

Ministry of High Education,
Culture and Science City in 6th of October,
The High Institute of Computer Science and Information Systems



المعهد العالي لعلوم الحاسب ونظم المعلومات

Graduation Project:

Smart Parking

Assistant:

Eng. Khalid Rayhan

Supervised By:

Dr. Usama Tharwat Elhagari

Project No.: N42226

Academic Year: 2021/2022

Ministry of High Education,
Culture and Science City in 6th of October,
The High Institute of Computer Science and Information Systems



المعهد العالي لعلوم الحاسب ونظم المعلومات

Graduation Project:

Smart Parking

Prepared by:

41001 – أبانوب عاطف وهيب عبد الملاك	42005 – بلال أحمد عبد الله محمد
41019 – أحمد عبد الناصر أحمد السيد	41012 – أحمد رجب صابر عبد الحميد
41037 – إسلام أحمد حنفي أحمد	41021 – أحمد عصام عبد المتعال محمود
41091 – عبد الرحمن محمد أحمد عبد الله	41057 – حسام عبده علي محمد المصري

Assistant:

Eng. Khalid Rayhan

Supervised By:

Dr. Usama Tharwat Elhagari

Project No.: N42226

Academic Year: 2021/2022

Acknowledgement

We have taken efforts in this project. However, it would not have been possible without the great “Culture and Science City” where our beloved “The High Institute of Computer Science and Information Systems” is hosted, and the support and help of its professors. On behalf of my colleagues, we would like to extend our sincere thanks to all of them.

First, we would like to thank Prof. Dr. Osama Sarwat for his patience, guide, supervision, providing necessary information regarding the project, and support to complete this project. We could not have imagined having a better advisor and mentor for our graduation project.

Also, we would like to express our sincere gratitude to our advisor, Eng. Khalid Rayhan for the continuous support of our graduation project and study, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped us during this year to complete this project.

Finally, we would like to expand our deepest gratitude to our family and friends who have supported us all the time, even though they have not the slightest idea of what we have been through to complete this project.

Abstract

This project introduces a smart and effective way of finding empty spaces and managing the number of vehicles moving in and out in multiple parking spaces by detecting a vehicle by ultrasonic sensors and thus providing feedback. The main purpose of this automated system is to find the vacancy in parking spaces available and navigate the user/driver to reach the desired space using easy-to-use Android application and based on a real-time database, thus reducing search time. Basically, this system is required for malls, multi-story parking structures, companies, and parking facilities. Another important purpose is ensuring that the requirement of labor is very low. Because of the time limitations, the scope of the project is development a proof-of-concept prototype.

Keywords: Automated, smart car parking system, ultrasonic sensor.

Table of Contents

Chapter 1: Introduction	9
1.1 Preamble	10
1.2 Overview	10
1.3 Background of problem	11
1.4 Statement of problem	11
1.5 Objectives	11
1.6 Scope	11
1.7 Significance of the Project	12
1.8 Organization of the project report	12
1.9 Summary	12
Chapter 2: Theoretical Background and Tools	13
2.1 Introduction	14
2.2 Project tools (hardware)	14
2.2.1 Node MCU ESP8266	14
2.2.2 Ultrasonic sensor	15
2.2.3 Servo motor	15
2.2.4 Breadboard	15
2.2.5 LED	16
2.2.6 Resistor	16
2.2.7 Male to female jumper wire	17
2.2.8 Male to male jumper wire	17
2.3 Summary	17
Chapter 3: Analysis and Design	18
3.1 Introduction	19
3.2 The requirements	19
3.3 Block diagrams	21
3.3.1 Block diagram of slots	21
3.3.2 Block diagram of Servo motors	22
3.3.3 Block diagram of software	24
3.4 Flowchart of slots	27
3.5 Use case	28
3.6 Activity Diagram	29

3.7 Context Diagram.....	30
3.8 DFD Diagram.....	30
3.8.1 Data Flow Diagram (Level 0)	31
3.9 ERD Diagram.....	32
3.10 System implementation.....	34
3.10.1 Hardware implementation.....	34
3.10.2 Project tools (software)	36
The user interface.....	38
3.11 Summary	49
Chapter 4: Conclusion and Future Work	50
Conclusion	51
Future work.....	51
References.....	52
Appendix A: Hardware codes.....	53
1- Gate of Garage 1& Garage 2.....	53
2- Slot 1 of Garage 1&2:	54
3- Slot 2 of Garage 1&2:	57
Appendix B: Software codes	59

Table of Figures

Figure 2.1 (Node MCU ESP8266).....	14
Figure 2.2 (Ultrasonic sensor).....	15
Figure 2.3 (Servo motor).....	15
Figure 2.4 (Breadboard).....	15
Figure 2.5 (LED).....	16
Figure 2.6 (Resistor)	16
Figure 2.7 (Male to female jumper wire).....	17
Figure 2.8 (Male to male jumper wire).....	17
Figure 3.1 (Block diagram of slots)	21
Figure 3.2 (Block diagram of servo motors).....	22
Figure 3.3 (Block diagram of software).....	24
Figure 3.4 (Flowchart of slots).....	27
Figure 3.5 (Use case of the system)	28
Figure 3.6 (Activity diagram)	29
Figure 3.7 (Context Diagram).....	30
Figure 3.8 (Data Flow Diagram - User).....	31
Figure 3.9 (Data Flow Diagram - Admin)	32
Figure 3.10 (ERD Diagram)	33
Figure 3.11 (Gate of the garage 1)	34
Figure 3.12 (Slot 1 of garage 1)	34
Figure 3.13 (Slot 2 of garage 1)	35
Figure 3.14 (Android application)	36
Figure 3.15 (Flutter logo).....	36
Figure 3.16 (Firebase logo).....	36
Figure 3.17 (Data of Garages)	37
Figure 3.18 (Data of Location)	37
Figure 3.19 (Data of User)	37
Figure 3.20 (Create an account page)	38
Figure 3.21 (Create an account page fields)	38
Figure 3.22 (Phone number verification page)	39
Figure 3.23 (6 digits to verify the phone number).....	39
Figure 3.24 (Login page)	40
Figure 3.25 (About page).....	40
Figure 3.26 (Map page).....	41
Figure 3.27 (Z Park details)	41
Figure 3.28 (October Park details	42
Figure 3.29 (Z Park slots)	43
Figure 3.30 (Choosing a slot).....	43
Figure 3.31 (Your cards page)	44
Figure 3.32 (Add payment info page).....	44
Figure 3.33 ((Setting a new cost for one of the garages).....	45
Figure 3.34 (Admin page).....	45
Figure 3.35 (Add new garage page).....	46
Figure 3.36 (Add new garage details).....	46

Figure 3.37 (Policy/terms and conditions page)	47
Figure 3.38 (Admin login page).....	47
Figure 3.39 (The empty garage).....	48
Figure 3.40 (The full garage)	48
Figure 3.41 (closed gate).....	49
Figure 3.42 (opened gate)	49

Chapter 1: Introduction

1.1 Preamble

The contribution of this project is to develop an application that helps users to park their cars easily, save time, and fuel.

1.2 Overview

In the present scenario around us, we see excess vehicles and the ineffectiveness to manage them in the correct order. As the population increases day by day, the rate of utilization also increases and coping up with the numbers becomes a task.

An omnipresent problem in Egypt and around the world is finding a parking space to park your vehicle. This task looks simple on side and interior lanes, but the actual problem arises when parking in malls, multistory parking structures, IT hubs, and parking facilities where several hundred cars are being parked and it becomes arduous difficult to find a spot. The general approach to finding a parking space is to go around and drive aimlessly until a free space is being found. Finding a parking space could be the easiest task or could be the most tedious one when it involves a wide area of distributed space across one or multiple levels. The time and fuel that are being consumed is unnecessary because the destination is unknown. The easiest way of approach is to provide a destination specific driving within the parking structure.

A smart car parking system gives a visual output showing an available parking space rather than driving aimlessly. The driver uses the application to choose the nearest parking space around them, and the status of the parking slots in it, whether they are full, booked, or empty. Besides every parking slot, there is an LED to show whether the parking slot is full or empty. So, if it's empty, it will be turned off. And if it's full, it will be turned on. These LEDs are controlled automatically with ultrasonic sensors and the feedback is provided through the status of the LEDs when a vehicle is detected. This system not only makes the accessibility easy but also manages the congestion of vehicles avoiding long search and wait times.

1.3 Background of problem

Many people who own cars face an annoying problem which is, “searching for parking place”, but our system comes to help those people and make their parking searching process quicker, easier, and more secure.

Sometimes we want to visit a place, but we can't find a place to park because there is no proper place to park our cars, or there is a place, but it is being occupied. Our system helps you in these kinds of situations to find the closest place to park your car and tells you if it is available or not.

1.4 Statement of problem

To design a smart parking system that helps users to find the nearest smart vehicle parking.

1.5 Objectives

- 1- To identify the components of the smart parking system.
- 2-To design the smart parking system.
- 3-To develop the smart parking system.
- 4-To test and evaluate the smart parking system.

1.6 Scope

We are going to implement this system in 6th of October City using two different garages, Z Park and October Park. Each of those two garages has two parking slots, which the users will be able to book any garage of them at any time.

The user can choose which slot at which garage he want to park using the application with one click.

In the future, this system can be extended to include the users around Egypt.

1.7 Significance of the Project

By using the smart parking system, the users can park their cars any time they want to.

- 1- Make the parking process easier for the user.
- 2- Lower the time of searching for a parking place.
- 3- Lower the costs of parking.
- 4- Protect the user's car from being stolen or getting damaged.

1.8 Organization of the project report

Chapter 1 presents a brief introduction about the system, lists the requirements for the system to work well, shows the problem statement, shows the objectives of the system, and explains how the system works.

Chapter 2 lists the hardware tools of the system, lists the software tools of the system, shows the interface of the application, and shows the scope of the system.

Chapter 3 shows the block diagrams, flow charts of the system, the interface, and the connection of the hardware component.

Chapter 4 discusses how the software and hardware are implemented in the system.

The conclusion and future work.

1.9 Summary

In this chapter, we took an overview of the system, background of the problem we need to fix, objectives of the system, and an explanation of how to make a solution for these problems using our system.

Chapter 2: Theoretical Background and Tools

2.1 Introduction

The chapter briefly discuss the components that were used to build the system.

2.2 Project tools (hardware)

The hardware components that are used in our system are.

2.2.1 Node MCU ESP8266

Node MCU is an open-source development board and firmware based in the widely used ESP8266 -12E Wi-Fi module. It allows you to program the ESP8266 Wi-Fi module with the simple and powerful LUA programming language or Arduino IDE. With just a few lines of code you can establish a Wi-Fi connection and define input/output pins according to your needs exactly like Arduino, turning your ESP8266 into a web server and a lot more. It is the Wi-Fi equivalent of ethernet module. The Node MCU ESP8266 is used to connect the components, such as ultrasonic sensor and servo motor, with the firebase to send the status of every ultrasonic sensor and servo motor.

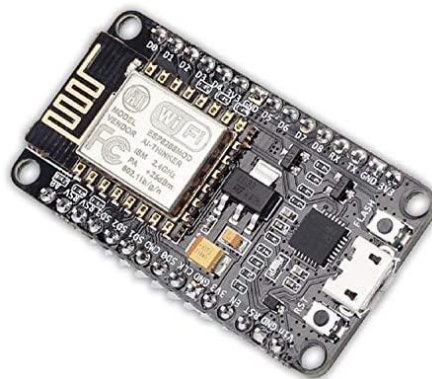


Figure 2.1 (Node MCU ESP8266)

2.2.2 Ultrasonic sensor

An ultrasonic sensor is an electronic device that measures the distance of a target object by emitting ultrasonic sound waves and converts the reflected sound into an electrical signal. It detects the cars and reports the feedback to the firebase with the help of Node MCU ESP8266.



Figure 2.2 (Ultrasonic sensor)

2.2.3 Servo motor

A servo motor is a type of motor that can rotate with great precision. Normally, this type of motor comprises a control circuit that provides feedback on the current position of the motor shaft. This feedback allows the servo motors to rotate with great precision. It opens and closes the gate of the parking space.



Figure 2.3 (Servo motor)

2.2.4 Breadboard

A breadboard is a device for temporary prototyping with electronics and test circuit designs. Most electronic components in electronic circuits can be interconnected by inserting their leads or terminals into the holes and then making connections through wires where appropriate. It interconnects all the components together.



Figure 2.4 (Breadboard)

2.2.5 LED

A light-emitting diode (LED) is a semiconductor light source that emits light when current flows through it. It shows whether the parking slot is empty or full.



Figure 2.5 (LED)

2.2.6 Resistor

A resistor is a passive two-terminal electrical component that implements electrical resistance as a circuit element. In electronic circuits, resistors are used to reduce current flow.



Figure 2.6 (Resistor)

2.2.7 Male to female jumper wire

It is used to interconnect the Node MCU ESP8266 with the breadboard.



Figure 2.7 (Male to female jumper wire)

2.2.8 Male to male jumper wire

It is used to interconnect the components, such as ultrasonic sensor, servo motor, etc., with the breadboard.



Figure 2.8 (Male to male jumper wire)

2.3 Summary

In this chapter, we talked about each component that is being used in our system in details.

Chapter 3: Analysis and Design

3.1 Introduction

In this chapter, we will discuss the requirements for the system to make it usable and suitable for users.

Our system works with the help of many things: Ultrasonic sensors to tell us if the place is being occupied or not, pre-paid cards which we are going to use to pay for the parking's fees, a database to record all parking slots that exist, a 2-step verification system that makes sure that the person who booked the parking slot is the one who is going to use it, and an Android application to guide the user through the whole process: finding the closest place to park their car and knowing if it is available or not.

The diagrams

- 1. Block diagram (slots – gate – software).**
- 2. Flowchart.**
- 3. Use case.**
- 4. DFD.**
- 5. ERD.**
- 6. Activity diagram.**
- 7. Context diagram.**

3.2 The requirements

1. The location of the parking space

There should be spaces to park the car in them. Also, there should be multiple parking spaces in different places to cover a big user base.

2. The slots that the users will park in them

Each parking space should have multiple parking slots to ensure the user will park in them easily.

3. The components that will be used for the system

There are a lot of hardware components that will be used in the system.

- 1- Ultrasonic sensor**
- 2- Node MCU ESP8266**
- 3- Servo motor**
- 4- Breadboard**
- 5- LED**
- 6- Resistor**
- 7- Male to female jumper wire**
- 8- Male to male jumper wire**

4- The usability of the application

The application should be bug free and working well anytime the user interacts with it.

5- The easiness of the application

The application should be an easy-to-use application, so the user could use it with no problems.

6- The quickness of parking the car

The user should be able to park the car fast without any problems.

7- The security of the parking place

The parking space that the user will park their car in it should be secure to save the car from being stolen or getting damaged.

3.3 Block diagrams

3.3.1 Block diagram of slots

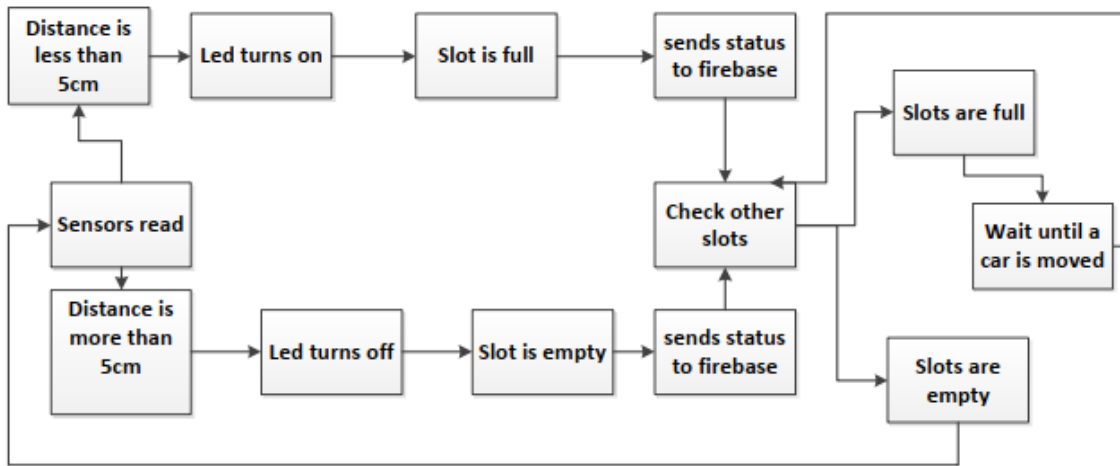


Figure 3.1 (Block diagram of slots)

1- Sensors read

The sensors will read if there is a car or not, and there will be two statuses for read.

