Smart Car Parking System

Submitted in fulfillment of the requirements for
the degree of

BSc. Information Technology

By

Mr. Omkar Vikas Kachre
TYBSCIT60

Department of Information Technology
From

BHARTIYAVIDYABHAVAN'S

M.M.COLLEGEOFARTS,N.M.INSTITUTEOFSCIENCE

H.R.J.COLLEGEOFCOMMERCE BHAVAN'SCOLLEGE

MUNSHINAGAR,ANDHERIWEST,

MUMBAI-400058

CERTIFICATE

This is to certify that the project entitled "**Smart Car Parking System**" is a bona fide work of **Mr. Omkar Vikas Kachre** submitted to the University of Mumbai in fulfillment of the requirement for the award of the degree of BSc. in Information Technology.

Project Guide :
Ms. Harshala Chaudhri

Head of Department:
MS. Harshala Chaudhri

Principal:
Prof.(Dr.) Z.P. Bhathena

Project Report Approval

The project report entitled **Smart Car Parking System** by **MR. Omkar Vikas Kachre** is approved for the degree of **Bachelor of Science in Information Technology.**

Examiners:

1.....

2.....

Date:

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Omkar Kachre (TYIT60)

Date:

Abstract

The Smart Car Parking System presented in this project harnesses the capabilities of Arduino and RFID technology to establish an advanced parking management solution. Utilizing RFID cards for user authentication, the system offers features such as real-time parking space availability tracking, occupancy monitoring, barrier control, and a user-friendly web application dashboard. The web application serves as a comprehensive platform for user registration, account management, and real-time monitoring, enhancing the efficiency and user experience in parking management.

Key functionalities include RFID card authentication for user access, real-time display of parking space availability, monitoring of parking space occupancy, and control of barriers or gates using relay modules and servo motors. The system also integrates a payment system for paid parking, enhancing its practicality and revenue generation potential. Security measures such as encryption for communication, secure user authentication, and access logs are implemented to ensure the system's robustness against unauthorized access.

Keywords: Arduino, RFID technology, Barrier Control, User Interface, ESP8266

CONTENTS:

Chapter 1 Introduction 9

1.1 Background 9

1.2 Objective 10

1.3 purpose, Scope and Applicability 11

1.3.1 Purpose 11

1.3.2 Scope 12

1.3.3 Applicability 12

1.4 Achievement 13

1.5 Organization of Report 13

Chapter 2 Survey of Technology 14

Chapter 3 Requirements and analysis 15

3.1 Problem Definition 15

3.2 Requirement Specification 16

3.3 Planning And Schedule 17-20

3.4 Hardware/Software Requirement 21

3.5 Preliminary Product Description 22

3.6 Conceptual Model 23

Chapter 4 system Design 24

4.1 Basic Modules 24-27

4.2 Data Design 28-33

4.2.1 Schema Design 34

4.2.2 Data Integrity and Constraints 35

4.3 Procedural Design 36-38

4.3.1 Logic Diagrams 39

4.3.2 Data Structures 40

4.3.3 Algorithm Design 41

4.4 User Interface Design 42

4.5 Security Issues 43

4.6 Test Case Design 44-47

Chapter 5 Implementation and Testing 48

5.1 Implementation and Approaches 48-49

5.2 Coding Details and Code Efficiency 50-54

5.2.1 Code Efficiency 50

5.3 Testing Approach 55

5.3.1 Unit Testing 55

5.3.2 Integration Testing 56

5.3.3 Beta Testing 57

5.3.4 Modifications and Improvements 58

5.5 Test Cases 59

Chapter 6 Results and Discussion 60

6.1 Test Reports 60-63

6.2 User Documentation 64-66

Chapter 7 Conclusions 67

7.1 Implementation and Approaches 67

7.1.1 Coding Details and Code Efficiency 67

7.2 Code Efficiency 67

7.3 Testing Approach 68

References 69

Glossary 69

Chapter 1: Introduction

Traditional parking systems struggle with meeting modern user needs due to manual processes and inefficiencies. There's a demand for innovative solutions integrating technology. Arduino and ESP8266 offer a promising solution by enhancing parking management through seamless connectivity and efficient computing power.

1.1 Background

Traditional parking systems have long grappled with the evolving demands of users, often falling short in terms of convenience and real-time monitoring. Manual processes prevalent in these systems lead to inefficiencies in space utilization and hinder adaptability to modern requirements. As urban populations continue to grow and vehicular traffic intensifies, there arises an urgent need for innovative solutions that can seamlessly integrate technology to enhance the overall parking experience.

In response to these challenges, the integration of Arduino and ESP8266 presents a compelling opportunity to revolutionize parking management. Arduino's versatile microcontroller capabilities provide a solid foundation for system control and data processing, while ESP8266's seamless WiFi connectivity enables real-time communication and remote monitoring. Together, they form a powerful duo capable of addressing the limitations of conventional parking approaches and ushering in a new era of smart parking solutions.

With Arduino and ESP8266 at the forefront, a Smart Car Parking System emerges as a beacon of innovation. This system offers features such as real-time space availability tracking, seamless user authentication, and remote management functionalities, enhancing the efficiency and convenience of parking for both drivers and facility managers. By harnessing the computational power and connectivity prowess of these technologies, urban parking management can undergo a transformative shift, catering to the needs of a rapidly evolving urban landscape.

1.2 Objectives

User Convenience: Implement RFID card authentication to facilitate seamless user access to the parking area, eliminating the need for traditional methods like physical tickets or manual checks.

Real-time Monitoring: Develop a system to track and display real-time information about parking space availability, enabling users to make informed decisions upon arrival.

Occupancy Management: Monitor and manage the occupancy status of each parking space to optimize space utilization and provide accurate information to users.

Barrier Control: Integrate relay modules and, if applicable, servo motors to control barriers or gates for entry and exit, ensuring secure access and exit points.

Web Application Dashboard: Create a user-friendly web application dashboard for user registration, account management, and real-time monitoring, enhancing the overall user experience.

Security Measures: Incorporate encryption for communication, secure user authentication, and access logs to prevent unauthorized access and ensure the integrity of the system.

Admin Panel: Create an admin panel for system administrators to manage the overall system, view logs, and perform maintenance tasks, ensuring efficient system administration.

1.3 Purpose, Scope and Applicability

1.3.1 Purpose

The Smart Parking System utilizing Arduino and ESP8266 RFID technology aims to revolutionize traditional parking management by introducing a modern, efficient, and user-centric approach. By integrating Arduino's microcontroller capabilities and ESP8266's WiFi connectivity, the project seeks to optimize parking space utilization, enhance user convenience, and provide real-time monitoring of parking availability. The primary objective is to address the shortcomings of conventional parking systems, such as manual processes and inefficient space utilization, by leveraging technology to automate parking operations and improve the overall parking experience for users.

Key Points:

Efficiency: The project focuses on streamlining parking processes through automation, reducing the need for manual intervention and optimizing the use of available parking space. By automating entry/exit gate control and providing real-time space availability information, the system aims to improve operational efficiency and minimize congestion in parking facilities.

Convenience: User convenience is prioritized through features like remote access capabilities and RFID-based user authentication, allowing for seamless entry/exit processes and secure parking access. The system also offers user-friendly interfaces for both parking facility managers and drivers, enhancing the overall parking experience.

Modernization: By integrating IoT technology with RFID authentication, the project represents a significant step towards modernizing parking management systems. It aligns with the growing demand for smart city initiatives and sustainable transportation practices, contributing to the development of smarter and more efficient urban infrastructure.

1.3.2 Scope

The scope of the Smart Parking System project encompasses the development of a comprehensive IoT solution for parking management, with a focus on Arduino and ESP8266 integration. The system includes RFID-based user authentication, real-time parking space monitoring, and remote access capabilities. It offers features such as automated entry and exit gate control, space availability display through LCD screens, and data logging for parking usage analysis. The project's versatility allows for scalability and customization to adapt to various parking facility sizes and requirements. Additionally, the system's compatibility with RFID technology enables secure and efficient user identification, ensuring reliable access control and enhanced security measures.

1.3.3 Applicability

The Smart Car Parking System utilizing RFID technology, Arduino and ESP8266 has wide-ranging applicability across various urban and commercial settings, offering innovative solutions to prevalent parking management challenges.

Urban Parking Infrastructure: Applicable to urban environments, where parking spaces are limited and in high demand. The system optimizes space utilization, providing real-time information to drivers and enhancing overall parking efficiency.

Commercial Complexes and Malls: Suitable for commercial complexes and shopping malls to manage large parking areas efficiently. The RFID-based authentication and real-time monitoring contribute to a seamless parking experience for visitors.

Corporate Campuses: Relevant for corporate campuses with designated parking areas. The system ensures controlled access, tracks occupancy, and provides valuable data for managing corporate parking resources effectively.

Residential Communities: Applicable to gated residential communities, ensuring secure and controlled access to parking spaces. Residents benefit from user-friendly features like real-time availability updates and personalized user profiles.

1.4 Achievement

The successful implementation and seamless operation of the Smart Car Parking System stand as primary achievements, showcasing the system's capability to integrate features such as RFID authentication and real-time monitoring effectively. Equally important is the positive reception from users, reflected in high adoption rates and user satisfaction. Notably, if the system optimizes parking space occupancy, leading to more efficient use of parking areas and reduced congestion, it signifies a major achievement in enhancing overall operational efficiency. The robust security measures in place, preventing unauthorized access and ensuring data protection, mark a critical milestone in providing a secure environment. Additionally, if the system delivers accurate, real-time information on parking space availability and generates insightful reports, it contributes significantly to user and administrator benefits. The successful integration of a payment system for paid parking is a noteworthy achievement, adding financial sustainability to the system. The system's adaptability to different environments and potential for expansion is also considered a key success factor.

1.5 Organization Of Report

The report on the Smart Car Parking System is structured to provide a comprehensive understanding of the project. The introduction sets the stage by highlighting the limitations of traditional parking systems and introducing the objectives of the Smart Car Parking System, which employs Arduino, ESP8266 and RFID technology. The background section delves into the rationale for choosing these technologies and their potential impact on addressing challenges in parking management. The methodology section provides insights into the technical approach, hardware, software architecture, and any unique considerations encountered during development. A detailed exploration of system components, including RFID authentication, real-time monitoring, and barrier control, follows in the subsequent section. The report then delves into the web application and user interface, showcasing the user-friendly dashboard and payment system integration. Security measures, including encryption and access logs, are discussed to highlight the system's robustness. Optional features, such as remote monitoring through an IoT platform, are explored for their contributions to system flexibility. The data logging and analytics section outlines the recorded data and potential insights gained. A dedicated segment is allocated to the admin panel, detailing its functionalities for system administrators. The report then presents results, achievements, and challenges faced during the project, offering valuable lessons learned. The conclusion summarizes key findings and achievements, while recommendations suggest potential improvements or future research directions. The report is enriched with references and an appendix containing supplementary materials and technical details for interested readers.

Chapter 2 Survey of Technology

Technologies used are as follows: -

RFID Technology:

RFID (Radio-Frequency Identification) is a fundamental technology that enables secure and contactless user authentication. RFID cards/tags communicate wirelessly with the RFID reader, allowing for efficient and quick access to the parking facility.

Arduino microcontroller:

Arduino serves as the core computing unit, providing the necessary processing power for the smart parking system. It facilitates the integration of various sensors, communication modules, and data processing components.

Web Application Development:

Web development technologies such as HTML, CSS, and JavaScript are employed to create a user-friendly web application dashboard. Frameworks like Flask or Django (for Python) can be used for server-side development, ensuring a responsive and intuitive interface.

IR Sensors:

Occupancy sensors, which can be ultrasonic or infrared sensors, are employed to monitor the status of each parking space. These sensors detect the presence of a vehicle and relay information to the system for real-time occupancy tracking.

Relay Modules and Servo Motors:

Relay modules and servo motors are used for controlling barriers or gates at entry and exit points. These components play a crucial role in managing the flow of vehicles and ensuring secure access.

Chapter 3 Requirement and analysis

3.1 Problem Definition

Urban areas face persistent challenges in managing parking spaces efficiently, resulting in congestion, user frustration, and suboptimal utilization of parking resources. Conventional parking systems often lack the technological sophistication needed to address these issues. The Smart Car Parking System aims to solve the following key problems:

Inefficient Access Control:

Traditional parking systems often rely on manual methods for access control, leading to inefficiencies in managing the entry and exit of vehicles. The lack of automated access control contributes to congestion and delays.

Limited User Convenience:

Users experience inconvenience due to manual ticketing systems and a lack of real-time information on parking space availability. This results in time-consuming searches for parking spaces and contributes to a suboptimal user experience.

Poor Space Utilization:

Many parking facilities struggle with poor space utilization, as there is often no mechanism in place to monitor and optimize the occupancy of individual parking spaces. This inefficiency exacerbates parking shortages in urban environments.

Manual Barrier Control:

Manual control of entry and exit barriers is a common practice, leading to delays and potential safety issues. The absence of automated barrier control systems contributes to inefficiencies in managing the flow of vehicles.

Lack of Real-time Monitoring:

Existing parking systems often lack real-time monitoring capabilities, making it challenging for administrators to promptly respond to changing conditions, such as sudden increases in demand or the occurrence of security incidents.

3.2 Requirement Specification

The requirements for the IoT Smart car parking system can be organized hierarchically. At the top level, it is designed to tackle the inefficiency of a traditional parking system. Functional requirements and non-functional are as follows:

Functional Requirements:

- Authenticate users through RFID cards for parking access.
- Display real-time parking space availability on the web application.
- Monitor and track the real-time occupancy status of each parking space.
- Control barriers or gates using relay modules and servo motors based on user authentication.
- Develop a user-friendly dashboard for account management and real-time monitoring.

