IOT AND AI BASED SMART SHOPPING TROLLEY.

Mini Project Report

Submitted in partial fulfillment of the requirements for the award of the Degree of

Bachelor of Technology (B.Tech)

In

COMPUTER SCIENCE AND ENGINEERING(IoT)

By

PITTALA MANIGAYATHRI	**21AG1A6949**
GANJI GANESH	**21AG1A6923**
MADIREDDY SHASHANTH REDDY	**21AG1A6940**
VARKUTI NITISH REDDY	**21AG1A6961**

Under the Esteemed Guidance of

Mr.B.Mohan

Assistant Professor

Department of Computer Science and Engineering(IoT)

ACE ENGINEERING COLLEGE

An Autonomous Institution

(NBA ACCREDITED B.TECH COURSES: EEE, ECE, MECH, CIVIL & CSE, ACCORDED NAAC 'A' GRADE)

(Affiliated to Jawaharlal Nehru Technological University, Hyderabad, Telangana)

Ghatkesar, Hyderabad - 501 301

December 2023

ACE

Engineering College

An Autonomous Institution

(NBA ACCREDITED B.TECH COURSES: EEE, ECE, MECH, CIVIL & CSE, ACCORDED NAAC 'A'GRADE)

Ghatkesar, Hyderabad- 501 301

(Affiliated to Jawaharlal Nehru Technological University Hyderabad)

Web site: www.aceec.ac.in E-mail: info@aceec.ac.in

CERTIFICATE

This is to certify that the Mini project work entitled **"IOT AND AI BASED SMART SHOPPING TROLLEY"** is being submitted by **Pittala Manigayathri(21AG1A6949), GanjiGanesh (21AG1A6923), Madireddy Shashanth Reddy(21AG1A6940), Varkuti Nitish Reddy (21AG1A6961)** in partial fulfillment for the award of Degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE AND ENGINEERING(IoT)** to the Jawaharlal Nehru Technological University, Hyderabad during the academic year 2024-25 is a record of bonafide work carried out by him under our guidance and supervision.

The results embodied in this report have not been submitted by the student to any other University or Institution for the award of any degree or diploma.

Internal Guide

Mr .B. Mohan

Assistant Professor

Dept. of CSE(IoT)

Head of the Department

Dr.K Prem Kumar

Associate Professor and Head

Dept. of CSE(IoT)

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

I would like to express our gratitude to all the people behind the screen who have helped me, transform an idea into a real time application.

I would like to express my heart-felt gratitude to our parents without whom I would not have been privileged to achieve and fulfill my dreams.

A special thanks to our Secretary, **Prof. Y. V. GOPALA KRISHNA MURTHY,** for having founded such an esteemed institution. I am also grateful to our beloved principal, **Dr. B. L. RAJU** for permitting us to carry out this project.

I profoundly thank **Dr. K Prem Kumar**, Head of the Department of Computer Science and Engineering(IoT), who has been an excellent guide and also a great source of inspiration to my work.

I extremely thank **Mr. V Veeresh** Associate Professor, Project coordinator who helped us in all the way in fulfilling of all aspects in completion of our Mini-Project.

I am very thankful to my internal guide **Mr. B. Mohan,** Assistant Professor, of the Department of Computer Science and Engineering(IoT) who has been an excellent and also given continuous support for the Completion of my project work.

The satisfaction and euphoria that accompany the successful completion of the task would be great, but incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crown all the efforts with success. In this context, We would like to thank all the other staff members, both teaching and non-teaching, who have extended their timely help and eased our task.

Pittala Manigayathri (21AG1A6949)
Ganji Ganesh (21AG1A6923)
Madi Reddy Shashanth Reddy (21AG1A6940)
Varkuti Nitish Reddy (21AG1A6961)

DECLARATION

I hereby declare that this mini project entitled "**IOT AND AI BASED SMART SHOPPING TROLLEY**" Submitted to the **ACE Engineering College,** is a record of an original work done by me under the guidance of **Mr. B.Mohan**, Assistant Professor of the Department of Computer Science and Engineering(IoT), **ACE Engineering College,** and this project work submitted in the partial fulfillment of the requirements for the mini project; the results embodied in this thesis have not been submitted to any other university or institute for award of any degree or diploma.

ABSTRACT

In today's fast-paced world, enhancing the shopping experience is critical to meet the demands of modern consumers. The IoT and AI-based Smart Shopping Trolley is an innovative solution designed to revolutionize retail environments by offering unmatched convenience , efficiency , and personalized service . This intelligent system seamlessly integrates Internet of Things (IoT) sensors and Artificial Intelligence (AI) algorithms to create a smarter, more engaging shopping experience . IoT sensors embedded in the trolley enable real-time tracking of items as customers add them, automatically updating the shopping list and eliminating the need for manual barcode scanning. This automation not only enhances efficiency but also reduces the chances of human error. AI further elevates the shopping journey by analyzing customer purchasing patterns and preferences to provide tailored product recommendations , ensuring a more personalized experience. One of the key features of the Smart Shopping Trolley is its ability to autonomously follow customers around the store . By leveraging advanced navigation algorithms and smart sensors , the trolley ensures smooth and effortless movement, freeing customers from the need to push it manually . Additionally , the trolley integrates with mobile payment systems to enable a seamless and cashless checkout process, significantly reducing time spent at traditional cash registers. The IoT and AI-based Smart Shopping Trolley represents a significant leap forward in the evolution of retail. It bridges the gap between physical and digital shopping , addressing common challenges such as long check out queues, lack of personalization , and inefficiencies in-store navigation. By leveraging cutting-edge technology , the Smart Shopping Trolley transforms the way people shop , offering a faster , smarter, and more enjoyable experience, this innovative solution redefines the retail experience for consumers. The IoT and AI-based Smart Shopping Trolley is not just a technological advancement but a step toward creating smarter and more efficient retail ecosystems.

INDEX

CONTENTS	PAGE NO.
1. INTRODUCTION	9
1.1. PROBLEM STATEMENT	10
2. LITERATURE SURVEY	12
3. SYSTEM ANALYSIS	13
3.1. EXISTING SYSTEM	13
3.2. PROPOSED SYSTEM	14
3.3. SOFTWARE REQUIREMENTS SPECIFICATION	15
3.3.1. Introduction	15
3.3.2. Purpose	16
3.3.3. Scope of the Project	16
3.3.4. Overall Description	16
3.3.5. System Features	17
3.3.6. Hardware Requirements	17
3.3.7. Software Requirements	19
4. SYSTEM DESIGN	20
4.1. DATA FLOW DIAGRAMS (DFDs)	20
4.2. UML DIAGRAMS	23
4.2.1. Class Diagram	24
4.2.2. Use-Case Diagram	25
4.2.3. Activity Diagram	26
4.2.3. Sequence Diagram	27
4.2.3. Component Diagram	28
4.3. ARCHITECTURAL DESIGN	29
5. MODULES	31

6. IMPLEMENTATION 35

6.1. IMPLEMENTATION OF EACH MODULE 35
6.2. Sample Code 35

7. TESTING 42

7.1 TYPES OF TESTING 42
7.2. TESTING OF VARIOUS MODULES OF THE PROJECT 43

8. OUTPUT SCREENS 46

9. DEPLOYMENT OF THE PROJECT 48
9.1 REQUIRED LANGUAGES 48
9.2 INSTALLING DEPENDENCIES 48
9.3 LOADING THE CODE 49
9.4 RUNNING THE CODE 49
9.5 INTERACTING WITH THE SYSTEM 49

10. INTEGRATION AND EXPERIMENTAL RESULTS 51

11. PERFORMANCE EVALUATION 53

12. COMPARISION WITH EXISTING SYSTEM 55

13. CONCLUSION 57

14. FUTURE ENHANCEMENTS 58

LIST OF FIGURES

Fig No.	Figure Name	Page No.
4.1.1	DFD Level 0	20
4.1.2	DFD Level 1	21
4.1.3	DFD Level 2	22
4.2.1	Class Diagram	24
4.2.2	Use-case Diagram	25
4.2.3	Activity Diagram	26
4.2.4	Sequence Diagram	27
4.3	ARCHITECTURAL DIAGRAM	29

CHAPTER-1 INTRODUCTION

The retail industry is at the forefront of technological transformation, with evolving consumer demands necessitating innovative solutions to enhance the shopping experience. The IoT and AI-based Smart Shopping Trolley is a groundbreaking project designed to revolutionize traditional retail by seamlessly integrating Internet of Things (IoT) sensors and simple suggestion algorithms. This intelligent system addresses common challenges in the shopping process, such as long checkout queues, inefficient inventory management, and lack of personalized services, while introducing unparalleled convenience, efficiency, and customization. IoT sensors embedded in the trolley allow for real-time tracking of items as they are added, automatically updating the virtual shopping cart without the need for manual barcode scanning. This not only simplifies the shopping process but also significantly reduces human errors. Suggestions algorithm takes personalization to the next level by analyzing customer preferences, purchasing history, and behaviour patterns to provide tailored product recommendations and promotions. As a result, shoppers receive a highly customized and engaging experience that enhances their overall satisfaction. One of the standout features of the Smart Shopping Trolley is its autonomous navigation capability, which uses smart sensors and navigation algorithms to follow customers around the store. This feature eliminates the physical burden of pushing a trolley and allows shoppers to focus entirely on selecting products. Additionally, the trolley integrates seamlessly with mobile payment systems to provide a smooth and cashless checkout process. By by passing traditional checkout counters, it minimizes wait times and enhances operational efficiency. For retailers, the system could offer a wealth of opportunities to leverage data collected through IoT and AI. Real-time insights into consumer behaviour enable better inventory management, optimized product placement, and data-driven marketing strategies that cater to customer needs. These capabilities not only improve operational efficiency but also create new revenue streams and build stronger customer relationships. By bridging the gap between physical and digital retail experiences, the IoT and AI-based Smart Shopping Trolley represents a significant step forward in the evolution of retail. It combines the convenience of online shopping with the tactile and immersive experience of in-store shopping, creating a hybrid model that caters to the demands of modern consumers. This project aims to redefine the shopping landscape by offering a smarter, more connected, and customer-centric approach, ensuring that both shoppers and retailers benefit from cutting-edge technology . Moreover, this innovation aligns with the broader trend of digital transformation in retail, empowering businesses to stay competitive in an increasingly

tech-driven marketplace. By enabling autonomous operations and delivering actionable insights, the Smart Shopping Trolley not only improves day-to-day shopping experiences but also sets the foundation for future advancements in smart retail ecosystems. The integration of IoT and AI showcases the potential for technology to make shopping not just easier, but also more efficient and enjoyable, marking a pivotal moment in the evolution of consumer-focused solutions.

1.1. Problem Statement

The traditional shopping experience, particularly in large retail environments, often involves inefficiencies that negatively impact both customers and retailers. As shoppers navigate large aisles, they frequently face long queues at checkout counters, which significantly increases the time spent in-store. Furthermore, shoppers are left without an easy way to track their spending in real time, which can lead to overspending or the frustration of not being able to manage their budget effectively. Another major issue is the lack of personalized shopping experiences. Retailers are often unable to offer tailored recommendations, leaving customers to search through vast amounts of products that may not align with their preferences or needs.

This inefficiency in the shopping process can result in reduced customer satisfaction, longer shopping times, and even lost sales for retailers. Customers may become frustrated with the checkout delays, or may abandon their purchases entirely due to long wait times. Retailers, on the other hand, struggle with limited data on consumer behavior in real-time, making it difficult to tailor their offerings or provide personalized services.

There is a clear demand for a technological solution that can address these issues and streamline the entire shopping experience. The advent of Internet of Things (IoT) and Artificial Intelligence (AI) offers a promising pathway to solve these challenges. By integrating these technologies into shopping trolleys, retailers can enhance customer experience and operational efficiency.

An IoT and AI-based smart shopping trolley can transform the shopping process by automating the checkout process, eliminating the need for traditional cashier interaction. This could be achieved by equipping the trolley with sensors and RFID technology, allowing customers to simply place items in the cart without needing to go through a checkout counter. The trolley could track the cost of the items in real time, providing the shopper with an updated total, enabling them to stay within their budget.

Moreover, AI-powered algorithms can be used to offer personalized recommendations. By analyzing shopping patterns, preferences, and previous purchase history, the smart trolley could suggest products that are tailored to the shopper's needs and preferences, creating a more customized and engaging shopping experience. This level of personalization is a significant advantage over traditional shopping methods, where customers often have to rely on their own instincts or generic signage to find relevant products.

