

Smart Parking Reservation System  
**ELEG/CPEG 480- Capstone Design Project II**



Project Members

Joanna Haddad (S00043888)  
Fager El-Noueiri (S00042467)  
Ashwaq Bonashi (S00043387)  
Hajar Abdullah (S00046006)

Project Supervisor(s)

Dr. Khalid Sultan

Department of Electrical & Computer Engineering

AMERICAN UNIVERSITY *of* KUWAIT

JUNE 13, 2021

# Smart Parking Reservation System

## Project Members

Joanna Haddad (S00043888)

Fager El-Noueiri (S00042467)

Ashwaq Bonashi (S00043387)

Hajar Abdullah (S00046006)

The capstone project report is being submitted in partial fulfillment of the requirements for the degree of  
**Bachelor of Engineering in Electrical/Computer Engineering**

## Project Supervisor(s):

Dr. Khalid Sultan

Supervisor's Signature:



Department of Electrical & Computer Engineering

AMERICAN UNIVERSITY *of* KUWAIT

JUNE 13, 2021

## **Declaration**

We certify that this project work titled “*Smart Parking Reservation System*” is our own work. The work has not been presented elsewhere for assessment. The material that has been used from other sources has been properly acknowledged / referred.

### Project Members

Joanna Haddad (S00043888)

Fager El-Noueiri (S00042467)

Ashwaq Bonashi (S00043387)

Hajar Abdullah (S00046006)

## **Plagiarism Certificate (Turnitin Report)**

This thesis has been checked for Plagiarism. Turnitin report endorsed by the Supervisor(s) is attached.

### Project Members

Joanna Haddad (S00043888)

Fager El-Noueiri (S00042467)

Ashwaq Bonashi (S00043387)

Hajar Abdullah (S00046006)

Signature of Supervisor(s)



## **Copyright Statement**

- Copyright in text of this project report rests with the student authors. Copies (by any process) either in full, or of extracts, may be made only in accordance with instructions given by the authors and lodged in the Library of AUK. Details may be obtained by the Librarian. This page must form part of any such copies made. Further copies (by any process) may not be made without the permission (in writing) of the authors.
- The ownership of any intellectual property rights which may be described in this project report is vested in AUK's Department of Electrical & Computer Engineering, subject to any prior agreement to the contrary, and may not be made available for use by third parties without the written permission of the AUK's Department of Electrical & Computer Engineering, which will prescribe the terms and conditions of any such agreement.
- Further information on the conditions under which disclosures and exploitation may take place is available from the Library of AUK, Kuwait.

## **Acknowledgements**

The team of the Smart Parking Reservation System would like to express its deepest gratitude to Dr. Khalid Sultan, AUK professors, fellow colleagues, and everyone that assisted us to accomplish our goal. We would not have landed on the final outcome of this project without their consistent encouragement.

Joanna Haddad (S00043888)

Fager El-Noueiri (S00042467)

Ashwaq Bonashi (S00043387)

Hajar Abdullah (S00046006)

“In honor of everyone who has committed their lives to the development of technology for the benefit of humankind. This will be our first move as engineers in creating a difference in the world.”

## **Abstract**

Due to the significant increase in the usage of vehicles, finding a parking spot has become a critical issue to car drivers. The parking spot problem can be daily faced in public places or even at workplaces. In some countries, such as Kuwait, it is usually hard to know the availability of a spot in a parking lot. For instance, almost all parking lots in Kuwait, including those at the hospitals and shopping malls, run in traditional way in which drivers enter the parking lot and start looking for an empty spot. Besides, some people tend to park near the sidewalk which can be disturbing to other drivers and results in parking limitation. The situation gets worst during rush hours. The aim of this project is to design a smart parking reservation system based on Internet of Things (IoT) technology in order to mitigate this problem. The parking lot will be equipped by smart gates that will operate using solar panel. The solar panel will provide clean energy to the whole floor. The advantage of relaying on renewable energy is to reduce the power outage that can stand as a barrier in many countries. Drivers will use the application to check the availability of a parking spot, enter the parking lot, open the gate of their reserved spot, and pay. Moreover, only those who reserved a parking spot can enter the main parking lot gate, otherwise it will prevent other cars from entering. The vision behind constructing a modern parking lot is to refine the infrastructure, reduce parking congestion, and save time. As an initial step, the project will be conducted on one floor. However, high standard civilization can be acquired by enlarging the floor scale to smart buildings and cities.

**Key Words:** *Parking system, IoT, parking lot, congestion, traffic.*

# Table of Contents

ELEG/CPEG 480- Capstone Design Project II .....	i
<b>Declaration .....</b>	<b>i</b>
<b>Plagiarism Certificate (Turnitin Report).....</b>	<b>ii</b>
<b>Copyright Statement .....</b>	<b>iii</b>
<b>Acknowledgements .....</b>	<b>iv</b>
<b>Abstract .....</b>	<b>vi</b>
<b>Table of Contents.....</b>	<b>vii</b>
<b>List of Figures .....</b>	<b>x</b>
<b>List of Tables.....</b>	<b>xii</b>
<b>CHAPTER 1: INTRODUCTION.....</b>	<b>2</b>
1.1 Introduction.....	2
1.2 Background.....	2
1.3 Problem Statement .....	3
1.4 Aims and Objectives of the Project.....	4
1.4.1 Research Questions:.....	5
1.5 Significance, Scope and Definitions .....	5
1.6 SWOT Analysis .....	7
1.7 Report Outline.....	8
1.7.1 Chapter One.....	8
1.7.2 Chapter Two .....	8
1.7.3 Chapter Three .....	8
1.7.4 Chapter Four.....	8
1.7.5 Chapter Five.....	8
1.7.6 Chapter Six .....	9
<b>CHAPTER 2: LITERATURE REVIEW .....</b>	<b>11</b>
2.1 Introduction .....	11
2.2 Related Works.....	11
2.2.1 Car Parking System Using IR Sensors [4].....	11
2.2.2 Car Parking System Based on Radio Frequency Identification [5].....	12
2.2.3 Smart Parking System Using Ultrasonic Detectors [6].....	13
2.2.4 A Systematic Review of Machine-vision-based Smart Parking Systems [7].....	14
2.2.5 QR Code based Smart Parking System [8] .....	15
2.2.6 Wireless based Smart Parking System using Zigbee [9].....	16
2.2.7 Smart Parking System Based on Bluetooth Low Energy Beacons with Particle Filtering [10].....	17
2.2.8 Applying smart parking system with internet of things (IoT) design [11].....	18
2.2.9 Smart Parking System Using IoT [12].....	19

2.2.10	Internet of Things (IoT) based Smart Parking Reservation System using Raspberry-pi [13] .....	20
2.2.11	Intelligent Parking Management System Based on Image Processing [14] .....	21
2.2.12	E-Parking Software (Getpass) [15].....	22
2.2.13	Mawqif - Smart Parking Assistant [16].....	23
2.2.14	Parkpiú Installed Car Parking System at The Palms Hotel in Kuwait [17] .....	24
<b>Parkpiú Installed Car Parking System at The Palms Hotel in Kuwait [17] .....</b>		<b>28</b>
2.3	Summary and Implications .....	28
2.4	Conclusion .....	29
<b>CHAPTER 3: METHODOLOGY, DESIGN AND ANALYSIS.....</b>		<b>31</b>
<b>3.1</b>	<b>Introduction .....</b>	<b>31</b>
3.2	Methodology .....	31
3.2.1	Requirements: .....	32
3.2.2	Software and System Design: .....	33
3.2.3	Implementation and Unit Testing: .....	35
3.2.4	Integration and System Testing: .....	35
3.2.5	Operations and Maintenance: .....	36
3.3	Research Design.....	36
3.3.1	Design Alternatives.....	37
3.4	Hardware Components.....	38
3.4.1	Microcontroller Board, EBot 8 Pro [20] .....	38
3.4.2	EBot IoT Board [23].....	40
3.4.3	Micro Servo [24].....	41
3.4.4	EBot RGB LED [25].....	42
3.4.5	EBot IR Sensor [26].....	43
3.4.6	Solar Panel .....	44
3.4.7	The Battery and Charge Controller.....	45
3.5	Software Programs .....	46
3.5.1	Arduino IDE [27].....	46
3.5.2	EBot Blockly [28].....	46
3.5.3	Visual Studio Code [29] .....	47
3.6	Analysis.....	47
3.6.1	Block Diagram Analysis:.....	47
3.6.2	System Architecture:.....	49
3.6.3	Cost Analysis .....	50
3.700	.....	50
3.7	Ethics and Limitations .....	50
<b>CHAPTER 4: IMPLEMENTATION .....</b>		<b>53</b>
4.1	Introduction .....	53
4.2	Hardware Implementation.....	53

4.2.1	An Overview of How the Prototype Works.....	54
4.2.2	Electronic Circuit.....	54
4.2.3	Power Circuit.....	59
•	Hardware Coding: .....	61
4.2.4	Programming the Two Microcontrollers.....	62
4.2.5	Final Implementation of the Prototype .....	65
4.3	Software Implementation .....	66
4.4	IEEE Standards .....	79
4.5	Conclusion .....	79
<b>CHAPTER 5: EVALUATION</b>	<b>.....</b>	<b>81</b>
<b>5.1 Introduction</b>	<b>.....</b>	<b>81</b>
5.2	Project Impact .....	81
5.2.2	Environmental and Geographical Impact .....	81
5.2.3	Business Impact.....	82
5.2.4	Marketing Impact.....	82
5.2.5	Economic Impact.....	83
5.3	Comparison.....	83
5.4	Conclusion .....	83
<b>CHAPTER 6: CONCLUSION AND FUTURE WORK</b>	<b>.....</b>	<b>85</b>
<b>6.1 Introduction</b>	<b>.....</b>	<b>85</b>
6.2	Conclusion .....	85
6.3	Future Work.....	86
<b>REFERENCES</b>	<b>.....</b>	<b>88</b>
<b>APPENDIX A: Hardware Programming</b>	<b>.....</b>	<b>91</b>
<b>Part 1</b>	<b>.....</b>	<b>91</b>
<b>Part 2</b>	<b>.....</b>	<b>98</b>
<b>Appendix B: Segment of the Software Programming</b>	<b>.....</b>	<b>102</b>
<b>Parking List (Dart)</b>	<b>.....</b>	<b>102</b>
<b>Parking Card (Dart)</b>	<b>.....</b>	<b>103</b>
<b>Confirm Booking Dialog (Dart)</b>	<b>.....</b>	<b>106</b>
<b>Wallet Dialog (Dart)</b>	<b>.....</b>	<b>112</b>

## List of Figures

Figure 1 <i>Comparison Between Traffic Congestion in year 2019 VS 2020 [1]</i> .....	3
Figure 2 <i>A Parking System Based on IR Sensors [4]</i> .....	12
Figure 3 <i>RFID Based Parking System [5]</i> .....	13
Figure 4 <i>Ultrasonic Detectors on Each Parking Spot [6]</i> .....	14
Figure 5 <i>QR Code Based Smart Parking System [8]</i> .....	15
Figure 6 <i>Number of Vehicles and Sections [8]</i> .....	16
Figure 7 <i>Wireless Based Smart Parking System Using Zigbee [9]</i> .....	16
Figure 8 <i>Wireless Based Smart Parking System Using Zigbee [9]</i> .....	17
Figure 9 <i>The System Chart [10]</i> .....	18
Figure 10 <i>The Procedure [11]</i> .....	19
Figure 11 <i>System Flow Chart [12]</i> .....	19
Figure 12 <i>Software [12]</i> .....	20
Figure 13 <i>System Flow Chart [13]</i> .....	21
Figure 14 <i>Empty Parking [14]</i> .....	22
Figure 15 <i>Occupied Parking [14]</i> .....	22
Figure 16 <i>Getpass Procedure[15]</i> .....	23
Figure 17 <i>Mawqif Procedure [16]</i> .....	23
Figure 18 <i>Parkpiú In Kuwait [17]</i> .....	24
Figure 19 <i>Comparison Between Key Features of Existing Solutions and Proposed Work</i> .....	29
Figure 20 <i>Waterfall Methodology</i> .....	31
Figure 21 <i>USER Requirement Specifications (URS)</i> .....	32
Figure 22 <i>Software of the System</i> .....	34
Figure 23 <i>Use Case Diagram for the Smart Parking Reservation System</i> .....	35
Figure 24 <i>EBot 8 Pro [20]</i> .....	38
Figure 25 <i>EBot IoT Board [23]</i> .....	40
Figure 26 <i>Micro Servo 9g [24]</i> .....	41
Figure 27 <i>RGB LED [25]</i> .....	42
Figure 28 <i>IR Sensor [26]</i> .....	43
Figure 29 <i>Solar Panel 18V</i> .....	44
Figure 30 <i>12V Battery</i> .....	45
Figure 31 <i>Charge Controller</i> .....	45
Figure 32 <i>Arduino IDE [27]</i> .....	46
Figure 33 <i>EBot Blockly [28]</i> .....	46
Figure 34 <i>Visual Studio Code [29]</i> .....	47
Figure 35 <i>Block Diagram for the System</i> .....	47
Figure 36 <i>System Architecture</i> .....	49
Figure 37 <i>Hardware Implementation Stages</i> .....	53
Figure 38 <i>Testing IoT Board Using Blockly</i> .....	55
Figure 39 <i>Connecting IoT Board to the First Microcontroller</i> .....	55
Figure 40 <i>EBot IoT Board Connection</i> .....	56
Figure 41 <i>Microcontrollers' Connections</i> .....	57
Figure 42 <i>The Connection of the IoT Board with the Two Microcontrollers</i> .....	58
Figure 43 <i>Solar Power System Diagram</i> .....	60
Figure 44 <i>Back of the Solar Panel</i> .....	60
Figure 45 <i>12V Battery Connected to the Charge Controller</i> .....	61
Figure 46 <i>The Four Inputs of the Charge Controller</i> .....	61
Figure 47 <i>The Main Loop for Status and Gates Control</i> .....	64
Figure 48 <i>Sample of EBOT 8 pro C++ Code</i> .....	65
Figure 49 <i>Initial Design</i> .....	65
Figure 50 <i>Smart Parking Reservation System Prototype</i> .....	66
Figure 51 <i>Admin Panel Sign in</i> .....	68
Figure 52 <i>Data of the Cloud</i> .....	69
Figure 53 <i>Parking Area Page</i> .....	69

Figure 54 <i>Add New Slot Feature</i> .....	70
Figure 55 <i>Add New Parking Area Feature</i> .....	70
Figure 56 <i>Admin's Profile Edit</i> .....	70
Figure 57 <i>Admin's Password Update</i> .....	71
Figure 58 <i>Application Files</i> .....	71
Figure 59 <i>Validation Process</i> .....	72
Figure 60 <i>Low Wallet Balance</i> .....	73
Figure 61 <i>Proceed Process</i> .....	73
Figure 62 <i>Submit Form</i> .....	73
Figure 63 <i>Main Functionality of the System</i> .....	74
Figure 64 <i>Booking from Area</i> .....	74
Figure 65 <i>Checks Up the Slots</i> .....	75
Figure 66 <i>Builds Slots</i> .....	75
Figure 67 <i>Check Circle Symbol for Selected Spot</i> .....	75
Figure 68 <i>Yellow Color for Selected Spot</i> .....	75
Figure 69 <i>Collects Details for Confirm Booking</i> .....	76
Figure 70 <i>Payment and Valid Balance</i> .....	76
Figure 71 <i>Sign In and Sign Up of Application</i> .....	77
Figure 72 <i>The Procedure of the Application</i> .....	78
Figure 73 <i>The Exit and Payment of the Application</i> .....	79

## List of Tables

Table 1 <i>The Four Aspects of Research Questions</i> .....	5
Table 2 <i>SWOT Analysis of the Project</i> .....	7
Table 3 <i>Summary of the key Features, Advantages, and Weaknesses of the Existing solutions</i> .....	25
Table 4 <i>Functional and Non-Functional Requirements</i> .....	33
Table 5 <i>Options Available for Microcontrollers</i> .....	37
Table 6 <i>Options Available for Sensors</i> .....	38
Table 7 <i>Features of the chosen Microcontroller [20]</i> .....	39
Table 8 <i>Features of the IoT Board [23]</i> .....	40
Table 9 <i>Features of the Micro Servo [24]</i> .....	41
Table 10 <i>Features of RGB LED [25]</i> .....	42
Table 11 <i>Features of IR Sensor [26]</i> .....	43
Table 12 <i>Features of the Solar Panel</i> .....	44
Table 13 <i>The Budget of the Project</i> .....	50
Table 14 <i>The Connection Between Each Component with Microcontrollers</i> .....	58
Table 15 <i>Systems Comparison</i> .....	83



# CHAPTER ONE

## INTRODUCTION

# CHAPTER 1: INTRODUCTION

## 1.1 Introduction

In chapter one, the project is introduced along with stating the problem statement, aims and objectives, and the idea behind the system. The main purpose of the project is to find a solution for drivers who spend time looking for a parking space. The project is called “Smart Parking Reservation System” which is designed for drivers that struggle to find an empty space in large parking lots. Moreover, the system will be based on solar energy.

## 1.2 Background

In Kuwait, one of the major problems that are unresolved is the inability to find a parking spot easily. This problem occurred due to the increase in traffic density. According to the *Dutch Company for Navigation Systems, “Tom Tom”*, the congestion level in Kuwait was 25 percent in year 2019 [1]. However, due to COVID-19 pandemic, these statistics were much lower because of the curfew restrictions. Figure 1 shows a comparison between the traffic congestion in year 2019 and 2020. It is hard to find a parking spot especially in Kuwaiti malls such as the Avenues and AlMubarikya. In Avenues mall parking is equipped with LED lights and digital screens to indicate the spot availability. In addition, Kuwait has many crowded places that are known for the increased vehicle density in rush hours. For example, in AlMubarkiya most drivers park near the sidewalks which can lead to congestions and accidents. Thus, it is difficult to find a parking lot using such technologies especially in developed countries. The parking availability is one of the main concerns for drivers in a daily basis. Another point to consider is that most parking lots are operated by conventional sources rather than non-conventional. It is rarely to find constructions operated on renewable energy in Kuwait. Also, most of its constructions depends on the finite resources that are negatively affecting the environment and economy. However, Kuwait is a sunny and hot country in summer seasons which has a potential to depend on solar energy. The degree in Kuwait can reach 50oC that can be taken for granted for the use of solar panels. Thus, the team thought about building a parking reservation system based on solar energy.

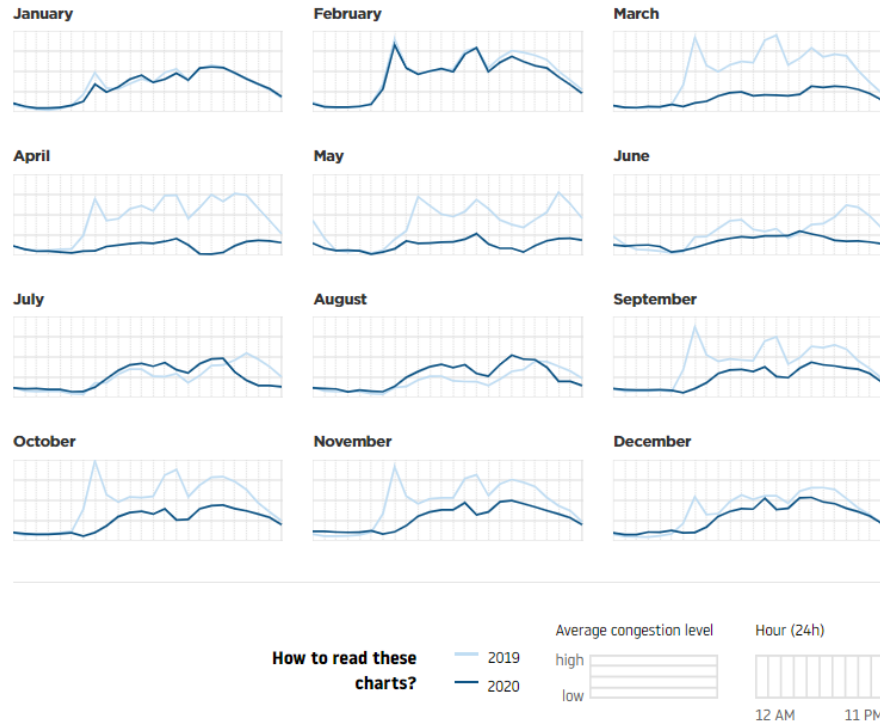


Figure 1 Comparison Between Traffic Congestion in year 2019 VS 2020 [1]

### 1.3 Problem Statement

Finding parking spots is not an easy task with the increase of population that requires many facilities. In the current market the available parking systems are mostly implemented with the use of LED lights, sensors, and HMI (Human Machine Interface). Even though there exist applications that provide parking reservations with online payment wallet. Some malls in Kuwait, such as Avenues, require advanced parking reservation systems to save time, lessen congestions and solve issues behind parking lots. According to *Arab Times*, about 40,000 car drivers in Kuwait City take the risk of the withdrawal of their number plates every day when they park their vehicles due to the lack of sufficient parking areas [2]. In many cases, some people tend to park illegally inside the parking lots which causes inconvenience to other drivers. On the other hand, many parking lots in Kuwait display the number of empty spots by sensors and LED lights. However, they cause congestions because drivers have to keep searching to locate the empty spots displayed. According to the *National Institute of Health*, "congestion can change driving patterns, resulting in an increased number of speedups, slowdowns, stops and starts, which increase emissions" [3]. In addition, many issues arise on the use and cost of conventional energy in parking lots of malls, hospitals, and airports. The energy consumption has been an issue for the past few years without

acknowledging its extreme effects on the pollution and financial crisis. Therefore, actions should be taken to cut off the load on energy consumption. One of the main reasons that led to this project is to enhance the current parking systems to satisfy the market demand using integration between software and hardware level.

## **1.4 Aims and Objectives of the Project**

The Smart Parking Reservation System brought to the market in order to refine driver's experience inside parking lots. The first aim is to decrease the number of vehicles waiting in queues for their lucky moment when they find a spot to park. Allowing the users to reserve a spot beside their destined store in a mall will be satisfying and time saving. Social responsibility defines the second aim of this system as the pollution and financial crisis are arising. The use of renewable clean energy will form a foundation in smart parking reservation systems. The solar panels can reduce electricity bills, pollution, and the dependency on finite resources. Because of that, the parking model is powered by the solar energy. Overall, the smart parking reservation system will enhance the availability of parking spots, advanced booking, and online payment methods. Also, the usage of Internet of Things provides a user-friendly system to reserve the preferred spots on a daily basis. The application will guarantee for the users the following:

- 1.** Solving the parking problems in crowded parking lots.
- 2.** Saving the environment by using clean energy.
- 3.** Helping people move forward towards smart cities.
- 4.** Facilitating reservation process via a user-friendly application.
- 5.** Allowing people to book their preferable spots remotely.
- 6.** Indicating the status of every parking spot whether it is booked, available or taken through an IoT-based system.
- 7.** Ensuring that parking lots are congestion free.

### 1.4.1 Research Questions:

Table 1 *The Four Aspects of Research Questions*

Trials Review	Systematic Review
1. Are existing reservation systems effective? 2. Are existing parking counting systems helpful?	1. Are solar panels installations worthwhile; will solar panels satisfy the needs of the project? 2. Which technologies can be reused from the existing solutions? 3. Is there any market demand regarding reservation systems?
Observational Review	Qualitative Review
1. Do existing parking reservation systems lessen the congestion in Kuwait? 2. Would the Smart Parking Reservation System be suitable to be implemented in different locations and types of parking lots (indoor – outdoor)? 3. Does the use of solar system panels reduce electricity bills? 4. Will the use of renewable energy in The Smart Parking Reservation System reduce pollution in Kuwait?	1. What are the demographic characteristics of the users that would use reservation systems including age, education, and ethnicity? 2. How can The Smart Parking Reservation System consider privacy including Kuwait's cultural beliefs? 3. What if users do not prefer to register into the reservation application? 4. How can the parking reservation application be user-friendly? 5. What are the regulations of Kuwait's Ministry of Electricity and Water for implementing such projects?

## 1.5 Significance, Scope and Definitions

Many developed countries have a high reliance on driving cars; therefore, they encounter difficulty to offer parking spots for a large number of drivers. In addition, developed countries are also well known to have huge constructions such as malls, hospitals, airports, and universities. Such constructions have high rate of visitors that need to find a parking lot as fast and affordable as possible. An example of these developed countries is Kuwait's parking lots. There exist parking reservation systems in Kuwait; therefore, it is important to implement the market research in order to acquire insights about the technologies used in reservation systems. Thus, research helps to analyze the available solutions in order to modify and fill in the gaps in the literature as the best possible way. Research is conducted to figure out the independent and dependent variables. High vehicles users affect the occupation of parking lots. The independent variable is the number of

vehicle users affecting the dependent variable representing the parking lot availability. Vehicles are the most used transportation mean in Kuwait, which require a reservation system to serve the needs of the drivers. Geographically, Kuwait has sandy areas that are used as parking spaces causing congestions. Also, Kuwait has an opportunity to rely on thermal energy coming from the sun radiation. However, Kuwait consumes energy attained from the fuel that will run out. Nowadays, people from all age groups use smart phones for many purposes. Thus, technology intervention will provide an opportunity to serve a parking reservation system. This time span will help the research to be conducted and implemented. In addition, population traits play a role in causing traffic congestion as many drivers tend to violate the law by parking illegally. For example, some people tend to park by and on the sidewalks. Another point to consider is that the solution must follow the country's cultural beliefs as some technologies may be undesired such as face identification. As a result, the significance of the Smart Parking Reservation System is identified in terms of meeting the market demand. The system will be controlled through a mobile application in which drivers can choose the desired parking space in their destination. Thus, the arising problems are an inspiration to generate solutions using a Parking Reservation System. In addition, the frequent definitions used in this project include, the Internet of Things (Iot), solar energy and parking congestions.