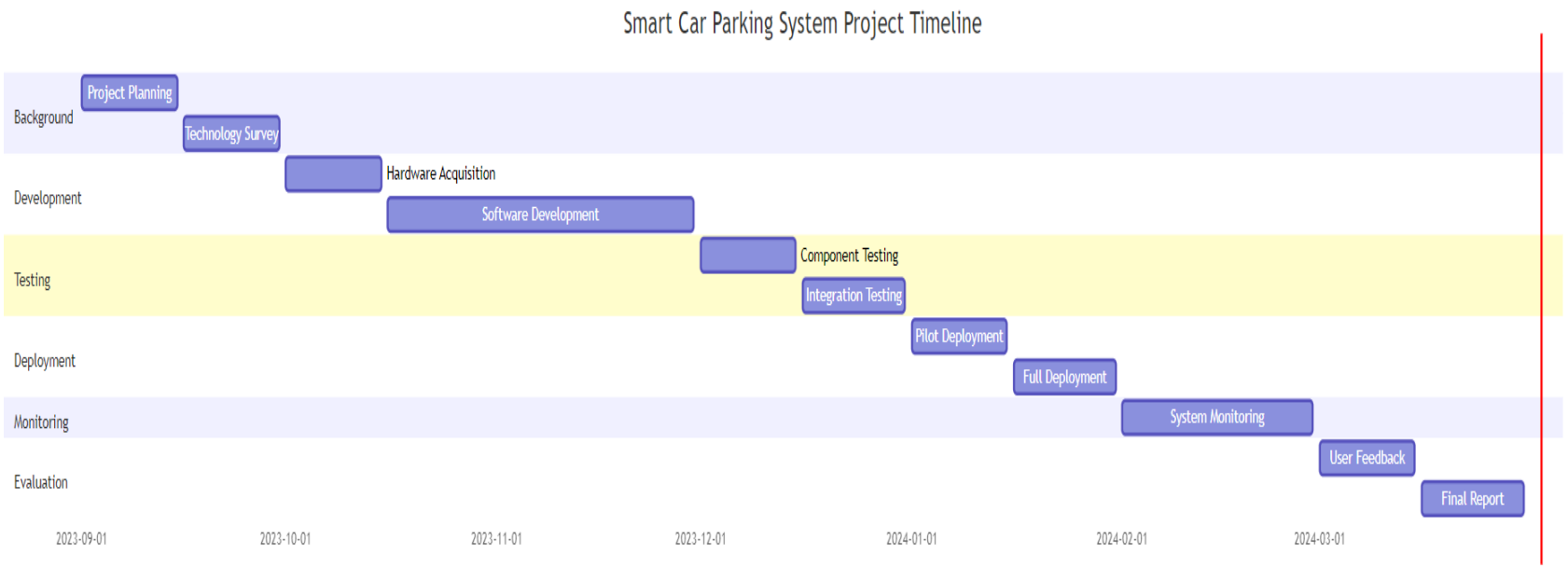
Non-Functional Requirements:

- Ensure optimal system performance during peak usage.
- Design scalable architecture to accommodate future growth.
- Minimize downtime and ensure graceful recovery from failures.
- Create an intuitive and user-friendly interface for easy system navigation.
- Provide 24/7 system availability with scheduled maintenance windows.
- Implement secure data transmission and comply with data protection standards.
- Ensure compatibility with common web browsers and devices.
- Document the codebase and design for ease of maintenance and updates.
- Maintain quick response times for user interactions.

3.3 Planning and scheduling

1) GANTT Chart

A Gantt chart is a powerful project management tool that visually represents a project plan. Traditionally, it consists of two main sections: on the left, a list of tasks is outlined, while on the right, a timeline with schedule bars illustrates the work duration. Alongside tasks, Gantt charts often include start and end dates, milestones, task dependencies, and assignees. To meet the evolving needs of modern software development, advanced roadmap tools such as Jira Software offer additional features like collapsible task structures and resource management panels. These tools empower teams to maintain a cohesive project strategy despite the iterative nature of the software development process.


Smart Car Parking System Project Timeline
Background
Development
Testing
Deployment
Monitoring
Evaluation
Project Planning
Technology Survey
Hardware Acquisition
Software Development
Component Testing
Integration Testing
Pilot Deployment
Full Deployment
System Monitoring
User Feedback
Final Report
2023-09-01
2023-10-01
2023-11-01
2023-12-01
2024-01-01
2024-02-01
2024-03-01

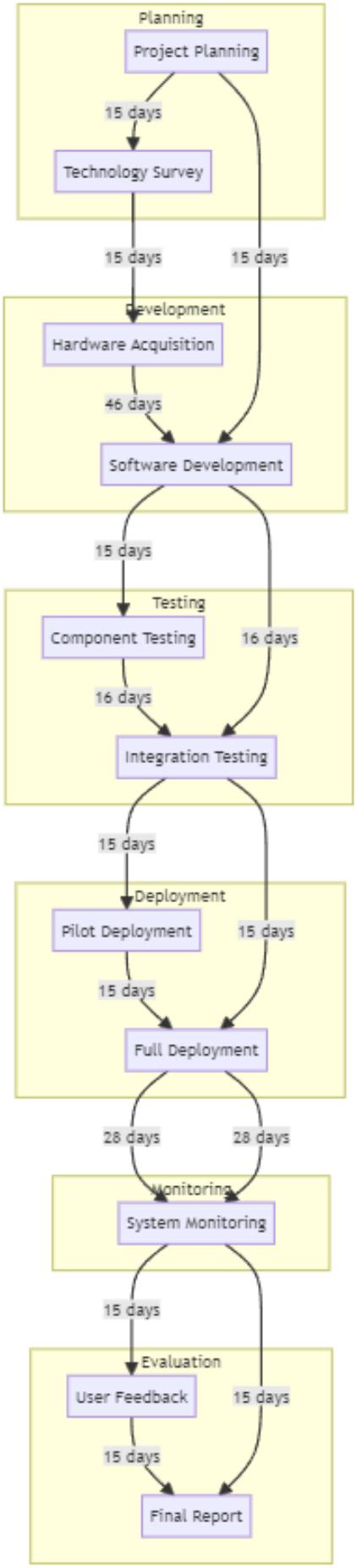
2)PERT Chart (Program Evaluation and Review Technique):

A PERT chart is a network diagram used to map out and schedule the tasks involved in completing a project. It stands for Program Evaluation and Review Technique.

Tasks are represented as nodes or circles, and arrows represent dependencies between tasks. PERT charts often include three time estimates for each task: optimistic, most likely, and pessimistic. These estimates are used to calculate the expected duration of each task.

PERT charts are useful for identifying critical paths in a project, which are the sequences of tasks that must be completed on time for the project to finish as planned.

They are particularly helpful for projects with a high degree of uncertainty or complexity, as they allow project managers to analyze different scenarios and make informed decisions.


Planning
Project Planning
15 days
Technology Survey
15 days
15 days
Development
Hardware Acquisition
46 days
Software Development
15 days
Testing
Component Testing
16 days
16 days
Integration Testing
15 days
Deployment
Pilot Deployment
15 days
15 days
Full Deployment
28 days
28 days
Monitoring
System Monitoring
15 days
Evaluation
User Feedback
15 days
15 days
Final Report

3.4 Hardware/Software requirements

Hardware Requirements:

- Arduino microcontroller
- RFID Reader
- RFID Cards/Tags
- Servo Motors
- IR Sensors
- Power Supply
- Jumper wires
- I2C LCD module
- ESP8266 wifi module

Software Requirements:

- Arduino IDE
- Operating system (e.g., Windows or Mac for development and programming)
- Adafruit IO Integration
- Web-development requirements: -

HTML, CSS, java script, Python

3.5 Preliminary Product Description

The IoT Smart Car Parking System is a solution designed to revolutionize traditional parking management systems. Leveraging the power of **Arduino microcontroller**, RFID technology, and **ESP8266 WIFI module,** this intelligent parking system aims to address common challenges associated with conventional parking infrastructures.

At its core, the system introduces RFID card authentication, allowing users to effortlessly access the parking facility by presenting their RFID cards or tags to the designated reader. This not only enhances the security of the parking space but also streamlines the entry process, providing a seamless and convenient experience for users. To complement this, the system facilitates user registration, enabling individuals to associate their RFID cards with personalized profiles. This user-friendly feature contributes to an efficient onboarding process and further enhances the overall parking experience.

Real-time parking space availability is a key feature provided by the web application dashboard. Users gain access to up-to-the-minute updates on available parking spaces, empowering them to make informed decisions before reaching the parking facility. Concurrently, the system employs occupancy monitoring to track and manage the status of each parking space, ensuring optimal space utilization and addressing the challenges associated with overcrowding and inefficiency.

The integration of barrier control mechanisms, utilizing relay modules and servo motors, adds an extra layer of security and automation. This feature ensures that only authenticated users can access the parking facility, contributing to a secure and controlled environment. The user-friendly web application dashboard provides an intuitive interface for users to manage their accounts, view parking history, and monitor the parking system in real-time, creating a seamless and responsive user experience.

3.6 Conceptual Model

delineates its fundamental elements and their interactions. Users, identified by RFID cards, engage with entry, and exit barriers for secure access to the parking facility. This user-RFID card relationship forms the bedrock of the system's authentication process. Simultaneously, parking spaces, uniquely identified entities, dynamically link with users upon entry, enabling real-time monitoring of occupancy. The web application acts as the primary interface, facilitating user interactions such as registration and monitoring, while administrators utilize an admin panel for holistic system management. Optionally, a payment system integrated into the web application provides users with the flexibility of opting for paid parking services, introducing potential revenue streams. This conceptual model serves as a comprehensive guide, visually portraying the interconnected components and relationships that define the RFID-based IoT Smart Car Parking System.

In essence, the model establishes a clear understanding of how users, RFID cards, parking spaces, the web application, administrators, and an optional payment system collaborate to create a robust and efficient smart parking solution. It sets the stage for the subsequent phases of development and implementation, ensuring a structured and well-defined system.

Chapter 4: System Design

4.1 Basic modules

A basic module in a project refers to a distinct component or subsystem that performs a specific function within the overall system. These modules are designed to be self-contained, facilitating modularity, ease of development, and system scalability. Each basic module typically encapsulates a set of related functionalities and interacts with other modules through well-defined interfaces.

1. **Arduino Microcontroller**: The Arduino microcontroller serves as the central processing unit of the smart parking system, facilitating communication between various components and executing programmed functions.

2. **RFID Reader**: The RFID reader interacts with RFID cards or tags to authenticate vehicles entering or exiting the parking facility. It reads the unique identifiers stored on the RFID cards/tags and communicates this information to the Arduino for processing.

3. **RFID Cards/Tags**: These are small electronic devices containing unique identification information that is read by the RFID reader. Each vehicle accessing the parking facility is equipped with an RFID card/tag for authentication purposes.

4. **Servo Motors**: Servo motors are used to control the opening and closing of entry/exit barriers or gates in the parking system. They receive commands from the Arduino microcontroller to move the barriers accordingly based on vehicle authentication status.

5. **IR Sensors**: Infrared (IR) sensors are deployed at various locations within the parking facility to detect the presence of vehicles in parking spaces. They provide real-time data to the Arduino, allowing for accurate monitoring of parking space occupancy.

6. **Power Supply**: A reliable power supply is essential to ensure the continuous operation of the smart parking system. It provides the necessary electrical power to all components, including the Arduino, RFID reader, servo motors, and sensors.

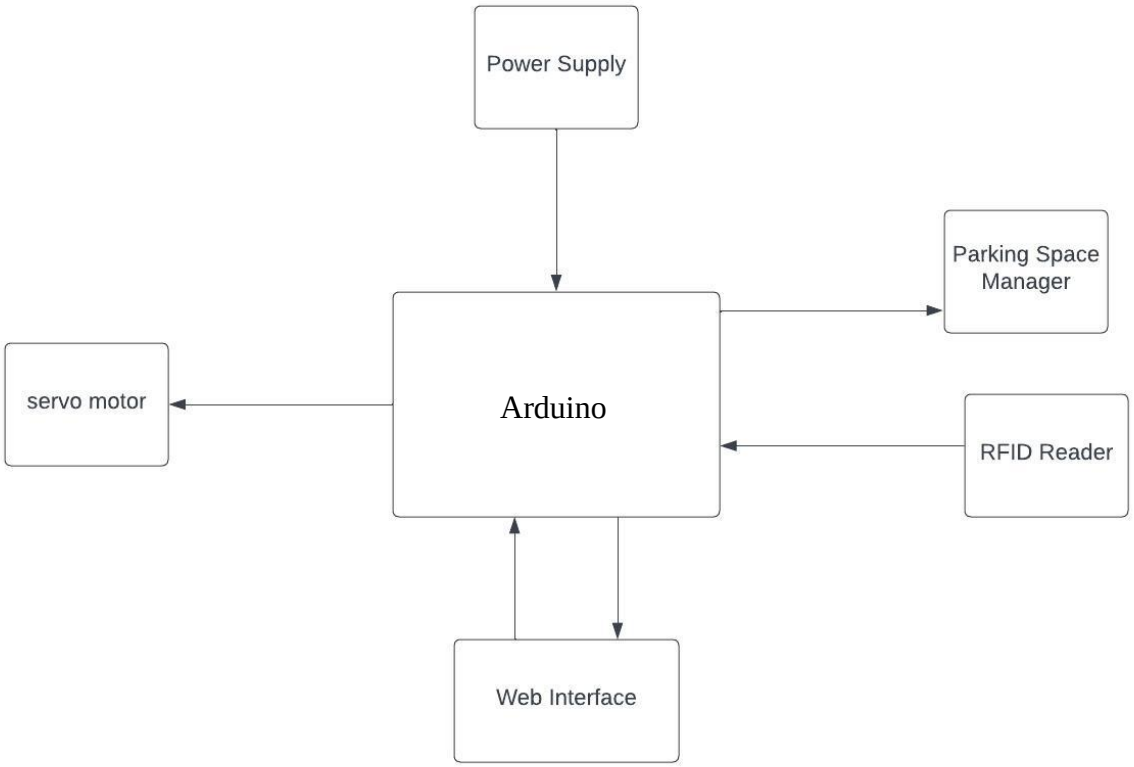
7. **Jumper Wires**: Jumper wires are used to establish electrical connections between different components of the smart parking system, facilitating data transmission and power distribution.

8. **I2C LCD Module**: The I2C LCD module serves as the user interface, displaying relevant information such as parking space availability, entry/exit instructions, and system status updates. It provides visual feedback to users and parking attendants.

