

Intelligent and Communicating Systems, ICS

2nd Year Specialty SIQ G02, 2CS SIQ2

Project report

Title:

Building a Smart Parking System

Studied by:

SOLTANI Meriem
HEDDADJI Nourelimane
TAIB Salma
ZERROUKI Selma

Contents

1	Introduction	5
2	Terminology	6
1.	Internet of Things (IoT)	6
2.	Raspberry Pi	6
3.	RFID Reader (RC522)	6
4.	Distance Sensor (HC-SR04)	7
5.	Servo Motor (SG90)	7
3	Literature Review	9
1.	Problem Statement	9
2.	Existing Systems Study	9
3.	Proposed Solution	10
4	System Description	11
1.	User-Centred Parking Process	11
1.1.	Mobile Application Reservation	11
1.2.	RFID Validation at Arrival	11
1.3.	Automated Gate Access	11
1.4.	Secure Entry and Time Extension	11
1.5.	Time Extension Feature	11
2.	System Architecture	12
5	Circuit	13
1.	Overall Description	13
2.	Circuit Components	13
3.	Circuit Connection	13

3.1.	RFID Reader	13
3.2.	Distance Sensor	14
3.3.	Servo Motor	15
4.	Main Function Overview	16
5.	System Overview	17
6	Mobile App Implementation	19
1.	Technology Stack	19
2.	Functionalities	19
3.	User Interface	20
3.1.	User Authentication	20
3.2.	Parking Reservation	22
3.3.	Extend Booking Time	24
7	Conclusion	25
A	Appendix	26
1.	Setting up GPIO Pins	26
2.	Database Connection	26
3.	RFID Access Control	27
4.	Gate Control	28
5.	Main Function	29

List of Figures

2.1	Raspberry Pi	6
2.2	RC522 RFID	7
2.3	Distance Sensor (HC-SR04)	7
2.4	Servo Motor (SG90)	8
4.1	System Architecture	12
5.1	RFID reader circuit connection	14
5.2	Servo motot circuit connection	15
5.3	Servo motor circuit connection	16
5.4	System Overview - Smart Parking Implementation	18
6.1	Main Page	20
6.2	Login Page	20
6.3	Forgot Password Page	21
6.4	Parking History Page	21
6.5	Parking Map Page	22
6.6	Booking Details	22
6.7	Payment Page	23
6.8	Booking Summary Page	23
6.9	Parking Time Page	24
6.10	Extend Time page	24

Introduction

In the ever-evolving landscape of technological advancements, the integration of Internet of Things (IoT) solutions has significantly reshaped traditional systems across diverse domains, such as home automation, traffic management, and more, contributing to an enhanced quality of life and simplifying tiring and time-consuming tasks.

Motivated by this paradigm shift, we undertook the challenge of developing an intelligent parking system designed to automatically guide users to available parking spaces at the lowest cost through a dedicated application. Our focus goes beyond mere convenience for users; we aim to contribute to the broader objective of making parking facilities more intelligent and resource-efficient.

This report presents a comprehensive study on the implementation of our innovative parking system. Through this exploration, we delve into the details of our approach, detailing the key components, methodologies employed, and the envisaged benefits of our IoT-driven solution.

Terminology

In this section, we explore essential terms and concepts integral to our smart parking system. A grasp of these terms is vital for unlocking the core principles and functionalities of our project.

1. Internet of Things (IoT)

The Internet of Things (IoT) refers to the network of physical devices connected to the internet, allowing them to communicate, collect and exchange data, and interact with the surrounding environment.

2. Raspberry Pi

Raspberry Pi, a versatile single-board computer, serves as the heart of our smart parking system. Its compact design and affordability make it an ideal choice for seamlessly integrating diverse functionalities.

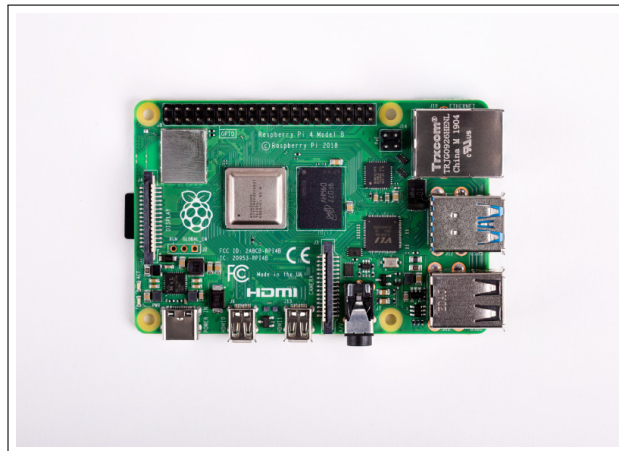


Figure 2.1: Raspberry Pi

3. RFID Reader (RC522)

An RFID reader is a device that uses Radio-Frequency Identification technology to communicate with RFID tags. In the context of our system, the RC522 RFID reader facilitates card identification for parking reservations.

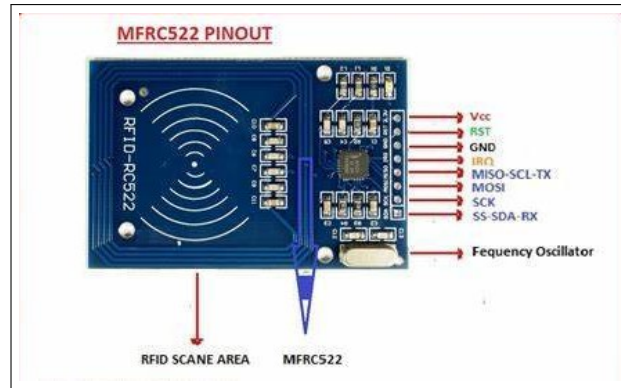


Figure 2.2: RC522 RFID

4. Distance Sensor (HC-SR04)

A sensor is a device that detects and measures physical quantities or environmental conditions.

The HC-SR04 distance sensor utilizes ultrasonic signals to measure the distance between the car and the gate. It plays a crucial role in determining when to open the gate for users with parking reservations.

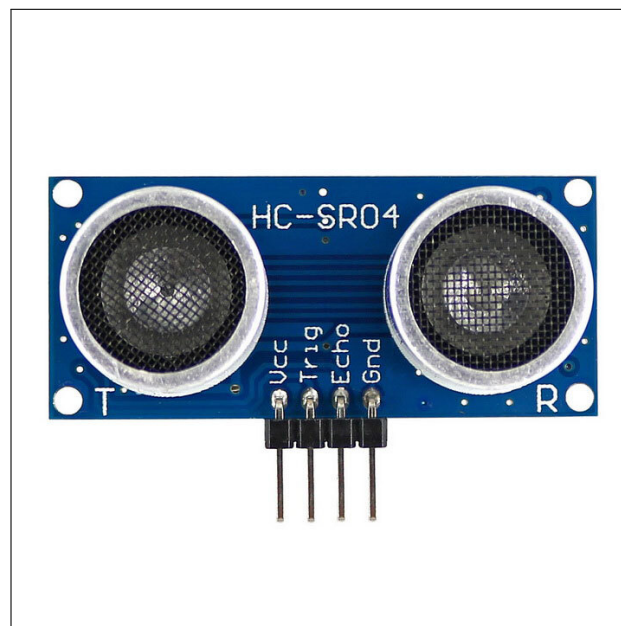


Figure 2.3: Distance Sensor (HC-SR04)

5. Servo Motor (SG90)

A servo motor, specifically the SG90 model, is employed to control the movement of the parking lot gate. It operates based on Pulse Width Modulation (PWM) signals to

open and close the gate automatically.