Distance is less than 5 cm

The LED will turn on to show there is a car here, and this slot is full.

The status of the slot will be sent to the firebase, which is full.

Distance is more than 5 cm

The LED will turn off to show there is not a car here, and this slot is empty.

The status of the slot will be sent to the firebase, which is empty.

2- Checks other slots

The sensors will read again in other slots to make sure there is an empty place or not.

Full slots

There are no empty slots in this parking space, and no one can park right now. The next user should wait until a car gets moved from the parking space to use the empty slot.

Empty slots

There are empty slots that could be used by anyone right now. And the sensors will check again if the car is near to the slot or not.

3.3.2 Block diagram of Servo motors

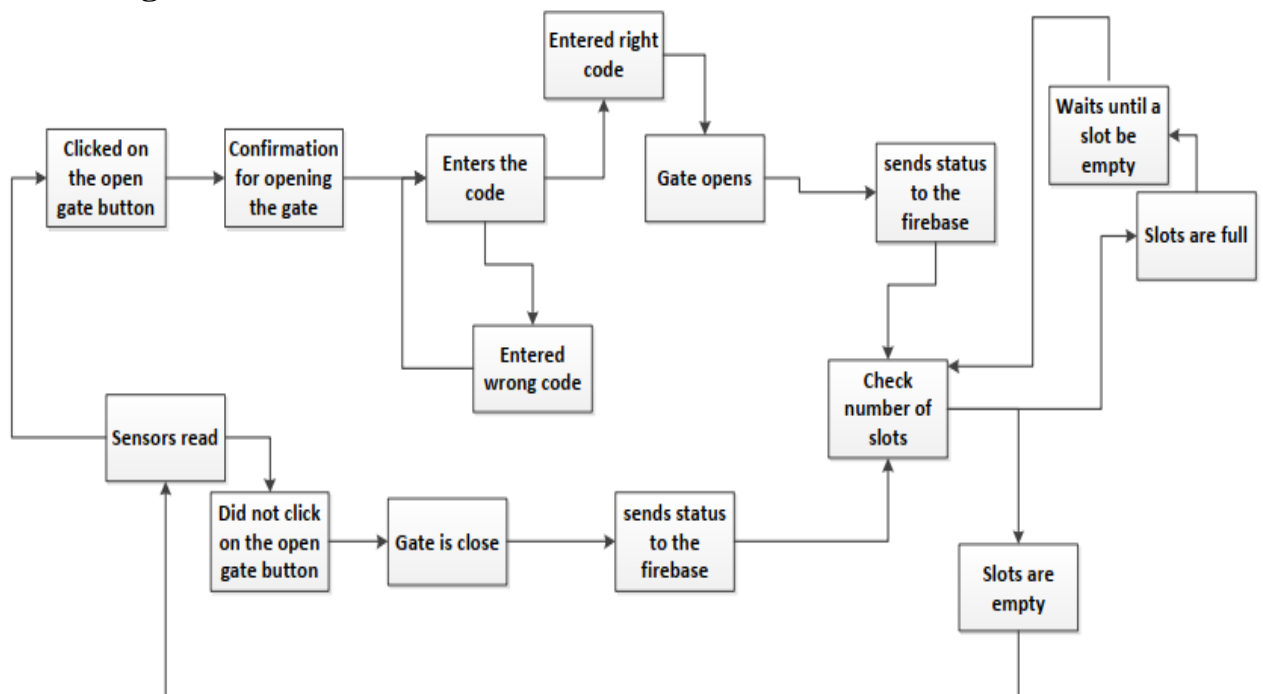


Figure 3.2 (Block diagram of servo motors)

1- Sensors read

The sensors will read if there is a car or not.

2- Clicked on the “open gate” button

The user will be able to open the gate when he arrives.

3- Did not click on the “open gate” button

The user will not be able to open the gate because they didn't click on the “open gate” button yet.

4- Confirmation for opening the gate

After clicking the “open gate” button, the user will see a message asking them to enter a confirmation code to make sure they really want to open the gate.

5- Checks number of slots

Check if there is a slot to park more cars in the parking space or not. If yes, the sensors will read again. If no, the user will have to wait until a car gets moved from the parking space to use the empty slot.

3.3.3 Block diagram of software

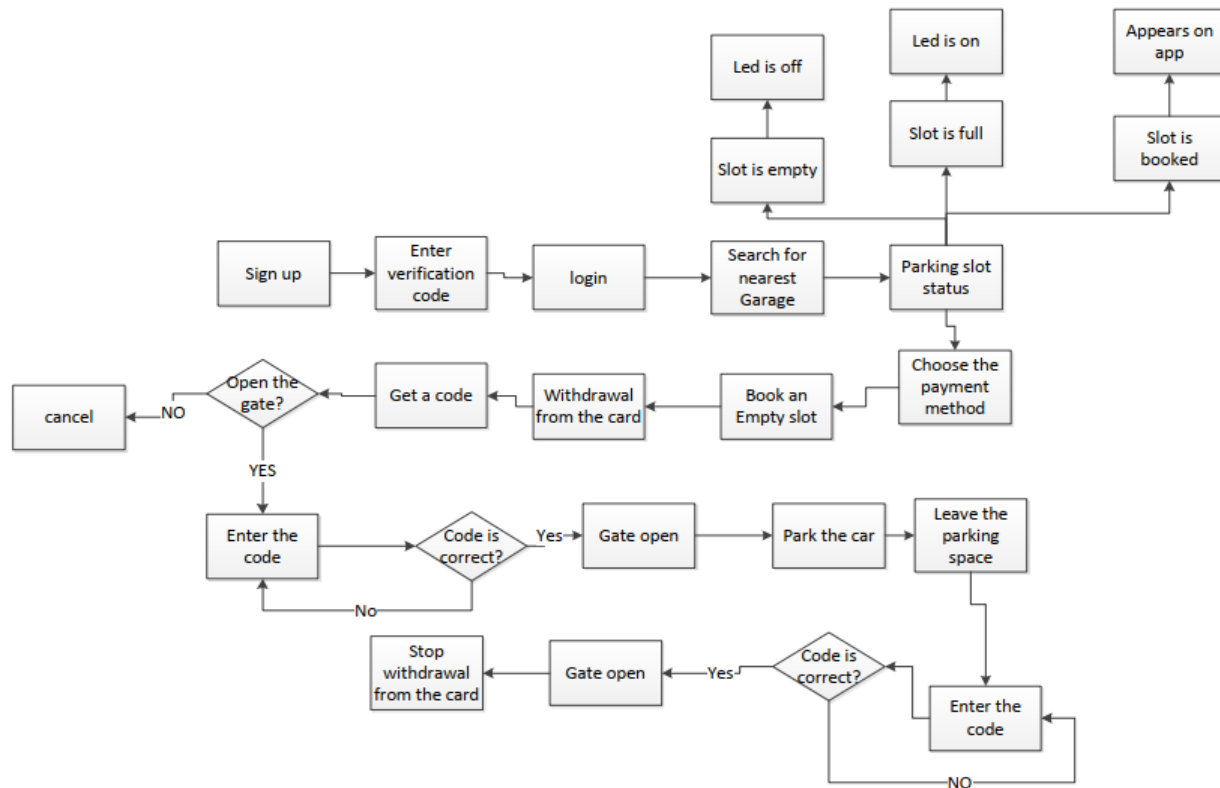


Figure 3.3 (Block diagram of software)

1- Sign up

At the first time, the user should enter their name, their personal phone number, their username, their email, and their password.

2- Enter verification code

After signing up, the user should enter the verification code to confirm the phone number.

3- Login

After the signing up process is complete, the user should login in with the username/email and password.

4- Search for the nearest parking place

The app will show to the user the nearest parking place and the parking fees.

5- Parking slots status

The sensors will check if the slots are empty, booked, or full.

Empty: The LED will be turned off.

Booked: The car will be half transparent in the application.

Full: The LED will be turned on.

6- Choose the payment method

The user will insert the payment method and choose it. It could be MEZA (Pre-paid card).

7- Book an empty slot

After the user choose their payment method and confirmed it, the user will have to book an empty slot.

7- Withdraw from the card

After the user choose their payment method and confirmed it, the application will withdraw from card per hour.

8- Get the code

After the withdrawal process starts, the user will get a code when they arrive at the gate.

9- Open Gate

The application will ask the user whether or not they want to open the gate.

10- Enter the code

The user will enter the confirmation code.

Code is correct the gate will open, then
the user will park their car in the parking slot.

Code is wrong the gate will not open,
then the user will have to enter the code again.

11- Leave the parking space

The user will have to enter another confirmation code, then leave the parking space.

12- Stop withdraw from card

After leaving the parking space, the withdrawal process will stop.

3.4 Flowchart of slots

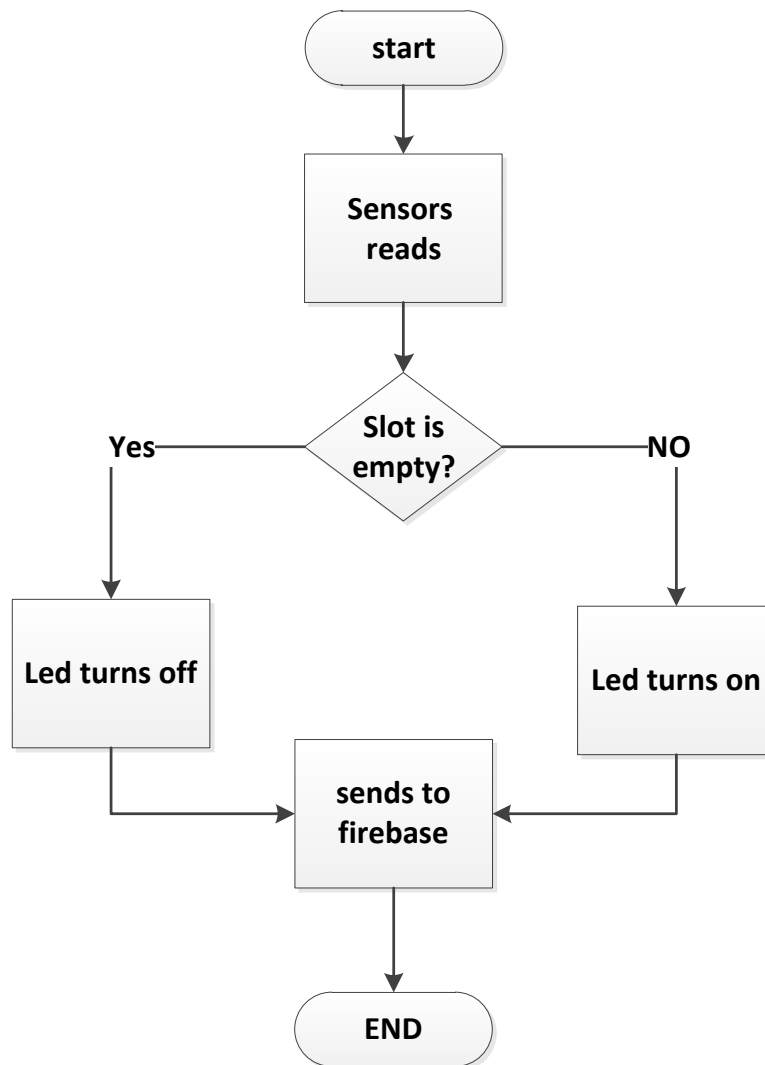


Figure 3.4 (Flowchart of slots)

3.5 Use case

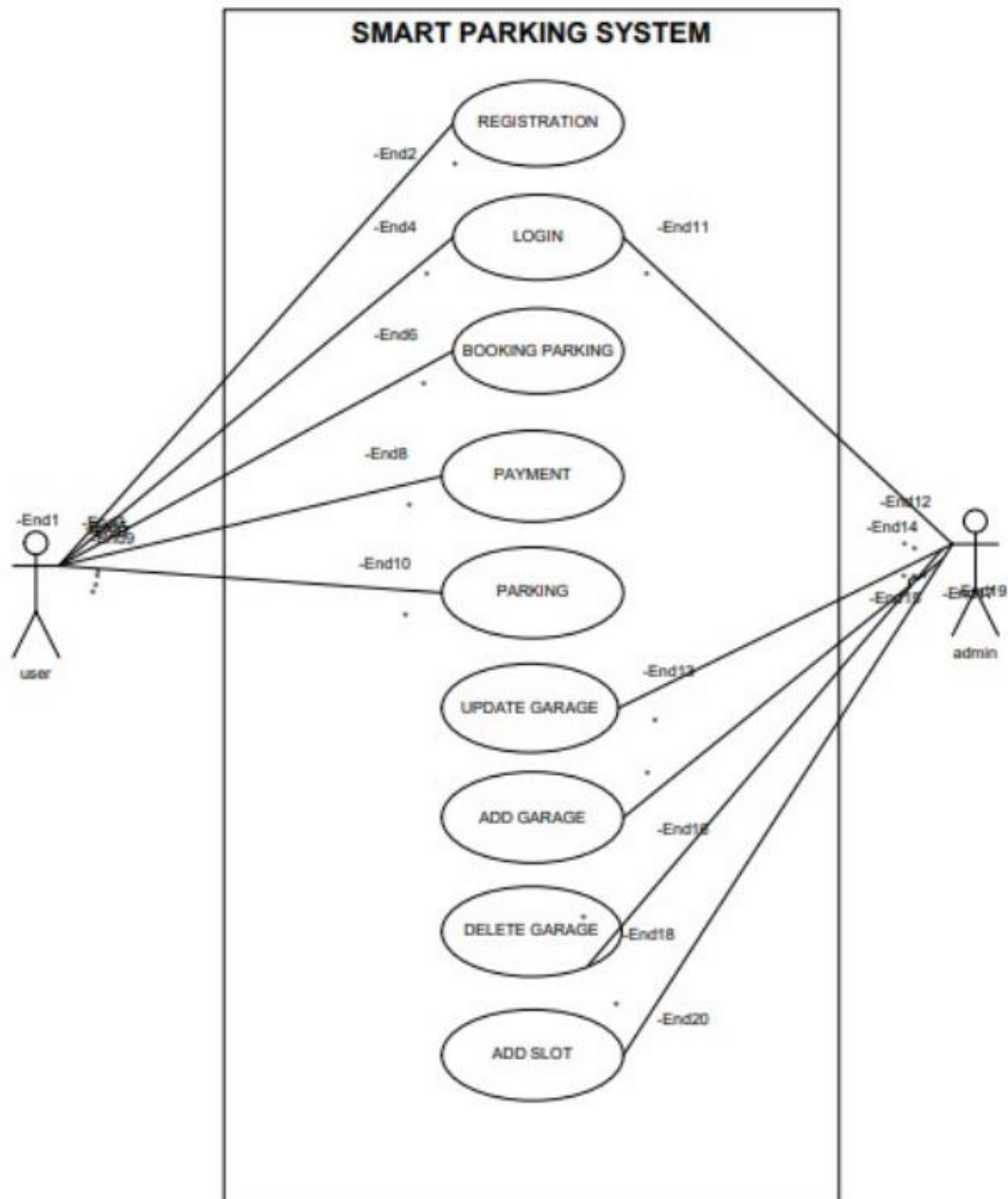


Figure 3.5 (Use case of the system)

