IoT-Enabled Smart Shopping Carts: Automating Checkout Processes for Frictionless Shopping

by

Kiran Chander Ravichandran

Bachelor of Engineering, Electrical and Electronics Engineering,
Anna University, Chennai, India, 2019

A Project Submitted in Partial Fulfillment of the Requirements for the Degree of
MASTER OF ENGINEERING
in the Department of Electrical and Computer Engineering

University of Victoria

IoT-Enabled Smart Shopping Carts: Automating Checkout Processes for Frictionless Shopping

by

Kiran Chander Ravichandran
Bachelor of Engineering, Electrical and Electronics Engineering,
Anna University, Chennai, India, 2019

Supervisory Committee

Dr. Ilamparithi Thirumarai Chelvan, Supervisor

(Department of Electrical and Computer Engineering)

Dr. Navneet Kaur Popli, Co-Supervisor

(Department of Electrical and Computer Engineering)

ABSTRACT

The growing crowds at shopping destinations like malls, grocery stores, supermarkets, and discount stores highlight the necessity of refining billing protocols to handle the increased number of shoppers. Moreover, considering the decline in sales experienced by physical stores like grocery stores in the face of growing online shopping trends, it becomes imperative to address this challenge and devise viable solutions. The Smart Shopping Cart project aims to enhance the retail shopping experience by leveraging the Internet of Things (IoT) technology to create a smart device that can be attached to standard shopping carts. By incorporating features such as RFID capabilities, MySQL database connectivity, and the Raspberry Pi Pico W microcontroller, the system provides seamless integration with user-friendly interfaces, allowing customers to interact with the shopping cart and scan products themselves effortlessly. This approach expedites the checkout process, reduces wait times, and improves operational efficiency for stores, while also offering valuable insights into consumer purchase patterns for inventory control and targeted marketing. By eliminating the need for manual scanning at checkout, customers save time and enjoy added convenience, thereby enhancing the overall shopping experience. Ultimately, Smart Carts offer an innovative application of IoT technology to streamline and modernize retail shopping, making it more efficient, convenient, and enjoyable for both customers and businesses alike.

Contents

List of Tables

List of Figures

Abbreviation

AI	Artificial Intelligence
CAGR	Compound Annual Growth Rate
CSS	Cascading Style Sheets
CS	Chip Select
DIP	Dual In-Line Package
ESP	Espressif modules
GDPR	General Data Protection Regulation
GPIO	General Purpose Input/ Output
GND	Ground
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IOT	Internet of Things
LCD	Liquid Crystal Display
MCU	Microcontroller Unit
MISO	Master Input Slave Output
MOSI	Master Output Slave Input
OLED	Organic Light Emitting Diode
PCB	Printed Circuit Board
PHP	Hypertext Preprocessor
RFID	Radio-frequency Identification
RST/RES	Reset
SCK	Serial Clock
SDA	Serial Data Line
SPI	Serial Peripheral Interface
SQL	Structured Query Language
UI	User Interface
UID	Unique Identifier
US	United States
Wi-Fi	Wireless Fidelity

Acknowledgment

Firstly, I would like to thank my Supervisor, Dr. Ilamparithi Thirumarai Chelvan, whose valuable suggestions, guidance, and insights have helped me throughout my Master of Engineering project research and coursework.

I extend my sincere appreciation to the faculty members of the Electrical and Computer Engineering Department at the University of Victoria for providing me with the opportunity to pursue my master's degree and for their continuous encouragement and support.

Finally, I would like to thank my parents for all their support and motivation throughout my Master's.

CHAPTER 1: INTRODUCTION

1.1 Background

In recent years, the retail industry has witnessed a significant transformation driven by advancements in technology. Traditional brick-and-mortar stores are increasingly adopting innovative solutions to enhance customer experience, streamline operations, and stay competitive in the digital age. One of the key technologies revolutionizing the retail landscape is the Internet of Things (IoT), which refers to the network of interconnected devices embedded with sensors, software, and other technologies to collect and exchange data [1]. IoT has enabled retailers to create smarter, more efficient systems that improve inventory management, personalize marketing efforts, and optimize the overall shopping experience [2]. Within the realm of IoT applications in retail, the concept of smart shopping carts has emerged as a promising solution to address common pain points faced by both retailers and customers. Smart shopping carts leverage IoT technology to automate various aspects of the shopping process, such as product identification, inventory tracking, and checkout procedures [3].

One of the key components of smart shopping cart systems is Radio-Frequency Identification (RFID) technology. RFID allows for the wireless identification and tracking of objects using radio waves, making it ideal for inventory management and asset-tracking applications. The combination of RFID and IoT has the potential to transform shopping carts into smart devices, providing customers with instant product information, seamless payment options, and a more interactive shopping journey [4].

Despite the potential benefits offered by IoT-based smart shopping carts, there is a gap in the literature regarding the implementation and evaluation of such systems in real-world retail environments. Therefore, this research aims to address this gap by designing, implementing, and evaluating an IoT-based smart shopping cart application using Raspberry Pi Pico W and RFID technology.

1.2 Problem Definition

In today's fast-paced society, where convenience and efficiency are paramount, the landscape of shopping experiences is evolving rapidly. The rise of online shopping giants like Amazon and eBay has reshaped consumer behavior, leading to a decline in sales for brick-and-mortar stores, particularly grocery stores. Customers now crave instant gratification, preferring the convenience of clicking a button on their devices and having items delivered to their doorsteps. This shift in consumer behavior has resulted in a decline in foot traffic and sales for brick-and-mortar stores, necessitating innovative solutions to attract and retain customers [5].

Moreover, traditional shopping methods are often plagued by inefficiencies such as long checkout queues, limited product information availability, and cumbersome payment processes [6]. Customers expect a seamless and personalized shopping experience that caters to their individual needs and preferences [7] [8]. In this context, there is a clear opportunity to leverage emerging technologies to address these challenges and enhance the shopping experience for customers. Smart shopping devices, equipped with RFID capabilities, microcontrollers, and database connectivity, have the potential to transform traditional shopping carts into intelligent tools that streamline the shopping process and improve customer satisfaction.

The objective of this thesis is to explore the feasibility and effectiveness of implementing smart shopping devices in brick-and-mortar stores. By integrating technologies such as RFID, microcontrollers, and database connectivity, our research aims to develop a solution that combines the convenience of online shopping with the sensory engagement of physical stores. Through collaboration with industry stakeholders and rigorous evaluation of the prototype, the goal is to pave the way for the widespread adoption of smart shopping devices, ushering in a new era of convenience and efficiency in brick-and-mortar stores.

1.3 Market Analysis

The retail industry has experienced a transformative shift in recent years, driven by advancements in technology and evolving consumer preferences. With the rapid digitalization of communication and the proliferation of smart devices, customers now expect enhanced shopping experiences that blend convenience with sensory engagement [9]. In the past, specifically in the year 2017, while e-commerce continued to grow steadily, with online sales projected to reach 10% of total retail sales by 2020, there was still a strong preference among Canadian consumers for in-store shopping experiences, particularly for certain product categories such as appliances/tools [10], as indicated by the data presented in Figure 1.1.

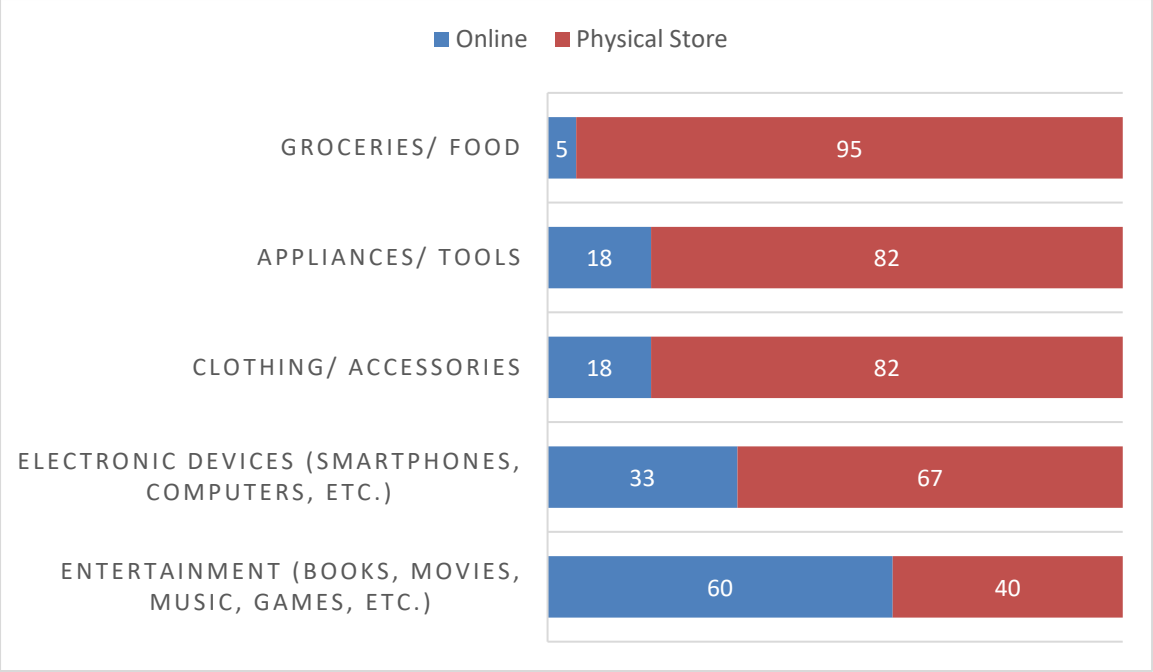


Figure 1.1: Consumer Shopping Preferences in Canada: Product Category Preferences [10]

Despite the rise of online shopping, brick-and-mortar stores still play a vital role in the retail ecosystem, offering customers the opportunity to physically see and touch products before making a purchase. This preference for in-store shopping presents an opportunity for retailers to leverage emerging technologies to enhance the traditional shopping experience. One such technology gaining

traction in the retail sector is IoT-based smart shopping carts, which combine the convenience of online shopping with the sensory engagement of physical stores [2].

Several companies have already embraced smart shopping cart technology, with notable examples including Amazon Go, Caper, and Veeve. These companies have reported increased customer satisfaction and engagement through the implementation of smart shopping carts, which offer features such as automatic checkout, personalized recommendations, and interactive shopping experiences [12].

Market research indicates significant growth potential for the smart shopping carts market, with a projected CAGR of 6.78% from 2021 to 2027 [11]. Factors driving this growth include the increasing demand for automatic checkout systems, the proliferation of smart shopping carts in supermarkets and retail stores, and the growing urbanization and preference for shopping in supermarkets and hypermarkets.

Figure 1.2 illustrates the growth trajectory of the global smart shopping market size, along with its projected expansion until 2027.

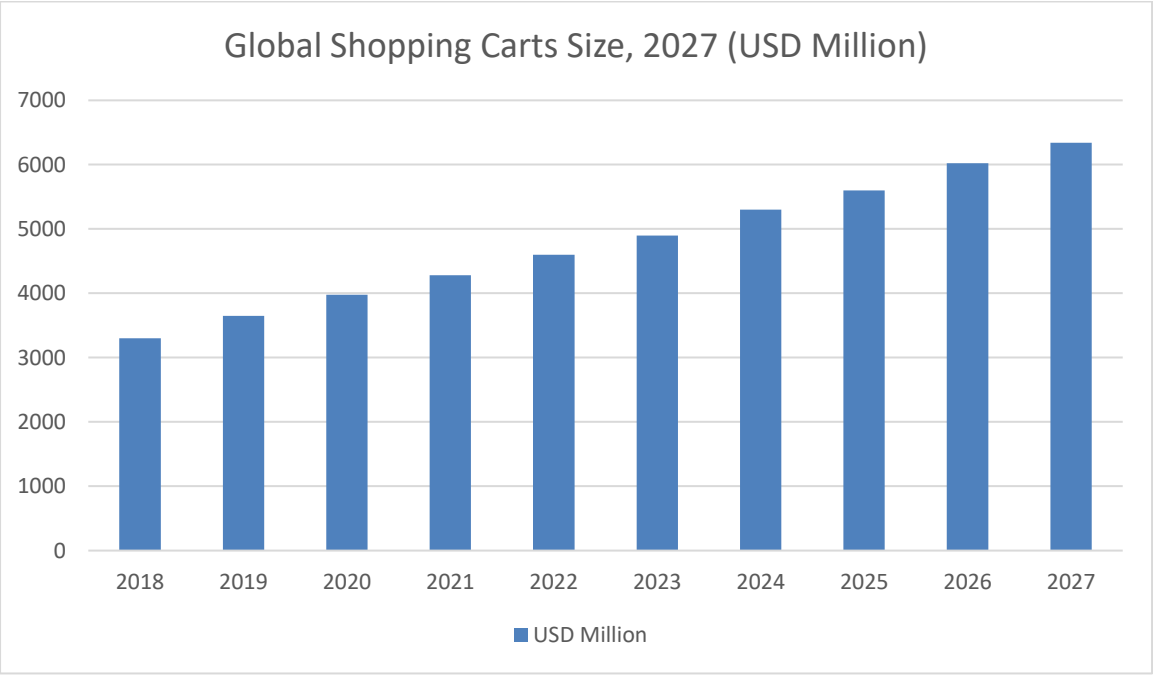


Figure 1.2: Global Smart Shopping Market Size Growth and Projection

The COVID-19 pandemic has both hindered and propelled the growth of the smart shopping carts market. While the pandemic led to disruptions in production and supply chains, resulting in lower-than-anticipated demand, it also accelerated the adoption of contactless shopping solutions, driving demand for smart shopping carts [11].

1.3.1 Key Trends and Insights

The smart shopping carts market is witnessing significant trends that are shaping its growth trajectory. One prominent trend is the increasing demand for smart carts equipped with advanced features such as artificial intelligence (AI) and computer vision technology. These smart shopping carts enhance the shopping experience by automatically scanning and weighing products, displaying prices, and providing personalized recommendations to customers. Market vendors are continuously introducing innovative technologies to improve customer satisfaction and streamline the shopping process, driving the growth of the market [11].