Figure 2.4: Servo Motor (SG90)

Literature Review

1. Problem Statement

As Algerian cities undergo rapid population growth, the corresponding rise in vehicle ownership leads to a surge in the number of vehicles competing for limited parking spaces.

This surge contributes to numerous challenges, including traffic congestion, heightened frustration among drivers, and increased fuel consumption and heightened air pollution due to extended search times for parking.

Furthermore, the lack of efficient parking solutions and smart parking systems intensifies these issues, particularly during peak work hours.

2. Existing Systems Study

To comprehensively understand the current parking landscape in Algeria, we conducted an in-depth study of existing parking systems. While some parking solutions are present, the extent of their smart capabilities and integration with mobile applications remains unclear.

- Most existing parking systems in Algeria primarily rely on traditional ticket-based approaches, with attendants manually managing entry and exit points.
- User interaction with existing parking systems appears to be conventional, relying on physical tickets and manual attendants.
- The absence of automated features impacts user experience.
- Challenges are evident during peak work hours, where the existing systems may struggle to efficiently manage the big influx of vehicles, contributing to increased frustration among users.
- Limited visibility into real-time information exacerbates congestion and frustration among drivers.
- Accessibility features for differently-abled individuals are not explicitly incorporated into the existing parking systems.
- There is limited information on how data generated by existing systems is managed, analyzed, and utilized for optimization.

In summary, the existing parking systems in Algeria face challenges related to manual operations, lack of automation, and inadequate accommodation of peak hours, impacting user experience and overall efficiency.

3. Proposed Solution

In response to the identified challenges surrounding existing parking solutions in Algeria, our project introduces a specialized smart parking system tailored to the unique needs of educational institutions.

Our solution envisions a seamlessly integrated system where teachers and students can efficiently interact with a dedicated mobile application, enhancing the overall user experience. This application interfaces with an intelligent Raspberry Pi-based system that oversees the parking infrastructure, effectively allocating parking slots to teachers and students in real-time. Drawing inspiration from global best practices, our proposed smart parking system aims to not only streamline the parking experience but also set a new standard for parking solutions in educational institutions in Algeria.

While our implementation focuses on a school scenario for practical demonstration, it is important to note that the underlying technology and principles can be generalized to address parking challenges in various settings beyond educational institutions.

System Description

1. User-Centred Parking Process

1.1. Mobile Application Reservation

Users begin by accessing our user-friendly mobile application, the primary interface for our Smart Parking System.

Through this app, users reserve parking slots for specified durations, with e-payment functionality integrated.

Crucial details of the booking, including payment confirmation, are securely stored in our database.

1.2. RFID Validation at Arrival

Upon arriving at the parking facility, the system employs RFID technology. A Raspberry Pi-controlled RFID ID reader quickly checks with the database to make sure the user's reservation is valid, making entering hassle-free.

1.3. Automated Gate Access

As users with a valid parking reservation approach the gate, a distance sensor managed by the Raspberry Pi measures how close the car is to the gate. When the user is close enough, the gate opens automatically. At the same time, the app starts a countdown timer, showing how much time is left for parking.

1.4. Secure Entry and Time Extension

The gate opens to let the user's vehicle into the parking area and then closes automatically, making sure only authorized users get in.

1.5. Time Extension Feature

To offer added flexibility, users have the option to extend the duration of their parking reservation. This feature allows users to adjust their parking time based on their needs, providing a convenient and user-friendly experience.

2. System Architecture

The illustration below depicts the interconnected elements of our Smart Parking System:

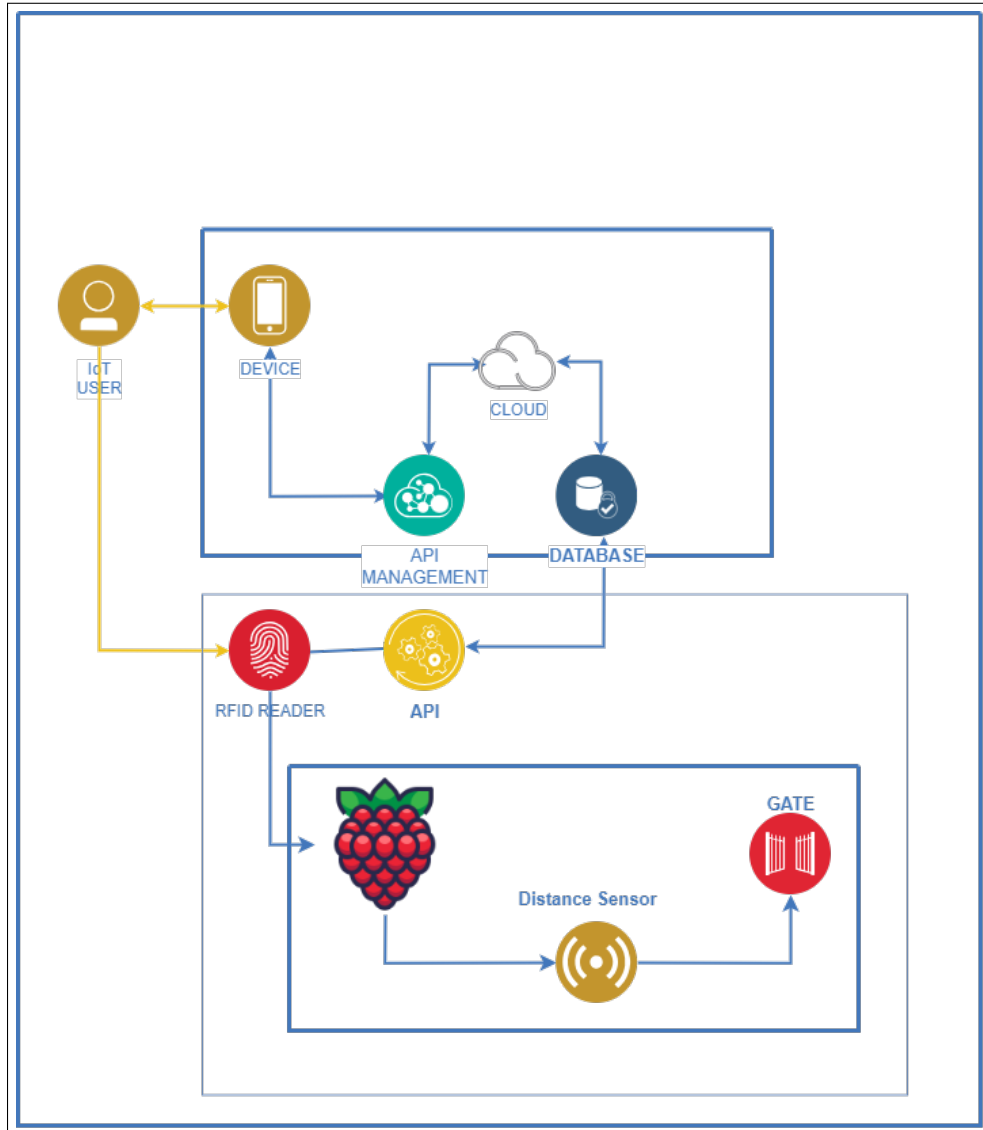


Figure 4.1: System Architecture

Circuit

1. Overall Description

Our goal is to integrate an RFID ID reader, a servo motor, and a distance sensor with a Raspberry Pi minicomputer to manage the parking lot gate based on card identification by the RFID reader and the car's proximity to the gate.