## 1.6 SWOT Analysis

Table 2 *SWOT Analysis of the Project*

Strengths	Weaknesses
<ul style="list-style-type: none"> <li>• Utilizes latest software technologies to coop with the latest technology trends (internet of things)</li> <li>• Solves an everyday issue which is finding Parking</li> <li>• can be installed on existing parking lots to enhance them as both branded and white label product.</li> <li>• Increase parking area capacity by organizing the parking lot.</li> <li>• Lowers waiting time in Parking lots.</li> <li>• Automates most of the process where it helps lower Parking lot operating cost.</li> </ul>	<ul style="list-style-type: none"> <li>• Relies on electrical power where it might not be available sometimes.</li> <li>• Users must have smartphones to use the service.</li> <li>• Users must be connected to the internet to use the service.</li> <li>• Users need to be trained therefore human interaction is a necessity for the transition period leading to high cost of deployment.</li> </ul>
Opportunities	Threats
<ul style="list-style-type: none"> <li>• There is a huge market for parking lots in congested areas.</li> <li>• Can be supported by traffic department to minimize road clogging.</li> <li>• If deployed on large scale revenue can be high and used to subsidize road infrastructure projects within the country.</li> <li>• Can partner with a bank to store wallet treasury to create another revenue stream.</li> <li>• Can be integrated as a service in other apps.</li> <li>• Can add more features such as electric cars chargers.</li> </ul>	<ul style="list-style-type: none"> <li>• Unavailability or weak coverage of internet.</li> <li>• In some countries the cost cannot be covered by the revenue in a short period.</li> <li>• Wallet licensing requirements.</li> <li>• Management of wallet treasury liability.</li> <li>• Competition can operate without solar energy in countries with cheap energy.</li> </ul>

## **1.7 Report Outline**

### **1.7.1 Chapter One**

In this chapter, the idea of the project was introduced through discussing the background, problem statement, and the aims and objectives of the project. Besides, the significance of the project and the SWOT analysis which analyzes the strengths, weaknesses, opportunities, and possible threats related to the Smart Parking Reservation System.

### **1.7.2 Chapter Two**

Chapter two reviews related works that have been put forward to mitigate the parking issue. Then, it compares the Smart Parking Reservation System to the related works to have a room for improvements.

### **1.7.3 Chapter Three**

Chapter three explains the methodology behind the project and the design and analysis needed to implement the smart parking system. In addition, it evaluates the design alternatives that are based on the components available on market. Moreover, the hardware components and software programs used in the project are mentioned.

### **1.7.4 Chapter Four**

In chapter four, the hardware and software parts are discussed in more details. The implementation of the hardware components is explained as all the connections and the testing steps are clarified. Also, it comprises the software programs used to operate the components and build the application.

### **1.7.5 Chapter Five**

In chapter five, the smart parking system will be evaluated from the secondary collected data of the related works and real-life parking congestions issue. Furthermore, the evaluation will be based on the different areas of application such as economic, environmental, social, and business. Lastly, it scientifically compares related works to our proposed project based on the features they support.

## **1.7.6 Chapter Six**

Chapter six is a summation of the project's idea as well as what has been achieved based on previous chapters. Also, it summarizes all the chapters phases including developing, designing, and implementing of the Smart Parking System. Moreover, future considerations and improvements are further discussed.



# CHAPTER TWO

## LITERATURE REVIEW

## **CHAPTER 2: LITERATURE REVIEW**

### **2.1 Introduction**

The Parking Reservation System has become a new way to deal with congestions. Many researchers and companies tried to solve the issues of parking spots. This section reviews the related works to reservation parking system as it covers descriptions and evaluations. Although the literature covers many implementations to parking systems, the review will focus on most relevant works.

### **2.2 Related Works**

The parking reservation systems have a wide range of history that is changing according to the technology development, scope, and preferences. The existing solutions vary in technology used in implementation between sensors, controllers, communication protocols, and services offered to the users. This chapter aims to analyze a variety of car parking systems in order to satisfy the market demand. The Literature Review covers the functionality, services, and techniques designed to reserve parking spots. The next sections discuss the reviewed works.

#### **2.2.1 Car Parking System Using IR Sensors [4]**

In this project, the car parking system is built to assist the drivers to find an empty spot and to reduce the congestion when searching for a spot. This system is represented by infrared sensors, LCD display, and LED lights. Parking spots have active IR sensors which consists of transmitters and receivers. The sensors detect the presence of cars once they are parked by emitting infrared radiation. In addition, an 8051 microcontroller is used to take actions in the system. Each spot availability is checked in order to display a number on the LCD screen. The number indicates the empty spots available in the parking. When the car is parked the LED will turn from green to red indicating that the slot is occupied. The LCD screen at the entrance assists the drivers to find areas in the parking lot that have available spots. The components of this system consist of IR sensors which are shown in figure 2. Tickets are given for the drivers at the entrance of the parking, so they have to scan the ticket in order to pay and exit [4].

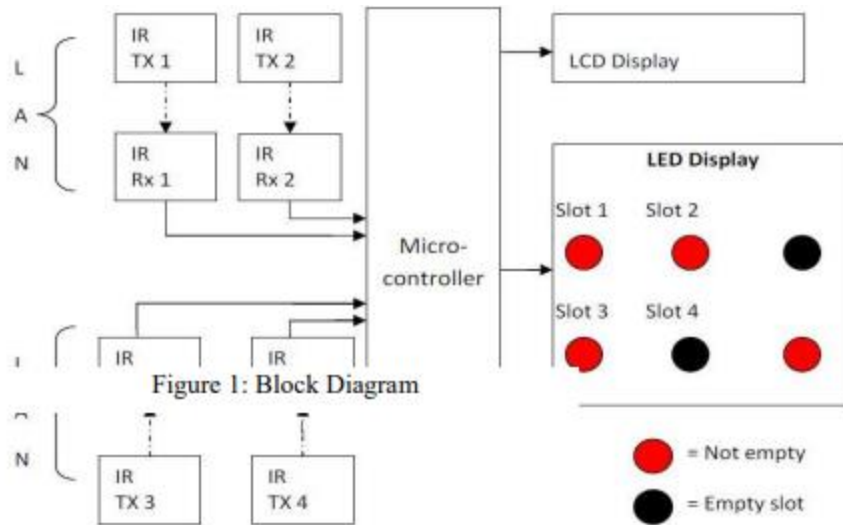


Figure 2 A Parking System Based on IR Sensors [4]

### 2.2.2 Car Parking System Based on Radio Frequency Identification [5]

In this paper, a Radio Frequency Identification Parking System is introduced. The system includes RFID tags, readers, and the RFID database. RFID technology is based on wireless communication that uses electromagnetic fields to identify the tags attached to objects. In this project, the RFID is used for tracking vehicles. It tracks drivers that enter and exit the parking area. In order to track the users, the assigned RFID must get scanned. The vehicles movement tracking is implemented for security purposes. Also, the RFID database stores the data that is transferred between the reader and the tags. The drivers must scan their assigned RFID tags through the readers to be able to enter the parking lot. According to the paper, the RFID tag works by the electromagnetic waves transmitted from the reader. The RFID readers, also known as interrogators which are devices that transmit radio waves in order to communicate with the RFID tags and receive from the database. The RFID database contains information about the expiry date, vehicles' plate numbers, and any other important data. This type of RFID tags has a frequency range of 13.56MHz as it communicates for approximately 3 meters. In order for the driver to enter the RFID parking area, they must register through the application. The admin's modules that the application has are the vehicle registration, RFID tag, and the staff module. [5].

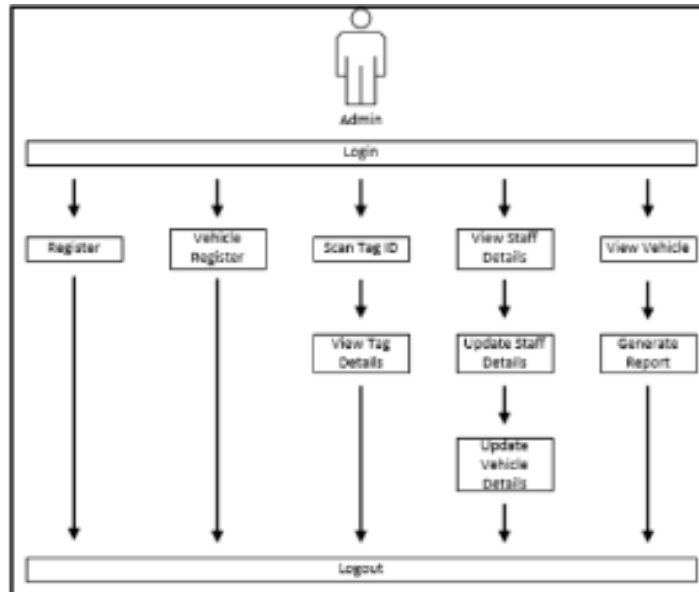


Figure 3 *RFID Based Parking System* [5]

### 2.2.3 Smart Parking System Using Ultrasonic Detectors [6]

This parking system is presented to help drivers find an empty spot in a shorter time. It is implemented by using ultrasonic sensors to detect whether the spot is vacant or occupied. At the entrance of the building, an LED board displays the number of the available parking spaces on each level to guide the drivers. After drivers choose the desired level, a display board shows them the number of empty spaces. Also, the system provides drivers with directions at the end of each aisle. Moreover, each parking spot has an LED that indicates the status whether it is occupied, vacant, assigned for handicapped, and reserved. The LED will light up in different colors including red, green, blue, and yellow respectively based on the states. There are ultrasonic sensors the spots that are based on echo location. The ultrasonic sensors work by transmitting sound waves to calculate the time between the transmitted and received waves. In addition, each spot has two parking detectors which alert the drivers for improper way of parking [6].

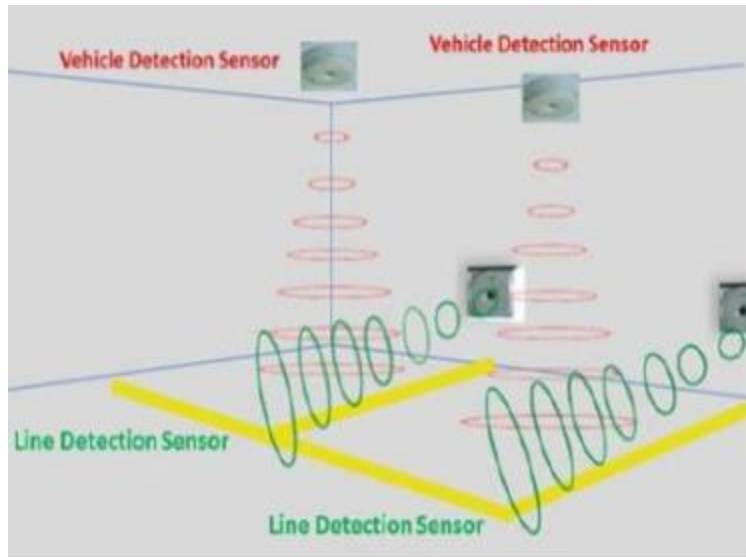


Figure 4 *Ultrasonic Detectors on Each Parking Spot* [6]

#### 2.2.4 A Systematic Review of Machine-vision-based Smart Parking Systems [7]

This system is based on survey conducted to solve parking lots problems using the vision technology. The system aims to solve the problems behind the parking lots. As mentioned, with the high rate of vehicle density finding a park lot is one of the exhausting processes during peak hours. This daily process wastes time and fuel, causes traffic congestion, and increases pollution. Because of that, it is conducted by implementing different methods of image processing and space detection. This paper reviews multiple vision-based systems and methods to detect vacant parking spaces via camera. The system has a series of vision algorithms as follows:

1. **Appearance based approach:** It is designed to compare the current parking space appearance with the occupied space original appearance.
2. **Recognition-based approach:** This approach uses a machine learning technique to recognize, detect and classify vehicles which occupy parking space. However, it cannot separate the desired object to capture as the vehicle can be surrounded by other objects such as walls or other objects shadows. The CNN system might be too complicated to adopt it in any park lot because it needs a high level of programming, precise calculations of space and high level of machine language. Thus, it needs to include three-dimensional image processing [7].

### 2.2.5 QR Code based Smart Parking System [8]

The system aims that the usage of QR code technology will win over sensors which is limited by weather conditions. The system architecture is applied using small vendors. Each user is assigned to a unique QR code that contains the user information such as name, car plate number, etc. The user has two options to reserve a parking space either from inside the parking lot or outside with the consideration to pay the charge in advance. The user can complete the payment through smart wallet such as Google Pay and BHIM. The entry time of the user starts once they scan the assigned QR code by the vendor machine. Moreover, the bills will be generated through the same process.

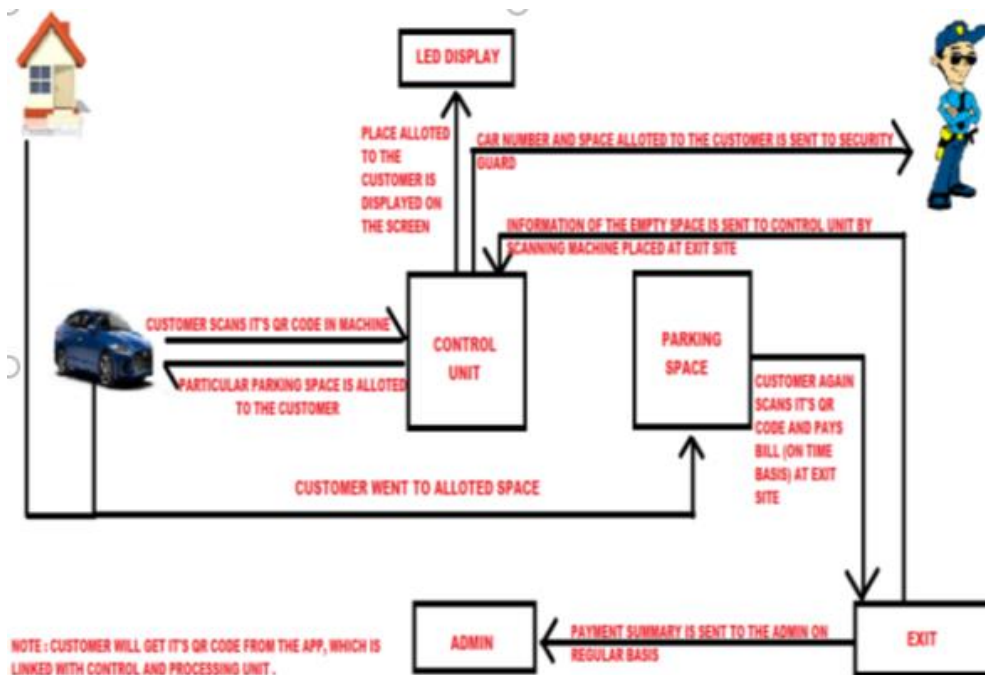


Figure 5 QR Code Based Smart Parking System [8]

The parking system is divided into online and on-site parking reservations to resolve any disputes that may occur. The vendor machine analyzes changes and updates the system on the admin's application. The algorithm of this system is based on first come-first-served, so the nearest park will be filled first. Thus, it may help to reduce congestions in the parking lot. Shown below, the figure illustrates the number of on-site reservations is higher compared to the other options. In addition, section C is used as a control unit in case someone parked at the wrong space [8].

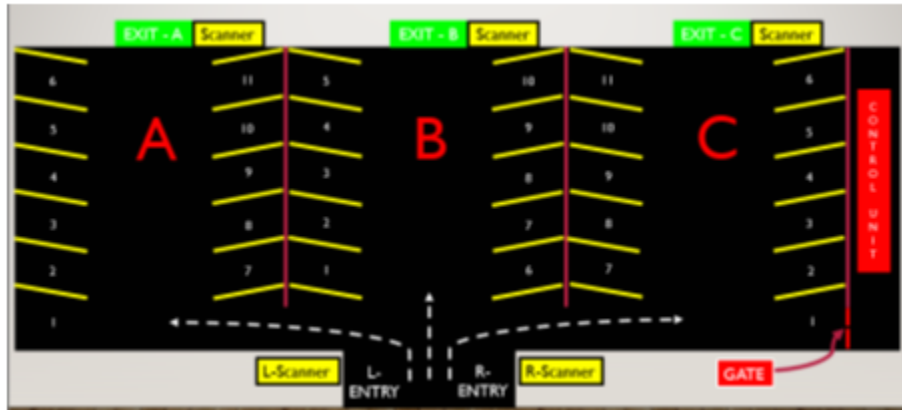


Figure 6 Number of Vehicles and Sections [8]

### 2.2.6 Wireless based Smart Parking System using Zigbee [9]

The installed LED systems in any parking lot counts the number of empty spots, but it does not indicate the location of the empty space. Usually, the installation of these LED boards is expensive. The paper aims to refine this process as it is designed to display the empty spots in real time. Also, this system is based on Zigbee that used for short range communications. It can be utilized with many applications such as WLAN. The next figure shows the flow chart using Zigbee.

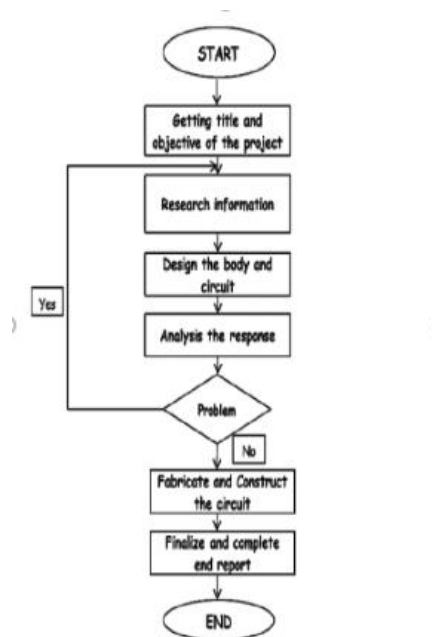


Figure 7 Wireless Based Smart Parking System Using Zigbee [9]

In this project, the software was developed using Visual Basic Studio. The software monitors the vacant parking spaces. It sends data from the units to display available car spaces. As

shown below, the figure illustrates the flow chart of the monitoring software. Besides, the operation will not be active unless the XBee communication protocol is a port between the software and the receiver [9].

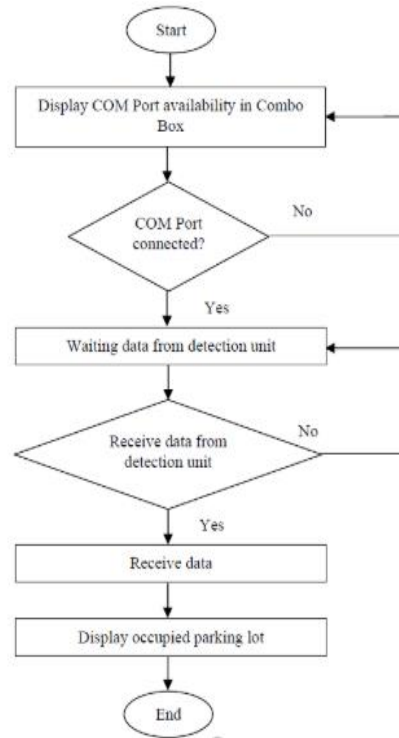


Figure 8 *Wireless Based Smart Parking System Using Zigbee [9]*

### 2.2.7 Smart Parking System Based on Bluetooth Low Energy Beacons with Particle Filtering [10]

BLE (Bluetooth low energy) referred as bacons are used to implement an intuitive parking system. The bacons are small transmitters to send signals as they contain unique identifiers of the location. The location can be determined via (RSSI) that stands for Received Signal Strength Indicator. The RSSI used in measuring the power level that the device is taking from an access point. The users can access the system from a smart phone as they can check the available car parking spaces. The system provides the real time data once the user picks space location. The system is designed to have a uniform resource locator (URL) that is transmitted by a unique bacon dedicated for each spot. When the spot is reserved, a clock will start to keep a track of the registration. Also, the clock will keep a track of the registration until the parking lot is released from occupation [10].

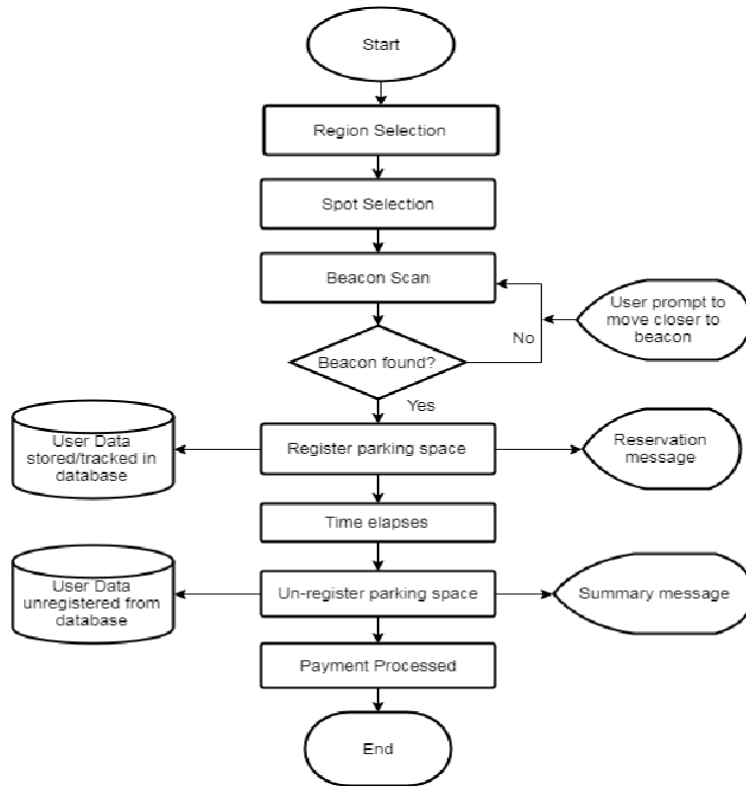


Figure 9 The System Chart [10]

### 2.2.8 Applying smart parking system with internet of things (IoT) design [11]

The system is based on Internet of Things (IoT) that shows the availability of empty spots. This system relies on the use of QR codes, in ground infrared sensors, parking lot monitor, and Raspberry Pi. The user can login and book a parking spot online to get a digital ticket. When the users arrive, they can enter the parking lot by scanning the QR code in the digital ticket. A parking board displays the number of empty parking spots in the floor that guides the drivers to find spots. After parking the cars, the infrared sensor sends a signal to the light located above each lot using short range wireless. The signal is sent in order to indicate the state of the parking. If the parking lot is empty, the light will be green. Thus, drivers can park their vehicles. In case the parking is unavailable or reserved, the light will be red. Also, this process transfers data to the cloud in order to update the system. Using the application, the users can pay for the parking fees. Also, users cannot exit the parking lot without the scanning QR code for security purposes [11].

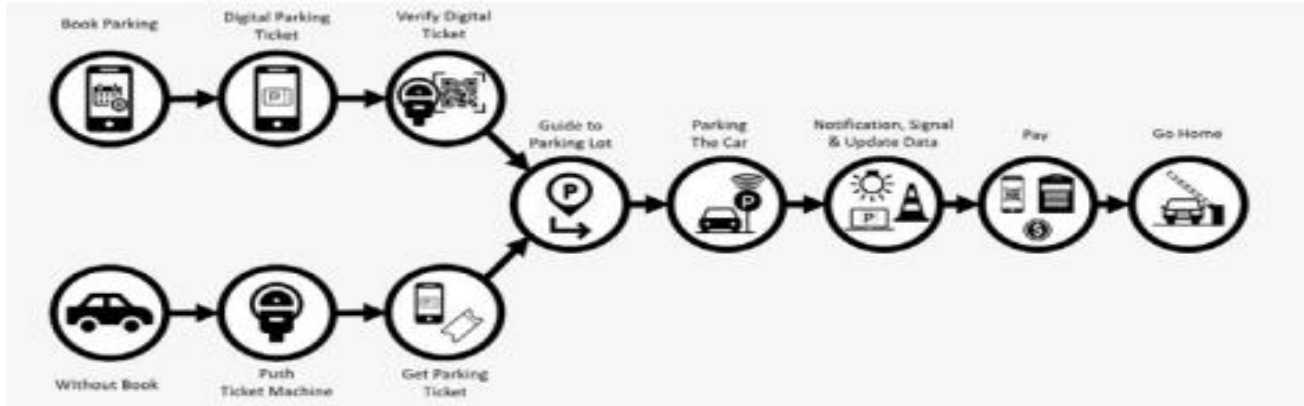


Figure 10 *The Procedure* [11]

### 2.2.9 Smart Parking System Using IoT [12]

The project is designed using Internet of Things (IoT) that handles the reservation process and show the spots availability. The project is based on the usage of GSM module, RFID card, and infrared sensors. RFID cards contain the drivers' registration; thus, it must be scanned for the entrance. After the cards are scanned, the data is transmitted to an Arduino board to check spots' availability. The infrared sensors sense the number of spots occupied. The GSM module notifies the users of empty spots. The system is equipped with a servo motor at the exit that does not open without RFID cards. The users are allowed to exit the parking only by using RFID cards.

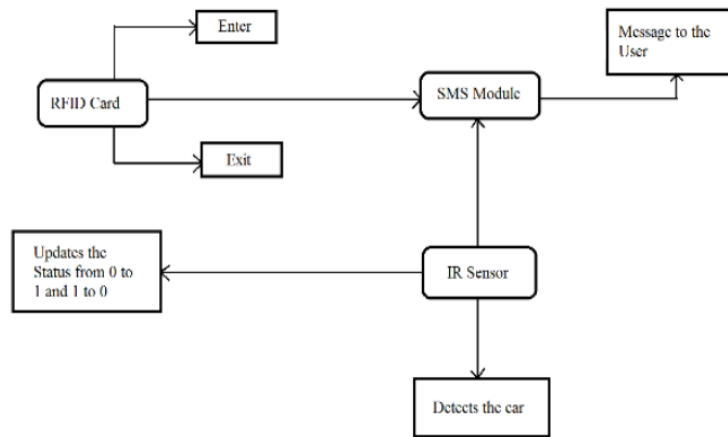


Figure 11 *System Flow Chart* [12]

The system is based on a cloud server that interacts with the users to check the spots availability. The sensor data is updated in the cloud; therefore, the SMS messages are dependent on the state of the parking. Also, the system's software has a Wi-Fi module in order to store the data in the cloud and communicate with the server [12].

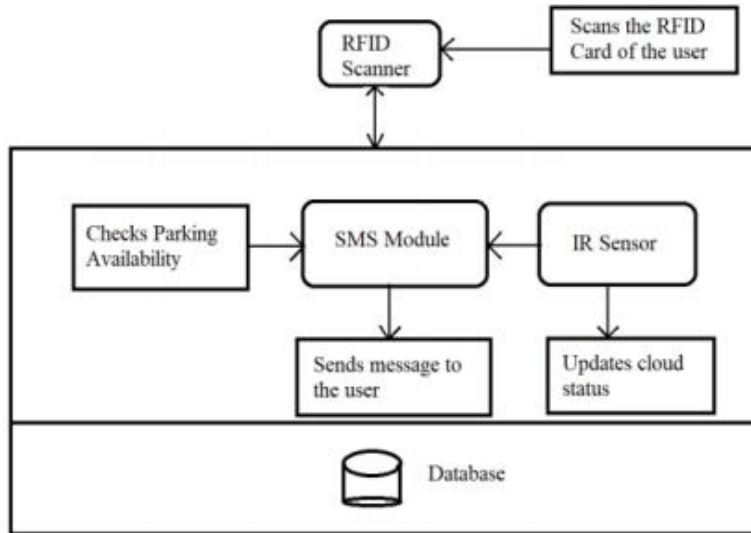


Figure 12 Software [12]

### 2.2.10 Internet of Things (IoT) based Smart Parking Reservation System using Raspberry-pi [13]

The system is designed using the Internet of Things (IoT) that provides registration services in order to park vehicles. The drivers enter their identity details on the mobile application that is connected to a server. The users have a time lapse of 15 minutes after reserving a parking spot through the application. When the users arrive, their vehicles' plates get scanned. The plate recognition is done using character segmentation to verify the number provided while registration. Also, the system is equipped with Raspberry Pi 3 to identify the users' faces in order to prevent thefts. At the entrance, each driver is provided with a number that indicates the nearest empty spot. After parking the car, a timer is activated to count the time that drivers spend. When drivers exit the parking, the timer is stopped. At the exit, the drivers' faces are recognized again which must match the database in order to leave the parking lot. In addition, the payment process is done through an e-wallet. [13].