The smart shopping cart market can be segmented into different types, with three primary technologies leading the forefront: Zigbee, RFID, and barcodes. Moreover, the segmentation analysis indicates that RFID technology is projected to outpace existing technologies like barcodes by 2027 [11]. This shift is attributed to the superior capabilities of RFID in terms of data collection, accuracy, and efficiency, making it the preferred choice for smart shopping cart applications. Figure 1.3 depicts the segmentation analysis of the smart shopping carts market projected for 2027.

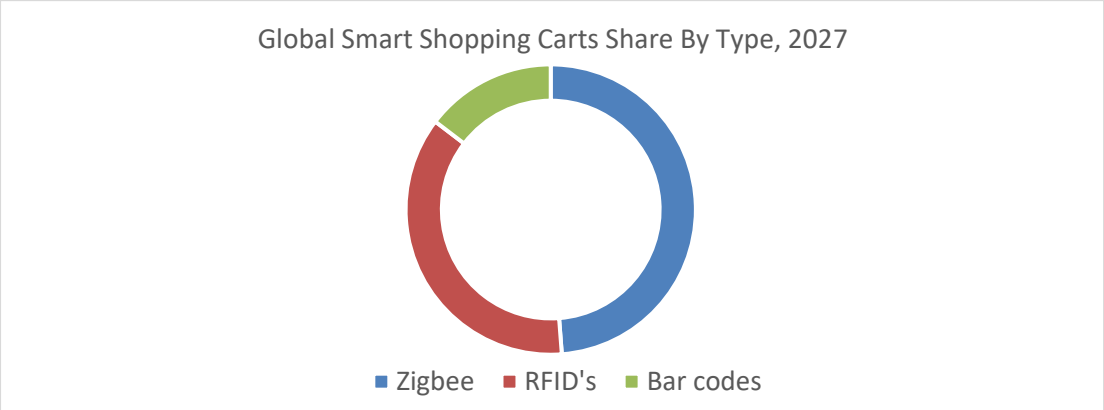


Figure 1.3: Segmentation Analysis of the Smart Shopping Carts Market (Projected for 2027)

Table 1.1 provides a comprehensive summary of the market growth and overview of smart shopping carts, detailing key aspects such as market size values, growth rates, forecast period, historical data availability, regional scope, and segments covered [11].

Report Coverage	Details
Market Size Value In	US$ 4280.18 Million in 2021
Market Size Value By	US$ 6342.84 Million by 2027
Growth Rate	CAGR of 6.78% from 2021 to 2027
Forecast Period	2024-2032
Base Year	2023
Historical Data Available	Yes
Regional Scope	Global
Segments Covered	Type and Application

Table 1.1: Summary of Smart Shopping Carts Market Growth

1.3.2 Limitations

Despite the promising growth prospects, the smart shopping carts market faces certain limitations, one of which is related to privacy and security concerns. The integration of data collection functions in smart shopping carts raises issues regarding the protection of sensitive customer information. With the collection of data on customer preferences, purchase history, and shopping habits, there is a risk of unauthorized access or misuse of data, leading to potential privacy breaches. Regulatory frameworks such as the General Data Protection Regulation (GDPR) in the European Union impose strict rules on data collection, processing, and storage to safeguard customer privacy. Compliance with such regulations poses challenges for retailers utilizing smart shopping carts, potentially hindering market growth.

1.4 Significance of the project

The significance of this project lies in its potential to democratize access to advanced retail technologies, particularly IoT-based smart shopping carts, by offering a cost-effective solution that is accessible to businesses of all sizes. As demonstrated by the findings of the market research, there is a growing demand for smart shopping carts, but the high cost of implementation has been a barrier to widespread adoption, particularly for small and medium-sized retailers [11]. By focusing on providing a low-cost solution, this project aims to address this barrier and unlock the benefits of smart shopping technology for a broader range of businesses.

1.4.1 Benefits of Cost-Effective Solutions

The adoption of cost-effective smart shopping cart solutions offers several benefits to retailers:

1) **Broader Market Appeal**: A more affordable smart shopping cart system appeals to a wider range of retailers, including small and independent stores that previously could not afford such technologies.
2) **Increased Adoption Rates**: Lower costs can lead to higher adoption rates across the retail industry, allowing more stores to provide a sophisticated shopping experience to their customers.
3) **Rapid Return on Investment**: With lower installation and maintenance costs, retailers can realize a faster ROI, leading to higher sales volumes and quicker recoupment of the initial investment.
4) **Flexible Implementation**: Affordability enables retailers to trial the technology with less financial risk, making it easier to test and adapt to their specific store environment before committing to a full rollout.
5) **Competitive Differentiation**: Even budget-conscious retailers can differentiate themselves from competitors by offering services that were once exclusive to larger chains with more resources.

6) **Scaling Opportunity**: Lower investment costs make it more feasible for retailers to implement smart shopping systems across multiple locations, ensuring a consistent and modern shopping experience for all customers.

1.5 Proposed Solution

The proposed solution for the smart shopping cart project utilizes the Raspberry Pi Pico W microcontroller, marking a significant advancement in the field. This solution integrates RFID technology, a MySQL database, and JavaScript for application development, aiming to create a cost-effective and efficient smart shopping cart system.

When a product is scanned using the RFID module, the microcontroller retrieves the unique identifier (UID) of the product. This UID is then sent to the web application via a simple HTTP client script, using HTTP POST requests. The web application, hosted on a server created through XAMPP, consists of a database built with MySQL and a user interface developed using HTML, CSS, Bootstrap, and JavaScript. Upon receiving the UID from the microcontroller, the web application queries the database for the corresponding product details. These details are then displayed to the user, who can choose to add the product to their shopping cart.

Additionally, the proposed solution includes an admin page accessible only to authorized employees. This page allows employees to manage the product database by adding new products, deleting existing ones, or editing product details. Authentication is required for access, which is achieved using employee ID and the RFID reader.

In summary, the proposed solution integrates RFID technology, a microcontroller, and a web application for an affordable and efficient smart shopping cart system. Seamless communication via HTTP POST requests enhances real-time product identification, elevating the shopping experience. By addressing cost barriers and leveraging innovation, it could revolutionize retail, fostering sustainable growth.

1.6 Report Organization

The report has been structured as follows.

Chapter 1 provides an overview of the project, including the background, problem statement, objectives, scope, limitations, and significance of the study.

Chapter 2 explores the landscape of IoT and its applications, previous research on IoT-based smart shopping carts, the relevance of RFID technology, and a review of existing systems and technologies pertinent to our project.

Chapter 3 delves into the description of the Raspberry Pi Pico W and RFID technology, discusses the system architecture and its components, details the development tools and software employed, and provides implementation insights.

Chapter 4 presents the developed smart shopping cart application, outlines the testing procedures and results, and evaluates the system's performance, interpretation of results, and a comparative analysis with existing systems.

Chapter 5 summarizes the key findings of the study, highlight its contributions to the field, and discuss implications for practice and future research endeavors.

CHAPTER 2: LITERATURE REVIEW

2.1 Overview of IoT in Retail

Internet of Things (IoT) technology has revolutionized various industries, including the retail sector, by enabling the seamless integration of physical and digital systems. In the retail industry, IoT facilitates the creation of smart, interconnected environments that enhance customer experiences, optimize operations, and drive business growth.

2.1.1 Applications of IoT in Retail

IoT technology has numerous applications in the retail industry, transforming various aspects of operations and customer interactions. Some prominent applications include:

1) **Inventory Management**: IoT sensors embedded in shelves, storage units, and products enable real-time monitoring of inventory levels, reducing stockouts, minimizing overstocking, and improving supply chain efficiency [25].
2) **Smart Vending Machines**: IoT-enabled vending machines equipped with sensors and digital displays offer personalized product recommendations, accept various payment methods, and provide real-time inventory updates [26].
3) **Customer Engagement**: IoT devices such as beacons and RFID tags enable retailers to deliver personalized promotions, discounts, and recommendations to customers based on their location, preferences, and purchase history [27].
4) **Loss Prevention**: IoT-based surveillance cameras, motion sensors, and RFID tags help retailers prevent theft, reduce shrinkage, and improve security by monitoring store premises and tracking the movement of goods [28].

2.1.2 Challenges and Considerations

Despite the benefits, the adoption of IoT in retail poses several challenges, including data privacy concerns, cybersecurity risks, interoperability issues, and the complexity of integrating legacy systems with IoT technologies. Retailers must carefully assess these challenges and implement robust strategies to address them effectively.

2.2 Introduction to Automated Shopping Carts

In the contemporary landscape of retail, the integration of technology has become increasingly pivotal in shaping the shopping experience. Automated shopping carts represent a significant advancement in this regard, offering a seamless and efficient way for consumers to navigate and purchase items within retail environments. This section summarizes key findings from previous research and then discusses how this project differs from existing approaches.

2.2.1 Review of Existing Studies

Several studies have explored the integration of RFID technology into shopping cart systems to enhance the shopping experience and streamline the billing process. The concept of utilizing RFID technology in shopping carts was introduced in [13]. This study proposed a system where each product is equipped with an RFID tag instead of relying on barcode scanners. The smart trolley featured an RFID reader, LCD display, and Zigbee transmitter, enabling seamless scanning of products and displaying their names and costs. Similarly, [14] presented an idea where every commodity in a mall is affixed with an RFID tag, and each trolley is equipped with an RFID reader connected via Zigbee. Despite its effectiveness in tracking items and transactions, this system lacked

a user-friendly interface.

The study in [15] showcased a cart equipped with an RFID reader, Zigbee transceiver, and LCD display, allowing for the tracking of purchases and displaying the total bill. However, it lacked built-in security checks. On the other hand, [16] proposed a Smart Cart system with RFID and Zigbee technology, offering automatic billing, product recommendations, and anti-theft measures. The research in [17] introduced an IoT-based intelligent trolley for shopping malls, utilizing RFID for billing and an ESP module for communication. Despite its advantages, challenges such as distance constraints and server load were noted. Meanwhile, [18] proposed an automated shopping trolley system utilizing barcode technology for billing, albeit with limitations regarding line-of-sight scanning.

The author of [19] proposed a cash register optimization system utilizing RFID for product scanning and online payments. However, it lacked measures for accidental product additions. [20] introduced RFID-based Automatic Billing Trolley technology as an improvement over manual billing methods. An Intelligent Shopping Cart system comprising server communication, user interface, and automatic billing components was proposed in [21]. Meanwhile, [22] suggested a Smart Shopping Cart utilizing wireless sensor networks and image processing for automated billing. [23] introduced a Smart Trolley with Instant Billing to Ease Queues at shopping malls using RFID and Zigbee modules. Lastly, [24] proposed the Automation of Shopping Cart to Ease Queues in Malls by using RFID technology for price scanning and Zigbee for communication.

2.3 Drawbacks of Existing Studies

The reviewed studies demonstrate the potential of RFID technology in revolutionizing shopping experiences. However, several limitations and challenges need to be addressed to optimize these systems. One major drawback is the lack of a user-friendly interface in many existing solutions, as highlighted in [13] and [15]. This could hinder widespread adoption among consumers who seek seamless and intuitive shopping experiences.

Furthermore, while RFID technology offers significant advantages in terms of tracking items and automating billing processes, it also comes with its set of limitations. For instance, distance constraints and server load issues, as observed in [17], pose significant challenges in implementing IoT-based solutions. Additionally, reliance on barcode technology, as seen in [18], restricts the flexibility and efficiency of the system due to line-of-sight scanning requirements.

Moreover, a common concern across the reviewed studies is the use of somewhat expensive microcontrollers. This could potentially increase the overall cost of implementing shopping cart systems, making them less feasible for widespread adoption, especially in cost-sensitive markets.

2.4 Advancing Shopping Cart Systems with State-of-the-art Technologies

In this project, we are harnessing the capabilities of a cutting-edge microcontroller called the Raspberry Pi Pico W. Released in June 2022, the Raspberry Pi Pico W represents a significant advancement in the field of physical computing. It builds upon the success of its predecessor, the Raspberry Pi Pico board, and introduces wireless connectivity for enhanced flexibility and functionality.

The introduction of Raspberry Pi Pico W marks a significant leap forward in the realm of microcontrollers, offering several advantages over those utilized in previous studies. Unlike traditional microcontrollers, Raspberry Pi Pico W comes equipped with integrated Wi-Fi and Bluetooth capabilities, enabling seamless wireless communication. This feature opens up possibilities for remote monitoring, control, and data transfer, enhancing the overall functionality of shopping cart systems. Despite its advanced features, Raspberry Pi Pico W remains cost-effective, offering a compelling alternative to expensive microcontrollers used in previous studies. This affordability opens doors for wider adoption and scalability of smart shopping solutions.

Incorporating Raspberry Pi Pico W into shopping cart systems presents a promising opportunity to leverage its advanced capabilities and drive significant improvements in functionality, performance, and cost-effectiveness.

CHAPTER 3: METHODOLOGY

3.1 Functional Description of Hardware Components

The two main hardware components utilized in this project are the Raspberry Pi Pico W microcontroller and the RFID module RC522. These components play pivotal roles in enabling the functionality of the smart shopping cart system.

3.1.1 Raspberry Pi Pico W

The Raspberry Pi Pico W is a wireless microcontroller board released in June 2022, designed specifically for physical computing applications. It serves as the central processing unit for the smart shopping cart system, orchestrating the interaction between various hardware and software components. Figure 3.1 illustrates the pin diagram of the Raspberry Pi Pico W, showcasing the layout and configuration of its GPIO (General Purpose Input/Output) pins.