2. Circuit Components

- Raspberry Pi 4
- Distance Sensor HC-SR04
- Servo Motor SG90
- RFID Reader RC522
- Jump wires
- Breadboard
- Resistors

3. Circuit Connection

3.1. RFID Reader

The RFID reader is crucial for card identification, allowing seamless entry for users with valid reservations. Connect the components as follows:

- Connect the 3v3 pin of the RC522 to Raspberry Pi pin 1 (3v3).
- Connect the RST of the RC522 to Raspberry Pi pin 22.
- Connect the GND of the RC522 to Raspberry Pi to pin 6.
- Connect the MISO pin of the RC522 to Raspberry Pi 21.
- Connect the MOSI pin of the RC522 to of Raspberry Pi pin 19.
- Connect the SCK pin of the RC522 to Raspberry Pi pin 23.
- Connect the SDA pin of the RC522 to Raspberry Pi pin 24.

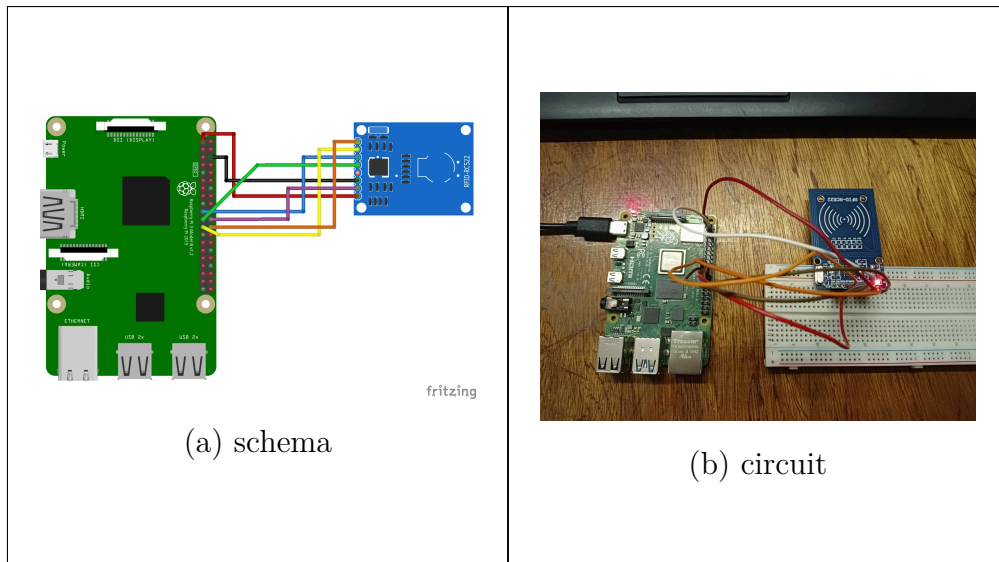


Figure 5.1: RFID reader circuit connection

3.2. Distance Sensor

The distance sensor is vital for determining the proximity of the car to the gate. Connect the components as follows:

- Connect the VCC pin of the HC-SR04 to Raspberry Pi pin 2 (5v).
- Connect the GND pin of the HC-SR04 to Raspberry Pi pin 6.
- Connect the trig pin of the HC-SR04 to Raspberry Pi pin 7.
- Connect one end of the first resistor (1kohm) to the echo pin of the HC-SR04.
- Connect the other end of the first resistor to Raspberry Pi pin 11.
- Connect one end of the second resistor (2kohm) to Raspberry Pi pin 11.
- Connect the other end of the second resistor to Raspberry Pi pin 6 (GND).

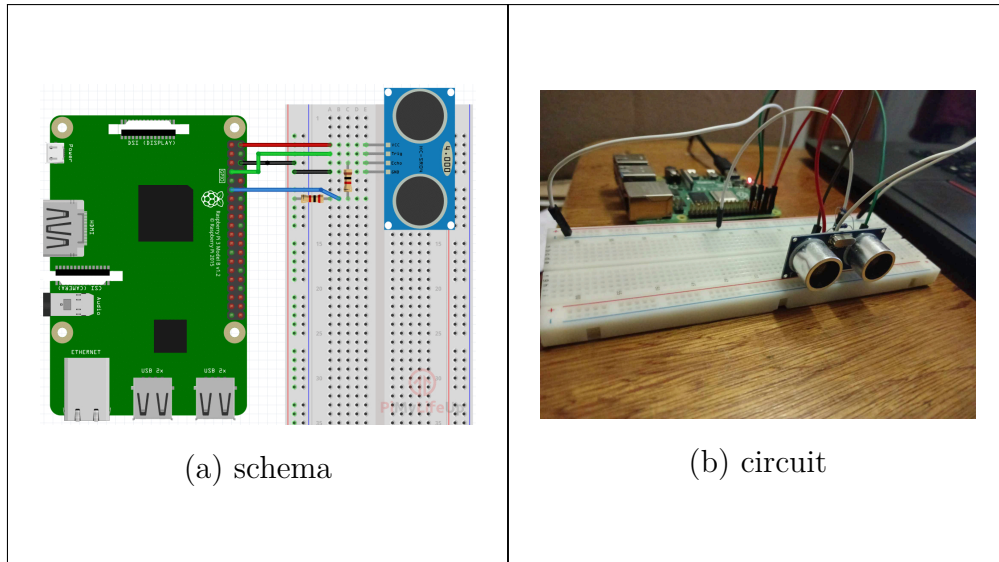


Figure 5.2: Servo motor circuit connection

3.3. Servo Motor

The servo motor controls the gate's movement, allowing for automated entry and exit. Connect the components as follows:

- Connect the VCC pin (red wire) of the SG90 to Raspberry Pi pin 2 (5v).
- Connect the GND pin (brown wire) of the SG90 to Raspberry Pi pin 6.
- Connect the signal pin (orange wire) of the SG90 of Raspberry Pi pin 12.