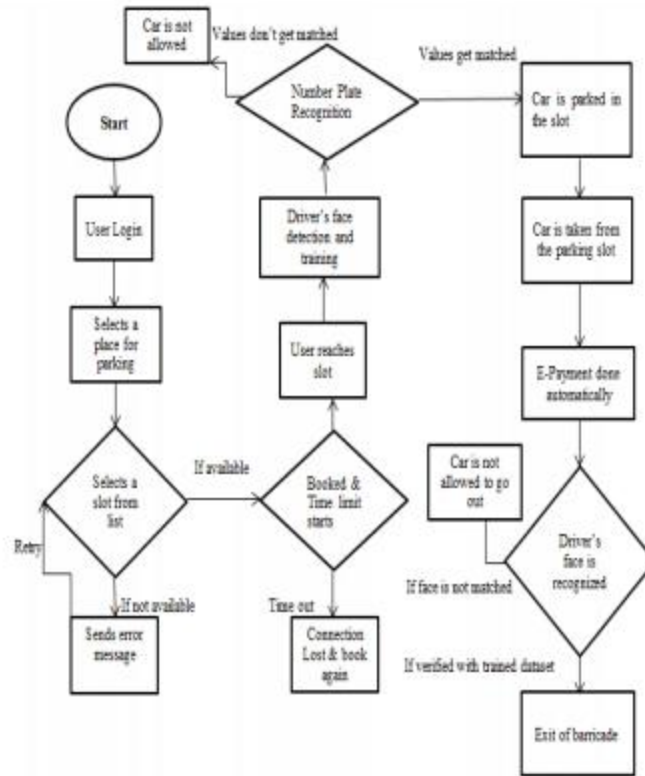


Figure 13 System Flow Chart [13]

### 2.2.11 Intelligent Parking Management System Based on Image Processing [14]

The project is designed to check the empty spots in a parking lot using RGB video image processing. The system uses image acquisition as there is a camera installed above the vehicles' spots. The installed camera captures the vehicles only, so it must be directed in an angle that prevents overlapping with other objects. Initially, the process starts by taking images of the parking lot without any cars in order for the system to record the data. The system is based on analyzing the RGB value through MATLAB commands to indicate the parking availability. It analyzes the difference between two video frames. The first frame is when there are no cars parked while the second frame is when cars are parked. Thus, this process indicates the availability of parking spots. Shown below is the image identification of empty and occupied spots [14].

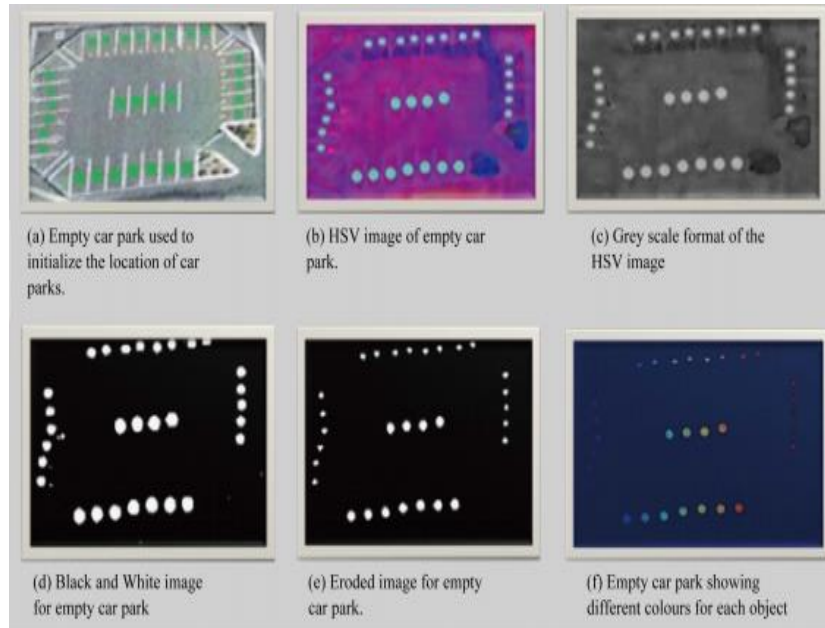


Figure 14 *Empty Parking* [14]

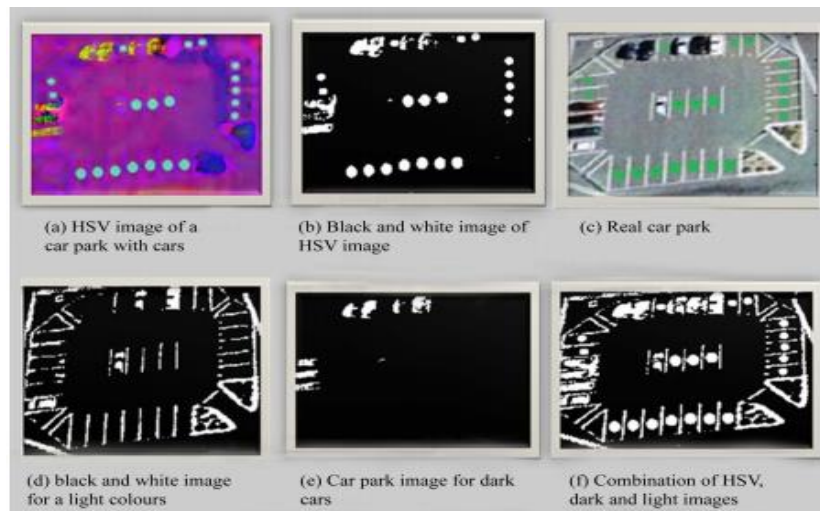


Figure 15 *Occupied Parking* [14]

### 2.2.12 E-Parking Software (Getpass) [15]

Getpass application is an existing system used in Kuwait to provide advanced reservation for the users. The system's application provides registration services that require the client to enter their identity information and cars plate numbers. This smart parking system uses Automatic Number Plate recognition for the entry and exit. Using the application, users can book a spot in a chosen row. However, users are not able to book a preferred spot as they can only choose a row in the parking lot. Also, users must take their time searching for an empty spot in those chosen rows.

Minimum amount of money must be available in the smart wallet before booking a spot. A timer is activated the moment users book a row until reservation is cancelled or the drivers exit the parking lot. This system minimizes the usage of security cameras since all cars' plates entered and exited are recorded. Thus, the process reduces administrative burden using the application. [15].

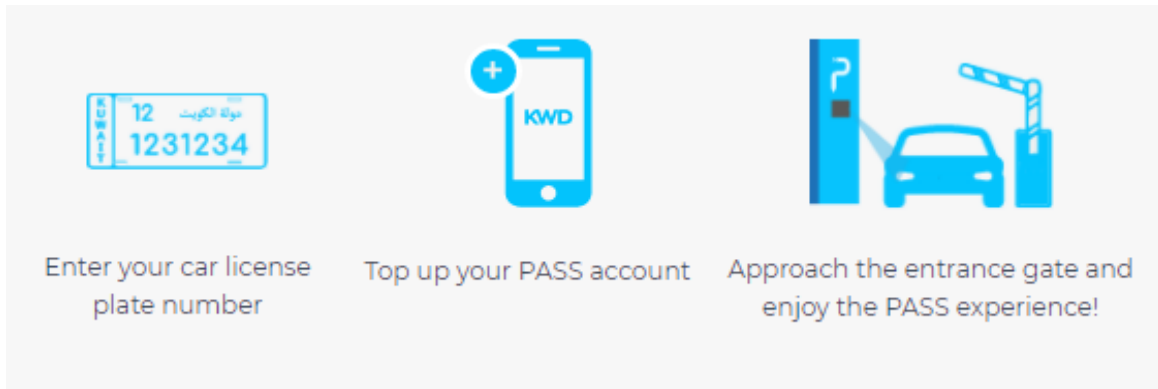


Figure 16 *Getpass Procedure*[15]

### 2.2.13 Mawaqif - Smart Parking Assistant [16]

Mawaqif parking system exists in Kuwait to provide parking reservation services for the drivers. Mawaqif is designed to recognize plates automatically using a camera at entrance and exit. Users must create an account on Mawaqif application in order to access the services. There must be a minimum amount of money in the smart wallet to be able to access the services provided by the application. Once the users make the payment, parking barcode tickets are assigned. When the users arrive, they must scan their assigned barcode tickets. After users scan tickets, the payment amount is deducted from the wallet. After scanning, the users will be able to view available parking spots through the application. Then, they must search for the spots displayed. The users can exit the parking lot without any further procedure [16].

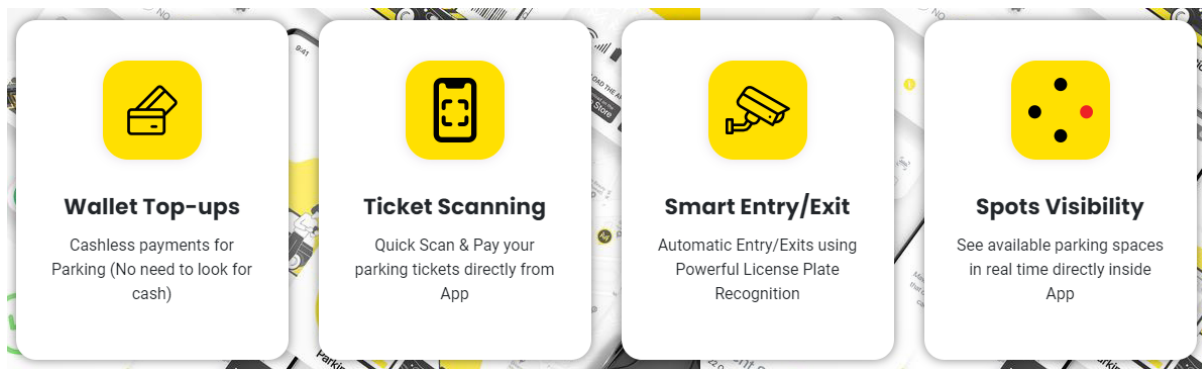


Figure 17 *Mawaqif Procedure* [16]

#### 2.2.14 Parkpiú Installed Car Parking System at The Palms Hotel in Kuwait [17]

Parkpiú is a car parking system available in The Palms Hotel located in Kuwait. This system parks cars automatically using traslo-elevators. The drivers have to enter a reception room and locate the cars on the traslo-elevator. Then, the cars are lifted by the elevator to reach an available parking spot. However, payment is done manually, and capacity of parking lots is limited [17].



Figure 18 *Parkpiú In Kuwait [17]*

Table 3 Summary of the key Features, Advantages, and Weaknesses of the Existing solutions

No	Work	Key Features	Advantages	Weaknesses
1.	Car Parking System Using IR Sensors [4]	<ul style="list-style-type: none"> <li>- Guiding drivers to their reserved spots</li> <li>- Showing status of spots.</li> </ul>	<ul style="list-style-type: none"> <li>- Low cost</li> <li>- The LCD monitor used in the entrance shows the status of empty spots.</li> <li>- The IR sensors guide the drivers to their chosen spots without searching which will save time and reduce traffic.</li> </ul>	<ul style="list-style-type: none"> <li>- Congestion in the entrance gate because of the LCD monitor and the time consumed by each driver to choose a spot.</li> </ul>
2.	Radio Frequency Identification (RFID) Based Car Parking System [5]	<ul style="list-style-type: none"> <li>- Handling the park entering process.</li> </ul>	<ul style="list-style-type: none"> <li>- Reduces needed manpower.</li> <li>- Improves traffic in rush hours.</li> <li>- Uses RFID tags to enter the parking lot instead of physical cards.</li> </ul>	<ul style="list-style-type: none"> <li>- Expensive</li> <li>- Inability to choose the preferred parking spot.</li> <li>- Problem in scanning the tags sometimes.</li> <li>- Implementation is difficult and time consuming.</li> </ul>
3.	Smart Parking System (SPS) Architecture Using Ultrasonic Detector [6]	<ul style="list-style-type: none"> <li>- Detecting improper parking.</li> <li>- Showing the spots availability.</li> </ul>	<ul style="list-style-type: none"> <li>- The ultrasonic sensors detect improper parking which alerts the driver to park inside the two lines of the spot.</li> <li>- Detects the availability of parking spots.</li> </ul>	<ul style="list-style-type: none"> <li>- Inaccurate readings can be encountered.</li> </ul>
4.	A Systematic Review of Machine-vision-based Smart Parking Systems [7]	<ul style="list-style-type: none"> <li>- Handling reservation process.</li> <li>- Showing the empty spots.</li> <li>- Showing the number of occupied spots.</li> </ul>	<ul style="list-style-type: none"> <li>- Counter-based system:                             <ul style="list-style-type: none"> <li>• Sensors detects number of entered and exited cars.</li> </ul> </li> <li>- Image and video-based system:                             <ul style="list-style-type: none"> <li>• Lower installation cost.</li> <li>• Very useful for outdoor parking areas.</li> <li>• Provides two categories which are car driven to focus on cars as objects or, space driven to consider the space availability.</li> </ul> </li> <li>- Vision-based Algorithms                             <ul style="list-style-type: none"> <li>• Single image can cover many parking spaces.</li> <li>• Lowered cost compared to sensor-based systems.</li> <li>• Images taken can be used as security monitoring.</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>- Counter-based system does not provide the location of the available park.</li> <li>- Vision based systems are expensive due to data amount that need to be transferred via a wireless network.</li> <li>- The CNN system might be too complicated to adopt it in any park lot because it needs a high level of programming, precise calculations of space and high level of machine language.</li> </ul>

			<p>- Convolutional Neural Networks: Deep learning of artificial intelligence to make computation act like humans.</p>	
5.	QR Code based Smart Parking System [8]	- Providing advanced and on-site reservations.	<p>- Small vendor to scan the QR code. - Each user gets a unique QR that provides [name-car number]. - Two types of booking: <b>Home:</b> needs advance payment to reserve a park <b>On site:</b> to scan the QR code. - The user needs to scan the code again to exit.</p>	<p>- QR codes are limited to smartphone only which make it a problem for drivers who still use old mobile phones. - Poor execution with lacking experience presents as an obstacle for adoption - No track of user journey after arriving on site from the QR code. - Payment is not part of the QR code system and offered through google wallet or any payment available.</p>
6.	Wireless based Smart Parking System using Zigbee [9]	- Showing the empty spots.	<p>- Zigbee:</p> <ul style="list-style-type: none"> <li>• Replaced the wired connections.</li> <li>• The system is cheap compared to using LEDs board which cost more and requires regular maintenance.</li> <li>• More accurate in terms of showing availability in real-time.</li> </ul>	<p>- Zigbee:</p> <ul style="list-style-type: none"> <li>• Suitable only for small places due to its short coverage range.</li> <li>• Low data transformation speed causes inaccuracy and complexity.</li> <li>• Insecure compared to a Wi-Fi.</li> </ul>
7.	Smart Parking System Based on Bluetooth Low Energy Beacons with Particle Filtering [10]	<p>- Tracking users' movements. - Guiding to the empty spots.</p>	<p>- It uses Bluetooth Low Energy (Beacons) which is very cheap, small, and suitable to any infrastructure. - Can be accessed via smart phones. - Provides real-time data. - User is given a unique URL that detects location of the spot. - Supported by Google's Eddy stone beacon protocol which is cost effective and useful.</p>	<p>- Trust and security limitations. - Tracking movement is not very practical. - Lack of data analysis which leads to limitations in developing the performance.</p>

8.	Applying smart parking system with internet of things (IoT) design [11]	<ul style="list-style-type: none"> <li>- Showing availability of empty spots.</li> <li>- Identifying users when exiting the park.</li> </ul>	<ul style="list-style-type: none"> <li>- Cloud database that provides security, automatic updates and reduced administrative burden. (Processing data)</li> <li>- Authentication services for example, users cannot exit the parking lot without a code.</li> </ul>	<ul style="list-style-type: none"> <li>- Users can enter the parking lot without reservations using paper tickets which leads to congestion and inconvenience to other users.</li> <li>- Many smartphones cannot read QR codes due to software version or the way of scanning which is a drawback.</li> </ul>
9.	Smart Parking System Using IoT [12]	<ul style="list-style-type: none"> <li>- Handling the reservation process.</li> <li>- Showing the spots availability.</li> </ul>	<ul style="list-style-type: none"> <li>- Users interact with a cloud server to check the slots availability that reduces human interactions.</li> </ul>	<ul style="list-style-type: none"> <li>- Users cannot choose their preferred slot.</li> <li>- RFID cards are used once the user enters the parking area to check the slots availability which consumes time and money.</li> <li>- GSM module can cause lags as multiple users are on the same bandwidth.</li> </ul>
10.	Internet of Things (IoT) based Smart Parking Reservation System using Raspberry-pi [13]	<ul style="list-style-type: none"> <li>- Registering to the service in order to park.</li> </ul>	<ul style="list-style-type: none"> <li>- Face identification is used to prevent problems such as vehicle theft.</li> <li>- Reduces congestion as vehicles enter the parking lot with the plate number.</li> <li>3. Authentication services as users must register to the system.</li> </ul>	<ul style="list-style-type: none"> <li>- Raspberry Pi camera can be affected by bad weather and physical touch.</li> <li>- Users usually have privacy concerns regarding face recognitions.</li> </ul>
11.	Intelligent Parking Management System Based on Image Processing [14]	<ul style="list-style-type: none"> <li>- Guiding drivers to the empty spots.</li> </ul>	<ul style="list-style-type: none"> <li>- MATLAB commands to indicate the parking availability</li> <li>- Optical scanners are used to identify the users.</li> </ul>	<ul style="list-style-type: none"> <li>- The optical scanner may fail to detect the right plate number due to poor quality caused by bad angles and weather condition.</li> <li>- The message sent at the parking and unparking time is not efficient if there are network issues.</li> <li>- GSM module can lead to delays.</li> </ul>
12.	E-Parking Software (Getpass) [15]	<ul style="list-style-type: none"> <li>- Advanced reservation.</li> </ul>	<ul style="list-style-type: none"> <li>- Reservation for a parking spot before arriving saves time.</li> <li>- Eliminates parking tickets and cash usage.</li> </ul>	<ul style="list-style-type: none"> <li>- The Automatic Number Plate Recognition will not be useful in case of front plate is not available or unclear recognitions.</li> <li>- Inability to reserve a specific spot, so users waste time in searching.</li> </ul>

				<ul style="list-style-type: none"> <li>- Minimum amount of money must be available in the smart wallet before booking a spot.</li> <li>- Minimizing usage of security cameras since all cars' plates entered and exited are recorded may not be that secure.</li> </ul>
13.	Mawqif - Smart Parking Assistant [16]	<ul style="list-style-type: none"> <li>- Advanced payment for users.</li> </ul>	<ul style="list-style-type: none"> <li>- The Smart wallet scans the parking tickets and deducts from the money available that is easier for the user.</li> <li>- If the cash is not available, they can pay online.</li> <li>- Mawqif does not charge fee on parking time.</li> </ul>	<ul style="list-style-type: none"> <li>- Parking ticket loss causes inconvenience to drivers in paying and exiting.</li> <li>- Not all issued parking tickets have a clear bar to scan.</li> <li>- Drivers who do not use the app must pay using the traditional way.</li> </ul>
14.	Parkpiú Installed Car Parking System at The Palms Hotel in Kuwait [17]	<ul style="list-style-type: none"> <li>- Handling parking process.</li> </ul>	<ul style="list-style-type: none"> <li>- The car gets automatically parked.</li> </ul>	<ul style="list-style-type: none"> <li>- Expensive to construct.</li> <li>- Manual payment.</li> <li>- Limited number of spots.</li> </ul>

### 2.3 Summary and Implications

Literature Review has revealed a gap in existing parking reservation systems. Many of reservation parking system implementations vary in efficiency and functionality. The variation between systems implicated the points of improvements that must be done. As shown in table 2.1, available systems oversight some issues such as inability to choose the preferred parking spot. In addition, current systems lack power saving plans which prevents them from the integration into the smart societies in the near future. To summarize, determining the advantages and disadvantages of diverse parking reservation systems will open a room for improvement to this project. The similarities and difference between this project and the related work are highlighted in Figure 19 as shown below:

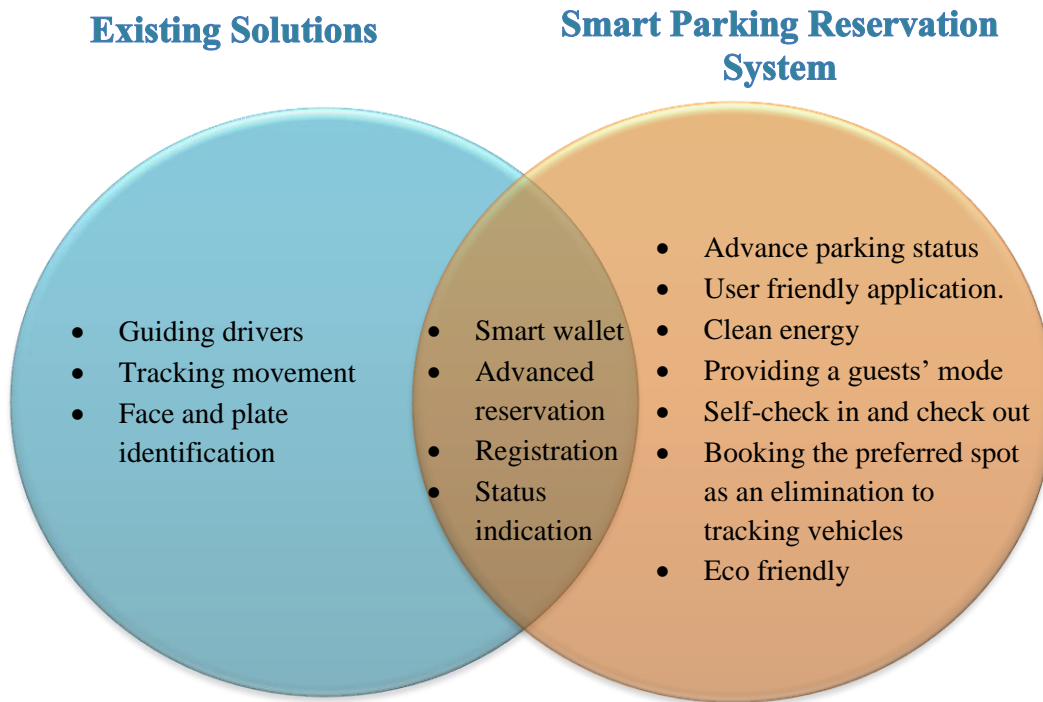


Figure 19 Comparison Between Key Features of Existing Solutions and Proposed Work

## 2.4 Conclusion

To conclude, Chapter 2 sheds the light on different approaches to implement parking reservation systems through analyzing a variety of existing projects. The Literature Review helped to improve the features and services required to meet the market demand. Some existing systems do not offer the same services as this project aims. However, they have the same objective in solving the issue behind finding a parking spot. Thus, the literature has a wide range of techniques that are needed to be modified in this project.



# CHAPTER THREE METHODOLOGY, DESIGN, AND ANALYSIS

## CHAPTER 3: METHODOLOGY, DESIGN AND ANALYSIS

### 3.1 Introduction

Chapter 3 discusses the methodology, design and analysis required to implement the Smart Parking Reservation System. The waterfall methodology is used to build the project in sequential approach. Also, a market research is conducted to search for design alternatives that are critical to choose the components of the project. Software and hardware components' features are studied to provide the budget of the Smart Parking Reservation System. Throughout the chapter, block diagrams will be used to clarify the system's functionality, analysis, and design. Towards the end of the chapter, the ethics and limitations are stated in order to maintain project efficiency.

### 3.2 Methodology

The methodology describes the steps required to build the hardware and software in order to reach the main goal of this project. The Smart Parking System follows the waterfall methodology which is a sequential method with aligned phases. The phases help engineers to deliver products to the market. They facilitate the process of deciding the project requirements, analysis, design, testing and coding. In waterfall methodology each phase must be completed before processing to the next one. Shown Below, the stages of the waterfall methodology.

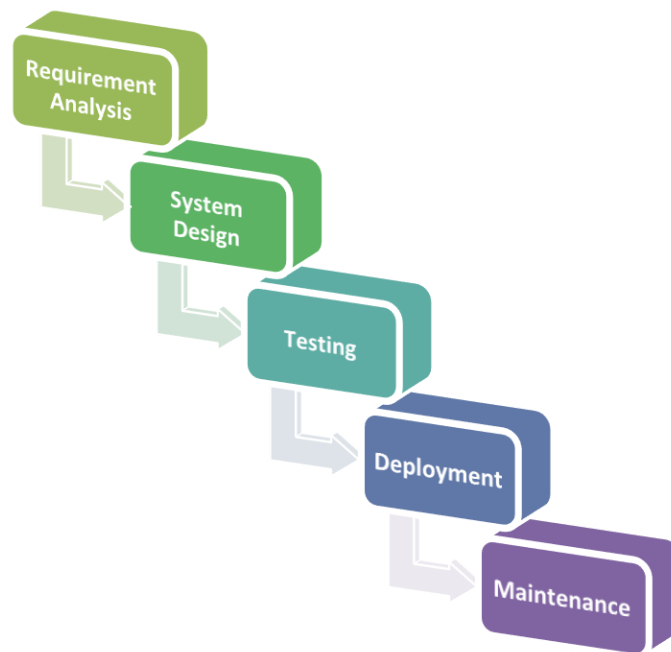


Figure 20 *Waterfall Methodology*

### 3.2.1 Requirements:

There are two types of requirements for any project development. The first one is user requirements which refers to the needs as it describes the benefits from the suggested product. The second one is the system requirement which describe functional and non-functional types.