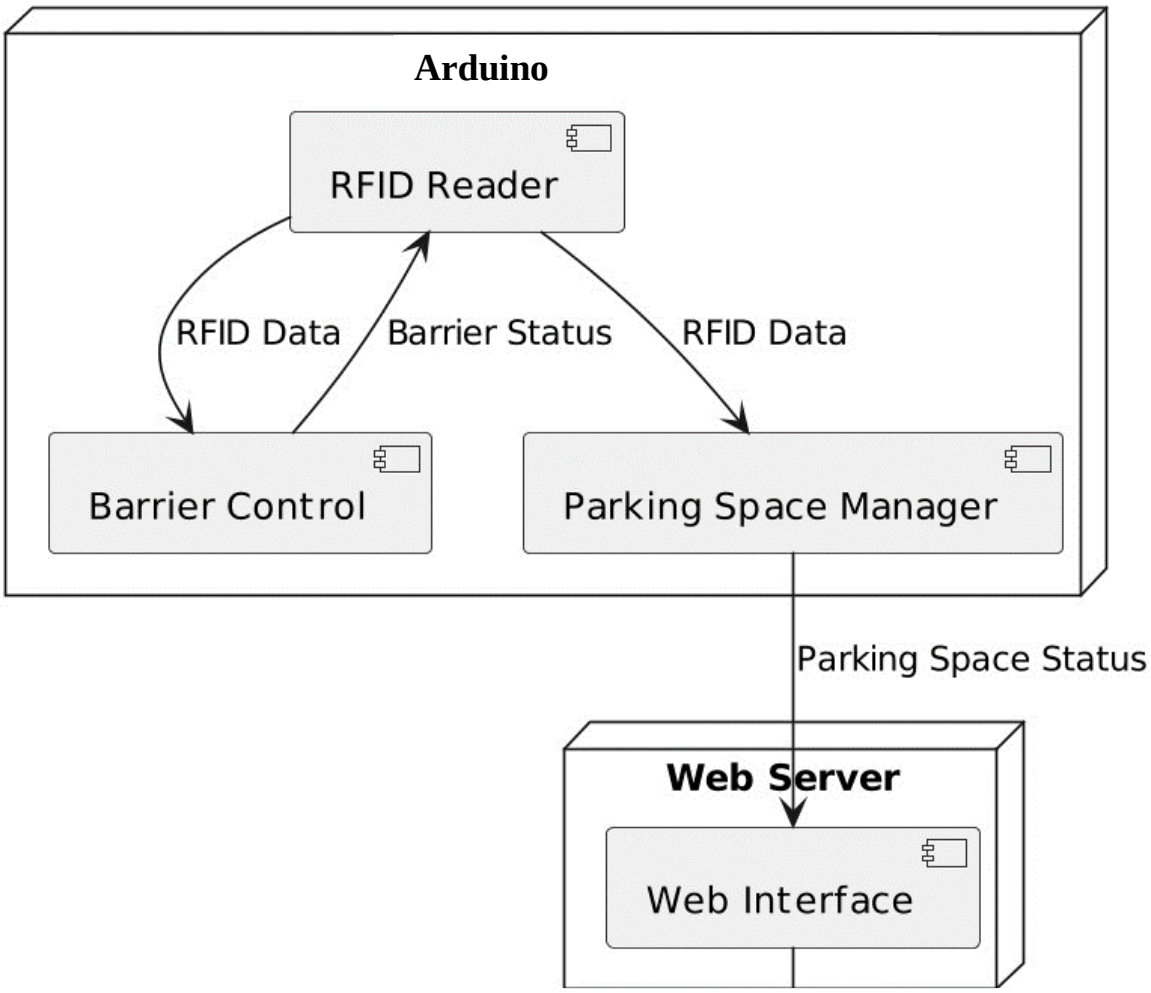
9. **ESP8266 Wi-Fi Module**: The ESP8266 WiFi module enables wireless communication between the smart parking system and external devices, such as smartphones or remote monitoring systems. It allows for remote access, control, and monitoring of the parking facility over a WiFi network.

Block Diagram (4.1 Basic Modules):

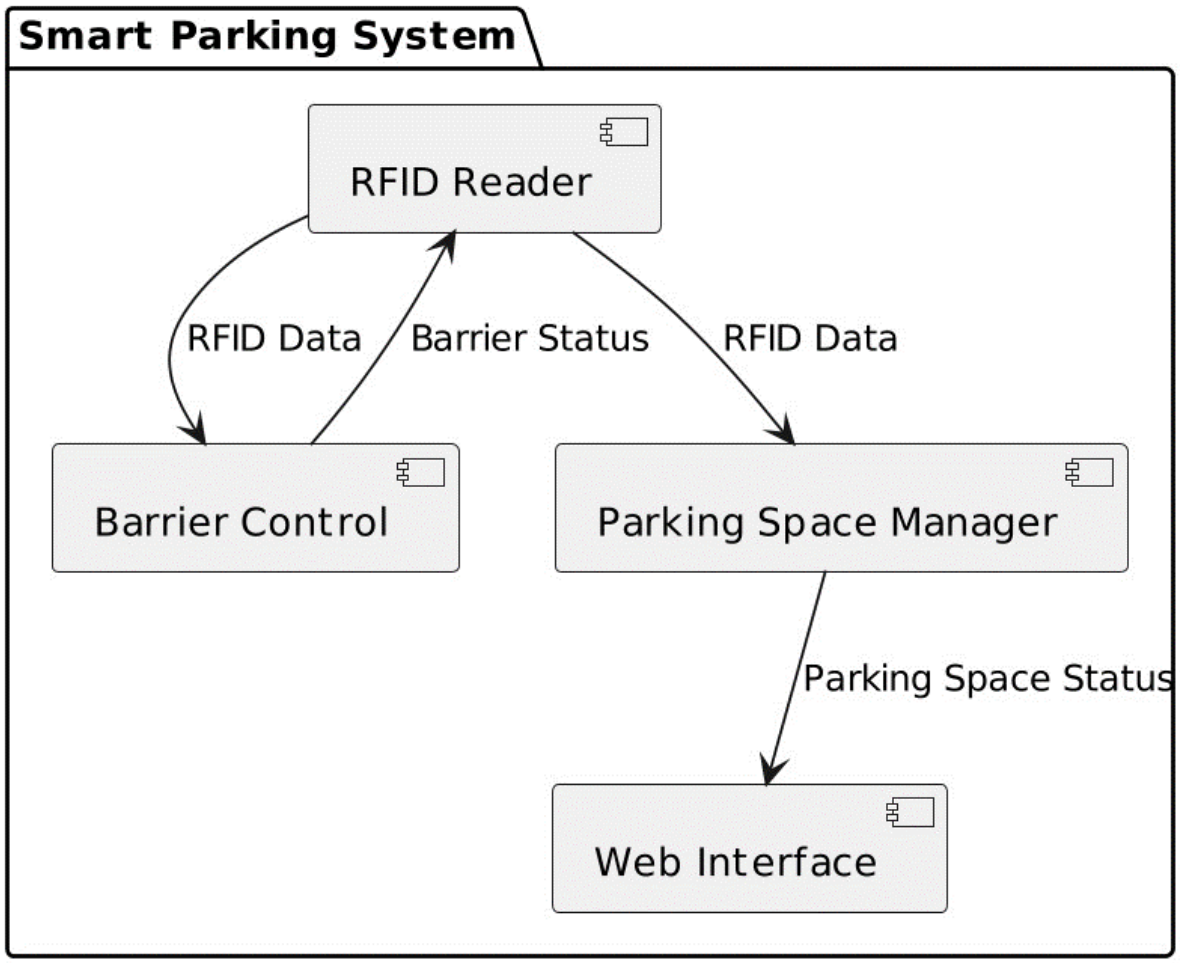
Block diagrams are typically used in the initial stages of system design to represent high-level modules or components and their interconnections.



Deployment Diagram (4.1 Basic Modules):
Deployment diagrams are part of basic modules as they illustrate the physical deployment of software components across different nodes in a system.



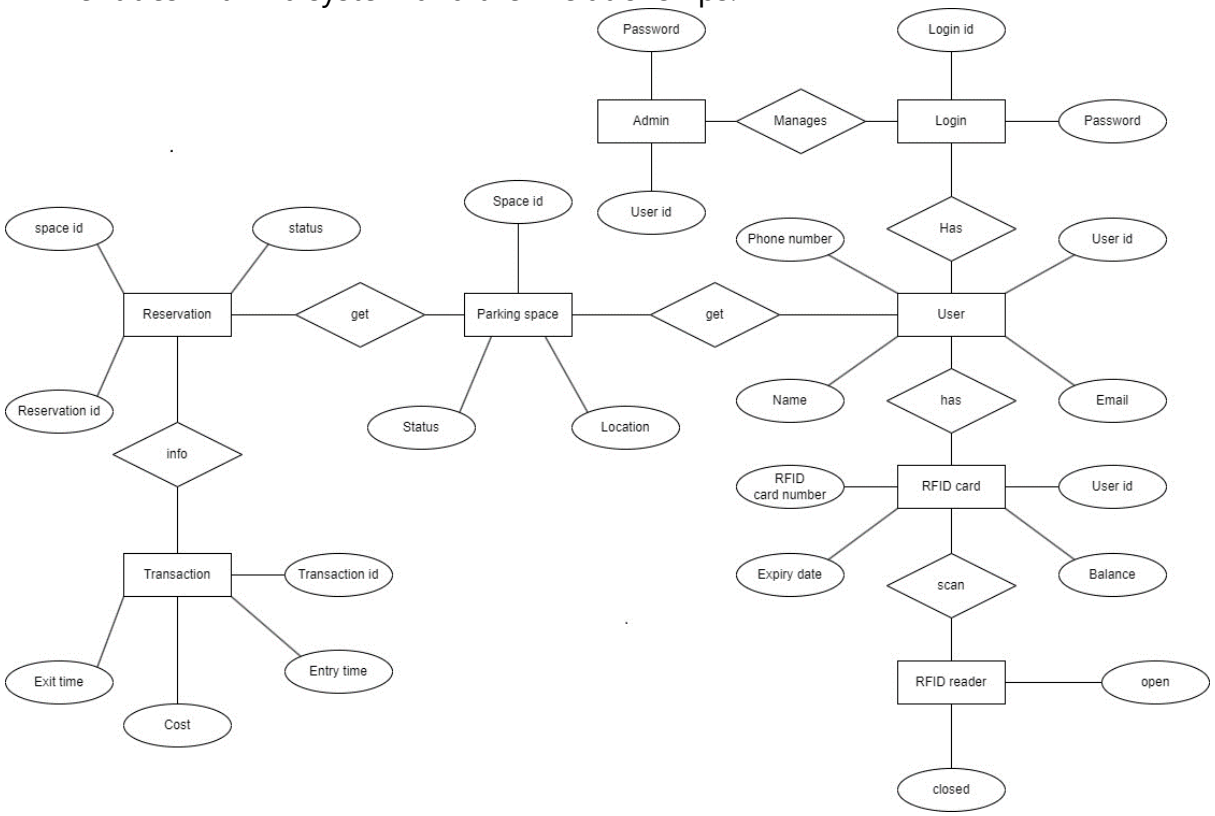
Package Diagram (4.1 Basic Modules):
Package diagrams are part of basic modules as they represent the organization of elements into packages or modules.



4.2 Data Design

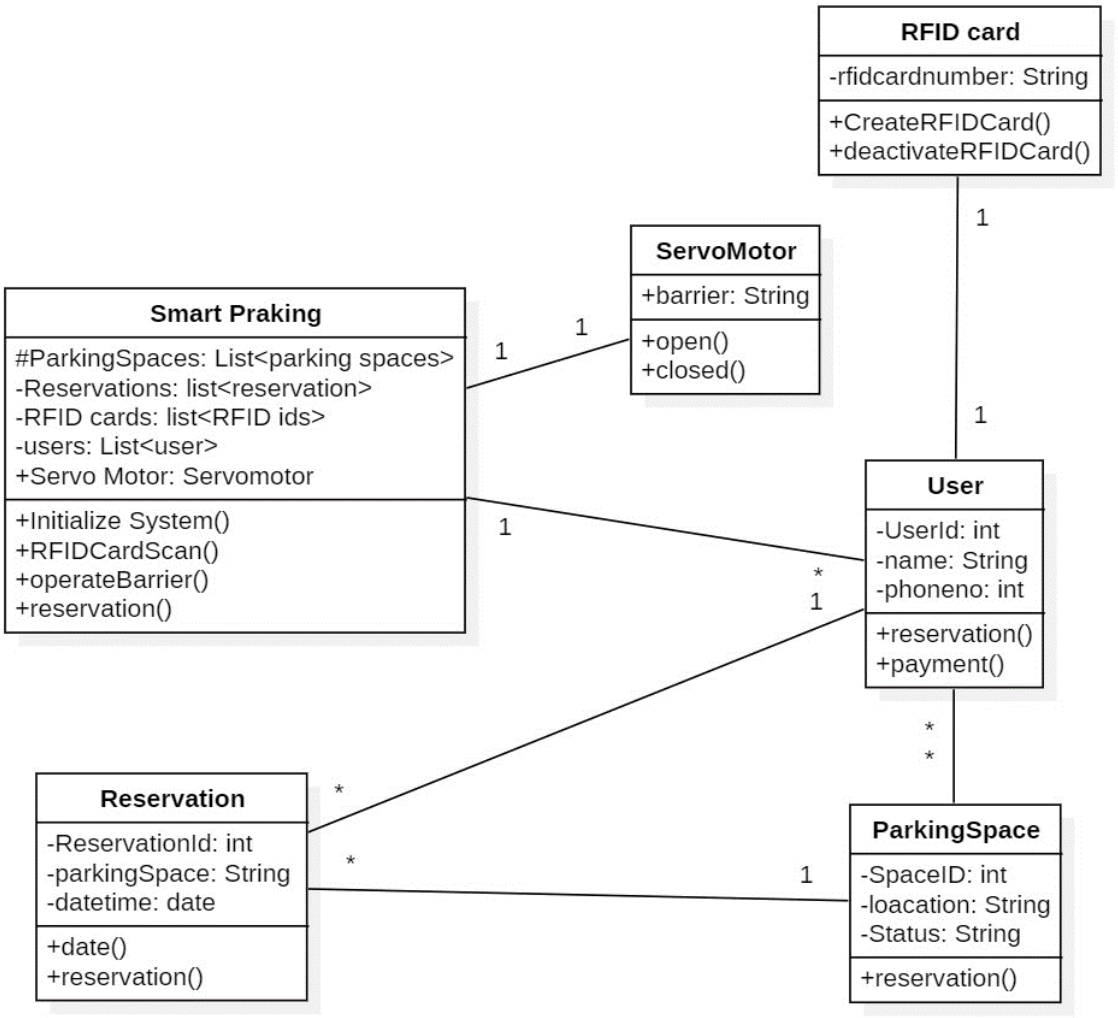
Data design is the structuring and organization of data to ensure efficient storage, retrieval, and management. It involves defining data elements, their relationships, and storage methods, forming the backbone of any data-driven system for smooth operations and scalability.

ER Diagram (4.2 Data Design):
Entity-Relationship (ER) diagrams are part of data design as they represent the entities within a system and their relationships.

