



Parking Finder Application

A graduation project report submission
In partial fulfilment of the requirements for the award of the degree
Bachelor of Science

Submitted by:

Hesham Radi Elsaady	89431
Belal Abdrabo Said	89411
Solafa Salem Ahmed	89409
Esraa Adel Emam	89517

Under the supervision of Supervisor(s):

Dr. Maged Khafagy

TA. Ahmed Abdallah

TA. Shereen Youssif

Department of Computer Science - CS
Misr University for Science and Technology - MUST
College of Computers and Artificial Intelligence Technologies - CAIT
June 2023



ACKNOWLEDGMENT

We would like to express our special thanks of gratitude to Dr. Maged Khafagy, who gave us a golden opportunity to this wonderful project on Parking Finder Application. We came to know so many new things that would help us a lot in our career and we are highly indebt to Dr. Maged Khafagy for his guidance and constant supervision, as well as providing us with the necessary information regarding the project and also for his support during the process of completing the project. His constant guidance and willingness to his vast knowledge made us understand this project and its manifestations in great depths that helped us to complete the assigned tasks on time. Our thanks and appreciation also go to our colleagues who helped us develop this project and people who are willingly helped us out with their position/abilities.



DECLARATION

I hereby certify that this work, which I now submit for assessment on the program of study leading to the award of Bachelor of Science in *(insert title of degree for which registered)* is entirely my own work, that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others and to the extent that such work has been cited and acknowledged within the references section of this report.

Signed: _____

Registration No.: _____

Date: Sunday, July 2nd, 2023



ABSTRACT

The obstacle of locating a vacant spot in streets that are also available is not an easy task for people. As drivers waste a lot of their time and effort driving through the streets trying to find a spot and they get more unsatisfied with this process that they go through repeatedly during their daily life. Hence, our purpose and motivation are to try to solve the previous issue. We want to save people's time and effort they exert every day, so we thought of an application that provide the user to book a parking hourly, daily or monthly and this will be pre- pay with visa or Fawry, also user can view the locations with their number of vacant spots and decide which place is better and which they would like to head to. Our project mainly focuses on solving the parking issue whether in the streets and help user to get his desired destination with estimated duration and price.

KEYWORDS: mobile application; parking system; parking QR.

TABLE OF CONTENTS

LIST OF FIGURES	i
LIST OF TABLES	ii
LIST OF ACRONYMS/ABBREVIATIONS.....	1
1 INTRODUCTION.....	3
1.1 Overview	5
1.2 Problem Definition	5
1.3 Project Objective	6
1.4 Motivation	6
1.5 Application Requirements	6
1.6 Software/Hardware Tools.....	6
2 LITERATURE SURVEY AND RELATED WORKS	9
2.1 Background and Related Works	11
2.1.1 QR-code Technology	11
2.1.2 ParkMobile	11
2.1.3 Best Parking	12
2.1.4 SpotHero.....	12
2.1.5 Parkopedia.....	12
2.1.6 Parking.com.....	12
2.1.7 Your Parking Space.....	12
2.1.8 Parking Telecom	12
2.1.9 Parknav.....	13
3 SYSTEM ANALYSIS AND DESIGN	15
3.1 Requirements determination.....	17
3.1.1 Functional Requirements FR.....	17
3.1.2 Non-Functional Requirements NFR	17
3.2 UML Diagrams.....	18
3.2.1 Context Models.....	18
3.2.1.1 Use-Case Diagram.....	18
3.2.1.2 Context Diagram	20
3.2.1.3 Process Diagram.....	22
3.2.2 Interaction Models	24
3.2.2.1 Sequence Diagram.....	24
3.2.3 Structural Models.....	28
3.2.3.1 Class Diagram	29
3.2.3.2 Entity Relationship Diagram (ERD).....	32
3.2.3.3 Data Flow Diagram (DFD)	33
3.2.3.3.1 Child Diagram.....	35
3.2.4 Behavioral Models.....	36
3.2.4.1 State-Chart Diagram.....	36
3.2.4.2 Activity Diagram.....	38
4 PROPOSED SYSTEM.....	43
4.1 Overview	45
4.2 System Architecture	45
4.3 Algorithms.....	46
4.4 Methodologies.....	46
4.5 Implemented Algorithms	48
4.5.1 E-mail & password Authentication Algorithm.....	48
4.6 Analysis of the primary results	48
4.7 Time plan	49



5	RESULTS AND DISCUSSION	51
5.1	Screenshots of our application	53
5.1.1	Login	53
5.1.2	Sign Up.....	54
5.1.3	Home Page	55
5.1.4	Drawer.....	56
5.1.5	Parking gates	57
5.1.6	Parking spots	57
5.1.7	Choose time.....	59
5.1.8	Unavailable time	60
5.1.9	Booking	61
5.1.10	Payment.....	62
5.1.11	Fawry	63
5.1.12	Visa	64
5.1.13	Payment integration (Visa).....	65
5.1.14	Verification	66
5.1.15	Approved.....	67
5.1.16	Declined	68
5.1.17	History.....	69
5.1.18	Feedback	69
5.1.19	QR-code	70
5.1.20	Map	71
	CONCLUSION	72
	REFERENCES.....	73

LIST OF FIGURES

Figure 3.2.1.1 Use-case diagram	20
Figure 3.2.1.2 Context diagram	21
Figure 3.2.1.3 Process diagram	23
Figure 3. 2.2.1-1 Sequence for registration	26
Figure 3.2.2.1-2 Sequence for login	26
Figure 3.2.2.1-3 Sequence for updating.....	27
Figure 3.2.2.3-4 Sequence for payment.....	27
Figure 3.2.2.3-5 Sequence for location.....	28
Figure 3.2.3.1 Class diagram.....	31
Figure 3.2.3.2-1 ERD	32
Figure 3.2.3.2-2 ERD	33
Figure 3.2.3.3-1 DFD	34
Figure 3.2.3.3-2 DFD	35
Figure 3.2.3.3.1 Child diagram.....	36
Figure 3.2.4.1 State-chart diagram	38
Figure 3.2.4.2 Activity diagram.....	41
Figure 4.2 System Architecture	45
Figure 4.5.Implemented Algorithm 1	48
Figure 5.1.1 Login	53
Figure 5.1.2 Sign Up.....	54
Figure 5.1.3- 1Home page	55
Figure 5.1.3- 2Home page	55
Figure 5.Drawer 1.4.....	56
Figure 5.1. 5Parking gates	57
Figure 5.1.6-1 Parking spots.....	57
Figure 5.1.6-2 Parking spots.....	58
Figure 5.1.6-3 Parking spots.....	58
Figure 5.1.6-4 Parking spots.....	59
Figure 5.1.7-1 Choose time	59
Figure 5.1.7-2 Choose time	60
Figure 5.1.8 Unavailable time	60
Figure 5.1.9-1 Booking.....	61
Figure 5.1.9-2 Booking.....	61
Figure 5.1.10 Payment.....	62
Figure 5.1.11 Fawry.....	63
Figure 5.1.12 Visa	64
Figure 5.1.13 Payment integration	65
Figure 5.1.14 Verification	66
Figure 5.1.15 Approved.....	67
Figure 5.1.16 Declined	68
Figure 5.1.17 History.....	69
Figure 5.1.18 Feedback	69
Figure 5.1.19-1 QR-code	70
Figure 5.1.19-2 QR-code	70
Figure 5.1.20 Map	71



LIST OF TABLES

Table 1	19
Table 2	21
Table 3	22
Table 4	25
Table 5	30
Table 6	37
Table 7	40



LIST OF ACRONYMS/ABBREVIATIONS

- QR Quick Response
- FR Functional Requirements
- NFR Non-Functional Requirements
- UML Unified Modeling Language
- ID Identification
- ERD Entity Relationship Diagram
- DFD Data Flow Diagram
- FDD Feature Driven Development
- ASD Adaptive System Development
- DSDM Dynamic System Development Method
- XP Extreme Programming
- OTP One Time Password
- VS Visual Studio





Chapter I

Introduction



Chapter One

INTRODUCTION

1.1 Overview

Nowadays the world is getting bigger and the digital world is being expanded. In addition to the issue of the expansion of population all over the world especially in Egypt. It is thought that most people already own at least one car. The number of drivers is getting bigger on a daily basis, and more people are having an issue that they cannot find a place to park their car easily and it takes them a lot of time. Hence, parking slots are getting more insufficient every day. The obstacle of locating a vacant spot in streets that are also available is not an easy task for people. As drivers waste a lot of their time and effort driving through the streets trying to find a spot and they get more unsatisfied with this process that they go through repeatedly during their daily life. Furthermore, our purpose and motivation are to try to improve utilization of objects detection by taking pictures or live frames of streets and garages to solve the previous issue as it will increase the efficiency in a digital way using technology. We want to save people's time and effort they exert every day. Our Project mainly focuses on solving the parking issue whether in the streets or garages. We all are encountering that issue on a daily basis and it leads to wasting a lot of time and effort to find a vacant spot. Our Goal is to facilitate the operation of finding vacant spots in the crowded streets, the uncrowded streets and the garages, also to minimize the messy process of parking. It is about having an online system that detects unoccupied spots in certain areas via object detection. Our app allows user to compare spots, rates, and pre-pay for parking.

1.2 Problem Definition

Our problem can be defined as the unorganized way of parking in Egypt. Drivers can waste much time trying to find a vacant spot in the street or nearby garage to park their cars. Such as searching for a vacant spot in a street especially if it's crowded it may take a long time, or searching for empty spot can take a long time too. Which may lead to the possibility of parking the car in an unauthorized or unsafe parking spot and result in even more crowded and unorganized streets.



1.3 Project Objective

As it raised our attention to the problem of the unorganized process of parking in the streets and the difficulty to find all garages in specific areas. It became our objective to solve this problem by using our application interface the user would be directed to a vacant spot of his own choice it will become an option for the driver to be directed to the nearest parking so users can search for specific places, share them with others and choose locations to head to. They directed to them easily with needing to pre-search for them and prepaid.

1.4 Motivation

What new technology does is create new opportunities to do a job that customers want done, so our motivation is to utilize the growth in technology in order to facilitate people's life and make their daily obstacles become easier. The beneficiary from this project is to everyone who wastes a lot of time in the process of locating a vacant spot, our purpose is to make an application that facilitates the recently mentioned dilemma based on computer vision and solves the problem of crowded streets.

1.5 Application Requirements

- Reduce wasting time for users.
- Provide user easy payment by Visa/Fawry.
- Provide users flexibility to express their feedback freely.
- Provide users ease of searching for a vacant parking space.

1.6 Software/Hardware Tools

Software Tools:

- **Operating system:**
Windows, Unix, Linux.
IOS , Android.
- **Coding language:**
Flutter, Dart.



- **Editor:**

Android Studio.

VS code.

Hardware Tools:

- **Computer, laptop:**

With Ram minimum of 2 GB.

With 20+ hard disk, SSD preferred

With CPU Core i3 running 2.5GHz or higher

- **Mobile Phone:**

Working with Android or IOS.





Chapter II

Literature survey and Related works





Chapter Two

LITERATURE SURVEY AND RELATED WORKS

This chapter is talk about a background information related to the QR code technology, as well as an overview of some of the related applications.

2.1 Background and Related work

2.1.1 QR-code Technology

The term QR-code stands for Quick Response Code. QR codes are square barcodes (two-dimensional barcodes) that were first developed and used in Japan. Like any other barcode, a QR code is nothing more than a way to store information in a machine-readable optical label. The data contained by a QR code can be anything from simple text to email addresses, phone numbers, and so on.

QR codes store data using patterns of black dots and white spaces arranged in a square grid. These patterns can be scanned and translated into human-readable information, with the help of an imaging device, like a camera or a scanner. However, the most common way to scan QR codes nowadays is to use your smartphone's camera and a specialized app for reading QR codes [1].

2.1.2 ParkMobile

Using ParkMobile, you pay and park easily using your mobile. Get the facility to arrange garage and stadium parking ahead of time. ParkMobile is offered in St. Louis, Minneapolis, Pittsburgh, Philadelphia, Washington, DC, San Francisco, Atlanta, Chicago, Miami, New Orleans, and Dallas [2].

2.1.3 Best Parking

Best Parking is a parking finder app that helps you find parking in many major cities like New York City (Manhattan), San Francisco, Chicago, Boston, Los Angeles, Seattle, Atlanta, and Toronto [3].



2.1.4 SpotHero

SpotHero is a digital parking reservation service. The drivers can use this parking finder app to help them connect with the nearest parking. The drivers can look for a location to park in their cities and can also reserve in advance. They can pay via credit card within the app itself. They run a website, app as well as a parking developer platform [4].

2.1.5 Parkopedia

Allows drivers to find the closest parking to their destination, tells them how much it will cost and whether the space is available. It is not only a parking finder app, but it is also a provider of on-street and off-street parking data. It provides all sorts of digital parking services. Parkopedia has spread across 90 countries and covers over 15,000 cities, including China, Australia, New Zealand, and South American countries [5].

2.1.6 Parking.com

It serves locations and lots in over 80 cities, including Boston, New York City, Washington DC, Chicago, Philadelphia, and Los Angeles. Whether you're looking for city parking, commuter parking, event parking, airport, or even discount parking, this app allows you to compare spots, rates, and pre-pay for parking [6].

2.1.7 Your Parking Space

The YourParkingSpace platform connects drivers with over 350,000 privately owned and commercially operated parking spaces across the UK, available to book hourly, daily, or monthly basis. Drivers can book parking on-demand through a website and mobile application. Can find spots on streets, airports, stadiums and train stations [7].

2.1.8 Parking Telecom

It provides a parking solution offered by Parking Telecom. Parking Telecom Mobile App is created for both parkers and parking owners. Users can locate parking spots easily, check availability in real time, and book their stay to make sure space is available when they arrive. Parking owners can bring in new visitors and get extra profit by publishing their parking spaces on the platform. Industrial IoT, AI and Big Data are used in creating this application [8].

2.1.9 Parknav

A system that solves the issue of finding parking lots which provides off-street and on-street real-time parking availability. Its services are available for mobile apps, car apps and websites. Using Machine Learning and Big Data in analyzing and collecting data about traffic, streets, cars ...etc [9].

App/Feature	SpotHero	Best parking	Parknav	Parkmobile	Our app
Pre-booking	X	X			X
Feedback	X				X
Bookmark	X			X	
Free trial			X		
Not free	X	X	X	X	X
Cancel reservation	X				
Scan code					X





Chapter III

System Analysis and Design





Chapter Three

SYSTEM ANALYSIS AND DESIGN

The analysis phase takes the general ideas in the business requirements and refines them into a detailed requirements definition, functional models, structural models, and behavioral models.

3.1 Requirements determination

The purpose of the requirements determination step is to turn the very high-level explanation of the business requirements into a more precise list of requirements that can be used as inputs to the rest of analysis (creating functional, structural, and behavioral models). This expansion of the requirements ultimately leads to the design phase.

3.1.1 Functional Requirements FR

1. Registration and login.
2. Request a location to park at.
3. Choose whether to go to streets
4. System returns list of locations available to the user's desired destination with estimated duration and price.
5. Choose to get directions to one of the locations returned.
6. Route directions from the user's location to the chosen spot are illustrated.
7. Users can view reviews about the chosen location, and can also add review.
8. Users can add specific locations to their list of bookmarks.
9. When a user starts a trip, the place is added to their history.
10. Users can edit their info (username, email, phone number, password).