#### 3.2.1.1 USER Requirement Specifications (URS):

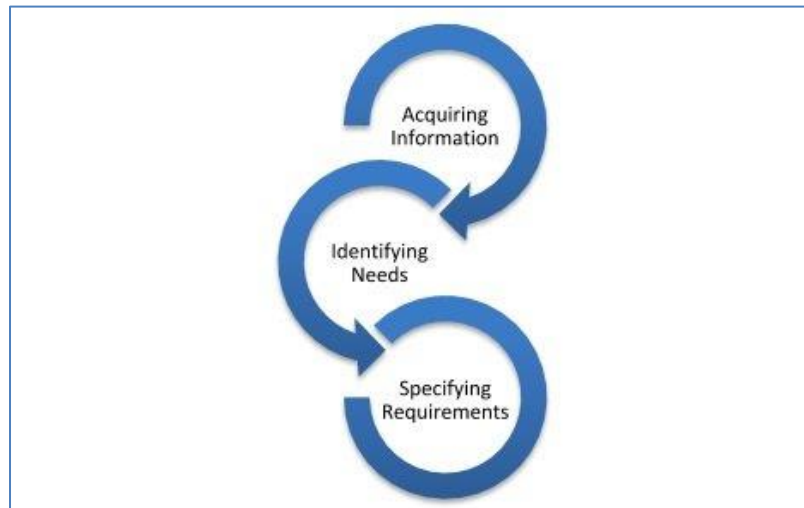


Figure 21 *USER Requirement Specifications (URS)*

#### 3.2.1.2 User Requirement:

- The system provides ability to register and login.
- Ease of accessibility.
- Must support operating systems on smart phones including iOS and android.
- System should be able to serve more than one user at the same time.
- System should preserve user private data and information.
- System should notify the user of their park status and personal wallet amount whenever reservations occur.

Table 4 *Functional and Non-Functional Requirements*

Functional Requirements	Non-Functional Requirements
<ul style="list-style-type: none"> <li>• Collects data from user when they register.</li> <li>• Input data: Name, email, mobile and password.</li> <li>• Requires: identifies who can enter the parking lot.</li> <li>• Description: users interact with the system via their mobile to enter the reserved parking spot.</li> <li>• Pre-condition shows the availability as yellow, green, and red lights which are booked, available and occupied respectively</li> </ul>	<ul style="list-style-type: none"> <li>• Usability: useful to be used by all users' segments.</li> <li>• Efficiency: user-friendly application for booking and payment.</li> <li>• Dependability: training a support team of employees to solve unintended problems that may face users due to bugs or errors. Such issues can be encounter in any software.</li> <li>• Performance: quick response for the user's request.</li> </ul>

### 3.2.2 Software and System Design:

The software is chosen based on the components required to build the system. One of the main ideas in this project is the usage of Internet of Things. Besides, each park has a gate that can be controlled via Internet of Things (IoT). It is implemented to let users self- check in and out, pay, and reserve using mobile phone. As a result, the system is operated via mobile applications, which provides users with the desired functions. The following figures show general flow charts of the software:

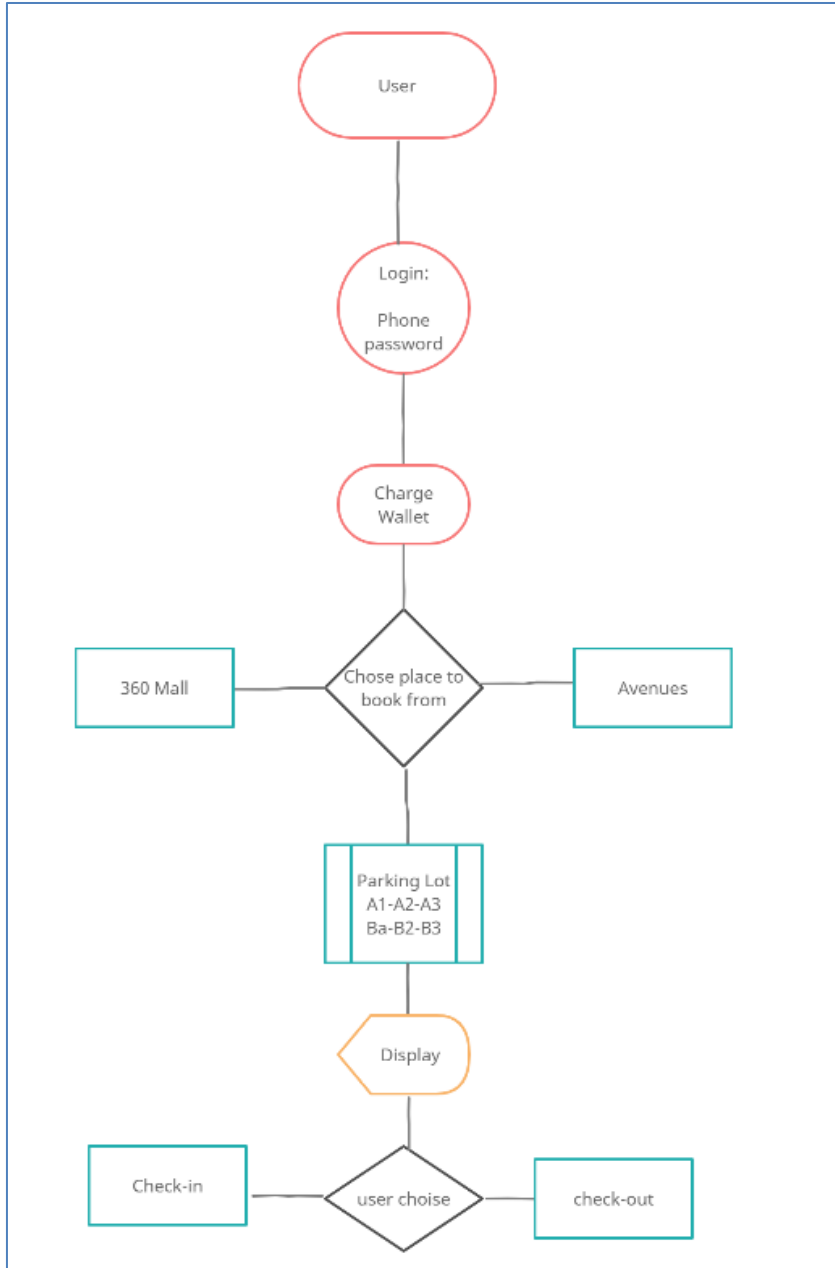


Figure 22 Software of the System

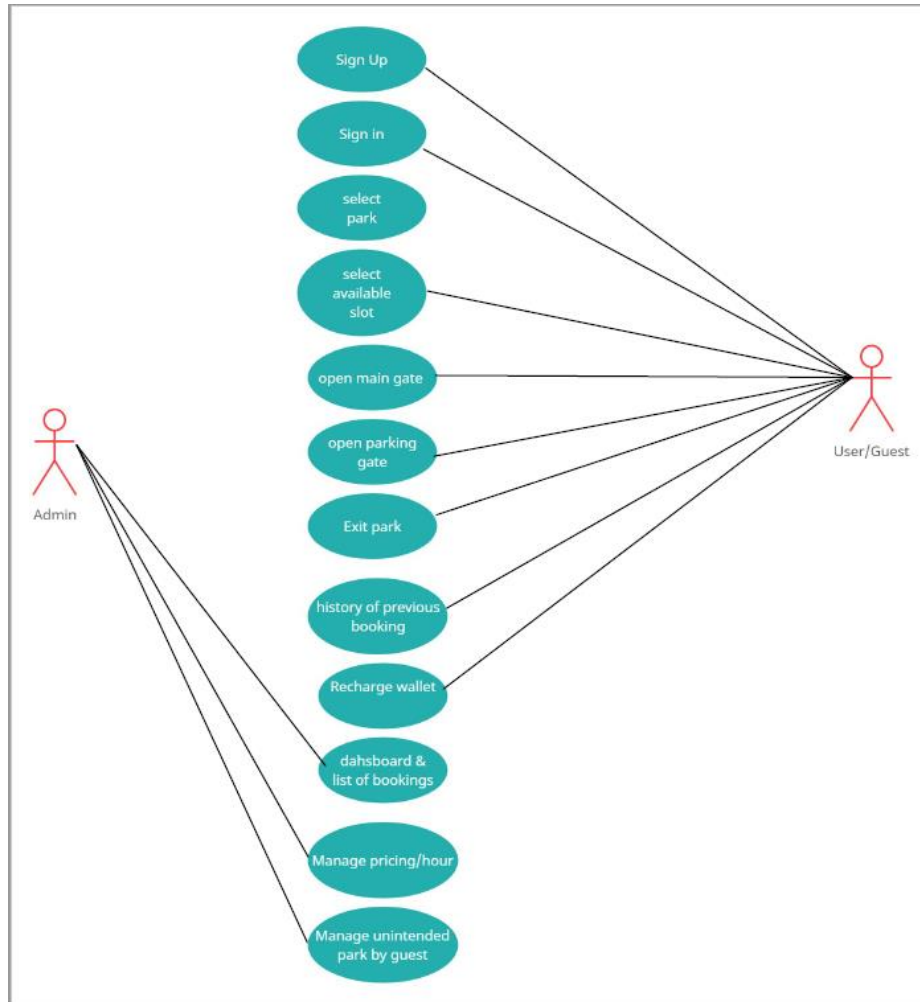


Figure 23 Use Case Diagram for the Smart Parking Reservation System

### 3.2.3 Implementation and Unit Testing:

The very initial step in implementation and testing is through programming the gate that is operated using a servo motor. The main gate is programmed to be related to the EBot 8 Pro boards. The simulation using software tool called EBot Blockly. The gates' bodies are designed using 3D printer to have the servo motor inside, so it allows them to open and close. Also, each gate's structure includes a cavity for the LED to indicate the state of the parking spot.

### 3.2.4 Integration and System Testing:

The system is operated using a solar panel of 18 Volts that is equipped with a voltage regulator for the output voltage and current. They come from solar panel to charge the battery responsible for powering the system. The battery of the system has the value of 12 Volts due to the solar panel that has an output voltage of 12-18 Volts. Thus, the integration of the system

hardware was implemented to match the needs of the project. This part will be elaborated further in the following sections.

### **3.2.5 Operations and Maintenance:**




The project is delivered to targeted market by installing a software related to the hardware in servers. The targeted market of Smart Parking Reservation System are the developed countries such as Kuwait. However, the Smart Parking Reservation System is a prototype that will be limited to a small community. Because of that, the local and global deployments of the project may not take place, or it will put into consideration in future. Also, the maintenance is applied based on the clients' experience with the project including bugs and updates. Versions are released to meet the market demands depending on fluctuations in the economy, technology, trends, and many other conditions. This part will be considered in the future development for the Smart Parking Reservation System.

## **3.3 Research Design**

The research design depends on the components available in market. After analyzing the products available in the market to implement the Smart Parking Reservation System, a variety of alternatives were found. The alternatives are categorized depending on the features, advantages, and disadvantages; therefore, a decision is made about the software and hardware components. The analysis of the components is conducted based on the independent and dependent variables of the project. The independent variable is the vehicle density that has an effect on the dependent variable which is the parking lot availability. The project aims to ensure efficiency and accuracy in detection, implementation, and features to be suitable for the scope of this project. The research design will be divided into possible two alternatives based on the microcontrollers and sensors.

### 3.3.1 Design Alternatives

Table 5 Options Available for Microcontrollers

	Arduino Micro [18]	Arduino Mega 2560 Rev3 [19]	EBot 8 Pro [20]
<b>Model</b>			
<b>Processor</b>	ATmega32U4 8-bit CPU <ul style="list-style-type: none"> <li>• 16MHz clock speed</li> <li>• 2.5KB SRAM</li> <li>• 32 KB flash memory</li> </ul>	ATmega2560 8-bit CPU 16MHz clock speed 8KB SRAM 256 KB flash memory	ATmega1284P 8-bit CPU 16MHz clock speed 16KB SRAM 128 KB flash memory 2 DC motors
<b>Features</b>	24 digital I/O pins 7 pins for PWM output 12 analog inputs USB connection	54 digital I/O pins 15 pins for PWM output 16 analog input pins USB connection	16 digital I/O pins 8 analog input pins 8 output connectors USB connection
<b>Advantages</b>	<ol style="list-style-type: none"> <li>1. Small size and light weight</li> <li>2. Switching regulators for wider voltage range</li> </ol>	<ol style="list-style-type: none"> <li>1. Large space to handle complex projects</li> <li>2. Many pins to use</li> </ol>	<ol style="list-style-type: none"> <li>1. Built-in features: buzzers, RGB LEDs, and temperature sensors that can be used to implement projects.</li> <li>2. Default programming by using Bluetooth programming using mobile phone</li> </ol>
<b>Disadvantages</b>	<ol style="list-style-type: none"> <li>1. Low memory</li> <li>2. Small size may be an issue due to the need to work in a tiny microcontroller</li> </ol>	<ol style="list-style-type: none"> <li>1. Large in size may be not suitable for a project that needs several microprocessors</li> </ol>	<ol style="list-style-type: none"> <li>1. Small in size that will be difficult to work with for some projects.</li> <li>2. Requires less wiring connection which can be suitable for prototypes.</li> </ol>

After analyzing features, advantages and disadvantages of different microcontrollers, a decision was made. In order to ensure quality and efficiency in Smart Parking Reservation System, the EBot 8 Pro is chosen as a microcontroller. The EBot 8 Pro meets the project requirements because it has features to achieve the goal. EBot 8 Pro will be illustrated in the hardware section.

Table 6 Options Available for Sensors

	Infrared Sensor [21]	Ultrasonic sensor [22]
Features	1838B 3.3 - 5V DC voltage 35° Angle 2cm – 30cm distance range 20mA supply current	HC-SR04 5V DC voltage 15° Angle 2cm-80cm distance range 15mA supply current
Advantages	1. Cost effective 2. Adjustable potentiometer to meet the requirements 3. Built-in light sensor	1. Cost effective
Disadvantages	1. Sensitive because it can shatter	1. Can be affected by noise 2. Sensing accuracy can be affected by changes in temperature

After discussing sensors alternatives, the IR sensor will best fit the Smart Parking Reservation System since it has features suitable for the project requirements. The IR sensor ensures efficiency required to implement this project. The IR sensor will be discussed more in hardware section.

### 3.4 Hardware Components

#### 3.4.1 Microcontroller Board, EBot 8 Pro [20]

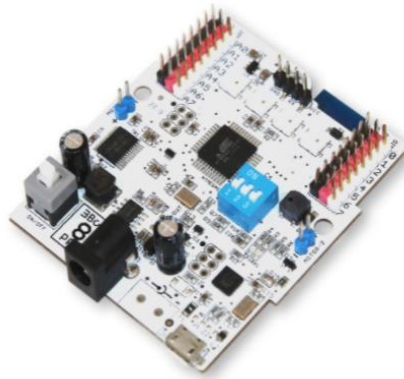


Figure 24 EBot 8 Pro [20]

EBot 8 Pro is an Atmega1284p microcontroller board that is based on Arduino. This microchip has 16 input and output pins as table 7 illustrated. The board has 8 pins for the input and 8 output connectors. In addition, EBot Pro has many features such as RGB LEDs, Bluetooth,

buzzer, etc. This microcontroller can be programmed using EBot Blockly and Arduino IDE either using a computer or a smartphone.

As mentioned in the Design Alternative section, EBot 8 Pro is chosen. It is the suitable microcontroller for the prototype because it has features that meet the requirements. The EBot 8 Pro is used to read inputs and provide outputs for the parking. The system's software is related to the microcontroller through the communication layer for interaction. In this project, the EBot 8 Pro is a suitable microcontroller because it requires less wiring as there is no need for a breadboard to make connections. In addition, it is a light weighted microcontroller which is suitable for prototypes. EBot 8 Pro programming software assists to eliminate errors because it provides graphical representations of the instructions that are based on C++ language.

Table 7 *Features of the chosen Microcontroller [20]*

Type	EBot 8 Pro
Processor	16 MHz
Operating Voltage	5 V
Output Current	150 mA
I/O Pins	16
Price	15 KD

### 3.4.2 EBot IoT Board [23]

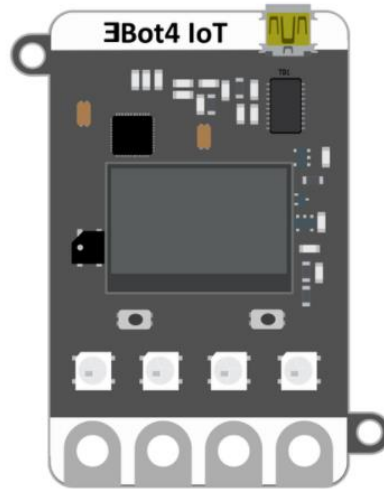


Figure 25 EBot IoT Board [23]

Since Smart Parking Reservation System is based on the Internet of Things, an IoT Board is needed for the communication between the system's application and microcontrollers. It has 4 analog inputs and 4 digital outputs. Furthermore, it has many built-in features, for example Wi-Fi, rechargeable battery, Bluetooth chips, power amplifier, and power management module. The EBot IoT Board is an ESP32 developmental powered board that can be connected to Wi-Fi in order to transfer sentences to the microcontrollers. Also, it is a useful component to build an IoT based projects that are related to cloud-based servers. The features of this IoT board are shown in table 8.

Table 8 Features of the IoT Board [23]

Model	EBOT IoT Board
Processor	26 MHz
Operating Voltage	5 V
I/O Pins	8
Memory Size	4 MB
Price	16 KD

### 3.4.3 Micro Servo [24]



Figure 26 *Micro Servo 9g [24]*

The micro servo is a rotational hardware component. there are two types of servos either open-loop servo or closed-loop servo depending on the range of rotation. The one that is used in this prototype is the closed-loop servo; also called standard hobby. It has a rotational movement range of 90 or 180 degrees. In addition, it has three important components which are the gears, DC motor, and the position potentiometer. The controller sends a signal to the position potentiometer in order to monitor the position of the output shaft. The micro servos are used for opening and closing the parking gates. The features of this micro servo are in table 9.

Table 9 *Features of the Micro Servo [24]*

Model	Micro Servo
Weight	9 g
Size	23.2 x 12.5 x22 mm
Limit Angle	120 degrees
Operating Voltage	4.8 V – 6 V
Gear Type	Plastic
Price	1.750 KD

### 3.4.4 EBot RGB LED [25]



Figure 27 RGB LED [25]

This is a WS2812B intelligent LED light controller which has a control circuit and an RGB chip integrated in one package of 5050 components. EBot RGB LED has four pins. Unlike other RGB LEDs, this component has serial string of 256 different levels for adjusting colors. It is needed to indicate the status of the parking space whether it is occupied, booked, and available. It lights up depending on the states in red, yellow, and green colors, respectively. The features of this component are shown in table 10.

Table 10 Features of RGB LED [25]

Model	RGB LED (WS2812B)
View Angle	120 degrees
Green	(522-525 nm) @ 660-720 mcd
Red	(620-625 nm) @ 390-420 mcd
Blue	(465-467 nm) @ 180-200 mcd
Pins number	4
Current	Draws 50 mA at 5 V with the three colors at full brightness
Price	1.500 KD

### 3.4.5 EBot IR Sensor [26]



Figure 28 IR Sensor [26]

The infrared (IR) sensors are electronic devices that detect and measure the infrared radiations which are useful to sense nearby objects. In specific frequency ranges, the IR sensors can sense the infrared signals and convert them into electric signals as outputs. The infrared radiations have wavelengths between 0.75 and 1000  $\mu\text{m}$ . This type of IR sensor is active as it can emit and detect the infrared radiations. It requires only three connections thus it is suitable for the prototype. In this project, it detects the presence of the car in the parking space. The features of the IR sensor are shown in table 11.

Table 11 Features of IR Sensor [26]

Model	EBot IR Sensor
Vcc	3.3 Vdc to 5 Vdc input supply
Gnd	Ground input
IR Emitter	Infrared LED Emitter
IR Receiver	receives signals transmitted by the Infrared emitter.
Price	1.500 KD

### 3.4.6 Solar Panel



Figure 29 *Solar Panel 18V*

The project is based on solar energy; therefore, solar panel is the main component to supply the prototype. A solar panel is a device that produces electricity when exposed to sunlight. It consists of photovoltaic cells which are interconnected together to form solar modules. When light strikes the solar panel, electrons are disconnected from their atoms and circulate freely across the cell. Thus, they will be ready to conduct electricity for the system. The solar panel generates electricity for the whole system which contributes to lower the costs of electricity bills and save the environment. Figure 29 shows the solar panel of the system. The features of this solar panel are in table 12.

Table 12 *Features of the Solar Panel*

<b>Model</b>	<b>Solar Panel</b>
<b>Max Power Voltage</b>	18 V
<b>Max Power Current</b>	1.11 A
<b>Weight</b>	2.2 Kg
<b>Dimension</b>	480*340*23mm
<b>Price</b>	9 KD

### 3.4.7 The Battery and Charge Controller



Figure 30 12V Battery



Figure 31 Charge Controller

The prototype requires 12 Volts to function based on the chosen components including the solar panel. Thus, a need for a battery and a charge controller is important for this PV system. A 12 Volts lead-acid battery is used as a form of energy storage (figure 30). The battery stores the excess output power from the photovoltaic panel. Also, it is used to compensate for the power demand of the load as if the output power of the solar panel is less than the load demand, these batteries come in handy. Lead-acid batteries are one of the rechargeable types. They are the most common used batteries in PV applications because of their well performance in deep discharging. Besides, their cost is low, and they have a long lifetime compared to other batteries. In addition to the battery, a charge controller is used to prevent the battery from deep discharging or overcharging thus preserving its life and ensuring a better performance as shown in figure 31. Moreover, the charger regulator regulates the power coming from the solar panel to the load. The charge controller has a screen that shows the voltage and current obtained from the solar panel.

## 3.5 Software Programs

### 3.5.1 Arduino IDE [27]



Figure 32 *Arduino IDE* [27]

The software used to program the microcontrollers and the IoT board is the Arduino Integrated Development Environment (IDE) as shown in figure 32, which is an open-source platform application. It uses C++ as its programming language. It is the most common used software because of its easy-to-use feature as it supports multiple of Arduino boards. This application is available in many operating systems such as MAC and Windows.

### 3.5.2 EBot Blockly [28]



Figure 33 *EBot Blockly* [28]

Programming EBot microcontrollers can be done using EBot Blockly which supports smartphones and computers. It is a software that uses graphical representations to execute instructions. Therefore, codes can be generated using drag and drop of the graphical available blocks. It has a live control features that allow the developers to control the devices used in the prototype such as the micro servo. This application is used to test the control circuit as an initial step in designing the prototype.

### 3.5.3 Visual Studio Code [29]



Figure 34 *Visual Studio Code [29]*

The Visual Studio Code is a source code editor software which supports many programming languages such as Java, C++, JavaScript, Node.js, and Python. The Visual Studio Code is used with the Mobile App Flutter to develop applications for iOS, Android, etc. Through Node.js, database MongoDB, Dart and JavaScript, the application for The Reservation Parking System is developed. Further details will be discussed in Chapter 4.

## 3.6 Analysis

### 3.6.1 Block Diagram Analysis:

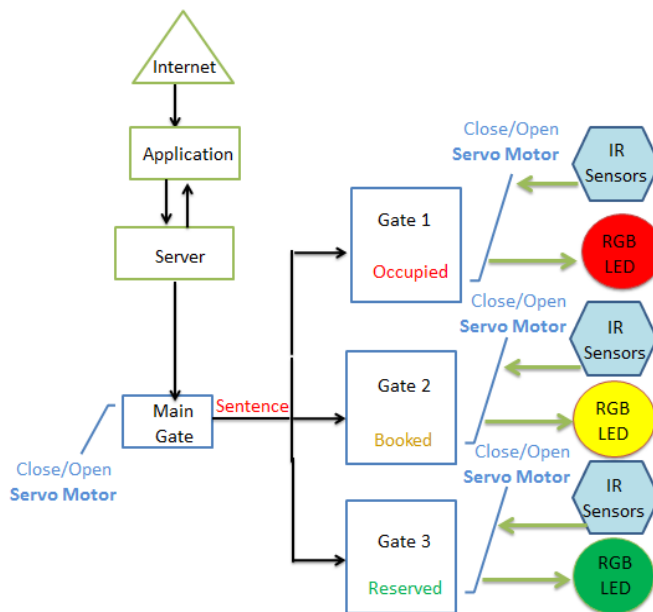


Figure 35 *Block Diagram for the System*

Shown above, a block diagram for three gates of the system representing the available, occupied, and booked possible states. The system changes the states of the parking spots between available, occupied, and reserved. Also, it can open or close the gates of the parking lot. To implement these features, the application is linked to a server via internet. The microcontrollers

are connected to the network to perform functions related to the system. In this project, 2 microcontrollers are needed for the main and spots' gates. Through the server, the application is linked to the hardware for data communication. In this project, a IoT board is utilized to link the system into internet and to connect it to the IoT server. The application sends information to the server, so it can communicate with the hardware. The server is connected to the IoT board, so the data can be received. The IoT board transfers the information coming from the application to the microcontrollers. Therefore, the IoT system is responsible to receive data from the internet to communicate with the remaining 2 microcontrollers. The data is either related for changing the status or open and close the gates. The information transmitted for the remaining microcontrollers in forms of sentences. For example, format 1 has "#S" which means the sentence is about the status that has one of the possibilities including booked (B), occupied (O), and available (A). Thus, the sentence is executed as yellow, red, and green LEDs, respectively. If there is an "#A." then it is related to opening and closing the gates. The process of the sentences' execution is discussed further in Chapter 4. The sentences are sent via internet which represents the link layer of the system. The link protocol in the parking reservation system is 802.11 which is the Wi-Fi protocol. In this system, the IP address is IPv4 that offers 32-bit addresses. The transport layer consists of error handling, setting, and controlling functions for the connection. The transmission line protocol of the system is MQTT which is based on TCP. The TCP is a full duplex protocol which allows the client and server to communicate at the same time. The application and the server can send and receive data back and forth. However, the microcontrollers and the IoT system are in one way communication. The IoT board transfers sentences to the microcontrollers which are responsible to understand and run the code on the system.