3.6 Activity Diagram

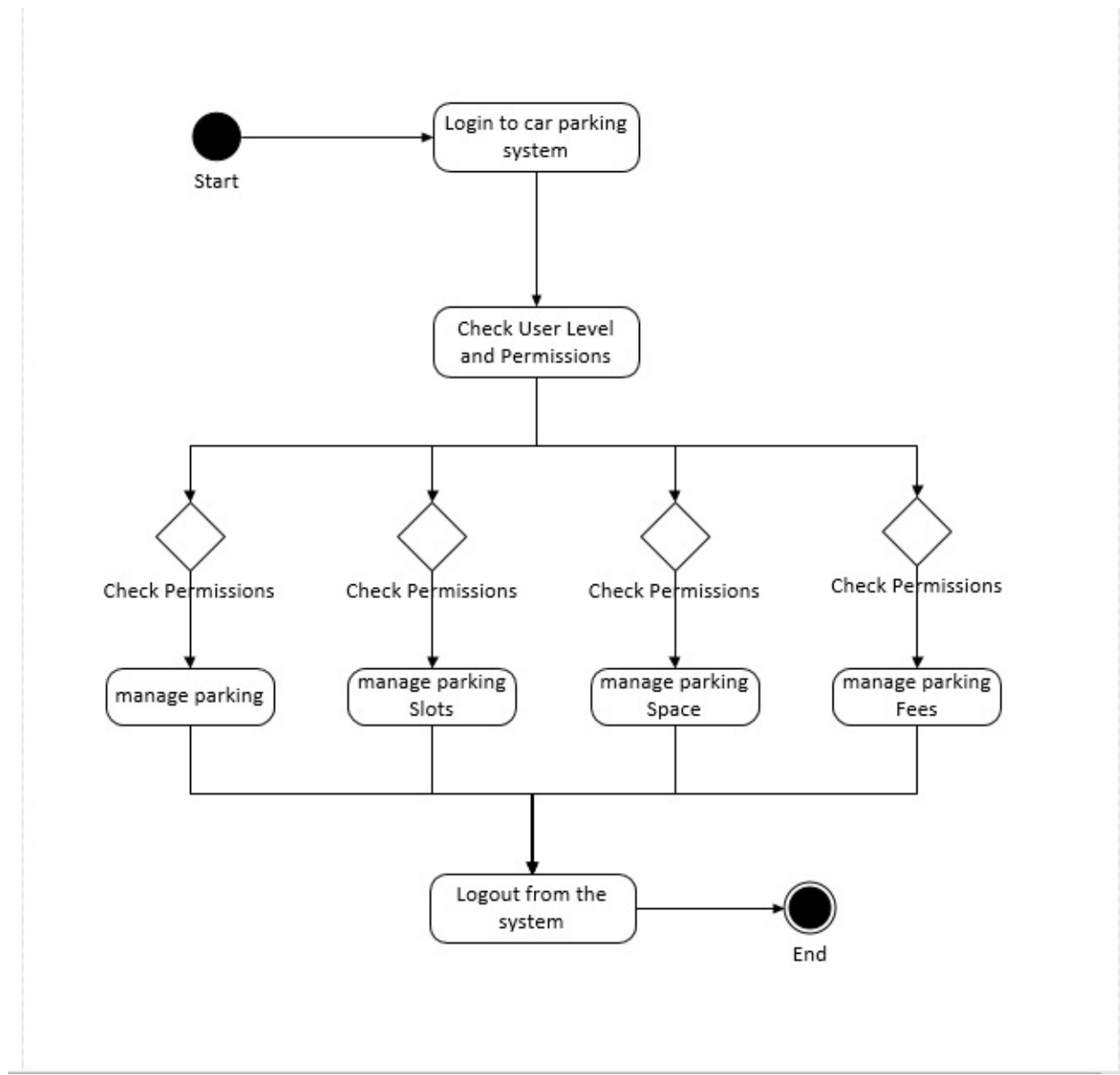


Figure 3.6 (Activity diagram)

3.7 Context Diagram

A context diagram is a data flow diagram, with only one massive central process that subsumes everything inside the scope of the system. It shows how the system will receive and send data flows to the external entities involved. Such as system, organizational groups, external data stores. Since a context diagram is a specialized version of the Data Flow Diagram, understanding a bit Data Flow Diagram can be helpful.

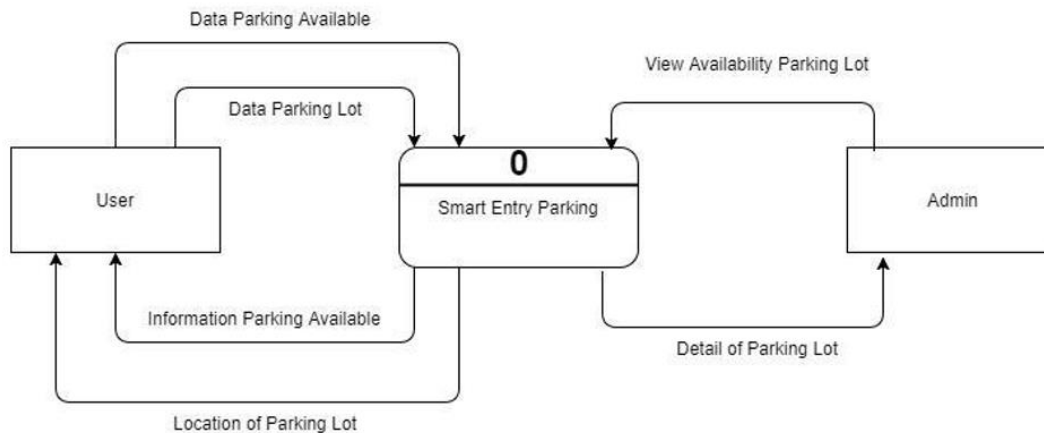


Figure 3.7 (Context Diagram)

3.8 DFD Diagram

Data Flow Diagram (DFD) provide a visual representation of the flow of information within a system. By drawing a data Flow Diagram, you can tell the information provided by and delivered to someone who takes part in system process, the information needed in order to complete the processes, the information needed in order to complete the processes and the information needed to be stored and accessed.

3.8.1 Data Flow Diagram (Level 0)

Figure show the flow in this system starting with user inserting information parking, choose parking, and make confirmation of parking lot. Admin can log in into the system and manage parking lot.

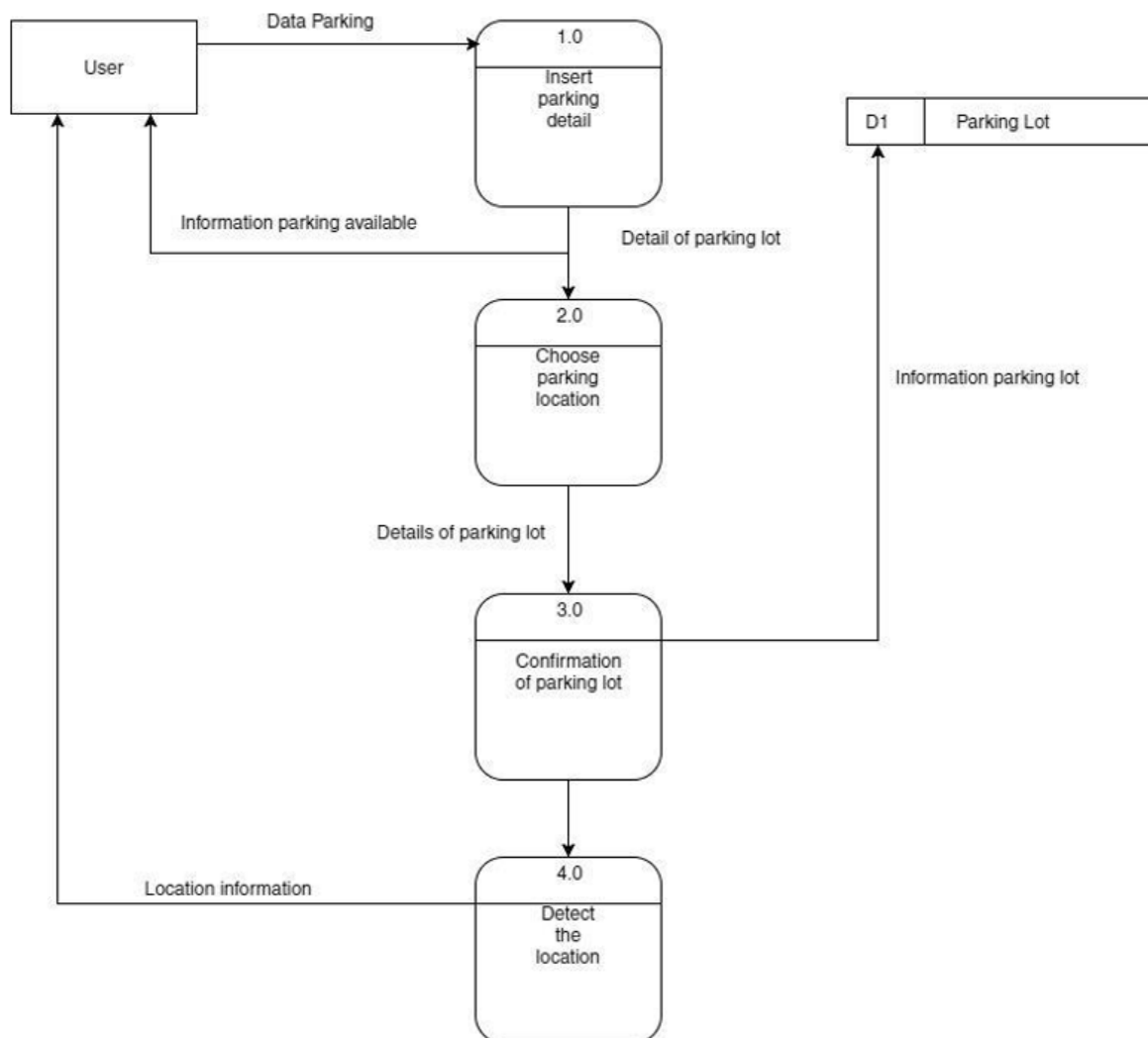


Figure 3.8 (Data Flow Diagram - User)

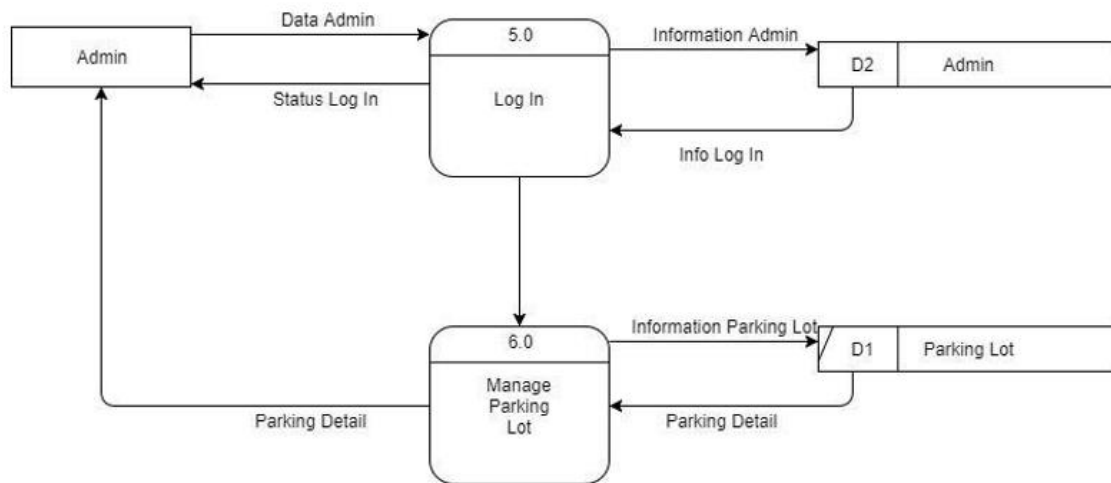


Figure 3.9 (Data Flow Diagram - Admin)

3.9 ERD Diagram

Entity Relationship Diagram in software engineering is an abstract to describe a database. It is typically used in computing in regard to the organization of data within database or information systems. An entity is a piece of data-an object or concept about which data is stored. A relationship is how the data is shared between entities. Figure 39 show the ERD if Smart Entry parking System. That show this system uses the derive table. Derive table is used to make a relationship between two or more main tables. It contains the primary key, foreign key from other tables.

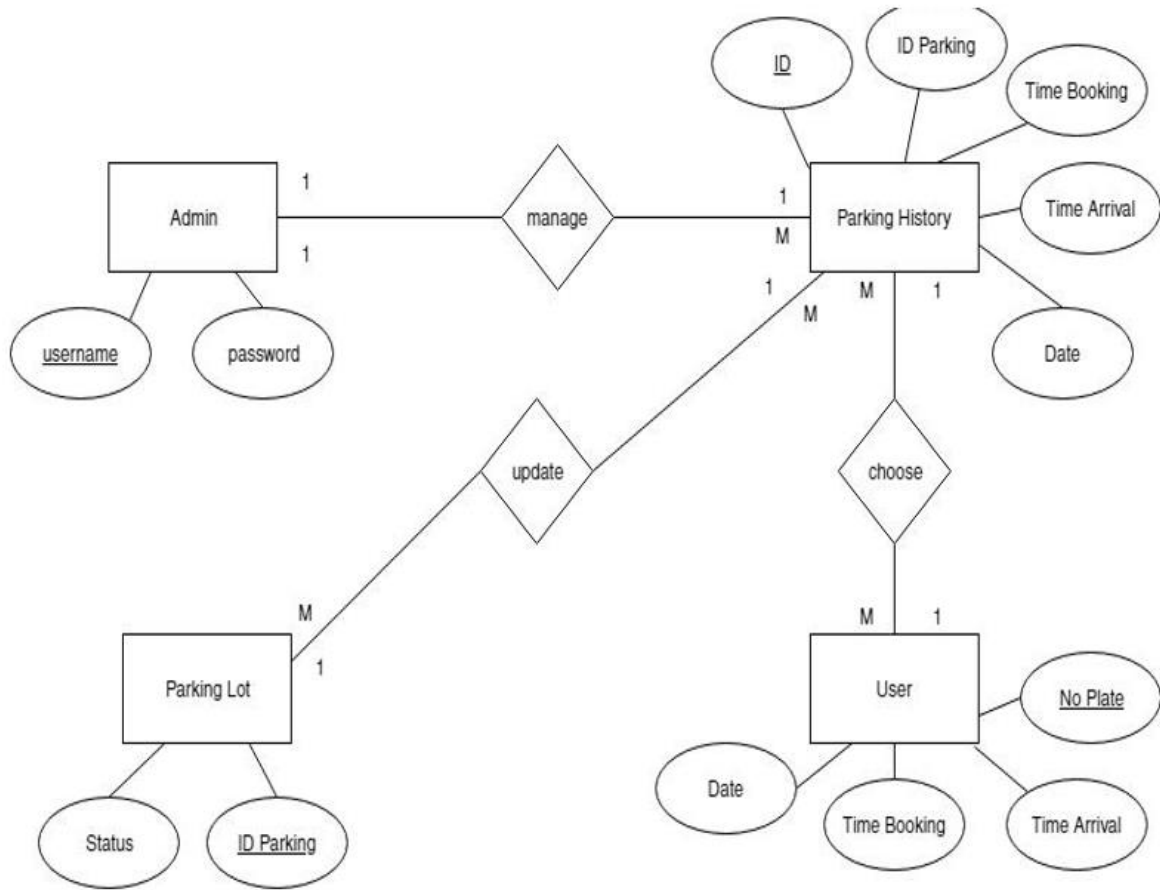


Figure 3.10 (ERD Diagram)

3.10 System implementation

3.10.1 Hardware implementation

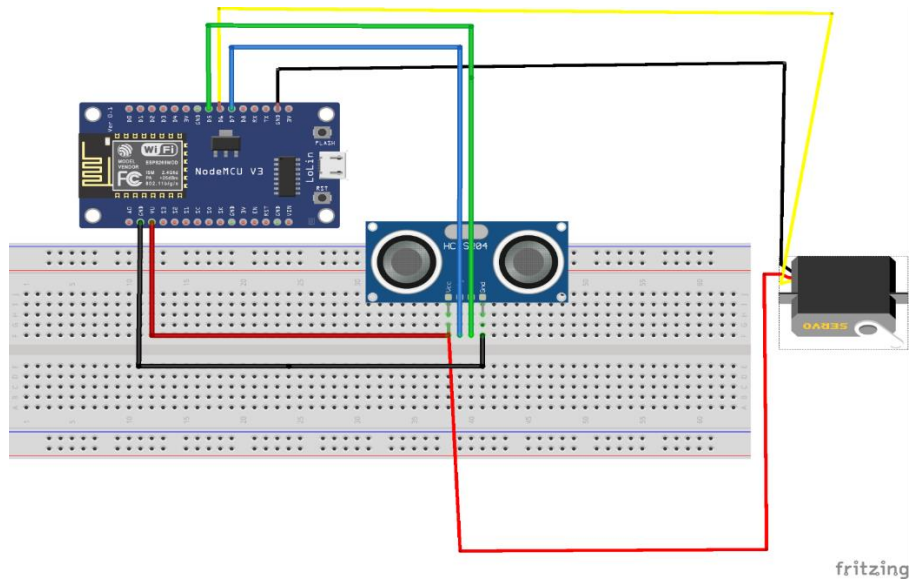


Figure 3.11 (Gate of the garage 1)

Note: Gate of garage 2 is the same.