3.1.2 Nonfunctional Requirements NFR

1. Availability

That the app is available for all age groups.

2. Reliability

We guarantee the reliability of our application so that the user's information and data cannot be found by anyone else.



3. Scalability

Is the measure of a system's ability to increase or decrease in performance in response to changes in application and system processing demands.

4. Performance

Application performance refers to the quality and efficiency at which the application operates.

5. Security

The system shall provide password-protected access to login/Register pages that are to be viewed only by admins.

6. Usability

The measure of how effectively, efficiently, and easily a user can navigate an interface, find information on it, and achieve his or her goals.

7. Extensibility

The ability for a system or components of a system to expand by assimilating new data, software or hardware components.

3.2 UML diagrams

3.2.1 Context models

Context model defines how context data are structured and maintained (It plays a key role in supporting efficient context management). It aims to produce a formal or semi-formal description of the context information that is present in a context-aware system. In other words, the context is the surrounding element for the system, and a model provides the mathematical interface and a behavioral description of the surrounding environment. Context models which include *use case, context and process diagrams*.

3.2.1.1 Use-Case Diagram

Creating use-case diagrams is a two-step process: First, the users work with the project team to write text-based *use-case descriptions*; second, the project team translates the use-case descriptions into formal *use-case diagrams*. Both the use-case descriptions and the use-case diagrams are based on the identified requirements and the activity diagram description of the business process. Use-case descriptions contain all the information needed to produce use-case diagrams.

Through the creation of use-case descriptions, users can describe the required details of each individual use case. A use-case description expresses the information in a less formal way that is usually simpler for users to understand.


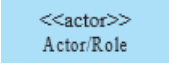




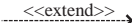

An actor: <ul style="list-style-type: none"> Is a person or system that derives benefit from and is external to the subject. Is depicted as either a stick figure (default) or if a nonhuman actor is involved, as a rectangle with <<actor>> in it (alternative). Is labeled with its role. Can be associated with other actors using a specialization/superclass association, denoted by an arrow with a hollow arrowhead. Is placed outside the subject boundary. 	 Actor/Role 
A use case: <ul style="list-style-type: none"> Represents a major piece of system functionality. Can extend another use case. Can include another use case. Is placed inside the system boundary. Is labeled with a descriptive verb-noun phrase. 	
A subject boundary: <ul style="list-style-type: none"> Includes the name of the subject inside or on top. Represents the scope of the subject, e.g., a system or an individual business process. 	
An association relationship: <ul style="list-style-type: none"> Links an actor with the use case(s) with which it interacts. 	
An include relationship: <ul style="list-style-type: none"> Represents the inclusion of the functionality of one use case within another. Has an arrow drawn from the base use case to the used use case. 	<<include>> 
An extend relationship: <ul style="list-style-type: none"> Represents the extension of the use case to include optional behavior. Has an arrow drawn from the extension use case to the base use case. 	<<extend>> 
A generalization relationship: <ul style="list-style-type: none"> Represents a specialized use case to a more generalized one. Has an arrow drawn from the specialized use case to the base use case. 	

Table 1

A use-case diagram illustrates in a very simple way the main functions of the system and the different kinds of users that will interact with it. Use cases are connected to actors through association relationships; these relationships show with which use cases the actors interact.

An actor is not a specific user, but instead is a role that a user can play while interacting with the system. An actor can also represent another system in which the current system interacts. Actors can provide input to the system, receive output from the system, or both.

A use case, depicted by an oval in the UML, is a major process that the system will perform that benefits an actor or actors in some way, and it is labeled using a descriptive verb–noun phrase. There are times when a use case includes, extends, or generalizes the functionality of another use case in the diagram. Based on the functional requirements and the activity diagram. In this section we show the main use cases for the system and the actors deal with it. First we have a user, the user will login/sign up, choose spot, enter location, show/clear history, add review, add/delete bookmark and booking then we have the admin that's generate the QR Code for user, set user level and permission and manage the app totally.

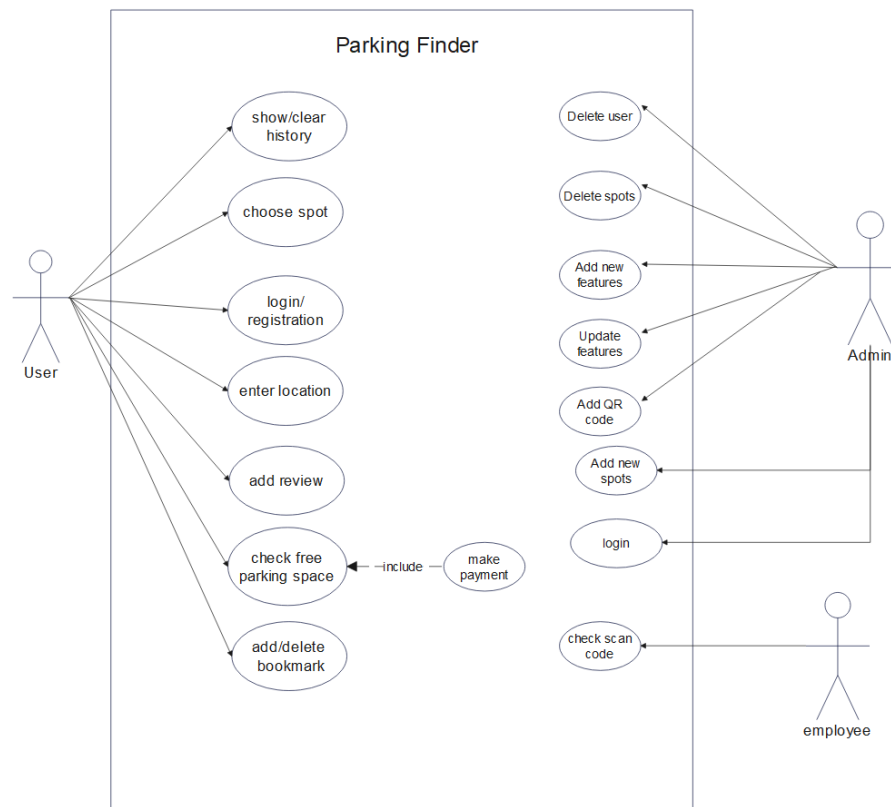


Figure Use-case diagram 3.2.1.1

3.2.1.2 Context Diagram

Also referred to as the Level 0 Data Flow Diagram, the Context diagram is the highest level in a Data Flow Diagram. It is a tool popular among Business Analysts who use it to understand the details and boundaries of the system to be designed in a project. It points out the flow of information between the system and external components.

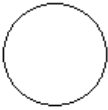
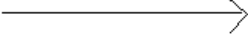

Symbol	Meaning
 Process	Single process: A circle is used to represent the entire system.
	Data flow: An arrow is used to represent the flow of data between the process and external entities.
	External entity: A square or rectangle represents any person or organisation that sends data to or receives data from the system.

Table 2

We have two kinds of people dealing with our app: admin and user, First the user will login/sign up, search of available spots and booking. Second we have the admin, the admin will login, generate the QR Code for user, set user level and permission and manage the app totally.

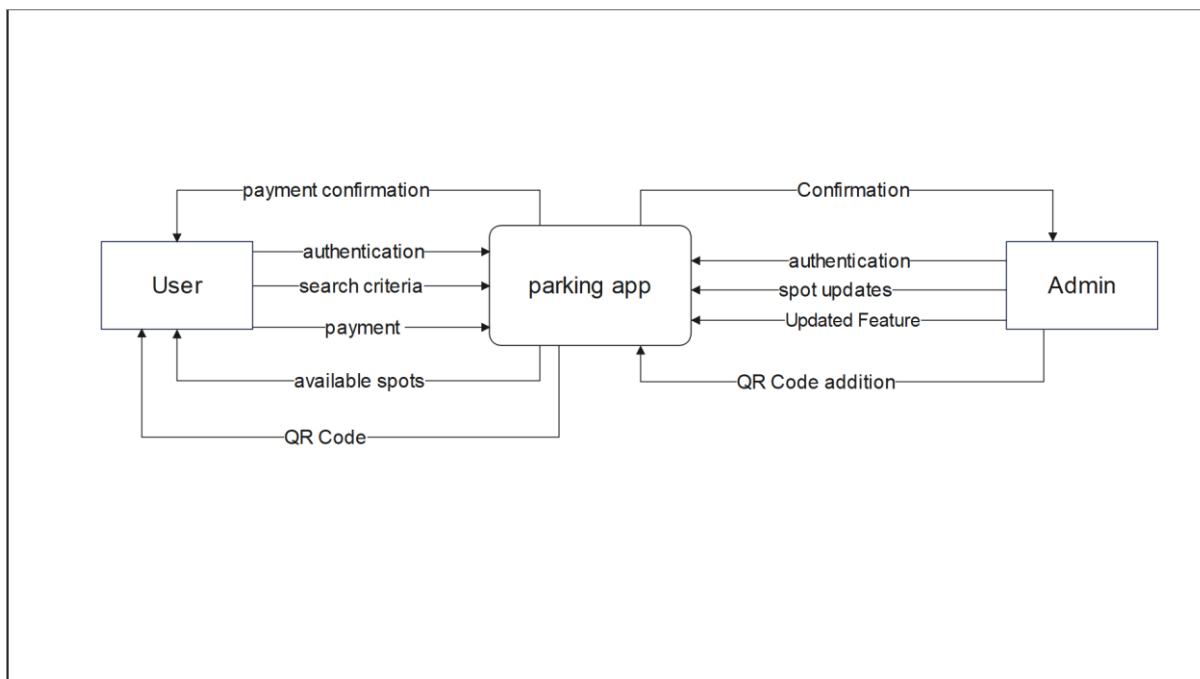


Figure Context diagram 3.2.1.2

3.2.1.3 Process Diagram

Also called Process Flow Diagram (PFD) is a type of flowchart that illustrates the relationships between major components at an industrial plant. It's most often used in chemical engineering and process engineering, though its concepts are sometimes applied to other processes as well. It's used to document a process, improve a process or model a new one. Depending on its use and content, it may also be called a Process Flow Chart, Flow sheet, Block Flow Diagram, Schematic Flow Diagram, Macro Flowchart, Top-down Flowchart, Piping and Instrument Diagram, System Flow Diagram or System Diagram. They use a series of symbols and notations to depict a process. The symbols vary in different places, and the diagrams may range from simple, hand-drawn scrawls or sticky notes to professional-looking diagrams with expandable detail, produced with software.







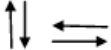
Symbol	Name	Function
	Process	Indicates any type of internal operation inside the Processor or Memory
	input/output	Used for any Input / Output (I/O) operation. Indicates that the computer is to obtain data or output results
	Decision	Used to ask a question that can be answered in a binary format (Yes/No, True/False)
	Connector	Allows the flowchart to be drawn without intersecting lines or without a reverse flow.
	Predefined Process	Used to invoke a subroutine or an Interrupt program.
	Terminal	Indicates the starting or ending of the program, process, or interrupt program
	Flow Lines	Shows direction of flow.

Table 3

First the user will login/sign up. Second the use search for parking spot if a spot is available the user can book the spot else the user search again for available spot. Third the user book the spot.

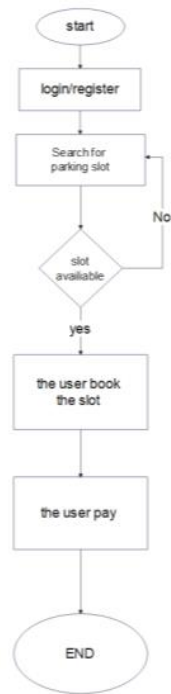


Figure Process diagram 3.2.1.3

3.2.2 Interaction models

An interaction model is a design model that binds an application together in a way that supports the conceptual models of its target users. Interaction models which include *use case and sequence diagrams*.

3.2.2.1 Sequence Diagram

Sequence diagrams are one of two types of interaction diagrams. They illustrate the objects that participate in a use case and the messages that pass between them over time for one use case. A sequence diagram is a dynamic model that shows the explicit sequence of messages that are passed between objects in a defined interaction. Because sequence diagrams emphasize the time-based ordering of the activity that takes place among a set of objects, they are very helpful for understanding real-time specifications and complex use cases.

Sequence diagrams describe interactions among classes in terms of an exchange of messages over time. They're also called event diagrams. A sequence diagram is a good way to visualize and validate various runtime scenarios. These can help to predict how a system will behave and to discover responsibilities a class may need to have in the process of modelling a new system.

Actors and objects that participate in the sequence are placed across the top of the diagram using actor symbols from the use-case diagram and object symbols from the object diagram.

A dotted line runs vertically below each actor and object to denote the lifeline of the actors/objects over time. Sometimes an object creates a temporary object; in this case an X is placed at the end of the lifeline at the point the object is destroyed (not shown).

A thin rectangular box, called the execution occurrence, is overlaid onto the lifeline to show when the classes are sending and receiving messages. A message is a communication between objects that conveys information with the expectation that activity will ensue. There are many different types of messages that can be portrayed on a sequence diagram. However, in the case of using sequence diagrams to model use cases, two types of messages are typically used: operation call and return.

Operation call messages passed between classes are shown using solid lines connecting two objects with an arrow on the line showing which way the message is being passed. Argument values for the message are placed in parentheses next to the message's name. The order of messages goes from the top to the bottom of the page, so messages located higher on the diagram represent messages that occur earlier on in the sequence, versus the lower messages that occur later.

A return message is depicted as a dashed line with an arrow on the end of the line portraying the direction of the return. The information being returned is used to label the arrow. However, because adding return messages tends to clutter the diagram, unless the return messages add a lot of information to the diagram, they can be omitted.


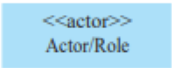



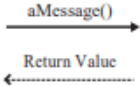
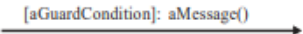


Term and Definition	Symbol
An actor: <ul style="list-style-type: none"> Is a person or system that derives benefit from and is external to the system. Participates in a sequence by sending and/or receiving messages. Is placed across the top of the diagram. Is depicted either as a stick figure (default) or, if a nonhuman actor is involved, as a rectangle with <<actor>> in it (alternative). 	 anActor 
An object: <ul style="list-style-type: none"> Participates in a sequence by sending and/or receiving messages. Is placed across the top of the diagram. 	
A lifeline: <ul style="list-style-type: none"> Denotes the life of an object during a sequence. Contains an X at the point at which the class no longer interacts. 	
An execution occurrence: <ul style="list-style-type: none"> Is a long narrow rectangle placed atop a lifeline. Denotes when an object is sending or receiving messages. 	
A message: <ul style="list-style-type: none"> Conveys information from one object to another one. A operation call is labeled with the message being sent and a solid arrow, whereas a return is labeled with the value being returned and shown as a dashed arrow. 	
A guard condition: <ul style="list-style-type: none"> Represents a test that must be met for the message to be sent. 	
For object destruction: <ul style="list-style-type: none"> An X is placed at the end of an object's lifeline to show that it is going out of existence. 	
A frame: <ul style="list-style-type: none"> Indicates the context of the sequence diagram. 	