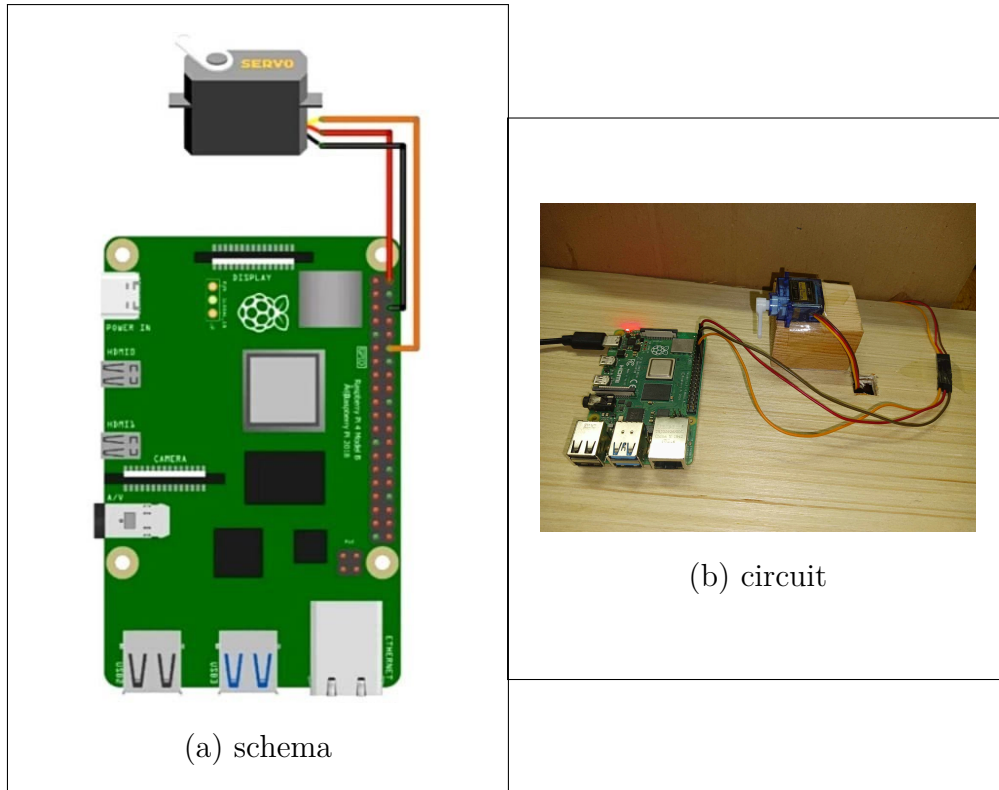


Figure 5.3: Servo motor circuit connection

4. Main Function Overview

The main function encapsulates the core logic of our smart parking system. While the comprehensive code details can be referred to in the appendix, this section provides an overview of the primary functionalities:

- `read_rfid()` is called in a loop to continually check and read RFID tags from users approaching the gate.
- `check_pending_bookings(rfid)` is invoked to determine if the user has an ongoing and valid parking reservation.
- Upon a valid booking, `update_active_booking(rfid)` is called to transition the booking status from pending to active, triggering the countdown timer in the mobile app.
- The functions `measure_distance()` and `control_gate()` are utilized to manage the opening and closing of the gate based on the proximity of the user's vehicle.

The following code snippet provides a condensed representation of the main function:

```

1 if __name__ == "__main__":
2     try:
3         while True:

```



```
4      # Read RFID tag
5      rfid = read_rfid()
6      if rfid:
7          # Check if the booking is pending
8          data = check_pending_bookings(rfid)
9          # Check if the booking is valid
10         isBookingValid = False
11         if data:
12             isBookingValid = check_valid_booking(data[0])
13         if isBookingValid:
14             # Activate the distance sensor
15             calculateDistance = True
16             while calculateDistance:
17                 distance = measure_distance()
18                 # Open the gate
19                 if distance and distance < 10:
20                     calculateDistance = False
21                     control_gate()
22                     update_active_booking(rfid)
23                 time.sleep(1)
24             time.sleep(1)
25
26     except Exception as e:
27         print("Error main:", e)
```

Listing 5.1: Main Function

5. System Overview

To provide a comprehensive view of our Smart Parking System, the following images showcase the practical implementation of the entire system:

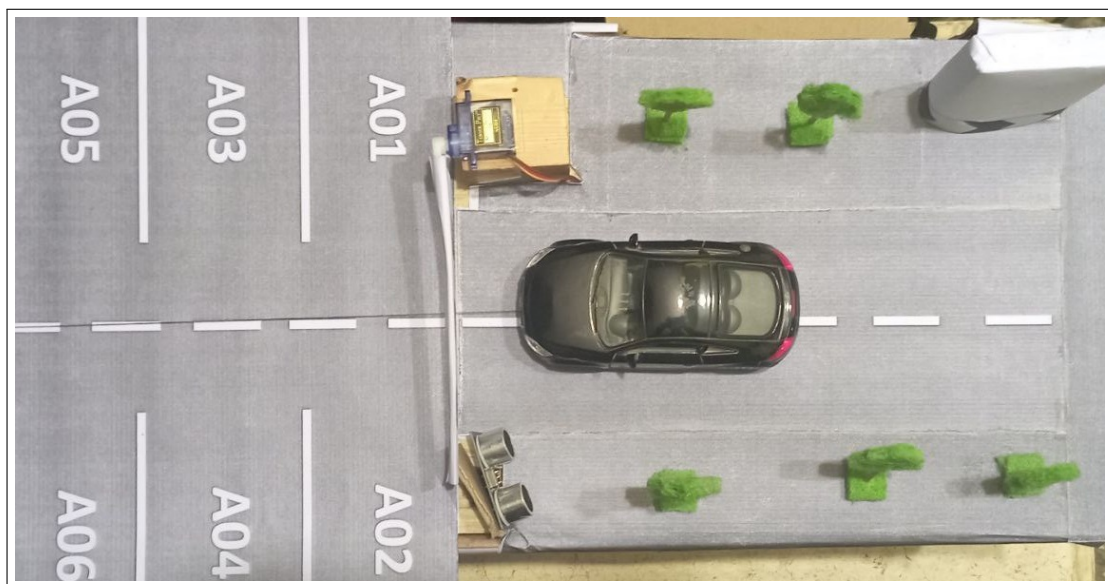


Figure 5.4: System Overview - Smart Parking Implementation

Mobile App Implementation

1. Technology Stack

Our chosen technology tools are :

- **Flutter:** A versatile framework for building cross-platform mobile applications, allowing us to provide a unified experience for both Android and iOS users.
- **Express.js :** A lightweight and flexible Node.js framework that handles the server-side logic, ensuring efficient communication between the mobile app and the database.
- **Supabase :** Our database solution, offering a secure and scalable platform. It simplifies database management, providing real-time data synchronization and seamless integration with Express.js.

2. Functionalities

- **User Authentication:** Secure user authentication for account management.
- **Parking Reservation:** Users can reserve parking slots through the app.
- **E-Payment:** Integration for e-payment to streamline the booking process.
- **Real-time Updates:** Users receive real-time updates on their parking status and remaining time.
- **Extension:** Users can extend the time of their booking.