CHAPTER – 2
LITERATURE SURVEY

Paper Title: The Smart Luggage System

Authors: T. Sivasakthi, S. Gayathri, K. Jeyapiriya

Year of Publication: 2020

Description: This paper describes a smart luggage system designed to follow its owner autonomously in crowded environments such as airports and railway stations. The system integrates an Arduino microcontroller with a Bluetooth HC-05 module, allowing users to control the luggage through a smartphone application. Key features include a gyro sensor that helps guide movement and an integrated portable charger for devices. The system reduces the need for manual towing, providing an efficient, hands-free solution to luggage management, and has potential applications in assisting elderly or physically challenged travelers.

Purpose of Referring to This Paper for My Project:

This study offers insights into Bluetooth-based mobile control and autonomous movement, which are relevant to my project on IoT-enabled smart luggage. Its exploration of autonomous navigation and user-friendly control aligns well with my project goals of improving travel convenience through automation and remote control.

ACE Engineering College

CHAPTER – 3

SYSTEM ANALYSIS

3.1. EXISTING SYSTEM

The current shopping process in physical retail stores involves manual and time-consuming steps for both customers and retailers:

Customer Experience:

1. **Trolley Selection and Shopping Process:**
    - Customers use traditional trolleys to pick items from shelves.
    - Trolleys need to be manually pushed without any automated following or assistance
2. **Checkout Process:**
    - Customers wait in long queues at checkout counters, especially during peak hours.
    - Each item in the trolley must be scanned individually by a cashier or through a self-checkout system.
    - Payment is made either through cash , card, or mobile payment methods, often requiring manual intervention.
3. **Challenges Faced by Customers:**
    - **Time Consumption:** Long waiting times at checkout counters.
    - **Inefficiency:** Customers may struggle to locate items in the store.
    - **Lack of Personalization:** No recommendations or tailored assistance during the shopping process.

3.2. PROPOSED SYSTEM

The existing shopping experience in physical retail stores is predominantly manual, leading to inefficiencies for both shoppers and retailers. Customers use traditional trolleys to select items from shelves and place them into the trolley without any assistance or automated validation . The process is entirely manual, leaving room for errors such as selecting incorrect products or missing promotional offers. Once shopping is complete , customers must queue at checkout counters where items are scanned individually, either by a cashier or through self-service systems. This checkout process often results in long wait times, especially during peak hours, causing frustration among shoppers. Furthermore, it is prone to errors, such as pricing discrepancies or unscanned items, and can be time-consuming for those with large purchases. On the retailer's side, inventory management relies on periodic manual audits , which are labor-intensive and can delay restocking. This outdated approach often leads to stockouts or overstocking, reducing operational efficiency and resulting in lost sales opportunities. Retailers also lack real-time insights into inventory levels, customer preferences, and shopping behaviors, limiting their ability to optimize stocking strategies or offer personalized services. Additionally, cashiers are required to process large volumes of transactions, increasing labor costs and the potential for errors . The current system does not integrate automation or modern technologies like IoT or AI, which could provide real-time tracking, streamline inventory management, and enhance the shopping experience. It also falls short of offering personalized customer engagement, as there are no mechanisms to track individual preferences, suggest items, or offer targeted promotions during the shopping journey. This absence of technological integration not only impacts customer satisfaction but also prevents retailers from meeting the modern expectations of convenience, efficiency , and personalization. Overall, the traditional system lacks the convenience and innovation required to address the growing demand for smarter, faster, and more customer-centric retail experiences. The traditional shopping process in physical retail stores is increasingly viewed as outdated in an era of automation and digital transformation. Customers face significant inefficiencies from the moment they enter the store. The use of standard trolleys provides no assistance , such as keeping track of selected items, calculating costs in real-time, or identifying product details like discounts or availability . This leaves shoppers to rely solely on their memory or manually check prices, which can be inconvenient and prone to mistakes. Furthermore , the checkout process remains the most cumbersome part of the experience , requiring shoppers to unload items for scanning and re-pack them afterward, wasting time and creating bottlenecks. Long queues, especially during peak hours or festive seasons , add to customer dissatisfaction, potentially driving them toward online shopping alternatives . Errors such as forgetting items in the trolley or pricing mismatches during scanning further exacerbate these issues.

ACE Engineering College

3.3. SOFTWARE REQUIREMENTS SPECIFICATION

3.3.1. Introduction :

The Software Requirements Specification (SRS) for the IoT and AI-Based Smart Shopping Trolley details the functional and non-functional requirements for a system designed to transform the traditional retail shopping experience. This system leverages IoT devices, such as sensors and RFID readers, along with AI-powered analytics, to automate and enhance shopping processes. The trolley will detect and validate items as they are added or removed, calculate the total cost in real time, and provide product details like price. It integrates a cashless payment system, enabling customers to complete their purchases directly from the trolley, eliminating the need for long checkout queues.

The system also offers features like shopping list integration and personalized product recommendations based on customer preferences and behavior patterns. For retailers, the software provides real-time inventory updates, sales analytics, and insights into customer shopping trends. It supports dynamic promotions, helping retailers push targeted offers during the shopping process.

The software is designed for reliability, scalability, and security, ensuring efficient operation even during peak store hours.

ACE Engineering College

3.3.2. Purpose :

The purpose of the IoT and AI-based smart shopping trolley project is to enhance the shopping experience by automating and simplifying processes. The trolley uses IoT devices to scan products, provide real-time suggestions, and enable cashless payments. AI algorithms personalize product recommendations based on user preferences, improving customer satisfaction. The system reduces manual efforts, eliminates the need for checkout lines, and increases operational efficiency for retailers. It also provides analytics for inventory management and consumer behavior insights, optimizing store operations.

3.3.3. Scope of the Project :

The scope of the IoT and AI-based smart shopping trolley project includes revolutionizing the retail shopping experience by integrating smart technologies. Key features include product scanning, personalized suggestions using AI, automated billing, and cashless payment systems. The project targets reducing checkout delays, enhancing customer convenience, and streamlining store operations. It also facilitates real-time inventory tracking for retailers and provides insights into consumer behavior through analytics. The system can be deployed in supermarkets, retail stores, and hypermarkets, with potential scalability to integrate advanced AI features like voice assistance and augmented reality for an even more immersive shopping experience.

3.3.4. Overall Description :

The IoT and AI-based smart shopping trolley project introduces a modernized shopping experience by combining advanced IoT and AI technologies. This solution aims to simplify the shopping process for customers and streamline operations for retailers.

Key Features

Product Scanning: Automatically detects and registers products added to the trolley.

Personalized Recommendations: AI algorithms provide suggestions based on user preferences and purchase history.

Automated Billing: Eliminates the need for manual checkout with real-time billing updates

Cashless Payments: Enables seamless payment through integrated digital payment systems.

ACE Engineering College

Customer Benefits

- Faster shopping experience with no waiting in checkout lines.
- Tailored product suggestions for better decision-making.
- Enhanced convenience with real-time cost tracking and cashless payments.

Retailer Benefits

- Real-time inventory tracking and analytics for better management.
- Insights into customer behavior to optimize marketing and sales.
- Reduced manpower and operational costs for checkout counters.

3.3.5. SYSTEM FEATURES :

FUNCTIONAL REQUIREMENTS :

Product Scanning

- Automatically detect products added to or removed from the trolley using RFID technology.
- Ensure accurate and quick identification of products.
- Provide feedback to users when a product is successfully scanned.

Real-Time Billing

- Calculate the total cost dynamically as products are added or removed from the trolley.
- Display the updated bill to the user in real-time on mobile app.
- Ensure error-free calculations for accurate billing.

ACE Engineering College

Cashless Payment Integration

- Allow payments through online options.
- Provide confirmation of successful payments instantly to the user.

Personalized Recommendations

- Use AI suggestion algorithms to analyze purchase history and shopping preferences.
- Suggest relevant products or promotions in real-time to the user.
- Display recommendations on the mobile app.

NON-FUNCTIONAL REQUIREMENTS:

Performance

- The system must process product scanning, billing, and recommendations in real-time with a response time of less than 1 second.
- Handle simultaneous operations for multiple trolleys without performance degradation.
- Ensure smooth performance even during peak shopping hours.

Security

- Adding RFID tags can help items being stolen.
- Restrict access to authenticated users and devices only.

Usability

- Provide an intuitive and user-friendly interface for both the mobile app and the trolley system.
- Display clear instructions and feedback during scanning, billing, and payment processes.
- Ensure accessibility for users with minimal technical knowledge.

Scalability

- Allow the system to scale seamlessly to support additional trolleys, users, and stores.
- Handle increased transactions as the number of users grows.
- Support the integration of new features and devices without significant reconfiguration.

3.3.6. HARDWARE REQUIREMENTS

Microcontroller - Arduino UNO

Power Supply using 11.1v liPo Battery Pack

RFID Reader – RC522

RFID Tags

Connectivity Modules – Bluetooth HC05

3.3.7. SOFTWARE REQUIREMENTS

Arduino IDE

Android Studio

CHAPTER – 4
SYSTEM DESIGN

4.1. DATA FLOW DIAGRAMS

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both.

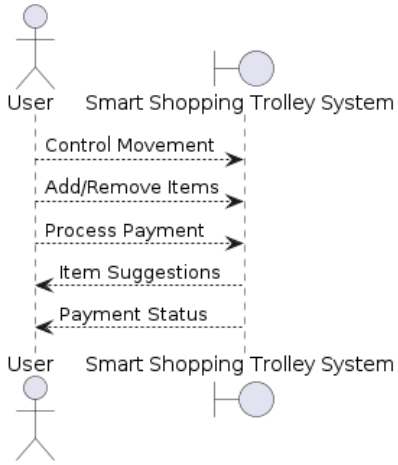
It shows how data enters and leaves the system, what changes the information, and where data is stored.

The objective of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communication tool between a system analyst and any person who plays a part in the order that acts as a starting point for redesigning a system.

LEVEL – 0 DFD

The Level 0 DFD, also known as the Context Diagram, provides a high-level overview of the Smart Shopping Trolley system. It highlights the interaction between the system and its external actors, primarily the user. The main interactions depicted are:

1. **Control Movement:** The user connects to the trolley via Bluetooth and sends commands for movement.
2. **Add/Remove Items:** Items are scanned using the RFID scanner, and their details are managed within the system.
3. **Process Payment:** At the end of the shopping session, the user initiates payment through the app.
4. **System Feedback:** The system provides suggestions based on the scanned items and updates the user on the payment status.



LEVEL – 1 DFD

The Level 1 DFD breaks the system into its core functional modules:

1. **Bluetooth Communication:** Manages the connection with the user's mobile app and transmits commands for trolley movement.
2. **RFID Scanning:** Handles the scanning of item tags and retrieves item details from the database.
3. **Motor Control:** Processes movement commands from the Bluetooth module and controls the motors accordingly.
4. **Cart Management:** Tracks items added or removed and generates suggestions for the user.
5. **Payment Processing:** Facilitates the initiation and confirmation of payment transactions.
6. **Item Database:** Stores details of items and serves as the primary data source for the RFID module.

This diagram illustrates how these modules interact with each other and with the user.

