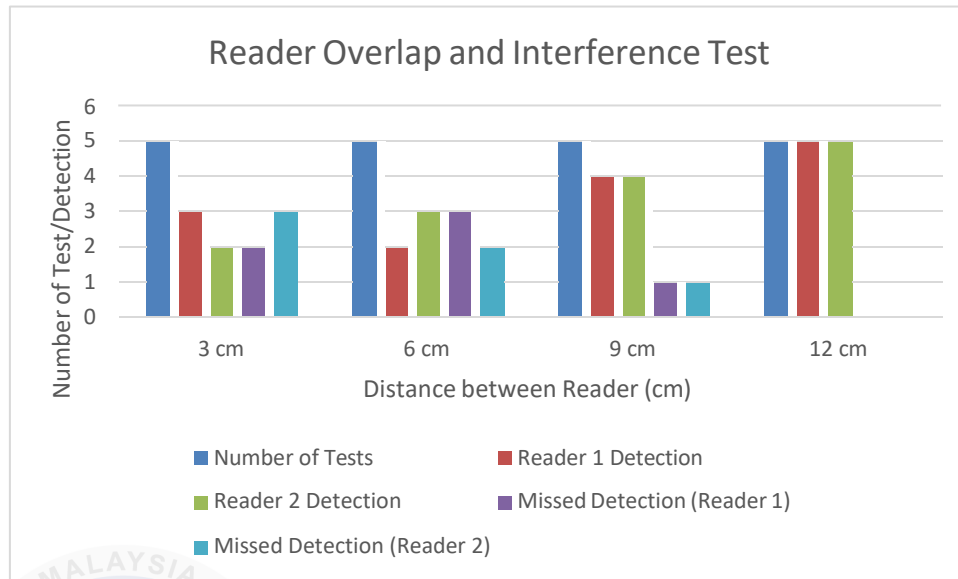


Figure 4.26 Data Chart

Figure 4.27 illustrates the results of the Reader Overlap and Interference Test, which evaluates how the proximity between two RFID readers affects their ability to detect tags. The test was performed at four distances: 3 cm, 6 cm, 9 cm, and 12 cm, with five detection attempts made by each reader at each distance.

At 3 cm, Reader 1 successfully detected 3 tags (orange bar), while Reader 2 detected 2 tags (gray bar). The missed detections are represented by the yellow and light blue bars, indicating 2 missed detections for Reader 1 and 3 for Reader 2. These results reveal significant interference, likely due to overlapping electromagnetic fields at this close distance.

At 6 cm, the detection patterns changed, with Reader 1 detecting 2 tags and Reader 2 detecting 3 tags. The yellow and light blue bars show 3 missed detections for Reader 1 and 2 for Reader 2, suggesting that interference continues to affect performance despite the increased distance.

At 9 cm, both readers showed improved performance, with each detecting 4 tags (orange and gray bars), and the missed detections reduced to 1 for both readers (yellow and light blue bars). This indicates a decrease in interference as the distance between the readers increases.

At 12 cm, both Reader 1 and Reader 2 achieved perfect detection rates, successfully identifying all 5 tags (orange and gray bars) with no missed detections (yellow and light blue bars absent). This demonstrates that interference was no longer a significant factor at this distance.

Overall, the figure demonstrates that interference between the two RFID readers diminishes as the distance between them increases, with optimal performance achieved at the tested maximum of 12 cm. This suggests that maintaining sufficient spacing between RFID readers is critical for minimizing overlap and ensuring accurate detections.

4.3.1.6 Orientation Test

| Orientation (°) | Number of Test | Successful Detection | Missed Detection |
|-----------------|----------------|----------------------|------------------|
| 0° | 5 | 5 | 0 |
| 30° | 5 | 4 | 1 |
| 60° | 5 | 2 | 3 |
| 90° | 5 | 0 | 5 |

Table 4.6 Orientation Test

The results of the Orientation Test in Table 4.6 show how the angle of an NFC tag relative to the RFID reader affects detection. At 0°, the reader achieved full detection success, while at 30° and 60°, performance declined moderately. At 90°, detection failed entirely. These results highlight the importance of keeping the NFC tag as close to 0° as possible for reliable detection.

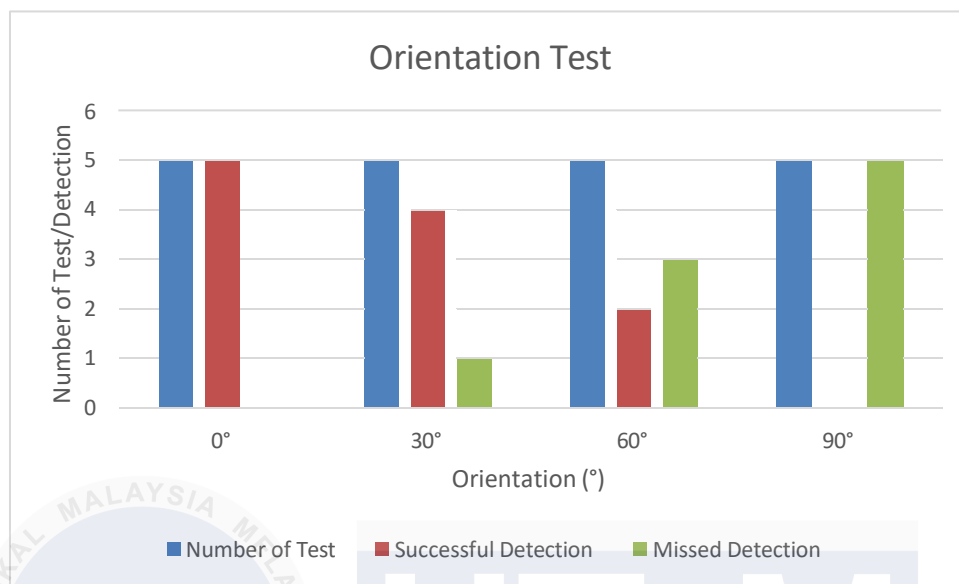


Figure 4.27 Data Chart

The Figure 4.28 above illustrates the results of the Orientation Test, which examines how varying the angle of an NFC tag relative to the RFID reader impacts detection success. At 0°, where the tag is perfectly aligned with the reader, all 5 tests resulted in successful detections, as shown by the equal heights of the blue and red bars, with no missed detections (green bar at zero). This confirms that the RFID reader operates optimally when the tag is flat and directly facing the reader.

At 30°, the red bar (successful detections) drops slightly to 4, while the green bar (missed detections) rises to 1, indicating a small reduction in performance due to the tag's slight deviation from perfect alignment. Despite this, the detection performance remains robust at this angle.

At 60°, the red bar decreases further to 2 successful detections, with the green bar rising to 3 missed detections. This demonstrates a more noticeable decline in detection efficiency, highlighting the reader's struggle as the tag orientation becomes less favorable.

Finally, at 90°, where the tag is fully perpendicular to the reader, the red bar drops to zero, and the green bar reaches its maximum height of 5, indicating that the reader failed to detect the tag in all tests. This confirms that the reader cannot reliably interact with tags at extreme angles.

Overall, the chart visually underscores the steady decline in detection performance as the angle increases from 0° to 90°, emphasizing the need to position NFC tags as close to 0° as possible for reliable system operation.

4.3.2 Challenge in Completing the Project

4.3.2.1 Arduino IDE Coding

The code from Appendix B is unsuccessful because, while it successfully reads the UID from two RFID readers and displays them via the serial monitor, it fails to send this data to a website. The RFID readers are correctly initialized, and the code establishes a connection to the specified Wi-Fi network, confirming the device's ability to communicate over the network. However, the absence of functionality for transmitting the UID data to a web server results in incomplete implementation, limiting the code's effectiveness in achieving its intended purpose.