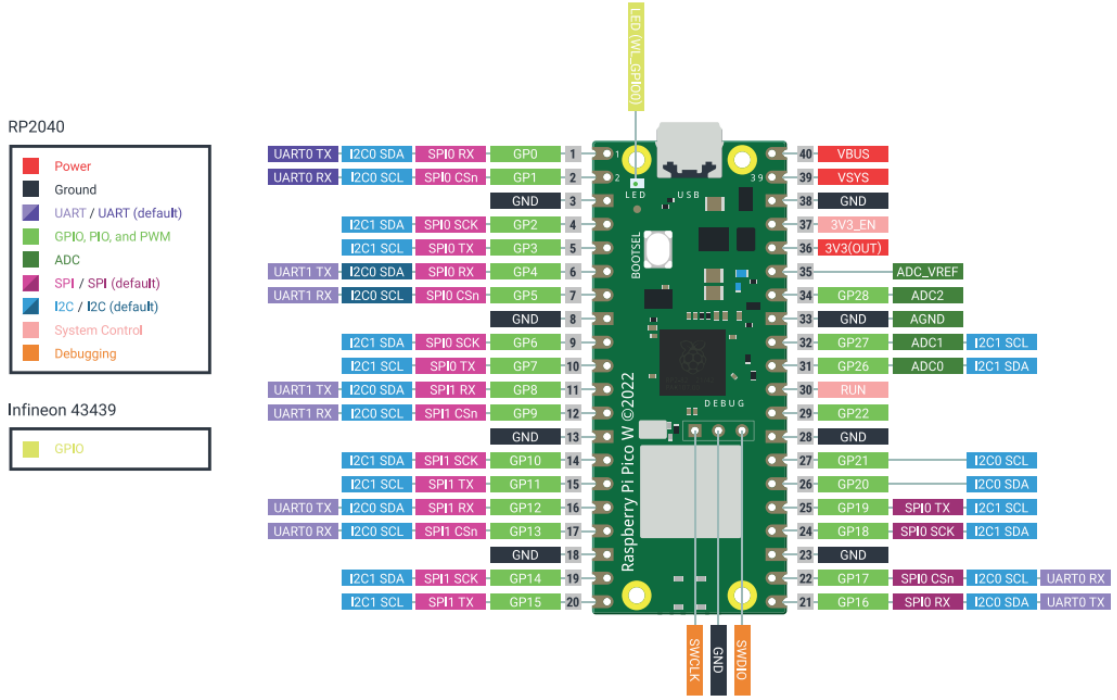


Figure 3.1: Pin Diagram of Raspberry Pi Pico W

Key features of the Raspberry Pi Pico W include:

1) **Wireless Connectivity**: The Raspberry Pi Pico W features built-in 802.11n Wi-Fi capability, allowing seamless wireless communication.
2) **Powerful Microcontroller**: Powered by the RP2040 microcontroller with 2MB of flash memory, the Pico W delivers high performance and versatility.
3) **Bluetooth Support**: Along with Wi-Fi, the Pico W also supports Bluetooth 5.2, enabling a wide range of wireless applications.
4) **Compact Design**: With a compact 40-pin DIP style PCB, the Pico W offers a minimal footprint suitable for various projects.
5) **Flexible I/O Pins**: The Pico W provides 26 multi-function 3.3V general-purpose I/O pins, with support for digital and analog functions.
6) **Comprehensive SDK**: Backed by a comprehensive software development kit (SDK), the Pico W simplifies the development process with software examples and documentation.

3.1.1.1 Advantages of Raspberry Pi Pico W

The introduction of Raspberry Pi Pico W marks a significant leap forward in the realm of microcontrollers, offering several advantages over those utilized in previous studies.

1) **Low Cost**: Despite its advanced features, the Raspberry Pi Pico W remains cost-effective, making it accessible to a wide range of users.
2) **High Availability**: With high-quality components and widespread availability, the Pico W ensures reliability and ease of procurement.
3) **Flexible Power Options**: The Pico W offers various options for powering the unit, including micro-USB, external supplies, or batteries, enhancing its versatility.
4) **Increased Processing Power**: With the powerful RP2040 microcontroller at its core,

Raspberry Pi Pico W boasts improved processing capabilities, allowing for faster data processing and more complex computations.

5) **Comprehensive Documentation**: Raspberry Pi provides extensive documentation and support for the Pico W, facilitating seamless integration into projects.

3.1.2 RFID Module RC522

The RC522 RFID module serves as the primary means of product identification and tracking within the smart shopping cart system. It enables the detection and communication with RFID tags attached to individual products. Figure 3.2 depicts the pin diagram of the RFID RC522 module, highlighting the arrangement and functionality of its various pins.

Figure 3.2: Pin diagram of RFID-RC522 Module [29]

Key features of the RC522 module include:

1) **RFID Tag Compatibility**: The RC522 module is compatible with various RFID tags, allowing for versatile integration with different types of products.
2) **Read Range and Accuracy**: With its robust radio frequency identification capabilities, the RC522 module can accurately read RFID tags at close range, ensuring reliable detection of products within the shopping cart.
3) **Communication Interface**: Utilizing SPI (Serial Peripheral Interface) communication protocol, the RC522 module interfaces seamlessly with the Raspberry Pi Pico W

microcontroller, enabling bidirectional data exchange.

4) **Integration Flexibility**: The compact size and standardized interface of the RC522 module make it easy to integrate into the smart shopping cart system, facilitating rapid development and deployment.

3.2 Software Stack of the Shopping Cart System

Designing an IoT-based smart shopping cart involves not only hardware components but also sophisticated software components to manage data, communication, and user interfaces. This section details the software technologies used to create both the front end and the back end of the smart shopping cart system, including how the web application and database are managed using XAMPP web server, and the specific programming languages and libraries utilized.

enhancing the user experience.

3.2.1 Front End Technologies

The front end of the smart shopping cart application is developed using HTML, CSS, Bootstrap, and JavaScript to provide a user-friendly interface with dynamic functionalities. Figure 3.3 shows how the web application is built using the frontend technologies.

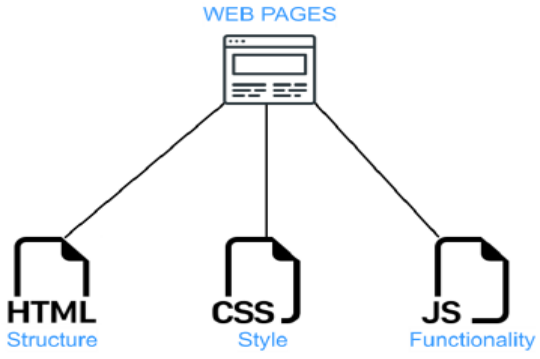


Figure 3.3: Web Application Development Components

1) **HTML (Hypertext Markup Language):**

- HTML forms the backbone of the web pages, providing the basic structure and content.
- It is used to create forms, tables, and other essential elements of the shopping cart interface.

2) **CSS (Cascading Style Sheets):**

- CSS is used to style the HTML elements, making the web application visually appealing.
- It ensures a consistent look and feel across different parts of the application.

3) **Bootstrap:**

- Bootstrap is a popular front-end framework that facilitates responsive design and pre-styled components.
- It simplifies the development of complex layouts and enhances the user interface with minimal custom CSS.

4) **JavaScript:**

- JavaScript is employed for client-side scripting to add interactivity and dynamic behavior to the web pages.
- It handles the functionality of buttons, form validation, and asynchronous data fetching.

3.2.2 Back End Technologies

The back end of the application handles the server-side logic, database interactions, and overall data management. The key technologies used are PHP and MySQL, hosted on the XAMPP server.

1) **PHP (Hypertext Preprocessor):**

- PHP is a server-side scripting language that is used to create dynamic web pages and handle server-side logic.
- It is embedded within HTML and executed on the server to generate dynamic content.
- In the smart shopping cart application, PHP scripts manage user authentication, product search, transaction processing, and database interactions.

2) **MySQL Database:**

- MySQL is a popular relational database management system used to store structured data.
- It is utilized to store product information, user data, transaction records, and inventory details.
- Tables in MySQL database include fields such as product ID, name, price, quantity, user information, and transaction timestamps.

3) **XAMPP Server:**

- XAMPP is a cross-platform web server solution stack that includes Apache, MySQL, PHP, and Perl.
- It provides an environment for hosting dynamic web applications locally.
- Apache serves web pages, PHP scripts interact with the MySQL database, and XAMPP manages the overall hosting environment.

3.2.3 Development Tools and Microcontroller Programming

The development of the smart shopping cart system involves utilizing specific tools and programming environments to code and control the microcontroller effectively.

1) **Thonny IDE:**

- Thonny IDE (Integrated Development Environment) is used for programming the microcontroller, specifically the Raspberry Pi Pico W.
- The code, written in MicroPython, is uploaded to the microcontroller via USB.
- The IDE facilitates the use of the RFID-RC522 library for RFID functionality and the SH1106 library for the OLED screen.

2) **RFID-RC522 Library:**

- This library is used to enable the microcontroller to communicate with RFID tags.
- It helps in reading data from RFID tags and integrates with the Raspberry Pi Pico W.

3) SH1106 Library:

- This library is used to manage the OLED screen connected to the Raspberry Pi Pico W.
- It allows displaying text and graphics, providing real-time feedback to the users about the system status.

3.3 System Design

The system adopts a four-layered architecture to facilitate data collection, efficient communication, data processing, and user interaction. Figure 3.4 illustrates the layered architecture of our IoT-based smart shopping cart system, showcasing the hierarchical arrangement of the physical, network, data processing, and application layers. This layered approach enables modular development, scalability, and flexibility in the design and implementation of our system, ensuring robustness and efficiency in handling various tasks and functionalities.

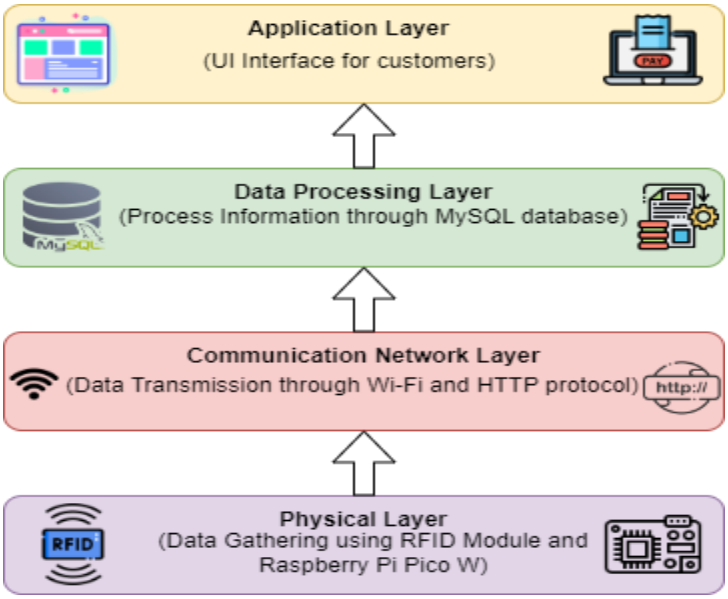


Figure 3.4: Layered Design of the IoT-Based Shopping Cart System

The first layer, known as the physical layer, encompasses the hardware components that form the foundation of the system. This includes devices such as the Raspberry Pi Pico W microcontroller,

RFID RC522 module, LED Screen, sensors, and any other physical components essential for the functionality of the system. These components interact directly with the physical environment, capturing data and transmitting signals. In this project, the Raspberry Pi Pico W is connected to the RFID RC522 module, which collects data when a product is scanned. This layer represents the physical interaction between hardware components, enabling the capture of product information during shopping activities.

Above the physical layer lies the network layer, which manages the communication between devices within the system. This layer is responsible for establishing connections, routing data packets, and ensuring reliable transmission of information. In our smart shopping cart system, the Wi-Fi capability of the Raspberry Pi Pico W is utilized to transmit data. This layer facilitates the transfer of information over the internet via the HTTP protocol, enabling seamless communication between devices and backend servers.

Sitting atop the network layer is the data processing layer, where data collected from the physical environment is processed, analyzed, and transformed into meaningful insights. This layer encompasses both edge computing capabilities embedded within the devices themselves, as well as centralized processing performed on backend servers or cloud platforms. In our project, this is the web server layer, which is implemented using a MySQL database powered by XAMPP. Within the data processing layer, the MySQL database stores critical information related to product details, pricing, user accounts, transactions, and inventory status. XAMPP serves as the middleware that facilitates communication between the frontend and backend components, ensuring seamless data exchange and robust system functionality.

Lastly, the fourth layer is the application layer, which serves as the interface between the system and the end-user. This layer encompasses the frontend user interface (UI), through which users interact with the smart shopping cart application. Here, users can view item details, add, or remove products from their cart, view their shopping list, and complete the checkout process. Additionally, the application layer may also include backend services responsible for managing user accounts,

processing payments, generating receipts, and providing personalized recommendations based on user preferences. In this project, the front-end interface is developed using technologies such as HTML, CSS, Bootstrap, and JavaScript to create a user-friendly and visually appealing experience. Meanwhile, the backend services are implemented using PHP to handle user authentication, data processing, and transaction management. Overall, the system is designed to provide a seamless and intuitive shopping experience for users, with a focus on ease of use and efficiency.

3.4 System Architecture

The smart shopping cart system operates seamlessly, integrating various hardware and software components to enhance the shopping experience for customers. At its core, the architecture revolves around efficient communication and processing of data, ensuring real-time updates and secure transactions.

3.4.1 High-Level Architecture

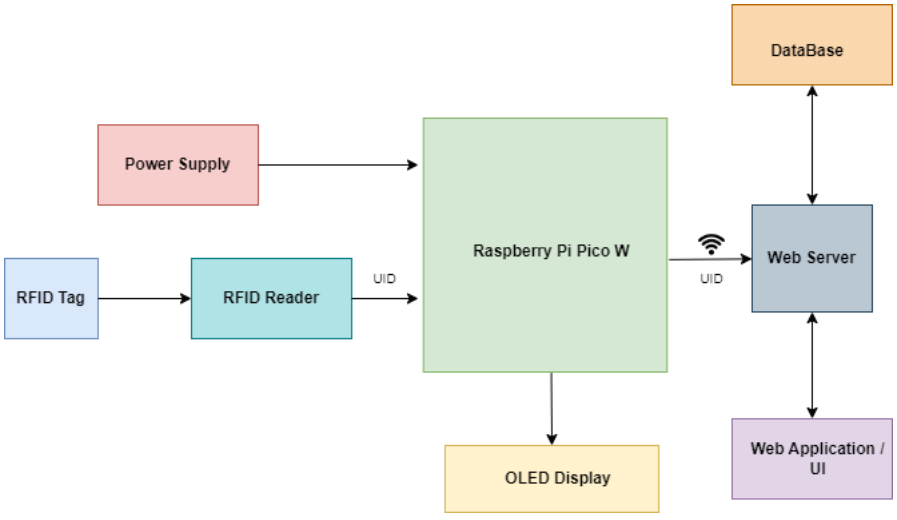


Figure 3.5: Block Diagram of the Proposed Smart Shopping Cart System

The system's architecture is depicted in a block diagram (Figure 3.5), illustrating the interconnectivity between the key components. At the heart of the system lies the Raspberry Pi Pico W microcontroller, equipped with an RFID scanner, a Wi-Fi module, and interfacing capabilities.