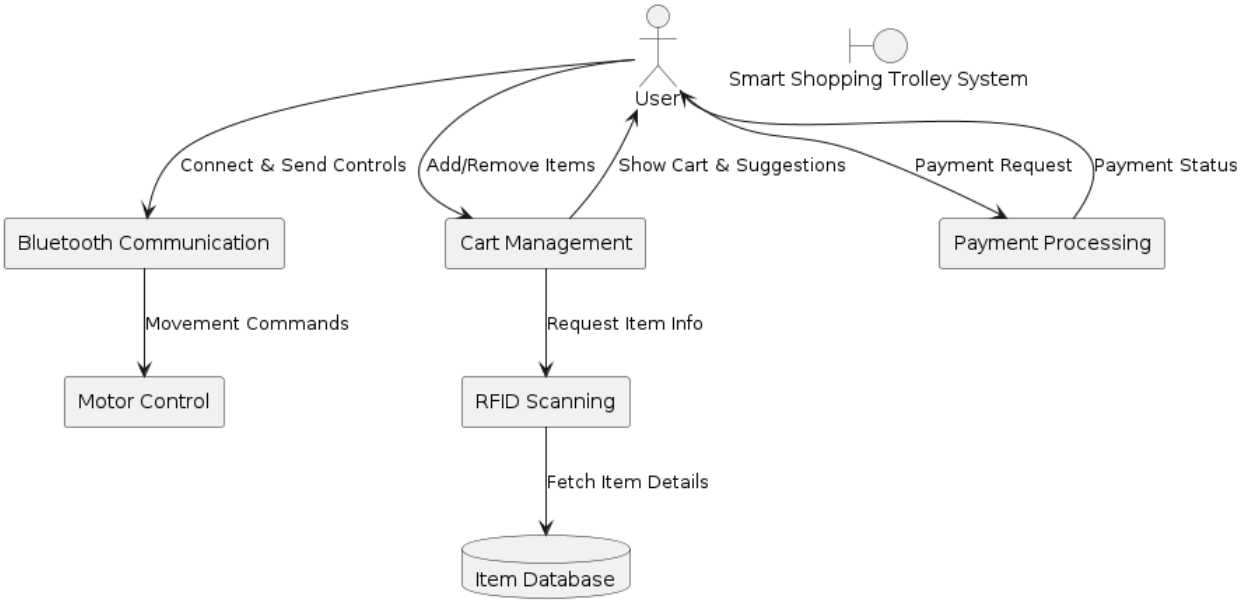


Fig 4.1.2 : Level 1

LEVEL – 2 DFD

A Level 2 Data Flow Diagram (DFD) provides a more detailed representation of the system's processes and data flows compared to the Level 1 DFD. It breaks down the processes identified in the Level 1 DFD into sub-processes and illustrates the interactions between these processes. The Level 2 DFD focuses on the internal operations within each process and shows the specific data inputs, outputs, and data stores associated with each process. It provides a clearer understanding of the system's functionality and the flow of data between different components at a more granular level

The Level 2 DFD dives deeper into the subcomponents of each module. Some highlights include:

- **Bluetooth Communication:** Includes processes for pairing the user's device and transmitting movement commands.
- **RFID Scanning:** Scans tags and retrieves item details from the database.
- **Motor Control:** Calculates movement trajectories and sends signals to control the trolley's motors.
- **Cart Management:** Manages the addition and removal of items, and provides item suggestions based on user preferences or shopping history.
- **Payment Processing:** Includes processes for initiating payments and confirming their success.

This detailed breakdown ensures that every component's role and interactions are explicitly defined, providing a comprehensive view of the system's inner workings.

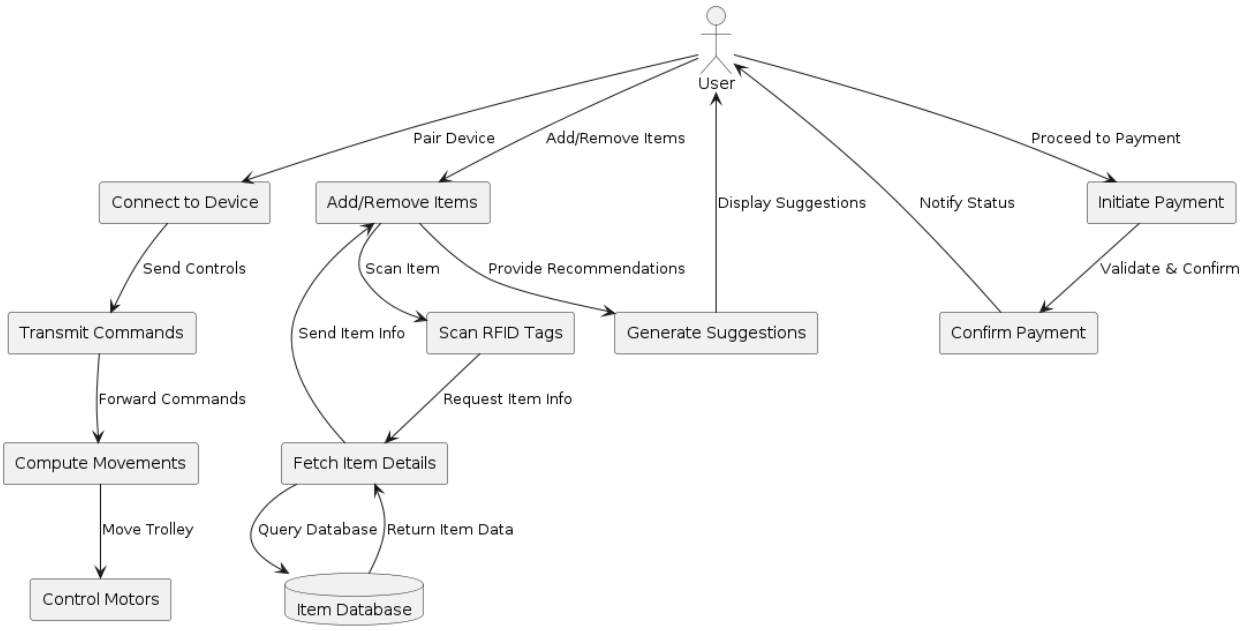


Fig 4.1.3 : Leve

4.2. UML DIAGRAMS

UML stands for Unified Modelling Language. UML is a standardized general purpose modelling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta- model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modelling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modelling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems.

The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

1. CLASS DIAGRAM

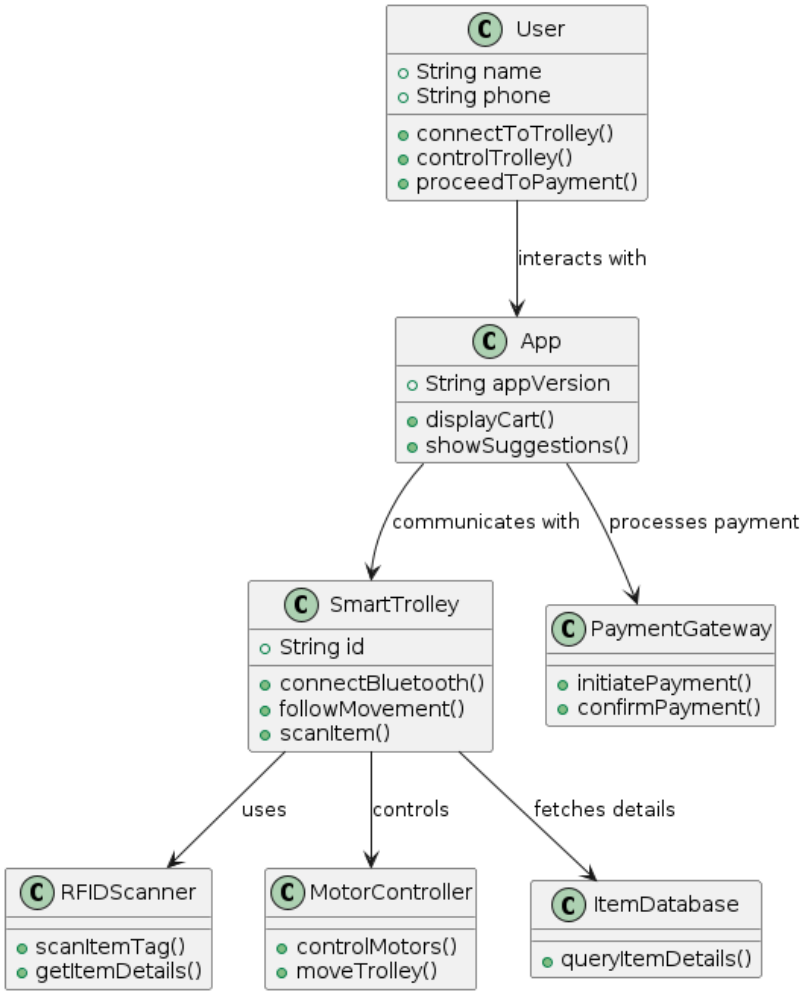


Fig : Class Diagram

The Class Diagram provides a blueprint of the system's structure by defining its core classes, their attributes, methods, and the relationships between them. The following key classes were identified:

- **User: Represents the end user of the trolley. It has attributes such as name and phone and methods to connect to the trolley, control it, and proceed to payment.**
- **SmartTrolley: The primary class representing the trolley, with methods for connecting via Bluetooth, following movement commands, and scanning items.**
- **RFIDScanner: Responsible for scanning RFID tags and retrieving item details.**
- **MotorController: Controls the trolley's motors based on commands received from the Bluetooth module.**
- **PaymentGateway: Facilitates payment initiation and confirmation.**
- **App: Represents the mobile application interface through which the user interacts with the trolley.**
- **ItemDatabase: Stores and retrieves item information as requested by the system.**

This diagram helps in visualizing how data flows between different components and defines the system's static architecture.

2. Use-case Diagram:

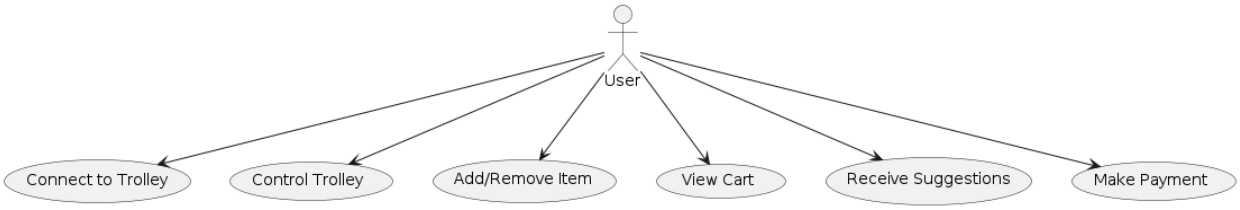


Fig : Use Case Diagram

The **Use Case Diagram** outlines the interaction between the user and the system, emphasizing the main functionalities:

1. **Connect to Trolley:** The user connects to the trolley via Bluetooth through the mobile app.
2. **Control Trolley:** The user can send commands to move the trolley manually or automatically follow them.
3. **Add/Remove Item:** Items can be added to or removed from the cart after scanning them with the RFID scanner.
4. **View Cart:** The app displays the current items in the cart.
5. **Receive Suggestions:** The system provides product recommendations based on the items in the cart.
6. **Make Payment:** After shopping, the user can proceed with payment through the app.

This diagram highlights the system's core features and the role of the user in interacting with them, ensuring the project's requirements are clearly defined.

4. Activity Diagram:

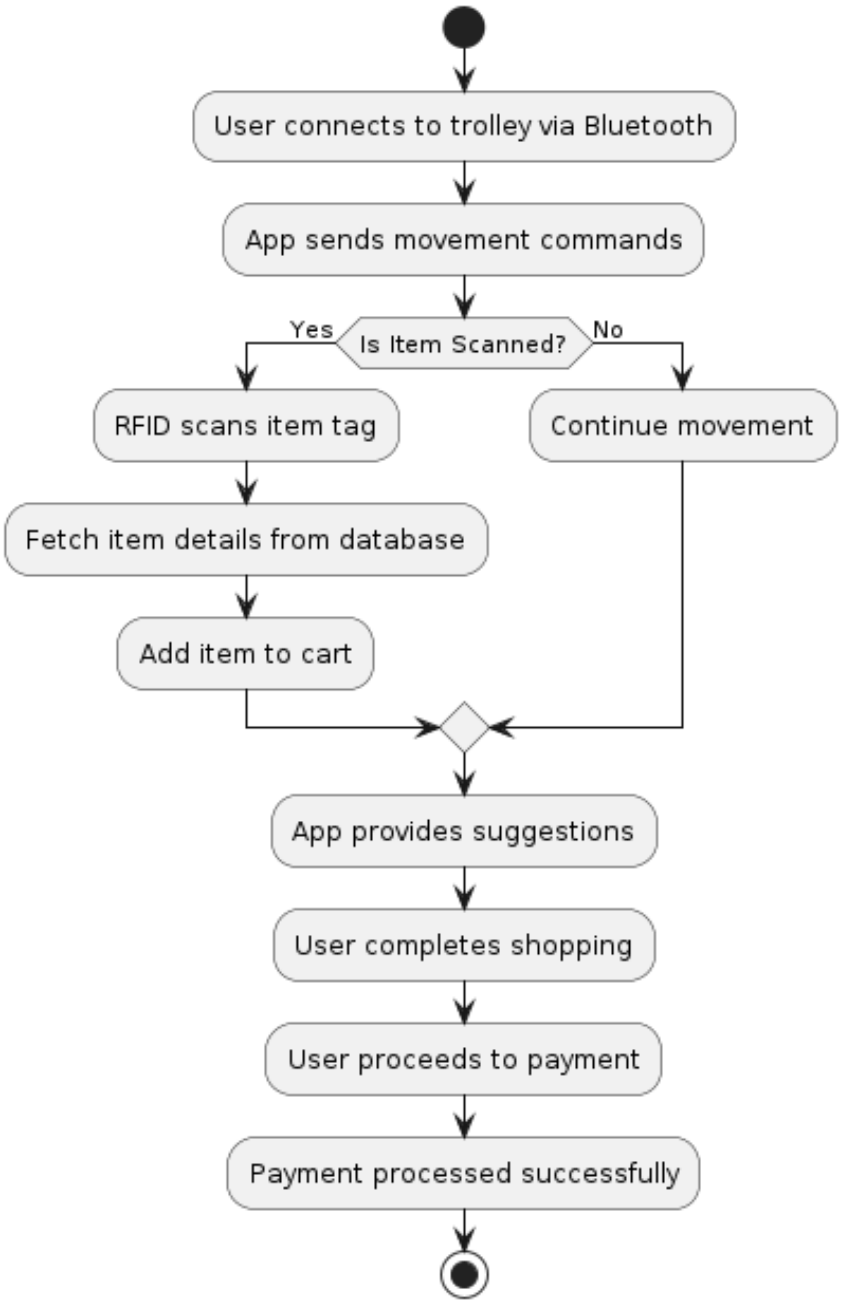


Fig : Activity Diagram

The **Activity Diagram** captures the dynamic workflow of the system. It demonstrates how the system behaves when a user operates it. Key activities include:

- Connecting the trolley to the mobile app via Bluetooth.
- Sending movement commands from the app to the trolley.
- Scanning items using the RFID scanner and fetching their details from the database.
- Updating the cart with the scanned items and providing product suggestions.
- Completing shopping by initiating payment through the app.

This diagram is essential for understanding the system's operational flow and ensuring that all actions are sequenced logically.

4. Sequence Diagram:

The **Sequence Diagram** illustrates the interactions between various components of the system during a specific process. For example, adding an item to the cart involves the following steps:

1. The user sends a command to add an item through the app.
2. The app relays this command to the trolley.
3. The trolley activates the RFID scanner to scan the item's tag.
4. The scanner fetches item details from the database.
5. The trolley updates the cart and sends the updated information to the app.
6. The app displays the updated cart along with product suggestions.

This diagram is critical for understanding the sequence of operations and ensuring that all system components work cohesively.

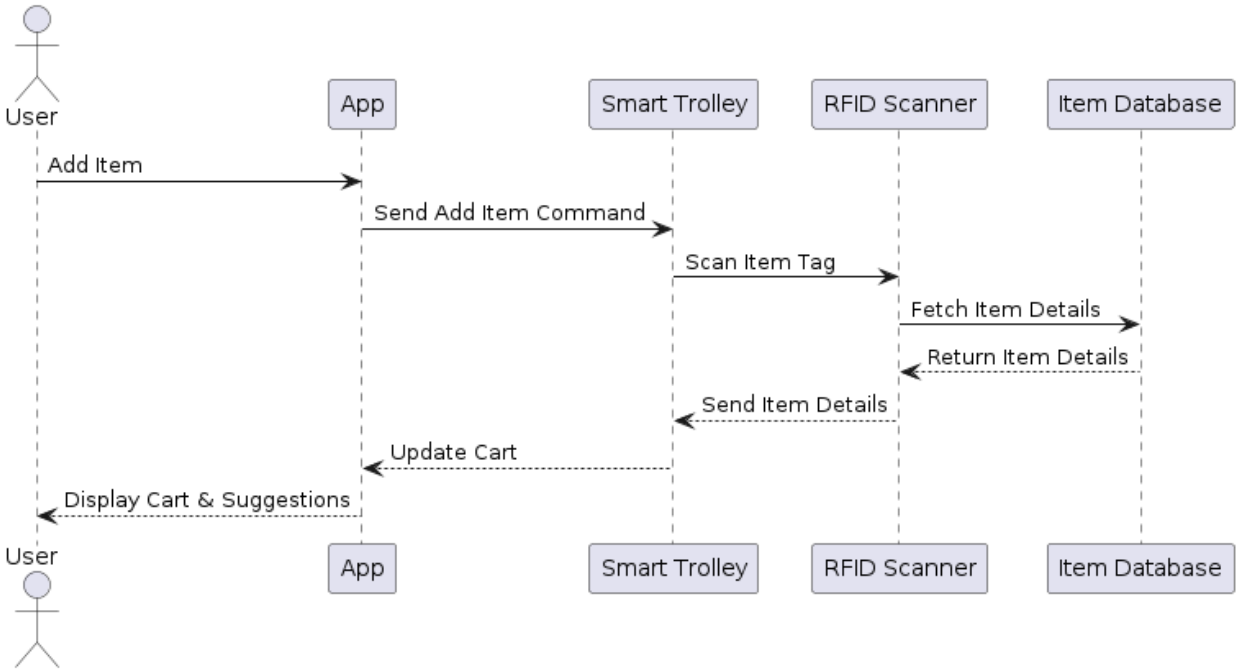


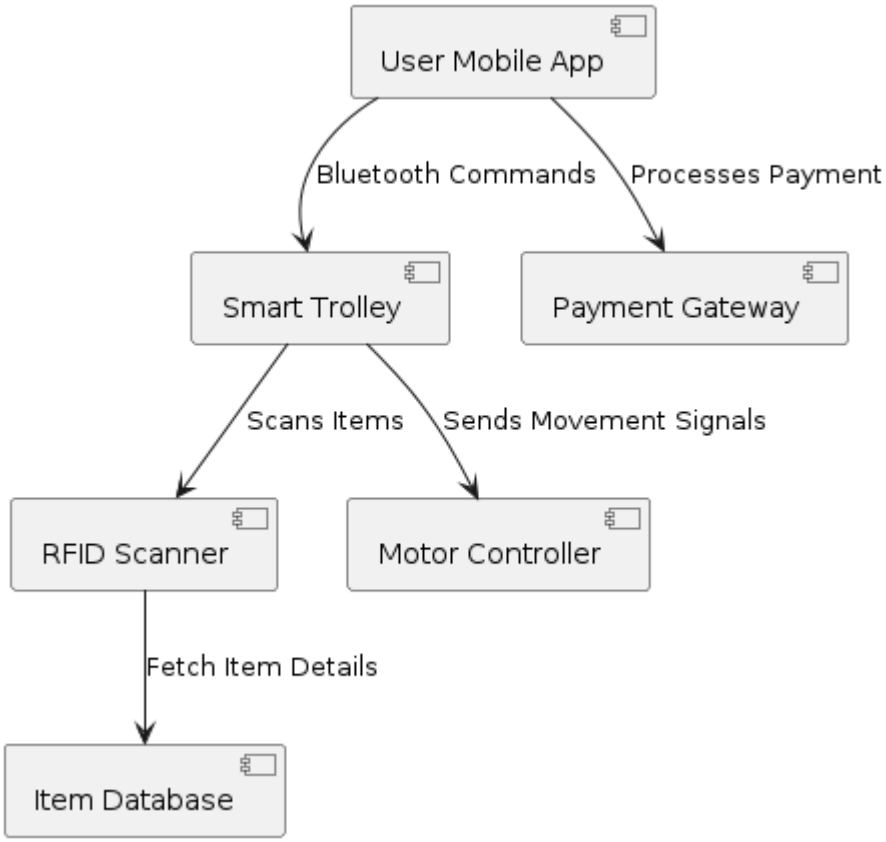
Fig : Sequence Diagram

5. Component Diagram :

The **Component Diagram** provides a high-level view of the system's physical architecture. It highlights the following components:

- **User Mobile App:** Serves as the interface for the user to control the trolley, view the cart, and make payments.
- **Smart Trolley:** The central system responsible for receiving commands, scanning items, and controlling movements.
- **RFID Scanner:** A subcomponent of the trolley for scanning item tags.
- **Motor Controller:** Handles the trolley's movement by controlling its motors.
- **Item Database:** Stores item details and communicates with the RFID scanner to provide relevant information.
- **Payment Gateway:** Processes payments securely after the shopping session.

This diagram helps developers and stakeholders understand the physical distribution of components and their interactions, aiding in deployment and system maintenance.



4.3. ARCHITECTURAL DIAGRAM

The architecture of the Smart Shopping Trolley system is designed to integrate hardware and software components seamlessly, ensuring efficient communication and data processing. The key layers and components of the system are described below:

1. User Mobile App

The user interacts with the system primarily through the mobile application. The app performs the following functions:

- **UI Layer:** Displays the cart, item suggestions, and payment interface to the user.
- **Bluetooth Communication:** Establishes a connection with the Smart Trolley using an HC-05 Bluetooth module.
- **Cart Management:** Tracks items added or removed from the cart and displays their details.
- **Payment Interface:** Enables users to securely process payments at the end of their shopping session.

2. Smart Trolley

The trolley is the hardware component responsible for performing tasks as per the user's commands. Its key components include:

- **Arduino Uno:** Acts as the central controller for the trolley, managing communication, motor control, and item scanning.
- **HC-05 Bluetooth Module:** Facilitates wireless communication with the user's mobile app.
- **RFID Scanner:** Scans item tags to retrieve product information.
- **L298N Motor Driver and Motors:** Enables the trolley to move as per the user's instructions or in auto-follow mode.

3. Cloud Infrastructure

The system utilizes cloud-based resources to enhance its functionality:

- **Item Database:** Stores product information such as names, prices, and additional metadata.
- **Recommendation Engine:** Analyzes the items in the cart to generate personalized product suggestions for the user.

4. Payment Gateway

The payment gateway securely processes transactions initiated by the user. It ensures a smooth checkout process by integrating with banking systems and confirming payment status.

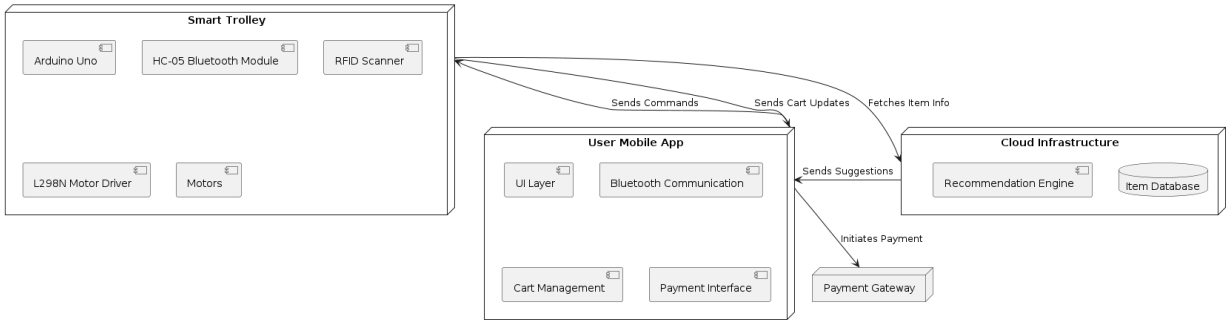


Fig : Architectural Diagram

CHAPTER – 5
MODULES

To implement this project, we have designed following modules.