4.3.2.2 Php Script

One of the key challenges in completing this project was working with the PHP script, particularly in implementing book location detection. I faced difficulties in accurately determining the location of a book using the RFID system. To address this, I attempted to use two RFID readers to differentiate the location of books on the shelves. However, integrating multiple readers and ensuring accurate data capture and location tracking proved to be complex due to synchronization issues, proper placement of readers, and ensuring consistent detection of NFC tags.

4.4 Summary

This chapter presents the findings and analysis from implementing and testing the RFID-based book tracking system developed to enhance library management. The system was evaluated on its functionality, accuracy, and performance in tracking book movements, ensuring real-time updates, and minimizing manual effort. Key aspects such as hardware setup, prototype design, web interface functionalities, database management, and operational tests are discussed in detail.

The hardware setup includes an ESP8266 microcontroller integrated with two MFRC522 RFID readers. These readers are configured to detect NFC-tagged books and transmit data to a web server for processing and display. The prototype features distinct zones, labeled as Shelf A and Shelf B, to demonstrate the system's ability to differentiate between book locations. Each book is tagged with an NFC sticker, and when scanned by the RFID readers, the system updates its location in real time. This functionality is supported by a web interface that enables book registration, editing, deletion, and tracking. A robust MySQL database

underpins the system, storing critical information such as book IDs, titles, authors, genres, timestamps, and locations.

The system's performance was assessed through several tests. Detection range tests revealed that the RFID readers can reliably detect NFC tags within a maximum range of 1.9 cm. However, beyond this range, detection fails, limiting the system's operational flexibility. Orientation tests demonstrated that detection efficiency decreases as the angle of the NFC tag relative to the reader increases. At 0° alignment, detection was flawless, but performance declined significantly at 30° and 60°, with complete failure at 90°. Interference tests highlighted the need for a minimum separation of 12 cm between RFID readers to avoid electromagnetic field overlap, which could cause missed detections.

The system also underwent Wi-Fi and server communication tests, which confirmed its ability to reconnect automatically after brief disconnections and return HTTP status codes of 200 for successful operations. However, challenges were identified in handling specific edge cases, such as scanning the same book at different locations. In such cases, the system overwrote previous location data instead of logging separate entries, indicating a need for improved data management logic.

Overall, the results demonstrate the potential of the RFID-based book tracking system in automating library operations. The system enhances efficiency by providing real-time updates, reducing manual inventory tasks, and maintaining accurate book records while reliably tracking and displaying the locations of books. These findings affirm the system's capability to streamline library management processes effectively.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

The development of a book-tracking system using a microcontroller and RFID technology has successfully addressed the limitations of traditional library management systems. These systems often suffer from inefficiencies, inaccuracies, and heavy reliance on manual processes, which can lead to misplaced books and frustrated users. The newly developed system provides an innovative solution by automating inventory management and enabling real-time tracking of book locations. Through rigorous testing, including Wi-Fi connectivity, server communication, distance measurements, and orientation evaluations, the system demonstrated high accuracy, reliability, and robustness.

While the system only manages to record the last availability of the book at a specified shelf, it can be further improved to also record whenever the book is taken out from the shelves. This system not only benefits librarians by reducing their workload and enhancing operational efficiency but also significantly improves the overall experience for library visitors. Visitors can quickly locate books with real-time updates on their availability and exact locations, minimizing search time and frustration. The web-based interface ensures user-friendly interaction, enabling visitors to access book details and locations effortlessly. By streamlining the process of finding and managing books, the system enhances user satisfaction and promotes a more organized and accessible library environment. Ultimately, this project has achieved its objectives of creating a scalable, efficient, and user-focused library management system.

5.2 Potential for Commercialization

This RFID-based book-tracking system holds considerable potential for commercialization, particularly for large-scale libraries in academic institutions, public facilities, and private organizations. Its real-time tracking capabilities, automated updates, and accurate inventory management make it a highly attractive solution for libraries seeking modernization. The modular design allows customization to include features like book borrowing, returning, fine management, and analytics, making it adaptable to a variety of library settings.

Additionally, the system's cost-effectiveness and scalability ensure that it is not limited to large institutions but is also suitable for smaller libraries with limited budgets. For library visitors, the system's ability to provide instant access to book availability and locations adds a significant value proposition, encouraging widespread adoption. Collaborations with technology vendors and library software providers could further enhance its features and expand its market reach, positioning it as a leading solution in library automation systems.

5.3 Future Works

To further improve this project, several enhancements are recommended. First, the implementation of a registration menu and a book record menu accessible exclusively to librarians is essential. These features would require secure username and password authentication to prevent unauthorized access. The registration menu would enable librarians to manage user registrations, assign library cards, and control access, while the book record menu would allow them to add, update, and delete book information, maintaining an accurate database. The system should also be able to detect the same book at different readers, record whenever the book is taken out from the shelves, enabling more accurate tracking of book movement and location changes.

To enhance security and scalability, encrypted data transmission between the microcontroller and the web server should be implemented to safeguard sensitive information. Transitioning to a cloud-based database would provide better scalability, reliability, and backup options, ensuring the system can handle larger data volumes efficiently.

Additionally, developing a mobile application would provide greater convenience for library visitors, allowing them to access book availability, locations, and other library services directly on their smartphones. Advanced analytics tools could be integrated to help librarians analyze book usage trends, optimize shelf organization, and predict future resource needs. Expanding the hardware capabilities, such as using long-range RFID readers or incorporating additional sensors, could further improve the accuracy and efficiency of book tracking. These proposed enhancements would transform the system into a robust, comprehensive, and future-ready solution for modern library environments, benefitting both library staff and visitors alike.

REFERENCES

- [1] Al Asrafi, A. A., Anti, A. H., & Ahmed, R. (2021). Smart library management system (SLMS) using RFID technology (Bachelor's thesis, University of Asia Pacific, Department of Electrical and Electronic Engineering, Dhaka, Bangladesh).
- [2] Okubanjo, A., Okandeji, A., Osifeko, O., Onasote, A., & Olayemi, M. (2022). Development of a hybrid radio frequency identification (RFID) and biometric-based library management system. *Gazi University Journal of Science*, 35(2), 567-584. <https://doi.org/10.35378/gujs.834087>.
- [3] Dharani Devi, P., Mirudhula, S., & Devi, A. (2021). Advanced library management system using IoT. In *Proceedings of the Fifth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)* (pp. 150-152). IEEE. <https://doi.org/10.1109/I-SMAC52330.2021.9640697>.
- [4] He, H., Liu, T., & Wang, E. (2017). Intelligent book positioning system for library based on RFID. In *Proceedings of the Conference on Industrial Electronics and Applications (ICIEA)*. IEEE. <https://doi.org/10.1109/ICIEA.2017.8006380>.
- [5] Khadgi, R., Dangol, S., Lamsal, S., & Shrestha, S. (2022). RFID-based library management system (Bachelor's thesis, Khwopa Engineering College, Department of Electronics & Communication Engineering, Bhaktapur, Nepal).
- [6] Mukund, N. R., Arun, S., Kusuma, S. M., Vishak, K. R., & Monisha, M. (2021). Intelligent RFID-based library management system. In *Proceedings of the International Conference on Electronics, Computing and Communication Technologies (CONECCT)*. IEEE.