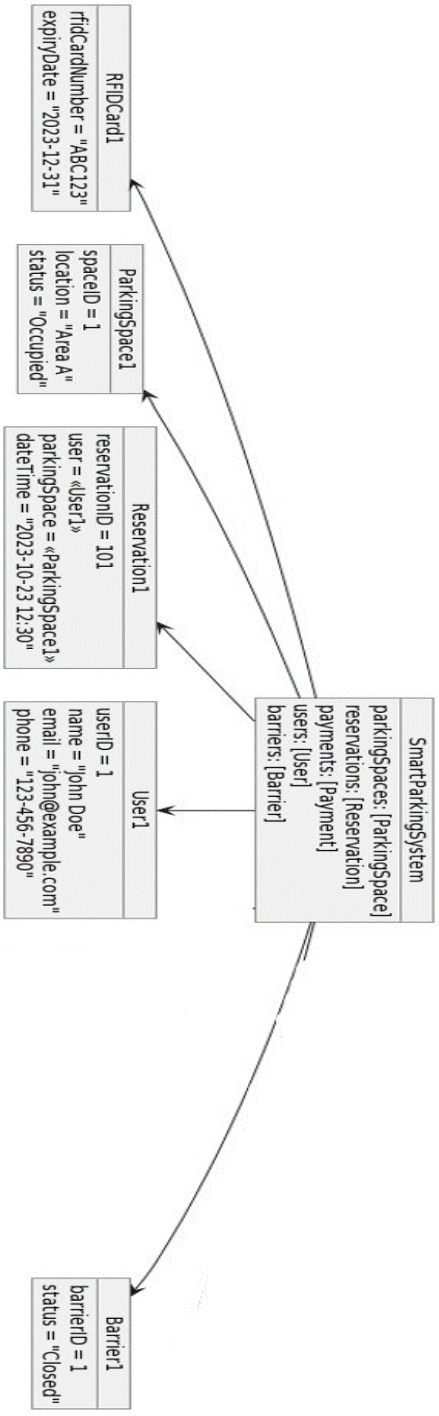


Class Diagram (4.2 Data Design):

Class diagrams are part of data design as they depict the structure of classes and their relationships in object-oriented programming.



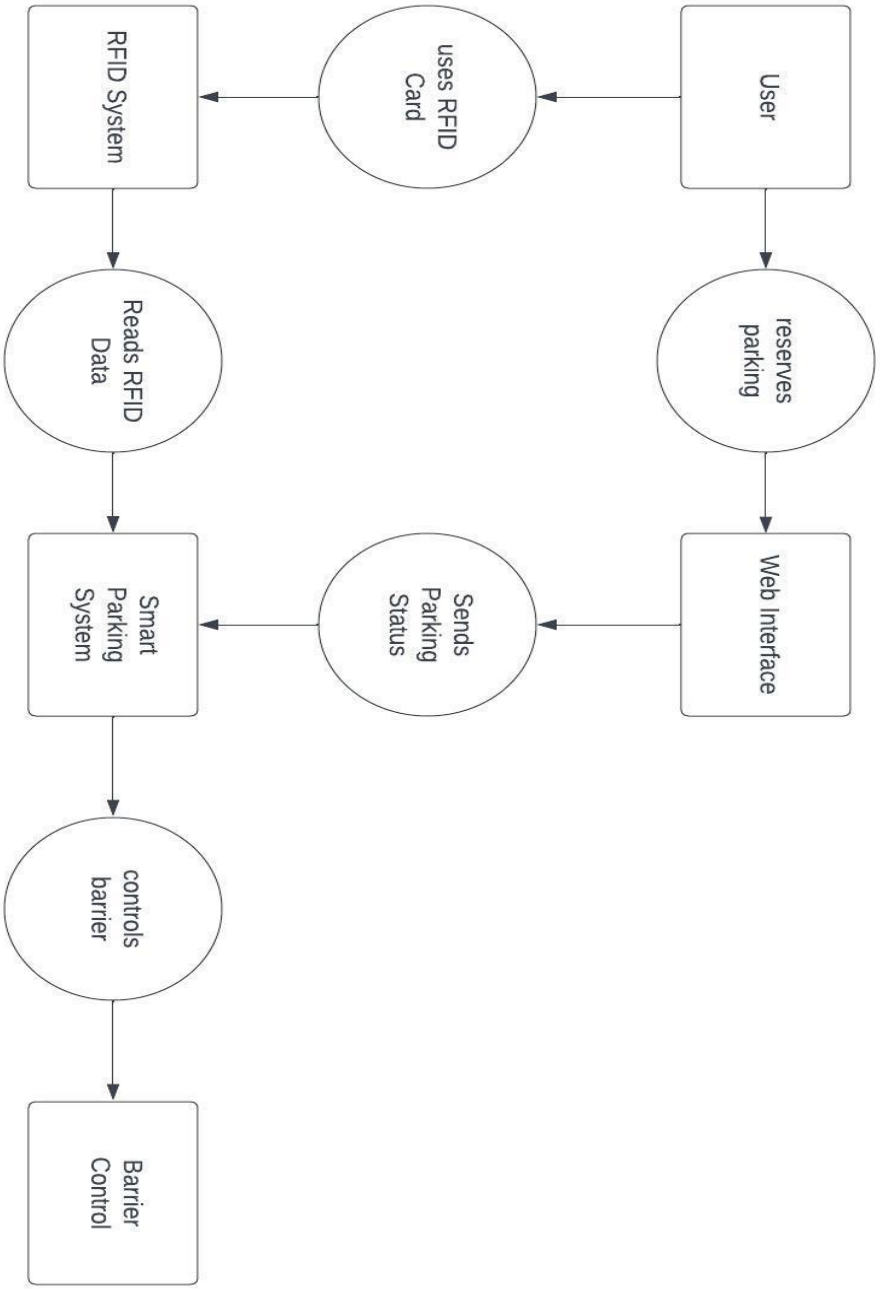
Object Diagram (4.2 Data Design):
Object diagrams are also part of data design as they provide a snapshot of instances of classes and their relationships at a particular moment.



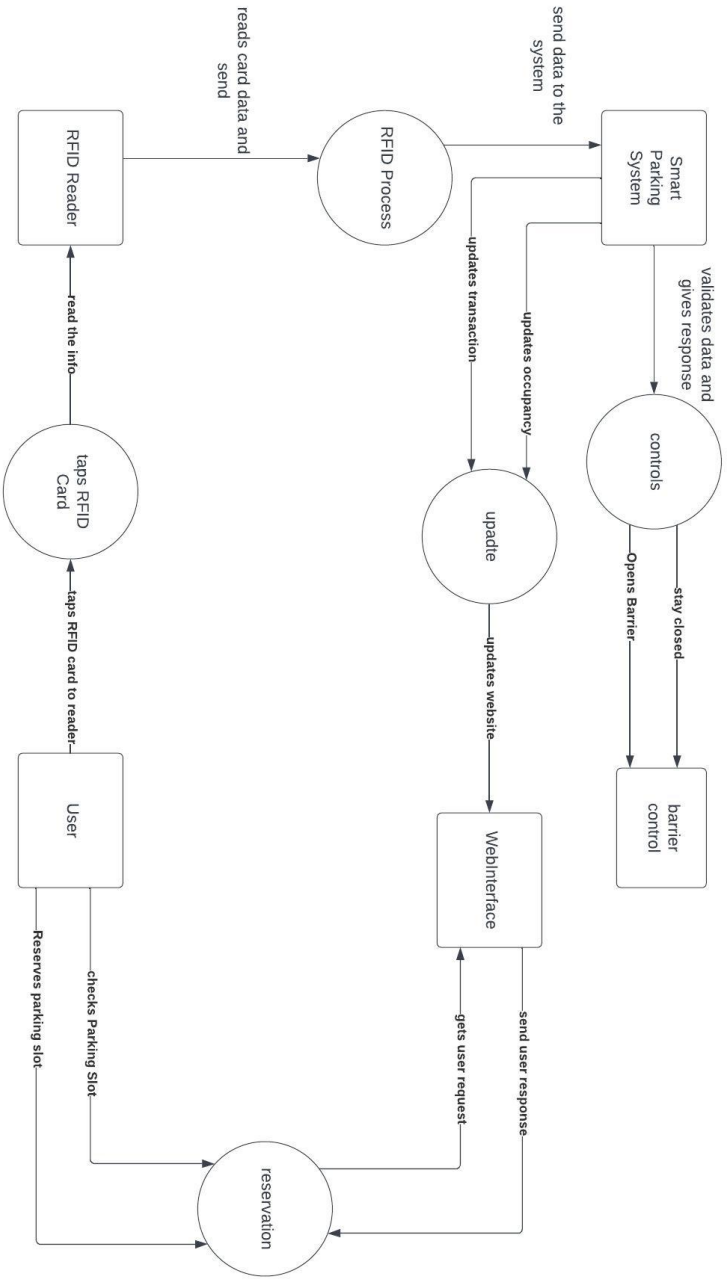
DFD Diagram (4.2 Data Design):

Data Flow Diagrams (DFD) are part of data design as they represent the flow of data within a system.

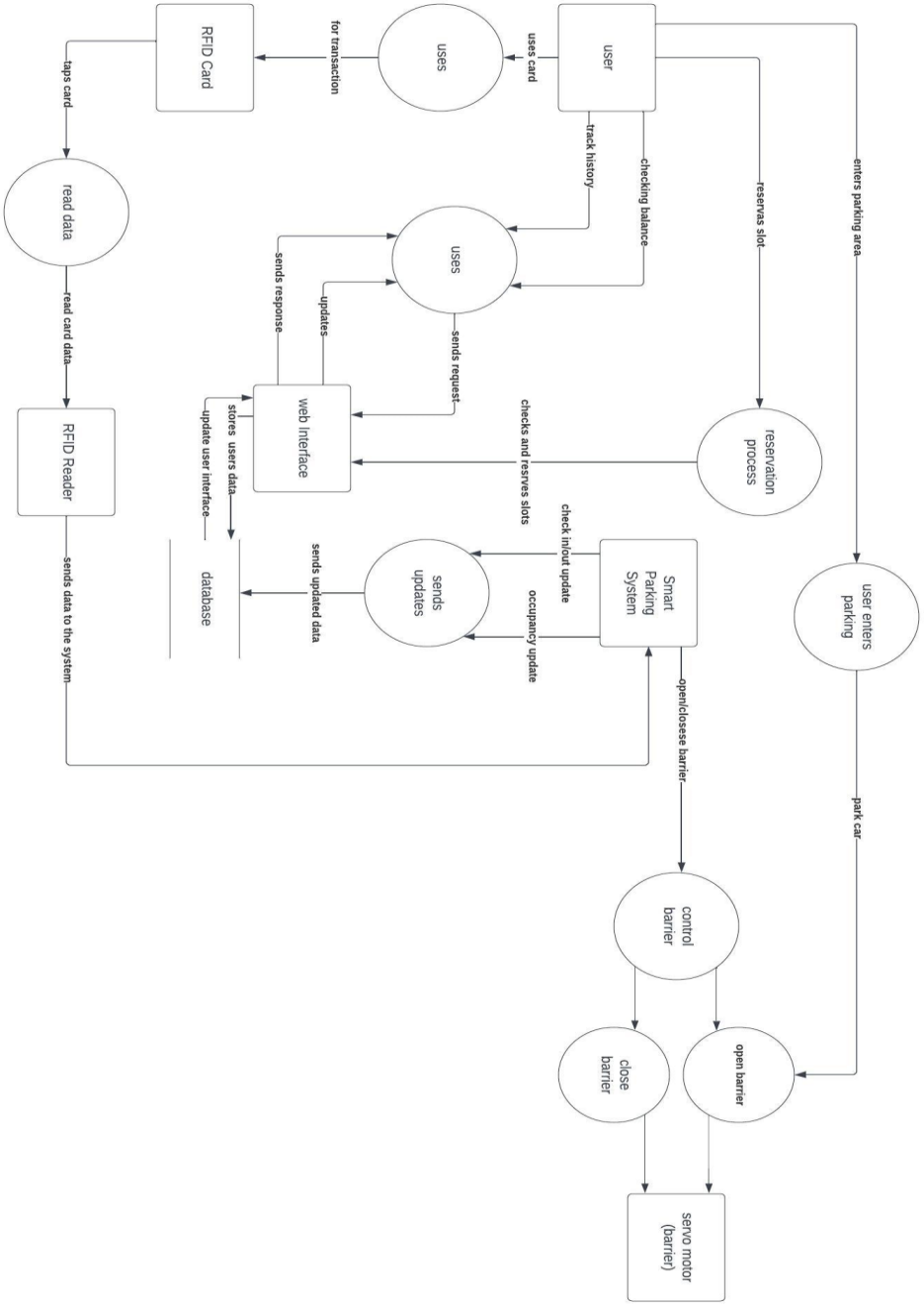
Level-0 DFD



Level-1 DFD


RFID Reader
reads card data and send
RFID Process
send data to the system
Smart Parking System
read the info
taps RFID Card
updates transaction
updates occupancy
upadte
validates data and gives response
controls
taps RFID card to reader
Opens Barrier
stay closed
updates website
User
WebInterface
barrier control
Reserves parking slot
checks Parking Slot
gets user request
send user response
reservation

Level-2 DFD

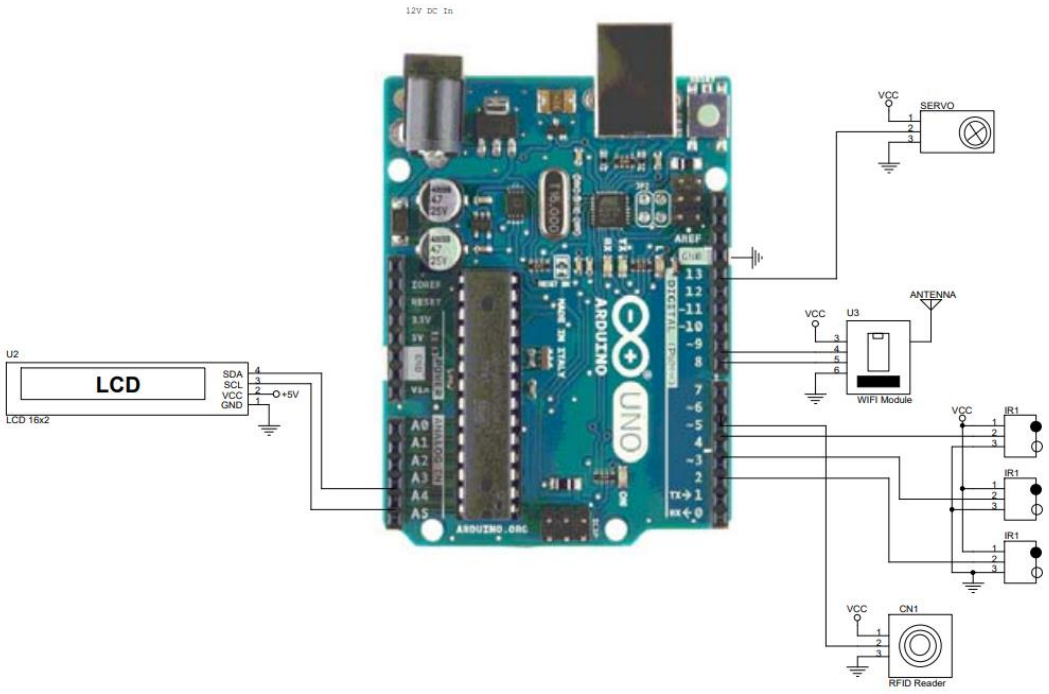

RFID Card
uses
user
for transaction
uses card
taps card
track history
checking balance
enters parking area
reservas slot
read data
uses
sends response
updates
sends request
read card data
web Interface
reservation process
stores users data
update user interface
checks and resrves slots
RFID Reader
check in/out update
Smart Parking System
user enters parking
sends updates
sends updated data
occupancy update
database
sends data to the system
open/closese barrier
park car
control barrier
close barrier
open barrier
servo motor (barrier)

4.2.1 Schema design

In the IoT Pet Feeder schema, commands from Google Assistant trigger actions through IFTTT, which communicates with Adafruit IO feeds. The NodeMCU controls the feeding process using a servo motor and LCD display, receiving instructions remotely and displaying real-time feeding schedules. Adafruit IO manages command reception and status updates, allowing users to feed their pets via voice commands or schedule feeding times. The system leverages NTP servers for accurate timekeeping, eliminating the need for a separate RTC module. This streamlined design integrates hardware, cloud services, and user interface components for a responsive and remotely accessible pet feeding solution.