This microcontroller serves as the central hub, facilitating communication between the RFID scanner, the web server, the product database, and the user interface (UI). Upon scanning a product, the RFID scanner captures its unique identifier (UID) and transmits it to the Raspberry Pi Pico W. Leveraging its Wi-Fi module, the Raspberry Pi Pico W establishes a connection with the web server, initiating a request to retrieve the corresponding product details from the database. The web server, powered by a backend application (XAMPP), processes the request, queries the product database, and retrieves the relevant information. Once obtained, the product details are updated in the UI in real-time, displaying the scanned items along with their names, descriptions, and prices.

The OLED display screen integrated into the smart shopping cart serves as a vital component, providing visual feedback to users. It indicates the status of the Wi-Fi connection, allowing customers to ascertain whether the device is connected to the network. Additionally, the OLED screen displays prompts and instructions, guiding users on when to start scanning products, further enhancing the user experience.

3.4.2 Architecture Workflow

Figure 3.6 shows the workflow flowchart, detailing the proposed smart shopping cart system's operations and component interactions to achieve the desired functionality.

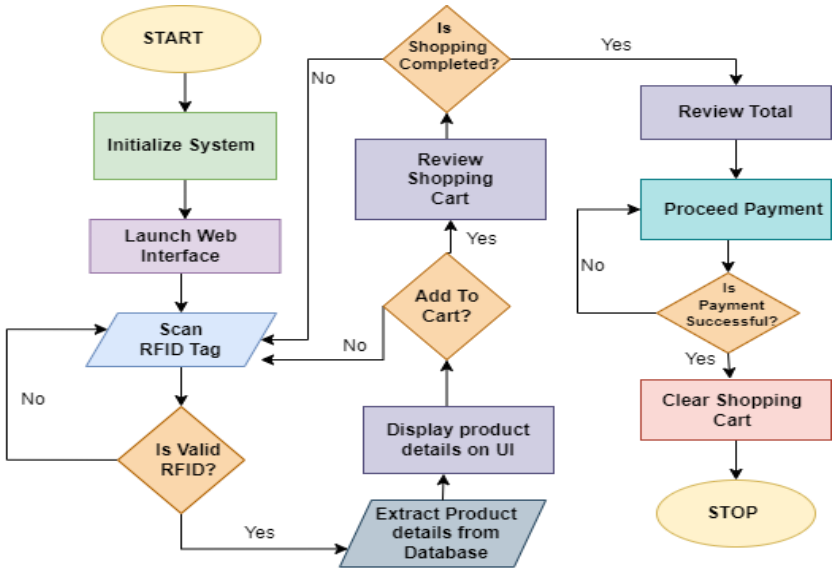


Figure 3.6: Workflow Operation of the Proposed Smart Shopping Cart System

The workflow of the smart shopping cart system begins with the initialization phase, where all the necessary hardware and software components are set up for operation. Upon powering up, the Raspberry Pi Pico W establishes a Wi-Fi connection, preparing the system to interact with the web server. The initialization process also involves launching the web interface, which serves as the user interaction platform for managing the shopping cart and processing transactions.

Customers start the shopping process by scanning the RFID tags of products they wish to purchase. Each RFID tag contains a unique identifier (UID) for the respective product. The RFID scanner reads this UID and transmits it to the Raspberry Pi Pico W. The system then verifies the validity of the scanned RFID tag. If the tag is invalid, an error message is displayed, prompting the customer to re-scan the product. Once a valid RFID tag is detected, the UID is sent to the web server via the Wi-Fi module in the Raspberry Pi Pico W, using the HTTP protocol for communication. The web server processes the request by querying the product database (typically a MySQL database) to retrieve comprehensive details about the product, including its name, description, and price. This information is then sent to the web application (UI).

The Raspberry Pi Pico W updates the OLED screen in real time when the product is scanned. The OLED screen also serves as an essential component, providing feedback on the system's status, such as the Wi-Fi connection and prompts for starting the scanning process.

Throughout the shopping process, customers can continue adding items to their cart by scanning additional RFID tags. The UI allows for dynamic updates, showing real-time changes as new products are added. Customers can also review their shopping cart at any time, checking the total cost and the list of items.

When the customer decides to complete their shopping, they proceed to the payment stage. The backend application handles this phase, providing a secure environment for processing payments. Once the payment information is entered, the backend system verifies and processes the transaction. Upon successful payment, the system clears the shopping cart, preparing it for the next user.

This workflow ensures that the entire shopping experience is smooth, efficient, and user-friendly. The integration of RFID technology, real-time data processing, and a responsive UI provides customers with a seamless shopping experience.

3.5 Implementation

The implementation of the smart shopping cart system involves detailed hardware integration, specifically the connections between the Raspberry Pi Pico W microcontroller, the RC522 RFID module, and the OLED display. Figure 3.7 illustrates the schematic circuit for these connections.

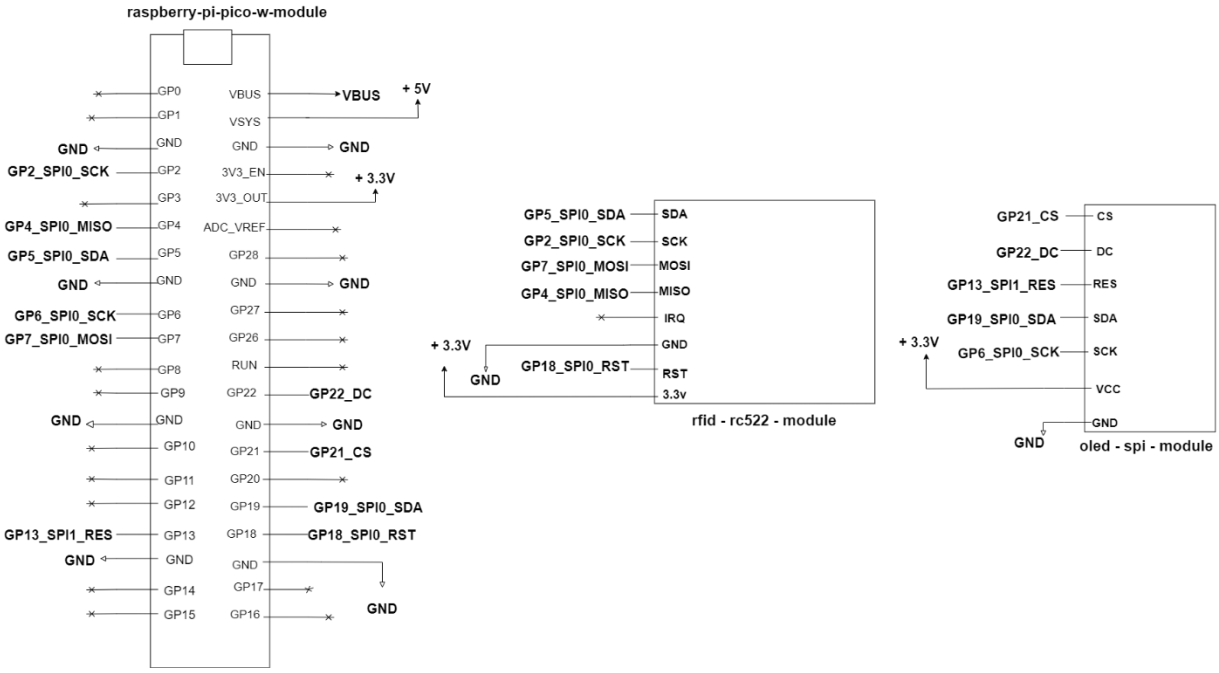


Figure 3.7: Schematic circuit of the hardware connections for the smart shopping cart system

3.5.1 Hardware Integration of Microcontroller and RFID Module (RC522)

The RC522 RFID module communicates with the Raspberry Pi Pico W via SPI (Serial Peripheral Interface) communication. SPI is chosen for its high-speed data transfer capabilities [30], essential for efficient RFID tag reading. The connections are as follows:

- **SCK (Serial Clock):** Connects to the SCK pin on the Raspberry Pi Pico W, ensuring synchronized data transfer between the two devices.
- **SDA (Slave Select/Chip Select):** Connects to a GPIO pin on the Raspberry Pi Pico W, enabling the microcontroller to select the RFID module for communication.
- **MOSI (Master Out Slave In):** Connects to the MOSI pin on the Raspberry Pi Pico W, facilitating data transfer from the microcontroller to the RFID module.
- **MISO (Master In Slave Out):** Connects to the MISO pin on the Raspberry Pi Pico W, carrying data from the RFID module to the microcontroller.
- **GND (Ground):** Connects to the ground pin on the Raspberry Pi Pico W.
- **VCC (Power Supply):** Connects to the 3.3V power pin on the Raspberry Pi Pico W.
- **RST (Reset):** Connects to a GPIO pin on the Raspberry Pi Pico W, enabling the microcontroller to reset the RFID module when necessary.

3.5.2 Hardware Integration of Microcontroller and OLED Display

The OLED display also uses SPI communication to interact with the Raspberry Pi Pico W. This ensures a consistent and reliable connection for displaying information. The connections are as follows:

- **SDA (Data Line):** Connects to the SDA pin on the Raspberry Pi Pico W, facilitating data transfer between the microcontroller and the OLED display.
- **SCK (Clock Line):** Connects to the SCK pin on the Raspberry Pi Pico W.
- **DC (Data/Command):** Connects to another GPIO pin on the Raspberry Pi Pico W, allowing the microcontroller to distinguish between sending commands and data to the OLED.
- **RES (Reset):** Connects to a GPIO pin on the Raspberry Pi Pico W, enabling the microcontroller to reset the display when necessary.
- **CS (Chip Select):** Connects to a GPIO pin on the Raspberry Pi Pico W, allowing the microcontroller to select the OLED display for communication.

- **GND (Ground):** Connects to the ground pin on the Raspberry Pi Pico W.
- **VCC (Power Supply):** Connects to the 3.3V power pin on the Raspberry Pi Pico W.

3.5.3 System Operation

Upon system initialization, the Raspberry Pi Pico W establishes connections with the RC522 RFID module and the OLED display. The OLED screen assists the user by indicating the status of the WiFi connection and signaling when the system is ready for scanning products.

When a customer places an item in the shopping cart, the RFID scanner reads the unique identifier of the RFID tag attached to the product. This data is sent to the Raspberry Pi Pico W, which then transmits it to a web server via the built-in WiFi module using the HTTP protocol. The web server processes the received data by querying a MySQL database to retrieve product details such as name and price. These details are then sent to the web application, which displays them on the user interface (UI) in real time, providing immediate feedback to the customer.

This setup not only ensures efficient product scanning and data retrieval but also enhances user experience by providing a clear and interactive interface on the web application.

3.5.4 SPI Communication

In the smart shopping cart system, the Raspberry Pi Pico W acts as the master device, controlling the communication with the RC522 RFID module. The SCK signal provides a clock source for synchronizing data transmission between the master and slave devices. It ensures that data is sampled at the correct times during communication. The microcontroller generates the clock signal (SCK) and sends data to the RFID module using the MOSI signal. Simultaneously, it receives data from the RFID module via the MISO signal. By coordinating the timing of these signals, SPI (Serial Peripheral Interface) communication ensures accurate and reliable data transmission between the microcontroller and the RFID module [30].

SPI communication offers several advantages for the smart shopping cart system, including high-speed data transfer, full-duplex communication, and efficient utilization of hardware resources. By leveraging SPI, the system can rapidly read RFID tags, retrieve product information, and provide real-time feedback to users, enhancing the overall shopping experience.

3.6 Powering the Raspberry Pi Pico W

To ensure the Raspberry Pi Pico W is properly powered for the smart shopping cart system, various methods of providing power to the microcontroller must be considered. This involves understanding the pin definitions and implementing safe connections with external power sources, such as batteries and voltage regulators.

3.6.1 Pin Definitions

The Raspberry Pi Pico W features several pins related to power management:

VBUS (PIN 40): This pin is connected to the micro-USB port, allowing the Pico W to be powered through it. It accepts a voltage range of 4.5V to 5.5V.

VSYS (PIN 39): This pin serves as the main system input voltage, which can vary between 1.8V and 5.5V. The onboard SMPS uses this voltage to generate 3.3V to power the RP2040 microcontroller and its GPIOs.

3V3_EN (PIN 37): This pin is used to enable the onboard SMPS for 3.3V. It is pulled high to the VSYS pin using a 100kΩ resistor.

3V3(OUT) (PIN 36): This pin outputs a regulated 3.3V voltage that can power external components such as sensors. The recommended maximum load current from this pin is under 300mA.

GND: These pins provide the reference ground for the Pico W and connected devices. There are 8 GND pins available on the edge pinouts of the Pico W, all serving the same purpose.

RUN (PIN 30): This is the enable pin for the RP2040 microcontroller, pulled up to 3.3V via onboard circuitry. It can be used to reset the Raspberry Pi Pico W.

3.6.2 Components Used

Several key components are used to power the Raspberry Pi Pico W reliably and safely. These include a voltage regulator and a diode.

1. **Voltage Regulator:** The LM7805 is a linear voltage regulator that outputs a steady 5V, suitable for powering the Raspberry Pi Pico W via the VSYS pin. The regulator accepts input voltages ranging from 7V to 35V and is capable of delivering up to 1A of current. This regulator ensures that the input voltage from the power source is converted to a stable 5V output.

2. **Adding a Diode for Safety:** A diode is added between the secondary power source and the VSYS pin to safely connect a battery or secondary power source to the Pico W. This prevents one power source from back-feeding the other. A Schottky diode is selected because it has a lower forward voltage drop than other diodes. However, a commonly available rectifier diode such as the 1N4007 diode will also work, provided the ~0.6V forward voltage drop in such a diode is accounted for.