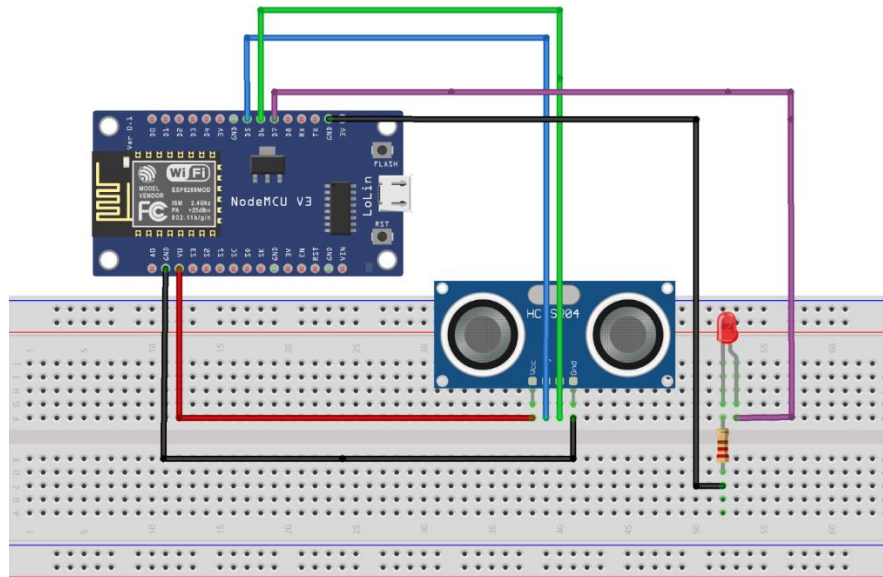


Figure 3.12 (Slot 1 of garage 1)

Note: Slot 1 of garage 2 is the same.

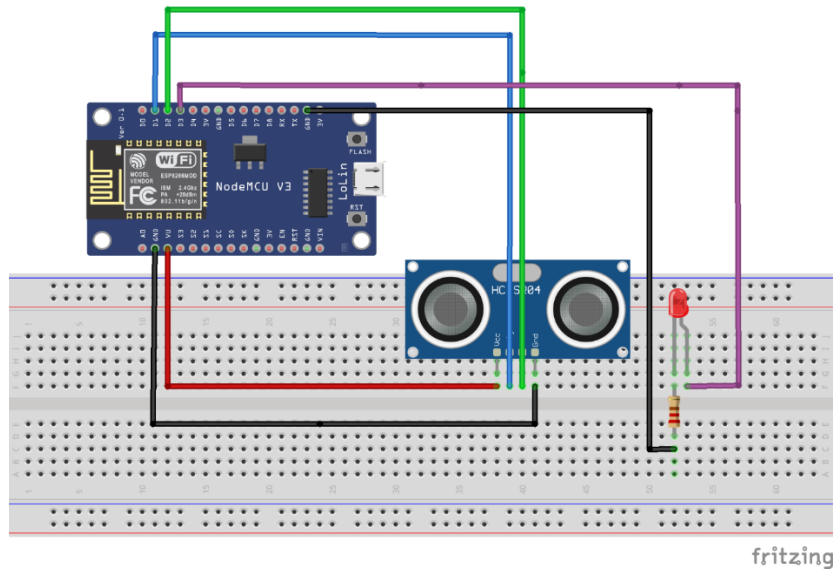


Figure 3.13 (Slot 2 of garage 1)

Note: Slot 2 of garage 2 is the same.

3.10.2 Project tools (software)

The application

The application is developed for users who want to park their car easily without searching a lot for an empty space. The application is developed using Flutter.



Figure 3.14 (Android application)



Figure 3.15 (Flutter logo)

The database

The application will be interconnected with a firebase database to store and access the user data.



Figure 3.16 (Firebase logo)

Data of Garages



Figure 3.17 (Data of Garages)

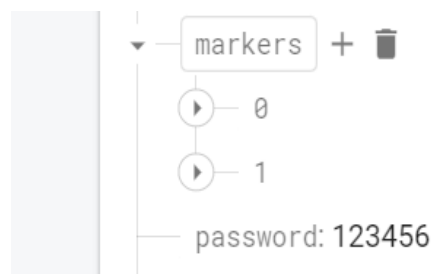


Figure 3.18 (Data of Location)

Data of User



Figure 3.19 (Data of User)

3.10.3 Software implementation

The user interface

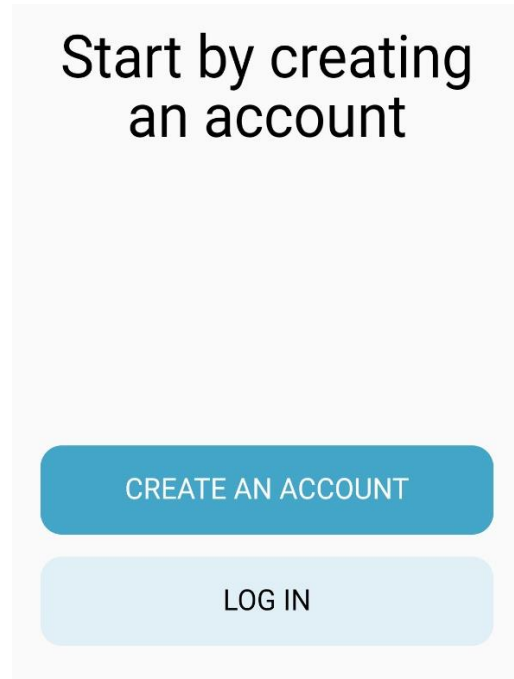


Figure 3.20 (Create an account page)

A form titled "CREATE AN ACCOUNT" in bold, black, uppercase letters. It contains three input fields, each with a label in a small blue pill-shaped box above it. The first field is labeled "Name" and contains the text "ughhgjhghh" with a person icon on the right. The second field is labeled "Phone Number" and contains the text "+201022535966" with a phone icon on the right. The third field is labeled "Password" and contains the text "123456" with three asterisks on the right. Below these fields is a large blue button with the text "CREATE AN ACCOUNT" in white, uppercase letters.

Figure 3.21 (Create an account page fields)

In the first time of using the application, the user should enter their data. And to do so, they have to **create an account**:

1. Enter their name.
2. Enter their phone number so we could be able to verify it by sending a verification code to it.
3. Enter a password for their account.
4. Press on the “create an account” button to proceed with creating an account process.

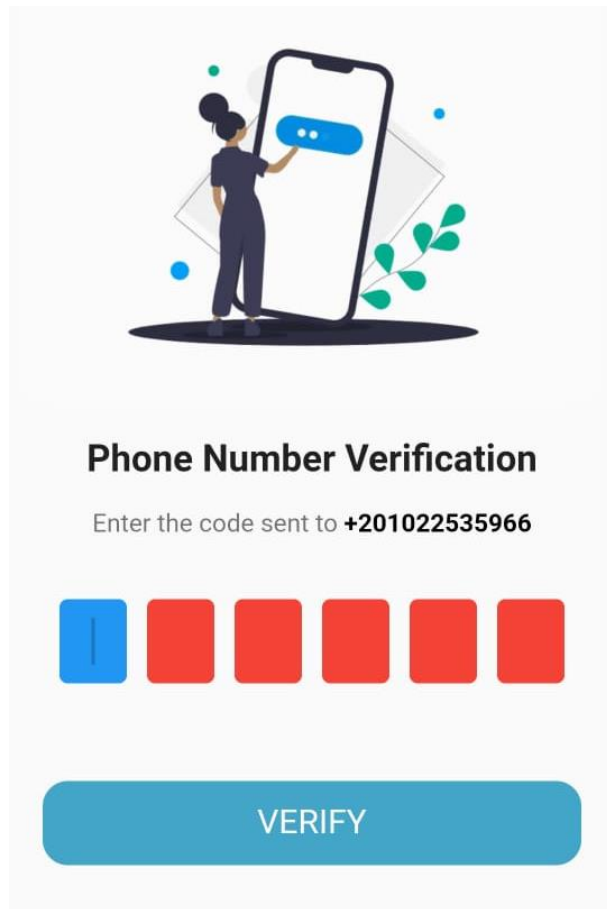


Figure 3.22 (Phone number verification page)

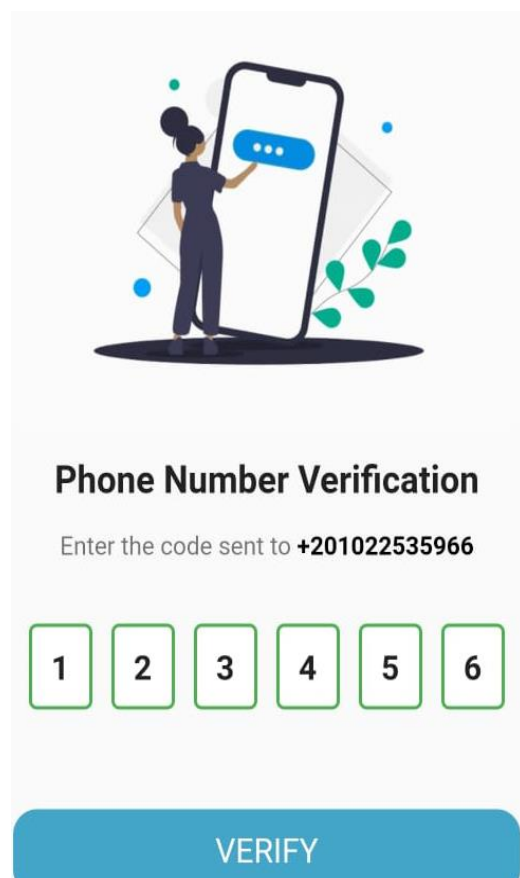


Figure 3.23 (6 digits to verify the phone number)

After creating an account using the application, the user will get a verification code to verify their number. The verification code will be sent in a few seconds.

The verification code will contain six digits that the user has to enter quickly before it gets expired.

If the user entered a wrong number, they will have to try again later.


Smart Parking


LOGIN

0101234567 
***** 

LOGIN

Figure 3.24 (Login page)

 about



mohmed

Phone Number : +201022535966

Log out

Figure 3.25 (About page)

After the user has verified their number using the verification code that was sent to them, they can access and use the application.

They can login to the application using the account credentials that they entered during the sign-up process.

After the user logged in to the application, they will see their username is shown. And of course, they can log out any time.

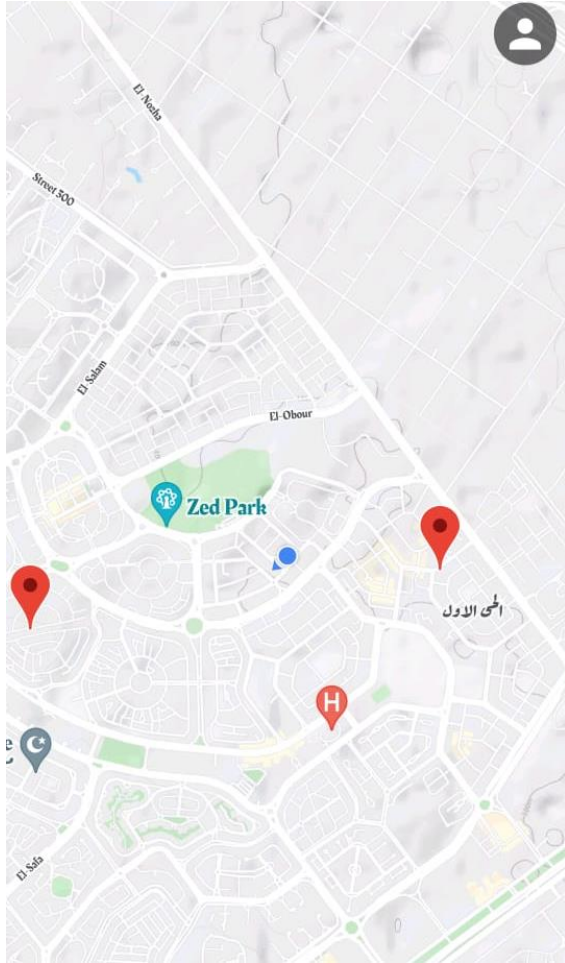


Figure 3.26 (Map page)

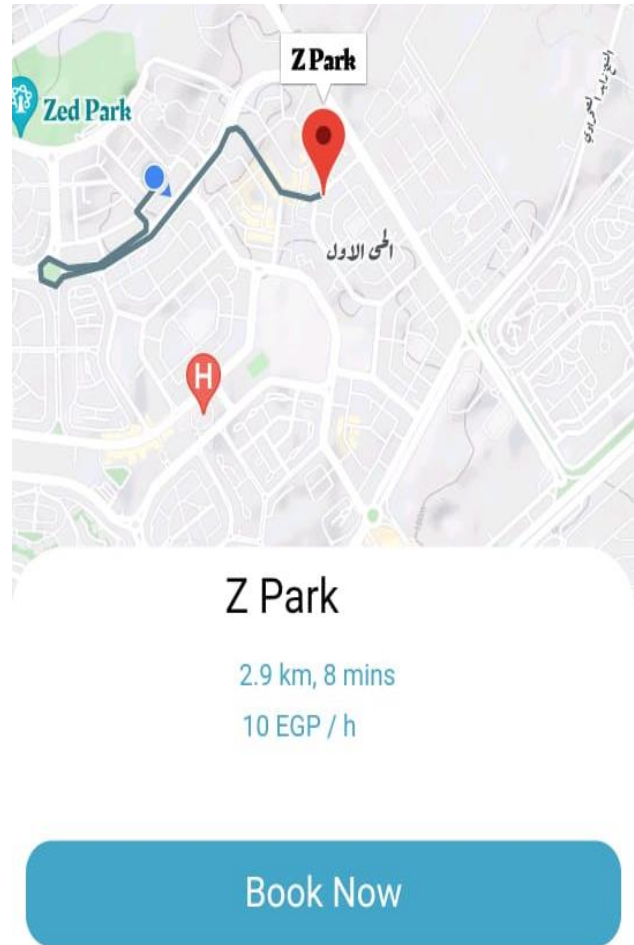


Figure 3.27 (Z Park details)

When the user login to the application the map appeared to user.

The user will see their location and the directions to the garages that they can park in it.

They can choose from two garages, and they can choose the suitable one for them according to multiple things, like prices and the distance, for example.