### 3.6.2 System Architecture:

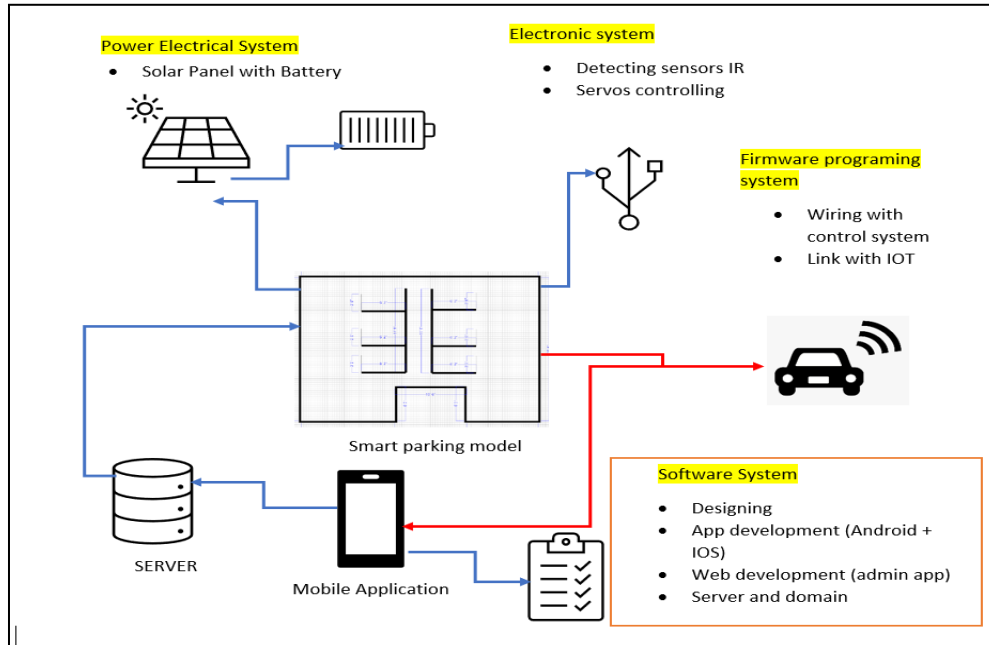


Figure 36 System Architecture

As shown in figure 3.17, the developed architecture of this project is based on the Internet of Things and solar energy. It outperforms the existing systems by releasing new and refined features to the parking reservation market. The system is supported by cloud-based server that is responsible of computations, communication layer and authentication services. Also, the usage of solar panels generates clean energy to the parking model in order to lessen the issues arising from the usage of finite resources and contribute to have an ecofriendly society. The first subsystem to be focused on is the electrical circuit that has a solar panel of size 48 by 34 centimeter providing 18 Volts. The parking prototype size of the project requires 12 Volts which must be taken in consideration. Because of that, the voltage will be supplied by a 12 Volts battery. The battery stores excess power coming from the solar panel that is the energy source for parking model. The electronic subsystem has the microcontrollers of the prototype. As mentioned, the microcontrollers can perform tasks related to reading inputs from the surroundings and returning outputs. For example, the microcontroller of the Smart Parking Reservation System can open and close the gates depending on the instructions. The software will be linked to the parking prototype through the communication layer in order to interact with the system. Another point to consider is that the parking model may have several microcontrollers boards, so the system does not depend on only one. Having more than microcontroller board to is critical to protect the system from an entire

collapse in case of errors or damages. It guarantees that if one board stops working, the others will keep operating which makes the system practical to be implemented in public places.

### 3.6.3 Cost Analysis

The entire cost of each component utilized in the project is shown in table 13 below. Aside from that, there is a budget for creating the prototype structure that was built by a carpenter. The following is how the total cost of the project was calculated:

Table 13 *The Budget of the Project*

No.	Component	Quantity	Cost - KD
1	Ebot 8 Pro	2	30 KD
2	EBot IoT Board	1	16 KD
3	EBot LED RGB	6	9 KD
4	Micro Servo	7	12.250 KD
5	EBot IR Sensor	6	9.000 KD
6	3D Printer Services	1	35.000 KD
7	Solar Panel (18V)	1	9 KD
8	Wires	20	15 KD
9	Charge Controller	1	7.000KD
10	Battery (12V)	1	8.5 KD
11	Mini Toy Cars	2	7.950 KD
12	Building the Prototype (Carpenter)	1	50 KD
<b>Total</b>			<b>3.700</b>

### 3.7 Ethics and Limitations

Engineers of this project have trusted the standards required to match the demand of the scope. The group members work professionally to enhance harmony, dignity, integrity, and

responsibility to successfully reach the goal of the project. The project is ecofriendly as it is based on solar energy that has a potential to be an alternative to other energy resources. Thus, it ensures that the project takes a part in saving the environment, saving people's health, and reduces costs. Also, the group members worked to provide safety for the users from all aspects including health and security. The project eliminated any unethical usage of users' personal information. Referring to Literature Review Chapter, the Smart Parking Reservation System avoided the use of faces and plates scanning. Also, it eliminated some features that may not be useful for a segment of the society since some users find it difficult to deal with applications. Therefore, the team ensured to make the application as simple as possible.

However, the usage of applications may not be useful to some clients with different age group and countries. Smart Parking Reservation System's application may be an advanced technology to some elderly living in the scope of the project. In addition, some countries may have unavailability and weak coverage of internet connections. On the other, the system is dependent on solar energy that can be affected by the weather. Thus, the team must brainstorm an alternative solution in such rare cases. However, the scope that the project is studying has a sunny weather and is considered a developed country to implement technology-based applications. Therefore, the expectations and needs of users in this scope are satisfied.

### **3.8 Conclusion**

In conclusion, this chapter discussed the waterfall method, the design alternatives, purchased software and hardware components, analysis of the system, and the ethics and limitations. It included block diagrams that are needed to clarify the functionality of the Smart Parking Reservation System. The functionality of each component in the system has been illustrated. In addition, this chapter provided the total budget for the system to be implemented.



# CHAPTER FOUR IMPLEMENTAITON

## CHAPTER 4: IMPLEMENTATION

### 4.1 Introduction

This chapter describes the technical parts contributed to the development of the Smart Parking Reservation System. It also includes the problem faced while implementing the prototype. Besides, it discusses the components, programming languages, connections and protocols that are utilized to build the hardware and software of the project. In addition, it comprises the IEEE standard related to the system. Chapter 4 required testing and correction to arrive at the result of the software and hardware.

### 4.2 Hardware Implementation

This section sheds the light on the process of building and implementing the hardware based on the selected final design that was discussed in Chapter 3. Also, it describes the process of developing the electronic circuit and power circuit. In addition, it includes the methods that were used to employ the prototype such as the programming language. The following Figure 37 shows the stages that were applied while building the prototype.

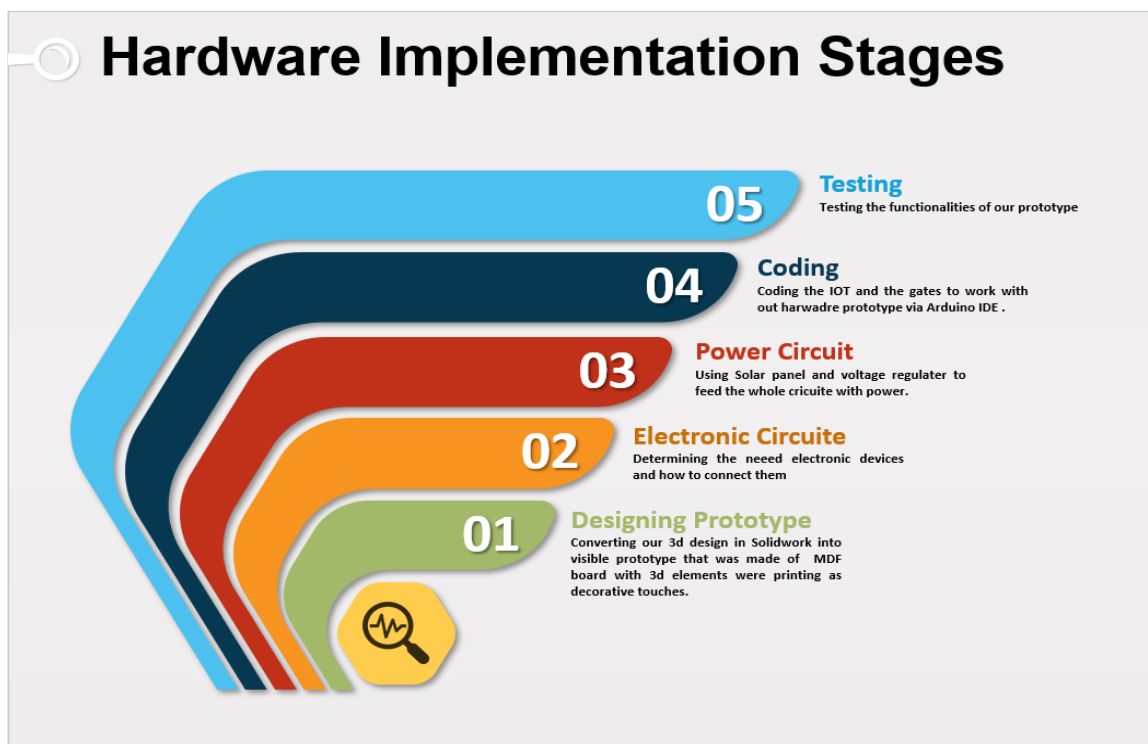


Figure 37 *Hardware Implementation Stages*

### **4.2.1 An Overview of How the Prototype Works**

Before explaining the hardware implementation step by step, it is important to understand the operation behind this prototype. The users can download “Smart Parking Reservation System” software on their smartphone. This software can give the drivers visual layout of the available parking lots in a specific location. Once the users book their preferred spots, the states indicated by the LED is going to change from green (available) to yellow (booked). After arriving, the drivers can use the IoT technology through their phone to open the main gate. Afterwards, the users can open their booked parking spots’ gates. After parking, the states change to be red (parked). When users decide to leave, they must exit through the mobile application by paying the fees through the smart wallet. Moreover, the Smart Parking Reservation System is supplied by a solar panel that generates energy to be stored in the battery. They are connected to a voltage regulator to regulate the energy based on the system’s needs.

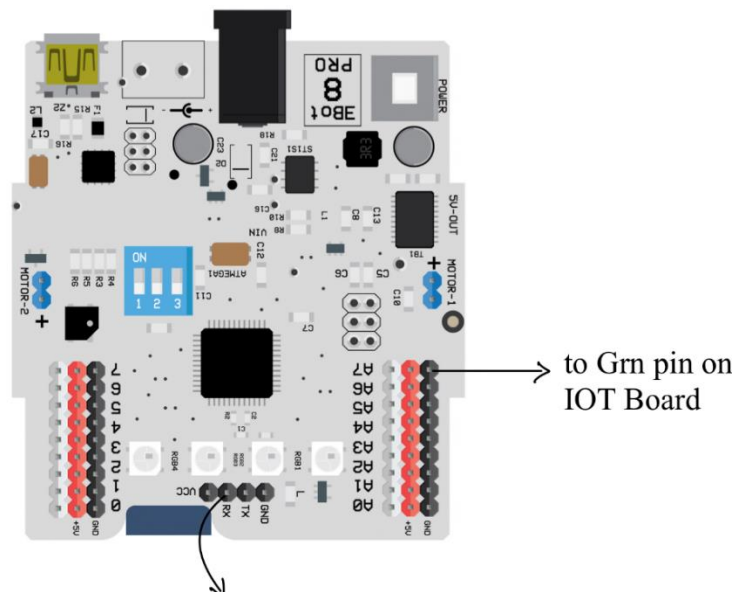
### **4.2.2 Electronic Circuit**

#### **4.2.2.1 IoT Board Connection**

Initially, the electronic circuit is needed to build the connections. In this prototype there was no need for the Wi-Fi programming as the electronic component EBot IoT board supports wireless connectivity by default. EBot IoT board receives the information coming from the Smart Parking Reservation System software and passes the signals to two EBot 8 Pro microcontrollers. The IoT board and the EBot microcontrollers were tested which helped in developing the project. Testing can be done even before developing the system via Ebot Blockly as shown in figure 38. It can be tested through a coding interface with visuals and electronic components. Thus, it helped to test the connection for the purpose of the initial selection of components. Blockly is a drag and drop visual programming tool that includes more than 75 components which are built-in to be applied. Each block is generated in real time coding to provide visual representation. Features help to avoid testing failure that might lead components to burn and cause the circuit to break. Overall, looking at available options of testing the blocks tool saves time as it assists to decide on the suitable components.

In this prototype, the two microcontrollers are connected in series. Also, the IoT board is connected in parallel with one of the two microcontrollers. As shown in figure 39, from the IoT board ‘Grn’ pin, the yellow wire is connected to the second microcontroller at the output pin A7 (ground) as clearly illustrated in figure 40. Then, pin 17 from the IoT board is wired with pin RX of the microcontroller.

Figure 38 Testing IoT Board Using Blockly



To Pin 17 on IOT Board

Figure 39 Connecting IoT Board to the First Microcontroller

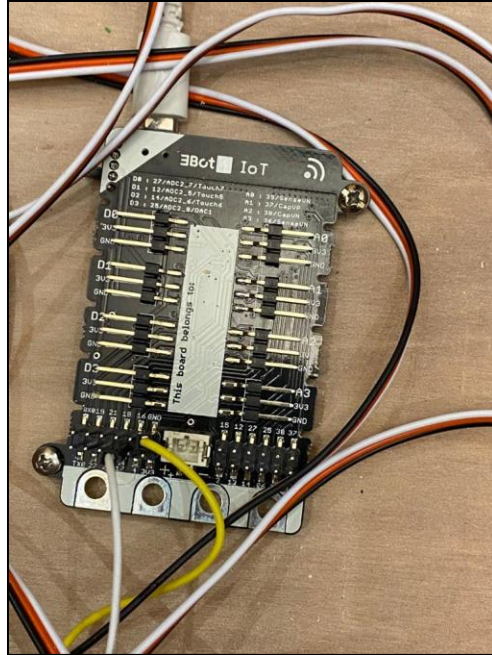


Figure 40 *EBot IoT Board Connection*

#### 4.2.1.1 The Components of One Gate:

The system has several gates as each parking spot has its own besides the main one. All gates include several electronic components which are RGB LEDs as inputs. They also have EBot IR sensors and micro servo motors as outputs. However, the main gate is excluded from having a sensor because it only has one micro servo for opening and closing the entrance. The components per one gate are listed as follows:

1. **RGB LED:** to show the status of a parking spot, this system has three status which are:
  - I. **Green:** Available
  - II. **Yellow:** Booked
  - III. **Red:** Occupied
2. **EBot IR sensor:** to detect the presence of a vehicle, so it sends signals to the RGB LED.
3. **Micro Servo Motor:** is a self-contained electrical device that rotates the gate's arm to control its opening and closing. The servo motor for the prototype was selected carefully based on the right speed to not affect the movement of each gate's arm.

#### 4.2.2.2 Connecting the gates to the microcontrollers

Each EBot microcontroller is responsible for a set of gates, in this project there are 7 gates in total which are divided into (A1, B1, A2, B2, A3, B3). The gates represent the parking spots and the main one. The first microcontroller is wired with the following gates (A1, A2, A3 and the main gate). The second microcontroller is connected to gates (B1, B2, and B3). Table 14 shows each pin connected to which component in the electronic circuit. As shown in figure 41, the micro servo wires have three colours, the brown from the ground, the red for voltage, and the yellow is for the signal that controls the angle of movement. The same approach is applied on the RGB LED and EBot IR sensor as they have three colours which are black, red, and white each indicating ground, voltage, and signal, respectively. Table 14 shows each component connected to which pin in the microcontrollers. Figure 42 shows the final connection of the two microcontrollers and the IoT board.

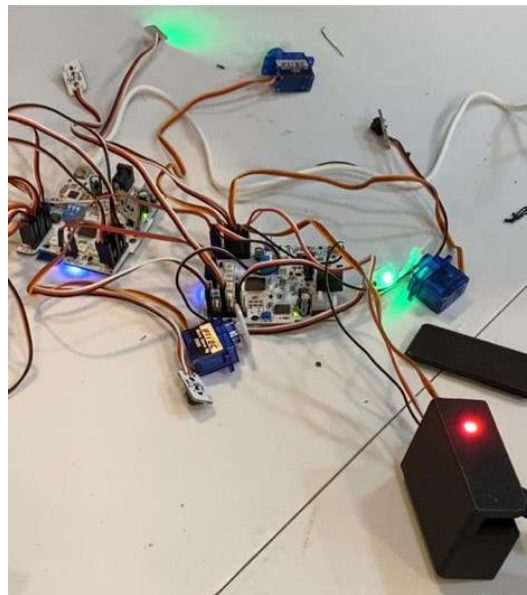


Figure 41 *Microcontrollers' Connections*

Table 14 *The Connection Between Each Component with Microcontrollers*

Microcontrollers	Component Name	Pin Number
<b>First Microcontroller</b> (controls B gates)	IR Sensor for gate B1 (input)	A0
	IR Sensor for gate B2 (input)	A3
	IR Sensor for gate B3 (input)	A6
	Micro Servo for gate B1 (output)	1
	Micro Servo for gate B2 (output)	4
	Micro Servo for gate B3 (output)	7
	RGB LED for gate B1 (output)	0
	RGB LED for gate B2 (output)	3
	RGB LED for gate B3 (output)	6
<b>Second Microcontroller</b> (controls A and Main gates)	IR Sensor for gate A1 (input)	A0
	IR Sensor for gate A2 (input)	A3
	IR Sensor for gate A3 (input)	A6
	Micro Servo for main gate (output)	5
	Micro Servo for gate A1 (output)	1
	Micro Servo for gate A2 (output)	4
	Micro Servo for gate A3 (output)	7
	RGB LED for gate A1 (output)	0
	RGB LED for gate A2 (output)	3
RGB LED for gate A3 (output)	6	

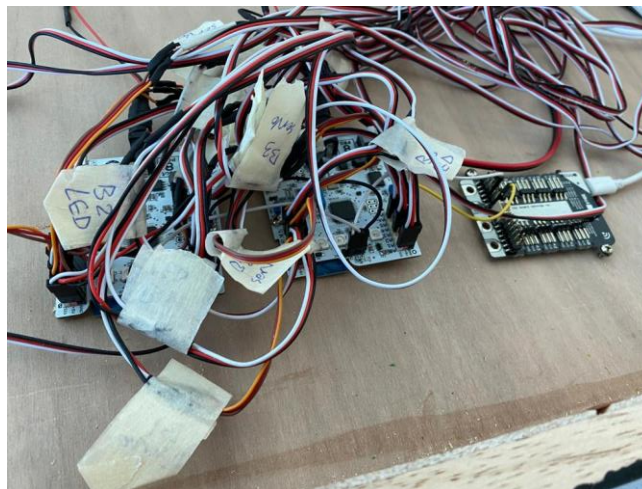


Figure 42 *The Connection of the IoT Board with the Two Microcontrollers*

#### 4.2.2.3 Equation for the Micro Servo

Maximum Movement Current = 250mA

Still Current = 10mA

- **In this prototype there are 7 servos, including 1 moving servo with 6 still servos:**

$$250 + (6 \times 10) = 310 \text{ mA}$$

Controller → Supplies up to 1A

#### 4.2.3 Power Circuit

The system needs to be supplied, but depending on batteries alone consumes power quickly. Thus, this project depends on clean energy that can charge the batteries. The solar panel is one of the main components to supply the Smart Parking Reservation System. As an initial step, the output of the solar panel is determined based on the system's needs as illustrated in Figure 43. The solar panel charges between 12-18 Volts, which is the one available on the market. Based on that, the battery and the charge controller are chosen to be of 12 Volts. As a result, the prototype functions at an input of 12 Volts. From table 7, all microcontrollers operate at inputs of 5 Volts. The solar panel is connected to the charge controller at the panel side having two wires including, the red wire representing the positive end and the black wire for the negative end as shown in figure 44. As shown in figure 45, The 12 Volts battery is connected to the charge controller at the battery side. The EBot IoT board has a USB port which is connected to the USB side of the charge controller. The charge controller passes 5 Volts to the IoT board to be powered with a maximum current of 0.5 Ampere. Also, the two microcontrollers are connected to the load side of the charge controller. The connection at the charge controller is shown in figure 46.

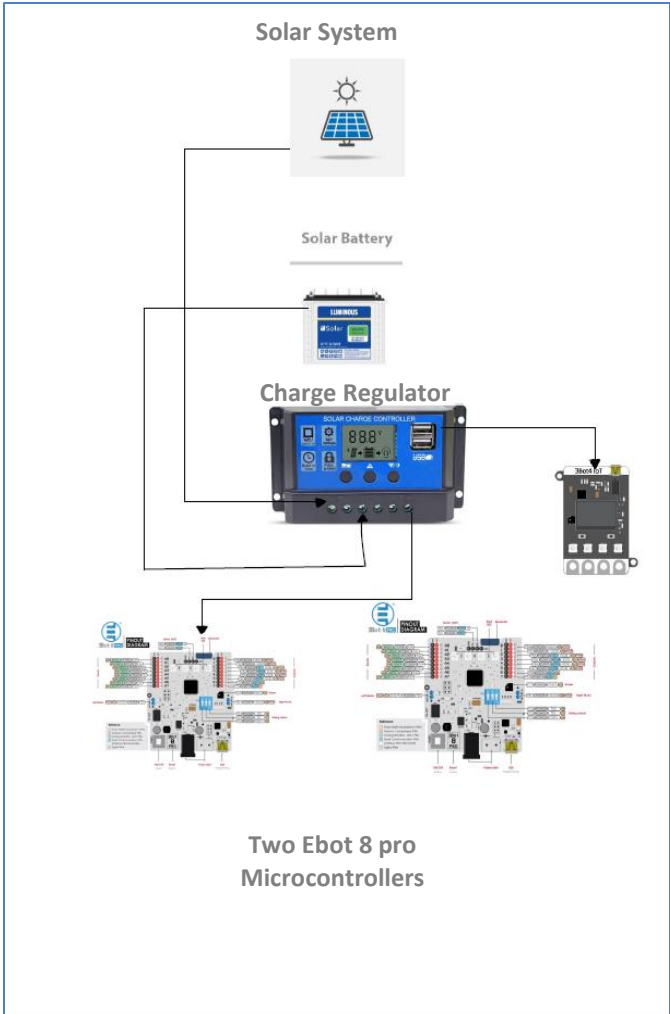


Figure 43 *Solar Power System Diagram*



Figure 44 *Back of the Solar Panel*



Figure 45 12V Battery Connected to the Charge Controller



Figure 46 The Four Inputs of the Charge Controller

- **Hardware Coding:**

The hardware components are coded using the C++ programming language. The hardware components programming is written based on the electronic circuit and the connection of the microcontrollers. In addition, the functionality of the Smart Parking Reservation System application is explained in the Software Implementation section.

#### 4.2.4 Programming the Two Microcontrollers

As mentioned, each EBot microcontroller has a set of gates. Since the Arduino is an open source, EBot 8 Pro is based on Arduino microcontroller that can be programmed via Arduino IDE using C++ language. Like any language, the libraries need to be defined as an initial step in writing programs. The defined libraries are determined by acknowledging the electronic components of the project such as the micro servo, EBot microcontroller, RGB and IR sensor. Table 14 indicates the pins connected to each component which is required in programming the microcontrollers.

The microcontroller that controls the ‘A’ gates is programmed with the main gate. The pins must be defined as inputs and outputs. The initial set up is defined as a void function which means once the system resets, there is no need to define the values every time it runs. The setup includes the following:

- The initial values of three RGB lights to be green as available parking spots.
- The four servos including the main gate should be initially closed by setting up their positions.

##### 4.2.4.1 Methods:

The first method to be defined is the open method that depends on the IR sensor range to sense the car existence. Since the testing is done using toy cars, an if statement is used to test two conditions. It checks the first time when the range is between 0 to 70. In this condition, the RGB will remain green unless the value exceed that range thus the RGB turns red. Shown below, the written code as follows:

```
void open1() { // method to open A1
  myservo1.write(0); // open position
  if (ir(A0) >= 0 && ir(A0) <= 70) //o to 70 this is range of the IR sensor between the sensor
  and the car height .
  {
    while (!(ir(A0) >= 200 && ir(A0) <= 1023)) { //infinite loop until the condition is achieved
      alarm();
    }
    strip0.setPixelColor(0, strip0.Color(0, 255, 0)); // green
  }
}
```

```

strip0.show();
}
else
{
while (!(ir(A0) >= 0 && ir(A0) <= 100)) {
alarm();
}
strip0.setPixelColor(0, strip0.Color(255, 0, 0)); // red
strip0.show();
}
delay(2000); //Delay for 0 hours 0 mins 3 sec 0 ms
myservo1.write(100); // close position
}

```

The same method is applied for the rest of the gates as open1, open2, open3 and open main. Then, it updates the RGB status by checking each gate's condition of the assigned values which represent the indication. If it is booked, available and occupied it changes the status to yellow, green, red respectively depending on the condition. In addition, the main loop for this code is programmed based on the system's functionality.

First, the microcontroller checks the received serial from the IoT board that contains each character index. If the index (0) starts with '#' character, then it checks the two possible cases of the serial number. To illustrate, the '#' character is added, so that the system recognizes the incoming serials from the application. This is applied to ensure safety and reduce errors. Also, to only accept sentences starting with an '#' character. The two following possible cases after the '#' character can be either 'A' to open or close the gate, or 'S' to indicate the status of the LEDs. In the status checking, another for loop is required in order to iterate 12 times. It is programmed this way since the first index (0) is only specified for the '#' character and index (1) is reserved for 'S' or 'A'. Thus, the loop will iterate only on the remaining characters.