3. **9V Battery:** The choice of a 9V battery for this project is based on several key factors. Firstly, 9V batteries are widely available and easy to replace, making them highly convenient for prototyping and development purposes. Additionally, the 9V battery provides a voltage range that comfortably fits within the input requirements of the LM7805 voltage regulator, ensuring reliable operation and sufficient headroom for voltage regulation. Portability is another significant advantage; the compact size of a 9V battery is crucial for mobile applications such as a smart shopping cart. Moreover, 9V batteries are known for their stable voltage output, which is essential for maintaining the consistent performance of the microcontroller and all connected components throughout the duration of use.

3.6.3 Power Circuit Layout

Figure 3.8 illustrates a schematic circuit diagram for the power setup of the Raspberry Pi Pico W, which utilizes a 9V battery and an LM7805 voltage regulator.

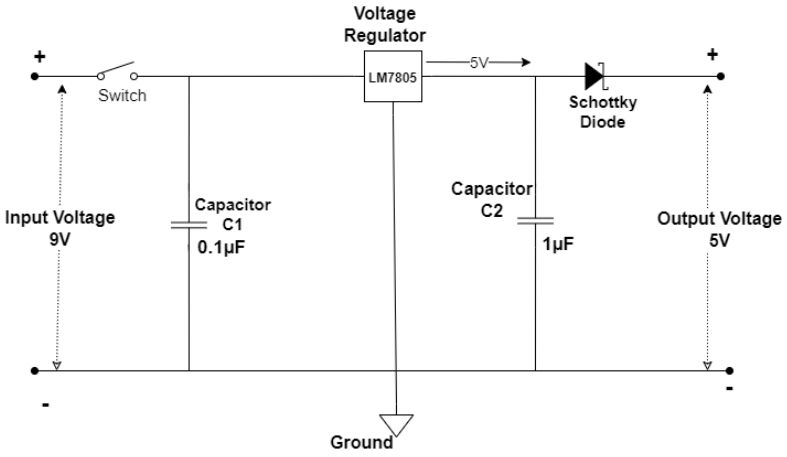


Figure 3.8: Power Supply Circuit for Raspberry Pi Pico W

This circuit layout ensures a stable and portable power supply, essential for the consistent performance of the smart shopping cart system. The inclusion of a switch allows for convenient power control, while the Schottky diode prevents back-feeding, ensuring the safety and longevity of the components involved. The total weight of the power circuit is approximately 50 grams, making it lightweight and convenient for mobile applications. The image displayed shows a circuit setup for powering the Raspberry Pi Pico W using a 9V battery and an LM7805 voltage regulator. This configuration ensures a stable and reliable power supply to the microcontroller, crucial for its operation in applications such as the smart shopping cart system. The 9V battery serves as the primary power source, providing a sufficient voltage that the LM7805 regulates down to a steady 5V output. Capacitors (C1 and C2) are strategically placed to stabilize the input and output voltages of the LM7805, minimizing noise and ensuring smooth operation. Additionally, a Schottky diode (D1) is incorporated to prevent back-feeding of power from the Pico W to the battery, enhancing safety and protecting the components from potential damage. This setup is ideal for mobile and embedded systems where compactness, portability, and consistent performance are essential requirements.

3.7 Scalability of the proposed system

In this project, an IoT-based shopping cart was developed using the Raspberry Pi Pico W, an RFID RC522 module, and an OLED display for hardware. For the software component, an XAMPP-hosted server was employed with a MySQL database managed through phpMyAdmin. The database includes multiple tables: one dedicated to the total list of products accessible exclusively by an administrator, and individual tables for each shopping cart.

Given the application in retail stores, where multiple shopping carts are likely to be in use by various customers simultaneously, it is essential for the proposed system to be easily scalable. Scalability was achieved by implementing additional hardware setups, each interacting with the same database but using a separate table for its respective shopping cart. This approach ensures that each cart's data remains isolated, avoiding conflicts with other carts.

To scale the software, new directories within the web server's root were created for each additional hardware setup. This configuration allows for a unique user interface for each new setup. Consequently, each additional hardware unit maintains its own shopping cart table within the database, preventing any interference with the tables of other carts.

The scalability of the system was tested and found to be effective. This approach demonstrated that the solution is not only scalable but also easy to implement with minimal effort. By isolating each shopping cart’s data in separate tables and managing multiple hardware setups through distinct directories, it was ensured that the system could handle an increase in the number of shopping carts without compromising performance or data integrity.

This method of scaling highlights the robustness and practicality of the proposed IoT-based shopping cart system, making it a viable option for large-scale retail deployment.

CHAPTER 4: RESULTS AND DISCUSSIONS

The developed smart shopping cart application integrates hardware components such as the Raspberry Pi Pico W microcontroller, RC522 RFID module, and an OLED display, with software components including web-based user interfaces and backend server-side scripts. The application facilitates real-time tracking of products placed in the shopping cart, retrieval of product details from a MySQL database, and seamless payment processing.

4.1 Testing Procedures and Results

The following test plans were followed to ensure that our smart cart works as expected.

4.1.1 Hardware Testing

The hardware components were thoroughly tested to ensure proper functionality and interaction. This involved testing the RFID scanner's ability to detect RFID tags, the microcontroller's communication with the RFID module and OLED display, and the overall responsiveness of the system.

Figure 4.1 shows the practical hardware setup test results for the proposed system.

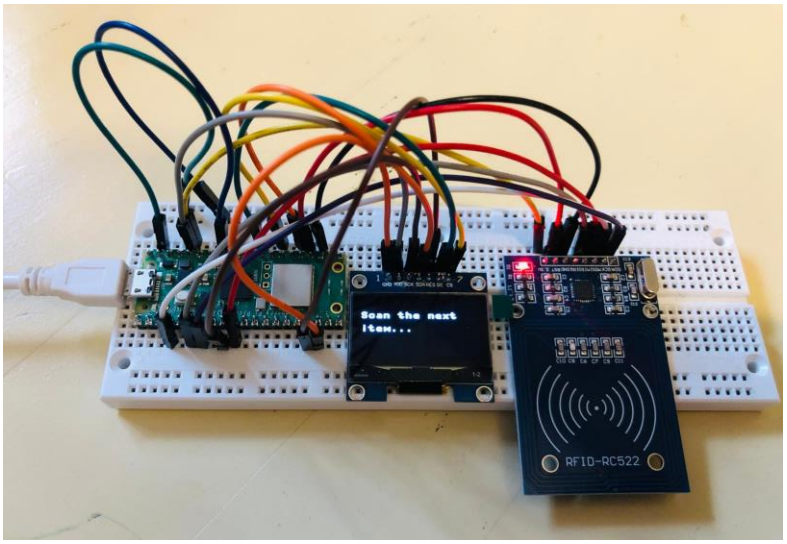


Figure 4.1: Hardware Setup Test Results for the Proposed System

4.1.2 Software Testing

Software testing plays a crucial role in ensuring the reliability, functionality, and performance of the developed smart shopping cart application. This process involves systematically verifying and validating the software components to identify any defects or errors and ensure that the application meets the specified requirements. In the context of the smart shopping cart application, software testing encompasses various aspects, including:

1) **Functional Testing:** This type of testing evaluates whether the software functions according to the specified requirements. For the smart shopping cart, functional testing ensures that features such as RFID scanning, product information retrieval, user interface interactions, and payment processing work as expected. Test scenarios are designed to cover different use cases, such as adding products to the cart, updating the cart contents, and completing a transaction. Figure 4.2 shows the test result of functional tests of the web application.

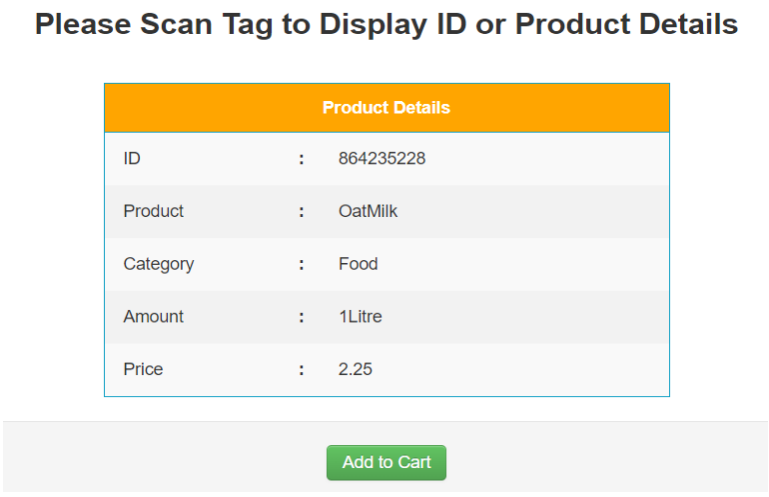


Figure 4.2: Functional Test of Product Information Retrieval

2) **Usability Testing**: Usability testing focuses on evaluating the user interface and overall user experience of the application. Testers assess factors such as navigation, layout, responsiveness, and intuitiveness of the web-based interface. Usability testing helps identify any usability issues or design flaws that may affect user satisfaction and adoption of the system. Figure 4.3 shows the test results of the usability test of the web application.

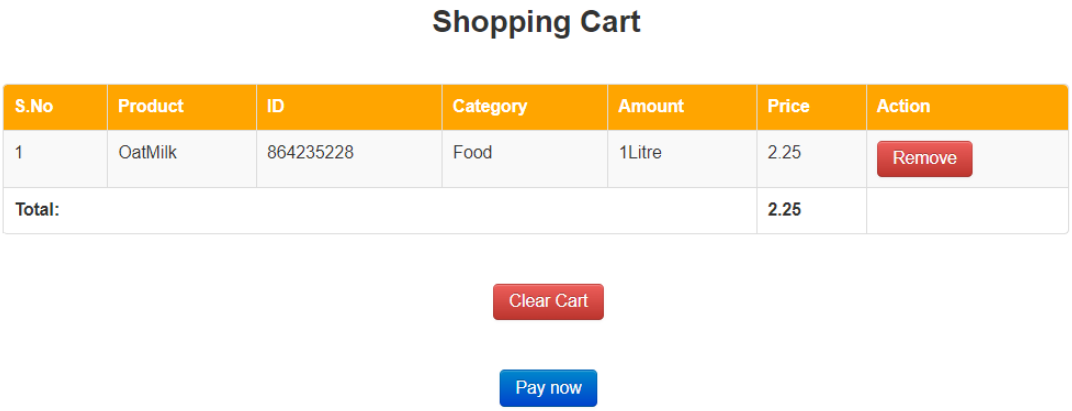


Figure 4.3: Usability test of the web application

3) **Compatibility Testing:** Compatibility testing ensures that the software functions correctly across different devices, operating systems, and web browsers. Since the smart shopping cart application relies on web-based interfaces, compatibility testing ensures that the application displays properly and functions consistently across various platforms and devices used by customers and store staff. For the proposed shopping cart system, compatibility tests were conducted by accessing the web application through mobile web browsers and alternative desktop web browsers. Figure 4.4 shows the test result of compatibility tests of the web application when tested using a mobile web browser (safari).

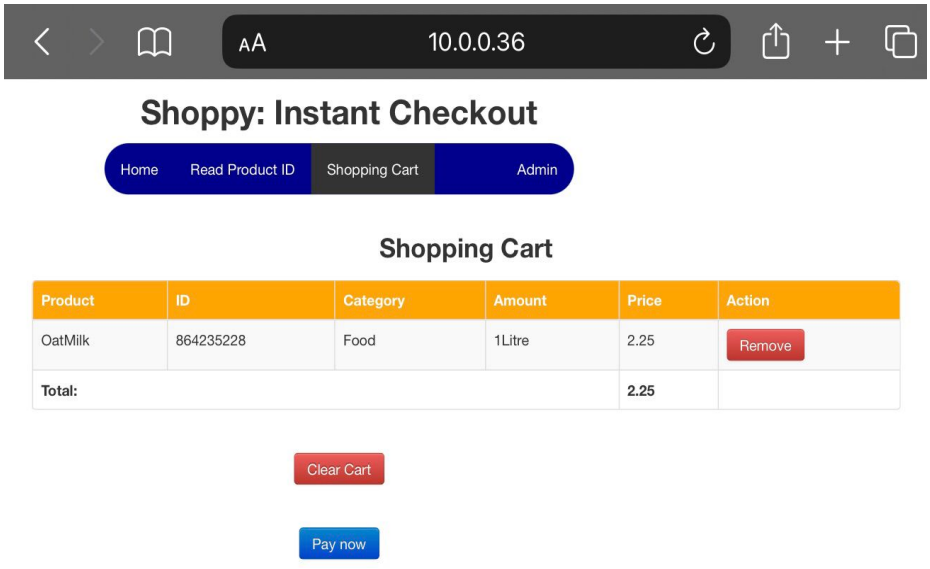


Figure 4.4: Compatibility Test of the Web Application

4) **Performance Testing:** Performance testing assesses the responsiveness, scalability, and stability of the application under different conditions. This includes measuring factors such as response times for RFID scanning, database queries, and transaction processing, as well as assessing the application's ability to handle multiple concurrent users and peak usage periods. Figure 4.5 shows the test results of the performance tests of the web application.