Table 4

We identified in a sequence diagram one actor, api and database.

First sequence diagram for registration: user can sign up and the information of user saved in the database and the database check if the user is exist or not, Second sequence diagram for login, Third sequence diagram for updating in this diagram the user can update his data (delete or add) information, Fourth sequence diagram for payment and in this diagram the user enter data of his CreditCard and database will check the CreditCard is correct or not, Fifth sequence diagram for location in this diagram the user can choose spot or location after that the user interface send the user request to map to check the location is available or not.

-Sequence diagram for registration

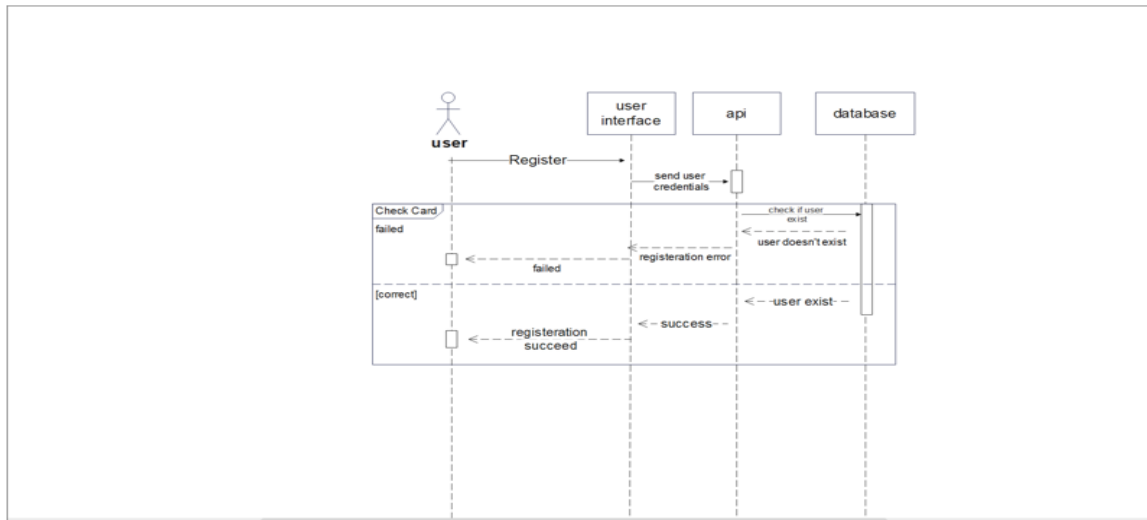


Figure. Sequence for registration 3.2.2.1-1

-Sequence diagram for login

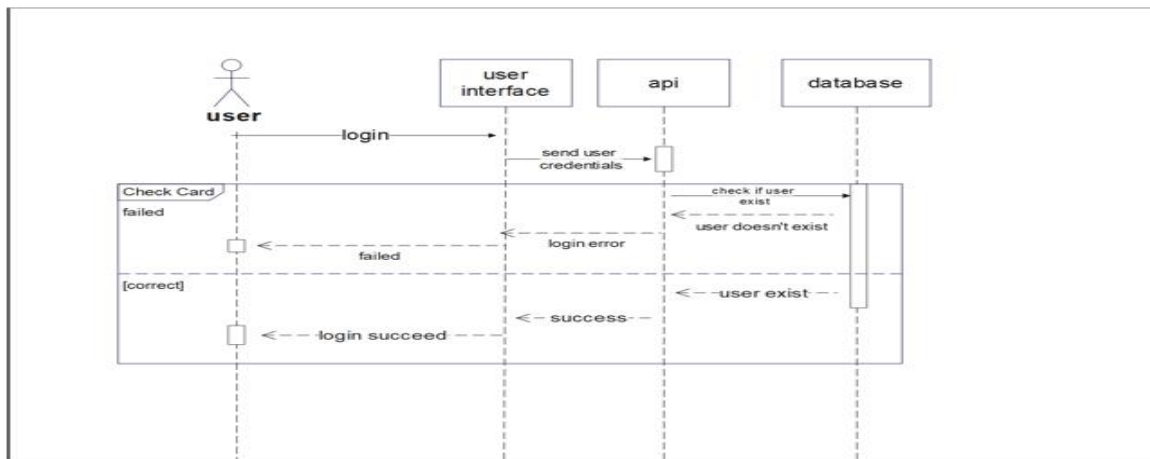


Figure. Sequence for login 3.2.2.1-2

- Sequence diagram for updating

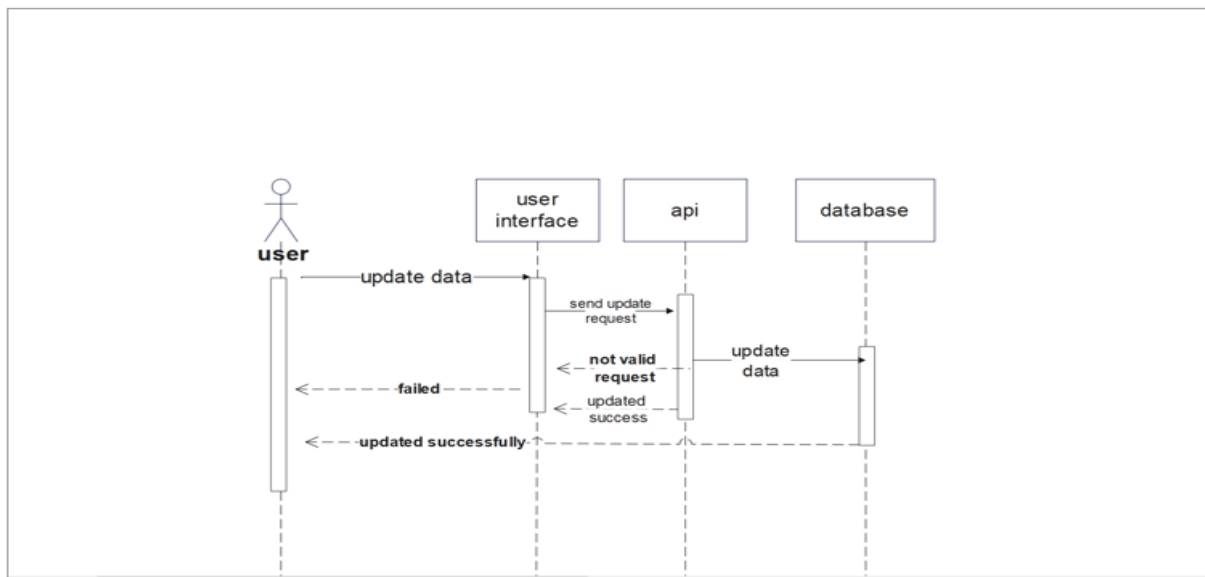


Figure. Sequence for updating 3.2.2.1-3

-Sequence diagram for payment

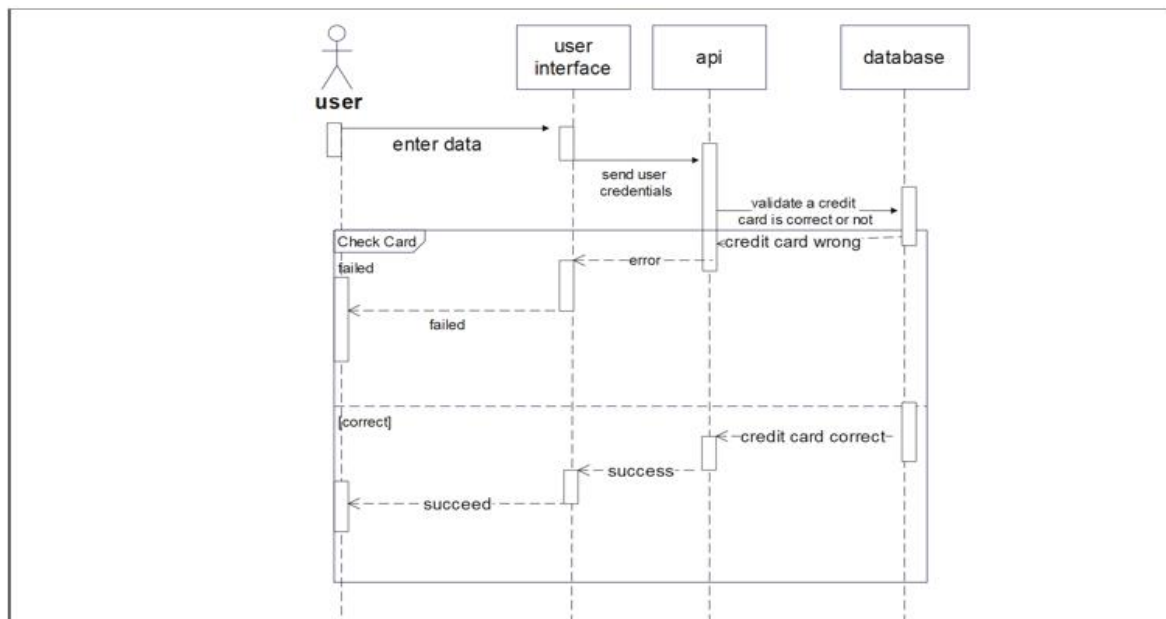


Figure. Sequence for payment 3.2.2.1-4

-Sequence diagram for location

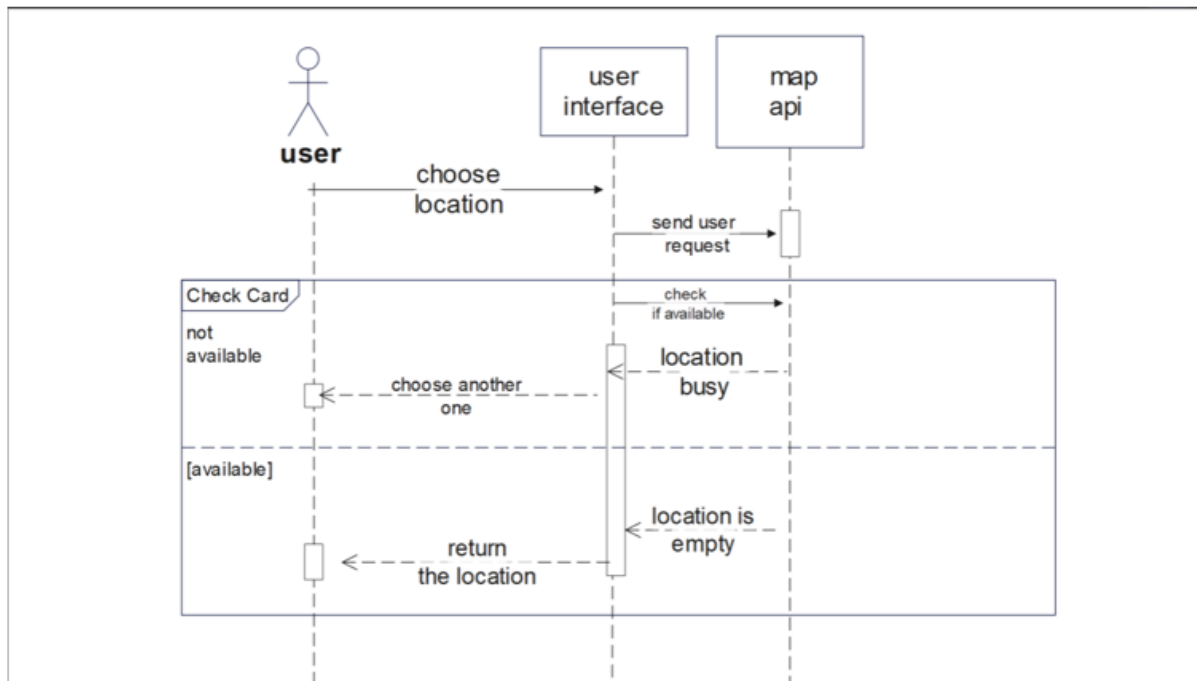


Figure. Sequence for register 3.2.2.1-5

3.2.3 Structural models

Structural or conceptual, model describes the structure of the data that supports the business processes in an organization. During analysis, the structural model presents the logical organization of data without indicating how the data are stored, created, or manipulated so that analysts can focus on the business, without being distracted by technical details. Later during design, the structural model is updated to reflect exactly how the data will be stored in databases and files.

Structural models of software display the organization of a system in terms of the components that make up that system and their relationships. Structural models may be static models, which show the structure of the system design or dynamic models, which show the organization of the system when it is executing. These are not the same things—the dynamic organization of a system as a set of interacting threads may be very different from a static model of the system components. Structural models which include *class*, *entity relationship* and *data flow diagrams*.

3.2.3.1 Class Diagram

A class diagram is a static model that shows the classes and the relationships among classes that remain constant in the system over time. The class diagram depicts classes, which include both behaviors and states, with the relationships between the classes. The main building block of a class diagram is the class, which stores and manages information in the system. During analysis, classes refer to the people, places, events, and things about which the system will capture information. Later, during design and implementation, classes can refer to implementation-specific artifacts such as windows, forms, and other objects used to build the system. Each class is drawn using three part rectangles, with the class's name at top, attributes in the middle, and operations at the bottom.

Attributes are properties of the class about which we want to capture information. Store derived attributes are attributes that can be calculated or derived; these special attributes are denoted by placing a slash (/) before the attribute's name. It is also possible to show the visibility of the attribute on the diagram. Visibility relates to the level of information hiding to be enforced for the attribute. The visibility of an attribute can either be public (), protected (#), or private (). A public attribute is one that is not hidden from any other object. As such, other objects can modify its value. A protected attribute is one that is hidden from all other classes except its immediate subclasses. A private attribute is one that is hidden from all other classes. The default visibility for an attribute is normally private.

Operations are actions or functions that a class can perform. The functions that are available to all classes are not explicitly shown within the class rectangle. Instead, only those operations unique to the class are included. An operation has no parameters, the parentheses are still shown but are empty. As with attributes, the visibility of an operation can be designated as public, protected, or private. The default visibility for an operation is normally public.

There are three kinds of operations that a class can contain: constructor, query, and update. A constructor operation creates a new instance of a class. As we just mentioned, if an operation implements one of the basic functions it is not explicitly shown on the class diagram, so typically we do not see constructor methods explicitly on the class diagram.

A query operation makes information about the state of an object available to other objects, but it does not alter the object in any way. If a query method merely asks for information from attributes in the class then it is not shown on the diagram because we assume that all objects have operations that produce the values of their attributes. An update operation changes the value of some or all the object's attributes, which may result in a change in the object's state.