- [7] Babu, S., Srividhya, S., & Aishwarya, U. (2023). RFID-based library management system. *International Research Journal of Modernization in Engineering Technology and Science*, 5(6), 1-10.
- [8] Suhaimi, M. M., Mohamed, Z., & Khusaini, N. S. (2023). Effectiveness of RFID smart library management system. *Journal of Mechanical Engineering*, SI 12, 133-152. <https://doi.org/10.24191/jmeche.v12i1.24642>.
- [9] Gannamraju, P., Yarramsetti, S., & Kumar, L. S. (2021). Radio frequency identification and internet of things-based smart library management system. *International Journal of Networking and Virtual Organisations*, 24(4), 329-346. <https://doi.org/10.1504/IJNVO.2021.116430>.
- [10] Keshinro, K. K., Balogun, W. A., Oyetola, J. B., & Omogoye, S. O. (2016). Development of RFID library management information system. *American Journal of Engineering Research (AJER)*, 5(4), 158-160.
- [11] Mozilla Developer Network. (n.d.). HTTP response status codes. Retrieved December 23, 2024, from <https://developer.mozilla.org/en-US/docs/Web/HTTP/Status>
- [12] W3Schools. (n.d.). HTTP messages. Retrieved, from https://www.w3schools.com/tags/ref_httpmessages.asp
- [13] PRAFUL SHINDE, PRADNYA CHIKANE, VAISHALI RAMCHANDRAN, PROF. RASHMI MAHAJAN (2019). "AUTOMATED BOOK MANAGEMENT AND TRACKING SYSTEM FOR LIBRARIES USING RFID". Department of Electronics & Telecommunication engineering, Lohegaon, Pune, Maharashtra (India). <https://ssrn.com/abstract=3382780>
- [14] RAMJI P. M. , SHUNBAGA PRADEEPA T. (2021). "DESIGN AND IMPLEMENTATION OF BOOK TRACKING SYSTEM IN LIBRARY".

International Journal of Innovative Research in Applied Sciences and Engineering (IJIRASE) Volume 4, Issue 7, DOI:10.29027/IJIRASE.v4.i7.2021.811-816, January 2021. Department of Electronics and Communication Engineering, Institute of Technology Coimbatore, Tamilnadu, India.

- [15] Teach Me Microcontrollers.(December 4, 2023) . NodeMCU Pinout., Retrieved from <https://www.teachmemicro.com/nodemcu-pinout/>
- [16] MySQL Documentation. (n.d.). Date and time functions. Retrieved, from <https://dev.mysql.com/doc/refman/8.4/en/date-and-time-functions.html>
- [17] Visual Studio Code. (December 11, 2024.). Visual Studio Code for the web. Retrieved, from <https://code.visualstudio.com/docs/editor/vscode-web>
- [18] University of Virginia, Computer Science Department. (2023, September 12). Setting up a database with XAMPP. Retrieved from <https://www.cs.virginia.edu/~up3f/cs4750/supplement/DB-setup-xampp.html#:~:text=XAMPP%20is%20an%20open%20source,install%20and%20set%20up%20XAMPP.>
- [19] Cloudways. (2025, January 5). How to join two tables in MySQL. Cloudways. Retrieved from [https://www.cloudways.com/blog/how-to-join-two-tables-mysql/#:~:text=Ans%3A%20Joining%20two%20tables%20in,and%20Union%20\(removes%20duplicates\)](https://www.cloudways.com/blog/how-to-join-two-tables-mysql/#:~:text=Ans%3A%20Joining%20two%20tables%20in,and%20Union%20(removes%20duplicates))
- [20] Stack Overflow. (2013). Show values from a MySQL database table inside an HTML table on a webpage. Stack Overflow. Retrieved from <https://stackoverflow.com/questions/17902483/show-values-from-a-mysql-database-table-inside-a-html-table-on-a-webpage>

APPENDICES

Appendix A Arduino IDE

```
#include <ESP8266WiFi.h>    // Include the Wi-Fi library

#include <ESP8266HTTPClient.h> // Include the HTTP Client library

#include <SPI.h>             // Include SPI library for MFRC522

#include <MFRC522.h>         // Include the MFRC522 library

// Pin Definitions for the RFID Readers

#define RST_1_PIN  D1        // Reset pin for Reader 1
#define SS_1_PIN   D2        // SDA (SS) pin for Reader 1

#define RST_2_PIN  D4        // Reset pin for Reader 2
#define SS_2_PIN   D3        // SDA (SS) pin for Reader 2

#define NR_OF_READERS 2      // Number of RFID readers

// Wi-Fi Credentials

const char* ssid = "Galaxy A126FFD"; // Replace with your Wi-Fi SSID
const char* password = "busybody";   // Replace with your Wi-Fi password

// Server URL

const          char*          serverURL          =
"http://192.168.242.166/NodeMCU_RC522_Mysql/getUID.php";
```

```

// Array to hold SS and RST pins for readers

byte ssPins[] = {SS_1_PIN, SS_2_PIN};

byte rstPins[] = {RST_1_PIN, RST_2_PIN};


// Create instances of MFRC522 for each reader

MFRC522 mfrc522[NR_OF_READERS];


// Variables for storing UIDs
String UID[NR_OF_READERS];
unsigned long lastReadTime[NR_OF_READERS] = {0}; // For debouncing
const unsigned long debounceDelay = 1000;    // 1 second cooldown

/**
 * Initialize Wi-Fi and RFID readers.
 */

void setup() {

  Serial.begin(115200); // Initialize serial communications with the PC

  SPI.begin();        // Initialize SPI bus


  // Initialize each RFID reader

  for (uint8_t reader = 0; reader < NR_OF_READERS; reader++) {

    mfrc522[reader].PCD_Init(ssPins[reader], rstPins[reader]);

    Serial.print("Reader ");

```

```

Serial.print(reader + 1); // Display "Reader 1" or "Reader 2"

Serial.print(": ");

mfr522[reader].PCD_DumpVersionToSerial(); // Print reader version info
}

// Connect to Wi-Fi
WiFi.begin(ssid, password);

Serial.print("Connecting to Wi-Fi");
while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
}

Serial.println("\nConnected to Wi-Fi!");
Serial.print("IP Address: ");

Serial.println(WiFi.localIP());
}

/**
 * Main loop to detect and send UIDs from RFID readers.
 */

void loop() {

    checkWiFi(); // Ensure Wi-Fi is connected

    for (uint8_t reader = 0; reader < NR_OF_READERS; reader++) {

```

```

unsigned long currentTime = millis();

if (currentTime - lastReadTime[reader] > debounceDelay) {

    if (readCard(mfrc522[reader], UID[reader], ssPins[reader])) {

        lastReadTime[reader] = currentTime; // Update last read time

        Serial.print("Reader ");

        Serial.print(reader + 1);

        Serial.println(" detected a card: " + UID[reader]);

        sendUIDToServer(UID[reader], reader + 1);

    }

}

delay(100); // Polling delay
}

/**
 * Read card UID from the specified reader.
 */

bool readCard(MFRC522& reader, String& UID, int ss_pin) {

    digitalWrite(ss_pin, LOW); // Select the reader

    if (!reader.PICC_IsNewCardPresent() || !reader.PICC_ReadCardSerial()) {

        digitalWrite(ss_pin, HIGH); // Deselect the reader

        return false;

    }
}

```



```

byte cardUID[4];

char buffer[32] = "";

for (int i = 0; i < 4; i++) {

    cardUID[i] = reader.uid.uidByte[i];

}

arrayToString(cardUID, 4, buffer);

UID = String(buffer);

reader.PICC_HaltA(); // Halt the card

digitalWrite(ss_pin, HIGH); // Deselect the reader

return true;

}

/**
 * Convert a byte array to a string.
 */

void arrayToString(byte array[], unsigned int len, char buffer[]) {

    for (unsigned int i = 0; i < len; i++) {

        byte nib1 = (array[i] >> 4) & 0x0F;

        byte nib2 = (array[i] >> 0) & 0x0F;

        buffer[i*2] = nib1 < 0xA ? '0' + nib1 : 'A' + nib1 - 0xA;

        buffer[i*2+1] = nib2 < 0xA ? '0' + nib2 : 'A' + nib2 - 0xA;

    }

    buffer[len*2] = '\0';

```

```

}

/**
 * Send UID to the server via HTTP POST request.
 */
void sendUIDToServer(String UID, int readerID) {
    if (WiFi.status() != WL_CONNECTED) {
        Serial.println("Wi-Fi not connected!");
        return;
    }

    HTTPClient http;
    WiFiClient client;

    String postData = "UIDresult=" + UID + "&readerID=" + String(readerID); // Send both
    UID and readerID

    http.begin(client, serverURL);

    http.addHeader("Content-Type", "application/x-www-form-urlencoded");

    int httpCode = http.POST(postData); // Send POST request

    if (httpCode > 0) {

        Serial.println("HTTP Response code: " + String(httpCode));

        String payload = http.getString();

        Serial.println("Server response: " + payload);
    }
}