3. User Interface

3.1. User Authentication

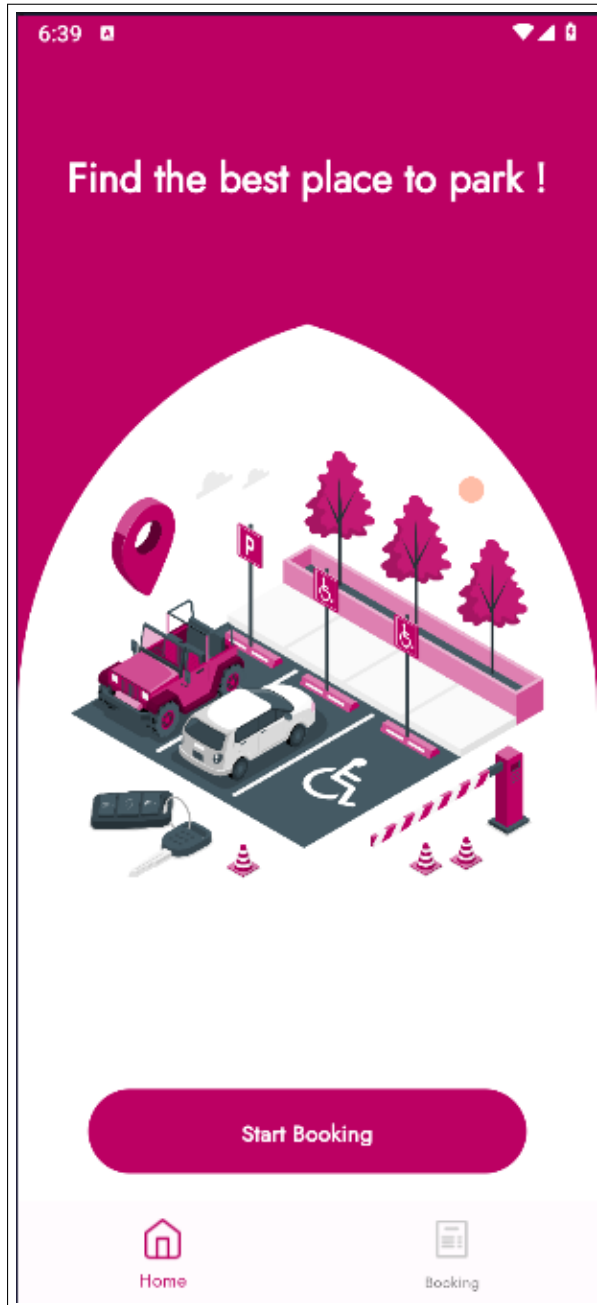


Figure 6.1: Main Page

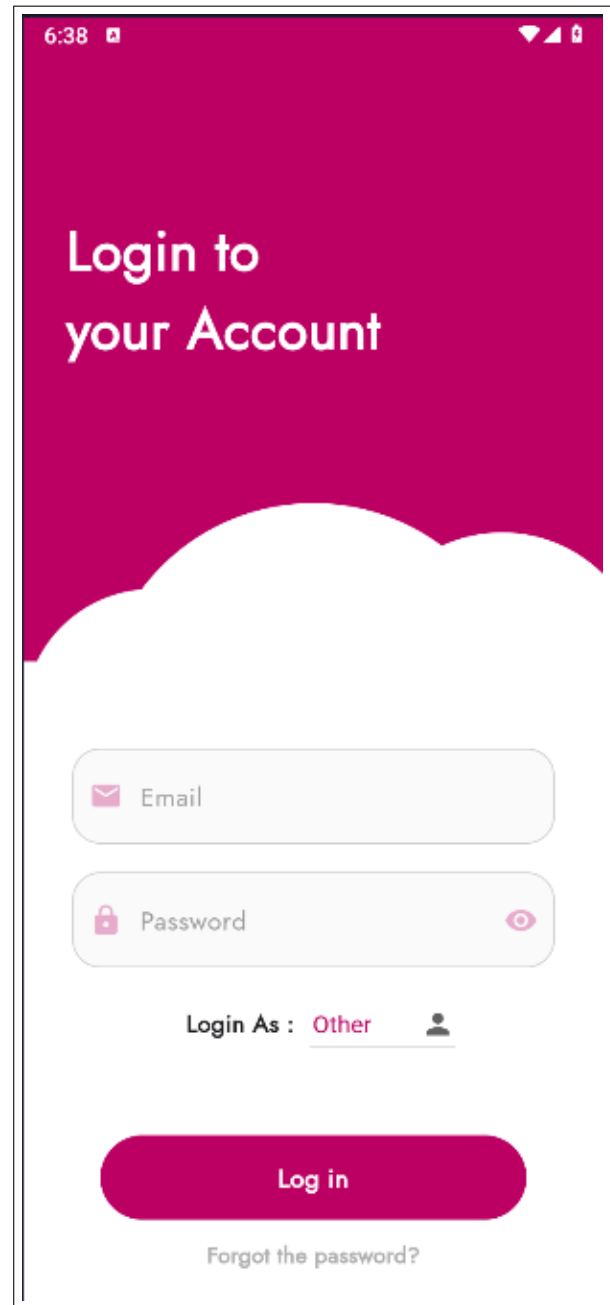


Figure 6.2: Login Page

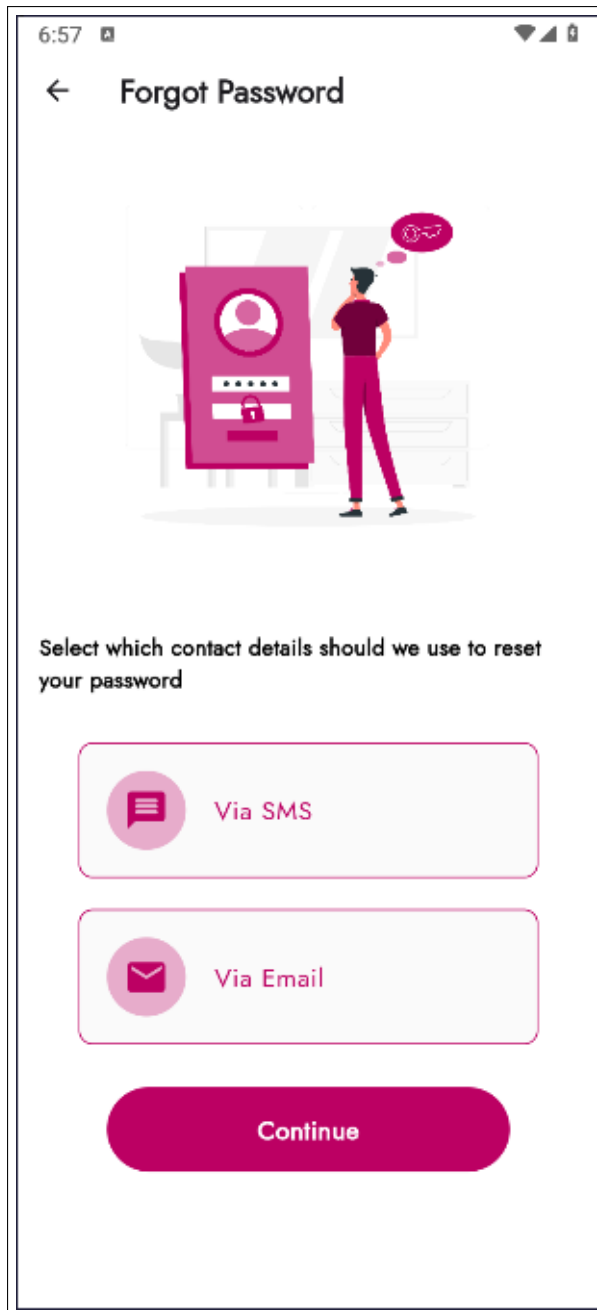


Figure 6.3: Forgot Password Page