### Example of serial number received from IoT

#	S	1	A	2	B	3	0	4	A	5	B	6	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---

The next method checks the gate and its current status. If the serial starts with '#' followed by 'S', it is directed to changing the status function. The first number indicates which gate, and the following character is the status. For example, if the serial contains "1A" then it means change status of gate 1 to available. Also, characters 'B' and 'O' indicate booked and occupied states. In addition, if the serial starts with '#' followed by A then it is going to check which gate to open or close from the four gates that are defined previously. Similarly, the code is applied for the second microcontroller except for defining the IR, LEDs, and servos for the B's gates.

```
void loop() {
  if (Serial.available() > 0) {      // if we recieved serial from IOT which >0 then begin
    text = Serial.readString();      //store this value in variable named text of type string
    if (text.charAt(0) == '#') {     //charAt will check the index of each character in a string

      if (text.charAt(1) == 'A') {
        if (text.charAt(2) == '4') {
          openMain() ;
        }
        else if (text.charAt(2) == '1') {
          open1() ;
        }
        else if (text.charAt(2) == '2') {
          open2();
        }
        else if (text.charAt(2) == '3') {
          open3();
        }
      }
      else if (text.charAt(1) == 'S') {
        for ( int i = 2 ; i < 14 ; i = i + 2 ) {

          updateStatusLEDs(text.charAt(i), text.charAt(i + 1)); // (int gateNumber, char gateStatus)
        }
      }
    }
  }
}
```

Figure 47 *The Main Loop for Status and Gates Control*

```

one|Arduino 1.8.11
File Edit Sketch Tools Help
one
#include <Servo.h> //servo library
#include "Ebot.h" // ebot library
#include "Adafruit_MePixel.h" // to define LEDs
#define IR(x) analogRead(x) //defining ir sensor in variable called x as an analog input
string task = "";
Adafruit_MePixel strip1 = Adafruit_MePixel(4, 11, NEO_GRB + NEO_K8800); //this is built in RGB
Adafruit_MePixel strip0 = Adafruit_MePixel(1, 0, NEO_GRB + NEO_K8800); //RGB 1 takes pin 0
Adafruit_MePixel strip3 = Adafruit_MePixel(1, 3, NEO_GRB + NEO_K8800); //RGB 2 takes pin 3
Adafruit_MePixel strip4 = Adafruit_MePixel(1, 4, NEO_GRB + NEO_K8800); //RGB 3 takes pin 4

Servo myservo1; // defining each servo here we will work on the first 3 servo for each gate including the main gate
Servo myservo2;
Servo myservo3;
Servo myservo4;

void setup() // the initial program set up where values of functions are defined only once.
{ ebot_setup(); // this is the function of ebot set up since our ebot has many built in function here we can control some of these function
  Serial.begin(115200); // this is to enable the serial communication that will be taken from IOT

  myservo1.attach(1); // servo pin for gate A1
  myservo2.attach(4); // servo pin for gate A2
  myservo3.attach(7); // servo pin for gate A3

  myservo4.attach(5); // servo pin for main gate

  pinMode(0, OUTPUT); // LED for gate A1
  pinMode(3, OUTPUT); // LED for gate A2
  pinMode(4, OUTPUT); // LED for gate A3

  pinMode(A0, INPUT); // IR for gate A1
  pinMode(A3, INPUT); // IR for gate A2
  pinMode(A4, INPUT); // IR for gate A3

  // in this blok of code we will set up the RGB first
  strip0.begin();
  strip0.setPixelColor(0, strip0.Color(0, 255, 0)); // for available park spot we want to make sure that the system starts with green which value =255 as
  strip0.show();

  strip3.begin();
  strip3.setPixelColor(0, strip3.Color(0, 255, 0));
  strip3.show();
}

```

Figure 48 Sample of EBOT 8 pro C++ Code

#### 4.2.5 Final Implementation of the Prototype

Initially, the idea of design was created as shown in figure 49. Figure 50 shows the final implementation of the project. There are six gates beside the main gate attached to the wooden box. Also, six sensors can be seen in each parking space. The design of the project is meant to be ecofriendly which is the reason of choosing trees, sprinklers, and a small lake to represent a clean energy project. All the connections are hidden inside the wooden box.

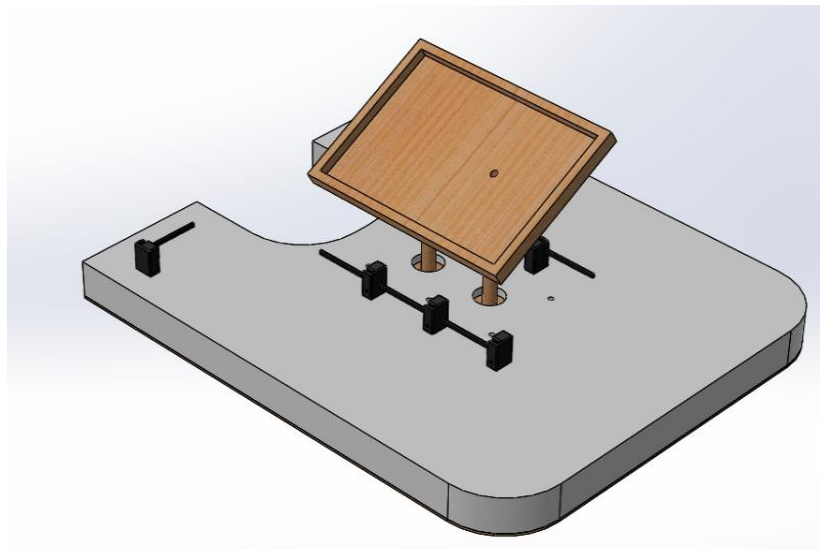


Figure 49 Initial Design

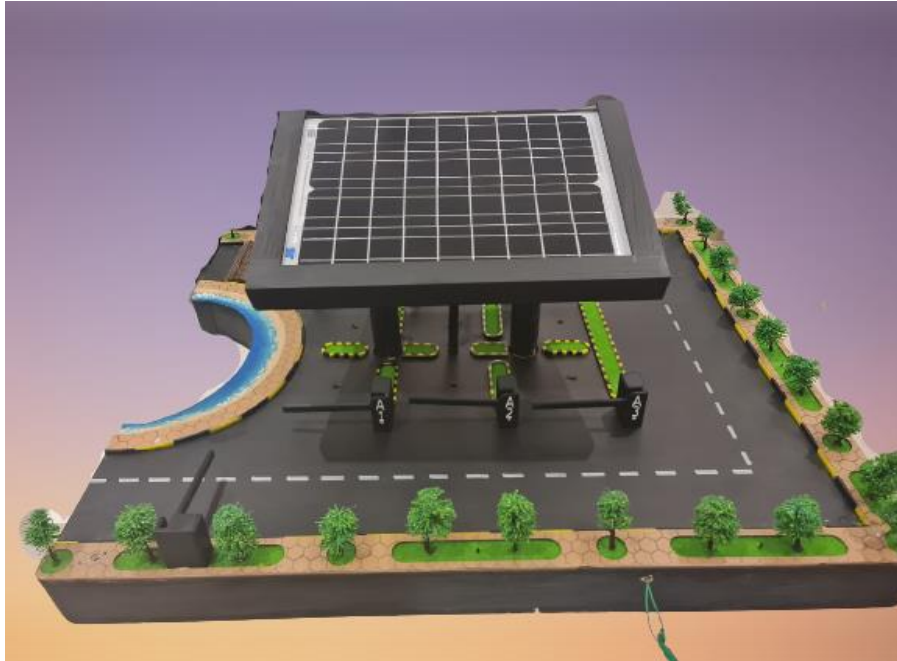


Figure 50 *Smart Parking Reservation System Prototype*

### **4.3 Software Implementation**

Mobile App Flutter is a hybrid framework used to build the Smart Parking Reservation System application. It allows to build native mobile application that can be downloaded directly to the device. It uses only one codebase and programming languages to create applications for different devices. It can be applied for multiple platforms such as Android, iOS, and Windows thus it is applicable for many devices. The Flutter is programmed using Dart that is an object-oriented language. Thus, the Flutter compiles the Dart program to create the user interface of the Smart Parking Reservation System. It has a framework which is an UI library that has text inputs, menu, scroll bars, and etc. In Smart Parking Reservation System, the application is built using libraries to give it a user-friendly interface. Thus, Flutter is utilized to represent the frontend of the Smart Parking Reservation System application. However, the backend of the application deals with the server using an intermediate called Node js.

The backend of the system consists of the Node js and the server. As mentioned before, The Smart Parking Reservation System uses MQTT (The Message Queuing Telemetry Transport) communication protocol to communicate with the IoT EBot board. MQTT can transfer a command to control an output or read information from a sensor to publish data. MQTT is a publish/subscribe protocol to transfer and receive messages. In this protocol, the devices must subscribe

to topics in order to receive the published information. It allows the devices to publish to the broker that can receive, filter, and publish the messages to the subscribers. In the Smart Parking Reservation System, the ESP32 IoT board is subscribed to a topic which receives the messages and controls the system. Since MQTT is a publish/ subscribe protocol it gathers more data with less network bandwidth which is suitable for sensor-based applications. In addition, the users of the Smart Parking Reservation System have to open and close the gates using the application; therefore, it is excellent option for controlling. The connectivity between the application and IoT system requires an API (Application Programming Interface). It connects the IoT or any other device from a mobile application. Node js is applied as a server programming language to retrieve, update, and remove the information. Any request will go to API and come back with a response. For example, the information provided when signing in into the Smart Parking Reservation System is checked in the server. Another example, booking parking slots sends request to server that updates the state and responds with confirmation. Such processes are done via the Node js that is an intermediate between the application and IoT system. In the Node js, the Adafruit IO protocol is utilized to communicate between the IoT device and the API. The Adafruit IO provides client services that supports MQTT communication protocol. The broker in the Smart Parking Reservation System is Adafruit IO which is the medium between the Application and the IoT. The MQTT protocol is connected to IoT system directly. Thus, these protocols were used to link between the IoT system and the application. To illustrate, after clicking the button for booking a slot, the process will go to API that is the Node js that will be connected to Adafruit IO. The Adafruit IO transmits the notifications representing the subscription. The Adafruit IO sends to the MQTT that is directly connected to IoT device that throws sentences to the system. Then, any process can be performed such as opening and closing the parking's gates. Moreover, in the Smart Parking Reservation System, the IoT device is connected to Wi-Fi. The 802.11 is the IEEE standard for communicating through wireless representing Wi-Fi. The Wi-Fi link protocol has an IP address of IPv4 offering 32-bit addresses.

In addition, the cloud for the application comprises the admin panel. Shown below, the admin panel that has the booking information including names, phone number, price, total cost, parking area, slot names, start dates and end dates. It supports adding parking areas to the application. The admin can specify name, description, price, and map for parking areas. Furthermore, the admin panel has an option to add parking slots to the areas created that appears

directly in the application. Thus, the admin panel represents the cloud that stores the data of the users and allows to control the application. The admin panel is implemented for the authentication and authorization purposes. The cloud -based server of the Parking Reservation System is Amazon Web Server that is designed with the usage of Vue js and Node js. Vue js is a user interface framework that has JavaScript as its main programming language. The Node js connects between application and admin panel that represents the backend of the admin panel. In instance, if a user is booking a slot through the application, the request goes to server directly that will store the information. In addition, the storage of the Smart Parking Reservation System is the MongoDB database. For example, if the user books a slot, it must be saved in particular location which is the MongoDB database. This database is used in the system’s application and admin panel to store the data. MongoDB is a NoSQL database which is compatible with Node js. MongoDB database is chosen because it allows to design from the code itself unlike the SQL that requires structures and tables. Also, In the Smart Parking Reservation System users have multiple bookings that is more efficiently collected by the MongoDB database.

### **Controlling the Admin Panel:**

- The admin has to sign into the smart parking reservation system panel:

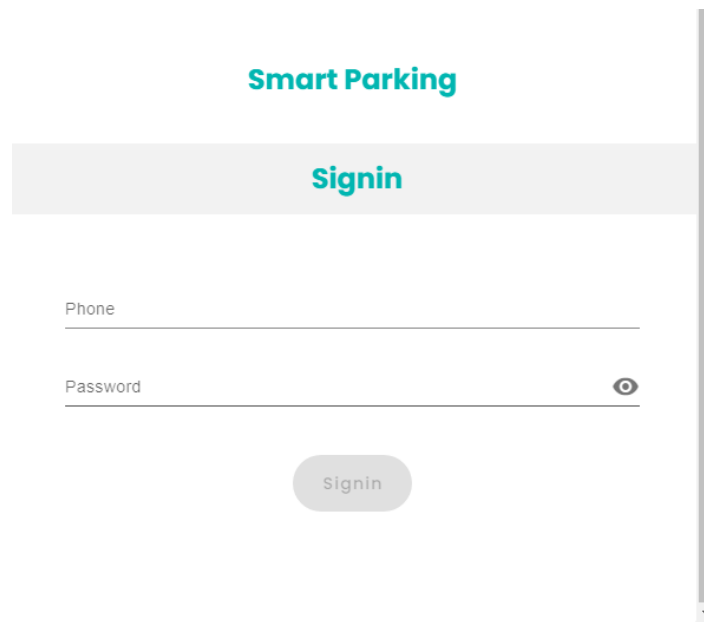


Figure 51 Admin Panel Sign in

- The admin can access the cloud data that is related the parking system as shown below:

Smart Parking Admin

Bookings Parking Areas

### Bookings (115)

#	Name	Email	Phone	Price	Total Cost	Parking area name	Parking slot name	Start date time	End date time
1	ashwaq		97728999	0.2	24	Avenues Phase 4	A1	31/05/2021, 1:29 pm	31/05/2021, 3:29 pm
2	eslam5		99245425	0.2	0	Avenues Phase 4	A2	29/05/2021, 4:26 pm	29/05/2021, 4:26 pm
3	ashwaq		97728999	0.2	0	Avenues Phase 4	A1	25/05/2021, 8:17 pm	25/05/2021, 8:19 pm
4	ashwaq		97728999	0.2	0	Avenues Phase 4	A2	25/05/2021, 8:16 pm	25/05/2021, 8:16 pm
5	ashwaq		97728999	0.2	0	Avenues Phase 4	A1	25/05/2021, 8:15 pm	25/05/2021, 8:15 pm
6	ashwaq		97728999	0.2	0	Avenues Phase 4	A2	25/05/2021, 8:08 pm	25/05/2021, 8:13 pm
7	ashwaq		97728999	0.2	0	Avenues Phase 4	A1	25/05/2021, 8:05 pm	25/05/2021, 8:07 pm
8	ashwaq		97728999	0.2	0	Avenues Phase 4	A1	25/05/2021, 8:03 pm	25/05/2021, 8:04 pm
9	ashwaq		97728999	0.2	0	Avenues Phase 4	A1	25/05/2021, 7:57 pm	25/05/2021, 7:59 pm
10	ashwaq		97728999	0.2	0	Avenues Phase 4	A1	25/05/2021, 7:55 pm	25/05/2021, 7:57 pm

Rows per page: 10 1-10 of 115 < >

Figure 52 Data of the Cloud

- From the parking area page, the admin can add more spots by clicking on the sign under “add parking slot” column.

Smart Parking Admin

Bookings Parking Areas

[+ Add parking area](#)

### Parking areas

#	Name	Description	Price	Map Uri	isActive?	Parking Slots	Add parking slot
1	Avenues Phase 4	Avenues mall parking area	0.2	<a href="https://goo.gl/maps/cmufb5zCIP5BZH9">https://goo.gl/maps/cmufb5zCIP5BZH9</a>	<input checked="" type="checkbox"/>	B3,A3,B2,A2,B1,A1	<a href="#">+</a>
2	360 Mall	Jassem Mohamed Al Kharafi Rd Block 7, Al Zahraa South Surra	0.2	<a href="https://goo.gl/maps/yhiGGnuNushgR2xx8">https://goo.gl/maps/yhiGGnuNushgR2xx8</a>	<input checked="" type="checkbox"/>	A1	<a href="#">+</a>

Rows per page: 1-2 of 2 < >

Figure 53 Parking Area Page

Figure 54 Add New Slot Feature

- Admin can add new place by clicking on add parking area and filling the information required:

Figure 55 Add New Parking Area Feature

- The admin profile can be edit by clicking on “admin” button. Also, to sign out from the admin panel:

Figure 56 Admin's Profile Edit

- Admin can update password as shown below:

## Update Password

Current Password	👁
Password	👁
Confirm Password	👁

Figure 57 Admin's Password Update

Visual Studio Code is chosen as an editor because it supports different computer programs used to implement the cloud-based application of the Smart Parking Reservation System. It can run, debug, control and show extensions of the codes.

The Dart dependency has a vs code that comes automatically when opening the application. It is not necessary to modify the Android and iOS packages. Pubspec.yaml is the main model that comprises the packages, images, and logo. "Flutter\_svg", for example, is a package that is used in the application for SVG files. SVG is an image format that uses a vector for graphical dimensions. As a result, they are built-in functions that are utilized in the development of the system's application.

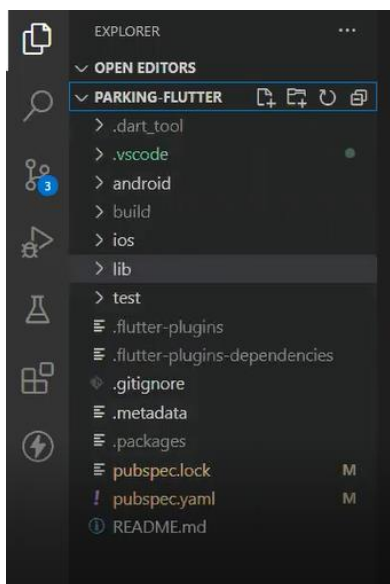


Figure 58 Application Files

The library folder, on the other hand, is where the application's core, orders, parking, payments, theme, and user programming are stored and modified. The library has the user interface, functionality, and connections of the system. The core has the main connection of the

application and API. Also, it includes the assets which are the images and languages used in the application. The helpers class functions are implemented for operations in the system. In instance, the validate class takes the email address as a string input. The validation process is used for signing in into the application, thus if the email addresses match it will allow the users to log in. If the emails do not match it will give a warning “Invalid email” which is found under the English language in assets.

```
5 class Validate {
6     static String email(String value) {
7         if (value.trim().isEmpty) return translation.of("required");
8         final RegExp nameExp = RegExp(r'^\w+@[a-zA-Z0-9_]+?\.[a-zA-Z]{2,3}$');
9         if (!nameExp.hasMatch(value)) {
10            return translation.of("invalid_email");
11        }
12        return null;
13    }
14 }
```

Figure 59 Validation Process

To begin booking, the users must have a balance in their smart wallet. The payment method checks if there is enough balance in the wallet. If the amount is low, the users will see an error notice stating that they must top up in order to proceed. The proceed process has a submit forum method that sends the amount, email, name, phone number, and redirect Uri parameters to the gateway. Thus, this function reviews the information entered to determine whether or not the payment was successful. If the information is incorrect, an error will be displayed; otherwise, the successful payment message will be displayed on the Knet Page. If the balance is sufficient, the users will be able to book without the need to recharge the wallet. The parking places page will be provided for the users after the system has checked the wallet balance.

```

children: [
  Text("Wallet Topup",
    style: Theme.of(context).textTheme.subtitle1), // Text
  widget.isBalanceLow
    ? Text(
      "Your Balance is too low. Please topup to proceed",
      style: Theme.of(context).textTheme.headline6.merge(
        TextStyle(
          color: Colors.red,
        ), // TextStyle
      ),
      textAlign: TextAlign.center,
    ) // Text
]

```

Figure 60 Low Wallet Balance

```

: Container(),
  SizedBox(height: 20),
  _amountTextField(),
  SizedBox(height: 40),
  ButtonMediumOutlined(
    fillColor: Palette.SECONDARY,
    focusNode: _btnFocus,
    buttonText: "PROCEED TO PAY",
    onTap: () {
      _submitForm();
    }
  )
)

```

Figure 61 Proceed Process

```

void _submitForm() {
  if (_formKey.currentState.validate()) {
    // Goto payment screen
    final PaymentInput paymentInput = PaymentInput(
      amount: double.parse(_amountController.text ?? "0"),
      // date: DateTimeHelper.getCurrentDateTimeString(),
      email: "unknown",
      name: authentication.authenticatedUser?.user?.name ?? "Guest",
      phone: authentication.authenticatedUser?.user?.phone ?? "Guest",
      redirectUri: "https://makemyflow.io/redirect/",
    );
  }
}

```

Figure 62 Submit Form

The data of parking areas comes from the API. The application sends sentences in the parking card part to the hardware of the system to control spots. The hardware will execute the

sentences received from the application. If parking area is equal to null or zero, a text will appear that there are no parking spots available. The parking card contains the list of areas offered that have parking lots. If there are available spots it will go to the list of the parking lots. The parking card class has the “book now” button that is directed to the list of parking slots. A message will appear if there are no parking spots available; else, it will go to building slots. If there are more than zero parking spots available, it will be moved to the building slots. Build slots functionality adds the availability, status, and name to the parking spots list. For example, the available slot takes a value of true to indicate the availability. The building slots has “map” function running over multiple spots as it checks the availability to give names and states. The list appears for the users when choosing the preferred spot. When a slot is selected a check circle icon will appear and turn the state to yellow indicating that it is booked. The confirm booking will be displayed when the user books the slot. The confirm booking class collects the parking price, wallet amount, and personal details. After collecting the information, the application will move directly to the payment. If the balance is low a dialog box will ask users to recharge in order to progress.

```
@override
Widget build(BuildContext context) {
  if (widget.parkingAreas == null) return Text('No parkings available');
  if (widget.parkingAreas.length == 0)
    return Text('No parkings available');
  else
    return Column(
      children: widget.parkingAreas
        .map((parking) => ParkingCard(parking: parking))
        .toList()); // Column
}
```

Figure 63 *Main Functionality of the System*

```
ButtonSmall(
  buttonText: "BOOK NOW",
  color: Palette.SECONDARY.withOpacity(_isAvailable ? 1 : 0.35),
  onPressed: () {
    if (_isAvailable) {
      showDialog(
        context: context,
        builder: (_) {
          return ConfirmBookingDialog(parking: widget.parking);
        },
      );
    }
  },
);
```

Figure 64 *Booking from Area*

```

    SizedBox(height: 12),
    Padding(
      padding: const EdgeInsets.symmetric(horizontal: 44.0),
      child: _pakingSlots.length > 0
        ? Wrap(
            spacing: 24,
            runSpacing: 12,
            children: buildSlots(),
          ) // Wrap
        : Text('No slots available'),
  ),
)

```

Figure 65 Checks Up the Slots

```

buildSlots() {
  int index = 0;
  return _pakingSlots.map((slot) {
    index++;
    return Stack(
      alignment: index % 2 != 0 ? Alignment.topLeft : Alignment.topRight,
      children: [
        InkWell(
          onTap: () {
            if (slot.status == ParkingStatus.AVAILABLE) {
              setState(() {
                _selctedSlot = slot;
              });
            }
          },
        ),
      ],
    ),
  });
}

```

Figure 66 Builds Slots

```

    _selctedSlot == slot
      ? Padding(
          padding: const EdgeInsets.all(4.0),
          child: Icon(
            Icons.check_circle,
            size: 28,
            color: Palette.SECONDARY,
          ),
        ),
    ),
)

```

Figure 67 Check Circle Symbol for Selected Spot

```

class Palette {
  static const PRIMARY = Color(0xFFA63E92);
  static const SECONDARY = Color(0xFF00B6B1);
}

```

Figure 68 Yellow Color for Selected Spot

```

double _walletAmount =
  state.userProfile?.user?.walletAmount ?? 0.
if (_walletAmount >= _parkingPrice) {
  if (_selctedSlot != null &&
    _phoneController.text != null)
    app.setPersonalDetails([
      personalDetails: PersonalDetails(
        address: "",
        name: "guest",
        email: "",
        phone: _phoneController.text), // P
    ]);
}

```

Figure 69 Collects Details for Confirm Booking

```

_bookParkingBloc.add([
  CreateBookParkingEvent(
    phone: _phoneController.text,
    parkingAreaName:
      widget.parking?.name ?? "unknown",
    parkingSlot: _selctedSlot?.id ?? "",
    parkingSlotName:
      _selctedSlot?.name ?? "unknown",
    price: widget.parking?.price ?? 0.1,
  ), // CreateBookParkingEvent
]);
} else if (_walletAmount < _parkingPrice) {
  showDialog(
    context: context,
    builder: (_) {
      return WalletDialog(
        isBalanceLow: true,
      ); // WalletDialog
    }
  );
}

```

Figure 70 Payment and Valid Balance

### 4.3.1 Application

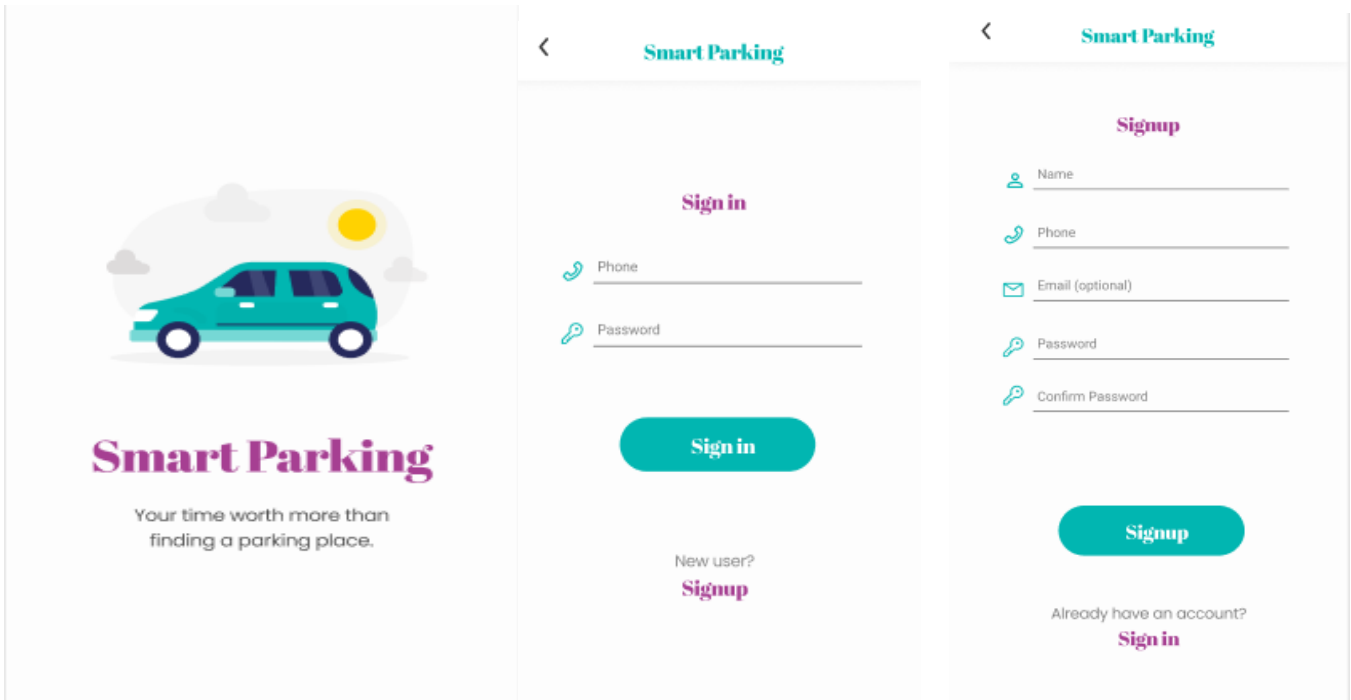


Figure 71 *Sign In and Sign Up of Application*

When users start the application, the logo will appear indicating that they are connected to the application. If they are new users, they can sign up to add their name, phone, email, and password. If the users are already subscribed, then they can proceed in just signing in into the application.