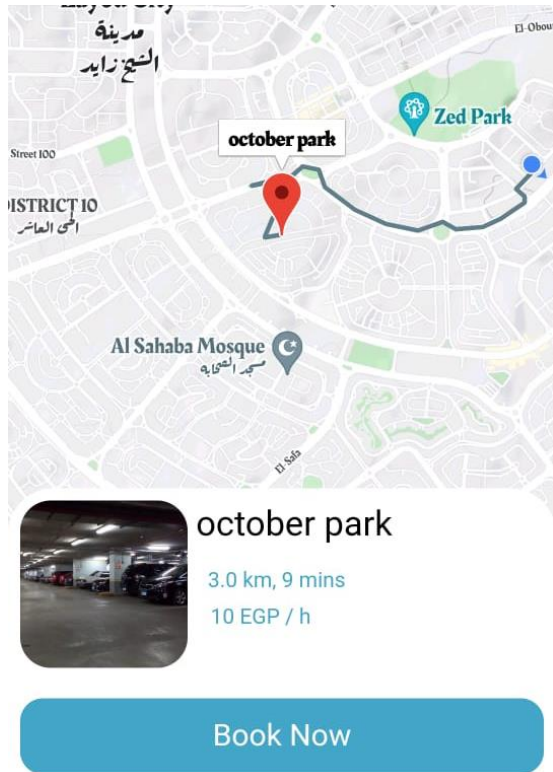


Figure 3.28 (October Park details)

There are two garages

1. Z Park
2. October Park

The user will choose the garage that they want to go, will see the directions for these garages, will see the price for each one per hour, and will see the distance to the garage in kilometers.

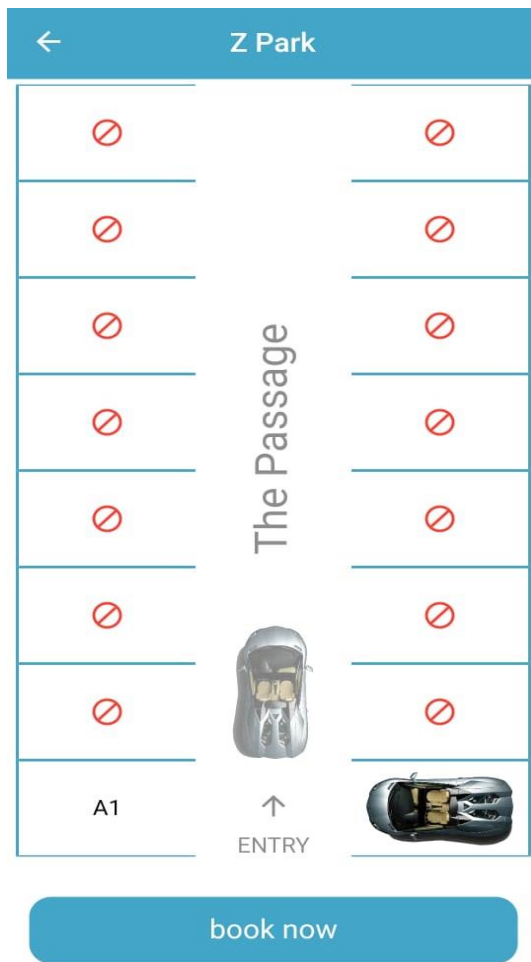


Figure 3.29 (Z Park slots)

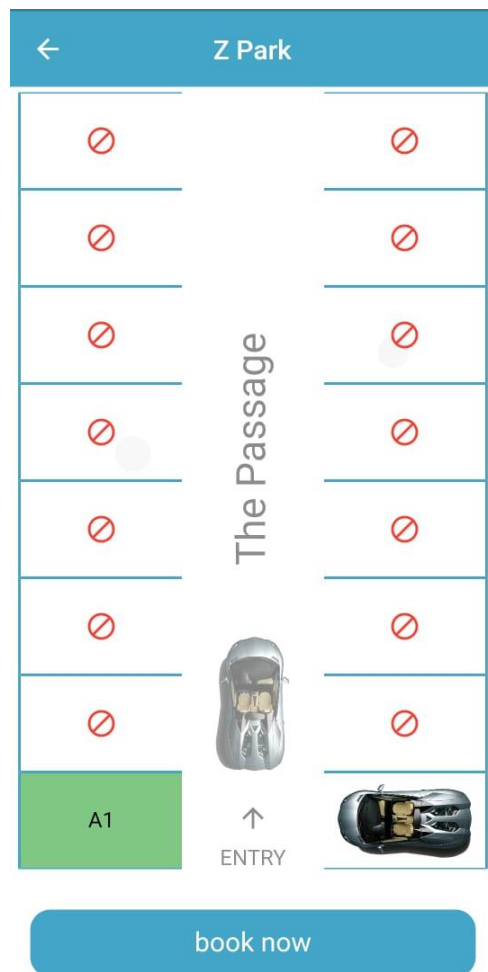


Figure 3.30 (Choosing a slot)

After choosing the garage, the user will see the status of the slots in it, and if they are empty or not. Then, the user will choose which slot they want to book.

After that, the slot will turn green to make sure that the user wants to book.

← Add Payment Info

XXXX XXXX XXXX XXXX
VALID THRU MM/YY
CARD HOLDER

Number

Expired Date CVV

Card Holder

Save Card Details

Figure 3.31 (Add payment info page)

← Your Cards

Add New Card +

5555 **** * 6355
VALID THRU 11/11
Saif Ahmed

Figure 3.32 (Your cards page)

To book the slot which the user needs, the user has to enter their prepaid card (MEZA). The user will enter the information of their card so the application withdraws from the card.

The information are

- Card holder name
- Number of card
- Expired date
- CVV

There is an option to add more cards if the user has more than one card.

When they choose the slot, they will have to choose the card which they want to pay with.

The admin interfaces

The admin interface can add or remove data (garages' data) from the application easily to manage it.

The application should have an admin interface.

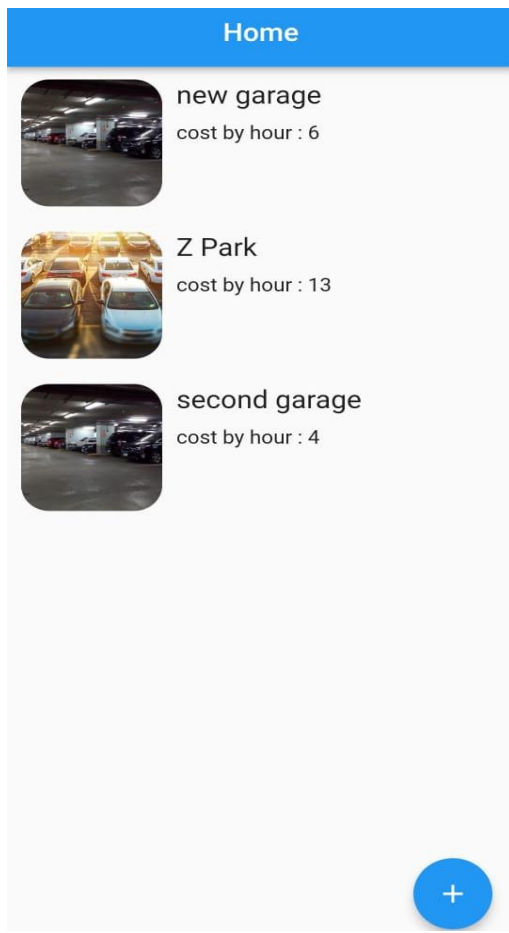


Figure 3.33 (Admin page)

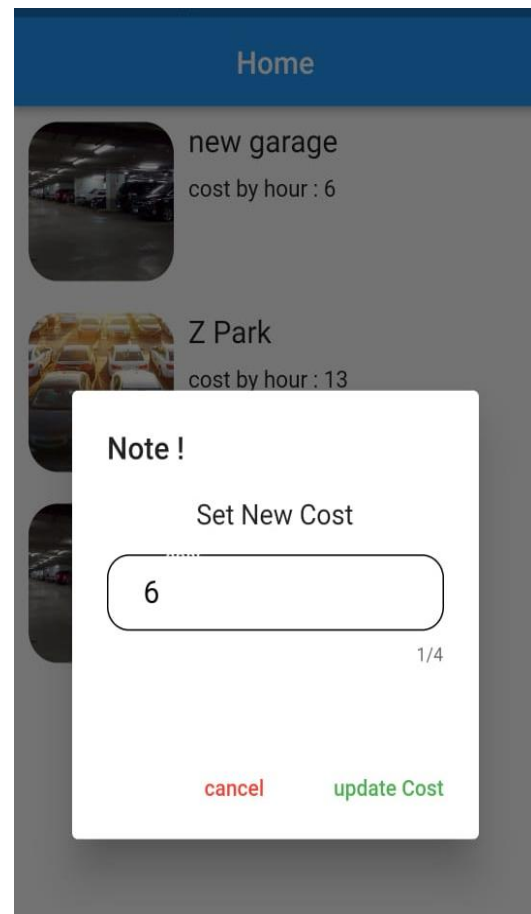


Figure 3.34 ((Setting a new cost for one of the garages)

The admin interface can control the garages. It can add or remove garages, update the cost by hour of the garage, and control the number of slots.

The screenshot shows a mobile application interface for adding a new garage. At the top is a blue header with a back arrow and the text "add new garage". Below the header are three white input fields with rounded corners. The first field is labeled "title", the second "title server", and the third "cost". Below these fields is a grey square button labeled "add image". At the bottom are two blue buttons: "Get Location" and "save".

Figure 3.35 (Add new garage page)

The screenshot shows the details page for adding a new garage. It has a blue header with a back arrow and the text "add new garage". Below the header are two white input fields with rounded corners. The first field contains the text "new garage" and the second field contains the number "10". Below these fields is a gallery of four small images showing different types of plants. Below the gallery are two blue buttons: "Get Location" and "save". At the bottom, the coordinates "longitude : 31.0010782" and "latitude : 30.041212" are displayed.

Figure 3.36 (Add new garage details)

Using the admin interface, the admin can add a garage, a title/name for the garage, a title server (the garage name in the server/database), and a cost for the new garage.

The admin can add an image and location for each garage.

1. We may collect location data from the device you are using to access the App, in order to enable us to provide parking and other services via the App. This may require you to enable or authorise location services via the permission system used by your mobile operating system. If you do not enable or authorise location services, the App may not be able to provide you with all of its features or services.

2. When booking, one hour will be charged

3. When you establish a user account or user profile with us we may collect personal details from you. This may include some or all of the following: your user name, password and an phone number.

4. When you use the App to pay for parking, we may collect transaction details from you. This may include user details. You will also need to enter payment details.

5. The counter will start to count after an hour and calculate the time you stayed at slot

6. If you do not reach the garage gate within 30 min minutes, your booking will be canceled and the cost of one hour will be charged

Okay

Login

Password

0/6

login

Figure 3.37 (Policy/terms and conditions page)

Figure 3.38 (Admin login page)

This first page informs the user about the policies/terms and conditions of the application. The second page is the admin login page, where the admin of the application controls the user application.



Figure 3.39(The empty garage)

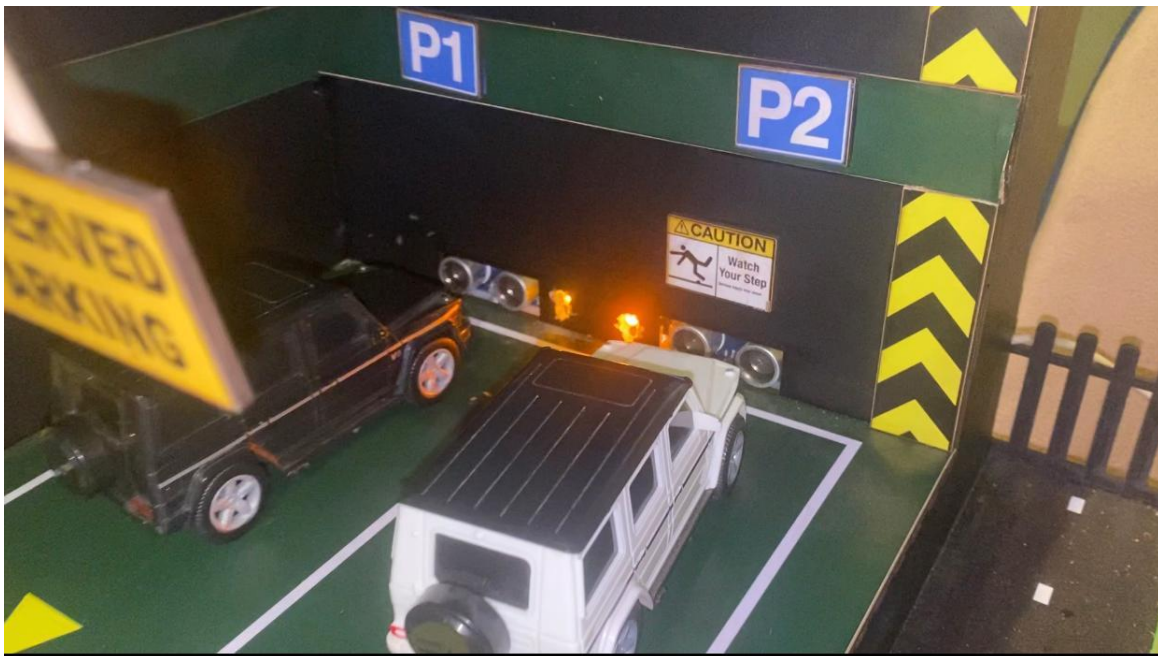


Figure 3.40(The full garage)



Figure 3.41(closed gate)



Figure 3.42(opened gate)

3.11 Summary

In this chapter, we talked about the requirements of the system, the diagrams that have been used to help us build this system, the system implementation (hardware and software), and the scope of the system.

Chapter 4: Conclusion and Future Work

Conclusion

The services provided by smart parking have become the essence of building smart cities. This project focused on implementing an integrated solution for smart parking. The proposed system has several advantages, including detecting nearby parking spaces using the Android application, the distance to them, calculating the time to get there, and calculating the expected cost to park there.

We have achieved all the stated objectives of this project as follows:

- **Objective 1: To identify the components of the smart parking system this objective has been achieved and explained in chapter 2 and 3.**
- **Objective 2: To design the smart parking system
This objective has been achieved in chapter 3**
- **Objective 3: To develop the smart parking system
This objective has been achieved in chapter 4**
- **Objective 4: To test and evaluate the smart parking system
This objective has been achieved in chapter 3**

Future work

- 1- In this project, it faced some limitations and problems that will be dealt with in the future. One of these limitations is the number of slots and garages. In the future, the number of slots and garages will increase.
- 2- The system will be applied to real garages instead of small prototypes in the future.
- 3- In the future, we will encrypt the prepaid card (MEZA) information to protect them from unauthorized access, so that means more security inside the system.

References

- 1- Bagula, Antoine, Lorenzo Castelli, and Marco Zennaro. On The Design of Smart Parking Networks in the Smart Cities: An Optimal Sensor Placement Model. *Open Access Sensors* 15 (2015): 15443-15467. Print.
- 2- Bhende, Manisha, and Sanjeev Wagh. Intelligent Car, Park Management System, Using Wireless Sensor Network. *International Journal of Computer Applications* 122 (2015): 1- 6. Print.
- 3- Fahmy, Hossam M. A. *Wireless Sensor Networks: Concepts, Applications, Experimentation and Analysis.* , 2016. Print.
- 4- Familiar, Miguel S., et al. "Building service-oriented smart infrastructures over wireless ad hoc sensor networks: A middleware perspective." *Computer Networks* 56.4 (2012): 1303-1328.
- 5- Gaurav , Kate, et al. Android Application for S-Park System. *International Journal of Research in Engineering and Technology* 4.10 (2015): 107-110. Print.
- 6- Idris, M.Y.I., Tamil, E.M., Noor, N.M., Razak Z. and Fong, K.W. 2009. Parking Guidance System Utilizing Wireless Sensor Network and Ultrasonic Sensor. *Information Technology Journal*, 8: 138-146.
- 7- Kianpisheh, Mustaffa, N., Limtrairut P. and Keikhosrokiani, P. Smart Parking System (SPS) Architecture Using Ultrasonic Detector, *International Journal of Software Engineering and Its Applications*, vol. 6, pp. 51-58, 2012.
- 8- Lynch J. P. and Loh. K. A Summary Review of Wireless Sensors and Sensor Networks for Structural Health Monitoring. *Shock and Vibration Digest*, Sage Publications,38(2):91-128, 2005.
- 9- Gibbons, P.B., Karp, B., Nath, S. and Seshan S. IrisNet: n architecture for a worldwide sensor Web. *IEEE Pervasive Computing*, 2(4):22 - 33, 2003.
- 10- Wang, Shuangyou, Junfang Tian, and Dianru Jiab. Research into A Wireless Smart Parking System. *The Italian Association of Chemical Engineering* 46 (2015): 241-246.
- 11- Print. Sparkfun, 2016. [Online]. Available: https://cdn.sparkfun.com/datasheets/Wireless/Zigbee/ds_xbeezbmodules.pdf.