```

```

    } else {

        Serial.println("HTTP request failed.");

    }

    http.end();

}

/**

 * Check and reconnect Wi-Fi if disconnected.

 */

void checkWiFi() {
    if (WiFi.status() != WL_CONNECTED) {
        Serial.println("Reconnecting to Wi-Fi...");
        WiFi.begin(ssid, password);
        while (WiFi.status() != WL_CONNECTED) {

            Serial.print(".");

            delay(500);

        }

        Serial.println("\nWi-Fi reconnected!");
    }

}

```

Appendix B Arduino IDE

```
#include <ESP8266WiFi.h>
#include <SPI.h>
#include <MFRC522.h>

// Updated Pin Definitions for Reader 1 and Reader 2
#define SS_PIN_1 D2 // SDA (SS) for the first RC522
#define RST_PIN_1 D1 // RST for the first RC522

#define SS_PIN_2 D3 // SDA (SS) for the second RC522
#define RST_PIN_2 D4 // RST for the second RC522

MFRC522 rfid1(SS_PIN_1, RST_PIN_1); // Create instance for first reader
MFRC522 rfid2(SS_PIN_2, RST_PIN_2); // Create instance for second reader

// WiFi credentials
const char* ssid = "Galaxy A126FFD"; // Replace with your WiFi SSID
const char* password = "busybody"; // Replace with your WiFi password

void setup() {
  Serial.begin(115200);
  SPI.begin(); // Initialize SPI bus (uses hardware-defined pins)
  rfid1.PCD_Init(); // Initialize the first reader
  rfid2.PCD_Init(); // Initialize the second reader
  Serial.println("RFID readers initialized.");

  // Connect to WiFi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("Connected to WiFi.");
}

void loop() {
  // Check first RFID reader
  if (rfid1.PICC_IsNewCardPresent()) {
    if (rfid1.PICC_ReadCardSerial()) {
      Serial.print("RFID 1: ");
      for (byte i = 0; i < rfid1.uid.size; i++) {
        Serial.print(rfid1.uid.uidByte[i], HEX);
        Serial.print(" ");
      }
      Serial.println();
      rfid1.PICC_HaltA(); // Halt PICC
      rfid1.PCD_StopCrypto1(); // Stop encryption to free reader
    }
  }
}
```

```

    }
}

// Check second RFID reader
if (rfid2.PICC_IsNewCardPresent()) {
    if (rfid2.PICC_ReadCardSerial()) {
        Serial.print("RFID 2: ");
        for (byte i = 0; i < rfid2.uid.size; i++) {
            Serial.print(rfid2.uid.uidByte[i], HEX);
            Serial.print(" ");
        }
        Serial.println();
        rfid2.PICC_HaltA(); // Halt PICC
        rfid2.PCD_StopCrypto1(); // Stop encryption to free reader
    }
}
delay(100); // Add a small delay to prevent overwhelming the serial output
}

```



اونيورسيتي تېكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Appendix C Home.php

```
<?php
$Write="<?php $" . "UIDresult="; " . "echo $" . "UIDresult;" . " ?>";
file_put_contents('UIDContainer.php',$Write);
?>

<!DOCTYPE html>
<html lang="en">
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta charset="utf-8">
<link href="css/bootstrap.min.css" rel="stylesheet">
<script src="js/bootstrap.min.js"></script>
<style>
html {
font-family: Arial;
display: inline-block;
margin: 0px auto;
text-align: center;
}

body {
background-color: #E8D1A7;
}

h2 {
font-family: "Courier New", Courier, monospace; /*"Franklin Gothic Medium",
"Arial Narrow", Arial, sans-serif;
/* Change the font family */
font-size: 28px; /* Adjust the font size */
color: #333333; /* Change font color */
}

h3 {
font-family: "Courier New", Courier, monospace; /*"Franklin Gothic Medium",
"Arial Narrow", Arial, sans-serif;
/* Change the font family */
font-size: 28px; /* Adjust the font size */
color: #333333; /* Change font color */
}

ul.topnav {
list-style-type: none;
margin: auto;
```

```
padding: 0;
overflow: hidden;
background-color: #9D9167;
width: 70%;
}
```

```
ul.topnav li {float: left;}
```

```
ul.topnav li a {
display: block;
color: white;
text-align: center;
padding: 14px 16px;
text-decoration: none;
}
```

```
ul.topnav li a:hover:not(.active) {background-color: #743014;}
```

```
ul.topnav li a.active {background-color: #84592B;}
```

```
ul.topnav li.right {float: right;}
```

```
@media screen and (max-width: 600px) {
ul.topnav li.right,
ul.topnav li {float: none;}
}
```

```
img {
display: block;
margin-left: auto;
margin-right: auto;
}
</style>
```

```
<title>Home : NodeMCU V3 ESP8266 / ESP12E with MYSQL Database</title>
</head>
```

```
<body>
```

```
<h2>DEVELOPMENT OF A BOOK TRACKING SYSTEM IN THE
LIBRARY USING A MICROCONTROLLER AND RFID</h2>
```

```
<ul class="topnav">
<li><a class="active" href="home.php">Home</a></li>
<li><a href="user data.php">Admin Record</a></li>
<li><a href="registration.php">Registration</a></li>
<li><a href="read tag.php">Book Display</a></li>
<li><a href="book record.php">Book Record</a></li>
</ul>
```

```
<br>
```

```
<h3>This Website will display book information</h3>
```

```


</body>
</html>

```

Appendix D User data.php

```

<?php
require 'database.php';

// Establish the connection
$pdo = Database::connect();
$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

// Fetch all book data from the table, excluding Location and Date
$sql = "SELECT BookID, Title, Author, BookGenre FROM
table_nodemcu_rfidrc522_mysql"; // Updated SQL query
$q = $pdo->prepare($sql);
$q->execute();
$bookData = $q->fetchAll(PDO::FETCH_ASSOC);

Database::disconnect();
?>

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet">
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></scri
pt>
<title>User Data : NodeMCU V3 ESP8266 / ESP12E with MYSQL Database</title>
<style>
table {
margin-top: 20px;
width: 100%;
border-collapse: collapse;
}

body {
background-color: #E8D1A7;
}

```



```

th, td {
    padding: 10px;
    text-align: left;
    border: 1px solid #ddd;
}
th {
    background-color: #f2f2f2;
}
td {
    background-color: #fafafa;
}
.topnav {
    list-style-type: none;
    margin: auto;
    padding: 0;
    overflow: hidden;
    background-color: #9D9167;
    width: 70%;
}
.topnav li {
    float: left;
}
.topnav li a {
    display: block;
    color: white;
    text-align: center;
    padding: 14px 16px;
    text-decoration: none;
}
.topnav li a:hover:not(.active) {
    background-color: #743014;
}
.topnav li a.active {
    background-color: #84592B;
}
.topnav li.right {
    float: right;
}

h2 {
    font-family: "Courier New", Courier, monospace;
    font-size: 28px;
    color: #333333;
}

h3 {
    font-family: "Courier New", Courier, monospace;
    font-size: 28px;

