



School of Engineering Science
Burnaby, BC V5A 1S6
jnaorbe@sfu.ca

June 27, 2021

Dr. Craig Scratchley
School of Engineering Science
Simon Fraser University
Burnaby, BC, V5A 1S6

Re: ENSC 405W/440 Design Specifications for OpenSpot

Dear Dr. Craig Scratchley,

Attached in this document you will find the design specifications for our smart parking system OpenSpot. Our team designed this system with the goal in mind of reducing the stress of finding open parking spots in busy parking lots. Using mounted cameras combined with computer vision, we allow users to view available parking spots through a mobile-friendly website and light indicators that display the density within a section of the parking lot.

This document outlines the design specifications for our smart parking system along with any necessary justifications. Design choices were made in accordance with the requirements specification document.

Our team consists of students from both Computer Engineering and Systems Engineering disciplines: Justin Naorbe, Curtis Lui, Gurmeh Shergill, Soroush Saheb-Pour-Lighvan, and Darius Nadem. With our determination and passion to create an exceptional product, we will be implementing a system to save valuable time for drivers.

We would like to thank you in advance for taking the time to read the attached design specification document. If you have any further questions or comments, please do not hesitate to reach out to our Chief Executive Officer Justin Naorbe at jnaorbe@sfu.ca.

Sincerely,

Justin Naorbe
Chief Executive Officer
OpenSpot



OpenSpot

ENSC 405W

OpenSpot Design Specification

Company 7

Justin Naorbe

Soroush Saheb-Pour-Lighvan

Curtis Lui

Darius Nadem

Gurmesh Shergill



Abstract

Due to growing populations and driving accessibility, parking in outdoor lots has become increasingly troublesome - especially during peak hours. Finding parking during these times has become dependent on luck and being in the right place at the right time. This ends up wasting the time of many people in the search for empty parking spots, which may lead to frustrated drivers and disputes. Our team has developed OpenSpot with the goal in mind to make finding an open parking spot stress free and efficient with the added bonus of security features.

The OpenSpot smart parking system allows users to view available parking spots through a mobile-friendly website. Upon arrival, drivers can view the density of cars parked through an LED light indicator mounted on light poles, which summarizes the availability of open stalls in the area. A detailed overview of the design specifications for our smart parking system are presented in this document.



Table of Contents

ABSTRACT	I
TABLE OF CONTENTS	II
LIST OF FIGURES	V
LIST OF TABLES	VI
VERSION HISTORY	VII
APPROVALS	VIII
GLOSSARY	IX
1 INTRODUCTION	1
1.1 BACKGROUND.....	1
1.2 SCOPE.....	2
1.3 INTENDED AUDIENCE.....	2
1.4 DESIGN SPECIFICATION CLASSIFICATION.....	2
2 SYSTEM OVERVIEW	2
3 PHYSICAL DESIGN	3
3.1 LAYOUT.....	4
3.2 MOUNTING.....	4
4 OPENSLOT MODULE	5
4.1 PERIPHERALS.....	5
4.1.1 <i>Camera</i>	5
4.1.2 <i>Microphone</i>	6
4.1.3 <i>LED Indicator</i>	7
4.1.4 <i>Network Module</i>	9
4.2 HARDWARE DESIGN.....	9
4.2.1 <i>Microcontroller</i>	9
4.2.2 <i>Power Consumption</i>	11
4.2.3 <i>Microcontroller Power</i>	12
4.2.4 <i>Camera Power</i>	12
4.2.5 <i>LED Power</i>	12
4.2.6 <i>Microphone Power</i>	13
5 MICROCONTROLLER SOFTWARE DESIGN	13
5.1 PERIPHERAL CONTROL.....	13
5.2 COMMUNICATION.....	14
6 BACKEND SERVER DESIGN	14
6.1 COMPUTER VISION (CV).....	14
6.2 AUDIO RECOGNITION.....	16
6.3 COMMUNICATION.....	17
6.4 DATABASE.....	17
7 WEBSITE DESIGN	17