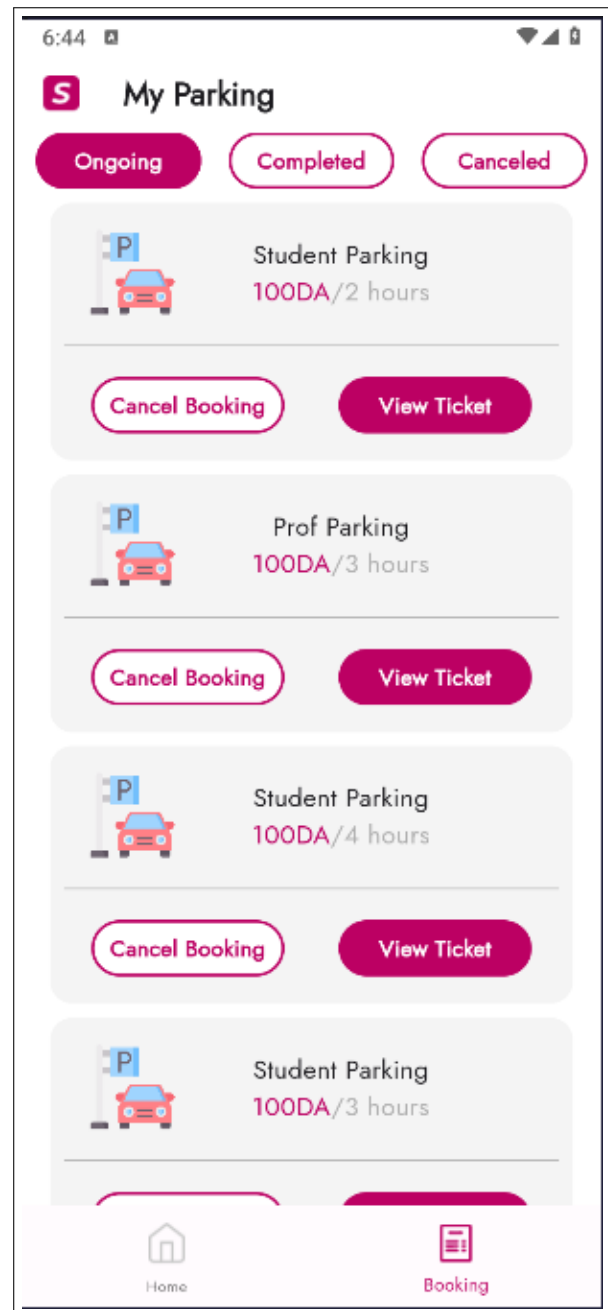


Figure 6.4: Parking History Page

3.2. Parking Reservation

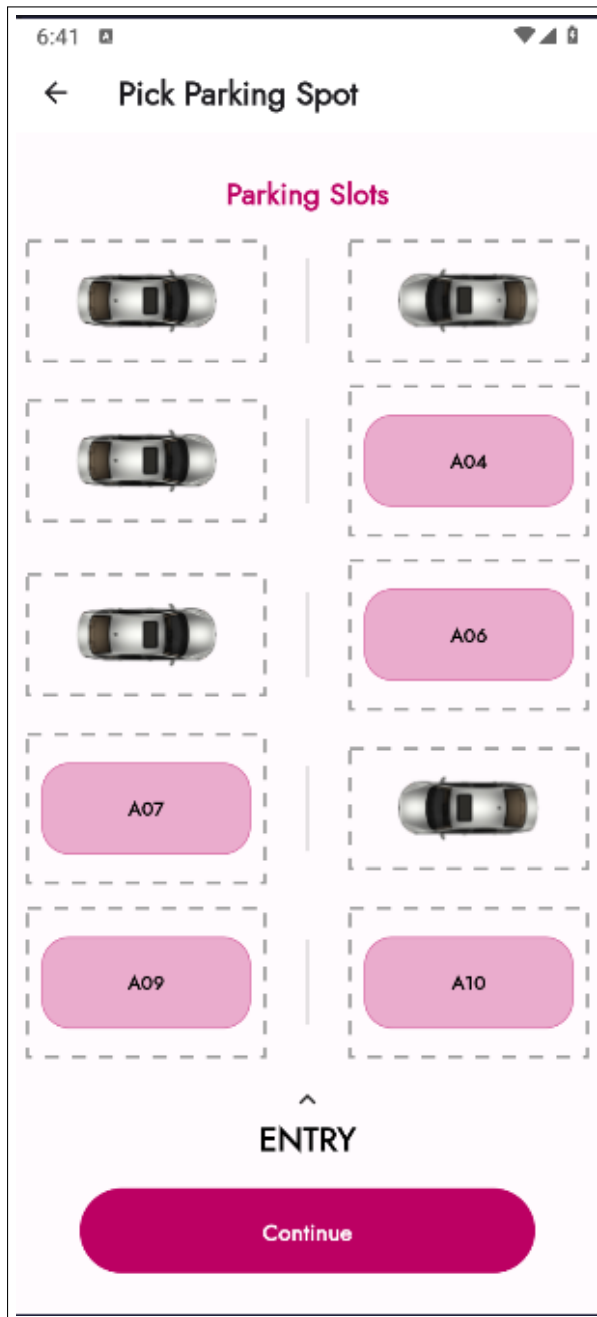


Figure 6.5: Parking Map Page

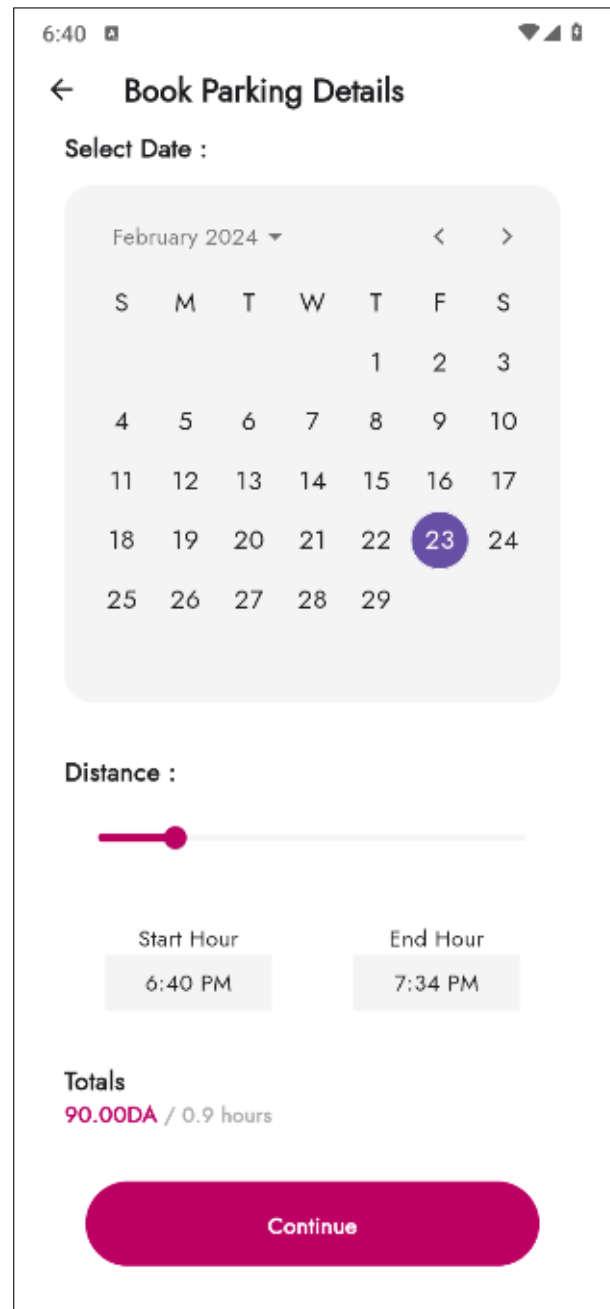




Figure 6.6: Booking Details

6:41

← Payment

Choose Payment Methods :