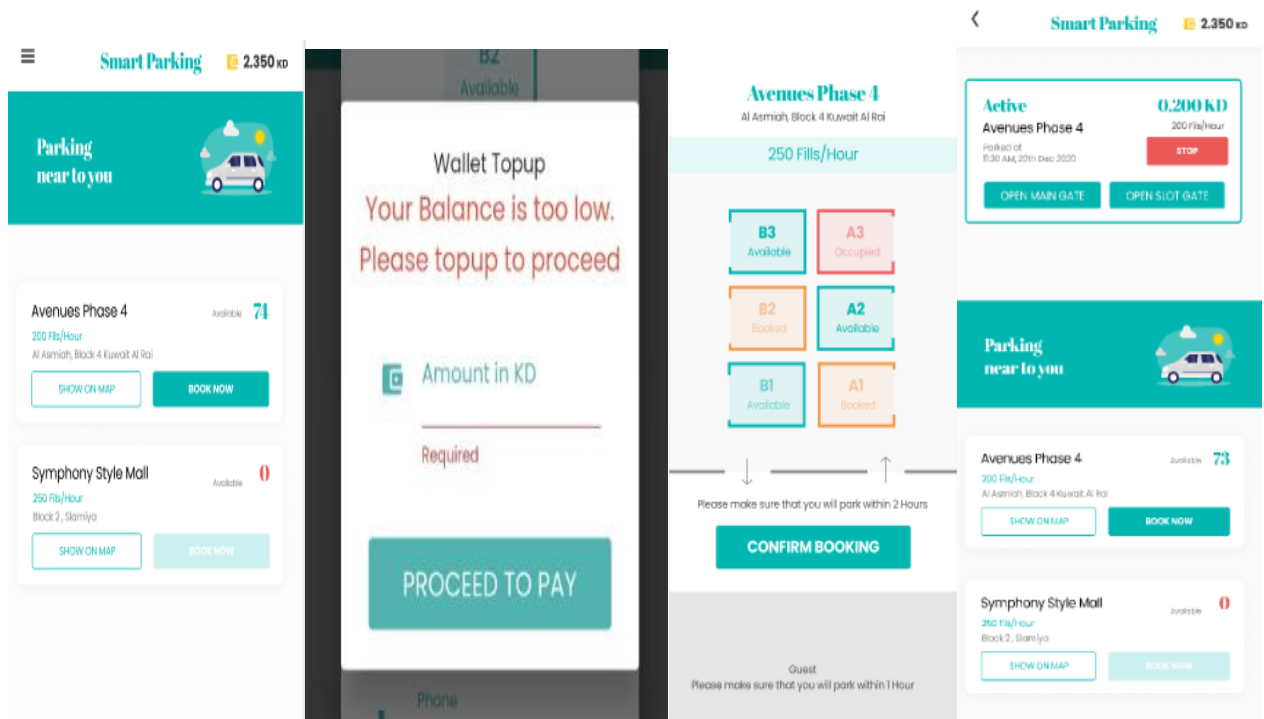


Figure 72 The Procedure of the Application

After logging in, the drivers can select the parking lot for their location. They can choose to show on map that locate the area or to book. The drivers must have a minimum of 200fils in their wallet to be able to book a slot. If there is insufficient balance a dialog box will appear asking users to recharge the wallet. The users can then look at the available, occupied, and reserved slots in the desired areas. They can select their preferred parking slot and confirm their reservation. However, the users must arrive and park within two hours of making the reservation. If the drivers are guests, they have one hour from the time of booking to arrive. As a result, subscribed users have access to more features than guests, which may motivate them to sign up for the application. The drivers have ten minutes to cancel the parking reservations. If they go beyond this time restriction, the application will take money from their wallet. Also, if they are thirty minutes late after two or one hour from the time of reservation it will be cancelled. From the time of booking, the drivers will be charged for each hour they are late. To enter the parking lot, the drivers must click the "Open Main Gate" button when they arrive. The drivers must then push the "Open Slot Gate" button open their reserved spot gate and park. At every step, the application is automatically updating the states and the number of available spots. When the "Stop" button is clicked it will be directed to the payment and the slots' gates will open to exit the parking lot.

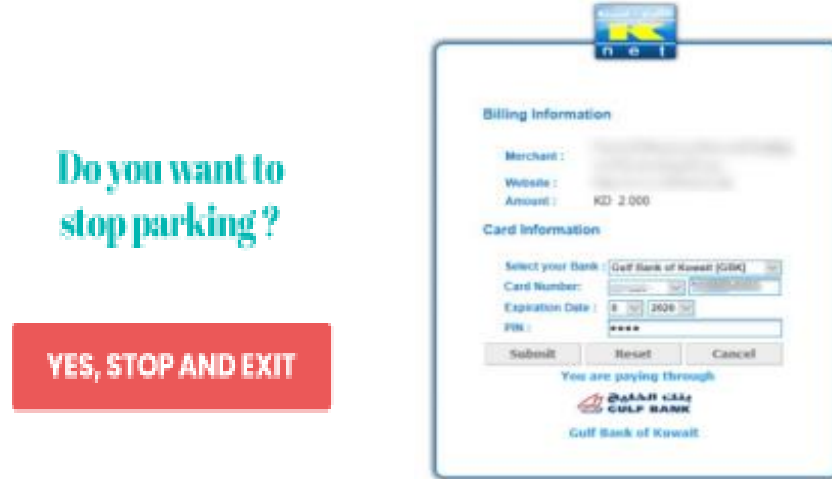


Figure 73 The Exit and Payment of the Application

When the drivers want to exit the parking lot, they must press "YES, STOP AND EXIT" button. The drivers must pay from their wallet that must be charged at the beginning of the booking process. The user gets charged 0.200 KD per hour when the parking is occupied.

- **Difficulties:** there are many difficulties faced in the project implementation including technical and runtime errors. The testing of servo motor required time as it was not responding. Also, the plan was to use two 6 Volts batteries, but instead 12 Volts battery was used. The application implementation took time to work properly as there was a confusion in the process of releasing the software to iOS and Android.

#### 4.4 IEEE Standards

The Smart Parking Reservation System used the IEEE Standard Wi-Fi (802.11).

#### 4.5 Conclusion

In conclusion, Chapter 4 covered the hardware and software development of the Smart Parking Reservation System. Technologies and techniques used to build the system were provided. This phase needed debug and testing in order to reach the final output of the software and hardware. Thus, it required many phases in order to optimize the functionality of the system. Also, the chapter includes the IEEE standard used to run the prototype.



# CHAPTER FIVE

# EVALUATION

## **CHAPTER 5: EVALUATION**

### **5.1 Introduction**

This Chapter discusses the relationship between the Smart Parking Reservation System and range of aspects. The chapter illustrated the impact of the project on several areas that includes the ethics and society, geography and environment, business, marketing, and economy. Also, it highlights the comparison between the Smart Parking Reservation System and other existing solutions from the Literature Review in a scientific approach.

### **5.2 Project Impact**

The Smart Parking Reservation System has a significant impact not only in the domains of electrical and computer engineering, but also in areas such as ethics and society, as well as commerce. The influence of the Smart Parking Reservation System on society, geography, the environment, business, marketing, and the economy will be discussed below.

#### **5.2.1 Ethics and Society Impact**

Smart Parking Reservation System has a social responsibility that recognizes the ethics of the targeted crowd. One of the factors to consider is that the parking system must adhere to the country's cultural norms, since some technology, such as face recognition, may be unwelcome. Also, some drivers find it unreliable to share their vehicles' plate numbers to an application. To gain customers' trust, the Smart Parking Reservation System intended to eliminate any facial or vehicle plate scanning. Furthermore, information that the user inputs into the Smart Parking Reservation System application is saved in the cloud and thus is authenticated.

People of all ages today use smart phones for a multitude of reason. The technological support in the system is crucial to different types of users. As a result, the Smart Parking Reservation System is designed to be a user-friendly application. Overall, the characteristics of this system can be implemented in any country. Thus, the system can be accepted by any cultural standard in different nations.

#### **5.2.2 Environmental and Geographical Impact**

One of the key goals of the Smart Parking Reservation System is to save the environment while also utilizing various geographic locations in the targeted region. Kuwait's climate provides the ability to rely on solar panels generating thermal energy. Kuwait, on the other hand, consumes

energy derived from finite resources. The conventional energy is contributing to the global pollution that is negatively affecting the balance of the ecosystem. Thus, the Smart Parking Reservation System takes a part in raising awareness of alternatives to power projects. As a result, this system participates in encouraging the use of the infinite solar energy that helps in reducing pollution. Kuwait has sandy areas that are used as parking spaces, which can cause traffic congestion. Additionally, drivers that park illegally in such areas may receive a penalty. Many malls and hospitals, on the other hand, do not have built-in parking. The Smart Parking Reservation System has the potential to have a geographical influence because of its capability to operate in any sort of location. This highlights how the difficulties that arise can be used to produce solutions by utilizing the Smart Parking Reservation System.

### **5.2.3 Business Impact**

Businesses can emerge as a result of the implementation of this project in the real world. Suppliers for various components of the Smart Parking Reservation System, such as solar panels and parking gates, will be required. In addition, for the project's continuity, maintenance specialized corporations are expected. For example, the solar panels must be cleaned in regular basis in order to operate efficiently. Developers are hired to help with the release of new software updates and bug fixes. Customer service representatives are also necessary for communicating with users in the event of difficulties or suggestions. A finance department is required to conduct feasibility studies in order for a project to be successful. This department is also critical for budgeting and profiting from the project. Engineers are employed to ensure that the project is up to date with the latest technologies. Marketing agents are also required for advertisement purposes.

### **5.2.4 Marketing Impact**

To reach people and make a difference, new company concepts require effective marketing. The Smart Parking Reservation System creates more room in the marketing world for people and businesses to promote fresh and innovative ideas. A project like this requires marketing to persuade property owners to develop smart parking for their facilities, such as malls and hospitals. Customers are drawn to social trends; thus, this system will focus on advertising on various social media platforms. The use of solar energy can be taken for granted to raise awareness and persuade users that the Smart parking Reservation System is a better option. Thus, using the application may make drivers feel as if they are participating in to saving the environment.

### 5.2.5 Economic Impact

The Smart Parking Reservation System has an economic impact in the targeted market. This system is based on solar energy which contributes to reducing electricity bills. In developed countries, there are many malls, hospitals, towers, universities, and other desired destinations which have crowded parking lots. In such places, people struggle to find a spot to park their cars. Therefore, implementing the Smart Parking Reservation System will make a profit out of its fees. As a result, properties owners will find this project desirable to cut off costs and take a percentage of the profit. Thus, this system will contribute to boosting the economy as it meets the market demand.

## 5.3 Comparison

This section scientifically compares the Smart Parking Reservation System with the similar projects from Literature Review in chapter 2. The comparison is made based on the technologies provided for the users. Thus, the comparison helps in evaluating this project.

Table 15 *Systems Comparison*

Application Features	“Smart Parking Reservation System”	“GetPass”	“Mawaqif”
Smart Wallet	✓	✓	✓
Reservation	✓	✓	✓
Advanced Parking	✓		
Spot Picking	✓		
Clean Energy	✓		
Scanning Process		✓	✓
Control Gates	✓		

## 5.4 Conclusion

To conclude, chapter 5 discusses the Smart Parking Reservation System Impact on different sectors. These sectors include ethics and society, geography and environment, business, marketing, and economic. Also, chapter 5 refers to the Literature Review section as it compares the Smart Parking Reservation System with the most similar existing systems. Thus, chapter 5 assists to evaluate the Smart Parking Reservation System.



# CHAPTER SIX

## CONCLUSION AND FUTURE WORK

## **CHAPTER 6: CONCLUSION AND FUTURE WORK**

### **6.1 Introduction**

Chapter 6 will tie all previous stages of the Smart Parking Reservation System development. The chapter will demonstrate the objectives that are met from the introduction chapter. Also, it comprises the future development section which is important to meet the market demand in the scope of the project. Future work states the development that can be made to improve the functionality and add features to the system.

### **6.2 Conclusion**

In conclusion, In Capstone I and II the Smart Parking Reservation System development required knowledge in engineering that was acquired throughout the undergraduate journey. Additionally, while building this prototype, various skills were developed, such as building a user-friendly application and modifying hardware connections to be more efficient. As a starting stage in developing the system, the problem definition, background, objective, scope, methodology, and SWOT analysis were presented in Chapter 1. Chapter 2 discusses existing solutions which include several techniques to resolving parking lot concerns. Moreover, Chapter 2 broadened the understanding of potential technologies and areas for improvement that aided in the development of the Smart Parking Reservation System. The methodology of the design was addressed in greater depth in Chapter 3 in order to demonstrate the execution phases. It explained the functional and non-functional requirements, alternatives, analysis, and budget of the project. Alternative components were investigated, and the project design was created as a result. Also, it addressed the ethics and limits that were considered throughout the design process. The implementation of the Smart Parking Reservation System's software and hardware was the subject of Chapter 4. It went over the coding and connections that were required to create the prototype. The relationship between the project and its field of application were considered in Chapter 5. It demonstrated the range of topics covered, including business, economics, marketing, ethics, social issues, the environment, and others. It also included a technical evaluation of the entire project. All of the preceding chapters were used to create the Smart Parking Reservation System. By introducing more features that satisfy market demand, the system has the potential to be improved.

### 6.3 Future Work

The main goal of the Smart Parking Reservation System is to address the issues raised in the problem definition in a timely manner. However, the following features can be added to meet market demand:

- 1. Lighting:** To provide parking lot lighting, lamps can be added to the system. The solar panels installed in the system can be used to power the lamps. They will turn on when it is dark to help drivers see clearly. There are various types of lamps available on the market that are ideal for various humidity levels. Thus, the lamps can be installed to indoors and outdoors parking lots.
- 2. Charger for Electric Cars:** In many developed countries, some drivers own electric cars that must be charged to operate. The parking lot may feature specialized areas with electric vehicle servicing equipment that provides power to the cars. The chargers will run on solar energy generated by the system's solar panels. A fee can also be charged to drivers who use this service.
- 3. Handicapped Parking Spots:** Parking spaces for handicapped people can be added near the entrances to the sites. The application can include spots that can be only booked by handicapped. To use this feature, the driver must give additional information. Confirmation from Ministry authorities is also required to design this functionality.
- 4. Increase the Size:** In practice, the number of solar panels in the system can be raised to construct an entire environmentally friendly building. It may also have additional parking levels. Solar energy will be used to power the entire structure. As a result, it will help to save the environment while also cutting the cost of electricity and reducing labour. Many developed countries have large shopping malls that require parking management. In Kuwait, for example, the Avenues mall has multiple entrances. People are forced to park further away from their intended entrance due to congestions in parking spaces, which causes them inconvenience. Thus, smart constructions are required to address such issues.
- 5. Connection Loss:** In the event that the user loses connection, the system sends an SMS message to their phone number. Users can receive SMS messages informing them of their booking time and cancellations. Users can also enter the parking lot by responding to the SMS.

- 6. Facilities:** The smart parking lot can include Wi-Fi routers and mobile phone chargers to let drivers access the application. Some drivers may lose connection; therefore, they can have the access to the router through the application. If users start the app without connecting to the internet, a message may display granting them access to Wi-Fi routers. Additionally, some drivers may run out of battery power, thus mobile phone charging stations will be available inside the parking lot.
- 7. Customer Support and Updates:** Customer support must be included in the Smart Parking Reservation system application. Users can report any problems or bugs they encounter in the application so that they can be fixed. It also provides information on user demand, which aids in the release of new software updates. Versions are often published in response to changes in the economy, trends, and technology. As a result, the Smart Parking Reservation System must keep up to date with changing conditions.
- 8. Prizes:** Drivers may also collect virtual tokens by using the app, which they can redeem for free services. The prizes may include free parking hours or the ability to use other parking lot facilities without being charged. As a result, it will encourage drivers to be environmentally conscious and make use of the Smart Parking Reservation System.

## REFERENCES

- [1] “Kuwait City Traffic Report: TomTom Traffic Index.” TomTom Traffic Index, Dutch Multinational Company, [www.tomtom.com/en\\_gb/traffic-index/kuwait-city-traffic/](http://www.tomtom.com/en_gb/traffic-index/kuwait-city-traffic/).
- [2] Times, Arab. “Looking out for Parking Space in Kuwait Has Become a Nightmare - 100,000 Cars Chase 60,000 Lots.” *ARAB TIMES - KUWAIT NEWS*, 4 Dec. 2016, [www.arabtimesonline.com/news/looking-parking-space-kuwait-become-nightmare-100000-cars-chase-60000-lots/](http://www.arabtimesonline.com/news/looking-parking-space-kuwait-become-nightmare-100000-cars-chase-60000-lots/).
- [3] Zhang, Kai, and Stuart Batterman. “Air Pollution and Health Risks Due to Vehicle Traffic-The Science of the Total Environment.” *U.S. National Library of Medicine (National Institute of Health)*, 15 Apr. 2013, [www.ncbi.nlm.nih.gov/pmc/articles/PMC4243514/](http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4243514/).
- [4] Agrawal, Abhishek, Kamde, Pritesh, et al. “Car Parking System Using IR Sensors.” *International Journal of Advance Research in Engineering, Science & Technology*, vol. 4, no. 2, 2017, pp 51-56, doi: [http://www.ijarest.com/papers/finished\\_papers/150207192005.pdf](http://www.ijarest.com/papers/finished_papers/150207192005.pdf).
- [5] Mazlan, Muhammad Syamil, et al. “Radio Frequency Identification (RFID) Based Car Parking System.” *JOIV: International Journal on Informatics Visualization*, vol. 2, no. 4-2, 2018, pp. 318-322., doi:10.30630/joiv.2.4-2.173.
- [6] Kianpisheh, Amin, Mustaffa, Norlia, et al. “Smart Parking System (SPS) Architecture Using Ultrasonic Detector.” *International Journal of Software Engineering and Its Applications*, vol. 6, no. 3, 2012, pp 51-56, doi: [230701092\\_Smart\\_Parking\\_System\\_SPS\\_Architecture\\_Using\\_Ultrasonic\\_Detector](https://doi.org/10.13001/230701092_Smart_Parking_System_SPS_Architecture_Using_Ultrasonic_Detector).
- [7] Abidin, Muhammad Zainal, Pulungan, Reza. “A Systematic Review of Machine-vision-based Smart Parking Systems.” *Scientific Journal of Informatics*, vol. 7, no. 2, 2020, pp 213-227, doi:10.15294/sji.v7i2.25654.
- [8] V. D. Dokania, M. M. Sevak, D. D. Patel and P. S. Barve, "QR Code based Smart Parking System," **2020 International Conference on Communication and Signal Processing (ICCSP)**, Chennai, India, 2020, pp. 167-170, doi: 10.1109/ICCSP48568.2020.9182180.

- [9] Asyrani Bin Sulaiman, Hamzah, et al. “Wireless based Smart Parking System using Zigbee.” *International Journal of Engineering and Technology (IJET)*, Melaka, Malaysia, vol. 5, no. 4 2013, pp 3282-3300, doi: 287003681\_Wireless\_based\_Smart\_Parking\_System\_using\_Zigbee.
- [10] Mackey, Andrew, et al. “Smart Parking System Based on Bluetooth Low Energy Beacons with Particle Filtering.” *IEEE Systems Journal*, vol. 14, no. 3, 2020, pp. 3371-3382., doi:10.1109/jsyst.2020.2968883.
- [11] Mufaqih, Moh Sukron, et al. “Applying Smart Parking System with Internet of Things (IoT) Design.” *IOP Conference Series: Materials Science and Engineering*, vol. 725, 2020, p. 012095., doi:10.1088/1757-899x/725/1/012095.
- [12] Seth, ElakyaR Juhi, et al. “Smart Parking System Using IoT.” *International Journal of Engineering and Advanced Technology (IJEAT)*, vol. 9, no. 1, Oct. 2019, pp. 6091–6095., doi:10.35940/ijeat.A1963.109119.
- [13] Thangam, E Cassin, et al. “Internet of Things (IoT) Based Smart Parking Reservation System Using Raspberry-Pi.” *International Journal of Applied Engineering Research*, vol. 13, no. 8, 2018, pp. 5729–5764., doi: [https://www.ripublication.com/ijaer18/ijaerv13n8\\_25.pdf](https://www.ripublication.com/ijaer18/ijaerv13n8_25.pdf).
- [14] Al-Kharusi, Hilal, and Ibrahim Bahadly. “Intelligent Parking Management System Based on Image Processing.” *World Journal of Engineering and Technology*, vol. 2, no. 2, 2014, pp. 55–67., doi:<https://www.scirp.org/journal/paperinformation.aspx?paperid=45644>.
- [15] “E-Parking Software (Getpass).” *Pass*, 1 Nov. 2018. [www.getpass.me/](http://www.getpass.me/).
- [16] “Mawqif: About Us.” *Mawqif Technologies - Parking Simplified*, 2019, [www.mawqiftech.com/about-us/](http://www.mawqiftech.com/about-us/).
- [17] Sangali, Paolo. “Parkpiù Installed Car Parking System at The Palms Hotel in Kuwait.” *Parking Network*, Parkpiù, 27 July 2018, [www.parking-net.com/parking-news/parkpiu/installed-car-parking-system-at-the-palms-hotel-in-kuwait](http://www.parking-net.com/parking-news/parkpiu/installed-car-parking-system-at-the-palms-hotel-in-kuwait).
- [18] “Pololu - Arduino Micro.” *Pololu Robotics & Electronics*, [www.pololu.com/product/2188](http://www.pololu.com/product/2188).
- [19] “Pololu - Arduino Mega 2560 R3.” *Pololu Robotics & Electronics*, [www.pololu.com/product/1699](http://www.pololu.com/product/1699).

- [20] “EBot 8 PRO.” EBot, [online]. Available: [ebots.cc/ebot-8pro](http://ebots.cc/ebot-8pro).
- [21] “IR Sensor Module Pinout, Features & Datasheet.” *Components101*, 30 Aug. 2018, [components101.com/sensors/ir-sensor-module](http://components101.com/sensors/ir-sensor-module).
- [22] “Ultrasonic Sensor Working Pinout Datasheet.” *Components101*, 18 Sept. 2017, [components101.com/ultrasonic-sensor-working-pinout-datasheet](http://components101.com/ultrasonic-sensor-working-pinout-datasheet).
- [23] “EBot IOT Board.” EBot, [online]. Available: [ebots.cc/ebot-IOTboard](http://ebots.cc/ebot-IOTboard)
- [24] “Servos Explained.” *Servos Explained - SparkFun Electronics*, [www.sparkfun.com/servos](http://www.sparkfun.com/servos).
- [25] “WS2812B Intelligent Control LED Integrated Light Source.” *WS2812B Data Sheet*, WorldSemi, [cdn-shop.adafruit.com/datasheets/WS2812B.pdf](http://cdn-shop.adafruit.com/datasheets/WS2812B.pdf).
- [26] “EBot IR Sensor.” EBot, [online]. Available: [ebots.cc/ebot-IRSensor](http://ebots.cc/ebot-IRSensor)
- [27] “Getting Started with Arduino IDE.” STEMpedia, 5 Feb. 2020, [thestempedia.com/tutorials/arduino-ide/](http://thestempedia.com/tutorials/arduino-ide/).
- [28] “EBot Blockly.” *EBot*, [ebots.cc/ebot-blockly](http://ebots.cc/ebot-blockly).
- [29] “Visual Studio Code Frequently Asked Questions.” *RSS*, Microsoft, 14 Apr. 2016, [code.visualstudio.com/docs/supporting/faq](http://code.visualstudio.com/docs/supporting/faq).