```

```

        color: #333333;
    }
</style>
</head>

<body>
    <h2 class="text-center">DEVELOPMENT OF A BOOK TRACKING SYSTEM IN
    THE LIBRARY USING A MICROCONTROLLER AND RFID</h2>

    <ul class="topnav">
        <li><a href="home.php">Home</a></li>
        <li><a class="active" href="user data.php">Admin Record</a></li>
        <li><a href="registration.php">Registration</a></li>
        <li><a href="read tag.php">Book Display</a></li>
        <li><a href="book record.php">Book Record</a></li>
    </ul>
    <br>

    <div class="container">
        <h3 class="text-center">Book Data Table</h3>
        <table class="table table-striped table-bordered">
            <thead>
                <tr bgcolor="#442D1C" style="color:#FFFFFF;">
                    <th>BookID</th>
                    <th>Title</th>
                    <th>Author</th>
                    <th>Book Genre</th>
                    <th>Action</th>
                </tr>
            </thead>
            <tbody>
                <?php
                if (!empty($bookData)) {
                    foreach ($bookData as $row) {
                        echo '<tr>';
                        echo '<td>' . $row['BookID'] . '</td>';
                        echo '<td>' . $row['Title'] . '</td>';
                        echo '<td>' . $row['Author'] . '</td>';
                        echo '<td>' . $row['BookGenre'] . '</td>'; // Removed Location and
Date
                        echo '<td>'; // Open a new cell for buttons
                        echo '<a class="btn btn-success" href="user data edit page.php?id=' .
$row['BookID'] . '>Edit</a>';
                        echo ' ';
                        echo '<a class="btn btn-danger" href="user data delete page.php?id=' .
$row['BookID'] . '>Delete</a>';
                        echo '</td>';
                        echo '</tr>';
                    }
                }
            </tbody>
        </table>
    </div>

```

```

        } else {
            echo "<tr><td colspan='4' style='text-align:center;'>No data
available</td></tr>"; // Adjusted colspan
        }
    ?>
</tbody>
</table>
</div>
</body>
</html>

```

Appendix E User data edit page.php

```

<?php
require 'database.php';

// Initialize variables
$bookID = $title = $author = $BookGenre = "";

// Check if the BookID is passed in the URL
if (isset($_GET['id']) && !empty($_GET['id'])) {
    $bookID = $_GET['id'];

    // Establish the connection to the database
    $pdo = Database::connect();
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // Fetch the book data using BookID
    $sql = "SELECT Title, Author, BookGenre FROM
table_nodemcu_rfdr522_mysql WHERE BookID = ?";
    $q = $pdo->prepare($sql);
    $q->execute(array($bookID));

    // Check if a row is returned
    $row = $q->fetch(PDO::FETCH_ASSOC);

    // If a row is found, populate the variables
    if ($row) {
        $title = $row['Title'];
        $author = $row['Author'];
        $BookGenre = $row['BookGenre'];
    } else {
        // If no book is found, redirect or show an error
        echo "No book found with ID: $bookID";
        exit();
    }
}

```

```

        Database::disconnect();
    } else {
        // Redirect if no BookID is provided
        header("Location: user data.php");
        exit();
    }

    // Handle form submission (update data)
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        $bookID = $_POST['BookID'];
        $title = $_POST['Title'];
        $author = $_POST['Author'];
        $BookGenre = $_POST['BookGenre'];

        // Establish the connection to the database
        $pdo = Database::connect();
        $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

        // Prepare the update query
        $sql = "UPDATE table_nodemcu_rfidrc522_mysql SET Title = ?, Author = ?,
BookGenre = ? WHERE BookID = ?";
        $q = $pdo->prepare($sql);
        $q->execute(array($title, $author, $BookGenre, $bookID));

        Database::disconnect();

        // Redirect to user data page after update
        header("Location: user data.php");
        exit();
    }
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet">
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></scri
pt>
    <title>Edit Book Data</title>
    <style>
        .form-container {
            width: 50%;
            margin: auto;
            padding-top: 20px;

```

```

    }

    body {
        background-color: #E8D1A7;
    }
</style>
</head>
<body>
    <h2 class="text-center">Edit Book Data</h2>

    <div class="form-container">
        <form action="user data edit page.php?id=<?php echo $bookID; ?>"
method="POST">
            <div class="mb-3">
                <label for="BookID" class="form-label">BookID</label>
                <input type="text" class="form-control" id="BookID" name="BookID"
value="<?php echo $bookID; ?>" readonly>
            </div>
            <div class="mb-3">
                <label for="Title" class="form-label">Title</label>
                <input type="text" class="form-control" id="Title" name="Title"
value="<?php echo htmlspecialchars($title); ?>" required>
            </div>
            <div class="mb-3">
                <label for="Author" class="form-label">Author</label>
                <input type="text" class="form-control" id="Author" name="Author"
value="<?php echo htmlspecialchars($author); ?>" required>
            </div>
            <div class="mb-3">
                <label for="BookGenre" class="form-label">Book Genre</label>
                <input type="text" class="form-control" id="BookGenre" name="BookGenre"
value="<?php echo htmlspecialchars($BookGenre); ?>" required>
            </div>
            <button type="submit" class="btn btn-primary">Update</button>
            <a href="user data.php" class="btn btn-secondary">Back</a>
        </form>
    </div>
</body>
</html>

```

Appendix F User data edit tb.php

```

<?php
    require 'database.php';

```

```

$Id = null;
if (!empty($_GET['BookID'])) {
    $Id = $_REQUEST['BookID'];
}

if (!empty($_POST)) {
    // Keep track of the posted values
    $BookID = $_POST['BookID'];
    $Title = $_POST['Title'];
    $Author = $_POST['Author'];
    $BookGenre = $_POST['BookGenre'];

    // Connect to the database
    $pdo = Database::connect();
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // Update query without Date and Location fields
    $sql = "UPDATE table_nodemcu_rfidrc522_mysql SET Title = ?, Author = ?,
BookGenre = ? WHERE BookID = ?";
    $q = $pdo->prepare($sql);
    $q->execute(array($Title, $Author, $BookGenre, $BookID)); // Corrected parameter
order

    Database::disconnect();
    header("Location: user data.php");
    exit();
}
?>

```

Appendix G User data delete page.php

```

<?php
require 'database.php';
$Id = 0;

// Fetch the id from the GET parameter if it's available
if ( !empty($_GET['id'])) {
    $Id = $_GET['id']; // Use GET to retrieve the ID from the URL
}

// Handle the delete action when the form is submitted
if ( !empty($_POST)) {
    // Ensure the ID is retrieved from the POST request
    $Id = $_POST['id'];
}

```

```

// Delete data from the database using the correct column name (BookID)
try {
    $pdo = Database::connect();
    $pdo->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

    // Assuming the identifier is BookID and not id
    $sql = "DELETE FROM table_nodemcu_rfidrc522_mysql WHERE BookID = ?";
    $q = $pdo->prepare($sql);
    $q->execute(array($id));

    // Check if the delete was successful
    if ($q->rowCount() > 0) {
        // Redirect to the user data page after deletion
        header("Location: user data.php");
        exit();
    } else {
        echo "No record found with that ID.";
    }

    Database::disconnect();
} catch (PDOException $e) {
    echo "Error: " . $e->getMessage();
}
}
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <link href="css/bootstrap.min.css" rel="stylesheet">
    <script src="js/bootstrap.min.js"></script>
    <title>Delete : NodeMCU V3 ESP8266 / ESP12E with MYSQL Database</title>
    <style>
        body {
            background-color: #E8D1A7;
        }
    </style>
</head>
<body>
    <h2 align="center">DEVELOPMENT OF A BOOK TRACKING SYSTEM IN THE
LIBRARY USING A MICROCONTROLLER AND RFID</h2>

    <div class="container">
        <div class="span10 offset1">
            <div class="row">
                <h3 align="center">Remove Book</h3>
            </div>

```

```

        <!-- Form for deleting the user -->
        <form class="form-horizontal" action="user data delete page.php"
method="post">
        <!-- Hidden input to pass the id for deletion -->
        <input type="hidden" name="id" value="<?php echo $id; ?>" />

        <p class="alert alert-error">Are you sure you want to remove this book?</p>

        <div class="form-actions">
        <button type="submit" class="btn btn-danger">Yes</button>
        <a class="btn" href="user data.php">No</a>
        </div>
        </form>
    </div>
</div> <!-- /container -->
</body>
</html>