Appendix A: Hardware codes

1- Gate of Garage 1& Garage 2

```
#include <FirebaseArduino.h>

#define FIREBASE_HOST ""
#define FIREBASE_AUTH ""
#define WIFI_SSID ""
#define WIFI_PASSWORD ""

#include <Servo.h>
#define TRIGGER_PIN D7
#define ECHO_PIN D5
#define SERVO_PIN D6

String fireStatus = "";           // servo status received from firebase
String gate="";

Servo servo;

long duration, distance;

void setup()
{ Serial.begin (9600);
  pinMode(TRIGGER_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  servo.attach(SERVO_PIN);
  servo.write(0);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to ");
  Serial.print(WIFI_SSID);
  while (WiFi.status() != WL_CONNECTED)
  {Serial.print(".");
    delay(500);}
  Serial.println();
  Serial.print("Connected to ");
  Serial.println(WIFI_SSID);
  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);           // connect to firebase
```

```

    Firebase.setString("firstGarage/gate", "close"); }           //send initial string of servo status

void loop() {
    long DURATION, DISTANCE;
    digitalWrite(TRIGGER_PIN, LOW);
    delayMicroseconds(2);
    digitalWrite(TRIGGER_PIN, HIGH);
    delayMicroseconds(10);
    digitalWrite(TRIGGER_PIN, LOW);
    DURATION = pulseIn(ECHO_PIN, HIGH);
    DISTANCE = (DURATION/2) / 29.1;
    fireStatus = Firebase.getString("firstGarage/gate");          // get servo status input from firebase
    if (fireStatus == "open")
    {
        Serial.println("gate is open");
        servo.write(90); // make external led ON
        if (DISTANCE <= 7){
            delay(9000);
            servo.write(0);
            Firebase.setString("firstGarage/gate", "close"); } }
    else {
        Serial.println("Command Error! Please send open");
        Serial.print("Distance = ");
        Serial.print(DISTANCE);
        Serial.println(" cm");
        delay(500);} }

```

2- Slot 1 of Garage 1&2:

```

#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <FirebaseArduino.h>
#define FIREBASE_HOST " "
#define FIREBASE_AUTH " "
#define WIFI_SSID " "

```

```

#define WIFI_PASSWORD ""
#define TRIGGER_PIN D1
#define ECHO_PIN D2
#define LED D3
int DISTANCE =0, DURATION=0;
void setup() {
  Serial.begin(9600);
  // connect to wifi.
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("connecting");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);}
  Serial.println();
  Serial.print("connected: ");
  Serial.println(WiFi.localIP());
  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
  delay(1000);
  pinMode(TRIGGER_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(LED, OUTPUT);}
void loop(){
  //Firebase.getInt("LED_STATUS");
  digitalWrite(TRIGGER_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIGGER_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIGGER_PIN, LOW);
  DURATION = pulseIn(ECHO_PIN, HIGH);
  DISTANCE = (DURATION/2) / 29.1;
  String data1 = Firebase.getString("firstGarage/A1");
  if (DISTANCE <= 7){
    Serial.println("Led Turned ON");

```

```

digitalWrite(LED,HIGH);
Firebase.setString("firstGarage/A1", "full");
// handle error
if (Firebase.failed()) {
    Serial.print("setting /number failed:");
    Serial.println(Firebase.error());
    return;}}
else if (data1 == "wait") {
    Serial.println("Led Turned off");
    digitalWrite(LED,LOW);
    Firebase.setString("firstGarage/A1", "wait");
    if (Firebase.failed()) {
        Serial.print("setting /number failed:");
        Serial.println(Firebase.error());
        return;}}
else {
    Serial.println("Led Turned off");
    digitalWrite(LED,LOW);
    Firebase.setString("firstGarage/A1", "empty");
    // handle error
    if (Firebase.failed()) {
        Serial.print("setting /number failed:");
        Serial.println(Firebase.error());
        return;}}
Serial.print("Distance = ");
Serial.print(DISTANCE);
Serial.println(" cm");
delay(500);}

```

Note: The code is the same for garage 2, but it will be ("secondGarage/A1", "empty) instead of ("firstGarage/A1", "empty)

3- Slot 2 of Garage 1&2:

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <FirebaseArduino.h>
#define FIREBASE_HOST ""
#define FIREBASE_AUTH ""
#define WIFI_SSID ""
#define WIFI_PASSWORD ""
#define TRIGGER_PIN D5
#define ECHO_PIN D6
#define LED D7
int DISTANCE =0, DURATION=0;
void setup() {
  Serial.begin(9600);
  // connect to wifi.
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("connecting");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);}
  Serial.println();
  Serial.print("connected: ");
  Serial.println(WiFi.localIP());
  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
  delay(1000);
  pinMode(TRIGGER_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(LED, OUTPUT);}
void loop(){
  //Firebase.getInt("LED_STATUS");
```

```

digitalWrite(TRIGGER_PIN, LOW);
delayMicroseconds(2);
digitalWrite(TRIGGER_PIN, HIGH);
delayMicroseconds(10);
digitalWrite(TRIGGER_PIN, LOW);
DURATION = pulseIn(ECHO_PIN, HIGH);
DISTANCE = (DURATION/2) / 29.1;
String data1 = Firebase.getString("firstGarage/A2");
if (DISTANCE <= 7){
  Serial.println("Led Turned ON");
  digitalWrite(LED,HIGH);
  Firebase.setString("firstGarage/A2", "full");
  // handle error
  if (Firebase.failed()) {
    Serial.print("setting /number failed:");
    Serial.println(Firebase.error());
    return;}}
else if (data1 == "wait") {
  Serial.println("Led Turned off");
  digitalWrite(LED,LOW);
  Firebase.setString("firstGarage/A2", "wait");
  // handle error
  if (Firebase.failed()) {
    Serial.print("setting /number failed:");
    Serial.println(Firebase.error());
    return;}}
else {
  Serial.println("Led Turned off");
  digitalWrite(LED,LOW);
  Firebase.setString("firstGarage/A2", "empty");
  // handle error
  if (Firebase.failed()) {

```

```

    Serial.print("setting /number failed:");

    Serial.println(Firebase.error());

    return;}}

Serial.print("Distance = ");

Serial.print(DISTANCE);

Serial.println(" cm");

delay(500);}

```

Note: The code is the same for garage 2, but it will be ("secondGarage/A2", "empty) instead of ("firstGarage/A2", "empty).

Appendix B: Software codes

• Main

```

import 'package:firebase_auth/firebase_auth.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/foundation.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:get_storage/get_storage.dart';
import 'package:google_maps_flutter/google_maps_flutter.dart';
import 'package:smart_parking/view/splash_screen.dart';
import 'firebase_options.dart';
main() async {
  // need to map
  if (defaultTargetPlatform == TargetPlatform.android) {
    AndroidGoogleMapsFlutter.useAndroidViewSurface = true;}
  // need to firebase
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform,);
  await GetStorage.init();
  //need to firebase auth
  FirebaseAuth.instance.currentUser;
  runApp(const MyApp());}
class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return GetMaterialApp(
      theme: ThemeData(
        bottomSheetTheme: const BottomSheetThemeData(
          backgroundColor: Colors.transparent,)),
      debugShowCheckedModeBanner: false,
      home: SplashScreen(),);}}

```

• intro

```

import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:smart_parking/view/login_signup/sign_up.dart';
import 'package:smart_parking/view/widgets/constance.dart';
import 'package:smart_parking/view/widgets/custom_bottom.dart';
import 'package:smart_parking/view/widgets/custom_text.dart';
import 'login_signup/login.dart';
class Intro extends StatelessWidget {
  // the first page intro to app to sign up or login in
  @override

```

```

Widget build(BuildContext context) {
  return Scaffold(
    body: Center(
      child: Padding(
        padding: const EdgeInsets.symmetric(horizontal: 20),
        child: SingleChildScrollView(
          child: Column(
            children: [
              const SizedBox(
                height: 150,
              ),
              CustomText(
                text: 'Start by creating an account',
                fontSize: 40,
              ),
              const SizedBox(
                height: 200,
              ),
              CustomButton(
                function: () {
                  Get.to(() => SignUpView());
                },
                text: TextString.createAccountString,
                borderRadius: 15,
                height: 60,
                width: double.infinity - 30,
              ),
              const SizedBox(
                height: 15,
              ),
              CustomButton(
                function: () {
                  Get.to(() => LoginView());
                },
                text: TextString.signInString,
                borderRadius: 15,
                height: 60,
                colorText: Colors.black,
                colorButton: secondColor,
                width: double.infinity - 30,
              ),
            ],
          ),
        ),
      ),
    ),
  );
}

```

• sign_up

```

import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:smart_parking/model_controller/login_signup/signup_controller.dart';
import 'package:smart_parking/view/widgets/constance.dart';
import 'package:smart_parking/view/widgets/custom_button.dart';
import 'package:smart_parking/view/widgets/custom_text.dart';
import 'package:smart_parking/view/widgets/custom_text_field.dart';

class SignUpView extends StatelessWidget {
  SignupController signupController = Get.put(SignupController());
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      body: Center(
        child: Padding(
          padding: const EdgeInsets.symmetric(horizontal: 20),
          child: SingleChildScrollView(
            child: Column(
              children: [
                //title
                CustomText(
                  text: TextString.createAccountString,
                  fontSize: 25,
                  fontWeight: FontWeight.bold,
                ),
                const SizedBox(
                  height: 40,
                ),
                //name, phone, password
                Form(
                  key: signupController.formState,
                  child: Column(
                    children: [
                      //name
                      CustomTextField(
                        title: TextString.nameString,
                        hint: 'Mohmed Ahmed',
                        autoFillHints: const [AutofillHints.name],

```

- **sign_up_controller**

61

```

        print(e.message);},
        codeSent: (String verificationId, int? resendToken) async {
            //after send sms
            verificationID = verificationId;
            Get.to(() => PinCodeVerificationScreen());},
            codeAutoRetrievalTimeout: (String verificationId) {}));}}
void verifyOTP() async {
    PhoneAuthCredential credential = PhoneAuthProvider.credential(
        verificationId: verificationID,
        smsCode: otpController.text);
    await auth.signInWithCredential(credential).then((value) async {
        //You are logged in successfully
        print(auth.currentUser!.uid);
        GetStorage().write("phoneNumber", phoneNumber.text);
        await FirebaseDatabase.instance.ref('users/${phoneNumber.text}').set({
            "name": name, "slotReserved": '', "startTimeOfBooking": '', "garageReserved": '',
            'phoneNumber': phoneNumber.text, 'password': password, 'inGarage': false, 'isReservation':
false,
        }).then((value) {
            Get.offAll(SwitchLogin());});});}}

```

• login

```

import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:smart_parking/model_controller/login_signup/login_controller.dart';
import 'package:smart_parking/view/widgets/constance.dart';
import 'package:smart_parking/view/widgets/custom_button.dart';
import 'package:smart_parking/view/widgets/custom_text_field.dart';
class LoginView extends StatelessWidget {
    @override
    Widget build(BuildContext context) {
        return Scaffold(
            backgroundColor: Colors.white,
            body: Padding(
                padding: const EdgeInsets.symmetric(horizontal: 15),
                child: Column(
                    mainAxisAlignment: MainAxisAlignment.center,
                    crossAxisAlignment: CrossAxisAlignment.center,
                    children: [
                        const SizedBox(
                            height: 50,),
                        Text(
                            TextString.smartParkingString,
                            style: TextStyle(
                                fontSize: 39,)),),
                        const SizedBox(
                            height: 20,),
                        Text(
                            TextString.signInString,
                            style: const TextStyle(fontSize: 30, color: primaryColor)),),
                        const SizedBox(height: 30,), signUp(), const SizedBox(height: 15,),],),),);}
    Widget signUp() {
        LoginController loginController = Get.put(LoginController());
        return Form(
            key: loginController.formState,
            child: Column(
                children: [
                    CustomTextField(
                        title: TextString.phoneNumberString,
                        hint: '0101234567',
                        textInputType: TextInputType.phone,
                        autoFillHints: const [AutoFillHints.telephoneNumber],
                        onChanged: (c) {
                            loginController.phoneNumber = c;},
                        iconData: Icons.phone,
                        validator: (value) {
                            if (value!.length > 13) {
                                return "the phone number can't to be larger than 13 numbers";
                            }
                            if (value.length < 13) {
                                return "the phone number can't to be less than 13 numbers";
                            }
                        }
                    )
                ]
            )
        );
    }
}

```

```

        return null;)),
const SizedBox(
  height: 20,),
CustomTextField(
  title: TextString.passwordString,
  hint: '*****',
  TextInputType: TextInputType.visiblePassword,
  onChanged: (c) {
    loginController.password = c;},
  iconData: Icons.password,
  validator: (value) {
    if (value!.length > 30) {
      return "the password can't to be larger than 15 letters";}
    if (value.length < 5) {
      return "the password can't to be less than 5 letters";}
    return null;)),
const SizedBox(
  height: 30,),
CustomButton(
  function: () {
    loginController.loginFun();},
  text: TextString.signInString,)),));}}

```

• login_controller

```

import 'package:firebase_database/firebase_database.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:get_storage/get_storage.dart';
import '../switch_login.dart';
class LoginController extends GetxController {
  late String phoneNumber = '', password = '';

  GlobalKey<FormState> formState = GlobalKey<FormState>();

  loginFun() async {
    if (formState.currentState!.validate()) {
      //Verify the password
      DatabaseReference starCountRef = await
FirebaseDatabase.instance.ref('users/$phoneNumber/password');
      starCountRef.get().then((value) {
        if (password == value.value) {
          GetStorage().write('phoneNumber', phoneNumber);
          Get.offAll(SwitchLogin());
        } else {
          print(password);
          print(value.value);
          //show error
          Get.snackbar(
            'Error !',"Your password is incorrect",
            snackPosition: SnackPosition.BOTTOM,backgroundColor: Colors.red.shade200,);}
      }).catchError((onError) {
        //show error
        Get.snackbar(
          'Error !',"Your phone number is incorrect",
          snackPosition: SnackPosition.BOTTOM,backgroundColor: Colors.red.shade200,);});}}

```

• verification_code

```

import 'dart:async';
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:pin_code_fields/pin_code_fields.dart';
import 'package:smart_parking/model_controller/login_signup/signup_controller.dart';
import 'package:smart_parking/view/widgets/custom_bottom.dart';
class PinCodeVerificationScreen extends StatefulWidget {
  @override
  _PinCodeVerificationScreenState createState() =>
    _PinCodeVerificationScreenState();}
class _PinCodeVerificationScreenState extends State<PinCodeVerificationScreen> {
  TextEditingController textEditingController = TextEditingController();
  StreamController<ErrorAnimationType>? errorController;

```