8	CONCLUSION	19
9	REFERENCES	21
10	APPENDICES	25
	APPENDIX A: SUPPORTING TEST PLANS.....	25
	A.1 Introduction.....	25
	A.1.1 Test Purpose	25
	A.1.2 Test Coverage	25
	A.1.3 Test Methods.....	25
	A.1.4 Test Responsibilities	25
	A.2 Physical/Mechanical Testing.....	25
	A.3 Electronics Testing.....	26
	A.4 Module Software Testing	27
	A.5 Backend Server Testing	30
	A.6 Website Testing.....	33
	APPENDIX B: SUPPORTING DESIGN OPTIONS	35
	B.1 Physical.....	35
	B.1.1 Body and Housing Structure	35
	B.1.1.1 Exterior.....	35
	B.1.1.2 Interior	35
	B.1.2 Mounting	36
	B.2 Hardware.....	36
	B.2.1 Microcontroller.....	36
	B.2.2 Camera.....	37
	B.2.3 LED Indicator.....	37
	B.2.4 Microphone	38
	B.2.5 Power Source.....	39
	B.3 Software	39
	B.3.1 Module Communication and Software	39
	B.3.2 Computer Vision	40
	B.3.3 Database	41
	B.3.4 Server Hosting	41
	B.3.5 Client and User Interface	42
	B.3.6 Web Server Framework.....	43
	APPENDIX C: USER INTERFACE AND APPEARANCE-PROTOTYPE DESIGN.....	44
	C.1 Introduction and Background	44
	C.1.1 Purpose.....	44
	C.1.2 Scope	44
	C.2 User Analysis	44
	C.3 Technical Analysis	45
	C.3.1 Discoverability	45
	C.3.2 Feedback.....	45
	C.3.3 Conceptual Model	45
	C.3.4 Affordance	45
	C.3.5 Signifiers	45
	C.3.6 Mappings	45
	C.3.7 Constraints.....	46
	C.4 Engineering Standards and Laws	46
	C.4.1 UI Design Standards.....	46
	C.4.2 Driving Safety Laws and Regulations	46
	C.5 Analytical Usability Testing	46



C.5.1	Website.....	46
C.5.2	Module.....	47
C.6	<i>Empirical Usability Testing</i>	47
C.6.1	End User Activity 1: Navigating to the Website and Signing up for Text Notifications	47
C.6.2	End User Activity 2: Viewing Parking Lots on the Website	47
C.6.3	End User Activity 3: Client notification of an active car alarm within a lot.....	47
C.6.4	End User Activity 4 Clients Able to Register and Login to the Website	48
C.6.5	End User Activity 5: Obtaining help.....	48
C.6.6	End User Activity 6: Account Management.....	48
C.7	<i>Graphical Presentation</i>	48
C.8	<i>Conclusion</i>	50



List of Figures

Figure 1.1.1: OpenSpot System Block Diagram	1
Figure 3.1: OpenSpot Module Mounted on Light Pole.....	3
Figure 3.1.1: OpenSpot Module Model	4
Figure 4.1.1.1: KEYESTUDIO Camera Module	6
Figure 4.1.2.1: USB Raspberry Pi Microphone.....	7
Figure 4.1.3.1: PixelDMX LED.....	8
Figure 4.2.1.1 Raspberry Pi 4 Layout [7].....	9
Figure 4.2.1.2: Raspberry Pi 4 GPIO Layout [9].....	11
Figure 5.1.1: Module Script Flowchart	13
Figure 6.1.1: Computer Vision Flowchart	14
Figure 6.1.2: Sample Instance Segmentation, Bounding Boxes, and Classification	15
Figure 6.1.3: Reference Photo (left) and Example Real Time Photo (right)	16
Figure C.7.1: Landing (main) Page of the Website – Desktop View	49
Figure C.7.2: Mobile Landing Page of Website - Portrait (Left) and Landscape (Right)	49