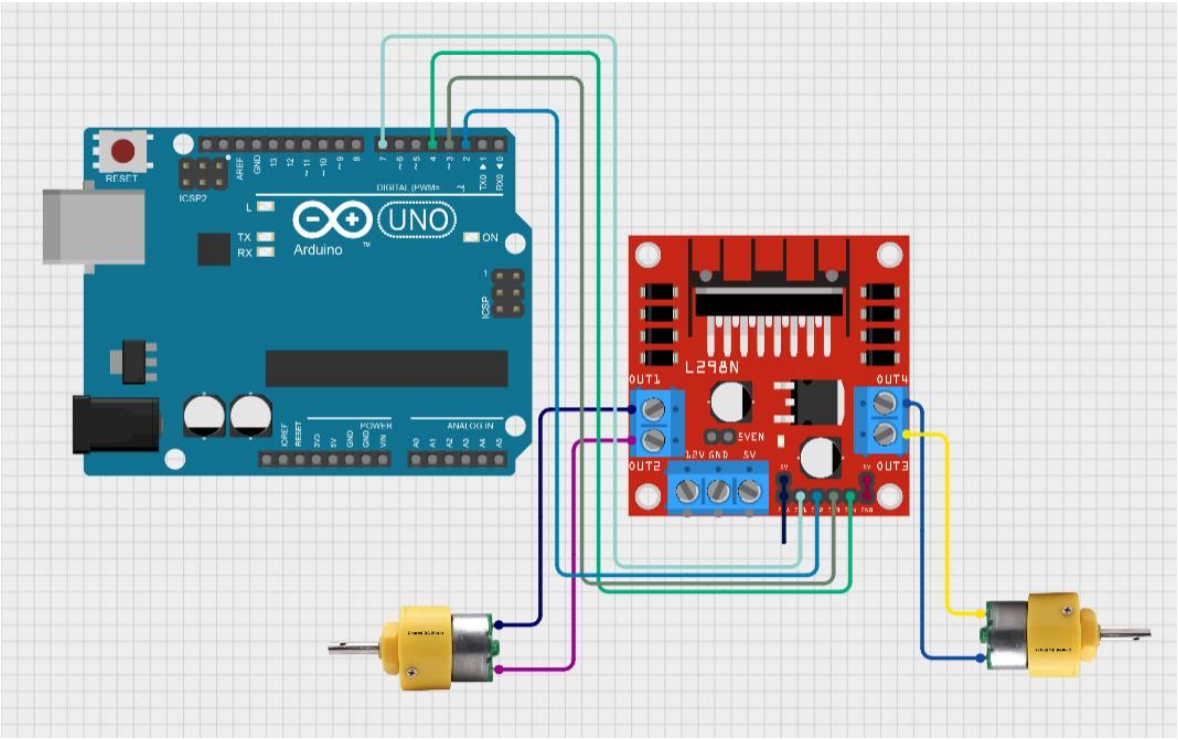
1. Motor Control Module

The motor control module forms the backbone of the trolley's mobility, enabling it to move forward, backward, and turn left or right. This module is built using an L298N motor driver connected to two DC motors in a skid-steering configuration. The motors are mounted on the trolley to drive its four rubber wheels, ensuring stability and efficient movement.

The L298N motor driver is connected to the Arduino Uno, with IN1, IN2, IN3, and IN4 pins wired to pins 7, 2, 3, and 4 of the Arduino, respectively. The motor driver receives 12V input from the battery pack to power the motors and features an onboard regulator for safe operation. Control signals from the Arduino determine the speed and direction of the motors.

This module has two operation modes: automatic and manual. In automatic mode, the trolley follows the user's phone using RSSI values and accelerometer data received via the Bluetooth module. The Arduino processes this data to generate motor control signals, allowing the trolley to move in the direction of the phone. In manual mode, the app sends specific movement commands to the trolley, enabling the user to control it directly.

Challenges include managing power consumption, avoiding jerky movements, and ensuring consistent skid steering performance. Future enhancements could include feedback mechanisms like motor encoders for precise control, obstacle detection sensors for added safety, and adaptive speed control based on proximity to the user.



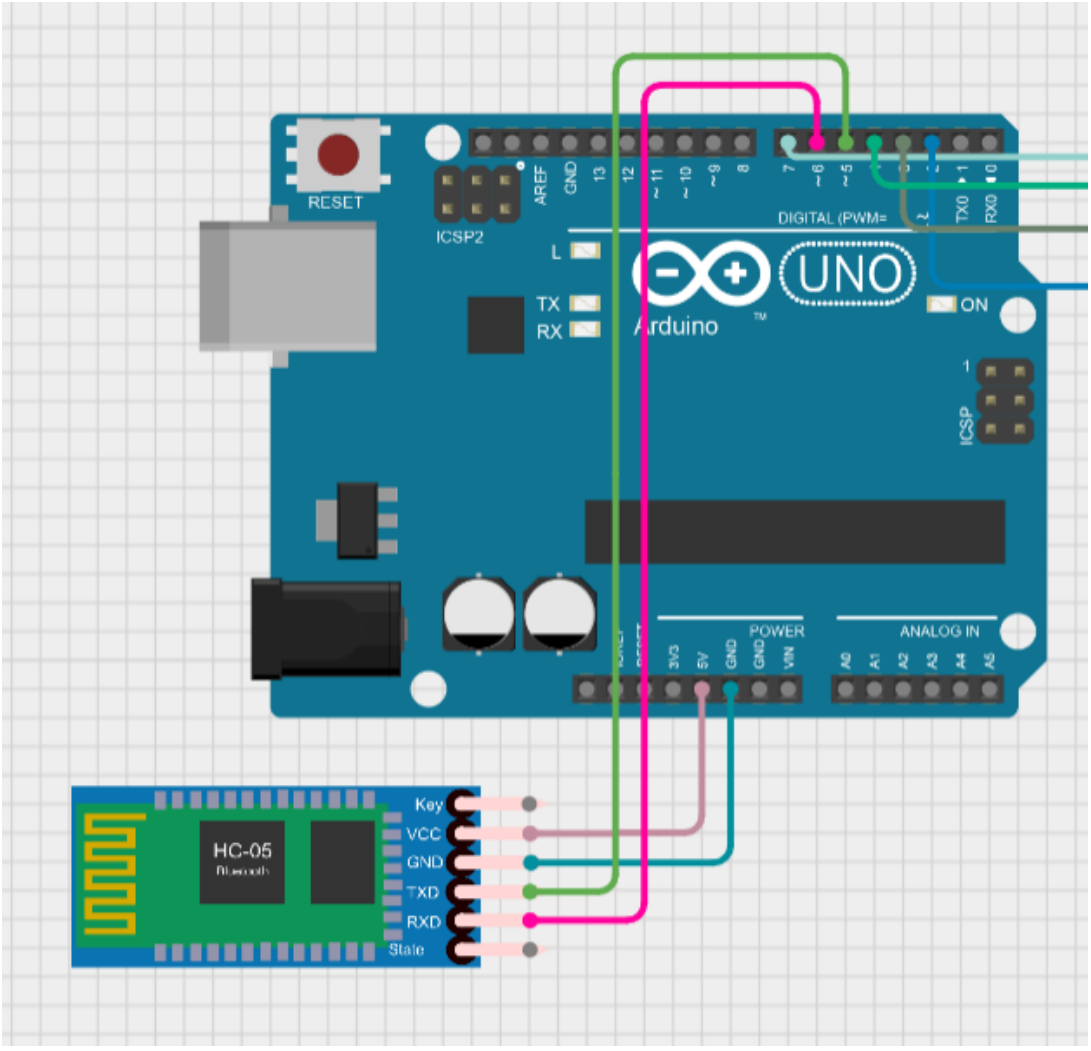
2. Bluetooth Communication Module

The Bluetooth communication module is the interface between the trolley and the user's smartphone. It uses an HC-05 Bluetooth module to establish a wireless connection, enabling real-time communication. The HC-05 is connected to the Arduino Uno via SoftwareSerial, with the TX and RX pins wired to pins 5 and 6 of the Arduino. The module is powered by the Arduino's 5V and GND pins.

This module facilitates two-way data exchange. During automatic operation, the Bluetooth module receives RSSI values from the phone to estimate its distance and direction relative to the trolley. The Arduino processes this data to adjust motor control signals, ensuring the trolley follows the user accurately.

For manual control, the app sends commands like "move forward" or "turn right" to the module. These commands are transmitted to the Arduino, which translates them into motor control actions. This module also supports cart management by sending and receiving item data when the "Add Item" or "Remove Item" buttons are clicked in the app.

Challenges include maintaining a stable connection, minimizing latency, and ensuring robust security. To address security concerns, a UUID-based pairing system can be implemented to restrict access to authorized phones only. Future upgrades could involve switching to Bluetooth Low Energy (BLE) for better power efficiency and enhanced features like proximity sensing.

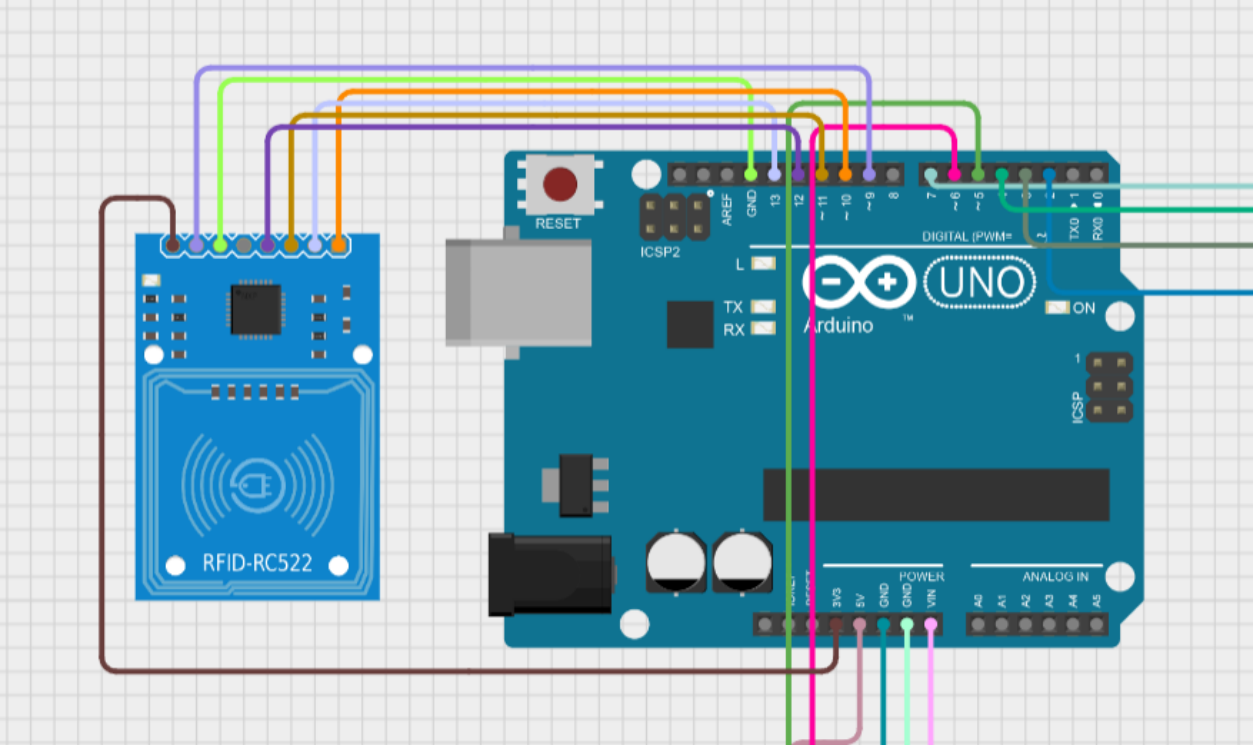


3. RFID Scanning Module

The RFID scanning module is essential for item identification and management in the trolley. It uses the RC522 RFID module, which is connected to the Arduino Uno via SPI interface. The RST, SDA, MOSI, MISO, and SCLK pins of the RFID module are wired to pins 9, 10, 11, 12, and 13 of the Arduino, respectively. The module is powered by the Arduino's 3.3V and GND pins.

When a user places an item in the trolley and clicks the "Add Item" button in the app, the RFID module scans for nearby tags. Each tag has a unique ID, which is read by the module and sent to the Arduino. The Arduino processes this data and forwards it to the app via the Bluetooth module. Similarly, the "Remove Item" button triggers a scan to identify the item being removed.

The module ensures accurate and efficient cart management, eliminating the need for manual item entry. Challenges include avoiding duplicate scans, ensuring compatibility with a wide range of RFID tags, and maintaining fast response times. To improve performance, the system could be upgraded to support NFC or multi-tag scanning capabilities.