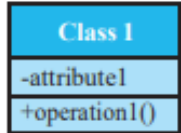
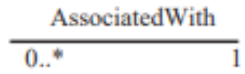



<p>A class:</p> <ul style="list-style-type: none"> • Represents a kind of person, place, or thing about which the system will need to capture and store information. • Has a name typed in bold and centered in its top compartment? • Has a list of attributes in its middle compartment? • Has a list of operations in its bottom compartment? • Does not explicitly show operations that are available to all classes. 	 <pre> classDiagram class Class1 { -attribute1 +operation1() } </pre>
<p>An attribute:</p> <ul style="list-style-type: none"> • Represents properties that describe the state of an object. • Can be derived from other attributes, shown by placing a slash before the attribute's name. 	<p>attribute name /derived attribute name</p>
<p>An operation:</p> <ul style="list-style-type: none"> • Represents the actions or functions that a class can perform. • Can be classified as a constructor, query, or update operation. • Includes parentheses that may contain parameters or information needed to perform the operation. 	<p>operation name ()</p>
<p>An association:</p> <ul style="list-style-type: none"> • Represents a relationship between multiple classes or a class and itself. • Is labeled using a verb phrase or a role name, whichever better represents the relationship. • Can exist between one or more classes. • Contains multiplicity symbols, which represent the minimum and maximum times a class instance can be associated with the related class instance. 	 <pre> classDiagram class1 "0..*" -- "1" class2 : AssociatedWith </pre>
<p>A generalization:</p> <ul style="list-style-type: none"> • Represents a-kind-of relationship between multiple classes. 	 <pre> classDiagram class1 -- > class2 </pre>
<p>An aggregation:</p> <ul style="list-style-type: none"> • Represents a logical a-part-of relationship between multiple classes or a class and itself. • Is a special form of an association? 	 <pre> classDiagram class1 "0..*" -- "1" class2 : IsPartOf </pre>
<p>A composition:</p> <ul style="list-style-type: none"> • Represents a physical a-part-of relationship between multiple classes or a class and itself. • Is a special form of an association? 	 <pre> classDiagram class1 "1..*" -- "1" class2 : IsPartOf </pre>

Table 5

First the admin class which hold the attributes for admin such his name, password, email, birthday, gender, photo and phone number.

It also shows the operations he is capable of like generate QR Code for user.

Second the user class which hold the attributes for user such his name, password, phone number, email and information of his CreditCard.

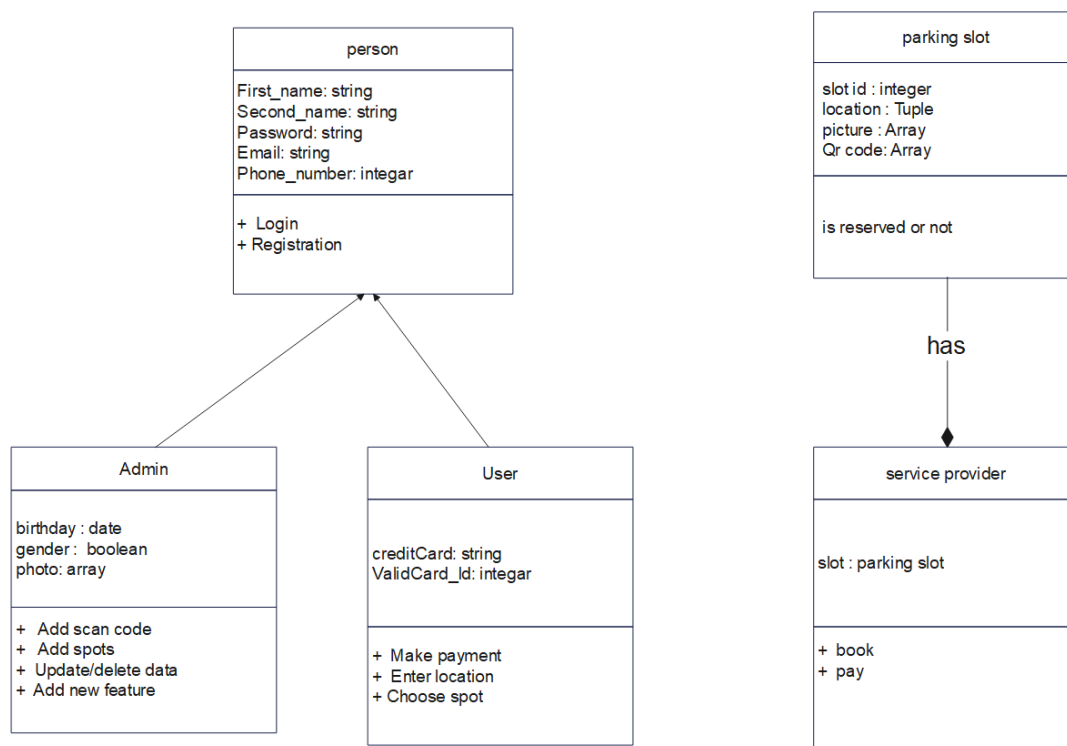


Figure. Class diagram 3.2.3.1

3.2.3.2 Entity Relationship Diagram (ERD)

Entity Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system. ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education and research. Also known as ERDs or ER Models, they use a defined set of symbols such as rectangles, diamonds, ovals and connecting lines to depict the interconnectedness of entities, relationships and their attributes. ER diagrams are related to data structure diagrams (DSDs), which focus on the relationships of elements within entities instead of relationships between entities themselves.

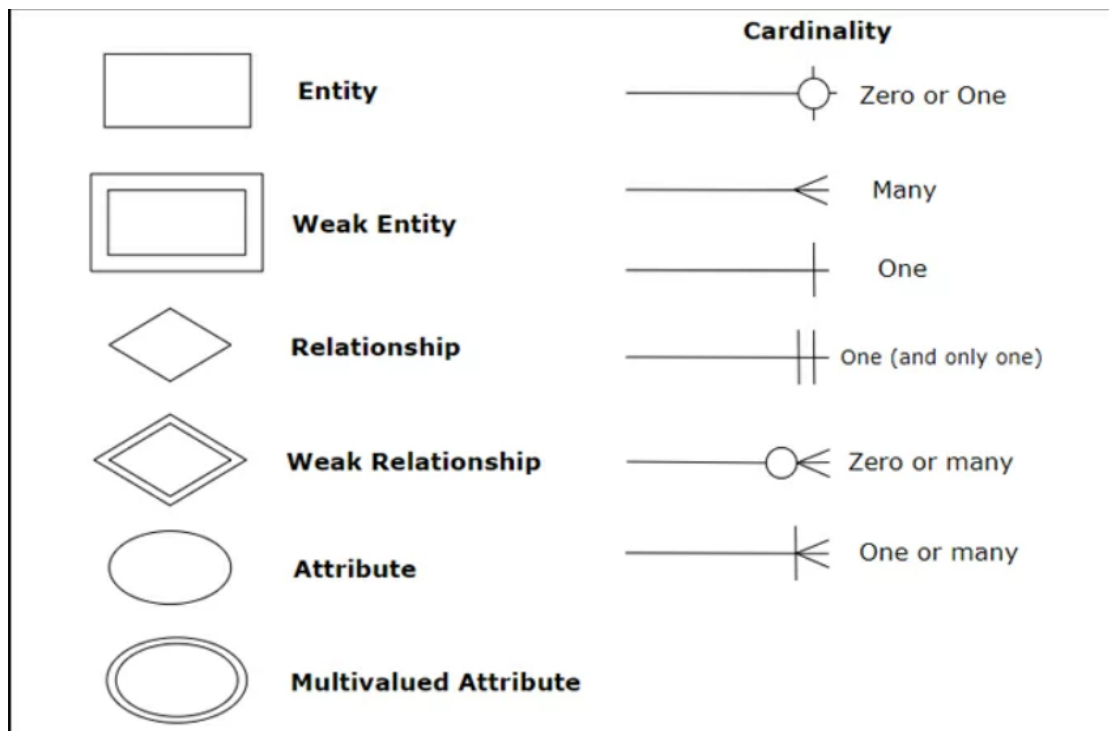


Figure. ERD 3.2.3.2-1

In our app we have user and admin and each of them has some attributes saved in our database like user has (First name, last name, email, phone number and password) and Admin has (First name, last name, email, phone number, date of birth and ID number) and each of them has function in the app which is the user send requests and the admin manage the app.



Figure 0.3.2-2

3.2.3.3 Data Flow Diagram (DFD)

Data-flow diagram is a way of representing a flow of data through a process or a system (usually an information system). The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow there are no decision rules and no loops. Specific operations based on the data can be represented by a flowchart.

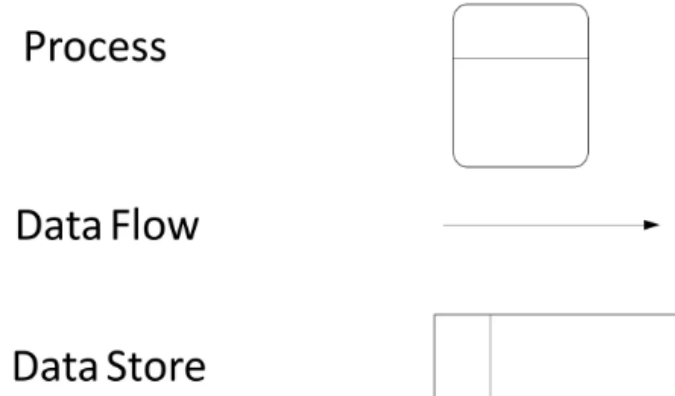


Figure. DFD 3.2.3.3-1

DFD means how data is processed through process where the user enter his data like name and email to sign in/sign up and this data stored in our database for authentication, then the user search for his slot to choose the area he wants then he book his slot to choose his desired destination with estimated duration and price then pay and he has two options first by Fawry and second by Visa. The admin also enter his data for authentication to be able to enter the app to manage it like do updates on spots and updates feature to help user to search or use the app in easy way.

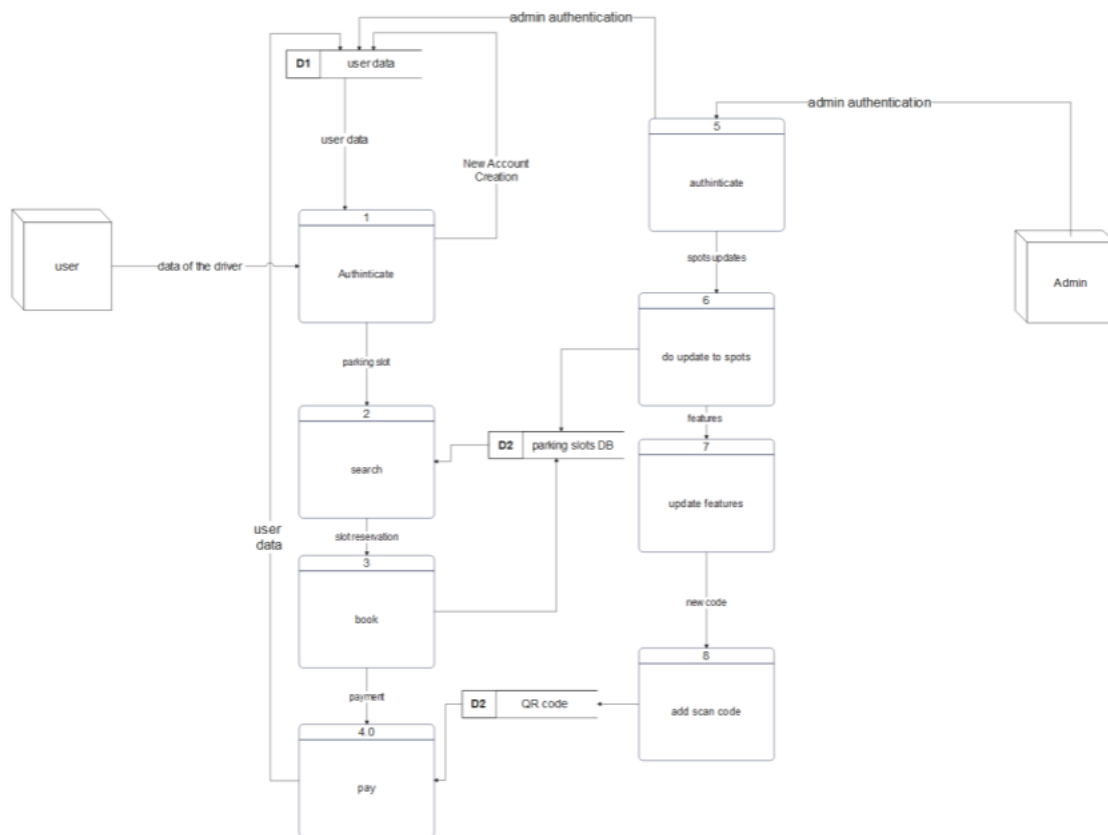


Figure. DFD 3.2.3.3-2

3.2.3.3.1 Child Diagram

process 4.1 of payment has two ways as may the user enter wrong information of his credit card so we validate it first and if there is something wrong , a message will appear to him to retry and then he will be able to see our QR code

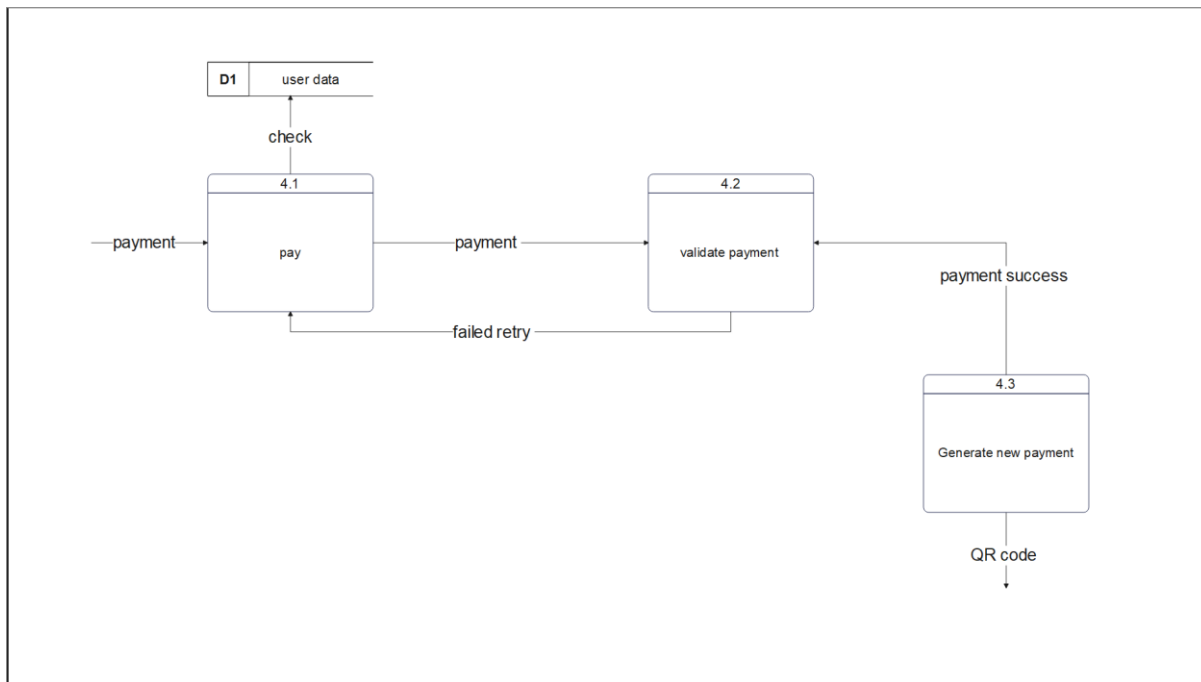


Figure. Child diagram 3.2.3.3.1

3.2.4 Behavioral models

Describe the internal dynamic aspects of an information system that supports the business processes in an organization. During analysis, behavioral models describe what the internal logic of the processes is without specifying how the processes are to be implemented. Later, in the design and implementation phases, the detailed design of the operations contained in the object is fully specified. Behavioral model which include *state chart and activity diagrams*.