Circuit diagram

Circuit diagram for this **IOT based Smart parking system** is given below. In this circuit diagram, a servo motor and LCD module is connected with Arduino ESP8266.



4.2.2 Data integrity and constraints

Data integrity:
ensures that data remains accurate, consistent, and reliable throughout its lifecycle. It involves maintaining the quality and reliability of data, preventing unauthorized access or modification, and ensuring data remains intact during storage, transmission, and processing.

Constraints:
Constraints define the rules and limitations imposed on data within a system to maintain its integrity and consistency. These constraints enforce business rules, data relationships, and allowable values, preventing invalid or inconsistent data from entering the system

1) Data Integrity for Time Synchronization:

Use NTP servers to ensure accurate time for scheduling pet feedings. This maintains the integrity of the feeding schedule, ensuring pets are fed at the correct times.

2) Input Validation for Feeding Commands:

Implement simple checks to confirm that voice commands through Google Assistant, like "Feed my pet", translate into valid feeding actions. This prevents misunderstandings or incorrect operations of the feeding mechanism.

3) Servo Motor Operation Limits:

Set a predefined range of motion for the servo motor to avoid overextension or damage. This acts as a constraint to ensure the feeder dispenses an appropriate amount of food.

4) Feeding Schedule Constraints:

Allow setting specific feeding times (morning, noon, evening) to regulate the pet's diet. These time slots act as constraints to prevent overfeeding or feeding at inappropriate hours.

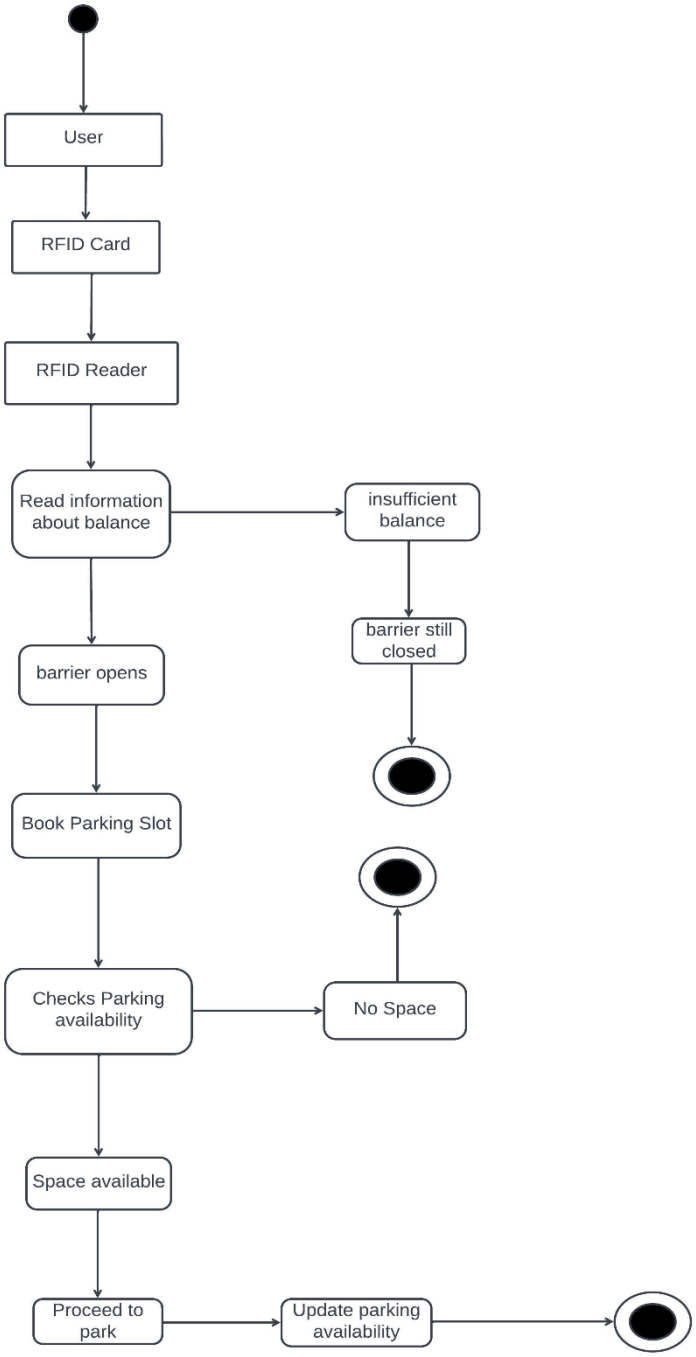
5) Network Reliability Check:

Ensure the NodeMCU maintains a stable connection to the internet for accurate time syncing and command reception. Implement a simple mechanism to retry the connection or alert the user if the feeder is offline.

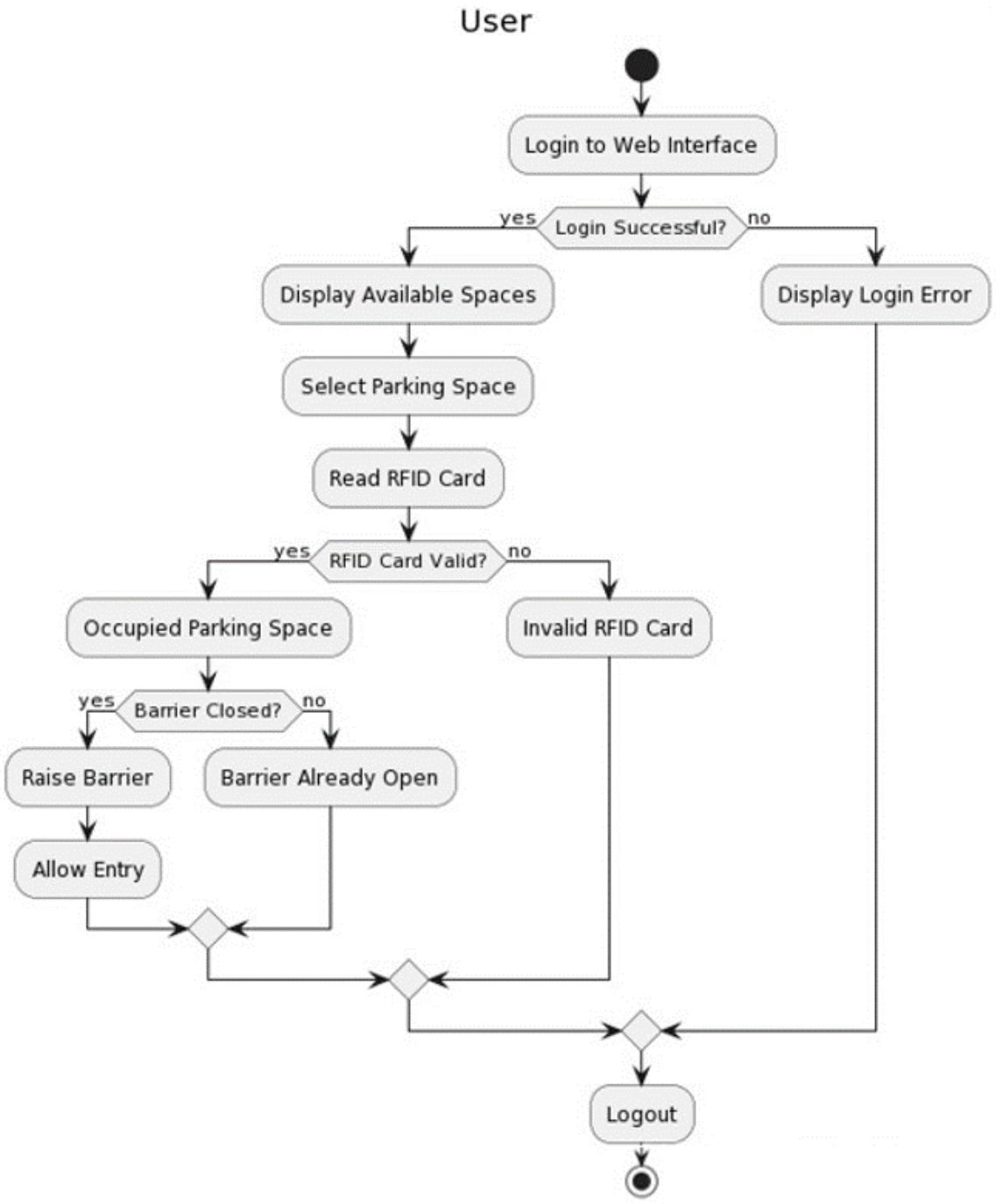
4.3 Procedural design

Procedural design in software engineering focuses on the processes or methods a system follows to accomplish tasks. It details the sequence of operations, the logic, and the flow of control within a program. Procedural design emphasizes how the system's components interact procedurally to achieve the desired outcomes, mapping out the steps and procedures necessary for each part of the system to function together cohesively.

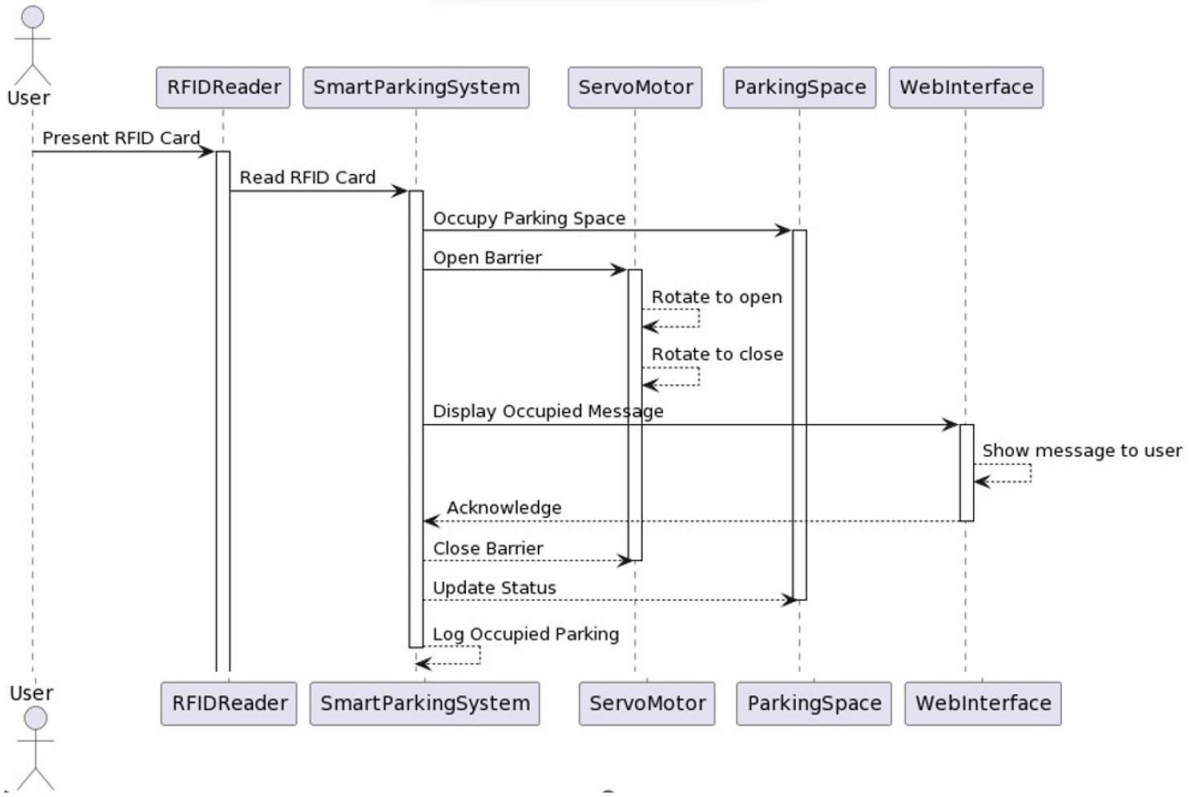
State Transition Diagram (4.3 Procedural Design):
State transition diagrams are part of procedural design as they depict the transitions between different states of an object or system



Activity Diagram (4.3 Procedural Design):
Activity diagrams are part of procedural design as they depict the flow of activities or processes within a system.