# APPENDIX A: Hardware Programming

## Part 1

```
#include <Servo.h> //servo library
#include "Ebot.h" // ebot library
#include "Adafruit_NeoPixel.h" // to define LEDs
#define ir(x) analogRead(x) //defining ir sensor in variable called x as an analog input
String text = "";
Adafruit_NeoPixel strip11 = Adafruit_NeoPixel(4, 11, NEO_GRB + NEO_KHZ800); //this is built in RGB
Adafruit_NeoPixel strip0 = Adafruit_NeoPixel(1, 0, NEO_GRB + NEO_KHZ800); //RGB 1 takes pin 0
Adafruit_NeoPixel strip3 = Adafruit_NeoPixel(1, 3, NEO_GRB + NEO_KHZ800); //RGB 2 takes pin 3
Adafruit_NeoPixel strip6 = Adafruit_NeoPixel(1, 6, NEO_GRB + NEO_KHZ800); //RGB 3 takes pin 6

Servo myservo1; // defining each servo here we will work on the first 3 servo for each gate including the main
gate
Servo myservo2;
Servo myservo3;
Servo myservoM;

void setup() // the initial program set up where values of functions are defined only once.
{ ebot_setup(); // this is the function of ebot set up since our ebot has many built in function here we can control
some of these function
  Serial.begin(115200); // this is to enable the serial communication that will be taken from IOT

  myservo1.attach(1); // servo pin for gate A1
  myservo2.attach(4); // servo pin for gate A2
  myservo3.attach(7); // servo pin for gate A3

  myservoM.attach(5); // servo pin for main gate

  pinMode(0, OUTPUT); // LED for gate A1
  pinMode(3, OUTPUT); // LED for gate A2
```

```

pinMode(6, OUTPUT); // LED for gate A3

pinMode(A0, INPUT); // IR for gate A1
pinMode(A3, INPUT); // IR for gate A2
pinMode(A6, INPUT); // IR for gate A3

// in this blok of code we will set up the RGB first

strip0.begin();
strip0.setPixelColor(0, strip0.Color(0, 255, 0)); // for available park spot we want to make sure that the system
starts with green which value =255 as
strip0.show();

strip3.begin();
strip3.setPixelColor(0, strip3.Color(0, 255, 0));
strip3.show();

strip6.begin();
strip6.setPixelColor(0, strip6.Color(0, 255, 0));
strip6.show();

//seting up srvo to be closed first not open initially
myservo1.write(100); // close position
myservo2.write(130); // close position
myservo3.write(130); // close position
myservoM.write(125); // close position
ebotSingInit(12); //Buzzer

}

void loop() {
  if (Serial.available() > 0 ) { // if we recieved serial from IOT which >0 then begin
    text = Serial.readString(); //store this value in variable named text of type string
  }
}

```

```

if (text.charAt(0) == '#') { //charAt will check the index of each character in a string

    if (text.charAt(1) == 'A') {
        if (text.charAt(2) == '4') {
            openMain() ;
        }
        else if (text.charAt(2) == '1') {
            open1() ;
        }
        else if (text.charAt(2) == '2') {
            open2();
        }
        else if (text.charAt(2) == '3') {
            open3();
        }
    }
    else if (text.charAt(1) == 'S') {
        for ( int i = 2 ; i < 14 ; i = i + 2 ) {

            updateStatusLEDs(text.charAt(i), text.charAt(i + 1)); // (int gateNumber, char gateStatus)

        }
    }
}

// here are all the methods which used in our program

void openMain() { //this is the open method for the main gate
    myservoM.write(35); // open position of servo
    delay(3000); //Delay for 0 hours 0 mins 3 sec 0 ms
    myservoM.write(125); // close position
}

void open1() { // method to open A1

```

```

myservo1.write(0); // open position
if (ir(A0) >= 0 && ir(A0) <= 70) //o to 70 this is range of the IR sensor between the sensor and the car height .
{
  while (!(ir(A0) >= 200 && ir(A0) <= 1023)) { //inifnit loop until the condition is achieved
    alarm();
  }
  strip0.setPixelColor(0, strip0.Color(0, 255, 0)); // green
  strip0.show();
}
else
{
  while (!(ir(A0) >= 0 && ir(A0) <= 100)) {
    alarm();
  }
  strip0.setPixelColor(0, strip0.Color(255, 0, 0)); // red
  strip0.show();
}
delay(2000); //Delay for 0 hours 0 mins 3 sec 0 ms
myservo1.write(100); // close position
}

void open2() {
  myservo2.write(40); // open position
  if (ir(A3) >= 0 && ir(A3) <= 100)
  {
    while (!(ir(A3) >= 200 && ir(A3) <= 1023)) {
      alarm();
    }
    strip3.setPixelColor(0, strip3.Color(0, 255, 0)); // red
    strip3.show();

  }
  else
  {
    while (!(ir(A3) >= 0 && ir(A3) <= 100)) {

```

```

    alarm();
  }
  strip3.setPixelColor(0, strip3.Color(255, 0, 0)); // red
  strip3.show();
}
delay(2000); //Delay for 0 hours 0 mins 3 sec 0 ms
myservo2.write(130); // close position
}

void open3() {
  myservo3.write(40); // open position
  if (ir(A6) >= 0 && ir(A6) <= 100)
  {
    while (!(ir(A6) >= 200 && ir(A6) <= 1023)) {
      alarm();
    }
    strip6.setPixelColor(0, strip6.Color(0, 255, 0)); // red
    strip6.show();

  }
  else
  {
    while (!(ir(A6) >= 0 && ir(A6) <= 100)) {
      alarm();
    }
    strip6.setPixelColor(0, strip6.Color(255, 0, 0)); // red
    strip6.show();
  }
  delay(2000); //Delay for 0 hours 0 mins 3 sec 0 ms
  myservo3.write(130); // close position
}

// this is will be taken from IOT to update the status of the LED

void updateStatusLEDs(char gateNumber, char gateStatus) {

```

```

int r = 0;
int g = 0;
int b = 0;
if (gateStatus == 'B') { //this mean booked
    r = 255; //mixing two colors
    g = 200;
    b = 0;
}
else if (gateStatus == 'O') { //occupy
    r = 255;
    g = 0;
    b = 0;
}
else if (gateStatus == 'A') { //available
    r = 0;
    g = 255;
    b = 0;
}
if (gateNumber == '6') {
    strip0.setPixelColor(0, strip0.Color(r, g, b));
    strip0.show();
    delay(10);
}
else if (gateNumber == '4') {
    strip3.setPixelColor(0, strip3.Color(r, g, b));
    strip3.show();
    delay(10);
}
else if (gateNumber == '2') {
    strip6.setPixelColor(0, strip6.Color(r, g, b)); /
    strip6.show();
    delay(10);
}
}

void alarm() { // tone just t give an alarm for the driver to park or exit once gate open or close.
    tone(12, 2093, 500);
}

```

```
delay(500); //Delay for 0 hours 0 mins 0 sec 500 ms
noTone(12);
delay(500); //Delay for 0 hours 0 mins 0 sec 500 ms
}
```

## Part 2

```
#include <Servo.h>
#include "Ebot.h"
#include "Adafruit_NeoPixel.h"
#define ir(x) analogRead(x)
String text = "";
Adafruit_NeoPixel strip11 = Adafruit_NeoPixel(4, 11, NEO_GRB + NEO_KHZ800);
Adafruit_NeoPixel strip0 = Adafruit_NeoPixel(1, 0, NEO_GRB + NEO_KHZ800);
Adafruit_NeoPixel strip3 = Adafruit_NeoPixel(1, 3, NEO_GRB + NEO_KHZ800);
Adafruit_NeoPixel strip6 = Adafruit_NeoPixel(1, 6, NEO_GRB + NEO_KHZ800); //Ebot libraries For
Servo , IR & RGB LEDs

Servo myservo1;
Servo myservo2;
Servo myservo3; // Servos configuration

void setup()
{ ebot_setup();
  Serial.begin(115200);

  myservo1.attach(1); // servo pin for gate B1
  myservo2.attach(4); // servo pin for gate B2
  myservo3.attach(7); // servo pin for gate B3

  pinMode(0, OUTPUT); // LED for gate B1
  pinMode(3, OUTPUT); // LED for gate B2
  pinMode(6, OUTPUT); // LED for gate B3

  pinMode(A0, INPUT); // IR for gate B1
  pinMode(A3, INPUT); // IR for gate B2
  pinMode(A6, INPUT); // IR for gate B3

  strip0.begin();
  strip0.setPixelColor(0, strip0.Color(0, 255, 0));
  strip0.show();
  strip3.begin();
  strip3.setPixelColor(0, strip3.Color(0, 255, 0));
  strip3.show();
  strip6.begin();
  strip6.setPixelColor(0, strip6.Color(0, 255, 0));
  strip6.show(); // intial conditions of LEDs

  myservo1.write(170); // close position
  myservo2.write(170); // close position
  myservo3.write(180); // close position

  ebotSingInit(12); //Buzzer
```

```

}

void loop() {
  if (Serial.available() > 0 ) {
    text = Serial.readString();
    Serial.println(text);
    if (text.charAt(0) == '#') {

      if (text.charAt(1) == 'B') {
        if (text.charAt(2) == '1') {
          open1();
        }
        else if (text.charAt(2) == '2') {
          open2();
        }
        else if (text.charAt(2) == '3') {
          open3();
        }
      }
      else if (text.charAt(1) == 'S') {
        for ( int i = 2 ; i < 14 ; i = i + 2 ) {

          updateStatusLEDs(text.charAt(i), text.charAt(i + 1)); // (int gateNumber, char gateStatus)

        }
      }
    }
  }
}

void open1() {
  myservo1.write(80); // open position
  if (ir(A0) >= 0 && ir(A0) <= 100)
  {
    while (!(ir(A0) >= 200 && ir(A0) <= 1023)) {
      alarm();
    }
    strip0.setPixelColor(0, strip0.Color(0, 255, 0)); // red
    strip0.show();
  }
  else
  {
    while (!(ir(A0) >= 0 && ir(A0) <= 100)) {
      alarm();
    }
    strip0.setPixelColor(0, strip0.Color(255, 0, 0)); // red
    strip0.show();
  }
}

```

```

}
delay(2000); //Delay for 0 hours 0 mins 3 sec 0 ms
myservo1.write(170); // close position
}

void open2() {
myservo2.write(80); // open position
if (ir(A3) >= 0 && ir(A3) <= 100)
{
while (!(ir(A3) >= 200 && ir(A3) <= 1023)) {
alarm();
}
strip3.setPixelColor(0, strip3.Color(0, 255, 0)); // red
strip3.show();
}
else
{
while (!(ir(A3) >= 0 && ir(A3) <= 100)) {
alarm();
}
strip3.setPixelColor(0, strip3.Color(255, 0, 0)); // red
strip3.show();
}
delay(2000); //Delay for 0 hours 0 mins 3 sec 0 ms
myservo2.write(170); // close position
}

void open3() {
myservo3.write(90); // open position
if (ir(A6) >= 0 && ir(A6) <= 100)
{
while (!(ir(A6) >= 200 && ir(A6) <= 1023)) {
alarm();
}
strip6.setPixelColor(0, strip6.Color(0, 255, 0)); // red
strip6.show();
}
else
{
while (!(ir(A6) >= 0 && ir(A6) <= 100)) {
alarm();
}
strip6.setPixelColor(0, strip6.Color(255, 0, 0)); // red
strip6.show();
}
delay(2000); //Delay for 0 hours 0 mins 3 sec 0 ms
myservo3.write(180); // close position
}

void updateStatusLEDs(char gateNumber, char gateStatus) {

```

```

int r = 0;
int g = 0;
int b = 0;
if (gateStatus == 'B') {
  r = 255;
  g = 200;
  b = 0;
}
else if (gateStatus == 'O') {
  r = 255;
  g = 0;
  b = 0;
}
else if (gateStatus == 'A') {
  r = 0;
  g = 255;
  b = 0;
}
if (gateNumber == '5') {
  strip0.setPixelColor(0, strip0.Color(r, g, b));
  strip0.show();
  delay(10);
}
else if (gateNumber == '3') {
  strip3.setPixelColor(0, strip3.Color(r, g, b));
  strip3.show();
  delay(10);
}
else if (gateNumber == '1') {
  strip6.setPixelColor(0, strip6.Color(r, g, b));
  strip6.show();
  delay(10);
}
}
void alarm() {
  tone(12, 2093, 500);
  delay(500); //Delay for 0 hours 0 mins 0 sec 500 ms
  noTone(12);
  delay(500); //Delay for 0 hours 0 mins 0 sec 500 ms
}

```

## Appendix B: Segment of the Software Programming

### Parking List (Dart)

```
import 'package:flutter/material.dart';
import 'package:parking_flutter/parking/components/parking_card.dart';
import 'package:parking_flutter/parking/models/parking.dart';

class ParkingList extends StatefulWidget {
  final List<Parking> parkingAreas;

  const ParkingList({Key key, this.parkingAreas}) : super(key: key);
  @override
  _ParkingListState createState() => _ParkingListState();
}

class _ParkingListState extends State<ParkingList> {
  // Parking parking = Parking(
  //   availablelots: 74,
  //   details: "Al Asmiah, Block 4 Kuwait Al Rai",
  //   location: "https://goo.gl/maps/sBLbc4eVn7bVrGvNA",
  //   name: "Avenues Phase 4",
  //   price: 0.25,
  // );
  @override
  Widget build(BuildContext context) {
    if (widget.parkingAreas == null) return Text('No parkings available');
    if (widget.parkingAreas.length == 0)
      return Text('No parkings available');
    else
      return Column(
        children: widget.parkingAreas
          .map((parking) => ParkingCard(parking: parking))
          .toList());
  }
}
```

## Parking Card (Dart)

```
import 'package:flutter/material.dart';
import 'package:parking_flutter/parking/models/parking.dart';
import 'package:parking_flutter/parking/models/parking_status.dart';
import 'package:parking_flutter/theme/base_components/button_small.dart';
import 'package:parking_flutter/theme/base_components/button_small_outlined.dart';
import 'package:parking_flutter/theme/color_palette.dart';
import 'package:url_launcher/url_launcher.dart';

import 'confirm_booking_dialog.dart';

class ParkingCard extends StatefulWidget {
  final Parking parking;

  const ParkingCard({Key key, @required this.parking}) : super(key: key);
  @override
  _ParkingCardState createState() => _ParkingCardState();
}

class _ParkingCardState extends State<ParkingCard> {
  @override
  Widget build(BuildContext context) {
    bool _isAvailable = false;
    int _slotsAvailable = getNumberOfAvailableSlots(widget.parking);
    _isAvailable = _slotsAvailable > 0;
    return Container(
      margin: EdgeInsets.fromLTRB(24, 24, 24, 0),
      padding: EdgeInsets.fromLTRB(24, 24, 24, 12),
      decoration: BoxDecoration(
        color: Colors.white,
        borderRadius: BorderRadius.circular(9),
        boxShadow: [
          BoxShadow(
            color: Colors.black.withOpacity(0.05),
            spreadRadius: 0,
            blurRadius: 12,
            offset: Offset(0, 2), // changes position of shadow
          ),
        ],
      ),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Row(
            mainAxisAlignment: MainAxisAlignment.spaceBetween,
            children: [
              Text(widget.parking.name ?? "Unavailable",
                style: TextStyle(fontWeight: FontWeight.bold)),
              Spacer(),
              Text(
                "Available",
                style: TextStyle(color: Colors.black38, fontSize: 12),
              ),
            ],
          ),
          SizedBox(width: 4),
          Text(
```

```

        _slotsAvailable.toString(),
        style: Theme.of(context).textTheme.subtitle1.merge(
          TextStyle(
            color: _isAvailable ? Palette.SECONDARY : Colors.red,
          ),
        ),
      ],
    ),
    widget.parking.price != null
      ? Text(
          (widget.parking.price * 1000).toStringAsFixed(0) +
            " Fils/Hour",
          style: TextStyle(color: Palette.SECONDARY, fontSize: 12))
      : Container(),
    Text(
      widget.parking.details ?? "",
      style: TextStyle(color: Colors.black38, fontSize: 12),
    ),
    SizedBox(height: 6),
    Row(
      mainAxisAlignment: MainAxisAlignment.spaceBetween,
      children: [
        ButtonSmallOutlined(
          color: Palette.SECONDARY.withOpacity(_isAvailable ? 1 : 0.35),
          buttonText: "SHOW ON MAP",
          onPressed: () {
            if (widget.parking.location != null && _isAvailable)
              _launchInBrowser(widget.parking.location);
          },
          // width: MediaQuery.of(context).size.width / 3,
          // color: Palette.SECONDARY,
        ),
        ButtonSmall(
          buttonText: "BOOK NOW",
          color: Palette.SECONDARY.withOpacity(_isAvailable ? 1 : 0.35),
          onPressed: () {
            if (_isAvailable)
              showDialog(
                context: context,
                builder: (_) {
                  return ConfirmBookingDialog(parking: widget.parking);
                },
              );
          },
          // color: Palette.SECONDARY,
          // width: MediaQuery.of(context).size.width / 3,
        ),
      ],
    ),
  ],
),
);
}

```

```

Future<void> _launchInBrowser(String url) async {
  if (url.isNotEmpty) {

```

```
if (await canLaunch(url)) {
  await launch(
    url,
    forceSafariVC: false,
    forceWebView: false,
    // headers: <String, String>{'my_header_key': 'my_header_value'},
  );
} else {
  // throw 'Could not launch $url';
}
}
}

int getNumberOfAvailableSlots(Parking parking) {
  int _availableSlots = 0;
  parking.parkingSlots?.forEach((slot) {
    if (slot.status == ParkingStatus.AVAILABLE) {
      _availableSlots++;
    }
  });
  return _availableSlots;
}
}
```

## Confirm Booking Dialog (Dart)

```
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:flutter_svg/svg.dart';
import 'package:parking_flutter/core/app/app.dart';
import 'package:parking_flutter/core/app/models/personal_details.dart';
import 'package:parking_flutter/core/helpers/validate.dart';
import 'package:parking_flutter/parking/bloc/book_parking/book_parking_bloc.dart';
import 'package:parking_flutter/parking/components/wallet_dialog.dart';
import 'package:parking_flutter/parking/models/parking.dart';
import 'package:parking_flutter/parking/models/parking_slot.dart';
import 'package:parking_flutter/parking/models/parking_status.dart';
import 'package:parking_flutter/theme/base_components/button_small.dart';
import 'package:parking_flutter/theme/base_components/custom_text_input.dart';
import 'package:parking_flutter/user/authentication.dart';
import 'package:parking_flutter/user/blocs/profile/profile_bloc.dart';

import '../theme/color_palette.dart';

class ConfirmBookingDialog extends StatefulWidget {
  final Parking parking;

  const ConfirmBookingDialog({Key key, @required this.parking})
    : super(key: key);
  @override
  _ConfirmBookingDialogState createState() => _ConfirmBookingDialogState();
}

class _ConfirmBookingDialogState extends State<ConfirmBookingDialog> {
  List<ParkingSlot> _parkingSlots;
  ParkingSlot _selectedSlot;
  BookParkingBloc _bookParkingBloc = BookParkingBloc();
  ProfileBloc _profileBloc;
  final _phoneController = TextEditingController();
  bool isPhoneValid;

  @override
  void dispose() {
    _phoneController.dispose();
    super.dispose();
  }

  @override
  void initState() {
    super.initState();
    _parkingSlots = widget.parking.parkingSlots ?? [];
    _profileBloc = BlocProvider.of<ProfileBloc>(context);
    if (authentication.isAuthenticated) {
      _phoneController.text = authentication.user?.phone ?? "";
    } else {
      _phoneController.text = app.personalDetails?.phone ?? "";
    }
    isPhoneValid = _phoneController.text != "";
  }
}
```

```

@override
Widget build(BuildContext context) {
  bool _isGuest = !authentication.isAuthenticated;
  return AlertDialog(
    contentPadding: EdgeInsets.symmetric(horizontal: 0, vertical: 24),
    content: BlocBuilder<BookParkingBloc, BookParkingState>(
      cubit: _bookParkingBloc,
      builder: (context, state) {
        if (state is BookParkingInProgress) {
          return CupertinoActivityIndicator();
        } else if (state is BookParkingSuccess) {
          _profileBloc.add(
            GetUserProfile(
              deviceId: app.deviceId,
            ),
          );
          return Column(
            mainAxisAlignment: MainAxisAlignment.min,
            children: [
              Icon(
                Icons.check,
                size: 100,
                color: Colors.green,
              ),
              Text('Booking completed')
            ],
          );
        } else
          return Container(
            child: SingleChildScrollView(
              child: Column(
                children: [
                  Text(
                    widget.parking.name,
                    style: Theme.of(context).textTheme.subtitle2.merge(
                      TextStyle(
                        color: Palette.SECONDARY,
                      ),
                    ),
                  ),
                  Text(
                    widget.parking.details ?? "",
                    style: TextStyle(color: Colors.black38, fontSize: 12),
                  ),
                  Container(
                    width: double.infinity,
                    margin: EdgeInsets.symmetric(vertical: 8),
                    padding: EdgeInsets.symmetric(vertical: 8),
                    color: Palette.SECONDARY.withOpacity(0.2),
                    child: Center(
                      child: Text(
                        (widget.parking.price * 1000).toStringAsFixed(0) +
                          " Fils/Hour",
                        style: TextStyle(
                          color: Palette.SECONDARY, fontSize: 12),
                      ),
                    ),
                  ),
                ],
              ),
            ),
          );
      },
    ),
  );
}

```

```

    ),
  ),
  SizedBox(height: 12),
  Padding(
    padding: const EdgeInsets.symmetric(horizontal: 44.0),
    child: _pakingSlots.length > 0
      ? Wrap(
          spacing: 24,
          runSpacing: 12,
          children: buildSlots(),
        )
      : Text('No slots available'),
  ),
  SizedBox(height: 24),
  Image.asset('lib/core/assets/images/park-guide.png'),
  SizedBox(height: 12),
  Text(
    'Please make sure that you will park \nwithin ${_isGuest ? 1 : 2} Hour(s)',
    style: TextStyle(
      color: Colors.red,
      fontSize: 12,
    ),
    textAlign: TextAlign.center,
  ),
  _phoneTextField(),
  BlocBuilder<ProfileBloc, ProfileState>(
    builder: (context, state) {
      if (state is GetProfileSuccess)
        return ButtonSmall(
          buttonText: "CONFIRM BOOKING",
          color: Palette.SECONDARY.withOpacity(
            _selctedSlot != null && isPhoneValid
              ? 1
              : 0.35),
          onPressed: () {
            double _parkingPrice =
              widget.parking.price ?? 0.1;

            double _walletAmount =
              state.userProfile?.user?.walletAmout ?? 0.0;
            if (_walletAmount >= _parkingPrice) {
              if (_selctedSlot != null &&
                _phoneController.text != null)
                app.setPersonalDetails(
                  personalDetails: PersonalDetails(
                    address: "",
                    name: "guest",
                    email: "",
                    phone: _phoneController.text),
                );
              _bookParkingBloc.add(
                CreateBookParkingEvent(
                  phone: _phoneController.text,
                  parkingAreaName:
                    widget.parking?.name ?? "unknown",
                  parkingSlot: _selctedSlot?.id ?? "",
                ),
              );
            }
          },
        );
    },
  ),

```

```

        parkingSlotName:
            _selctedSlot?.name ?? "unknown",
        price: widget.parking?.price ?? 0.1,
    ),
);
} else if (_walletAmount < _parkingPrice) {
    showDialog(
        context: context,
        builder: (_) {
            return WalletDialog(
                isBalanceLow: true,
            );
        },
    );
}
);
else {
    return Container();
}
),
),
),
);
},
),
);
}

buildSlots() {
    int index = 0;
    return _pakingSlots.map((slot) {
        index++;
        return Stack(
            alignment: index % 2 != 0 ? Alignment.topLeft : Alignment.topRight,
            children: [
                InkWell(
                    onTap: () {
                        if (slot.status == ParkingStatus.AVAILABLE) {
                            setState(() {
                                _selctedSlot = slot;
                            });
                        }
                    },
                ),
                child: SlotIndicator(
                    name: slot.name,
                    status: slot.status,
                    isLeft: index % 2 != 0,
                ),
            ],
            _selctedSlot == slot
                ? Padding(
                    padding: const EdgeInsets.all(4.0),
                    child: Icon(

```

```

        Icons.check_circle,
        size: 28,
        color: Palette.SECONDARY,
    ),
)
: SizedBox(width: 1)
],
);
}).toList();
}

_phoneTextField() {
return Padding(
padding: const EdgeInsets.symmetric(horizontal: 32.0),
child: CustomTextField(
isNumber: true,
height: 72.0,
title: "Phone",
icon: Icons.phone,
enabled: true,
controller: _phoneController,
validator: (value) {
return Validate.mobile(value);
},
onChanged: (value) {
if (Validate.mobile(value) == null) {
setState(() {
isPhoneValid = true;
});
} else {
setState(() {
isPhoneValid = false;
});
}
},
),
);
}
}

class SlotIndicator extends StatefulWidget {
const SlotIndicator({
Key key,
@required ParkingStatus status,
@required String name,
this.isLeft = true,
this.selectThisSlot,
}) : _status = status,
_name = name,
super(key: key);

final ParkingStatus _status;
final String _name;
final bool isLeft;
final Function selectThisSlot;
}

```

```

@override
_SlotIndicatorState createState() => _SlotIndicatorState();
}

class _SlotIndicatorState extends State<SlotIndicator> {
  @override
  Widget build(BuildContext context) {
    Color _color = widget._status == ParkingStatus.AVAILABLE
      ? Palette.SECONDARY
      : widget._status == ParkingStatus.BOOKED
      ? Colors.amber
      : Colors.red;
    return InkWell(
      onTap: widget.selectThisSlot,
      child: Stack(
        alignment: Alignment.center,
        children: [
          SvgPicture.asset(
            widget.isLeft
              ? 'lib/core/assets/images/park-slot-left.svg'
              : 'lib/core/assets/images/park-slot-right.svg',
            color: _color,
          ),
          Column(
            children: [
              Text(
                widget._name,
                style: TextStyle(
                  color: _color,
                  fontWeight: FontWeight.bold,
                ),
              ),
              Text(
                widget._status == ParkingStatus.AVAILABLE
                  ? "Available"
                  : widget._status == ParkingStatus.BOOKED
                  ? "Booked"
                  : "Occupied",
                style: TextStyle(fontSize: 12, color: _color),
              ),
            ],
          ),
        ],
      ),
    );
  }
}

```

## Wallet Dialog (Dart)

```
import 'package:flutter/material.dart';
import 'package:parking_flutter/core/helpers/validate.dart';
import 'package:parking_flutter/payments/input_models/payment_input.dart';
import 'package:parking_flutter/payments/payment_router.dart';
import 'package:parking_flutter/theme/base_components/button_medium_outlined.dart';
import 'package:parking_flutter/theme/base_components/custom_text_input.dart';
import 'package:parking_flutter/theme/color_palette.dart';
import 'package:parking_flutter/user/authentication.dart';

class WalletDialog extends StatefulWidget {
  final bool isBalanceLow;

  const WalletDialog({Key key, this.isBalanceLow = true}) : super(key: key);
  @override
  _WalletDialogState createState() => _WalletDialogState();
}

class _WalletDialogState extends State<WalletDialog> {
  final _amountController = TextEditingController();
  final _amountFocus = FocusNode();
  final _btnFocus = FocusNode();

  final _formKey = GlobalKey<FormState>();
  @override
  Widget build(BuildContext context) {
    return AlertDialog(
      contentPadding: EdgeInsets.symmetric(horizontal: 12, vertical: 24),
      content: Container(
        child: SingleChildScrollView(
          child: Form(
            key: _formKey,
            child: Column(
              children: [
                Text("Wallet Topup",
                  style: Theme.of(context).textTheme.subtitle1),
                widget.isBalanceLow
                  ? Text(
                      "Your Balance is too low. Please topup to proceed",
                      style: Theme.of(context).textTheme.headline6.merge(
                        TextStyle(
                          color: Colors.red,
                        ),
                      ),
                      textAlign: TextAlign.center,
                    )
                  : Container(),
                SizedBox(height: 20),
                _amountTextField(),
                SizedBox(height: 40),
                ButtonMediumOutlined(
                  fillColor: Palette.SECONDARY,
                  focusNode: _btnFocus,
                  buttonText: "PROCEED TO PAY",
                  onTap: () {
```

```

        _submitForm();
    },
    ),
  ],
),
),
),
);
}

_amountTextField() {
return Padding(
padding: const EdgeInsets.symmetric(horizontal: 24.0),
child: CustomTextField(
isNumber: true,
height: 72.0,
title: "Amount in KD",
icon: Icons.account_balance_wallet,
enabled: true,
controller: _amountController,
focusNode: _amountFocus,
textInputAction: TextInputAction.next,
validator: (value) {
return Validate.value(value);
},
onFieldSubmitted: (value) {
if (Validate.value(value) == null) {
_changeFieldFocus(
context,
_amountFocus,
_btnFocus,
);
}
},
),
);
}

void _submitForm() {
if (_formKey.currentState.validate()) {
// Goto payment screen
final PaymentInput paymentInput = PaymentInput(
amount: double.parse(_amountController.text ?? "0"),
// date: DateTimeHelper.getCurrentDateTimeString(),
email: "unknown",
name: authentication.authenticatedUser?.user?.name ?? "Guest",
phone: authentication.authenticatedUser?.user?.phone ?? "Guest",
redirectUri: "https://makemyflow.io/redirect/",
);
Navigator.pushNamed(
context,
PaymentRouter.TAP_PAYMENT,
arguments: {
'paymentInput': paymentInput,
},

```

```
);  
}  
}  
  
_changeFieldFocus(  
  BuildContext context,  
  FocusNode currentFocus,  
  FocusNode nextFocus,  
) {  
  currentFocus.unfocus();  
  FocusScope.of(context).requestFocus(nextFocus);  
}  
}
```