3.2.4.1 State-chart Diagram

It's a type that used to describe the behavior of systems. Is composed of a finite number of states; sometimes, this is indeed the case, while at other times this is a reasonable abstraction.

The state of an object is defined by the value of its attributes and its relationships with other objects at a particular point in time.

An event is something that takes place at a certain point in time and changes a value or values that describe an object, which, in turn, changes the object's state. **A transition** is a relationship that represents the movement of an object from one state to another state. Some transitions have a guard condition. A guard condition is a Boolean expression that includes attribute values, which allows a transition to occur only if the condition is true.

An action is an atomic, non-decomposable process that cannot be interrupted. From a practical perspective, actions take zero time, and they are associated with a transition. **An activity** is a non-atomic, decomposable process that can be interrupted. Activities take a long period of time to complete, they can be started and stopped by an action, and they are associated with states.







Term and Definition	Symbol
A state: <ul style="list-style-type: none"> ■ Is shown as a rectangle with rounded corners. ■ Has a name that represents the state of an object. 	
An initial state: <ul style="list-style-type: none"> ■ Is shown as a small, filled-in circle. ■ Represents the point at which an object begins to exist. 	
A final state: <ul style="list-style-type: none"> ■ Is shown as a circle surrounding a small, filled-in circle (bull's-eye). ■ Represents the completion of activity. 	
An event: <ul style="list-style-type: none"> ■ Is a noteworthy occurrence that triggers a change in state. ■ Can be a designated condition becoming true, the receipt of an explicit signal from one object to another, or the passage of a designated period of time. ■ Is used to label a transition. 	
A transition: <ul style="list-style-type: none"> ■ Indicates that an object in the first state will enter the second state. ■ Is triggered by the occurrence of the event labeling the transition. ■ Is shown as a solid arrow from one state to another, labeled by the event name. 	
A frame: <ul style="list-style-type: none"> ■ Indicates the context of the behavioral state machine. 	

Table 6

A state is shown as a rectangle with rounded corners, has a name that represents the state of an object.

An initial state is shown as a small, filled-in circle, represents the point at which an object begins to exist.

A final state is shown as a circle surrounding a small, filled-in circle (bull's-eye), Represents the completion of activity. **An event** is a noteworthy occurrence that triggers a change in state, can be a designated condition becoming true, the receipt of an explicit signal from one object to another, or the passage of a designated period of time and is used to label a transition. **A transition** indicates that an object in the first state will enter the second state, is triggered by the occurrence of the event labeling the transition and is shown as a solid arrow from one state to another, labeled by the event name.

A frame indicates the context of the behavioral state machine.

In this diagram we are showing an overview of listing process and how it is going to work. Real estate listing goes through several different states over time.

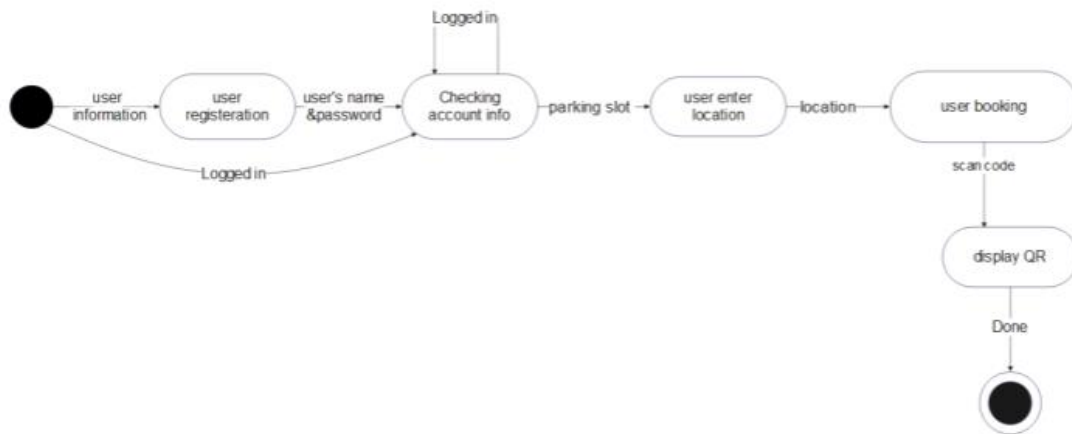


Figure. State-chart diagram 3.2.4.1

3.2.4.2 Activity Diagram

Activity diagrams depict the primary activities and the relationships among the activities in a process. An **activity diagram** is essentially a flowchart that shows activities performed by a process. As a first step, the project team gathers requirements from the users. Next, using the gathered requirements, the project team models the overall business process using activity diagrams. Next the team uses the identified activities to pinpoint the use cases that occur in the business.

Activities are depicted in an activity diagram as a *rounded rectangle*. Furthermore, they should have a name that begins with a *verb* and ends with a *noun*. Activities typically modify or transform objects.

Object nodes model these objects in an activity diagram. Object nodes are depicted in an activity diagram as *rectangles*. Essentially, object nodes represent the flow of information from one activity to another activity.

Activity diagram is essentially an advanced version of flow chart that modeling the flow from one activity to another activity. Activity Diagrams describe how activities are coordinated to provide a service which can be at different levels of abstraction.



Typically, an event needs to be achieved by some operations, particularly where the operation is intended to achieve a number of different things that require coordination, or how the events in a single use case relate to one another, in particular, use cases where activities may overlap and require coordination.

There are two different types of *flows* in activity diagrams: **control flow** and **object flow**. **Control Flows** model the paths of execution through a business process. A control flow is depicted as a solid line with an arrowhead on it showing the direction of flow. Control flows can be attached only to activities.

Object flows model the flow of objects through a business process. Because activities modify or transform objects, object flows are necessary to show the actual objects that flow into and out of the activities. An object flow is depicted as a dashed line with an arrowhead on it showing the direction of flow. An individual object flow must be attached to an activity on one end and an object node on the other end.

There are seven different types of **control nodes** in an activity diagram: *initial*, *final-activity*, *final-flow*, *decision*, *merge*, *fork*, and *join*. An **initial node** depicts the beginning of a set of activities. An initial node is shown as a small, filled-in circle. A **final-activity node** is used to stop the process being modeled. A final-activity node is represented as a circle surrounding a small, filled-in circle. A **final-flow node** is like a final-activity node, except that it stops a specific path of execution through the business process but allows the other concurrent or parallel paths to continue. A final-flow node is shown as a small circle with an X in it.

The **decision** and **merge** nodes support modeling the decision structure of a business process. The **decision node** is used to represent the actual test condition that determines which of the paths exiting the decision node is to be traversed. The **merge node** is used to bring back together multiple mutually exclusive paths that have been split based on an earlier decision.

The **fork** and **join** nodes allow parallel and concurrent processes to be modeled. The **fork node** is used to split the behavior of the business process into multiple parallel or concurrent flows.

The purpose of the **join node** is like that of the merge node. The join node simply brings back together the separate parallel or concurrent flows in the business process into a single flow.



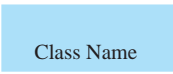

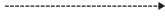



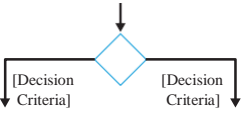

An action: <ul style="list-style-type: none"> Is a simple, nondecomposable piece of behavior. Is labeled by its name. 	
An activity: <ul style="list-style-type: none"> Is used to represent a set of actions. Is labeled by its name. 	
An object node: <ul style="list-style-type: none"> Is used to represent an object that is connected to a set of object flows. Is labeled by its class name. 	
A control flow: <ul style="list-style-type: none"> Shows the sequence of execution. 	
An object flow: <ul style="list-style-type: none"> Shows the flow of an object from one activity (or action) to another activity (or action). 	
An initial node: <ul style="list-style-type: none"> Portrays the beginning of a set of actions or activities. 	
A final-activity node: <ul style="list-style-type: none"> Is used to stop all control flows and object flows in an activity (or action). 	
A final-flow node: <ul style="list-style-type: none"> Is used to stop a specific control flow or object flow. 	
A decision node: <ul style="list-style-type: none"> Is used to represent a test condition to ensure that the control flow or object flow only goes down one path. Is labeled with the decision criteria to continue down the specific path. 	
A merge node: <ul style="list-style-type: none"> Is used to bring back together different decision paths that were created using a decision node. 	

Table 7

In this diagram we are showing an overview of whole system that how it is going to work. The fork and join nodes allow parallel and concurrent processes to be modeled. We have two kinds of people dealing with our app : admin and user , First we have two kinds of user , old user and new user , the old user will login directly but the new user need to register first then both of them have the ability to book an spot and can change the location of spot after booking it. Second we have the admin, the admin generate the QR Code for user, set user level and permission and manage the app totally.

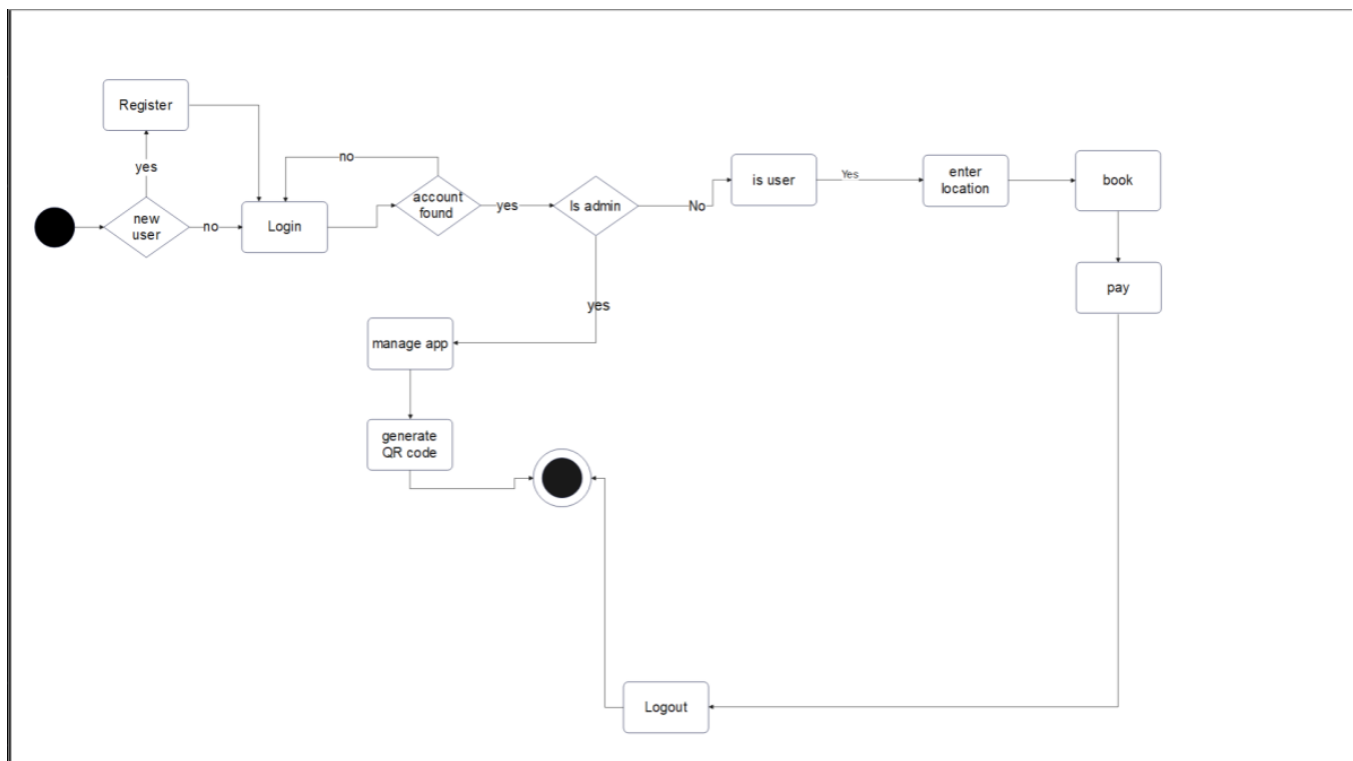


Figure. Activity diagram 3.2.4.2





Chapter IV

Proposed System



Chapter Four

PROPOSED SYSTEM

4.1 Overview

The user enter his information then he needs to look for parking spaces in the area that he wants and put the time and date; then, he books a spot by entering his vehicle plate number, before he reaches the parking lot, the app will guide him to the assigned parking space to park the vehicle. The parking fee can be paid through the app itself by visa or fawry.

4.2 System Architecture

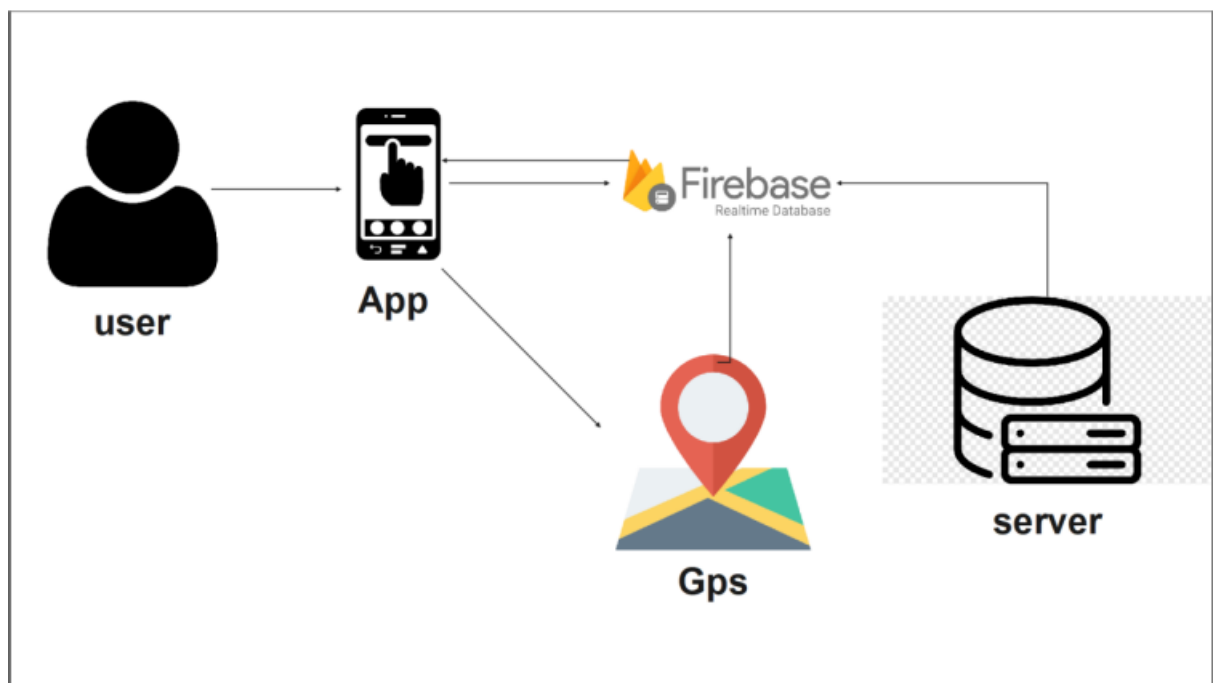


Figure. System Architecture 4.2

4.3 Algorithms

- **Firestore Authentication :**
- To verify E-mail and password.
- **Firestore Fire store :**
- Cloud real time database.
- **One-time passwords (OTP):** It's an algorithm that provide a mechanism for logging on to a network or service using a **unique password that can only be used once**, as the name suggests. One-time passwords are a form of strong authentication, providing much better protection to eBanking, corporate networks, and other systems containing sensitive data. OTP is an automatically generated numeric or alphanumeric string of characters that authenticates a user for a single transaction or login session. It's also more secure than a static password, especially a user-created password, which can be weak and/or reused across multiple accounts. OTPs may replace authentication login information or may be used in addition to it to add another layer of security.