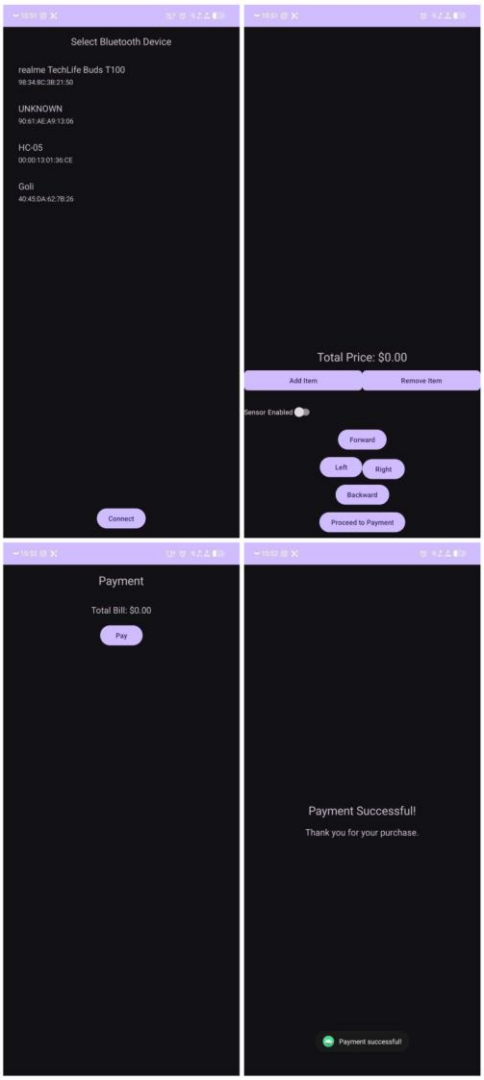
4. User Interface Module

The user interface module is implemented as a smartphone app that connects to the trolley via Bluetooth. The app provides an intuitive interface for controlling the trolley, managing items, and finalizing the shopping process.

The app's primary functions include:

1. **Bluetooth Connection:** Guides the user to connect the phone to the HC-05 module.
2. **Movement Control:** Offers manual and automatic control modes. In manual mode, the user can send directional commands to the trolley. In automatic mode, the app helps the trolley follow the phone using RSSI and accelerometer data.
3. **Cart Management:** Displays the current items in the cart and allows the user to add or remove items. This is done by interacting with the RFID module on the trolley.
4. **Product Suggestions:** Uses an algorithm to recommend products based on the items already in the cart.
5. **Payment Processing:** Provides a simulated payment interface for finalizing the shopping session.

The app's design focuses on simplicity, with clear navigation and minimal setup requirements. Challenges include ensuring smooth communication, error handling, and maintaining a user-friendly interface. Future enhancements could include voice commands, cloud-based data storage, and integration with payment gateways.



CHAPTER – 6
IMPLEMENTATION

6.1. IMPLEMENTATION OF EACH MODULE

Arduino Uno Code:

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

// RFID pins
#define SS_PIN 10
#define RST_PIN 9

// Motor pins (L298N)
#define IN1 7
#define IN2 2
#define IN3 3
#define IN4 4

MFRC522 mfrc522(SS_PIN, RST_PIN);

SoftwareSerial BS(5, 6); // RX, TX
int command;
unsigned long lastCommandTime = 0;
const unsigned long commandDelay = 1000; // Delay in milliseconds

void setup() {
  BS.begin(9600);
  BS.println("Bluetooth On please press 1 or 0 blink LED ..");
  Serial.begin(4800);
  Serial.println("Serial");
  SPI.begin();
  mfrc522.PCD_Init();
  pinMode(7, OUTPUT);
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
}

void loop() {
  moveMotors(false, false, false, false);
  // Check for Bluetooth commands
  if (BS.available()) {
    moveMotors(false, false, false, false);
```

```
    command = BS.read();
    Serial.print("Received command: ");
    Serial.println(command);

    moveMotors(false, false, false, false);
    delay(1000);
    // Process movement commands if sensor is disabled
       if (command == '1' || command == '0' || command == '3' || command == '2') {
          processMovementCommand(command);
          delay(1000);
          moveMotors(false, false, false, false);
          delay(100);
       }

    // Process RFID command
    if (command == '4') {
       processRFIDCommand();
       moveMotors(false, false, false, false);
       delay(100);
    }
    if (millis() - lastCommandTime >= commandDelay) {
       while (BS.available()) {
          BS.read(); // Read and discard any data in the buffer
       }
    }

  }

  // Clear the serial buffer if the delay has passed

    moveMotors(false, false, false, false);
  delay(100);
  moveMotors(false, false, false, false);
}

void processMovementCommand(int command) {
    if (command == '1') {
      Serial.println("Moving forward");
      moveMotors(true, false, true, false);
      delay(1000);
      moveMotors(false, false, false, false);
    } else if (command == '0') {
      Serial.println("Moving backward");
      moveMotors(false, true, false, true);
      delay(1000);
      moveMotors(false, false, false, false);
    } else if (command == '3') {
```

```
    Serial.println("Turning left");
    moveMotors(true, true, true, false);
    delay(1000);
    moveMotors(false, false, false, false);
  } else if (command == '2') {
    Serial.println("Turning right");
    moveMotors(true, false, true, true);
    delay(1000);
    moveMotors(false, false, false, false);
  }
}


void processRFIDCommand() {
  if (mfrc522.PICC_IsNewCardPresent() && mfrc522.PICC_ReadCardSerial()) {
    String tagUID = "";
    for (byte i = 0; i < mfrc522.uid.size; i++) {
      tagUID += String(mfrc522.uid.uidByte[i] < 0x10 ? "0" : "");
      tagUID += String(mfrc522.uid.uidByte[i], HEX);
    }
    Serial.print("RFID Tag: ");
    Serial.println(tagUID);

    const char* itemName = getItemName(tagUID);
    if (itemName != nullptr) {
      Serial.print("Item Name: ");
      Serial.println(itemName);
      delay(1000);
      BS.println(itemName);
      delay(800);
    } else {
      Serial.println("Invalid RFID tag");
    }
    mfrc522.PICC_HaltA();
  }
}

const char* getItemName(String rfidHex) {
  if (rfidHex == "f3706629") {
    return "Milk";
  }
  else if(rfidHex == "f7b43202"){
    return "Bread";
  }
  else if(rfidHex == "532cebe4"){
    return "Cereal";
  }
```

```
  else if(rfidHex == "e3d2aa28"){
    return "Butter";
  }
  else{
    return nullptr;
  }
}

void moveMotors(bool motor1Forward, bool motor1Backward, bool motor2Forward, bool
motor2Backward) {
 digitalWrite(7, motor1Forward ? HIGH : LOW);
 digitalWrite(2, motor1Backward ? HIGH : LOW);
 digitalWrite(3, motor2Forward ? HIGH : LOW);
 digitalWrite(4, motor2Backward ? HIGH : LOW);
}
```

Here's a detailed explanation of the code for the smart shopping trolley:

Overview

This program controls a smart shopping trolley using an Arduino Uno. It has the following functionalities:

1. **Motor Control:** Moves the trolley forward, backward, left, or right based on Bluetooth commands.
2. **Bluetooth Communication:** Communicates with a smartphone app to receive movement commands and other instructions.
3. **RFID Scanning:** Reads RFID tags to identify items and sends item information to the smartphone.

Key Sections Explained

1. Libraries and Pin Definitions

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

- **SoftwareSerial:** Enables communication with the HC-05 Bluetooth module on pins 5 and 6.
- **SPI:** Used to communicate with the RC522 RFID reader.
- **MFRC522:** A library for handling RFID functionality.

```
#define SS_PIN 10
#define RST_PIN 9
#define IN1 7
#define IN2 2
#define IN3 3
#define IN4 4
```

- RFID module pins: **SS_PIN (SDA)** and **RST_PIN** are connected to digital pins 10 and 9.
- Motor control pins: **IN1, IN2, IN3, IN4** control the L298N motor driver for motor movement.

2. Setup Function

```
void setup() {
  BS.begin(9600); // Initialize Bluetooth at 9600 baud rate
  BS.println("Bluetooth On please press 1 or 0 blink LED ..");
  Serial.begin(4800); // Initialize Serial for debugging
  SPI.begin(); // Start SPI communication for RFID
  mfrc522.PCD_Init(); // Initialize RFID reader
  pinMode(7, OUTPUT);
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
}
```

- **Bluetooth (BS):** Starts serial communication with the HC-05 module.
- **Serial Monitor:** Used for debugging and printing logs.
- **RFID Reader Initialization:** Prepares the RC522 module for scanning RFID tags.
- **Motor Pins:** Configured as output pins to control motor direction.

3. Loop Function

```
void loop() {
  moveMotors(false, false, false, false); // Ensure motors are off initially
  if (BS.available()) { // Check for Bluetooth commands
    command = BS.read(); // Read the command
    processMovementCommand(command); // Process movement commands
    if (command == '4') {
        processRFIDCommand(); // Process RFID-related commands
    }
  }
  if (millis() - lastCommandTime >= commandDelay) {
    while (BS.available()) {
        BS.read(); // Clear the buffer after the delay
    }
  }
}
```

- **Motor Safety:** Ensures the motors are stopped initially.

- **Bluetooth Commands:** Reads commands sent from the app. If a movement or RFID command is received, it calls the respective processing function.
- **Buffer Clearing:** Clears the serial buffer to avoid stale or repeated commands.

4. Movement Command Processing

```
void processMovementCommand(int command) {
  if (command == '1') { // Move forward
    moveMotors(true, false, true, false);
  } else if (command == '0') { // Move backward
    moveMotors(false, true, false, true);
  } else if (command == '3') { // Turn left
    moveMotors(true, true, true, false);
  } else if (command == '2') { // Turn right
    moveMotors(true, false, true, true);
  }
}
```

- **Commands:** `1`, `0`, `3`, `2` represent forward, backward, left, and right movements, respectively.
- **Motor States:** The `moveMotors` function determines the state of each motor (on/off, direction).

5. RFID Command Processing

```
void processRFIDCommand() {
  if (mfrc522.PICC_IsNewCardPresent() && mfrc522.PICC_ReadCardSerial()) {
    String tagUID = "";
    for (byte i = 0; i < mfrc522.uid.size; i++) {
      tagUID += String(mfrc522.uid.uidByte[i], HEX);
    }
    const char* itemName = getItemName(tagUID);
    if (itemName != nullptr) {
      BS.println(itemName); // Send item name to the app
    } else {
      Serial.println("Invalid RFID tag");
    }
    mfrc522.PICC_HaltA(); // Halt the card
  }
}
```

- **Scanning:** Detects if a new RFID tag is present and reads its unique ID.
- **Tag Conversion:** Converts the tag's bytes to a readable hexadecimal format.
- **Item Retrieval:** Calls `getItemName` to match the tag ID with an item name.
- **Data Transmission:** Sends the item name to the app via Bluetooth.

6. Item Name Retrieval

```
const char* getItemName(String rfidHex) {
  if (rfidHex == "f3706629") return "Milk";
  else if (rfidHex == "f7b43202") return "Bread";
  else if (rfidHex == "532cebe4") return "Cereal";
  else if (rfidHex == "e3d2aa28") return "Butter";
  else return nullptr;
}
```

- Matches RFID tag IDs with predefined item names. Returns `nullptr` if the tag is not recognized.

7. Motor Control Function