List of Tables

Table 3.1: Housing Design Specification	3
Table 3.1.1: Layout Design Specifications.....	4
Table 3.2.1: Mounting System Design Specifications	5
Table 4.1.1.1: Camera Specifications [6]	5
Table 4.1.1.2: Camera Design Specifications	6
Table 4.1.2.1: Microphone Design Specification	7
Table 4.1.2.2:Microphone Specification	7
Table 4.1.3.1: LED Indicator Design Specification	8
Table 4.1.3.2: PixelDMX LED Specifications.....	8
Table 4.1.4.1: Network Module Design Specifications	9
Table 4.2.1.1: Raspberry Pi 4 Specifications [8].....	10
Table 4.2.1.2: Microcontroller Design Specifications	10
Table 4.2.2.1: Power Supply Design Specification	12
Table 5.2.1: Module Script Design Specifications.....	14
Table 6.1.1: Computer Vision Design Specifications	16
Table 6.2.1: Audio Recognition Design Specifications.....	16
Table 6.3.1: Server Communication Design Specifications	17
Table 6.4.1: Database Design Specifications	17
Table 7.1: Website Design Specifications	18
Table B.1.1.1.1: Exterior Body and Housing Design Options.....	35
Table B.1.2.1: Mounting Design Options.....	36
Table B.2.1.1: Microcontroller Design Options	36
Table B.2.2.1: Camera Design Options	37
Table B.2.3.1: Indicator Light Design Options	37
Table B.2.4.1: Microphone Design Options	38
Table B.2.5.1: Power Source Design Options	39
Table B.3.1.1: Module Communication Design Options	39
Table B.3.2.1: Computer Vision Application Design Options	40
Table B.3.3.1: Design Options for Database	41
Table B.3.4.1: Design Options for Server Hosting	41
Table B.3.5.1: Design Options for Client & User Interface	42
Table B.3.6.1: Design Options for Web Server Framework.....	43




Version History

Version #	Implemented By	Revision Date	Approved By	Approval Date	Reason
1.0	Justin Naorbe Soroush Saheb-Pour-Lighvan Curtis Lui Darius Nadem Gurmesh Shergill	2021-07-10	Justin Naorbe	2021-07-10	Initial Design Definition Draft



Approvals

Signature:		Date: July 10, 2021
Print Name:	Justin Naorbe	
Title:	Mr.	
Role:	CEO	



Glossary

Client	An individual or organization using the services and products of OpenSpot. E.g., parking lot owners, universities, etc.
Mask R-CNN	A deep neural network that is extended from Faster R-CNN by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition and specializes in instance segmentation [1]
MongoDB Atlas	A general purpose, document-based, and distributed cloud database that uses JSON-like documents [2]
Peripheral(s)	Any input and output components connected to the Raspberry Pi. E.g., camera, microphone, indicator lights, etc.
PLA	Acronym of Polylactic Acid
Script	A collection of commands in a file designed to be executed like a program.
UI	Acronym of User Interface
User	Everyday drivers who are looking for parking spots



1 Introduction

1.1 Background

The OpenSpot team is developing a smart parking system that will allow for a stress-free parking experience. Through our cameras and LED lights, we will be able to provide real-time information to the driver at the parking lot or on their mobile device. At the parking lot, drivers will see the density of parked cars via our pole mounted LED lights which display various colours associated with the availability of parking spots. In addition, our website will provide more in-depth information as to which exact parking spots are available at the respective lot. An overview of our system is shown in Figure 1.1.1, showing how the various subsystems interact with one another. With the implementation of our system, we plan to reduce traffic congestion at parking lots, which cause 30% of day-to-day traffic and in return lowering carbon emissions [3]. Furthermore, with our security feature that detects car alarms we intend to reduce car thefts that occur in parking lots as it is increasing globally and locally – especially in British Columbia as it ranks 3rd highest in vehicle thefts, having a total of 13,352 thefts just in 2019 [4].