```

bool hasError = false;
String currentText = "";
final formKey = GlobalKey<FormState>();
@override
void initState() {
  errorController = StreamController<ErrorAnimationType>();
  super.initState();}
@override
void dispose() {
  errorController!.close();
  super.dispose();}
// snackBar Widget
snackBar(String? message) {
  return ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(
      content: Text(message!),
      duration: const Duration(seconds: 2),),),);
@override
Widget build(BuildContext context) {
  SignupController signupController = Get.put(SignupController());
  return Scaffold(
    body: GestureDetector(
      onTap: () {},
      child: SizedBox(
        height: MediaQuery.of(context).size.height,
        width: MediaQuery.of(context).size.width,
        child: ListView(
          children: <Widget>[
            const SizedBox(height: 30),
            //gif
            SizedBox(
              height: MediaQuery.of(context).size.height / 3,
              child: ClipRRect(
                borderRadius: BorderRadius.circular(30),
                child: Image.asset('assets/image/otp.gif'),),),
            const SizedBox(height: 8),
            //title
            const Padding(
              padding: EdgeInsets.symmetric(vertical: 8.0),
              child: Text(
                'Phone Number Verification',
                style: TextStyle(fontWeight: FontWeight.bold, fontSize: 22),
                textAlign: TextAlign.center,)),
            Padding(
              padding:
                const EdgeInsets.symmetric(horizontal: 30.0, vertical: 8),
              child: RichText(
                text: TextSpan(
                  text: "Enter the code sent to ",
                  children: [
                    TextSpan(
                      text: "${signupController.phoneNumber.text}",
                      style: const TextStyle(
                        color: Colors.black,
                        fontWeight: FontWeight.bold,
                        fontSize: 15)),],
                  style:
                    const TextStyle(color: Colors.black54, fontSize: 15)),
                textAlign: TextAlign.center,)),
            const SizedBox(
              height: 20,),
            Form(
              key: formKey,
              child: Padding(
                padding: const EdgeInsets.symmetric(
                  vertical: 8.0, horizontal: 30),
                child: PinCodeTextField(
                  appContext: context,
                  pastedTextStyle: TextStyle(
                    color: Colors.green.shade600,
                    fontWeight: FontWeight.bold),),

```



```

Padding(
  padding: const EdgeInsets.all(8.0),
  child: SizedBox(
    height: 60,
    width: 40,
    child: CircleAvatar(
      backgroundColor: Colors.transparent,
      child: OpenContainer(
        transitionType: homeController.transitionType,
        openBuilder: (context, VoidCallback _) {
          return PersonalPage();
        },
        closedElevation: 0.0,
        closedShape: const RoundedRectangleBorder(
          borderRadius: BorderRadius.all(
            Radius.circular(fabDimension / 2),
          ),
        ),
        closedColor: Color(0x90000000),
        closedBuilder:
          (BuildContext context, VoidCallback openContainer) {
            return Center(
              child: Icon(
                Icons.person,
                size: 30,
                color: Colors.white,
              ),
            );
          },
      ),
    ),
  ),
),
body: GetBuilder<HomeController>(
  init: HomeController(),
  builder: (c) => c.initialCameraPosition == null
    ? const Center(
        child: CircularProgressIndicator(
          color: primaryColor,
        ),
      )
    : Stack(
        alignment: AlignmentDirectional.bottomEnd,
        children: [
          GoogleMap(
            myLocationEnabled: true,
            myLocationButtonEnabled: false,
            zoomControlsEnabled: false,
            initialCameraPosition: c.initialCameraPosition!,
            onMapCreated: (GoogleMapController controllerMap) {
              c.googleMapController = controllerMap;
            },
            polylines: {
              if (c.info != null)
                Polyline(
                  polylineId: const PolylineId('overview_polyline'),
                  color: Colors.blueGrey,
                  width: 3,
                  points: c.info!.polylinePoints
                    .map((e) => LatLng(e.latitude, e.longitude))
                    .toList(),
                ),
            },
            mapType: MapType.terrain,
            markers: c.markers,
          ),
          Positioned(
            child: c.info == null
              ? Container()
              : Container(
                  height: 225,
                  decoration: BoxDecoration(
                    color: Colors.white,
                    borderRadius: BorderRadius.only(
                      topLeft: Radius.circular(30),
                      topRight: Radius.circular(30),
                    ),
                  ),
                  child: bottomSheet(),
                ),
          ),
        ],
      ),
),
Widget bottomSheet() {
  PlacesInGarageController places = Get.put(PlacesInGarageController());
  return GetBuilder<HomeController>(
    init: HomeController(),
    builder: (c) => Padding(
      padding: const EdgeInsets.only(left: 8.0, right: 8, top: 8),
      child: Column(
        children: [
          Row(
            children: [

```

```

Container(
  height: 110,
  width: 110,
  decoration: BoxDecoration(
    borderRadius: BorderRadius.circular(20),
    image: DecorationImage(
      fit: BoxFit.cover,
      image: NetworkImage(c.image!),)),),
Container(
  height: 110,
  child: Padding(
    padding: const EdgeInsets.only(left: 5.0, top: 5),
    child: SingleChildScrollView(
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        mainAxisAlignment: MainAxisAlignment.start,
        children: [
          CustomText(
            text: c.titleParking!,
            color: Colors.black,
            fontSize: 23,),
          SizedBox(
            height: 10,),
          c.userMode.toJson()['isReservation']
            ? Padding(
                padding: const EdgeInsets.only(left: 8.0),
                child: RichText(
                  text: new TextSpan(
                    style: new TextStyle(
                      fontSize: 17.0,
                      color: Colors.black,),
                  children: <TextSpan>[
                    new TextSpan(
                      text: 'Your slot is ',
                      style: new TextStyle(
                        color: primaryColor)),
                    new TextSpan(
                      text: places.slotSelected,
                      style: new TextStyle(
                        fontWeight: FontWeight.bold,
                        color: Colors.red)),],),),
                : SizedBox(),
            ),
          SizedBox(
            height: 8,),
          Padding(
            padding: const EdgeInsets.only(left: 8.0),
            child: CustomText(
              text:
                '${c.info?.totalDistance}, ${c.info?.totalDuration}',
              color: primaryColor,
              fontSize: 14,),),
          SizedBox(
            height: 10,),
          Padding(
            padding: const EdgeInsets.only(left: 8.0),
            child: c.userMode.toJson()['isReservation']
              ? CustomText(
                  text:
                    'will be paid ${ (c.resultBetweenTowDates * (c.costPerHour /
60)).toInt() } EGP',
                  color: primaryColor,
                  fontSize: 14,)
              : CustomText(
                  text: '${c.costPerHour} EGP / h ',
                  color: primaryColor,
                  fontSize: 14,),),),),
          SizedBox(
            height: 20,),
          !c.userMode.toJson()['isReservation']
            ? CustomButton(
                function: () {

```

```
//
        c.listenFirebaseUser();
        places.slotSelected = '';
        Get.to(() => PlacesInGarageView());},
        text: 'Book Now',)
    : c.userMode.toJson()['inGarage']
    ? CustomButton(
        function: () async {
            c.openGateToExit();},
        text: 'Open the gate to exit',)
    : Row(
        children: [
            Expanded(
                child: CustomButton(
                    function: () async {
                        c.openGateToCross();},
                    text: 'Open Gate',),),
            SizedBox(
                width: 10,),
            Expanded(
                child: CustomButton(
                    function: () async {
                        c.cancelOfReservation(wait: 'jnajk');},
                    text: 'Cancel',
                    colorButton: Colors.red,),),],),),),),);}
```

• home_controller

```
import 'dart:async';
import 'dart:math';
import 'package:animations/animations.dart';
import 'package:firebase_database/firebase_database.dart';
import 'package:flutter/material.dart';
import 'package:flutter_polyline_points/flutter_polyline_points.dart';
import 'package:geolocator/geolocator.dart';
import 'package:get/get.dart';
import 'package:get_storage/get_storage.dart';
import 'package:google_maps_flutter/google_maps_flutter.dart';
import 'package:smart_parking/model_controller/models/user_model.dart';
import 'package:smart_parking/model_controller/places_in_garage_controller.dart';
import 'package:smart_parking/view/widgets/custom_text_field.dart';
import 'package:directions_model.dart';
import 'package:directions_repository.dart';
class HomeController extends GetxController {
    final ContainerTransitionType transitionType = ContainerTransitionType.fade;
    late GoogleMapController googleMapController; late UserModel userMode;
    String serverTitleGarage = ''; double? myLatitude = null, myLongitude = null;
    String? titleParking, image;
    late int resultBetweenTowDates;
    late int costPerHour;
    //all marker
    Set<Marker> markers = Set(); //markers for google map
    CameraPosition? initialCameraPosition;
    //to make a path between two points
    Directions? info;
    PolylinePoints polylinePoints = PolylinePoints();
    Map<PolylineId, Polyline> polylines = {};
    List<LatLng> polylineCoordinates = [];
    @override
    void onClose() {
        googleMapController.dispose();
        super.onClose();}
    @override
    onInit() async {await listenFirebaseUser(); super.onInit();}
    listenFirebaseUser() async {
        //to get data user as stream
        DatabaseReference starCountRef = await FirebaseDatabase.instance
            .ref('users/${GetStorage().read('phoneNumber')}/');
        await starCountRef.onValue.listen((DatabaseEvent event) async {
            print('event ${event.snapshot.value}');
            userMode = UserModel.fromJson(
                Map<String, dynamic>.from(event.snapshot.value as dynamic));
        });
    }
    //get my location
```

```

        await determinePosition().then((value) {
            myLatitude = value.latitude; myLongitude = value.longitude;
            initialCameraPosition = CameraPosition(
                target: LatLng(myLatitude!, myLongitude!), zoom: 13.9);
            update();});
        if (userMode.toJson()['isReservation']) {
            calculateBetweenTwoDates();}
        if (userMode.toJson()['startTimeOfBooking'] != '') {
            final dateNow = DateTime.now();
            final difference = dateNow
                .difference(DateTime.parse(userMode.toJson()['startTimeOfBooking'] as String)).inHours;
            await FirebaseDatabase.instance
                .ref('${userMode.toJson()['garageReserved']}').child(userMode.toJson()['slotReserved']).get().then((
            value) async {
                if (value.value == 'wait' && difference == 1) {await cancelOfReservation(wait:
                'wait');});}
            //to get markers
            DatabaseReference markersFromFirebase =
            await FirebaseDatabase.instance.ref('markers');
            await markersFromFirebase.get().then((value) async {
                List listMarkers = value.value as List;
                await createMarkers(listMarkers: listMarkers);
                await getLastLocation(listMarkers: listMarkers);
                update();});
            update();});}
//get permission gps
Future<Position> determinePosition() async {
    bool serviceEnabled;
    LocationPermission permission;
    serviceEnabled = await Geolocator.isLocationServiceEnabled();
    if (!serviceEnabled) {
        return Future.error('Location services are disabled');}
    permission = await Geolocator.checkPermission();
    if (permission == LocationPermission.denied) {
        permission = await Geolocator.requestPermission();
        if (permission == LocationPermission.denied) {
            return Future.error("Location permission denied");}}
    if (permission == LocationPermission.deniedForever) {
        return Future.error('Location permissions are permanently denied');}
    Position position = await Geolocator.getCurrentPosition();
    return position;}
// create Marker
createMarkers({listMarkers}) {
    for (int i = 0; i < listMarkers.length; i++) {
        markers.add(
            Marker(infoWindow: InfoWindow(
                title: listMarkers[i]['title'],),
                onTap: () {
                    // Get directions
                    if (!userMode.toJson()['isReservation']) {
                        getPolyLines(
                            latitudeEnd: listMarkers[i]['latitude'],
                            longitudeEnd: listMarkers[i]['longitude'],
                            title: listMarkers[i]['title'],
                            serverTitle: listMarkers[i]['serverTitle'],
                            cost: listMarkers[i]['cost'],
                            image: listMarkers[i]['imageUrl']);
                    } else {
                        Get.snackbar('Note !', "You must cancel the reservation in the other garage",
                            snackPosition: SnackPosition.BOTTOM,
                            backgroundColor: Colors.red.shade200,));
                        markerId: MarkerId('id-$i'),
                        position: LatLng(
                            listMarkers[i]['latitude'], listMarkers[i]['longitude']));
                    update();}
                print(markers);}
            getPolyLines(
                {latitudeEnd, longitudeEnd, title, serverTitle, cost, image}) async {
                    final directions = await DirectionsRepository().getDirections(
                        origin: LatLng(myLatitude!, myLongitude!),
                        destination: LatLng(latitudeEnd, longitudeEnd));

```

```

        info = directions;costPerHour = cost;titleParking = title;serverTitleGarage =
serverTitle;this.image = image;update();}
    addPolyLine(List<LatLng> polylineCoordinates) {
        PolylineId id = PolylineId("poly");
        Polyline polyline = Polyline(
            polylineId: id,
            color: Colors.blueGrey,
            points: polylineCoordinates,
            width: 2,);
        polylines[id] = polyline;
        update();}
    openGateToExit() async {
        String randomNumber = GetRandomNum();
        showDialog(
            context: Get.context!, // barrierDismissible: barrierDismissible,
            builder: (BuildContext dialogContext) {
                return openGateDialog(
                    onPressed: () async {
                        PlacesInGarageController places = Get.put(
                            PlacesInGarageController());
                        await FirebaseDatabase.instance
                            .ref('users/${GetStorage().read('phoneNumber')}')
                            .update({'inGarage': false,'isReservation': false,'slotReserved':
'', 'garageReserved': '', 'startTimeOfBooking': ''
                            });
                        await FirebaseDatabase.instance.ref(serverTitleGarage).update({'gate': 'open',
places.slotSelected: 'empty'});
                        Get.back();
                        Get.snackbar('Note !',
                            "The gate is open, please cross now ,amount has been deducted
${(resultBetweenTowDates *
                            (costPerHour / 60)).toInt()} EGP",
                            snackPosition: SnackPosition.TOP,
                            backgroundColor: Colors.green.shade200,);
                        places.slotSelected = '';
                        update();}, randomNumber: randomNumber);});}
    GetRandomNum() {
        const _chars = '123456789';
        Random _rnd = Random();
        String getRandomString(int length) =>
            String.fromCharCodes(Iterable.generate(
                length, (_) => _chars.codeUnitAt(_rnd.nextInt(_chars.length)));
        return getRandomString(4).toString();}
    openGateToCross() async {
        String randomNumber = GetRandomNum();
        showDialog(
            context: Get.context!, // barrierDismissible: barrierDismissible,
            // false = user must tap button, true = tap outside dialog
            builder: (BuildContext dialogContext) {
                return openGateDialog(onPressed: () async {
                    PlacesInGarageController places = Get.put(PlacesInGarageController());
                    await
FirebaseDatabase.instance.ref('users/${GetStorage().read('phoneNumber')}').update({'inGarage':
true,});
                    await FirebaseDatabase.instance.ref(serverTitleGarage).update({'gate': 'open'});
                    Get.back();
                    Get.snackbar('Note !', "The gate is open, please cross now to ${places.slotSelected}",
                        snackPosition: SnackPosition.TOP,
                        backgroundColor: Colors.green.shade200,);
                    update();}, randomNumber: randomNumber);});}
    cancelOfReservation({wait}) async {
        PlacesInGarageController places = Get.put(PlacesInGarageController());
        await FirebaseDatabase.instance.ref('users/${GetStorage().read('phoneNumber')}').update({'
            'inGarage': false,'isReservation': false,'slotReserved': '', 'garageReserved':
'', 'startTimeOfBooking': ''});
        await FirebaseDatabase.instance
            .ref(serverTitleGarage).update({'places.slotSelected: 'empty'});
        places.slotSelected = '';
        if (wait == 'wait') {
            Get.snackbar('Note !', "Booking canceled",
                snackPosition: SnackPosition.TOP,backgroundColor: Colors.red.shade200,);} else {

```