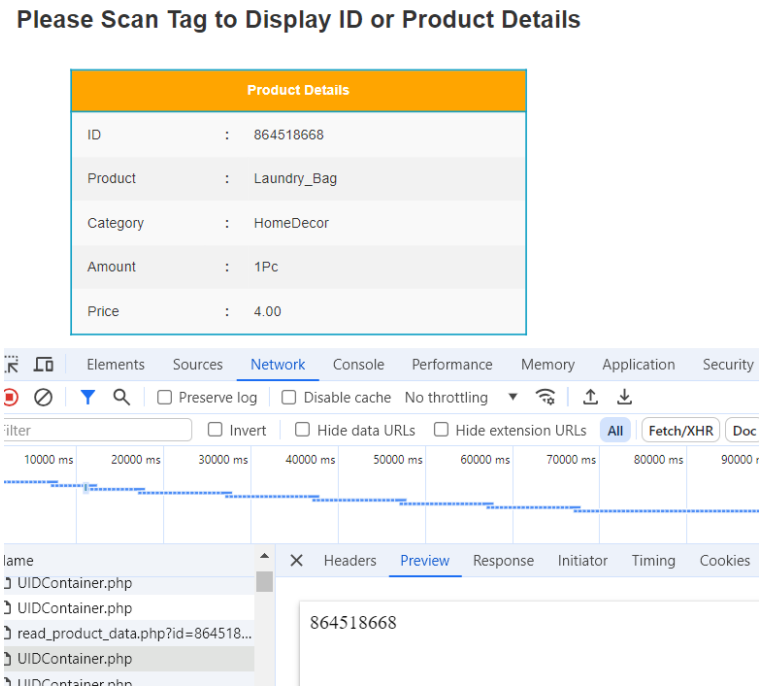


Figure 4.5: Performance Test of the Web Application

Overall, comprehensive software testing ensures that the developed smart shopping cart application meets quality standards, performs reliably in production environments, and delivers an optimal user experience for customers and store staff alike.

4.1.3 Integration Testing

Integration testing assessed the seamless integration of hardware and software components, including the synchronization of data between the microcontroller, web server, and database. This involved testing the flow of data from RFID scanning to product display on the user interface and payment processing. Figure 4.6 shows the test result of the integration testing done on the proposed smart shopping cart system.

Figure 4.6: Integration testing using Thonny IDE

4.1.4 Stress Testing

Stress testing was performed on the MySQL database hosted on a test machine (Lenovo IdeaPad 3 15ITL6, Intel Core i3-1115G4, 8 GB RAM, 256 GB SSD, Windows 11 Home). Apache JMeter was used to simulate various loads, and the database's performance was evaluated under different levels of concurrent users (50, 100, 200, 500, and 1000) and high data volumes (9.5 MiB, 112K rows of data). Performance was good with up to 152 users, with low latency (4 - 40 milliseconds) and successful query execution. Beyond 155 users, latency exceeded 100 milliseconds, and query failures increased due to resource contention. It was noted that latency was good (under 20 milliseconds) for the first 38 users, but over 150 users caused errors due to too many connections.

Efficient handling of a moderate number of tables (up to 152) was achieved, but performance issues were noted with a large number of tables (over 155). Hence, optimization of queries, indexing, and resource management will be necessary for future scalability. Upgrading hardware or optimizing the database schema (the structure of tables, fields, and indexes) may also be required to address significant performance challenges. Continuous monitoring and optimization are crucial to managing increased loads and maintaining performance.

Regarding the limits, the InnoDB storage engine (default storage engine of MySQL) supports up to 1,017 columns per table and has a row size limit of slightly less than 8,000 bytes for 16KB pages and about 16,000 bytes for 64KB pages. The overall row size limit imposed by MySQL is 65,535 bytes, excluding BLOB and TEXT columns, which are stored separately. The maximum table size in MySQL can reach up to 64TB with appropriate configuration, such as using InnoDB, enabling innodb_file_per_table, configuring innodb_data_file_path, using table partitioning, and ensuring OS/file system support for large files. Figure 4.7 shows the test result of the stress testing done on the proposed smart shopping cart system.

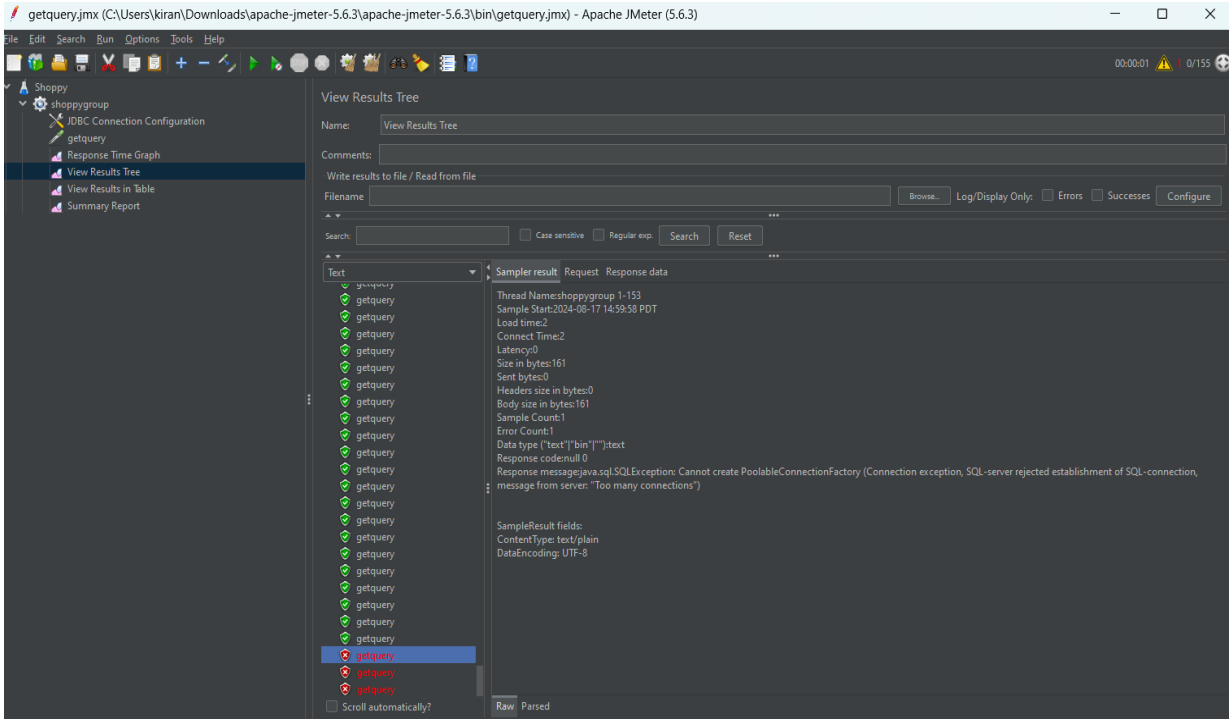


Figure 4.7: Stress testing using JMeter

4.1.5 Fault Injection Testing

Fault injection testing was conducted to evaluate the MySQL database's resilience and recovery capabilities by intentionally introducing various fault scenarios, such as simulated network latency, server crashes, disk errors, resource contention, and unexpected MySQL shutdowns, to identify potential weaknesses and improve system robustness. The MySQL shutdown was caused by issues

like blocked ports, missing dependencies, improper privileges, or crashes, leading to an unexpected termination of the MySQL service. The results showed that the database handled brief network interruptions well but faced transaction retries and timeouts during prolonged disruptions. During server crashes or MySQL shutdowns, the system became disconnected from the database, preventing the retrieval of product details when a product was scanned. Recovery from server crashes was successful, although recovery time varied with transaction volume. Disk errors were managed effectively, maintaining data integrity, but high CPU and memory usage led to increased query latency and slower response times. Overall, the database remained stable.

To improve the system, measures such as enhancing network resilience with redundancy and failover mechanisms, optimizing recovery procedures, reinforcing data redundancy, improving resource allocation and monitoring, and advancing automated error detection can be implemented. Fault injection testing offers valuable insights into the database's performance under adverse conditions, helping to identify areas for improvement and contributing to a more resilient system. Figure 4.8 shows a screenshot of the MySQL server's reaction to the introduced faults.

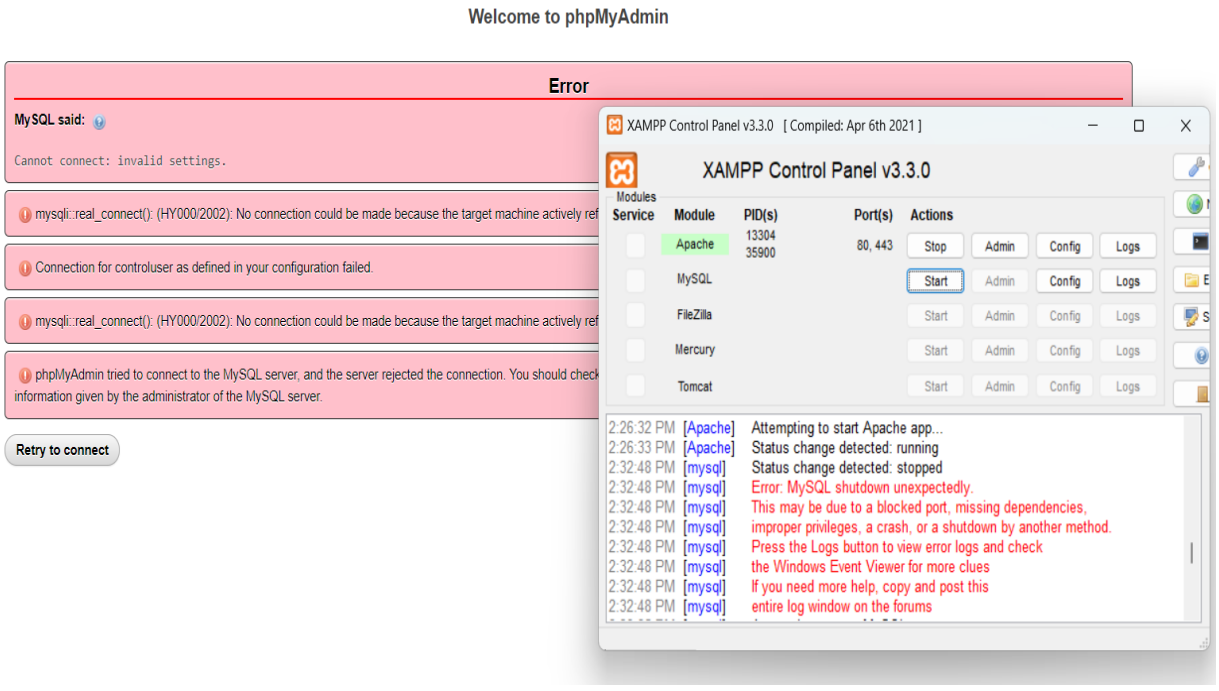


Figure 4.8: Fault Injection on MySQL server

4.2 Interpretation of the Results

The results of the proposed project demonstrate the successful implementation of a smart shopping cart system that effectively tracks products, retrieves real-time product information, and facilitates seamless transactions. Utilizing RFID technology with the Raspberry Pi Pico W microcontroller, our system accurately captures product identifiers and transmit this data in real-time to a MySQL database via a Wi-Fi module and HTTP protocol. The web-based user interface, developed using HTML, CSS, Bootstrap, and JavaScript, dynamically updates to display detailed product information, enhancing the shopping experience. Payment processing ensures a smooth checkout, and the system demonstrates high performance and reliability in testing.

4.3 Limitations, Security Concerns, and Market Adaptability

4.3.1 Limitations of the proposed system

While the smart shopping cart system presents a significant advancement in retail technology, several limitations must be addressed to enhance its market adaptability:

1) **Database Constraints:** The current MySQL database design restricts the addition of duplicate entries for products, as the product ID is set as a unique key. This limitation can hinder the management of seasonal products, which may need to be available only during specific times of the year. Migrating to a cloud-based server can alleviate these constraints by allowing for more flexible database management. In a cloud environment, we can implement a more dynamic schema that accommodates seasonal products without the limitations of a traditional database.
2) **Scalability Issues:** As the number of users increases, the system may face performance bottlenecks. Cloud technology can provide scalable resources that adjust based on demand, ensuring consistent performance even during peak shopping times.

3) **User Interface Limitations:** Although the web-based UI is user-friendly, it may not cater to all demographics. Future iterations should consider accessibility features to accommodate diverse users including users with disabilities.

4.3.2 Security Concerns

The integration of IoT technologies in smart shopping cart systems introduces several security vulnerabilities, including:

4.3.2.1 Cybersecurity Threats

1) **Data Breaches:** Unauthorized access to sensitive customer information, such as personal data and payment details.
2) **Denial-of-Service (DoS) Attacks:** Overloading the system to disrupt service availability, potentially affecting the entire shopping experience.
3) **Man-in-the-Middle (MitM) Attacks:** Intercepting and potentially altering communications between the cart and backend systems.

The most vulnerable components of the system are the communication channels (e.g., Wi-Fi) and the payment processing module.

4.3.2.2 Improving Security Measures

1) **End-to-end Encryption:** Implement Transport Layer Security (TLS) for data transmission to protect sensitive information from interception and tampering.
2) **Secure Payment Gateways**: Utilize trusted payment gateways with robust security protocols, such as Payment Card Industry Data Security Standard (PCI DSS) compliance and tokenization to enhance transaction security.
3) **Regular Security Audits:** Conduct periodic security assessments, vulnerability scanning, and penetration testing to identify and rectify vulnerabilities. Utilize tools like OWASP ZAP, which helps identify common security flaws such as SQL injection and cross-site scripting,

or Burp Suite for thorough testing.

4) **Multi-Factor Authentication (MFA):** Implement MFA for user access, including hardware tokens or biometric verification, to add layer of security.
5) **Firmware and Software Updates:** Regularly update the firmware of IoT devices and software components to patch known vulnerabilities and enhance security features.
6) **Network Segmentation:** Segment the IoT devices on a separate network from critical systems to limit the potential impact of a compromised device.
7) **Intrusion Detection Systems (IDS):** Deploy IDS to monitor network traffic for suspicious activities and potential threats.

By implementing these advanced security measures, the smart shopping cart system can significantly improve its resilience against potential attacks, ensuring a secure and seamless shopping experience.

4.3.3 Adoption Blockages in the Retail Industry

Despite the advantages of smart shopping carts, several barriers hinder their widespread adoption in the retail sector:

1) **Cost Concerns:** Retailers may be hesitant to invest in new technologies due to high initial costs and uncertain ROI.
2) **Technical Integration:** Integrating new systems with existing legacy systems can be complex and time-consuming, leading to resistance from retailers.
3) **Resistance to Change:** Many businesses are accustomed to traditional checkout processes and may resist transitioning to automated systems.
4) **Training Requirements:** Staff may require training to effectively use and manage the new technology, which can be seen as an additional burden.
5) **Maintenance and Support:** Retailers may worry about ongoing maintenance costs and the need for technical support.