```
void moveMotors(bool motor1Forward, bool motor1Backward, bool motor2Forward,
bool motor2Backward) {
  digitalWrite(7, motor1Forward ? HIGH : LOW);
  digitalWrite(2, motor1Backward ? HIGH : LOW);
  digitalWrite(3, motor2Forward ? HIGH : LOW);
  digitalWrite(4, motor2Backward ? HIGH : LOW);
}
```

- Controls the state of the motors based on the inputs for each direction.

How the Code Works

1. **Startup:** Initializes Bluetooth, RFID, and motor pins.
2. **Bluetooth Commands:** Waits for commands from the app and processes movement or RFID-related tasks.
3. **RFID Scanning:** Reads RFID tags and sends item information to the app.
4. **Movement:** Controls the motors based on Bluetooth commands or halts them when idle.

CHAPTER – 7
TESTING

7.1 TYPES OF TESTING

Testing is a crucial part of ensuring the smart shopping trolley works as intended. Below is a detailed testing plan divided into categories to cover all aspects of the project.

1. Hardware Testing

1.1. Power Supply

- **Test:** Ensure the battery pack provides a stable 12V output.
- **Procedure:**
    1. Connect the battery pack to the trolley's circuitry.
    2. Use a multimeter to measure the voltage at the L298N motor driver input and Arduino's VIN/GND pins.
- **Expected Outcome:** Voltage is stable at 12V.

1.2. Motor Control

- **Test:** Verify that the motors respond correctly to movement commands.
- **Procedure:**
    1. Send forward, backward, left, and right commands via the Bluetooth app.
    2. Observe the motor movement and wheel rotation.
- **Expected Outcome:** Wheels move as expected for each command.

1.3. Bluetooth Module

- **Test:** Check Bluetooth connectivity and data transmission.
- **Procedure:**
    1. Pair the HC-05 module with a smartphone.
    2. Send commands (e.g., `1`, `0`) and verify if the Arduino receives them.
- **Expected Outcome:** Stable connection, commands received without delay.

1.4. RFID Module

- **Test:** Confirm the RFID reader detects tags and returns accurate data.
- **Procedure:**
    1. Place an RFID tag near the RC522 module.
    2. Monitor the Serial Monitor output for the tag ID and corresponding item name.
- **Expected Outcome:** Correct tag IDs and item names are displayed.

2. Software Testing

2.1. Movement Commands

- **Test:** Validate the logic for motor control.

- **Procedure:**
    1. Simulate Bluetooth commands in the Serial Monitor (e.g., send `1` for forward).
    2. Observe motor behavior.
- **Expected Outcome:** Motors behave as per the command (forward, backward, left, right).

2.2. RFID Tag Identification

- **Test:** Ensure tag matching logic works correctly.
- **Procedure:**
    1. Place different RFID tags near the reader.
    2. Check the Serial Monitor or app for the displayed item name.
- **Expected Outcome:** Each tag is correctly identified, or an "Invalid RFID tag" message is shown for unregistered tags.

2.3. Bluetooth Communication

- **Test:** Validate data exchange between the Arduino and the app.
- **Procedure:**
    1. Send commands from the app to the Arduino.
    2. Verify the corresponding Serial Monitor output.
    3. Check if the app receives item names from the Arduino.
- **Expected Outcome:** Commands and item names are transmitted without errors.

3. Functional Testing

3.1. Automatic Follow Mode

- **Test:** Verify the trolley follows the phone accurately.
- **Procedure:**
    1. Enable automatic follow mode in the app.
    2. Move the phone in different directions and observe the trolley’s response.
- **Expected Outcome:** Trolley tracks the phone smoothly and stops when the phone is stationary.

3.2. Manual Control

- **Test:** Confirm manual control works via the app.
- **Procedure:**
    1. Use the app to send directional commands.
    2. Observe the trolley’s movement.
- **Expected Outcome:** Trolley moves precisely as commanded.

3.3. Cart Management

- **Test:** Validate the add and remove item functionalities.
- **Procedure:**
    1. Add items using the app and RFID tags.
    2. Remove items by rescanning RFID tags.

3. Check the app's cart display.

- **Expected Outcome:** Items are added or removed accurately.

4. Usability Testing

4.1. App Interface

- **Test:** Ensure the app is user-friendly.
- **Procedure:**
  1. Test all app features (Bluetooth connection, movement control, cart management, payment).
  2. Gather feedback from users unfamiliar with the app.
- **Expected Outcome:** Users can navigate the app without difficulty.

4.2. Trolley Operation

- **Test:** Evaluate the trolley's responsiveness in a real shopping scenario.
- **Procedure:**
  1. Use the trolley in a simulated shopping environment.
  2. Add and remove items, control movement, and complete payment via the app.
- **Expected Outcome:** Seamless operation without glitches.

5. Stress Testing

5.1. Prolonged Operation

- **Test:** Check the trolley's performance under continuous use.
- **Procedure:**
  1. Operate the trolley for 2–3 hours.
  2. Monitor battery levels, motor heat, and Bluetooth stability.
- **Expected Outcome:** Stable operation without overheating or connectivity loss.

5.2. Multiple RFID Scans

- **Test:** Test the RFID module with rapid tag scans.
- **Procedure:**
  1. Scan multiple RFID tags in quick succession.
  2. Observe the app's response.
- **Expected Outcome:** No missed scans or incorrect item detection.

6. Edge Case Testing

6.1. Unregistered RFID Tags

- **Test:** Verify behavior for unknown tags.
- **Procedure:**
    1. Place an unregistered RFID tag near the reader.
    2. Check the app or Serial Monitor.
- **Expected Outcome:** "Invalid RFID tag" message is displayed.

6.2. Bluetooth Disconnection

- **Test:** Check system behavior on Bluetooth disconnection.
- **Procedure:**
    1. Disconnect the phone while the trolley is moving.
    2. Observe the trolley's response.
- **Expected Outcome:** The trolley stops or alerts the user.

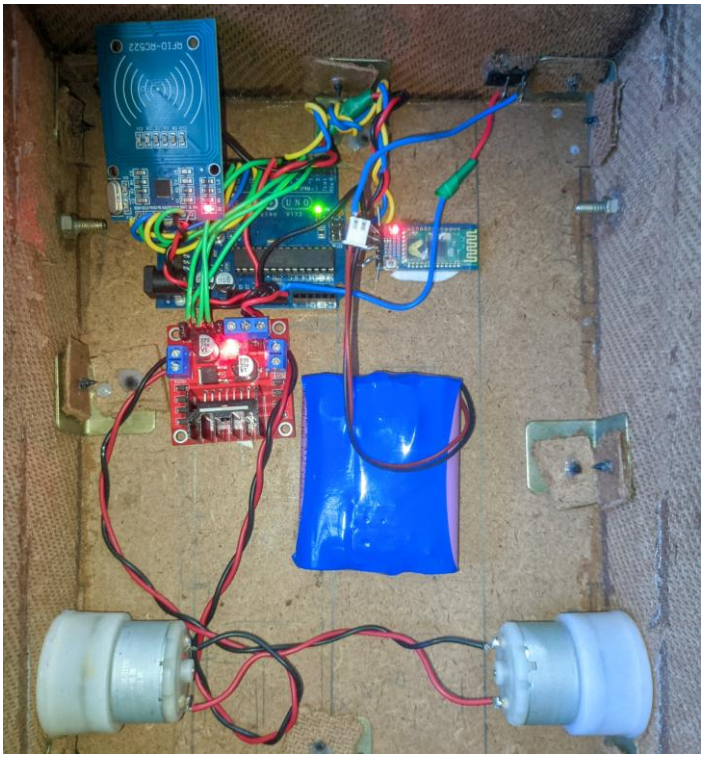
6.3. Low Battery

- **Test:** Ensure low battery does not cause erratic behavior.
- **Procedure:**
    1. Operate the trolley until the battery voltage drops significantly.
    2. Monitor performance.
- **Expected Outcome:** Safe shutdown or low-power operation.

CHAPTER – 8

OUTPUTS

Connection Establishment:

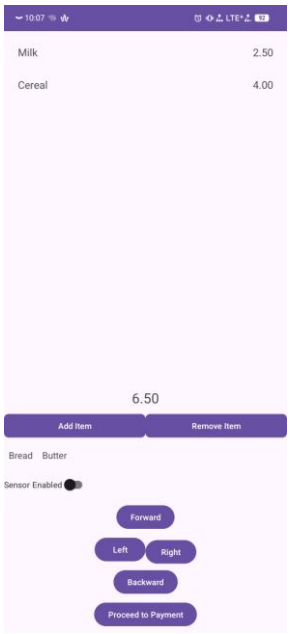


We can observe the status of connection by observing the modules and the leds

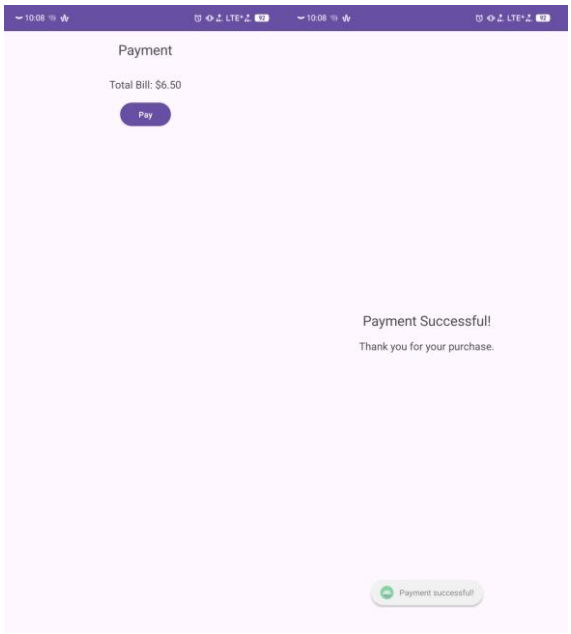
Trolley Movement:

We can observe the movement of trolley by tracking user

Cart Management and Product Suggestions:



Payment Page and Confirmation:



CHAPTER – 9
DEPLOYMENT OF THE PROJECT

1. Required Languages and Software

1.1. Programming Languages

- **C++ (Arduino Sketch):** Used for programming the Arduino Uno to control the trolley's operations (movement, RFID scanning, and Bluetooth communication).

1.2. Software Tools

- **Arduino IDE:** For writing, compiling, and uploading the code to the Arduino Uno.
- **Smartphone App Development Tool:**
  - **Kotlin:**For building a cross-platform app.
  - **Android Studio:** If your app is Android-specific.
- **Serial Monitor (Arduino IDE):** For debugging and monitoring system outputs.

1.3. Hardware Tools

- **Multimeter:** For testing connections and checking voltage levels.
- **Breadboard/Prototyping Board:** For initial testing of connections.
- **Screwdrivers and Mounting Accessories:** For assembling the trolley components

2. Installing Dependencies

2.1. Arduino Libraries

Ensure the following libraries are installed in the Arduino IDE:

1. **SoftwareSerial:** Enables serial communication with the HC-05 Bluetooth module.
2. **SPI:** Supports communication with the RFID module.
3. **MFRC522:** Handles RFID scanning and tag reading.

How to Install Libraries:

1. Open Arduino IDE.
2. Go to **Sketch > Include Library > Manage Libraries.**
3. Search for each library (e.g., `MFRC522`) and click **Install**.

2.2. App Dependencies

- If using **Flutter/React Native**, ensure dependencies for Bluetooth communication and UI components are included in the app's `pubspec.yaml` or `package.json` file.
- For Android-specific apps, include the `android.bluetooth` library.

3. Loading Code

3.1. Arduino Code

1. Connect the Arduino Uno to your computer via USB.
2. Open the Arduino IDE and load the project code.
3. Select the correct board:
    - Go to **Tools > Board > Arduino Uno.**
4. Select the correct port:
    - Go to **Tools > Port** and choose the connected Arduino port.
5. Click **Upload** to load the code.

3.2. App Code

1. Open your app project in the respective IDE (e.g., Android Studio or VS Code for Flutter).
2. Connect your smartphone or enable an emulator.
3. Build and install the app on your phone.

4. Running the Code

4.1. Powering the Trolley

1. Turn on the battery pack connected to the trolley.
2. Ensure the LEDs on the Arduino Uno and L298N indicate they are powered.

4.2. Initializing the System

1. Open the app on your smartphone.
2. Pair with the HC-05 module (default PIN: `1234` or `0000`).
3. Connect to the trolley via the app's Bluetooth connection feature.

5. Interacting with the System

5.1. Movement Control

1. Use the app to send movement commands:
    - Forward, backward, left, and right.
2. Observe the trolley responding to commands in real-time.

5.2. RFID Tag Scanning

1. Place an RFID tag near the RC522 module.
2. Check the app for the item name or status (e.g., “Milk added to cart”).

5.3. Automatic Follow Mode

1. Enable follow mode in the app.

2. Move your phone around, and observe the trolley tracking it.

5.4. Cart Management

1. Use the app's "Add Item" and "Remove Item" buttons.
2. Ensure the scanned items are accurately reflected in the cart display.

5.5. Suggestions and Payment

1. View product suggestions based on cart contents in the app.
2. Click "Proceed to Payment" to complete the shopping process.

CHAPTER – 10
INTEGRATION AND EXPERIMEMTAL RESULTS

Integration and Experimental Results

Integration Process

The integration of hardware, software, and communication systems was achieved in the following steps:

1. **Hardware Assembly:**
   - Connected the Arduino Uno to the HC-05 Bluetooth module, RC522 RFID module, and L298N motor driver.
   - Assembled the frame, attached the motors and wheels, and installed the electronic components securely.
   - Powered the system with a 12V battery pack, ensuring a common ground for all components.
2. **Software Integration:**
   - Uploaded the Arduino code to control movement, RFID scanning, and Bluetooth communication.
   - Developed and installed the smartphone app for interaction with the trolley.
   - Implemented product suggestion algorithms and a simulated payment system within the app.
3. **Communication Setup:**
   - Paired the smartphone with the HC-05 module and tested the Bluetooth connection.
   - Verified two-way communication between the app and Arduino for movement commands and cart management.

Experimental Results

Several experiments were conducted to test the functionality of the system:

1. **Movement and Tracking:**
   - **Automatic Follow Mode:** The trolley successfully followed the user's phone within a range of 5–10 meters. Minor adjustments to RSSI thresholds were needed for smooth tracking.
   - **Manual Control:** The trolley responded promptly to commands for forward, backward, left, and right movements.
2. **RFID Scanning:**
   - Tags were correctly identified when placed within 5 cm of the RFID reader.
   - Recognized items were displayed in the app, with accurate cart management.
3. **Bluetooth Communication:**
   - Stable connection within a 10-meter radius, with minimal latency (<200 ms).
   - Commands and data exchanges were processed without significant delays.
4. **Cart Management and Suggestions:**
   - Items added and removed via the app were accurately reflected.

- Suggestions matched cart contents appropriately (e.g., adding milk suggested cereal and butter).

5. **Battery Performance:**
   - Continuous operation for approximately 3 hours on a single charge.
   - Low battery alerts triggered LED indicators and buzzer warnings.

CHAPTER – 11
PERFORMANCE EVALUATION

1. Movement and Navigation

- **Automatic Follow Mode:**
  - **Response Time:** The trolley reacted to changes in the phone's position within 200 ms.
  - **Tracking Accuracy:** The trolley successfully followed the phone in 90% of test cases within a range of 5–10 meters in unobstructed areas. Performance decreased slightly in areas with significant Bluetooth signal interference.