 Carte Dhahabia ☐

 Carte CIB ☐

Continue

Figure 6.7: Payment Page

6:41

← Review Summary

Parking Area	Parking Lot of Son Manolia
Address	9569, trantow Courts
Vehicle	Toyota Land Cru (AFD 6397)
Parking Spot	1st Floor (A05)
Date	May 11, 2023
Duration	4 hours
Hours	09:00 AM - 13:00 PM

Amount	8.00
Taxes & Fees (10%)	0.8
<hr/>	
Total	8.08

Total 8.08

Continue

Figure 6.8: Booking Summary Page

3.3. Extend Booking Time

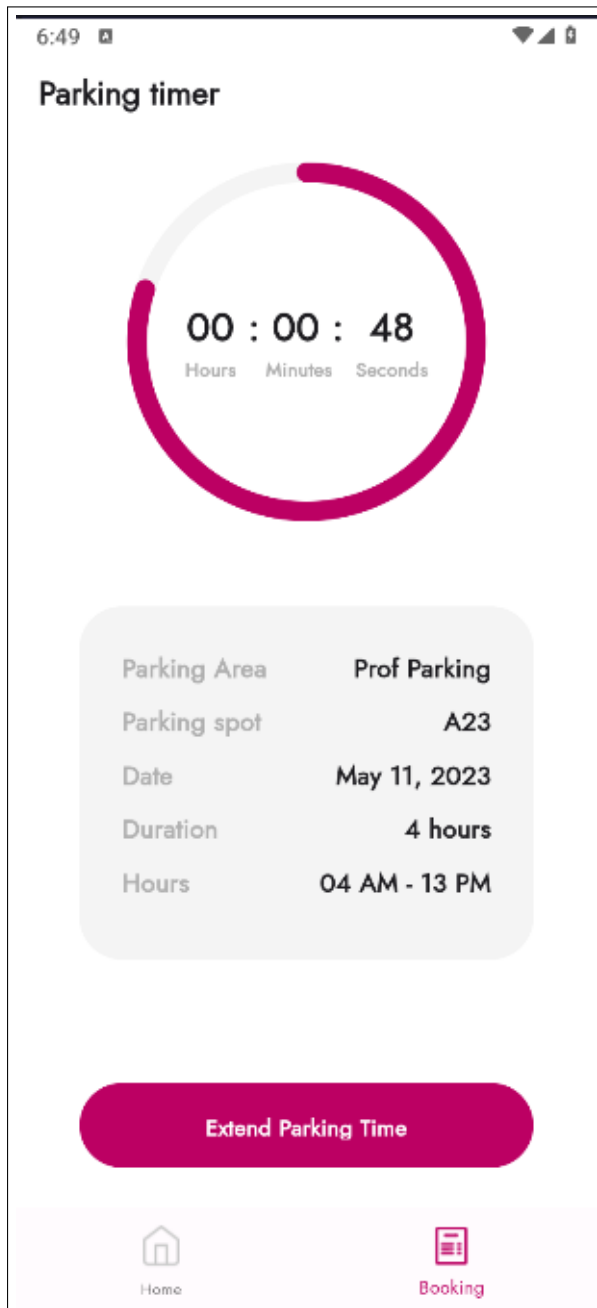


Figure 6.9: Parking Time Page

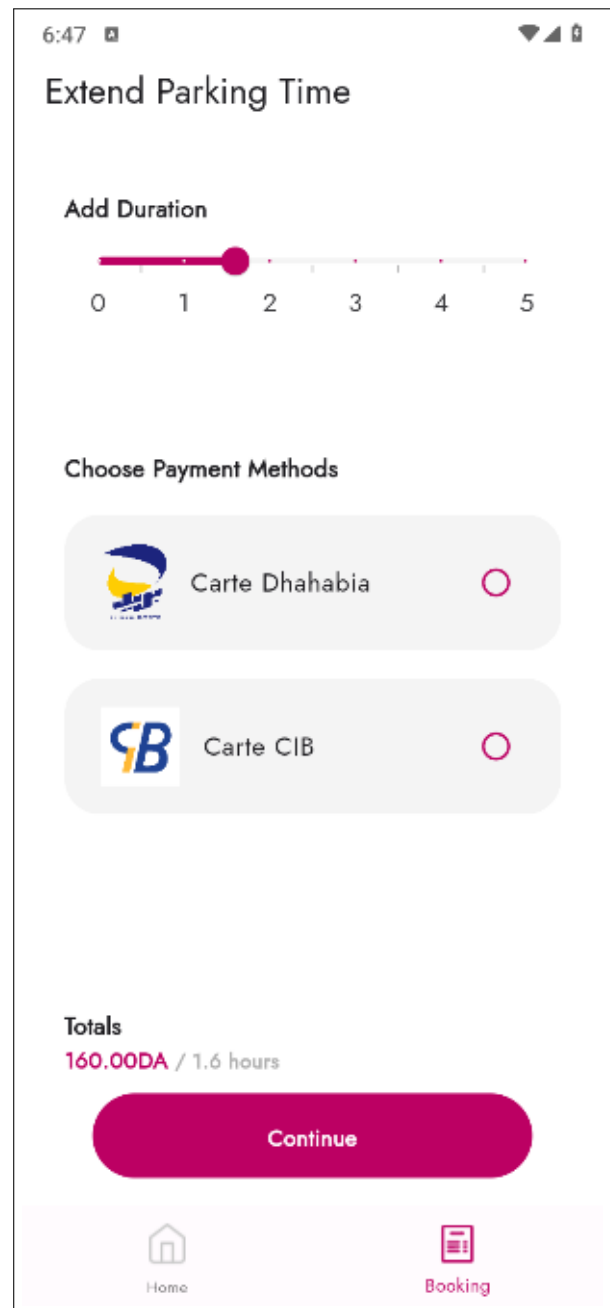


Figure 6.10: Extend Time page

Conclusion

In conclusion, the development of our intelligent parking system has been a journey marked by challenges, innovation, and a commitment to improving urban mobility.

The implementation of RFID technology, distance sensors, and servo motors has resulted in a robust and secure parking solution. Users can effortlessly navigate the application to reserve parking slots, while our system, driven by a Raspberry Pi-based infrastructure, ensures smooth entry and exit procedures.

While initially designed to meet the parking challenges within educational institutions, it is important to emphasize the adaptability and scalability of our solution. The modular and scalable design of our system enables seamless integration into various urban environments, making it applicable beyond educational institutions. Our technology and principles can be easily generalized, offering a versatile solution to address parking challenges in diverse settings.

Looking to the future, our project perspectives open avenues for further refinement and expansion. We aspire to explore additional features for the mobile app, providing users with a more personalized and dynamic parking experience. Investigating alternative user authentication systems, like camera recognition, presents exciting possibilities for enhanced security and convenience.