As a company we face two main challenges: environmental conditions, and competitors. Many similar companies that provide a smart parking system utilize sensors per parking spot, resulting in a high cost per spot. At OpenSpot, we plan to cover 10-20 parking spot per module with our camera system. In addition, we offer features such as notifications to drivers or security features that have not been implemented in the industry yet. Our model differentiator is also the root of our challenges, as the computer vision program works with images supplied by our camera system. In the situation of extreme weather, such as a snowstorm our functionality might hinder. This is an area our developers are aware of and are working on an algorithm to perform when weather is not optimal.

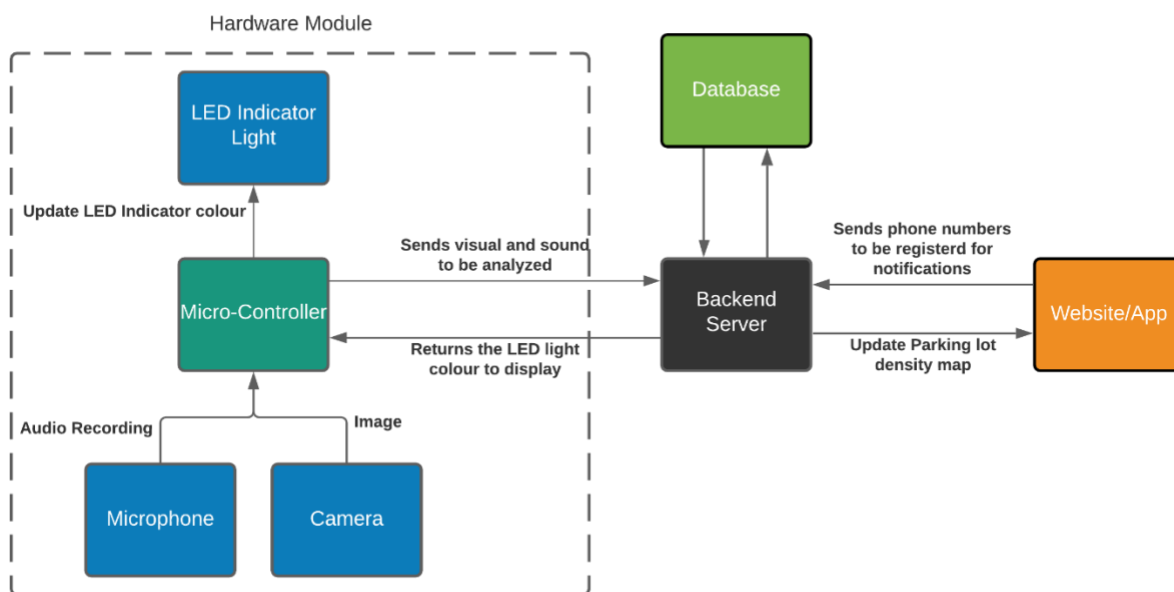


Figure 1.1.1: OpenSpot System Block Diagram



1.2 Scope

The purpose of this document is to specify design requirements and justifications for the OpenSpot smart parking solution which includes: the OpenSpot Module and proprietary software system. The implementation details for each subsystem of the proof-of-concept (PoC) will be discussed.

Appendix A contains the Supporting Test Plans for each subsystem and component. Appendix B contains Supporting Design Options for every part of the system. Lastly, Appendix C contains the description of the User Interface and Appearance-Prototype Design.

The PoC covers the following aspects of our provided solution: detection of open parking spots through a computer vision application, communication from our module to a locally hosted server, and updating the colour of an LED indicator to match the current vacancy status of the field of vision of the camera.

1.3 Intended Audience

This document will function as OpenSpot's design requirement guide for all members of the team, any investors or clients, Dr. Craig Scratchley, Dr. Andrew Rawicz, Srishti Yadav, and Timothy Yu.