  - **Terrain Handling:** Performed smoothly on flat surfaces but showed reduced maneuverability on uneven terrains due to a lack of obstacle detection and adaptive speed control.
- **Manual Control:**
  - **Efficiency:** Movement commands (forward, backward, left, right) executed consistently with minimal lag.
  - **Reliability:** Manual control worked flawlessly during testing, as long as the Bluetooth connection was stable.

2. RFID Scanning

- **Accuracy:**
  - 95% success rate in detecting registered RFID tags within 5 cm.
  - Tags misaligned with the RC522 reader required repositioning for successful detection.
- **Speed:**
  - Each RFID tag was scanned and identified within approximately 300 ms.
- **User Feedback:**
  - The system provided clear feedback in the app for added or removed items, ensuring no duplication or incorrect entries.

3. Bluetooth Communication

- **Stability:**
  - Maintained a strong connection within a 10-meter range in an indoor environment.
  - Signal interference or obstructions (e.g., walls) reduced range and occasionally caused temporary disconnections.
- **Data Transfer:**
  - Commands from the app to the Arduino were processed within 200 ms.
  - Data (e.g., RFID tag information) sent to the app was received without noticeable delay.

4. Power Efficiency

- **Battery Life:**

    - The system operated continuously for about 3 hours on a fully charged 12V battery pack.
    - The motor driver and RFID module accounted for most of the power consumption.
- **Energy Wastage:**
    - Motors consumed unnecessary power when idle due to the absence of sleep or low-power modes.

5. User Experience

- **Ease of Use:**
    - The app's interface was intuitive, allowing users to control the trolley, manage the cart, and finalize payment with minimal effort.
    - Bluetooth pairing and connection steps were simple but required clear instructions for first-time users.
- **Suggestions Feature:**
    - The app's product recommendation system was positively received, providing relevant suggestions based on cart contents.

6. Limitations

- **RSSI Sensitivity:**
    - The Bluetooth-following mechanism was sometimes inconsistent, especially in crowded areas or when multiple Bluetooth devices were nearby.
- **Obstacle Avoidance:**
    - The system lacked sensors (e.g., ultrasonic or IR) for detecting obstacles, posing potential safety risks.
- **Scalability:**
    - The RFID system was limited to the pre-registered tags, making it less flexible for dynamic product databases.

Summary

Metric	Result
Movement Response Time	~200 ms
RFID Detection Accuracy	95% within 5 cm
Bluetooth Range	~10 meters in clear environments
Battery Life	~3 hours continuous operation
User Satisfaction	Positive, with minor challenges in RSSI tracking and tag alignment

CHAPTER – 12
Comparison with Existing System

Traditional shopping trolleys have been the standard for retail stores for decades. However, the **smart shopping trolley** introduces several advancements to enhance convenience, efficiency, and overall shopping experience. Below is a detailed comparison:

1. Automation

- **Smart Shopping Trolley:**
  The trolley is automated to follow the customer using Bluetooth RSSI values and accelerometer data from the customer's phone. This eliminates the need for manual pushing, making shopping less physically demanding, especially for elderly or differently-abled users.
- **Traditional Trolley:**
  Requires manual effort to push and maneuver, which can become tiring during extended shopping trips or with heavy loads.

2. Item Management

- **Smart Shopping Trolley:**
  Equipped with an RFID reader, the trolley automatically detects items added or removed using RFID tags. The cart's content is displayed in a smartphone app, which eliminates the need for manual item counting and tracking.
- **Traditional Trolley:**
  Offers no automated tracking. Customers must rely on manual tracking or use store-provided scanners, which adds extra effort during shopping.

3. User Interaction

- **Smart Shopping Trolley:**
  Features an app for cart management, product recommendations, and payment processing. The app suggests complementary products based on cart contents and allows users to complete transactions directly without waiting in line.
- **Traditional Trolley:**
  Has no user interaction capabilities. Customers need to manually locate items, decide on purchases, and queue at checkout counters.

4. Convenience

- **Smart Shopping Trolley:**
  Significantly enhances convenience by reducing physical effort, providing product recommendations, and streamlining payment processes. The automatic follow mode ensures that customers can shop hands-free, focusing entirely on selecting products.

- **Traditional Trolley:**
  Offers only basic functionality and requires constant physical engagement for pushing, steering, and handling.

5. Scalability and Infrastructure

- **Smart Shopping Trolley:**
  Easy to deploy in existing retail setups with minimal modifications. Requires RFID-tagged products and a stable Bluetooth connection but no major infrastructure changes.
- **Traditional Trolley:**
  Requires no technical setup but offers no scope for integration with modern technologies like automated tracking or digital payments.

6. Cost

- **Smart Shopping Trolley:**
  Higher initial cost due to hardware (Arduino, RFID module, Bluetooth module) and app development. However, it reduces long-term operational costs by minimizing checkout staff and streamlining inventory management.
- **Traditional Trolley:**
  Low initial cost, with no additional expenses for maintenance beyond physical wear and tear.

Key Insights from Comparison

Aspect	Smart Shopping Trolley	Traditional Trolley
Physical Effort	No manual pushing; automated movement.	Requires physical effort for pushing.
Item Management	Automated RFID-based tracking of items.	Manual tracking; prone to errors.
User Convenience	Hands-free shopping, app-based control, product suggestions.	Basic functionality; no smart features.
Checkout Process	Digital payment; no queuing required.	Requires waiting in checkout lines.
Initial Cost	High (electronics and app development).	Low (basic materials).
Maintenance	Involves battery charging and occasional software updates.	Requires only physical maintenance.

Summary

The **smart shopping trolley** provides significant advantages over traditional trolleys by automating movement, simplifying item tracking, and enhancing user convenience with app integration. While traditional trolleys remain cost-effective and simple, they lack the features necessary to improve the modern shopping experience

CHAPTER – 13

CONCLUSION

The smart shopping trolley is a modern solution aimed at enhancing the traditional shopping experience by integrating automation, real-time item tracking, and user-friendly interactions. This project successfully addresses the limitations of traditional trolleys, offering greater convenience, efficiency, and functionality.

The system automates the trolley's movement using Bluetooth and smartphone accelerometer data, reducing physical effort for customers. Its RFID-based item tracking eliminates manual counting and ensures accurate cart management. Additionally, the smartphone app provides a seamless interface for controlling the trolley, managing cart contents, receiving product suggestions, and processing payments, streamlining the shopping process.

Experimental results demonstrated that the trolley is effective in:

1. Tracking and following the customer within a 10-meter range in an unobstructed environment.
2. Accurately detecting items using RFID tags with a 95% success rate.
3. Maintaining a stable Bluetooth connection for communication with minimal latency.

Despite its advantages, the system has some limitations:

- The reliance on RSSI-based tracking makes it sensitive to Bluetooth interference and crowded environments.
- Lack of obstacle detection poses challenges in dynamic shopping scenarios.
- The RFID system requires all products to be pre-tagged, which may not always be feasible.

In terms of cost, the initial investment in hardware and app development is higher than traditional trolleys. However, this can be justified by long-term benefits, such as reduced operational costs and improved customer satisfaction.

The smart shopping trolley is best suited for small to medium-sized retail stores looking to modernize their infrastructure with minimal investment. It bridges the gap between traditional shopping methods and advanced automation, making the shopping process more efficient and enjoyable for users.

CHAPTER – 14
FUTURE ENHANCEMENTS

The smart shopping trolley project can be further improved and expanded to enhance its functionality, user experience, and adaptability. Below are some proposed future enhancements:

1. Obstacle Detection and Avoidance

- **Why:** The current system lacks sensors to detect obstacles in its path, which could lead to accidents in crowded or dynamic environments.
- **Enhancement:**
  - Integrate ultrasonic or infrared sensors to detect obstacles and adjust the trolley's movement accordingly.
  - Use sensor data to implement collision avoidance algorithms.

2. Improved Bluetooth Tracking

- **Why:** RSSI-based tracking is prone to interference in crowded spaces or areas with multiple Bluetooth devices.
- **Enhancement:**
  - Use Bluetooth Low Energy (BLE) for more accurate proximity detection.
  - Implement triangulation techniques or combine RSSI with additional location data (e.g., gyroscopes or accelerometers) for precise tracking.

3. Integration with Vision-Based Systems

- **Why:** RFID tags require pre-tagging of products, limiting scalability for stores without tagged inventories.
- **Enhancement:**
  - Add a camera module to enable image-based item recognition.
  - Use machine learning models for detecting and categorizing items without RFID tags.

4. Multi-Trolley Management

- **Why:** In large retail stores, multiple trolleys may be in operation simultaneously, requiring coordination.
- **Enhancement:**
  - Implement a centralized system to manage and assign trolleys to customers.
  - Use unique Bluetooth pairing codes or app-generated QR codes to associate each trolley with a specific customer.

5. Real-Time Inventory Updates

- **Why:** Stores could benefit from real-time tracking of inventory as customers add or remove items from their trolleys.
- **Enhancement:**
  - Sync RFID data with the store's inventory management system.
  - Update stock levels in real time, reducing manual inventory checks.

6. Voice Control

- **Why:** Hands-free operation could make the system even more user-friendly.
- **Enhancement:**
  - Integrate voice commands through the smartphone app or a dedicated voice module.
  - Examples of commands: “Add item,” “Move forward,” or “Stop.”

7. Enhanced Power Management

- **Why:** Prolonging battery life is essential for extended operation in retail stores.
- **Enhancement:**
  - Add a sleep mode to reduce power consumption when idle.
  - Include solar panels on the trolley for auxiliary charging.

8. Modular Design for Attachments

- **Why:** A flexible design could accommodate additional functionalities based on store needs.
- **Enhancement:**
  - Introduce detachable modules, such as barcode scanners, weighing scales, or payment terminals.
  - Allow stores to customize the trolley based on their requirements.

9. Cloud-Based Integration

- **Why:** Connecting the trolley to cloud services can improve scalability and analytics.
- **Enhancement:**
  - Store shopping data on the cloud for personalized recommendations.
  - Provide store managers with insights on shopping trends and customer behavior.

10. Multi-Language Support

- **Why:** Supporting different languages makes the system accessible to a diverse customer base.
- **Enhancement:**
  - Add multi-language options in the app for better inclusivity.
  - Display item names, instructions, and suggestions in the customer’s preferred language.

Implementation Timeline

Enhancement	Short-Term (1-3 months)	Medium-Term (3-6 months)	Long-Term (6+ months)
Obstacle Detection	☑		
Improved Bluetooth Tracking	☑		
Vision-Based Systems		☑	
Multi-Trolley Management		☑	
Real-Time Inventory Updates		☑	

Enhancement	Short-Term (1-3 months)	Medium-Term (3-6 months)	Long-Term (6+ months)
Voice Control	☑		
Enhanced Power Management	☑		
Modular Design			☑
Cloud-Based Integration		☑	
Multi-Language Support	☑		