4.4 Methodologies

Agile Methodology

The Agile model is a project management methodology purposely adopted for the development of sophisticated software. The framework allows for iterations, which helps a lot in minimizing mistakes and errors that commonly occur.

The model divides the project into a series of development cycles or short time boxes, which are assigned to each professional on the project team. It is a collaborative approach that allows a response to rapid change. It is flexible enough to accommodate changes in project requirements throughout the mobile app development lifecycle.

Other methodologies fall under the agile umbrella include:

- **Feature Driven Development (FDD)** – a lightweight and incremental model that focuses on features as the name suggests. It features a series of iterations and inspections. This framework demands a high level of design expertise and planning.
- **Lean software development** – it is an integration of the agile methodology and lean manufacturing principles and practices. Aim at optimizing time and reducing waste, cost and effort.

- **Scrum** – focuses on the management aspects of software development in intricate knowledge work, research and advanced technologies with an emphasis on teamwork, iteration and accountability.
- **Crystal Methods** – It is one of the most lightweight agile methodologies. It focuses on team member talent skills, interactions and communication—this model groups projects in terms of system criticality, team size and priorities.
- **Rapid Application Development (RAD)/ Rapid-application building (RAB)**- focuses on timely delivery in a fast-paced environment with the use of prototyping and iterative development.
- **Adaptive Software Development (ASD)** – It is an outgrowth of the RAD that provides continuous adaptation to change in project requirement or market needs.
- **Dynamic Systems Development Method (DSDM)**- it is an iterative and incremental Agile approach based on RAD, but with governance and strict guidelines. It is applicable in four principal phases.
- **Extreme Programming (XP)** – Focuses on software quality and responsiveness with emphasis on the changing needs of the client. It features a high-level collaboration with minimal documentation.
- **Kanban**– a framework that utilizes a lean manufacturing scheduling system to develop software in a long development cycle. The aim is to manage and improve flow systems in mobile app development.

4.5 Implemented Algorithms

4.5.1 E-mail & password Authentication Algorithm

```
lib > screen > Auth > Sign up.dart > _SignUpState > build
253:   },
254:   child: const Text(
255:     "Sign Up",
256:     style: TextStyle(
257:       fontSize: 16,
258:       color: Colors.white,
259:       fontWeight: FontWeight.w700), // TextStyle
260:   ), // Text
261:   onPressed: () async {
262:     if (fkey.currentState!.validate()) {
263:       try {
264:         final credential = await FirebaseAuth.instance
265:           .createUserWithEmailAndPassword(
266:             email: emailAddress,
267:             password: password,);
268:         // Navigator.push(
269:         //   context,
270:         //   MaterialPageRoute(
271:         //     builder: (context) => SignIn()),);
272:         if (credential.user!.emailVerified == false) {
273:           FirebaseAuth.instance.currentUser!
274:             .sendEmailVerification();
275:         }
276:       } on FirebaseAuthException catch (e) {
277:         if (e.code == 'weak-password') {
278:           print('The password provided is too weak.');
```

Figure. Implemented algorithm 4.5.1

4.6 Analysis of the primary results

We have check that's our app implement our functional and non-functional requirements which is:

- Authentication
- Authorization levels
- Historical Data
- Availability
- Reliability
- Scalability
- Performance
- Security
- Usability
- Extensibility

4.7 Time plan







Chapter V

Results and Outcomes



Chapter Five

RESULTS AND OUTCOMES

Now our application has been finished and you can use it to book available spot, view parking detail, write your feedback of application and you can share location area. We have also implemented all of our requirements determination and achieve our goal from this project which is help users.

5.1 Screenshots of our application

5.1.1 Login

Is the first screen of the application, where the user can log in with his first name, last name, email address and password to access his account and have the application authenticate his account and open it without any crashes or issues. That he already has an account if he doesn't have an account, then the user will click on button (signup) if the user entered a wrong email or password it will appear an error under the text and appear what is wrong.

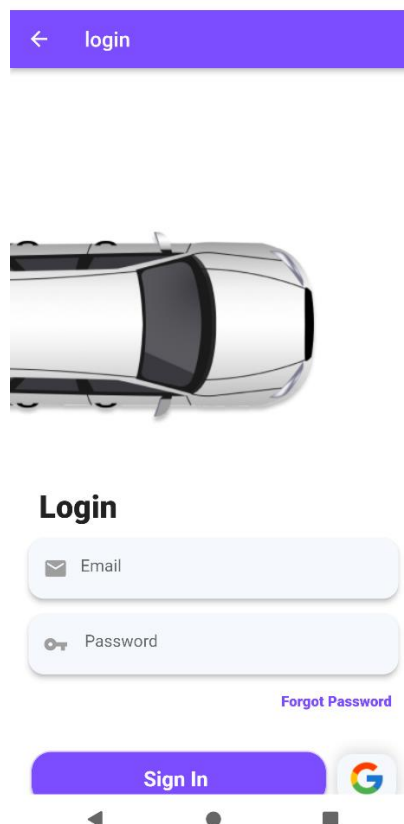


Figure 5.1.1

5.1.2 Sign up

The user enters this page mean he didn't have an account

- The email must contain characters: letters (a-z), numbers, underscores, periods, and dashes.
- An underscore, period, or dash must be followed by one or more letter or number.
- And be sure the email he didn't sign up before to valid email

Password must be strong password contain at least 8 characters or numbers mobile number must start with the country code like 02 for Egypt should be checked if the values meet the following requirements:

1. Phone numbers should be 10-12 digits in length
2. Only numbers are accepted
3. "+" symbol in the beginning of the number is still acceptable

Then he clicks on button sign up if all condition is not validated. Or of the user forget to enter on text field it will appear an error and the condition must be valid If all conditions are true, the user will.

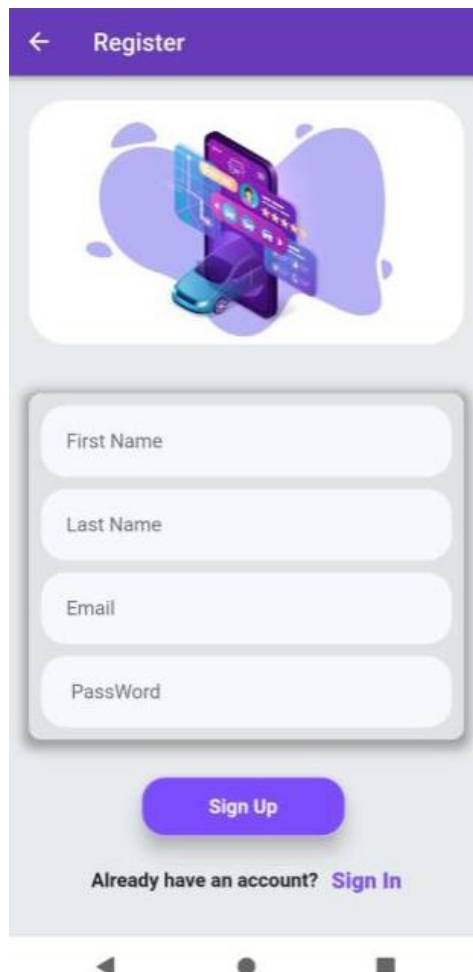


Figure 5.1.2

5.1.3 Home page

The user is entered this page after he has an account. Then he can choose his location area. The user must accept the permission to open his (GPS) to appear his location on map. Then he can enter the destination he wants to go. Also from the home page the user can book his slot or see his payment and can give his feedback about us.

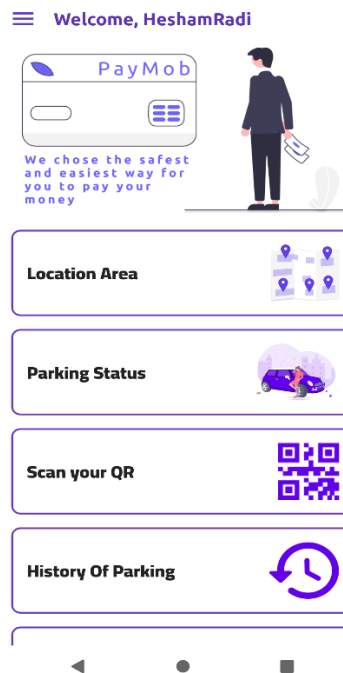


Figure 5.1.3-1

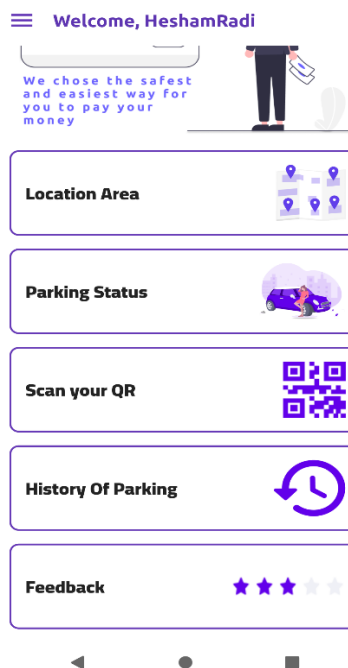


Figure 5.1.3-2

5.1.4 Drawer

In that screen we show the services that given from the application to the user.

- You can saw the parking detail of yours.
- You can saw the location area of your spot after ending it.
- You can share location of spot.
- You can book your spot.
- The image of the user also his name and his email.
- You can give us your feedback.
- Finally you can logout from here.

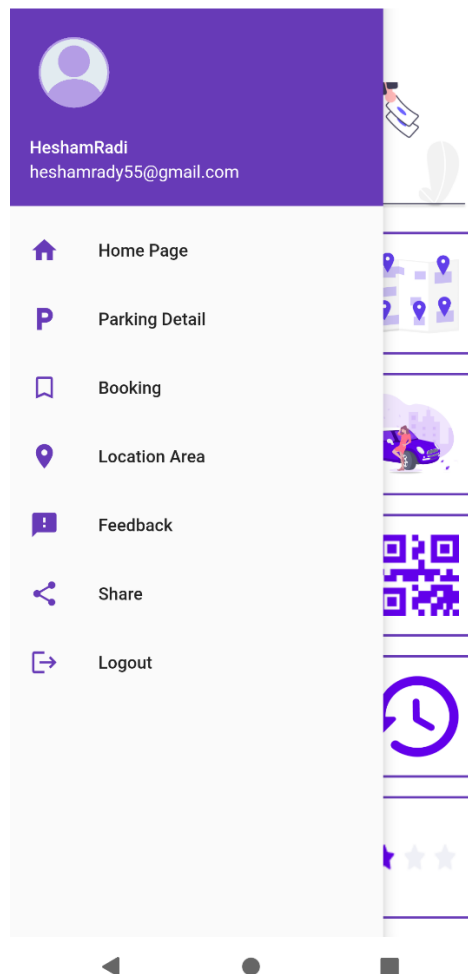


Figure 5.1.4

5.1.5 Parking gates

User choose this page to choose the gate, user has two options.

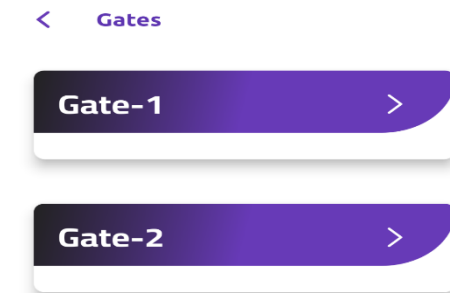


Figure 5.1.5

5.1.6 Parking spots

User choose this page to choose his desired spot, user has four options.

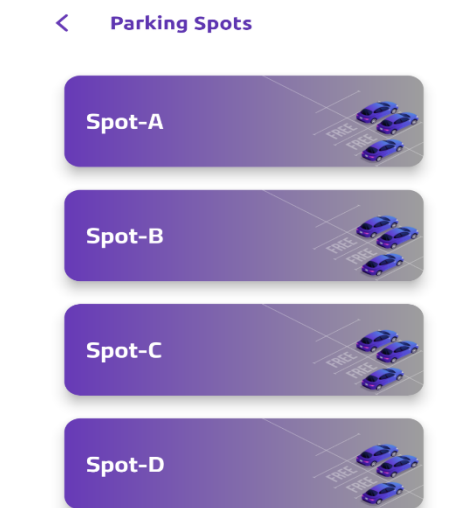


Figure 5.1.6-1



< Spot A

A1	A2
A3	A4
A5	A6
A7	A8
A9	A10

Figure 5.1.6-2

< Spot B

B1	B2
B3	B4
B5	B6
B7	B8
B9	B10

Figure 5.1.6-3



Figure 5.1.6-4

5.1.7 Choose time

User choose when will he arrives to park his car and when will leave.

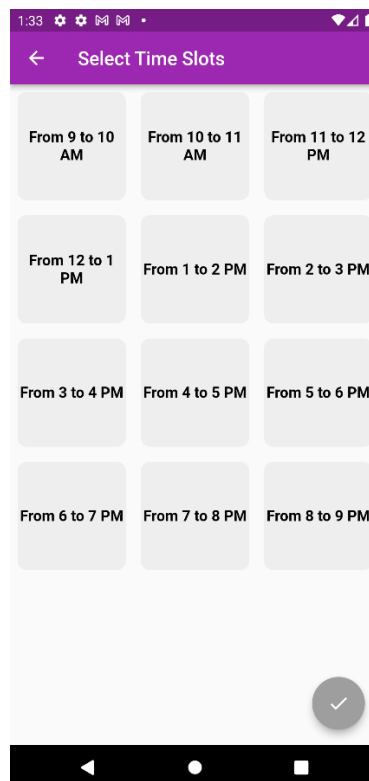


Figure 5.1.7-1

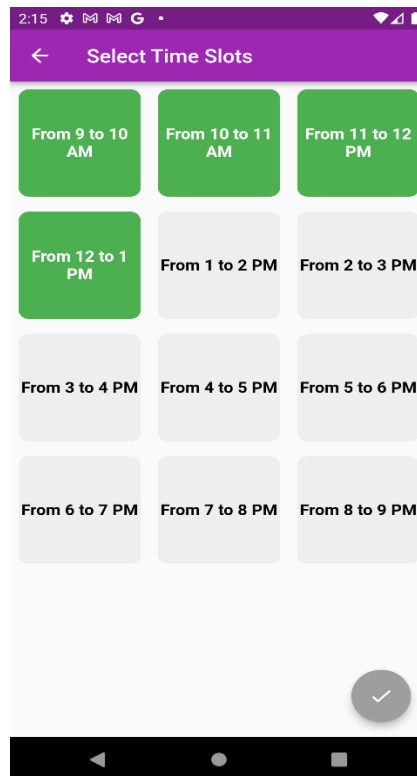


Figure 5.1.7-2

5.1.8 Unavailable time

If this message appears to the user this means user chose time already booked so user had to choose another time.