```

Appendix H Registration.php

```

<?php
$Write="<?php $" . "UIDresult="; " . "echo $" . "UIDresult;" . " ?>";
file_put_contents('UIDContainer.php',$Write);
?>

<!DOCTYPE html>
<html lang="en">
<html>
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta charset="utf-8">
    <link href="css/bootstrap.min.css" rel="stylesheet">
    <script src="js/bootstrap.min.js"></script>
    <script src="jquery.min.js"></script>
    <script>
        $(document).ready(function(){
            $("#getUID").load("UIDContainer.php");
            setInterval(function() {
                $("#getUID").load("UIDContainer.php");
            }, 500);
        });
    </script>

    <style>

```



```

html {
    font-family: Arial;
    display: inline-block;
    margin: 0px auto;
}

body {
    background-color: #E8D1A7;
}

h2 {
    font-family: "Courier New", Courier, monospace; /*"Franklin Gothic Medium",
"Arial Narrow", Arial, sans-serif;
    /* Change the font family */
    font-size: 28px; /* Adjust the font size */
    color: #333333; /* Change font color */
}

h3 {
    font-family: "Courier New", Courier, monospace; /*"Franklin Gothic Medium",
"Arial Narrow", Arial, sans-serif;
    /* Change the font family */
    font-size: 28px; /* Adjust the font size */
    color: #333333; /* Change font color */
}

textarea {
    resize: none;
}

ul.topnav {
    list-style-type: none;
    margin: auto;
    padding: 0;
    overflow: hidden;
    background-color: #9D9167;
    width: 70%;
}

ul.topnav li {float: left;}

ul.topnav li a {
    display: block;
    color: white;
    text-align: center;
    padding: 14px 16px;
    text-decoration: none;
}

```

```

ul.topnav li a:hover:not(.active) {background-color: #743014;}

ul.topnav li a.active {background-color: #84592B;}

ul.topnav li.right {float: right;}

@media screen and (max-width: 600px) {
  ul.topnav li.right,
  ul.topnav li {float: none;}
}
</style>

<title>Registration : NodeMCU V3 ESP8266 / ESP12E with MYSQL Database</title>
</head>

<body>

  <h2 align="center">DEVELOPMENT OF A BOOK TRACKING SYSTEM IN THE
  LIBRARY USING A MICROCONTROLLER AND RFID</h2>
  <ul class="topnav">
    <li><a href="home.php">Home</a></li>
    <li><a href="user data.php">Admin Reord</a></li>
    <li><a class="active" href="registration.php">Registration</a></li>
    <li><a href="read tag.php">Book Display</a></li>
    <li><a href="book record.php">Book Record</a></li>
  </ul>

  <div class="container">
    <br>
    <div class="center" style="margin: 0 auto; width:495px; border-style: solid; border-
    color: #442D1C;">
      <div class="row">
        <h3 align="center">Registration Form</h3>
      </div>
      <br>
      <form class="form-horizontal" action="insertDB.php" method="post" >
        <div class="control-group">
          <label class="control-label">BookID</label>
          <div class="controls">
            <textarea name="bookID" id="getUID" placeholder="Please Tag your Card /
            Key Chain to display BookID" rows="1" cols="1" required></textarea>
          </div>
        </div>

        <div class="control-group">
          <label class="control-label">Title</label>
          <div class="controls">
            <input id="div_refresh" name="title" type="text" placeholder="" required>
          </div>

```

```

</div>

<div class="control-group">
  <label class="control-label">Author</label>
  <div class="controls">
    <input name="author" type="text" placeholder="" required>
  </div>
</div>

<div class="control-group">
  <label class="control-label">Book Genre</label>
  <div class="controls">
    <input name="BookGenre" type="text" placeholder="" required>
  </div>
</div>

<div class="form-actions">
  <button type="submit" class="btn btn-success">Save</button>
</div>
</form>
</div>
</div> <!-- /container -->
</body>
</html>

```

اونیورسیتی تکنیکل ملیسیا ملاک
 UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Appendix I Read tag.php

```

<?php
// Write the scanned UID to a PHP file to be used by JavaScript
$Write = "<?php \$UIDresult="; echo \$UIDresult; ?>";
file_put_contents('UIDContainer.php', $Write);
?>

<!DOCTYPE html>
<html>
  <head>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta charset="utf-8">
    <link href="css/bootstrap.min.css" rel="stylesheet">
    <script src="js/bootstrap.min.js"></script>
    <script src="jquery.min.js"></script>
    <script>
      $(document).ready(function(){

```

```

    $("#getUID").load("UIDContainer.php"); // Load UID on page load
    setInterval(function() {
        $("#getUID").load("UIDContainer.php"); // Refresh every 500ms
    }, 500);
});
</script>
<style>
    html {
        font-family: Arial;
        display: inline-block;
        margin: 0px auto;
        text-align: center;
    }

    body {
        background-color: #E8D1A7;
    }

    h2 {
        font-family: "Courier New", Courier, monospace;
        font-size: 28px;
        color: #333333;
    }

    h3 {
        font-family: "Courier New", Courier, monospace;
        font-size: 28px;
        color: #333333;
    }

    ul.topnav {
        list-style-type: none;
        margin: auto;
        padding: 0;
        overflow: hidden;
        background-color: #9D9167;
        width: 70%;
    }

    ul.topnav li { float: left; }

    ul.topnav li a {
        display: block;
        color: white;
        text-align: center;
        padding: 14px 16px;
        text-decoration: none;
    }

```

```

ul.topnav li a:hover:not(.active) { background-color: #743014; }

ul.topnav li a.active { background-color: #84592B; }

ul.topnav li.right { float: right; }

@media screen and (max-width: 600px) {
  ul.topnav li.right,
  ul.topnav li { float: none; }
}

td.lf {
  padding-left: 15px;
  padding-top: 12px;
  padding-bottom: 12px;
}
</style>

<title>Read Tag : NodeMCU V3 ESP8266 / ESP12E with MYSQL Database</title>
</head>

<body>
  <h2 align="center">DEVELOPMENT OF A BOOK TRACKING SYSTEM IN
  THE LIBRARY USING A MICROCONTROLLER AND RFID</h2>
  <ul class="topnav">
    <li><a href="home.php">Home</a></li>
    <li><a href="user data.php">Admin Record</a></li>
    <li><a href="registration.php">Registration</a></li>
    <li><a class="active" href="read tag.php">Book Display</a></li>
    <li><a href="book record.php">Book Record</a></li>
  </ul>

  <br>

  <p id="getUID" hidden></p>

  <br>

  <div id="show_user_data">
    <!-- Placeholder for user data -->
    <form>
      <table width="452" border="1" bordercolor="#FFFFFF" align="center"
      cellpadding="0" cellspacing="1" bgcolor="#000" style="padding: 2px">
        <tr>
          <td height="40" align="center" bgcolor="#442D1C"><font
          color="#FFFFFF"><b>User Data</b></font></td>
        </tr>
      </table>
    </form>
  </div>

```

```

        <tr>
            <td bgcolor="#f9f9f9">
                <table width="452" border="0" align="center" cellpadding="5"
cellspacing="0">

                    <tr>
                        <td width="113" align="left" class="lf">BookID</td>
                        <td style="font-weight:bold">:</td>
                        <td align="left">-----</td>
                    </tr>

                    <tr bgcolor="#f2f2f2">
                        <td align="left" class="lf">Title</td>
                        <td style="font-weight:bold">:</td>
                        <td align="left">-----</td>
                    </tr>

                    <tr>
                        <td align="left" class="lf">Author</td>
                        <td style="font-weight:bold">:</td>
                        <td align="left">-----</td>
                    </tr>

                    <tr>
                        <td align="left" class="lf">Timestamp</td>
                        <td style="font-weight:bold">:</td>
                        <td align="left">-----</td>
                    </tr>

                    <tr>
                        <td align="left" class="lf">Location</td>
                        <td style="font-weight:bold">:</td>
                        <td align="left">-----</td>
                    </tr>

                </table>
            </td>
        </tr>
    </table>
</form>
</div>

<script>
    var myVar = setInterval(myTimer, 1000);
    var myVar1 = setInterval(myTimer1, 1000);
    var oldID = "";
    clearInterval(myVar1);

    function myTimer() {