4.3.4 Enhancements through Cloud Technology

Migrating the system to a cloud-based server can significantly enhance its capabilities and market potential:

1) **Cost Efficiency:** Cloud services typically operate on a pay-as-you-go model, allowing retailers to only pay for the resources they use. This can keep costs low while providing access to powerful computing resources.
2) **Scalability:** Cloud technology allows for easy scaling of resources based on demand. Retailers can adjust their usage during peak shopping seasons without the need for significant upfront investment in hardware.
3) **Database Flexibility:** In a cloud environment, we can implement a dynamic database schema that allows for adding seasonal products without the constraints of traditional databases, enabling retailers to manage inventory more effectively.
4) **Enhanced Security:** Cloud providers often have robust security measures in place, including data encryption, regular security updates, and compliance with industry standards, which can enhance the overall security of the system.
5) **Accessibility:** Cloud-based systems can be accessed from anywhere, allowing retailers to manage their inventory and sales data remotely, which is particularly beneficial for multi-location businesses.
6) **Data Analytics:** Cloud platforms often come with built-in analytics tools that can help retailers gain insights into customer behavior, inventory management, and sales trends, enabling data-driven decision-making.

By addressing these limitations and security concerns, and by enhancing the system's appeal to business owners through cloud technology, the smart shopping cart can become a viable solution in the retail market.

4.4 Comparative Analysis

The following table (Table 4.1) compares the features and advantages of our proposed system with existing shopping cart research:

Factor	Proposed System	Existing Research
Technology	RFID with Raspberry Pi Pico W	Barcode / RFID with traditional microcontrollers
Cost	Low-cost components like Raspberry Pi Pico W	Higher cost due to expensive microcontrollers like Arduino Uno, Node MCU
User Interface	Web-based UI with dynamic updates	LCD Display with buttons or SMS-based billing
Security	Basic security measures	Basic security measures
Scalability	Easily scalable with additional sensors and modules	Limited Scalability
Ease of Use	User-friendly with real-time product details on web application	Less intuitive, requiring manual input
Data Processing	Fast, using MySQL database and efficient communication protocols	Slower, often reliant on hard-coded product details
Communication	Wi-Fi module using HTTP protocol	Bluetooth or Zigbee
Maintenance	Easy to update and maintain with modular components	Complex and costly maintenance
Real-time Updates	Yes, via web-based UI	Limited real-time capabilities
Programming	Uses PHP, HTML, CSS, Bootstrap, JavaScript, and MicroPython	Often limited to microcontroller-specific code

Error Handling	Efficient, with clear error messages	Basic, often requires manual intervention
Feedback Time	Instant	Delayed
Flexibility	High, easily adaptable to new requirements	Low, fixed system configuration
Setup Time	Quick, due to modular components	Lengthy, due to complex hardware setup

Table 4.1: Comparison with Existing Systems

CHAPTER 5: CONCLUSION AND FUTURE WORK

5.1 Conclusion

The development of the smart shopping cart system marks a significant leap forward in enhancing the retail shopping experience through the integration of IoT technologies. By utilizing the Raspberry Pi Pico W microcontroller, RFID technology, and a web-based user interface, this project successfully addresses several limitations of existing systems. The smart cart facilitates real-time product tracking, efficient data management, and payment processing, thus providing a seamless and intuitive shopping experience for users.

Throughout the project, key components such as the RFID module RC522 and the Raspberry Pi Pico W demonstrated their effectiveness in product identification and data transmission. The MySQL database, accessed via PHP scripts hosted on the XAMPP server, ensured reliable data storage and retrieval. The web-based UI, developed using HTML, CSS, Bootstrap, and JavaScript, provided dynamic updates and interactive functionalities, significantly enhancing user engagement.

Testing procedures confirmed the system's high performance, accuracy, and reliability. The smart shopping cart system consistently displayed correct product details upon scanning, maintained stable Wi-Fi connections for data transmission, and processed payments securely. The integration of these components into a cohesive system showcases the potential of IoT-based solutions in revolutionizing retail operations.

5.2 Future Work

The future trajectory of smart shopping cart systems offers a compelling arena for innovation and advancement. To fortify the system against cyber threats, robust cybersecurity measures must be integrated, ensuring data integrity and user privacy. Concurrently, optimizing inventory management practices through predictive analytics can enhance operational efficiency and minimize overhead costs. Moreover, refining the user interface by incorporating LED screens and personalized recommendations promises to elevate the shopping experience, catering to diverse consumer preferences.

Continued exploration of computer vision technology holds immense potential for streamlining product scanning processes and revolutionizing inventory management. Integrating machine learning algorithms further augments the system's capabilities, enabling adaptive product recognition and personalized recommendations. Additionally, adopting blockchain technology can enhance transaction security and transparency, fostering trust and reliability in payment processing systems. Embracing augmented reality applications and voice-activated shopping assistance further enriches the user experience, offering immersive interactions and tailored support. As smart shopping cart systems evolve, leveraging emerging technologies and data-driven insights will be paramount to unlocking their full potential in enhancing retail operations and customer satisfaction.

BIBLIOGRAPHY

[1] A. Lele, ``Internet of Things (IoT)," in Disruptive Technologies for the Militaries and Security (Smart Innovation, Systems and Technologies). 2019, doi: 10.1007/978-981-13-3384-2_11.

[2] D. Bandyopadhyay and J. Sen, ``Internet of Things: Applications and challenges in technology and standardization," in Wireless Personal Communications. 2011, doi: 10.1007/s11277-011-0288-5.

[3] Vaidhyanathan, R. M., et al. "SMART SHOPPING CART USING RFID." Irjet.net, 2008, https://www.irjet.net/archives/V8/i4/IRJET-V8I4415.pdf.

[4] Shahroz, Mobeen, et al. "IoT-Based Smart Shopping Cart Using Radio Frequency Identification." IEEE Access: Practical Innovations, Open Solutions, vol. 8, 2020, pp. 68426–68438, doi:10.1109/access.2020.2986681.

[5] Malenkov, Yury, et al. "Digitalization and Strategic Transformation of Retail Chain Stores: Trends, Impacts, Prospects." Journal of Open Innovation Technology Market and Complexity, vol. 7, no. 2, 2021, p. 108, doi:10.3390/joitmc7020108.

[6] Y. Berdaliyev and A. P. James, ``RFID-cloud smart cart system," in Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI), Sep. 2016, pp. 23462352, doi: 10.1109/ICACCI.2016.7732405.

[7] A. A. Anil, ``RFID based automatic shopping cart," Int. J. Adv. Sci. Res. Eng., vol. 1, pp. 3945, 2018.

[8] S. Dheple, D. Kumari, M. Jadhav, D. Lihitkar, and A. P. Umakanttupe, ``Smart shopping cart with automatic billing for supermarket," Tech. Rep., pp. 16, 2018.

[9] Kaur, J., Arora, V., & Bali, S. (2020). Influence of technological advances and change in marketing strategies using analytics in retail industry. International Journal of System Assurance Engineering and Management, 11(5), 953–961. https://doi.org/10.1007/s13198-020-01023-5

[10] Korzinski, D. (2017, December 11). Bricks & mortar vs. clicks & mobile: Canadians prefer physical stores, but increasingly shop online. Angus Reid Institute. https://angusreid.org/canadian-online-shopping/

[11] Smart Shopping Carts Market Size, Share, Growth, And Industry Analysis by Type (Zigbee, RFIDs, Bar Codes), By Application (Supermarket, Shopping Malls, Others), Regional Insights, and Forecast To 2032 https://www.businessresearchinsights.com/market-reports/smart-shopping-carts-market-108229

[12] Smart shopping cart market trends, industry size, research and forecast 2033. (n.d.). Thebusinessresearchcompany.com. Retrieved May 9, 2024, from https://www.thebusinessresearchcompany.com/report/smart-shopping-cart-global-market-report

[13] Iyer, J., Dhabu, H., & Mohanty, S. K. (2008). Smart trolley system for automated billing using RFID and ZIGBEE. Ijetae.com. https://www.ijetae.com/files/Volume5Issue10/IJETAE_1015_20.pdf

[14] Chandrasekar, P., & Sangeetha, T. (2014). Smart shopping cart with automatic billing system through RFID and ZigBee. International Conference on Information Communication and Embedded Systems (ICICES2014), 1–4.

[15] Vrinda, M., Department of Computer Science ITM University, Gurgaon, & Niharika, N. (2014). Novel model for automating purchases using intelligent cart. IOSR Journal of Computer Engineering, 16(1), 23–30. https://doi.org/10.9790/0661-16172330

[16] Yewatkar, A., Inamdar, F., Singh, R., Ayushya, & Bandal, A. (2016). Smart cart with automatic billing, product information, product recommendation using RFID & zigbee with anti-theft. Procedia Computer Science, 79, 793–800. https://doi.org/10.1016/j.procs.2016.03.107

[17] Dhavale Shraddha D, Dhokane Trupti J, Shinde Priyanka S. (2016). IOT Based Intelligent Trolley for Shopping Mall. https://www.ijedr.org/papers/IJEDR1602225.pdf

[18] Sainath, S., Surender, K., Vikram Arvind, V., & Thangakumar, J. (n.d.). Automated shopping trolley for super market billing system. Ijcaonline.org. Retrieved May 10, 2024, from https://research.ijcaonline.org/icccmit2014/number3/icccmit7026.pdf

[19] Budic, D., Martinovic, Z., & Simunic, D. (2014). Cash register lines optimization system using RFID technology. 2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 459–462.s

[20] Shankar, J. R., Nandkumar, A. P., Vaijanath, T. S., & Pawar, S. U. (n.d.). RFID based Automatic Billing Trolley. Psu.edu. Retrieved May 10, 2024, from https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=b79d9707bb36fdfba38f560c926e92688690205b

[21] Kumar, R., Gopalakrishna, K., & Ramesha, K. (n.d.). Intelligent Shopping Cart. Ijesit.com. Retrieved May 10, 2024, from https://www.ijesit.com/Volume%202/Issue%204/IJESIT201304_64.pdf

[22] Gangwal, U., Roy, S., & Bapat, J. (n.d.). Smart shopping cart for automated billing purpose using wireless sensor networks. Interiorkraft.com. Retrieved May 10, 2024, from https://interiorkraft.com/upload_pdf/sensorcomm_2013_7_30_10155.pdf

[23] Kumar, M., Singh, J., & Sanduja, V. (n.d.). Smart trolley with instant billing to ease queues at shopping malls using ARM LPC148: A review. https://doi.org/10.17148/IJARCCE.2015.4808

[24] Athisha, L., Abhishek, A., Harshith, R., Darshan, K. S. R., & Srinidhi, K. M. (n.d.). Automation of shopping cart to ease queue in malls by using rfid. Irjet.net. Retrieved May 10, 2024, from https://www.irjet.net/archives/V2/i3/Irjet-v2i3217.pdf

[25] Sankhe, Nitin & Pandit, Chinmayee & Dhodi, Priya & Katkar, Sunil. (2019). IOT BASED RETAIL STOCK MANAGEMENT (A Smart Shelf), Issue 4 www.jetir.org (ISSN-2349-5162).

[26] Shrinath, E., & Jayanthi, S. (2024). Design and implementation of smart vending machine using IoT and comparison with RFID based vending machine. 16TH INTERNATIONAL ENGINEERING AND COMPUTING RESEARCH CONFERENCE (EURECA).

[27] Pangriya, R. (2022). Beacon technology the future of retail: A review of the literature and SWOT analysis. Review of Professional Management, 2(2), 120–130. https://doi.org/10.1177/09728686221143564

[28] Venkata Naga Jayudu, T., Sree, N. S., Syamala, B. K., Sowdhanya, K., & Saranya, R. C. (n.d.). Iot based anti-theft security system. Jetir.org. Retrieved May 10, 2024, from https://www.jetir.org/papers/JETIR2304529.pdf

[29] Sadaat, Hamza & Shoukat, Omer, "RFID-Based Smart Parking", 2022 doi: 50. 10.13140/RG.2.2.24165.37602.