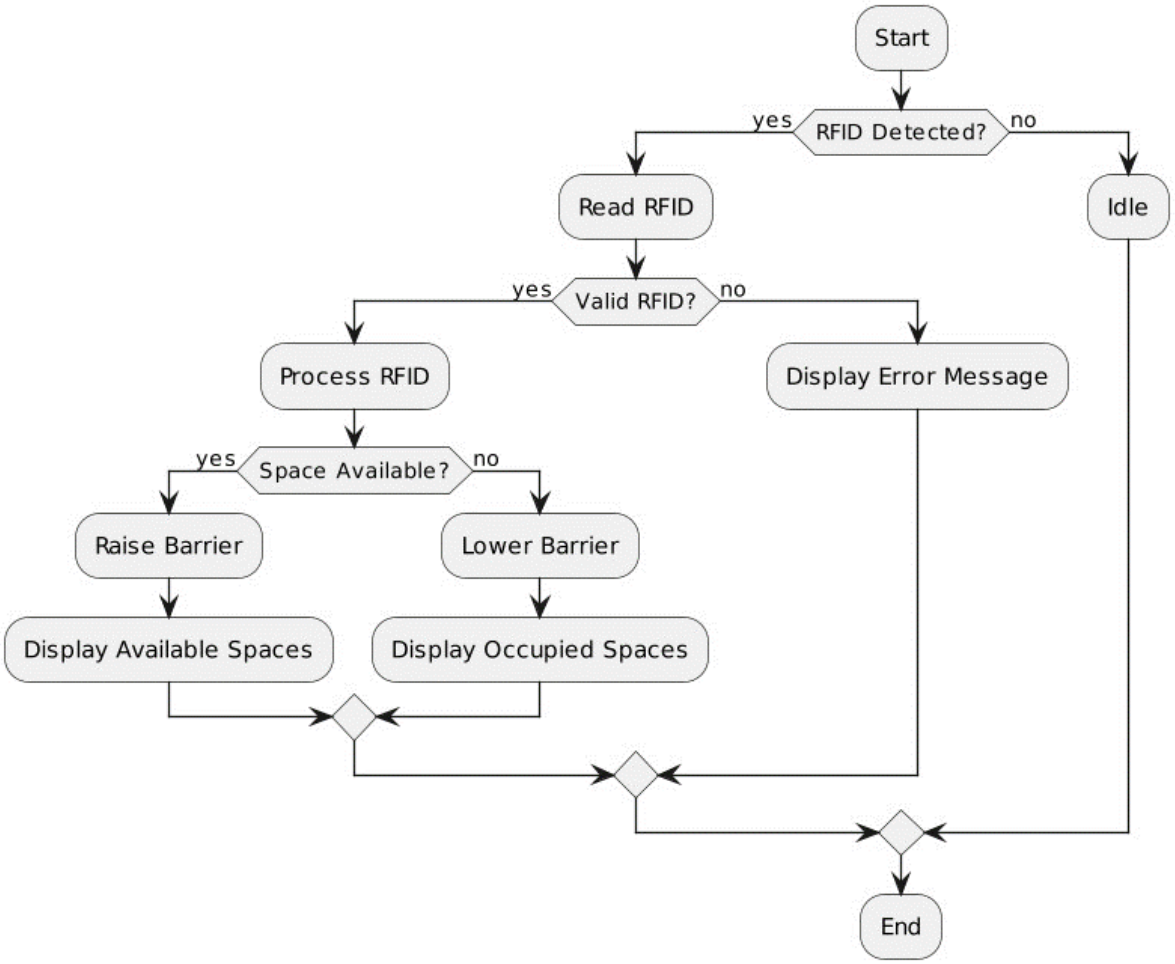


Sequence Diagram (4.3 Procedural Design):
Sequence diagrams are part of procedural design as they illustrate how objects interact with each other in a particular sequence over time.



4.3.1 Logic diagram : A logic diagram is a visual representation of the logical relationships and decision paths within a system, process, or circuit. It employs symbols and lines to depict how different components and operations interact, emphasizing the flow of control and the conditions that influence the path taken through a system or process.

Control Flow (4.3.1 Procedural Design/Logic Diagram):
Control flow diagrams are part of procedural design/logic diagram as they depict the flow of control
within a program or algorithm.



4.3.2 Data Structure

In the context of data structures, a tree diagram represents a hierarchical model consisting of nodes connected by edges. It starts with a root node and expands in branches, with each node having zero or more child nodes, leading to leaf nodes without any children. This structure enables efficient organization, storage, and retrieval of data, supporting operations like searching, sorting, and traversal. Tree diagrams are fundamental in illustrating various tree-based data structures, such as binary trees, binary search trees, and n-ary trees, each serving specific purposes in computing, like organizing data in hierarchical systems, facilitating fast data lookup, and representing structured relationships.

4.3.3 Algorithm design

Algorithm design refers to the process of creating a set of instructions or a step-by-step procedure for solving a problem or performing a task efficiently. It involves defining the steps necessary to solve a problem, considering factors such as correctness, efficiency, and scalability. Algorithm design is crucial in computer science and programming because it forms the foundation for writing code that can effectively solve real-world problems.

Algorithm Overview

Step 1: Setup and Initialization

Step 2: RFID Tag Detection

Step 3: Space Availability Check

Step 4: Entry/Exit Control

Step 5: Displaying Status on LCD

4.4 User Interface design

The user interface (UI) of a Smart Parking System is intuitively designed to streamline the parking experience for both facility managers and drivers. For managers, the dashboard offers a comprehensive overview of the parking facility, displaying real-time updates on available parking spaces and entry/exit

activities. It provides tools for managing entry/exit barriers efficiently and offers analytical features for monitoring parking usage trends. On the other hand, drivers interact with the system through a mobile or web application, which presents them with essential information such as the availability of parking spaces, navigation to vacant spots, and options for reserving and making payments.

Additionally, LCD displays installed at entry/exit points offer clear instructions and real-time updates on parking availability, ensuring smooth traffic flow within the facility. For added convenience, some systems may incorporate voice control functionality, allowing users to perform tasks such as checking space availability or opening barriers through voice commands. Overall, the UI prioritizes simplicity, clarity, and user-friendliness to enhance the parking experience for all stakeholders involved.

4.5 Security issues

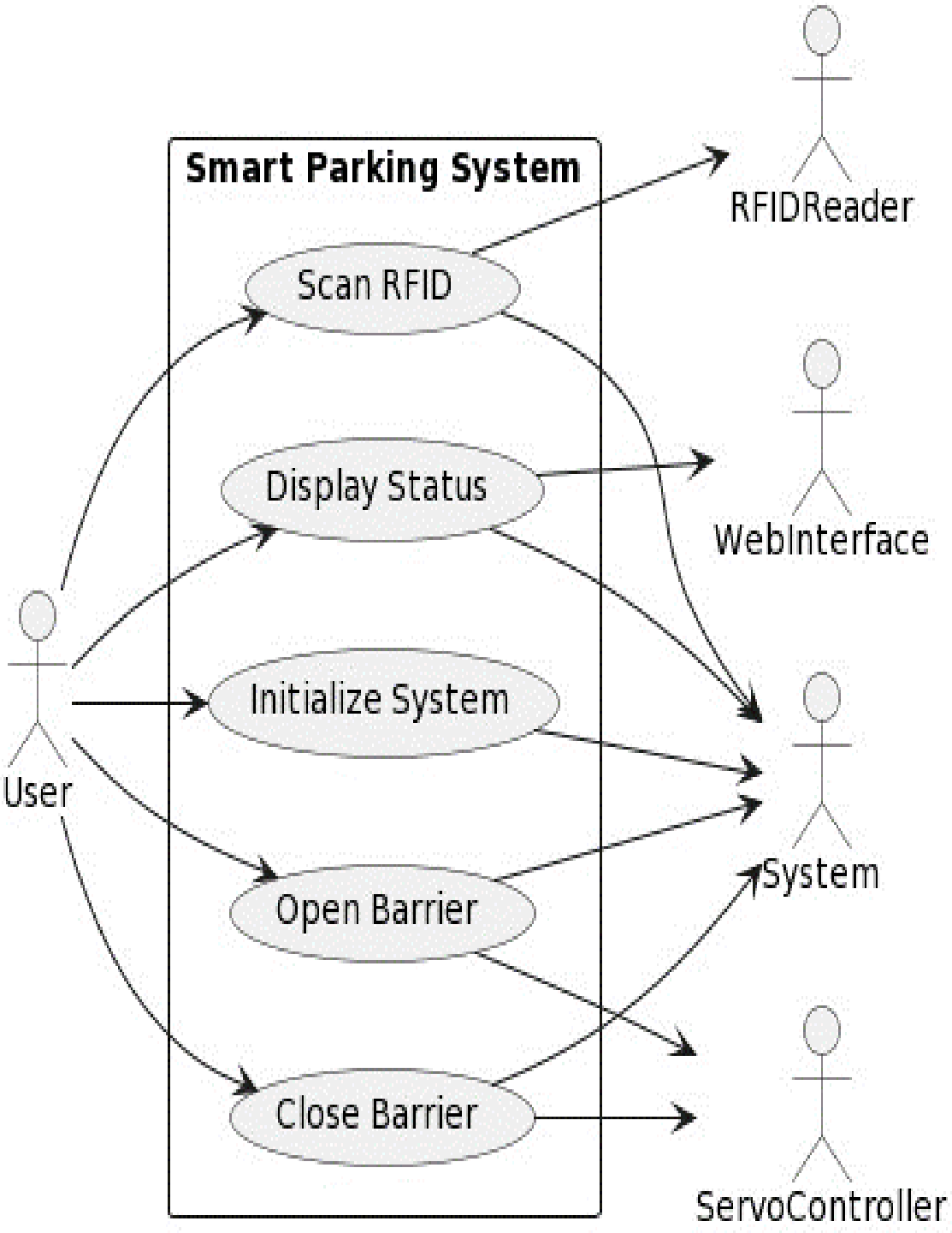
Security Issue	Description	Potential Impact	Mitigation
Unauthorized Access	Unauthorized individuals gaining access to the parking facility or system components.	Theft, damage	Use RFID authentication and encryption to prevent access.
Data Breaches	Breaches of sensitive data stored within the system, including user information and payment details.	Privacy violations, losses	Encrypt data and update security protocols regularly.
Malware Attacks	Malicious software infiltrating the system, disrupting operations or stealing information.	System downtime, data loss	Use antivirus software, update system firmware regularly.
Physical Security	Physical breaches of the parking facility, like vandalism or tampering with system components.	Equipment damage, downtime	Implement surveillance and tamper-resistant hardware.
Denial of Service	Deliberate attempts to overwhelm the system with excessive traffic, causing service disruptions.	Downtime, revenue loss	Employ rate limiting and DoS protection services.
Network Vulnerabilities	Weaknesses in network infrastructure or communication protocols that could be exploited by attackers.	Unauthorized access, data interception	Employ firewalls and encryption to secure communications.
Social Engineering	Manipulative tactics used by attackers to deceive users or gain unauthorized access through human vulnerabilities.	Compromised credentials	Conduct security awareness training and implement multi-factor authentication.
Lack of Updates	Failure to regularly update system firmware or security patches, leaving vulnerabilities unaddressed.	Increased susceptibility	Establish a patch management process and conduct regular updates.

4.6 Test case design

Test case design involves creating a structured approach to evaluate the functionality, performance, and reliability of software or systems. It includes defining specific test scenarios, inputs, expected outcomes, and criteria for determining pass or fail conditions. Test cases are designed to cover various aspects of the system, including normal operation, edge cases, error handling, and boundary conditions. Effective test case design ensures comprehensive test coverage, identifies defects early in the development cycle, and validates that the system meets the specified requirements and quality standards. It involves techniques such as equivalence partitioning, boundary value analysis, and state transition testing to systematically generate test cases that effectively validate the functionality and behavior of the system under test.

Implementation of test cases are provided in chapter 5 and chapter 6

Use Case Diagram (4.6 Test Cases Design):
Use case diagrams are typically used in test cases design as they represent interactions between actors and the system.



Event Table

Event	Trigger	Action	Description
Vehicle Entry	Vehicle detected by entry sensors	Update occupancy status	When a vehicle enters, the system detects it via entry sensors and updates the occupancy status of available spaces.
Vehicle Exit	Vehicle detected by exit sensors	Update occupancy status	Upon vehicle exit, the system updates the occupancy status, potentially triggering billing processes.
RFID Authentication	User presents RFID tag/card	Grant access/record entry	User authentication occurs when RFID tag/card is presented, allowing access and recording entry into the system.
Space Availability	Real-time sensor data	Display available spaces	Real-time updates on parking space availability are displayed on LCD screens or web interfaces for driver convenience.
Entry/Exit Gate Control	Vehicle authentication signal	Open/close entry/exit gate	Entry/exit gates automatically open/close in response to vehicle authentication and occupancy status.
Overcapacity Alert	Maximum capacity reached	Alert parking administrators	Notifications are sent to administrators when the facility reaches maximum capacity to prevent overcrowding.

Chapter 5: Implementation and Design

5.1 Implementation approaches

Setup and Configuration of Arduino IDE:
Begin by installing the Arduino IDE and configure it to support the Arduino Uno or Mega board, depending on your project's requirements. Ensure you have the necessary libraries installed for components like RFID reader, LCD display, and ESP8266 WiFi module if required.

Circuit Assembly:
Follow the provided circuit diagram to connect the RFID reader, LCD display, and ESP8266 WiFi module to the Arduino board. Double-check the wiring to ensure all connections are correct, paying special attention to power and ground connections.

Software Setup:
Download and install any required software libraries for the RFID reader, LCD display, and ESP8266 WiFi module. Configure the WiFi module to connect to your local network, and set up any necessary communication protocols between the Arduino and the components.

Database Setup:
If your smart parking system requires a database to store parking space availability or user information, set up a database server and create the necessary tables and fields. Ensure the database is accessible to the Arduino or ESP8266 for data retrieval and storage.

Webpage Development:
Design and develop a user interface for the smart parking system's web application. Include features such as real-time parking space availability, user authentication, reservation options, and payment processing if required. Ensure the webpage is responsive and accessible from various devices.

Testing and Debugging:
Test each component of the smart parking system individually to ensure they function correctly. Perform integration testing to verify that all components work together seamlessly. Debug any issues encountered during testing and make necessary adjustments to the code or hardware configuration.

Deployment:
Once testing is complete and any issues have been resolved, deploy the smart parking system in the desired location. Ensure all components are securely installed and properly configured for long-term operation.

Monitoring and Maintenance:
Implement monitoring tools to track the performance of the smart parking system over time. Regularly monitor system logs, database usage, and user feedback to identify any issues or areas for improvement. Perform routine maintenance tasks such as software updates, hardware inspections, and database backups to ensure the system remains reliable and secure.

5.2 Coding details and code efficiency

The Smart Parking System utilizes RFID technology and microcontrollers like Arduino or ESP8266 to manage parking spaces effectively. When a vehicle enters, its RFID tag is detected and validated against a database, marking the space as occupied. Real-time status updates are displayed on an LCD screen at the entrance and a webpage accessible to users. As vehicles exit, their RFID tags are again verified, updating the status to available. Users can interact with the system through the LCD display or webpage, accessing information and making reservations. Overall, the system streamlines parking management, reduces congestion, and enhances user experience.