```

```

var getID = document.getElementById("getUID").innerHTML;
oldID = getID;
if (getID != "") {
    myVar1 = setInterval(myTimer1, 500);
    showUser (getID);
    clearInterval(myVar);
}
}

function myTimer1() {
    var getID = document.getElementById("getUID").innerHTML;
    if (oldID != getID) {
        myVar = setInterval(myTimer, 500);
        clearInterval(myVar1);
    }
}

function showUser (str) {
    if (str == "") {
        document.getElementById("show_user_data").innerHTML = "";
        return;
    } else {
        if (window.XMLHttpRequest) {
            xmlhttp = new XMLHttpRequest();
        } else {
            xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
        }
        xmlhttp.onreadystatechange = function() {
            if (this.readyState == 4 && this.status == 200) {
                document.getElementById("show_user_data").innerHTML
this.responseText;
            }
        };
        xmlhttp.open("GET", "read tag user data.php?id=" + str, true);
        xmlhttp.send();
    }
}

var blink = document.getElementById('blink');
setInterval(function() {
    blink.style.opacity = (blink.style.opacity == 0 ? 1 : 0);
}, 750);
</script>
</body>
</html>

```

Appendix J Read tag user data.php

```
<?php
require_once 'vendor/autoload.php';
use GeoIp2\Database\Reader;
require 'database.php'; // Ensure your database connection file is included

// Start session (if using session storage for readerID)
session_start();

// Initialize message variable
$msg = null;

// Include the PHP files that contain the UID and readerID values
include('readerContainer.php'); // This will set the $readerID variable
include('UIDContainer.php'); // This will set the $UIDresult variable

// Check if 'id' parameter is set in the URL
if (isset($_GET['id'])) {
    // Get the book ID from the URL parameter
    $id = $_GET['id'];

    // Default to "Unknown Reader" if $readerID is not set
    $location = "Unknown Reader";

    // Determine the location based on the reader ID
    if ($readerID == 1) {
        $location = "Shelf A";
    } elseif ($readerID == 2) {
        $location = "Shelf B";
    }

    // Connect to the database
    $pdo = Database::connect();
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // Check if the book exists in the database
    $checkSql = "SELECT BookID FROM table_nodemcu_rfidrc522_mysql WHERE
BookID = ?";
    $checkStmt = $pdo->prepare($checkSql);
    $checkStmt->execute([$id]);

    // If the book exists, insert/update the location
    if ($checkStmt->rowCount() > 0) {
        // Insert or update the data with the new location
        $sql1 = "INSERT INTO page_data (BookID, Location) VALUES (?, ?)";
        $q1 = $pdo->prepare($sql1);
        $q1->execute([$id, $location]);
    }
}
```



```

}

// Fetch the most recent record, ensuring it includes the correct location
$sql = "
    SELECT
        t.BookID, t.Title, t.Author, p.Timestamp, p.Location
    FROM
        table_nodemcu_rfidrc522_mysql AS t
    LEFT JOIN
        page_data AS p
    ON
        t.BookID = p.BookID
    WHERE
        t.BookID = ?
    ORDER BY
        p.Timestamp DESC
    LIMIT 1
";
$q = $pdo->prepare($sql);
$q->execute([$id]);
$data = $q->fetch(PDO::FETCH_ASSOC);

// Fallback if data is missing
if (!$data) {
    $msg = "The ID of your Card / KeyChain is not registered !!!";
    $data['BookID'] = $id;
    $data['Title'] = "----- ";
    $data['Author'] = "----- ";
    $data['Timestamp'] = "----- ";
    $data['Location'] = $location; // Ensure the location defaults to the current reader's
location
}
}

// Display the user data in a table
echo '<table width="452" border="1" bordercolor="#442D1C" align="center"
cellpadding="0" cellspacing="1" bgcolor="#000" style="padding: 2px">
    <tr>
        <td height="40" align="center" bgcolor="#442D1C"><font
color="#FFFFFF"><b>User Data</b></font></td>
    </tr>
    <tr>
        <td bgcolor="#f9f9f9">
            <table width="452" border="0" align="center" cellpadding="5"
cellspacing="0">
                <tr>
                    <td width="113" align="left" class="lf">BookID</td>
                    <td style="font-weight:bold">:</td>
                    <td align="left">' . htmlspecialchars($data['BookID']) . '</td>

```

```

        </tr>
        <tr bgcolor="#f2f2f2">
            <td align="left" class="lf">Title</td>
            <td style="font-weight:bold">:</td>
            <td align="left">' . htmlspecialchars($data['Title']) . '</td>
        </tr>
        <tr>
            <td align="left" class="lf">Author</td>
            <td style="font-weight:bold">:</td>
            <td align="left">' . htmlspecialchars($data['Author']) . '</td>
        </tr>
        <tr>
            <td align="left" class="lf">Timestamp</td>
            <td style="font-weight:bold">:</td>
            <td align="left">' . htmlspecialchars($data['Timestamp']) . '</td>
        </tr>
        <tr>
            <td align="left" class="lf">Location</td>
            <td style="font-weight:bold">:</td>
            <td align="left">' . htmlspecialchars($data['Location']) . '</td>
        </tr>
    </table>
</td>
</tr>
</table>';
?>

```

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Appendix K Book record.php

```

<?php
require 'database.php';

// Establish the connection
$pdo = Database::connect();
$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

// Fetch data from two tables using INNER JOIN
$sql = "SELECT
    t1.BookID,
    t1.Title,
    t1.Author,
    t2.Timestamp,
    t2.Location
FROM table_nodemcu_rfidrc522_mysql t1
INNER JOIN page_data t2

```

```

ON t1.BookID = t2.BookID";

$q = $pdo->prepare($sql);
$q->execute();
$bookData = $q->fetchAll(PDO::FETCH_ASSOC);

Database::disconnect();
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet">
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></scri
pt>
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
    <title>Book Record : NodeMCU V3 ESP8266 / ESP12E with MYSQL
Database</title>
    <style>
        table {
            margin-top: 20px;
            width: 100%;
            border-collapse: collapse;
        }
        body {
            background-color: #E8D1A7;
        }

        th, td {
            padding: 10px;
            text-align: left;
            border: 1px solid #ddd;
        }
        th {
            background-color: #f2f2f2;
        }
        td {
            background-color: #fafafa;
        }
        .topnav {
            list-style-type: none;
            margin: auto;
            padding: 0;
            overflow: hidden;

```

```

        background-color: #9D9167;
        width: 70%;
    }
    .topnav li {
        float: left;
    }
    .topnav li a {
        display: block;
        color: white;
        text-align: center;
        padding: 14px 16px;
        text-decoration: none;
    }
    .topnav li a:hover:not(.active) {
        background-color: #743014;
    }
    .topnav li a.active {
        background-color: #84592B;
    }
    .topnav li.right {
        float: right;
    }
    h2 {
        font-family: "Courier New", Courier, monospace;
        font-size: 28px;
        color: #333333;
    }
    h3 {
        font-family: "Courier New", Courier, monospace;
        font-size: 28px;
        color: #333333;
    }
</style>
</head>

<body>
    <h2 class="text-center">DEVELOPMENT OF A BOOK TRACKING SYSTEM IN
    THE LIBRARY USING A MICROCONTROLLER AND RFID</h2>

    <ul class="topnav">
        <li><a href="home.php">Home</a></li>
        <li><a href="user data.php">Admin Record</a></li>
        <li><a href="registration.php">Registration</a></li>
        <li><a href="read tag.php">Book Display</a></li>
        <li><a class="active" href="book record.php">Book Record</a></li>
    </ul>
    <br>