[30] Pallavi*, P., Priyanka, V., & Sai, D. Y. P. (2020). Design & Verification of Serial Peripheral Interface (SPI) protocol. International Journal of Recent Technology and Engineering (IJRTE), 8(6), 793–796. https://doi.org/10.35940/ijrte.f7356.038620

APPENDIX A: CODE LISTINGS

A.1. Microcontroller Code for Communication with RFID RC522, OLED Display, and Web Server

```
from machine import Pin, SPI
from mfrc522 import MFRC522
import time
import network
import urequests as requests
from sh1106 import SH1106_SPI
from oled import Write, GFX
from oled.fonts import ubuntu_mono_15, ubuntu_mono_20

wlan = network.WLAN(network.STA_IF)
wlan.active(True)
ssid = "SPSETUP-3A1C"
password = "filter4548bacon"
wlan.connect(ssid, password)

WIDTH = 128  # OLED display width
HEIGHT = 64  # OLED display height
spi_port = 0
SCLK = 6
MOSI = 19
CS = 21
DC = 22
RST = 13

rfid_reader = MFRC522(spi_id=0, sck=2, miso=4, mosi=7, cs=5, rst=18)
spi = SPI(spi_port, baudrate=1000000, mosi=Pin(MOSI), sck=Pin(SCLK))
display = SH1106_SPI(WIDTH, HEIGHT, spi, dc=Pin(DC), res=Pin(RST), cs=Pin(CS))

# Manually set segment remap and scan direction
display.write_cmd(0xA1)  # Segment remap (A0 or A1)
display.write_cmd(0xC8)  # Scan direction (C0 or C8)
fontSize20 = Write(display, ubuntu_mono_20)

# Wait for connect or fail
wait = 10
while wait > 0:
    if wlan.status() < 0 or wlan.status() >= 3:
        break
    wait -= 1
    print('Waiting for Wi-Fi connection...')
    display.fill(0)
```

```
    display.text("Waiting for", 0, 15, 1)
    display.text("Wi-Fi Connection", 0, 30, 1)
    display.text("...", 0, 45, 1)
    display.show()
    time.sleep(1)

# Handle connection error
if wlan.status() != 3:
    display.fill(0)
    display.text("Wi-Fi Connection", 0, 5, 1)
    display.text("Failed with", 0, 20, 1)
    display.text(ssid, 0, 35, 1)
    display.text("Try Again...", 0, 55, 1)
    display.show()
    raise RuntimeError('Wi-Fi connection failed')
else:
    print('Connected to Wi-FI: '+ ssid)
    print('IP: ', wlan.ifconfig()[0])
    display.fill(0)
    display.text("Connected to", 0, 15, 1)
    display.text("Wi-FI: ", 0, 30, 1)
    display.text(ssid, 0, 45, 1)
    display.show()
    time.sleep_ms(1000)

print("Scan the product...")
display.fill(0)
display.text("Start scanning", 0, 15, 1)
display.text("your items...", 0, 30, 1)
display.show()

server_url = "http://10.0.0.36/shoppy/getUID.php"

while True:
    rfid_reader.init()
    (card_status, tag_type) = rfid_reader.request(rfid_reader.REQIDL)
    if card_status == rfid_reader.OK:
        (card_status, card_id) = rfid_reader.SelectTagSN()
        if card_status == rfid_reader.OK:
            rfid_card = int.from_bytes(bytes(card_id), "little", False)
            print("Detected Product ID: " + str(rfid_card))
            display.fill(0)
            display.text("Detected Item", 0, 15, 1)
            display.text("ID: " + str(rfid_card), 0, 30, 1)
            display.show()
            try:
```

```
                response = requests.post(server_url, headers={"Content-Type":
"application/x-www-form-urlencoded"}, data="UIDresult=" + str(rfid_card))
                print("UID sent:" + str(rfid_card))
                print("HTTP status code:", response.status_code)
                print("Response payload:", response.text)
            except OSError as e:
                print("Failed to send HTTP request:", e)
                display.fill(0)
                display.text("Turn On (or) ", 0, 15, 1)
                display.text("Restart the", 0, 30, 1)
                display.text("web server", 0, 45, 1)
                display.show()
                time.sleep_ms(8000)
            time.sleep_ms(1000)
            print("Scan the next product...")
            display.fill(0)
            display.text("Scan the next", 0, 15, 1)
            display.text("item...", 0, 30, 1)
            display.show()
time.sleep_ms(500)
```

A.2. Source Code for the Shopping Cart page in the Web Application

```
<?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="https://code.jquery.com/jquery-3.6.0.min.js"></script> <!--
Include jQuery library -->
        <script>
            var uidInterval;
            // Check local storage for the toggle state on page load
            $(document).ready(function() {
                var instantModeEnabled = localStorage.getItem('instantModeEnabled');
                if (instantModeEnabled === 'true') {
                    $('#instantModeToggle').prop('checked', true);
                    startFetchingUID(); // Start fetching UID if instant mode was
enabled
                } else {
                    $('#instantModeToggle').prop('checked', false);
                }
            });

            function toggleInstantMode() {
                var instantMode = $("#instantModeToggle").prop("checked");
                // Store the toggle state in local storage
                localStorage.setItem('instantModeEnabled', instantMode);

                if (instantMode) {
                    startFetchingUID();
                } else {
                    stopFetchingUID();
                }
            }

            function startFetchingUID() {
                $("#getUID").load("UIDContainer.php");
                uidInterval = setInterval(function() {
```

```
                $("#getUID").load("UIDContainer.php", function() {
                    var uid = $("#getUID").text().trim();
                    if (uid) {
                        addToCart(uid);
                    }
                });
            }, 500);
        }

        function stopFetchingUID() {
            clearInterval(uidInterval);
        }

        function addToCart(uid) {
            $.ajax({
                url: 'instantmode.php',
                type: 'POST',
                data: { uid: uid },
                success: function(response) {
                    var result = JSON.parse(response);
                    if (result.status === 'success') {
                        console.log('Product added to cart');
                        setTimeout(function() {
                            window.location.reload();
                        }, 100); // Reload the page after a brief delay
                    } else {
                        console.error(result.message);
                    }
                },
                error: function(xhr, status, error) {
                    console.error('Error adding product to cart:', error);
                }
            });
        }
    </script>
    <script>
        $(document).ready(function () {
            $("#clearCartBtn").click(function () {
                // Send AJAX request to clear cart
                $.ajax({
                        url: 'clear_cart.php',
                    type: 'POST',
                    data: {
                        action: 'clear_cart'
                    },
                    success: function (response) {
```

```
                        // This function will be called on successful clearance
                        console.log('Cart cleared successfully:', response);
                        setTimeout(function(){
                            window.location.reload();
                        }, 100); // Reload the page
                    },
                    error: function (xhr, status, error) {
                        // This function will be called if there's an error
                        console.error('Error clearing cart:', error);
                    },
                });
            });
        });
    </script>
    <style>
    html {
        font-family: Arial;
        display: inline-block;
        margin: 0px auto;
        text-align: center;
    }

    ul.topnav {
        border-radius: 40px;
        list-style-type: none;
        margin: auto;
        padding: 0;
        overflow: hidden;
        background-color: #00008B;
        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: #FFA500;}

    ul.topnav li a.active {background-color: #333;}
```

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

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

        .table {
            margin: auto;
            width: 90%;
        }

        thead {
            color: #FFFFFF;
        }

        /* Style for the toggle switch */
        .toggle-label {
            font-size: 1.25em;
            font-weight: bold;
            display: inline-block;
            margin-left: 10px;
        }

        .switch {
        position: relative;
        display: inline-block;
        width: 60px;
        height: 34px;
        }

        .switch input {
        opacity: 0;
        width: 0;
        height: 0;
        }

        .slider {
        position: absolute;
        cursor: pointer;
        top: 0;
        left: 0;
        right: 0;
        bottom: 0;
        background-color: #ccc;
        transition: .4s;
```

```
        border-radius: 34px;
        }

        .slider:before {
        position: absolute;
        content: "";
        height: 26px;
        width: 26px;
        left: 4px;
        bottom: 4px;
        background-color: white;
        transition: .4s;
        border-radius: 50%;
        }

        input:checked + .slider {
        background-color: #2196F3;
        }

        input:checked + .slider:before {
        transform: translateX(26px);
        }

        /* Style the slider */
        .slider.round {
        border-radius: 34px;
        }

        .slider.round:before {
        border-radius: 50%;
        }
        </style>

        <title>Shopping Made Easier with Shoppy</title>
    </head>
    <body>
        <h2>Shoppy: Instant Checkout</h2>
        <ul class="topnav">
            <li><a href="home.php">Home</a></li>
            <li><a href="read_product.php">Read Product ID</a></li>
            <li><a class="active" href="shopping_cart.php">Shopping Cart</a></li>
            <li class="right"><a href="admin_access.php">Admin</a></li>
        </ul>
        <br>
        <div class="container">
            <div class="row">
```

```
                <h3>Shopping Cart</h3>
            </div>
            <br>
            <!-- Toggle Switch for Instant Mode -->
            <label class="switch">
            <input type="checkbox" id="instantModeToggle"
onchange="toggleInstantMode()">
            <span class="slider round"></span>
            </label>
            <span class="toggle-label">Instant Mode</span>
            <br><br>
            <!-- UID Container (hidden) -->
            <div id="getUID" style="display: none;"></div>
            <div class="row">
                <table class="table table-striped table-bordered">
                  <thead>
                    <tr bgcolor="#FFA500" color="#FFFFFF">
                      <th>S.No</th>
                      <th>Product</th>
                      <th>ID</th>
                      <th>Category</th>
                      <th>Amount</th>
                      <th>Price</th>
                      <th>Action</th>
                    </tr>
                  </thead>
                  <tbody>
                  <?php
                   include 'database.php';
                   $pdo = Database::connect();
                   $totalPrice = 0; // Initialize $totalPrice variable
                   $sql = 'SELECT * FROM shopping_cart ORDER BY serialnumber ASC';
                   foreach ($pdo->query($sql) as $row) {
                            echo '<tr>';
                            echo '<td>'. $row['serialnumber'] . '</td>';
                            echo '<td>'. $row['product'] . '</td>';
                            echo '<td>'. $row['id'] . '</td>';
                            echo '<td>'. $row['category'] . '</td>';
                            echo '<td>'. $row['amount'] . '</td>';
                            echo '<td>'. $row['price'] . '</td>';
                            echo ' ';
                            echo '<td><a class="btn btn-danger"
href="shopping_cart_remove.php?serialnumber='.$row['serialnumber'].'">Remove</a>';
                            echo '</td>';
                            echo '</tr>';
                            $totalPrice += $row['price']; // Add each price to total
```

```
                    }
                    Database::disconnect();
                     echo '<tr>'; // Display total row
                     echo '<td colspan="5"><strong>Total:</strong></td>';
                     echo '<td><strong>'. $totalPrice . '</strong></td>';
                     echo '<td></td>';
                     echo '</tr>';
                     echo '<tr>';
                     echo '</tr>';
                    ?>
                    </tbody>
                </table>
            </div>
        </div> <!-- /container -->
        <br><br>
        <div class="col-md-6">
            <button id="clearCartBtn" class="btn btn-danger">Clear Cart</button>
        </div>
        <br><br>
        <td colspan="6" align="center"><a class="btn btn-primary"
href="payment.php?totalPrice=<?php echo $totalPrice; ?>">Pay now</a></td>
        <!-- Images -->
        <div class="image-container" style="margin: 20px auto; max-width: 80%;">
            <!-- Left Image -->
            <img src="storehours.png" alt="" style="width: 30%; float: left; margin-
right: 60px;">
            <!-- Right Image -->
            <img src="donate.png" alt="" style="width: 30%; float: right; margin-left:
60px;">
        </div>
        </div>
    </body>
</html>
```

APPENDIX B: DATASHEETS AND TECHNICAL DOCUMENTS

B.1. Brief Datasheet of Raspberry Pi Pico W

Overview

The Raspberry Pi Pico W adds wireless features to the popular Raspberry Pi Pico. It uses the RP2040 chip, which is made to be fast, affordable, and easy to use for microcontroller projects. The RP2040 has a lot of memory, a dual-core processor, and a special system called Programmable I/O (PIO) that makes it powerful and flexible. It is built using modern technology to be efficient with power, which helps it run longer on batteries.

The RP2040 is easy to use, with plenty of documentation, a good MicroPython port, and a UF2 bootloader in ROM. This makes it simple for beginners and hobbyists to start using it. The Raspberry Pi Pico W can connect to Wi-Fi (2.4GHz 802.11 b/g/n) and Bluetooth 5.2. It has a built-in antenna and can work in both station and access-point modes, so both C and MicroPython developers can use it for network tasks.

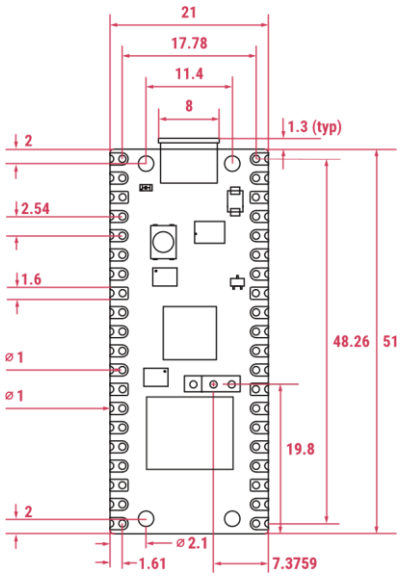
The Raspberry Pi Pico W comes with 2MB of flash memory and a power supply chip that can handle input voltages from 1.8 to 5.5V. It has 26 GPIO pins, three of which can be used as analog inputs. The pins are arranged on 0.1” pitch through-hole pads with castellated edges, making them easy to connect.

The Raspberry Pi Pico W is sold as a single unit or in 480-unit reels for larger projects. This makes it useful for both individual makers and big manufacturers. More detailed information and resources can be found on the Raspberry Pi Foundation’s website to help users get the most out of the Raspberry Pi Pico W.

Specification

Form factor:	21 mm × 51 mm
CPU:	Dual-core Arm Cortex-M0+ @ 133MHz
Memory:	264KB on-chip SRAM; 2MB on-board QSPI flash
Interfacing:	26 GPIO pins, including 3 analogue inputs
Peripherals:	• 2 × UART • 2 × SPI controllers • 2 × I2C controllers • 16 × PWM channels • 1 × USB 1.1 controller and PHY, with host and device support • 8 × PIO state machines
Connectivity:	2.4GHz IEEE 802.11b/g/n wireless LAN, on-board antenna Bluetooth 5.2 • Support for Bluetooth LE Central and Peripheral roles • Support for Bluetooth Classic
Input power:	1.8–5.5V DC
Operating temperature:	-20°C to +70°C
Production lifetime:	Raspberry Pi Pico W will remain in production until at least January 2034
Compliance:	For a full list of local and regional product approvals, please visit pip.raspberrypi.com

Physical Specification



Note: all dimensions in mm

WARNINGS

- Any external power supply used with Raspberry Pi Pico W shall comply with relevant regulations and standards applicable in the country of intended use.
- This product should be operated in a well-ventilated environment, and if used inside a case, the case should not be covered.
- Whilst in use, this product should be placed on a stable, flat, non-conductive surface, and should not be contacted by conductive items.
- The connection of incompatible devices to Raspberry Pi Pico W may affect compliance, result in damage to the unit, and invalidate the warranty.
- All accessories used with this product should comply with relevant standards for the country of use and be marked accordingly to ensure that safety and performance requirements are met.
- The cables and connectors of all peripherals used with this product must have adequate insulation so that relevant safety requirements are met.

SAFETY INSTRUCTIONS

To avoid malfunction or damage to this product, please observe the following:

- Do not expose to water or moisture, or place on a conductive surface whilst in operation.
- Do not expose to heat from any source; Raspberry Pi Pico W is designed for reliable operation at normal ambient temperatures.
- Take care whilst handling to avoid mechanical or electrical damage to the printed circuit board and connectors.
- Whilst it is powered, avoid handling the printed circuit board, or only handle it by the corners to minimize the risk of electrostatic discharge damage.