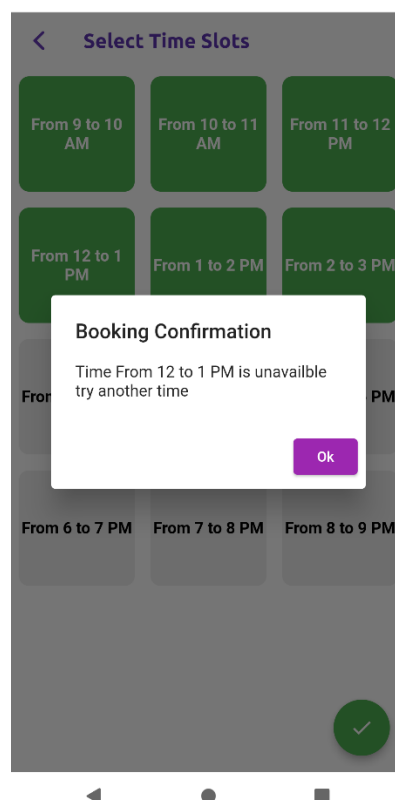


Figure 5.1.8

5.1.9 Booking

In this page you find:

- 1- Number of license plate of the car (Car ID).
- 2- Number he will paid will appear in total.

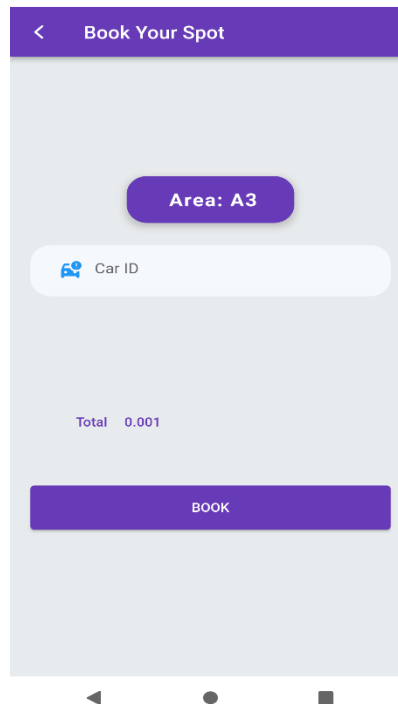


Figure 5.1.9-1

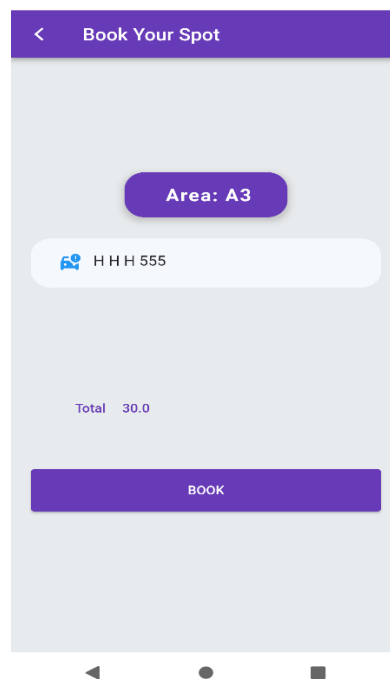


Figure 5.1.9-2

5.1.10 Payment

User choose his desired way to pay has two options: First by Visa and Second by Fawry.

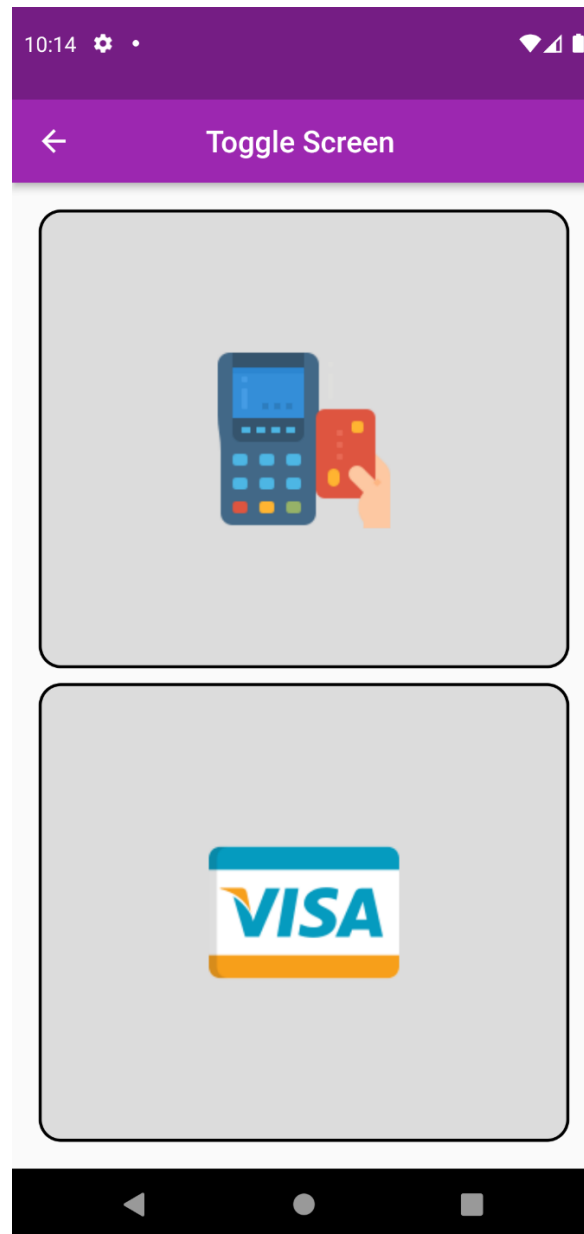


Figure 5.1.10

5.1.11 Fawry

If the user choose Fawry so he should go to any market to pay and gives to the employee the reference code which is appearing in this screen.

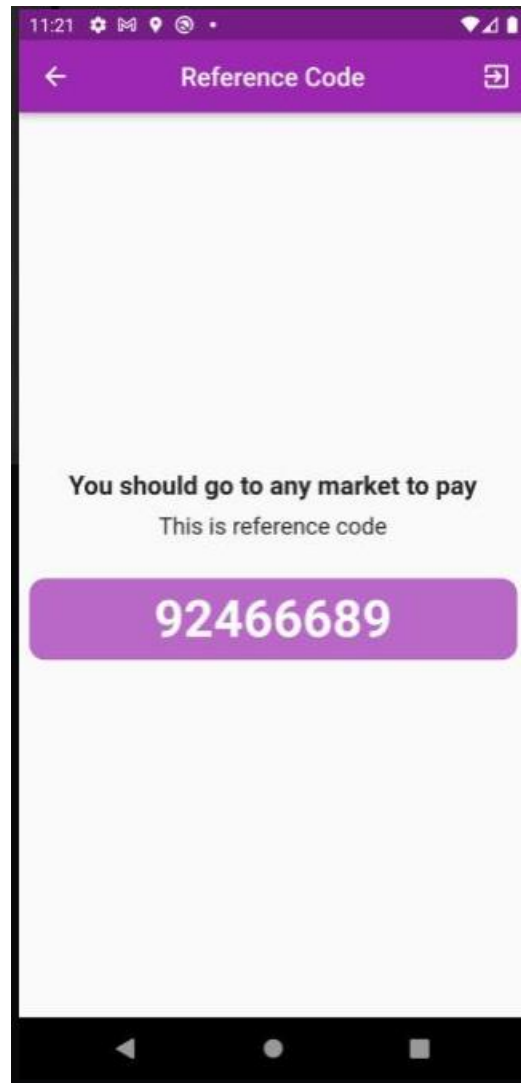


Figure 5.1.11

5.1.12 Visa

If the user choose to pay by visa so he will enter his card number, holder name, expiry month and year.

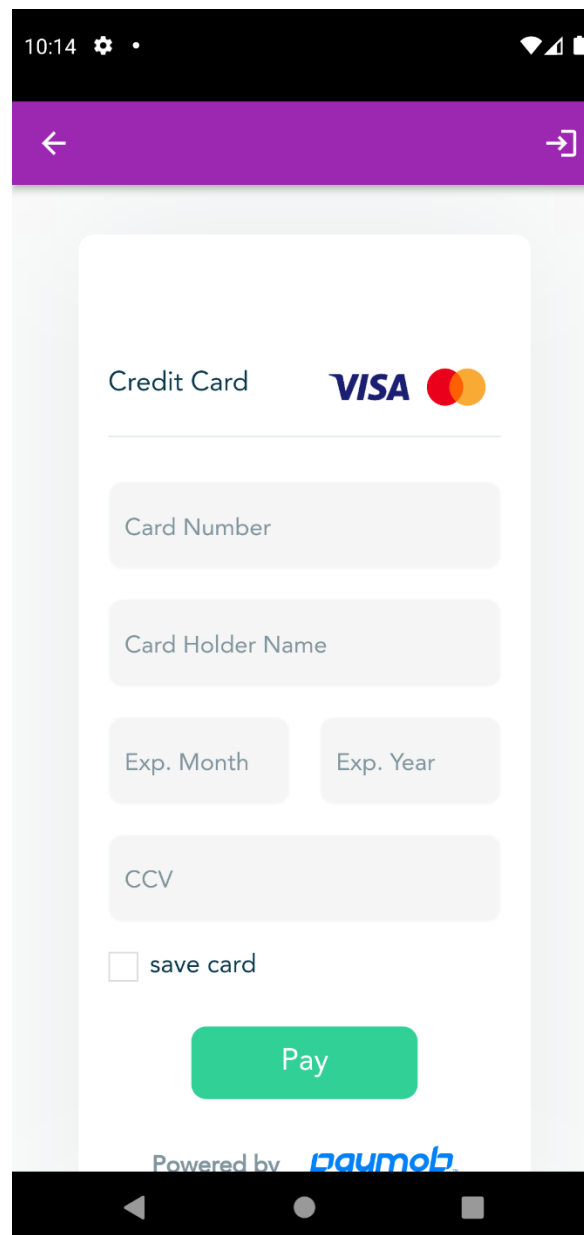


Figure 5.1.12

5.1.13 Payment integration (Visa)

In this screen, user fill his information (first name, last name, e-mail and phone number) to complete the booking and payment process.

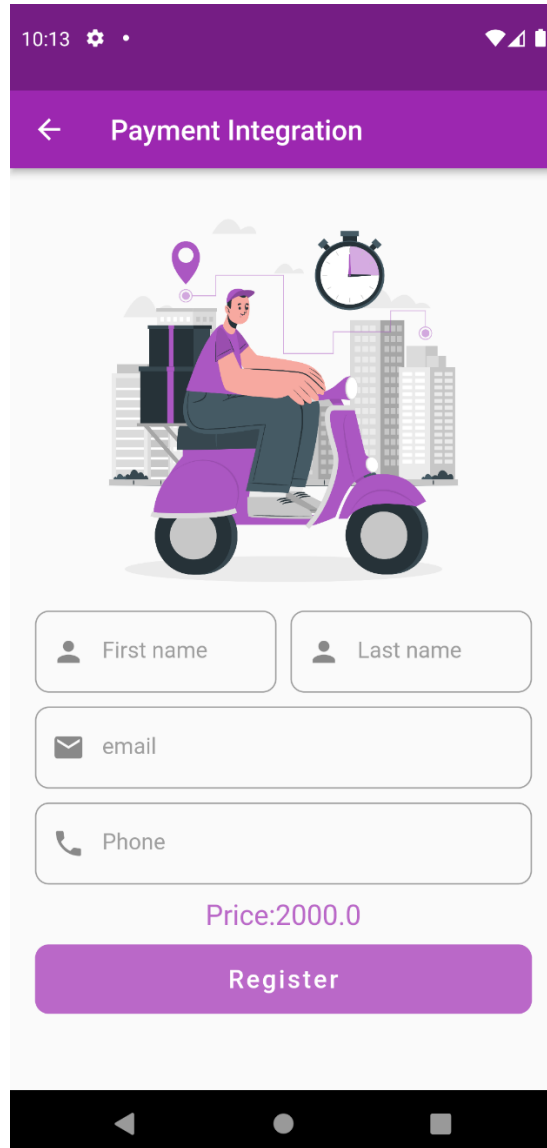


Figure 5.1.13

5.1.14 Verification

After he enter his information a verification will occur by redirected to the bank to check if the user's balance and his information correct or not.

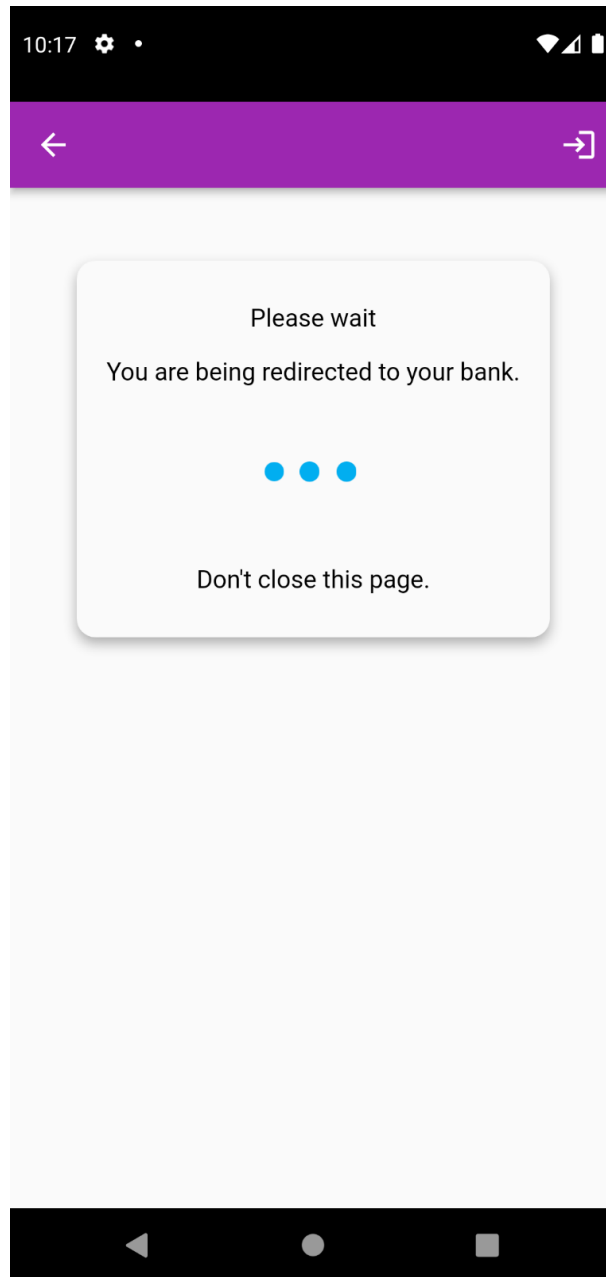


Figure 5.1.14

5.1.15 Approved

If the information is correct so his bookings will be approved.