```

        Get.snackbar(
            'Note !',"Your reservation has been canceled successfully",
            snackPosition: SnackPosition.TOP,
            backgroundColor: Colors.green.shade200,));
    update();
}
calculateBetweenTowDates() {
    final date2 = DateTime.now();
    final difference = date2.difference(
        DateTime.parse(userMode.toJson()['startTimeOfBooking'] as String)).inMinutes;
    resultBetweenTowDates = difference;
    update();
}
openGateDialog({onPressed, randomNumber}) {
    String confirmationNumber = '';
    return AlertDialog(
        title: Text('Note !'),
        content: SizedBox(
            height: 127,
            child: Column(
                children: [
                    RichText(
                        text: TextSpan(
                            style: TextStyle(
                                fontSize: 17.0,
                                color: Colors.black),
                            children: <TextSpan>[
                                TextSpan(text: 'To confirm entering the portal, repeat this number  '),
                                TextSpan(
                                    text: randomNumber,
                                    style: TextStyle(
                                        fontWeight: FontWeight.bold, color: Colors.red)),],)),
                    SizedBox(
                        height: 7,),
                    CustomTextField(
                        title: 'confirmation number',
                        onChanged: (c) {
                            confirmationNumber = c;},
                        maxLength: 4,)],),),
    actions: <Widget>[
        MaterialButton(
            child: Text(
                'cancel',
                style: TextStyle(color: Colors.red)),
            onPressed: () {
                Get.back();}),
        MaterialButton(
            child: Text(
                'open gate',
                style: TextStyle(color: Colors.green)),
            onPressed: () {
                if (confirmationNumber == randomNumber) {
                    onPressed();
                } else {
                    Get.snackbar('Error !',"The confirmation number is incorrect",
                        snackPosition: SnackPosition.TOP,
                        backgroundColor: Colors.red.shade200,));}),
    ],),);
}
getLastLocation({listMarkers}) {
    if (userMode.toJson()['isReservation'] == true) {
        for (int i = 0; i < listMarkers.length; i++) {
            if (userMode.toJson()['garageReserved'] ==
                listMarkers[i]['serverTitle']) {
                PlacesInGarageController places = Get.put(PlacesInGarageController());
                print('listMarkers[i] : ${listMarkers[i]['serverTitle']}');
                getPolylines(
                    latitudeEnd: listMarkers[i]['latitude'],
                    longitudeEnd: listMarkers[i]['longitude'],
                    title: listMarkers[i]['title'],
                    serverTitle: listMarkers[i]['serverTitle'],
                    cost: listMarkers[i]['cost'],
                    image: listMarkers[i]['imageUrl']);
                places.slotSelected = userMode.toJson()['slotReserved'];
            }
        }
    }
}

```

- **personal_page**

```
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:get_storage/get_storage.dart';
import 'package:smart_parking/model_controller/home_controller.dart';
import 'package:smart_parking/view/widgets/constance.dart';
import 'package:smart_parking/view/widgets/custom_text.dart';
import '../switch_login.dart';
import 'widgets/custom_button.dart';

class PersonalPage extends StatelessWidget {
  HomeController homeController = Get.put(HomeController());
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('about'),
        centerTitle: true,
        backgroundColor: primaryColor,
      ),
      body: SafeArea(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.start,
          crossAxisAlignment: CrossAxisAlignment.center,
          children: [
            SizedBox(
              height: 40,
            ),
            Padding(
              padding: const EdgeInsets.symmetric(vertical: 10),
              child: Container(
                width: 100,
                height: 100,
                clipBehavior: Clip.antiAlias,
                decoration: BoxDecoration(
                  shape: BoxShape.circle, color: primaryColor,
                ),
                child: Icon(
                  Icons.person,
                  size: 70,
                  color: Colors.white,
                ),
              ),
            ),
            SizedBox(
              height: 20,
            ),
            CustomText(
              text: homeController.userMode.toJson()['name'],
              color: Colors.black,
              fontSize: 30,
            ),
            SizedBox(
              height: 20,
            ),
            Column(
              mainAxisAlignment: MainAxisAlignment.spaceEvenly,
              crossAxisAlignment: CrossAxisAlignment.center,
              children: [
                Container(
                  padding: const EdgeInsets.all(20.0),
                  width: 300,
                  decoration: BoxDecoration(
                    color: primaryColor,
                    borderRadius: BorderRadius.circular(20),
                  ),
                  child: CustomText(
                    text:
                      "Phone Number :
${homeController.userMode.toJson()['phoneNumber'].toString()}",
                    color: Colors.white,
                  ),
                ),
                SizedBox(
                  height: 20,
                ),
                SizedBox(
                  width: 250,
                  child: CustomButton(
                    function: () {
                      GetStorage().remove('phoneNumber');
                      Get.offAll(SwitchLogin());
                    },
                    text: 'Log out',
                  ),
                ),
              ],
            ),
          ],
        ),
      ),
    );
  }
}
```



```

sizeText: 25,
colorText: Colors.white,
colorButton: Colors.red,)),)),),),),));}}

```

• places_in_garage_view

```

import 'dart:math' as math;
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:smart_parking/model_controller/home_controller.dart';
import 'package:smart_parking/model_controller/places_in_garage_controller.dart';
import 'package:smart_parking/view/widgets/constance.dart';
import 'package:smart_parking/view/widgets/custom_button.dart';
import 'package:smart_parking/view/widgets/custom_text.dart';
class PlacesInGarageView extends StatefulWidget {
  @override
  State<PlacesInGarageView> createState() => _PlacesInGarageViewState();
class _PlacesInGarageViewState extends State<PlacesInGarageView> {
  HomeController homeController = Get.put(HomeController());
  PlacesInGarageController placesInGarageController =
    Get.put(PlacesInGarageController());
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        centerTitle: true,
        title: Text(homeController.titleParking!),
        backgroundColor: primaryColor,
        elevation: 0,),
      body:
        Container(
          height: double.infinity,
          width: double.infinity,
          color: Colors.white,
          child: Column(
            children: [
              Row(
                children: [
                  ListCarParking(true, 'A1'),
                  Expanded(
                    flex: 7,
                    child: SizedBox(
                      height: MediaQuery.of(Get.context!).size.height - 200,
                      child: Column(
                        crossAxisAlignment: CrossAxisAlignment.center,
                        mainAxisAlignment: MainAxisAlignment.end,
                        children: [
                          RotatedBox(
                            quarterTurns: -1,
                            child: CustomText(text: 'The Passage', fontSize: 30, color:
Colors.black45,)),
                          SizedBox(
                            height: 50,),
                          RotatedBox(
                            quarterTurns: 1,
                            child: Opacity(
                              opacity: 0.5,
                              child: Image(
                                image: AssetImage('assets/image/car.png'), height: 70,)),),
                          SizedBox(
                            height: 20,),
                          Icon(Icons.arrow_upward, color: Colors.black45,),
                          SizedBox(
                            height: 10,),
                          CustomText(text: 'ENTRY', color: Colors.black45,)),)),
                  ListCarParking(false, 'A2'),
                ],
              ),
              SizedBox(
                height: 30,),
              Padding(
                padding: const EdgeInsets.only(left: 15.0, right: 15),
                child: CustomButton(
                  function: () {

```

```

        PlacesInGarageController controller =
            Get.put(PlacesInGarageController());
        controller.bookNowFun();},
        text: 'book now',),),),),));}}
Widget ListCarParking(left, idSlot) {
  return Expanded(
    flex: 10,
    child: Container(
      padding: EdgeInsets.only(left: 6, right: 6),
      height: MediaQuery.of(Get.context!).size.height - 200, // width: double.infinity ,
      child: ListView.builder(
        shrinkWrap: true,
        reverse: true,
        physics: const NeverScrollableScrollPhysics(),
        itemCount: 8,
        itemBuilder: (BuildContext context, int index) {
          return CarImage(index: index, left: left, idSlot: idSlot);},),),));}
Widget CarImage({index, left, idSlot}) {
  return GetBuilder<PlacesInGarageController>(
    init: PlacesInGarageController(),
    builder: (placesInGarageController) => MaterialButton(
      onPressed: () => placesInGarageController.selectSlotFun(idSlot),
      padding: EdgeInsets.all(0),
      child: Container(
        width: double.infinity,
        decoration: left
          ? BoxDecoration(
              border: Border(
                top: BorderSide(width: 1.0, color: primaryColor),
                bottom: BorderSide(width: 1.0, color: primaryColor),
                left: BorderSide(width: 2.0, color: primaryColor),))
            : BoxDecoration(
              border: Border(
                top: BorderSide(width: 1.0, color: primaryColor),
                bottom: BorderSide(width: 1.0, color: primaryColor),
                right: BorderSide(width: 2.0, color: primaryColor),)),
        child: index > 0
          ? SizedBox(
              height: 70,
              child: Icon(
                Icons.block,
                color: Colors.red,)),)
          : (idSlot == 'A1'
              ? placesInGarageController.slotModel.toJson()['A1'] == 'empty'
                : placesInGarageController.slotModel.toJson()['A2'] == 'empty')
            ? Container(
                height: 70,
                color: placesInGarageController.slotSelected == ''
                  ? Colors.transparent
                  : placesInGarageController.slotSelected == idSlot
                    ? Colors.green.shade300
                    : Colors.transparent,
                child: CustomText(
                  text: idSlot,)),)
            : (idSlot == 'A1'
              ? placesInGarageController.slotModel.toJson()['A1'] == 'full'
                : placesInGarageController.slotModel.toJson()['A2'] == 'full')
                ? left
                  ? leftImage()
                    : imageCar()
                  : (idSlot == 'A1'
                      ? placesInGarageController.slotModel
                        .toJson()['A1'] == 'wait'
                      : placesInGarageController.slotModel
                        .toJson()['A2'] == 'wait')
                    ? Opacity(
                        opacity: 0.5,
                        child: Stack(
                          alignment: AlignmentDirectional.center,
                          children: [
                            left ? leftImage() : imageCar(),

```

```

        Positioned(
          child: Icon(
            Icons.watch_later_outlined, color: Colors.white, size: 30,)),),)
      : Container(),),),);
Widget leftImage() {
  return Transform(
    alignment: Alignment.center,
    transform: Matrix4.rotationY(math.pi),
    child: Image(
      image: AssetImage('assets/image/car.png'),
      height: 70,)),);
Widget imageCar() {
  return Image(
    image: AssetImage('assets/image/car.png'),
    height: 70,);
Widget waitingImageCar() {
  return Image(
    image: AssetImage('assets/image/car.png'),
    height: 70,);
}

```

• places_in_garage_controller

```

import 'package:firebase_database/firebase_database.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:smart_parking/model_controller/home_controller.dart';
import 'package:smart_parking/model_controller/models/model_slot.dart';
import '../view/payment/home_payment_view.dart';

class PlacesInGarageController extends GetxController {
  var slotSelected = '';
  late SlotModel slotModel;
  selectSlotFun(idSlot) {
    if (idSlot == 'A1'
        ? slotModel.toJson()['A1'] == 'empty'
        : slotModel.toJson()['A2'] == 'empty') {
      slotSelected = idSlot;
      update();
      print(slotSelected);
    } else {
      Get.snackbar(
        'Notes !', "You are not able to choose this place",
        snackPosition: SnackPosition.BOTTOM,
        backgroundColor: Colors.red.shade200,));
    }
  }
  @override
  onInit() async {
    await listenFirebaseUser();
    super.onInit();
  }
  listenFirebaseUser() async {
    //to get data my garage as stream
    print('listenFirebaseUser');
    HomeController homeController = Get.put(HomeController());
    DatabaseReference starCountRef = await
    FirebaseDatabase.instance.ref(homeController.serverTitleGarage);
    await starCountRef.onValue.listen((DatabaseEvent event) {
      slotModel = SlotModel.fromJson(
        Map<String, dynamic>.from(event.snapshot.value as dynamic));
      print(slotModel);
      update();});
  }
  void bookNowFun() {
    if (slotSelected != '') {
      Get.to(() => HomePaymentView());
    } else {
      Get.snackbar(
        'Notes !', "You should select any slot before booking",
        snackPosition: SnackPosition.BOTTOM, backgroundColor: Colors.red.shade200,));
    }
  }
}

```

• home_payment_view

```

import 'package:firebase_database/firebase_database.dart';
import 'package:flutter/material.dart';
import 'package:flutter_credit_card/flutter_credit_card.dart';
import 'package:get/get.dart';

```

```

import 'package:get_storage/get_storage.dart';
import 'package:smart_parking/model_controller/home_controller.dart';
import 'package:smart_parking/model_controller/places_in_garage_controller.dart';
import 'package:smart_parking/view/home_view.dart';
import 'package:smart_parking/view/payment/new_payment_view.dart';
import 'package:smart_parking/view/widgets/custom_text.dart';
import '../widgets/constance.dart';

class HomePaymentView extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        centerTitle: true,
        title: Text(TextString.yourCards),
        backgroundColor: primaryColor,
        elevation: 0,
      ),
      body: Padding(
        padding: const EdgeInsets.only(right: 15.0, left: 15, top: 15),
        child: Column(
          children: [
            //add new car button
            Container(
              height: 80,
              decoration: BoxDecoration(
                color: primaryColor, borderRadius: BorderRadius.circular(20)),
              child: Padding(
                padding: const EdgeInsets.only(
                  left: 15,
                  right: 20,),
                child: MaterialButton(
                  padding: const EdgeInsets.all(0),
                  onPressed: () {Get.to(() => NewPaymentView());},
                  child: Row(
                    mainAxisAlignment: MainAxisAlignment.spaceBetween,
                    children: [
                      CustomText(
                        text: TextString.addNewCard,
                        fontSize: 20,
                        color: Colors.white,),
                      Container(height: 30,
                        width: 30,
                        decoration: BoxDecoration(
                          color: Colors.white70,
                          borderRadius: BorderRadius.circular(5)),
                        child: const Icon(
                          Icons.add,
                          color: Colors.black,)),),),),), //list cards
                    ],
                  ),
                ),
              ),
            ),
          ),
        ),
      ),
    );
  }
}

StreamBuilder(
  stream: FirebaseDatabase.instance
    .ref('users/${GetStorage().read('phoneNumber')}').child('card').onValue,
  builder: (context, snapshot) {
    if (!snapshot.hasData) {
      return Container();
    }
    if (snapshot.hasError) {
      return Container();
    }
    List values;
    try {
      DatabaseEvent dataValues =
        snapshot.data! as DatabaseEvent; //here's the typo;
      values = dataValues.snapshot.value as List;
      print(values);
    } catch (e) {
      values = [];
      print('catch : $e');
    }
    return Expanded(
      child: ListView.builder(
        shrinkWrap: true,
        itemCount: values.length,
        itemBuilder: (BuildContext context, int index) {

```



```

var listCards = [];
try {
    list = value.value as List;
} catch (e) {
    list = [];
}
final data = {'cardNumber': cardNumber, 'expiryDate': expiryDate, 'cardHolderName':
cardHolderName, 'cvvCode': cvvCode,};
print("data " + data.toString());
print("list " + list.toString());
if (list.isNotEmpty) {
    listCards.addAll(list);
}
listCards.addAll([data]);
print("listCard " + listCards.toString());
await
FirebaseDatabase.instance.ref('users/${GetStorage().read("phoneNumber")}')
.child('card').set(listCards).then((value) {
    Get.back();});});});});

```

نظام مراقبة السيارات الذكي

أعداد السيارات تزداد يوميًا، ونتيجةً لذلك، يزداد الزحام. وبالتالي، فمواقف السيارات غالبًا ما تكون ممتلئة، ولا يمكن لأي شخص أن يعثر على مكان فارغ ليضع سيارته فيه. وإذا كان يوجد أماكن فارغة لوضع السيارات فيها، فالموضوع سوف يستغرق وقتًا كثيرًا.

لحل هذه المشاكل التي تم ذكرها سابقًا، لقد تم بناء هذا النظام الذكي. هذا النظام الذكي سوف يساعد هؤلاء الأشخاص على إيجاد أماكن فارغة بسهولة وفي وقت قصير.

هذا النظام ينقسم إلى قسمين: القسم الأول مرتبط بالمكونات المادية، والقسم الثاني مرتبط بالتطبيق.

القسم الأول يحتوي على: أجهزة الاستشعار بالموجات فوق الصوتية، لمبات، أسلاك، مقومات، محرك كهربائي، ولوح إلكترونية.

القسم الثاني يحتوي على: التطبيق الذي سوف يتم استخدامه للبحث عن أقرب مكان فارغ وحجزه بكل سهولة. يستطيع المستخدم أن يستخدم بطاقات الائتمان مسبقة الدفع مثل بطاقات (ميزة)، وذلك لتقليل أخطار استخدام بطاقات الائتمان الشخصية، مما يزيد الأمان.

أيضًا، فالمستخدم يمكنه أن يعتمد ويثق كل الثقة في هذا التطبيق طوال الوقت.

وأخيرًا وليس آخرًا، نأمل أن يستفاد الكثير من الناس من هذا النظام الذكي كما خططنا له.