5.2.1 Coding efficiency

Arduino Microcontroller

finalProjFinalScript.ino

```
#include <EEPROM.h>
#include <SoftwareSerial.h>
#include <LiquidCrystal_I2C.h>

#include<Servo.h>
Servo myservo;

#define ir1 14
#define ir2 15
#define ir3 16

#define servoPin 8

      String ssid ="MI_X2";
      String password="12345678";
      String server = "www.hobbykits4u.com";
      String Tx_URL = "/car_parking.json";


SoftwareSerial rfid_rx(3, 2);
SoftwareSerial esp(4, 5);

String slt1;
String slt2;
String slt3;
String slt_str,last_str,data;
char input[12];
int count = 0;
byte user_id;
byte user_sts_1 = 0;
byte user_sts_2 = 0;
byte user_sts_3 = 0;


void setup(){
  lcd.begin();
```

```
  lcd.backlight();

    pinMode(ir1, INPUT);
    pinMode(ir2, INPUT);
    pinMode(ir3, INPUT);

    lcd.setCursor(0,0);
    lcd.print("Welcome To IOT");
    lcd.setCursor(0,1);
    lcd.print("CAR Parking SYS");
    delay(3000);

    myservo.attach(servoPin);


    reset();
    connectWifi();
}

void loop()
{
 check_Parking();

      if(slt_str == "111"){
          lcd.setCursor(0,0);
          lcd.print("  Parking FULL  ");
          lcd.setCursor(0,1);
          lcd.print(" P1=" + slt1 + " P2=" + slt2 + " P3=" + slt3 + " ");
      }
      else{
          lcd.setCursor(0,0);
          lcd.print(" P1=" + slt1 + " P2=" + slt2 + " P3=" + slt3 + " ");
          lcd.setCursor(0,1);
          lcd.print(" SCAN RFID Tag  ");
      }

         get_RFID();

           if (temp > 20){
              last_str = slt_str;
              check_Parking();
              Tx_IOT();
              temp = 0;
           }
           else{
              temp ++;
           }

      delay(100);
}

void get_RFID(){
          if (rfid_rx.available()) {
              input[count] = rfid_rx.read();
              count++;

              if ((strncmp(input, "550077F49C4A", 12) == 0)){
```

```
  user_id = 101;
  if (user_sts_1 == 0){
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("USER 1 <> ENTRY");
      user_sts_1 = 1;
      delay(2000);
  }
  else{
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("USER 1 <> EXITE");
      user_sts_1 = 0;
      delay(2000);
  }
}
else if ((strncmp(input, "5500B45D66DA", 12) == 0)){
  user_id = 102;
  if (user_sts_2 == 0){
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("USER 2 <> ENTRY");
      user_sts_2 = 1;
      delay(2000);
  }
  else{
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("USER 2 <> EXITE");
      user_sts_2 = 0;
      delay(2000);
  }
}
else if ((strncmp(input, "550078374258", 12) == 0)){
  user_id = 103;
  if (user_sts_3 == 0){
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("USER 3 <> ENTRY");
      user_sts_3 = 1;
      delay(2000);
  }
  else{
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("USER 3 <> EXITE");
      user_sts_3 = 0;
      delay(2000);
  }
}
else{
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Invalid USER");
  lcd.setCursor(0,1);
  lcd.print("TRY again !");
  delay(2000);
```

```
              }

          }
}

void check_Parking(){
      slt_str = " ";
      if(digitalRead(ir1) == 0){
        slt1 = "F";
        slt_str = "1";
      }
      else{
        slt1 = "M";
        slt_str = "0";
      }

      if(digitalRead(ir2) == 0){
        slt2 = "F";
        slt_str = slt_str + "1";
      }
      else{
        slt2 = "M";
        slt_str = slt_str + "0";
      }

      if(digitalRead(ir3) == 0){
        slt3 = "F";
        slt_str = slt_str + "1";
      }
      else{
        slt3 = "M";
        slt_str = slt_str + "0";
      }
}

void Tx_IOT(){

        data = String(user_id) + String(user_sts_1) + String(user_sts_2) +
String(user_sts_3) + slt_str;

                lcd.clear();
                lcd.setCursor (0,0);
                lcd.print("Tx Data..");
                delay(10);

                esp.println("AT+CIPSTART=\"TCP\",\"" + server);

                if( esp.find("OK")) {
                   Serial.println("ready  for send_data");
                } delay(500);


                      String postRequest =
                "POST " + Tx_URL + " HTTP/1.0\r\n" +
                "Host: " + server + "\r\n" +
                "Accept: *" + "/" + "*\r\n" +
                "Content-Type: application/x-www-form-urlencoded\r\n" +
```

```
                "\r\n" + data1;
                String sendCmd = "AT+CIPSEND=";

                esp.print(F("AT+CIPSEND="));
                esp.println(postRequest.length() );
                delay(500);

                if(esp.find(">"))
                {
                Serial.println("Sending..");
                esp.print(postRequest);
                if( esp.find("SEND OK"))
                {
                while (esp.available())
                {
                lcd.setCursor (0,1);
                lcd.print("OK.....");
                }
                }
              }

}


 void reset() {
        delay(1000);
        esp.println(F("AT+RST"));
        if(esp.find("OK")) Serial.println(F("Reset Connection"));

}

void connectWifi() {

        lcd.clear();
        lcd.setCursor (0,0);
        lcd.print("Connect To WIFI");
        delay(1000);

        String cmd = "AT+CWJAP=\"" +ssid+"\",\"" + password + "\"";
        Serial.print(F("Connect To Network >>"));
        delay(1000);
        esp.println(cmd);

        if(esp.find("OK")) {
           delay(100);
           Serial.println(F("Connected!"));
           delay(2000);
           temp = 0;
        }
        else {
           Serial.println(F("Cannot connect to wifi"));
           lcd.clear();
           lcd.setCursor (0,0);
           lcd.print("WIFI Error = ");
           delay(1000);
           connectWifi();
        }
```

```
    }

void gate_con(){
     myservo.attach(servoPin);
     delay(10);

     for(int i = 0; i < 90; i++) {
         myservo.write(i);
         delay(10);
      }
      delay(2000);
      for(int i = 90; i > 0; i--) {
          myservo.write(i);
          delay(10);
      }
      delay(50);
}
```

5.3 Testing Approach

Testing is a critical phase in ensuring the functionality and reliability of the IoT Pet Feeder system. Here are three key testing approaches:

5.3.1 Unit Testing

Is the first level of software testing where individual components or modules of a software are tested in isolation to verify that each unit operates correctly. This process is primarily performed by developers to ensure that their code meets the design specifications and behaves as intended, often using automated tools to execute the test cases.

Test Objectives

1) **RFID Reader Testing:**
    a. Test RFID reader functionality by simulating tag detection.
    b. Verify that the RFID reader can correctly read and validate RFID tags.
    c. Test for accurate detection of multiple tags in close proximity.
2) **ESP8266 Connectivity Testing:**
    a. Test the connection between ESP8266 module and the server.
    b. Verify that the ESP8266 can establish a stable Wi-Fi connection.
    c. Test data transmission reliability and speed over Wi-Fi.
3) **LCD Display Testing:**
    a. Verify that the LCD display can correctly show real-time parking status updates.
    b. Test different scenarios, such as displaying "occupied" or "available" status.
4) **Webpage Functionality Testing:**
    a. Test the functionality of the webpage for users to view parking availability.
    b. Verify that the webpage updates in real-time based on parking space occupancy.
5) **Servo Motor Operation Testing:**
    a. Test the servo motor functionality for opening and closing entry/exit barriers.
    b. Verify that the servo motor operates smoothly and reliably based on parking status changes.
6) **Security Testing:**
    a. Test the system for vulnerabilities, such as unauthorized access.

5.3.2 Integration Testing

Focuses on combining individual units or modules of a software into a group to identify any defects in the interaction between these integrated components. This testing phase aims to detect problems in the interfaces and interaction between integrated units, ensuring that they work together seamlessly to perform their intended functions.

Test Objectives:

RFID Reader Integration
Test the integration of RFID reader with the system to ensure it accurately detects and reads RFID tags.
Verify that RFID tag information is correctly transmitted to the central system for processing.
ESP8266 Connectivity Integration:
Test the integration between ESP8266 module and the server to ensure seamless communication.
Verify that parking status updates are transmitted accurately from the ESP8266 to the server, and vice versa.
LCD Display Integration:
Test the integration of LCD display with the system to ensure it receives and displays real-time parking status updates.
Verify that the LCD display is synchronized with the central system for consistent information display.
Webpage Integration:
Test the integration of the webpage with the central system to ensure it accurately reflects parking availability.
Verify that parking status changes are immediately reflected on the webpage for users to view.
Servo Motor Integration:
Test the integration of servo motors with the system to ensure they respond appropriately to parking status changes.
Verify that entry/exit barriers are opened or closed as per parking availability updates.
User Interface Integration:
Test the integration of user interfaces, including physical interfaces and web interfaces, with the central system.
Verify that users can interact seamlessly with the system to view parking availability and make reservations.
System Component Interaction:
Test interactions between different system components to ensure they work together harmoniously.
Verify that RFID reader, ESP8266 module, LCD display, webpage, and database communicate effectively to manage parking operations.

5.3.3 Beta Testing

Is one of the final testing stages where the nearly finished product is released to a limited number of end-users outside of the company to obtain real-world exposure and feedback. This process helps identify any remaining flaws or usability issues from the user's perspective, ensuring the product's quality and readiness for the general market release.

Test Objectives:

1. **Choose Testers**:
    - Select a diverse group of testers, including drivers, facility managers, and administrators.
    - Ensure testers have varying technical skills and parking system knowledge.
2. **Set Up Testing**:
    - Create a test environment mimicking real parking conditions.
    - Equip spaces with RFID readers, ESP8266 modules, and LCD displays.
3. **Test Scenarios**:
    - Design different tests covering system functions like RFID tag detection, real-time updates, and gate control.
    - Have testers perform tasks and share their experiences.
4. **Collect Feedback**:
    - Use surveys or interviews for feedback submission.
    - Ask testers to report issues and suggest improvements.
5. **Analyze Data**:
    - Review feedback to find common issues and priorities.
    - Sort feedback by severity and user impact.
6. **Make Changes**:
    - Develop a plan to address feedback and implement improvements.
    - Release updates based on user feedback.
7. **Evaluate**:
    - Assess the system after updates to ensure issues are resolved.
    - Confirm user satisfaction and note any remaining problems.

5.4 Modifications and improvements

During the testing phase of my project, I encountered a range of errors and bugs within the code and functionalities. Minor issues were promptly resolved during unit testing through direct troubleshooting efforts. However, more complex challenges necessitated additional research and learning. To navigate these obstacles, I leveraged online resources, particularly tutorials on Google and YouTube, to find effective solutions and gain deeper insights.

The integration testing phase revealed certain functional inconsistencies when different components were combined, alongside challenges in achieving expected behavior in the system's operations. These discoveries led me to a thorough review and experimentation with the code to identify and implement the necessary adjustments.

This phase of testing was instrumental in enhancing my problem-solving skills and ensuring the project's functionality and reliability were up to standard.

Improvements

1. **Enhanced User Interface:**
    - Improve the user interface of the web application and mobile app for easier navigation and better user experience.
    - Incorporate features like real-time parking availability updates, intuitive controls, and user-friendly notifications.
2. **Optimize RFID Detection:**
    - Fine-tune RFID reader sensitivity and placement to ensure reliable detection of RFID tags for seamless entry and exit.
    - Implement error handling mechanisms for cases of missed or faulty tag readings.

5.5 Test Cases

Test Case ID	Description	Input	Expected Result
TC01	User enters parking facility	User enters facility	System detects available parking spots
TC02	User searches for parking	User searches	System provides navigation to nearest available spot
TC03	User reserves parking space	User reserves spot	System confirms reservation and allocates spot
TC04	User exits parking facility	User exits facility	System updates parking availability status
TC05	Administrator monitors system	Administrator login	System provides real-time occupancy data
TC06	System experiences downtime	System error occurs	System fails to update parking availability
TC07	System faces connectivity issue	Network disconnected	System reconnects automatically when connection restored

Chapter 6: Results and discussion

6.1 Test Reports

Test Case ID	Description	Input	Expected Output	Actual Output	Status
TC01	User enters parking facility	User enters facility	System detects available parking spots	System correctly detects available spots	Pass
TC02	User searches for parking	User searches	System provides navigation to nearest spot	System successfully navigates to spot	Pass
TC03	User reserves parking space	User reserves spot	System confirms reservation and allocates	System confirms reservation and allocates	Pass
TC04	User exits parking facility	User exits facility	System updates parking availability status	System updates availability status	Pass
TC05	Administrator monitors system	Administrator login	System provides real-time occupancy data	System displays real-time occupancy data	Pass
TC06	System experiences downtime	System error occurs	System fails to update parking availability	System unable to update availability	Fail
TC07	System faces connectivity issue	Network disconnected	System reconnects when connection restored	System unable to reconnect	Fail