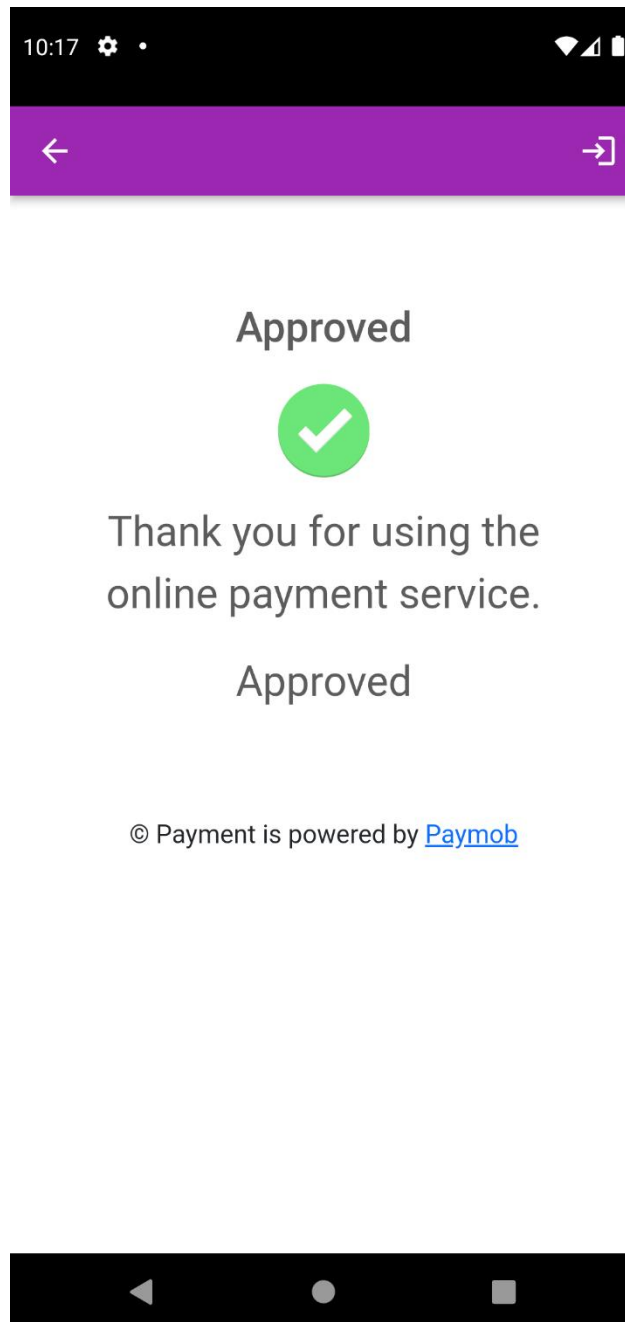


Figure 5.1.15

5.1.16 Declined

If the information is incorrect so his bookings will be declined.

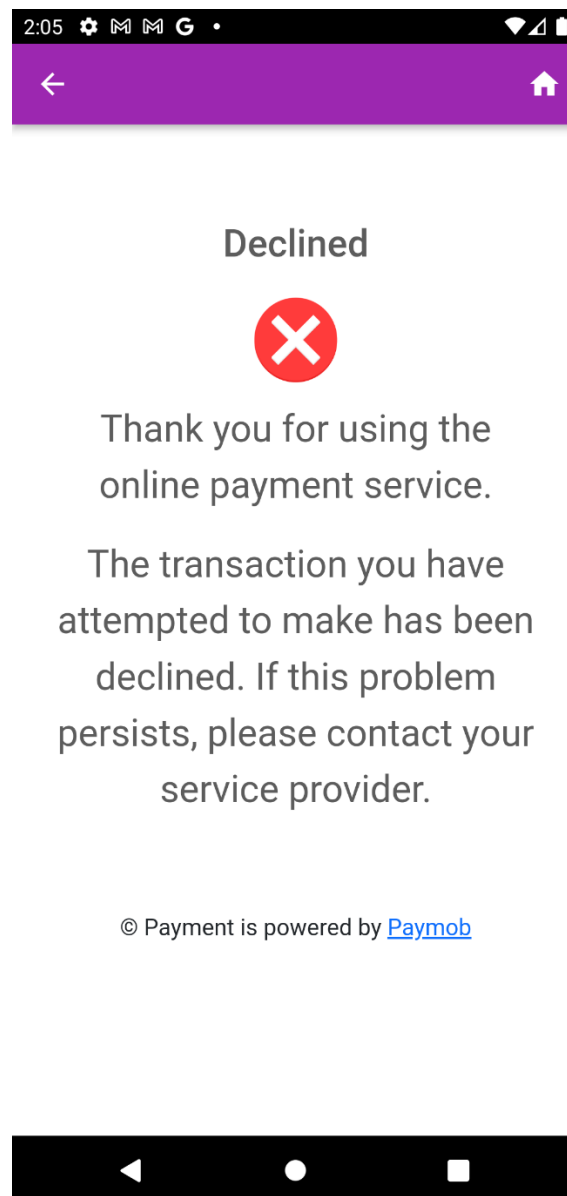


Figure 5.1.16

5.1.17 History

User can check his spot information, price and his timestamp from here.

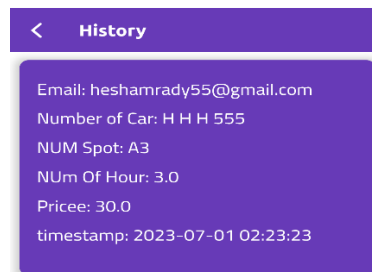


Figure 5.1.17

5.1.18 Feedback

As we care about user's opinion and his opinion matters to us, so we make this page to communicate with user in easy way and improve from us.

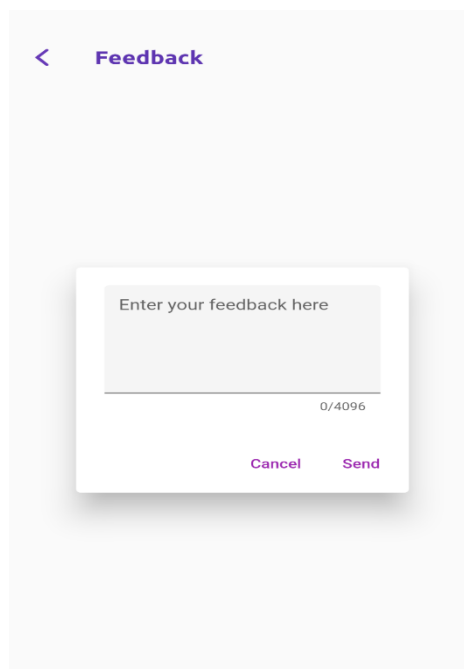


Figure 5.1.18

5.1.19 QR code

This is final screen that appear to the user, when he arrive to the parking it will be scanned. And user won't be able to choose scan QR unless he book his slot and pay first as shown in Fig(4.1.18-2).



Figure 5.1.19-1

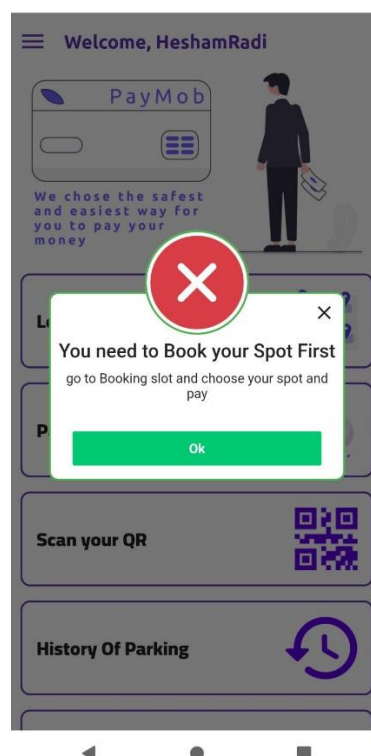


Figure 5.1.19-2

5.1.20 Map

This screen appear when the user choose location area from the home page, it helps him to go from his current location to his destination (parking), the user can put the pickup location by himself from the map or saved location or using the location service on smart phones.

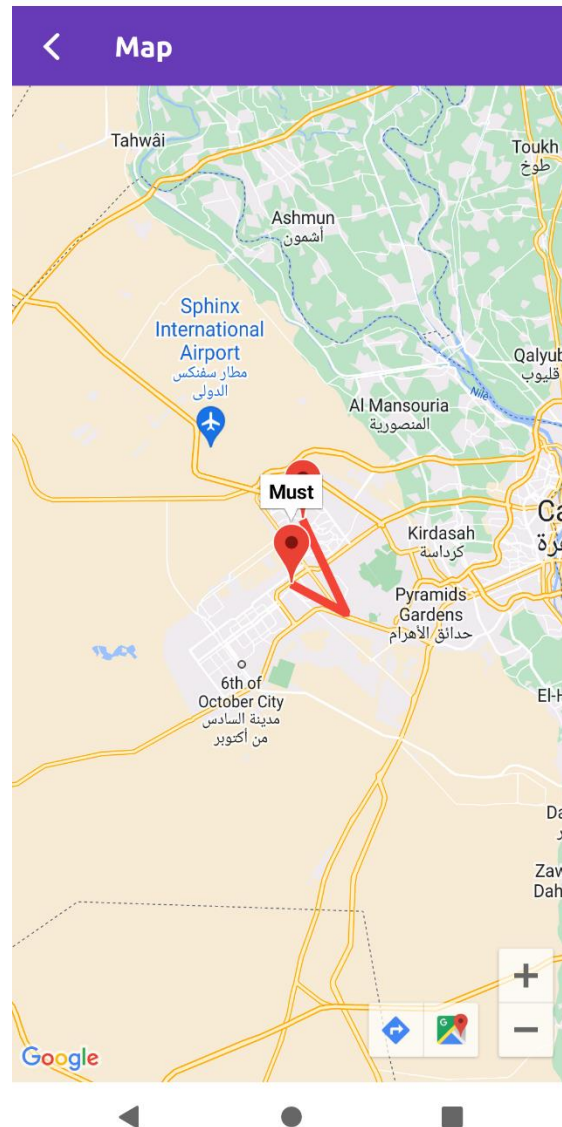


Figure 5.1.20

CONCLUSION

Our goal was to make an application that provides users an easy way of reserving a parking online where users can view various parking spaces and select nearby or specific area of their choice to view whether space is available or not. If the booking space is available, then user can book it for specific time slot. The booked space will be marked and will not be available for anyone else for the specified time. Users can also view previous parking booking details, it will saves user time and effort. At the end, we have come up with a program that made it easier for drivers to find suitable vacancies for them in terms of time and money.

REFERENCES

1. <https://www.digitalcitizen.life/simple-questions-what-are-qr-codes-and-why-are-they-useful/>
2. [ParkMobile Parking App | Find & Pay for Parking - On-street & Reservation](#)
3. [BestParking: Find and Book Parking Anywhere](#)
4. Homepage. 2022. How to Create a Parking Finder App in 6 Steps [Cost + Tips]. [ONLINE] Available at: <https://www.spaceotechnologies.com/blog/parking-spot-finder-appdevelopment>.
5. Welcome to parkopedia! About Parkopedia. (n.d.). (2022, July 15). From <https://www.parkopedia.com/about-us/>.
6. Corporation, S. P. P., & Sp+. (n.d.). Mobile parking app: Find parking near you. Mobile Parking App | Find Parking Near You | Parking.com. (2022, July 15). From <https://parking.com/apps>
7. Drivers - how it works: Yourparkingspace. Your Parking Space. (n.d.). (2022, July 15). From <https://www.yourparkingspace.co.uk/usingyps/drivers>
8. Smart on-street parking solutions: Parking telecom blog. Parking Telecom. (2020, April 1). From <https://parkingtelecom.com/en/smart-onstreet-parking-solutions/>
9. On-street parking solutions - find real-time on-street and off-street parking. Parknav®. (2022, February 1). From <https://parknav.com/onstreet-parking-solutions>
10. L. Wenghong, X. Fanghua, and L.Fasheng, "Design of inner intelligent car parking system," in International Conference on Information Man.
11. Gao, X., Gu, Z., Kayaalp, M., Pendarakis, D., & Wang, H. 2017. ContainerLeaks: Emerging security threats of information leakages in container clouds. In 2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN) (pp. 237- 248). IEEE
12. R.H Ellis et al. Structuring a systems analysis of parking. Highway Research Record (1970).
13. Barata, E., L. Cruz, and J.P. Ferreira. Parking at the UC Campus: Problems and Solutions. Cities, Vol. 28, No. 5, 2011, pp. 406–413.
14. Van der Goot, D. A Model to Describe the Choice of Parking Places. Transportation Research Part A: General, Vol. 16, No. 2, 1982, pp. 109–115.
15. Khattak, A., and J.W. Polak. Effect of Parking Information on Travelers' Knowledge and Behaviour. Transportation, Vol. 20, No. 4, 1993, pp. 373–393.
16. Young, W. A Review of Parking-Lot Design Models. Transport Reviews, Vol. 8, No. 2, 1988, pp. 161–181.
17. Saltzman, R.M. Three Proposals for Improving Short-Term On-Street Parking. Socio-Economic Planning Sciences, Vol. 28, No. 2, 1994, pp. 85–100.

APPENDICES²

Appendix A: Surveys

Appendix B: Code used for development

Appendix C: Algorithm used

Appendix D: sample report

Or any other resources you wish to add, such as figures, details, data, feedback,,.etc.

المستخلص

إن عقبة تحديد موقع بقعة شاغرة في الشوارع المتوفرة أيضًا ليست مهمة سهلة للأشخاص. نظرًا لأن السائقين يضيعون الكثير من وقتهم وجهدهم في القيادة في الشوارع الذين يحاولون العثور على مكان فيها ويحصلون على رضا أكثر من هذه العملية التي يمرون بها مرارًا وتكرارًا خلال حياتهم اليومية. وبالتالي ، فإن هدفنا ودوافعنا هو محاولة حل هذه المشكلة. نريد توفير وقت وجهد الناس الذين يمارسونه كل يوم، لذلك فكرنا في تطبيق يوفر للمستخدم لحجز موقف سيارات كل ساعة أو يوميًا أو شهريًا وسيكون الدفع من خلال البطاقة الائتمانية أو فوري، ويمكن للمستخدم أيضًا عرض المواقع مع عدد المواقع الشاغرة وتحديد المكان الأفضل الذي يرغب في التوجه إليه. يركز مشروعنا بشكل أساسي على حل مشكلة وقوف السيارات سواء في الشوارع أو الأماكن المغلقة ومساعدة المستخدم على الحصول على وجهته المطلوبة بمدة تقديرية وسعر.

الكلمات الرئيسية: تطبيق الهاتف المحمول, نظام وقوف السيارات.



تطبيق للبحث عن أماكن انتظار للسيارات

تقديم تقرير مشروع التخرج
في استيفاء جزئي لمتطلبات منح درجة بكالوريوس العلوم

مقدم من :

هشام راضي السعدي 89431

بلال عبدربه سعيد 89411

سلافه سالم احمد 89409

اسراء عادل امام 89517

تحت إشراف:

الدكتور: ماجد خفاجي

م. أحمد عبدالله

م.م شيرين يوسف

جامعة مصر للعلوم والتكنولوجيا - MUST
كلية الحاسبات وتقنيات الذكاء الاصطناعي - CAIT
قسم علوم الحاسب - CS
يونيو - 2023