```

```

<div class="container">
  <h3 class="text-center">Book Data Table</h3>
  <table class="table table-striped table-bordered">
    <thead>
      <tr bgcolor="#442D1C" style="color:#FFFFFF;">
        <th>BookID</th>
        <th>Title</th>
        <th>Author</th>
        <th>Timestamp</th>
        <th>Location</th>
      </tr>
    </thead>
    <tbody id="book-data">
      <?php
      if (!empty($bookData)) {
        foreach ($bookData as $row) {
          echo '<tr>';
          echo '<td>' . $row['BookID'] . '</td>';
          echo '<td>' . $row['Title'] . '</td>';
          echo '<td>' . $row['Author'] . '</td>';
          echo '<td>' . $row['Timestamp'] . '</td>'; // Display the Timestamp
          echo '<td>' . (isset($row['Location']) ? $row['Location'] : 'N/A') . '</td>'; //
Optional for Location
          echo '</tr>';
        }
      } else {
        echo "<tr><td colspan='5' style='text-align:center;'>No data available</td></tr>";
      }
      ?>
    </tbody>

  </table>
</div>
</body>
</html>

```

Appendix L Database.php

```

<?php
class Database
{
  // Database credentials
  private static $dbName = 'node_rfidrc522_mysql'; // Your database name
  private static $dbHost = 'localhost';           // Your database host

```

```

private static $dbUsername = 'root';          // Your database username
private static $dbUserPassword = 'root123';  // Your database password

// The PDO connection instance
private static $cont = null;

// Private constructor to prevent initialization
public function __construct() {
    die('Init function is not allowed');
}

/**
 * Connect to the database using PDO.
 * Uses the Singleton pattern to ensure only one instance.
 *
 * @return PDO instance
 */
public static function connect()
{
    // If connection already exists, return the existing connection
    if (null == self::$cont) {
        try {
            // Set connection options
            $options = array(
                PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION, // Enable
error handling
                PDO::ATTR_TIMEOUT => 30 // Set connection timeout
            );

            // Attempt to connect using PDO
            self::$cont = new PDO(
                "mysql:host=" . self::$dbHost . ";dbname=" . self::$dbName,
                self::$dbUsername,
                self::$dbUserPassword,
                $options
            );

            // Set the character set to UTF-8
            self::$cont->exec("SET NAMES 'utf8'");

        } catch (PDOException $e) {
            // Log error and show user-friendly message
            error_log("Database Connection Error: " . $e->getMessage());
            die('Unable to connect to the database at this time. Please try again later.');
```

```

    }

    /**
     * Disconnect from the database.
     */
    public static function disconnect()
    {
        // Close the connection by setting it to null
        self::$cont = null;
    }
}
?>

```

Appendix M getUID.php

```

<?php
if (isset($_POST["UIDresult"]) && isset($_POST["readerID"])) {
    $UIDresult = $_POST["UIDresult"];
    $readerID = $_POST["readerID"];

    // Save the UID to UIDContainer.php
    $Write = "<?php $" . "UIDresult=" . $UIDresult . " "; " . "echo $" . "UIDresult;" . " ?>";
    file_put_contents('UIDContainer.php', $Write);

    // Save the readerID to readerContainer.php
    $Write = "<?php $" . "readerID=" . $readerID . " "; " . "echo $" . "readerID;" . " ?>";
    file_put_contents('readerContainer.php', $Write);
} else {
    echo "Missing UIDresult or readerID!";
}
?>

```

Appendix N insertDB.php

```

<?php
require 'database.php';

if (!empty($_POST)) {
    // Debugging: print the post data
    print_r($_POST);
}

```

```

// Keep track of post values (using isset to avoid warnings)
$BookID = isset($_POST['bookID']) ? $_POST['bookID'] : "";
$title = isset($_POST['title']) ? $_POST['title'] : "";
$Author = isset($_POST['author']) ? $_POST['author'] : "";
$BookGenre = isset($_POST['BookGenre']) ? $_POST['BookGenre'] : "";

// Check if the BookID already exists in the database
$pdo = Database::connect();
$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
$sql_check = "SELECT COUNT(*) FROM table_nodemcu_rfdr522_mysql WHERE
BookID = ?";
$q_check = $pdo->prepare($sql_check);
$q_check->execute(array($BookID));
$count = $q_check->fetchColumn();

// If the BookID already exists, handle the error (e.g., don't insert, or update the record)
if ($count > 0) {
    echo "Error: This BookID already exists in the database.";
    Database::disconnect();
    exit;
}

// Insert data if no duplicate BookID found
$sql = "INSERT INTO table_nodemcu_rfdr522_mysql (BookID, Title, Author,
BookGenre) VALUES (?, ?, ?, ?)";
$q = $pdo->prepare($sql);
$q->execute(array($BookID, $title, $Author, $BookGenre));

Database::disconnect();
header("Location: user data.php");
}
?>

```

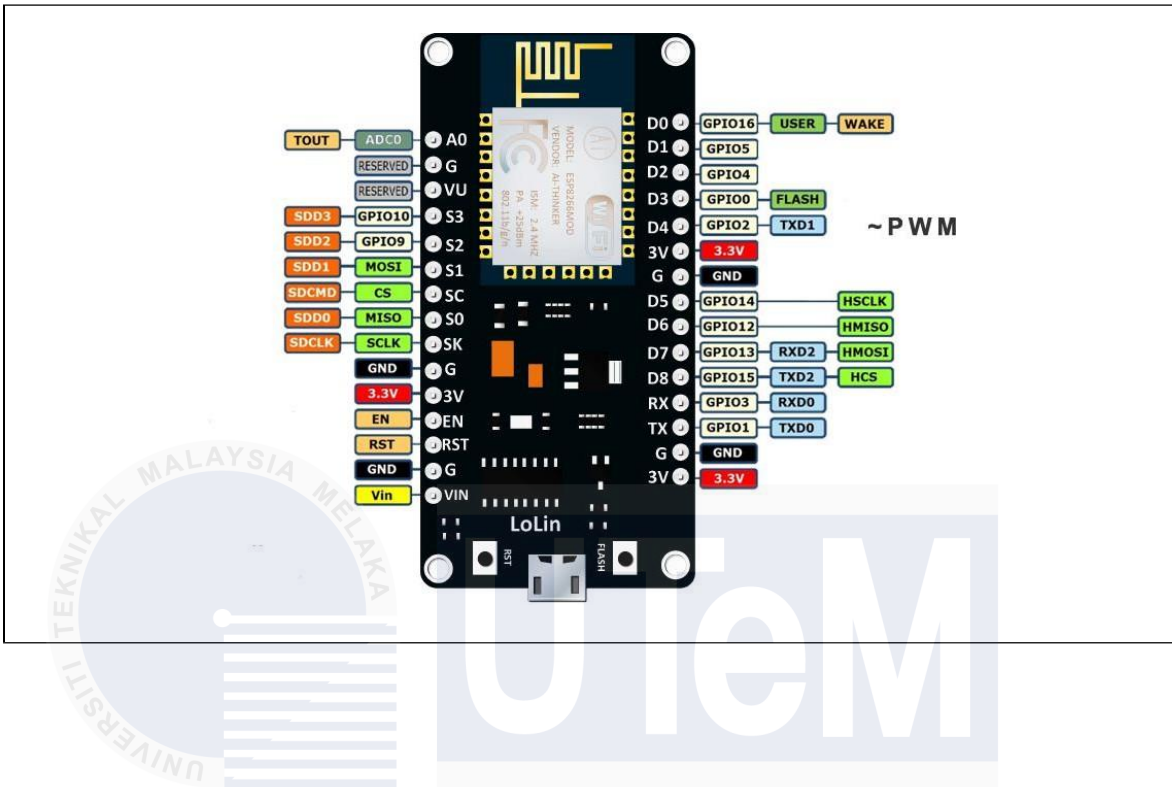
Appendix O UIDContainer.php

```
<?php $UIDresult=""; echo $UIDresult; ?>
```

Appendix P readerContainer.php

```
<?php $readerID=""; echo $readerID; ?>
```


Appendix Q ESP8266 pintout



Appendix R RFID reader pinout