In essence, our smart parking system represents a contribution to the evolution of intelligent urban environments. Our commitment to continuous improvement and innovation drives us to explore new horizons and embrace the possibilities that lie ahead.

Appendix

In this appendix, we will be presenting mainly code portions

1. Setting up GPIO Pins

This code snippet initializes and configures the GPIO pins on the Raspberry Pi for various components of our smart parking system. Specifically, it defines constants for the trigger pin, echo pin, and servo motor pin. The GPIO pins are set up using the RPi.GPIO library.

```
1  import RPi.GPIO as GPIO
2
3
4  # Global Constants
5  PIN_TRIGGER = 7
6  PIN_ECHO = 11
7  SERVO_PIN = 12
8
9  # Set up GPIO pins
10 GPIO.setmode(GPIO.BOARD)
11 GPIO.setup(PIN_TRIGGER, GPIO.OUT)
12 GPIO.setup(PIN_ECHO, GPIO.IN)
13 GPIO.setup(SERVO_PIN, GPIO.OUT)
14
15 # Set up HC-SR04 sensor
16 GPIO.setup(PIN_TRIGGER, GPIO.OUT)
17 GPIO.setup(PIN_ECHO, GPIO.IN)
18
19 # Set up servo motor
20 GPIO.setup(SERVO_PIN, GPIO.OUT)
21 servo = GPIO.PWM(SERVO_PIN, 50)
22 servo.start(0)
```

Listing A.1: Setting up GPIO Pins

2. Database Connection

This code establishes a connection to the Supabase database, our cloud database solution. The `create_client` function from the `supabase` library is used to create a client, and the connection parameters (URL and key) are provided.

```
1  from supabase import create_client
2
3
4  # Set up Supabase connection
5  url = 'https://bgynecsgqrdfrkcyshsz.supabase.co'
6  key = 'your_supabase_key'
7  supabase = create_client(url, key)
```

Listing A.2: Supabase Connection

3. RFID Access Control

This section of code handles RFID access control.

- The `read_rfid()` function uses a SimpleMFRC522 reader to capture RFID card details.
- The `check_pending_bookings` function queries the Supabase database to identify pending bookings for a specific RFID card (user).
- The `check_valid_booking` function ensures the validity of a booking based on date and time, and `update_active_booking` updates the booking status in the database.

```

1  def read_rfid():
2      try:
3          reader = SimpleMFRC522()
4          id, _ = reader.read() # Ignore the text, only capture the ID
5          return id
6      finally:
7          pass
8
9
10 # Function to check pending bookings
11 def check_pending_bookings(rfid):
12     try:
13         response = supabase.table("booking").select("*").eq("rfid",
14 rfid).eq("status", "pending").execute()
15         data = response.data
16         return data
17     except Exception as e:
18         print("Error checking pending bookings:", e)
19         return None
20
21 # Function to check if the booking is valid
22 def check_valid_booking(booking):
23     try:
24         # Extract booking details
25         start_time = booking['start_time']
26         end_time = booking['end_time']
27         booking_date = booking['date']
28
29         # Get current date and time
30         current_date = datetime.datetime.now().date()
31         current_time = datetime.datetime.now().time()
32
33         # Convert start_time and end_time strings to time objects
34         start_time_obj = datetime.datetime.strptime(start_time, '%H:%M
35 :%S').time()

```

```

34         end_time_obj = datetime.datetime.strptime(end_time, '%H:%M:%S')
        .time()
35         booking_date_obj = datetime.datetime.strptime(booking_date, '%Y
-%m-%d').date()
36
37         # Check if the booking date is the current date
38         if booking_date_obj == current_date:
39             # Check if the current time is between start_time and
end_time
40             if start_time_obj <= current_time <= end_time_obj:
41                 return True
42             else:
43                 return False
44         else:
45             return False
46     except Exception as e:
47         print("Error checking valid booking:", e)
48         return False
49
50     # Function to update the booking status in supabase
51     def update_active_booking(rfid):
52         try:
53             # Get the current time
54             current_time = datetime.datetime.now().isoformat()
55             # Perform the update operation in the Supabase table
56             response = supabase.table("booking").update({"status": "active"
, "check_in_time": current_time}).eq("rfid", rfid).execute()
57         except Exception as e:
58             print("Error updating booking status:", e)

```

Listing A.3: RFID Access Control

4. Gate Control

This portion of the code includes functions to measure the distance using an ultrasonic sensor (HC-SR04) and control the parking gate using a servo motor (SG90).

The `measure_distance()` function calculates the distance between the sensor and the vehicle, and `control_gate()` orchestrates the opening and closing of the gate.

```

1     # Function to measure distance
2     def measure_distance():
3         try:
4             GPIO.output(PIN_TRIGGER, GPIO.HIGH)
5             time.sleep(0.00001)
6             GPIO.output(PIN_TRIGGER, GPIO.LOW)
7
8             while GPIO.input(PIN_ECHO) == 0:
9                 pulse_start_time = time.time()
10            while GPIO.input(PIN_ECHO) == 1:
11                pulse_end_time = time.time()
12
13            pulse_duration = pulse_end_time - pulse_start_time
14            distance = round(pulse_duration * 17150, 2)
15            return distance

```

```

16         except Exception as e:
17             print("Error measuring distance:", e)
18             return None
19
20 # Function to control the gate using the servo motor
21 def control_gate():
22     try:
23         # Open gate
24         servo.ChangeDutyCycle(0)
25         time.sleep(1)
26         servo.ChangeDutyCycle(2)
27         time.sleep(1)
28         servo.ChangeDutyCycle(4)
29         time.sleep(1)
30         servo.ChangeDutyCycle(6)
31         time.sleep(1)
32         servo.ChangeDutyCycle(8)
33         time.sleep(6)
34         # Close gate
35         servo.ChangeDutyCycle(7)
36         time.sleep(1)
37         servo.ChangeDutyCycle(5)
38         time.sleep(1)
39         servo.ChangeDutyCycle(3)
40         time.sleep(1)
41         servo.ChangeDutyCycle(1)
42         time.sleep(1)
43         servo.ChangeDutyCycle(0)
44     except Exception as e:
45         print("Error opening gate:", e)

```

Listing A.4: Gate Control

5. Main Function

The `main` function orchestrates the overall functionality of the smart parking system. It continuously reads RFID tags, checks for pending bookings, validates the booking, activates the distance sensor, opens the gate, and updates the booking status.

```

1 # Main function
2 if __name__ == "__main__":
3     try:
4         while True:
5             # Read RFID tag
6             rfid = read_rfid()
7             if rfid:
8                 # Check if the booking is pending
9                 data = check_pending_bookings(rfid)
10                # Check if the booking is valid
11                isBookingValid = False
12                if data:
13                    isBookingValid = check_valid_booking(data[0])
14                if isBookingValid:
15                    # Activate the distance sensor
16                    calculateDistance = True

```

```
17         while calculateDistance:
18             distance = measure_distance()
19             # Open the gate
20             if distance and distance < 10:
21                 calculateDistance = False
22                 control_gate()
23                 update_active_booking(rfid)
24
25             time.sleep(1)
26         time.sleep(1)
27
28     except Exception as e:
29         print("Error main:", e)
```

Listing A.5: Main Function

This structured explanation provides a clear understanding of each code section's purpose and functionality within the smart parking system.