Preview of the UI(webpage) and IOT smart parking system

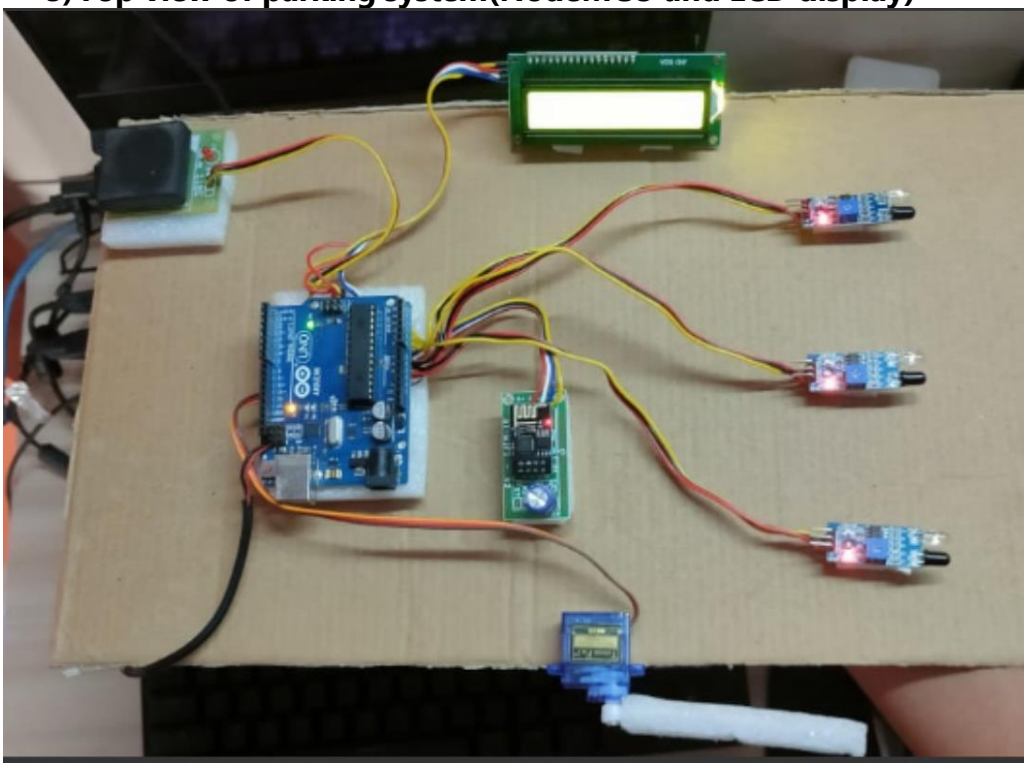
1) dashboard

IOT Based Car Parking SYS

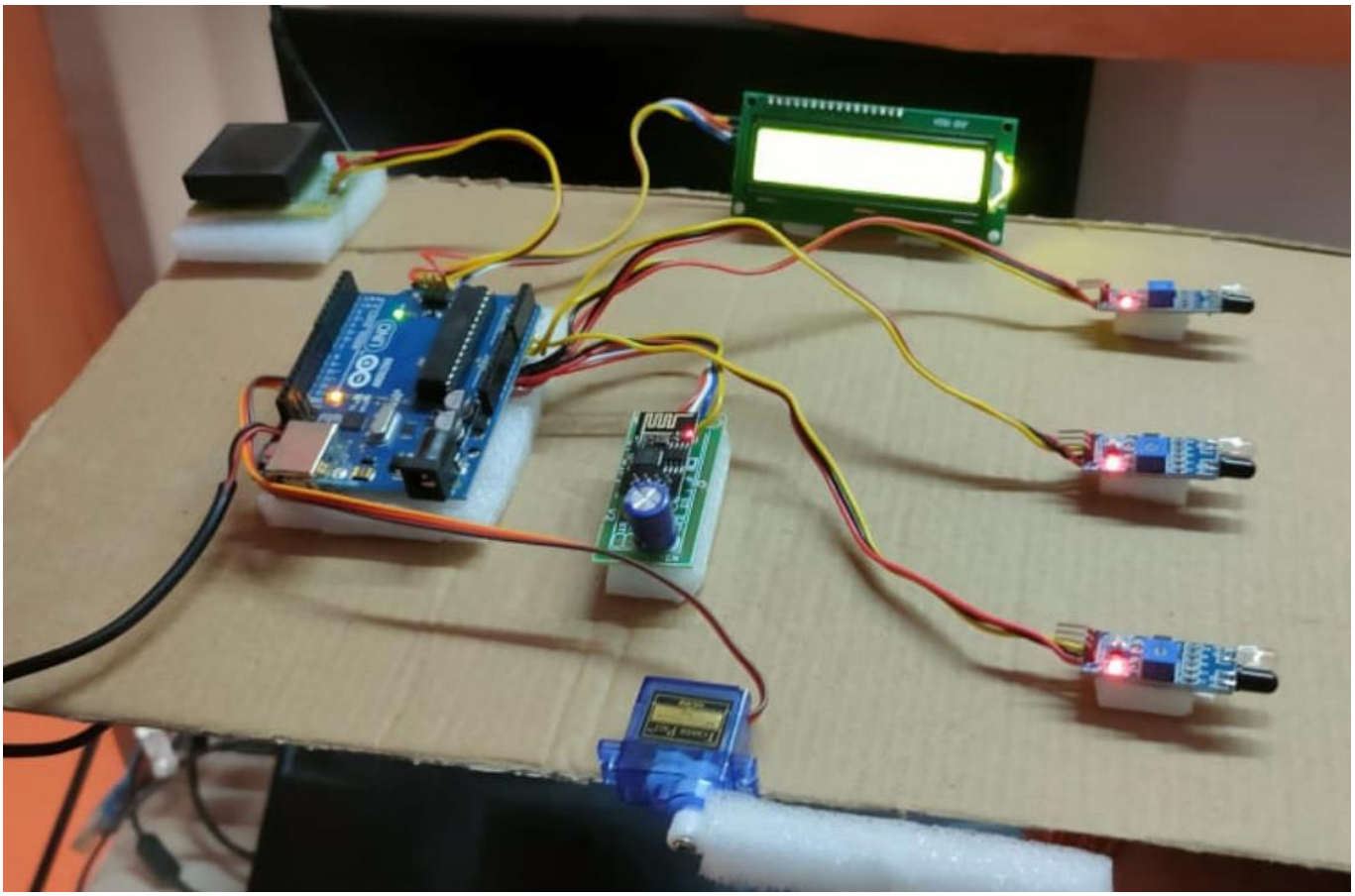
User ID	Mode	Time
101	Exite	01,Apr,2024 - 09:07:21 PM
102	Exite	01,Apr,2024 - 11:54:23 PM
103	Exite	01,Apr,2024 - 09:19:22 PM

Parking Slot No	Msg
Slot 1	M
Slot 2	M
Slot 3	M
< ---------->	

3) Top view of parking system(NodeMCU and LCD display)



4) Overall view of parking system



5) Dashboard(Adafruit feed logs)

User ID	Mode	Time
101	Exite	01,Apr,2024 - 09:07:21 PM
102	Exite	02,Apr,2024 - 05:37:40 PM
103	Exite	01,Apr,2024 - 09:19:22 PM

Parking Slot No	Msg
Slot 1	M
Slot 2	M
Slot 3	M
< ---------- >	

6.2 User documentation

1)Introduction:

Welcome to the user documentation for the IoT smart parking system. This guide will walk you through the software and hardware components of the pet feeder, along with their functions and working. By following this document, you'll gain a clear understanding of how to operate and maintain your IoT smart parking system effectively.

2)Software Overview:

The software component of the IoT Pet Feeder includes:

1) User Interface (UI): This is the system's front end, allowing users to interact visually. It displays parking availability.

2) Backend Services: These manage the system's core functions, like tracking parking space occupancy and ensuring smooth communication between hardware and servers.

3) Communication Protocols: Enable data exchange between system components, like MQTT for IoT devices and HTTP for user interfaces..

4) Security Features: Includes user authentication, access control to protect data and system integrity.

3)Hardware Overview:

The hardware components of the IoT Pet Feeder are:

1) **RFID Readers:** These devices read RFID tags/cards attached to vehicles for user identification and access control.
2) **NodeMCU ESP8266:** Acts as the main communication hub, facilitating data exchange between the system's components and the backend server over Wi-Fi.
3) **Arduino Microcontroller:** Controls the overall operation of the system, including processing user input, managing sensor data, and controlling servo motors for gate/barrier operations.
4) **Servo Motors:** Control entry/exit barriers or gates to regulate vehicle access to parking spaces.
5) **16x2 LCD Display:** Provides real-time information to users, such as available parking spots and navigation instructions.
6) **Power Supply:** Provides the necessary electrical power to operate the system's components.
7) **IR Sensors:** Ultrasonic or infrared sensors detect the presence of vehicles in parking spaces, enabling accurate occupancy monitoring.

3) Working of the Software

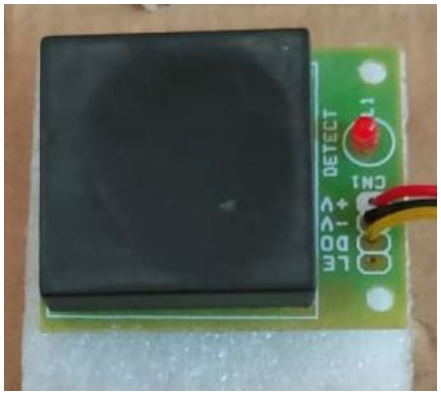
1. **Initialization:** Upon startup, the software initializes all necessary components, including establishing connections with RFID readers, IR sensors, NodeMCU ESP8266, and Arduino microcontroller.
2. **User Authentication:** When a vehicle approaches the parking facility, the RFID reader scans the RFID tag attached to the vehicle. The software then verifies the RFID tag against a database of authorized users to grant access.
3. **Occupancy Monitoring:** IR sensors continuously monitor the occupancy status of each parking space. The software collects data from these sensors to determine the availability of parking spots in real-time.
4. **Parking Space Allocation:** If a parking space is available, the software allocates it to the approaching vehicle. This information is communicated to the user through the LCD display or a mobile/web application.
5. **Entry/Exit Control:** Upon allocation of a parking space, servo motors control entry/exit barriers or gates to allow the vehicle to enter the parking facility. Similarly, when the vehicle exits, the barriers/gates are operated to regulate traffic flow.
6. **Data Logging and Analysis:** The software logs various data points, including occupancy status, entry/exit times, and user authentication logs. This data is analyzed to generate reports on parking usage patterns and trends.
7. **Communication with Backend Server:** The NodeMCU ESP8266 communicates with a backend server over Wi-Fi to transmit data collected from the parking facility. This enables remote monitoring, management, and analysis of parking data.
8. **User Interface:** The software provides a user-friendly interface, either through an LCD display or a mobile/web application, allowing users to view real-time parking availability, reserve parking spots, and receive navigation instructions.

1) Components and Screenshots:

1) NodeMCU ESP8266:

This screenshot shows the NodeMCU connected to Wi-Fi

2) RFID Reader



3) 16x2 LCD Module:

This screenshot displays the current parking space availibility

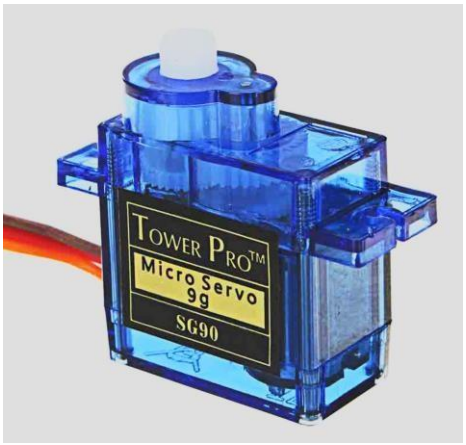
3) Website Dashboard:

User ID	Mode	Time
101	Exite	01,Apr,2024 - 09:07:21 PM
102	Exite	02,Apr,2024 - 05:37:40 PM
103	Exite	01,Apr,2024 - 09:19:22 PM

Parking Slot No	Msg
Slot 1	M
Slot 2	M
Slot 3	M
< ---------- >	

This screenshot shows the website dashboard with live updated data.

4) Servo motor:



This screenshot showcases the servo motor component of the IoT smart parking system,illustrating its integration into the system.

5) IR sensor

6) Arduino uno

4) Conclusion:

With this user documentation, you now have a comprehensive understanding of the Smart Parking System's software and hardware components, along with its functionality. Follow the instructions provided to effectively manage parking operations and optimize space utilization. Ensure smooth entry and exit processes for vehicles using the system's features. In case of any issues, refer back to this documentation or seek assistance from online resources to resolve them promptly.

Chapter 7: Conclusions

7.1 conclusion

The Smart Parking System utilizing Arduino, ESP8266, RFID technology, LCD display, and webpage integration represents a significant advancement in parking management. By leveraging IoT technologies, the project offers an efficient and user-friendly solution to address the challenges of traditional parking systems. Through real-time monitoring, automated entry/exit control, and user-friendly interfaces, the system enhances convenience for both drivers and parking facility managers.

7.1.1 significance of the system

The significance of the Smart Parking System lies in its ability to optimize parking space utilization and improve overall parking efficiency. By integrating various hardware components with web-based interfaces, the system streamlines parking operations, reduces congestion, and enhances user experience. Its potential to revolutionize urban parking infrastructure and contribute to smart city initiatives underscores its significance in modernizing urban transportation systems.

7.2 Limitations of system

Despite its innovative features, the Smart Parking System has limitations that need consideration. Its reliance on RFID technology for user authentication may pose challenges in cases of RFID card malfunctions or unauthorized access attempts. Additionally, the system's effectiveness may be affected by environmental factors such as signal interference or power outages, highlighting the need for robust backup systems and security measures.

7.3 Future scope of the project

The future scope of the Smart Parking System project offers opportunities for innovation and enhancement to meet evolving needs and challenges in parking management:

1) **New Areas of Investigation:**
    1) **New Areas of Investigation:**
       Smart Health Monitoring: Explore integrating health monitoring features such as weight tracking, activity monitoring, and even vital sign measurements for pets, providing comprehensive health insights.
    2) **Behavioral Analysis:**
       Investigate the potential for incorporating sensors or AI algorithms to analyze pet behavior patterns, helping owners understand their pets' moods, preferences, and overall well-being.
    3) **Environmental Sensing:**
       Consider adding environmental sensors to monitor factors like temperature, humidity, and air quality in the vicinity of the pet feeder, ensuring optimal living conditions for pets.
    4) **Remote Interaction:**
       Explore possibilities for enabling remote interaction with pets, such as audio or video communication through integrated cameras or speakers, fostering a closer bond between owners and their pets.

2) **Incomplete Work Due to Time Constraints/Problems Encountered:**
    1) **Enhanced Sensor Integration:**
       Develop and integrate advanced sensor technologies to improve detection accuracy and reliability, especially in challenging weather conditions or crowded parking lots.
    2) **Adaptive Pricing Mechanism:**
       Address the implementation of dynamic pricing algorithms to adjust parking fees based on factors such as demand, time of day, and special events, maximizing revenue and optimizing space utilization.
    3) **Data Security Measures:**
       Further enhance data security protocols to protect user information and prevent unauthorized access to sensitive parking data, ensuring compliance with privacy regulations and building trust with users.
    4) **Mobile Application Optimization:**
       Refine the mobile application interface and functionality to provide users with seamless access to parking information, intuitive navigation, and convenient payment options, enhancing the overall user experience and adoption rates.

Refrences

- https://ieeexplore.ieee.org/abstract/document/9792789
- https://www.youtube.com/@rahullOT
- https://www.techtarget.com/iotagenda/definition/IFTTT-If-This-Then-That
- https://hobbykits4u.com/uno_iot_24/car_parking.php

Glossary

This glossary provides definitions for acronyms, abbreviations, and terms used throughout the project report. If any additional terms are introduced in the report, they should be defined upon their first occurrence in the text, and if they are used extensively, they should be added to this glossary for clarity.

1. **Arduino:** An open-source electronics platform based on easy-to-use hardware and software, commonly used for creating interactive projects and prototypes.
2. **ESP8266**: A low-cost Wi-Fi microcontroller chip manufactured by Espressif Systems, widely used in IoT applications for its affordability and versatility.
3. **RFID (Radio-Frequency Identification**): A technology that uses electromagnetic fields to automatically identify and track tags attached to objects, facilitating contactless identification and access control.
4. **LCD (Liquid Crystal Display**): A flat-panel display technology that uses liquid crystals to produce images, commonly used in electronic devices such as digital clocks, calculators, and display screens.
5. **I2C (Inter-Integrated Circuit**): A synchronous serial communication protocol used for connecting multiple integrated circuits in embedded systems, allowing for efficient data transfer between devices.
6. **Webpage**: A document or resource accessed through the World Wide Web, typically containing information in the form of text, images, and multimedia, and designed for viewing in a web browser.
7. **IR Sensor (Infrared Sensor)**: A sensor that detects infrared radiation emitted or reflected by objects, commonly used for proximity sensing, motion detection, and object tracking.
8. Servo Motor: A type of motor that can be precisely controlled to rotate to a specific angle, commonly used in robotics, remote control systems, and industrial automation.