1.4 Design Specification Classification

The following labelling scheme is how design specifications are representing in the document:

Des X.Y.Z.#-S

X – Represents the section number

Y – Represents the subsection number (if applicable)

Z – Represents the subsubsection number (if applicable)

– Represents the design specification number

S – References the stages for the requirement, either **A**, **B**, or **P**

A – Represents Alpha Phase (Proof-of-concept Prototype)

B – Represents Beta Phase (Engineering Prototype)

P – Represents Production Phase

2 System Overview

Our system consists of three main components: the hardware module, the backend server, and the website. Figure 1.1.1 shows the system overview of the design and the flow of communication between the peripherals, components, and modules.

The hardware module houses our camera, light indicator, microphone, and microcontroller in a weatherproof box that is mounted on pre-existing light poles. The system is powered by connecting to the power lines within the light poles. Figure 3.1, shows a model of our module mounted on a light pole. The microcontroller communicates with the backend by sending pictures and audio recordings as a request and receives instruction on what colour to display on the light indicator. The backend server responds to the requests and performs all the necessary computation on images and audio recordings. The backend server also communicates with our MongoDB database to track and store the status of each parking spot. It is also used to pull information and respond to website requests. The website is for the client and user to interface with. It provides clients and users a view of the status of each individual parking spot at the respective parking lot.



3 Physical Design

Since the OpenSpot module will be elevated at a great height, the housing design and materials are essential in maintaining the proper functionality of the electrical components. The housing will be a single box that encloses all the electrical components. The housing must absorb the impact from any environmental factors to protect the components. A mock-up of our module on a light pole is shown in Figure 3.1



Figure 3.1: OpenSpot Module Mounted on Light Pole

The average light pole is between 9-14 feet (2.7m – 4.2m) [5]. Therefore, the module will be mounted at an average height of about 3.5m. The design specifications for the housing are listed in Table 3.1 which includes how the components will be protected from environmental factors and accessibility for maintenance.

Table 3.1: Housing Design Specification

Design ID	Design Description	Requirement ID Reference
Des 3.1-B	The housing will be able to fit all necessary electrical components inside. There should be a panel so that the electronic components are accessible for maintenance.	Req 3.2.1-B Req 3.4.1-B
Des 3.2-P	The housing should be able to withstand an impact from about 3.5m in height.	N/A
Des 3.3-P	The housing should be at least IP64 rated to protect electronic components from external factors. This includes temperature changes, harsh weather, and animals.	Req 3.1.1-B Req 3.1.2-B Req 3.1.3-B Req 3.1.4-B



A sample housing made from cardboard will be used for the alpha phase to allow for experimentation and direct access to internal components. For the beta phase, the housing will be 3D printed to ensure accurate measurements while being cost effective. For the production phase, extensive material testing is needed to realize the most optimal housing in terms of impact absorption, heat dissipation, environmental resistance, and weight.

3.1 Layout

The layout of the components within the housing is important for two reasons. The first reason being, that the placement of components will help with the setup of mounting the module onto the light pole, as well as help with the maintenance. Secondly, misplacement of components within the housing may lead to thermal issues. This is particularly important since the housing will need to be sealed to prevent foreign bodies from entering the enclosure. These design choices for the optimized layout of components are described in Table 3.1.1.

Table 3.1.1: Layout Design Specifications

Design ID	Design Description	Requirement ID Reference
Des 3.1.1-A	The layout of the components within the housing will be laid out so that there are no thermal issues.	N/A

The camera, microphone, and LED indicator are exposed from the housing to observe the parking lot. Therefore, the microcontroller should be placed in a way that all components can be connected to prevent clutter to ease maintenance tasks. The camera and microphone will be facing the front of the module and the LED indicator will be on the bottom of the module to maximize visibility. An example of this is shown in Figure 3.1.1.

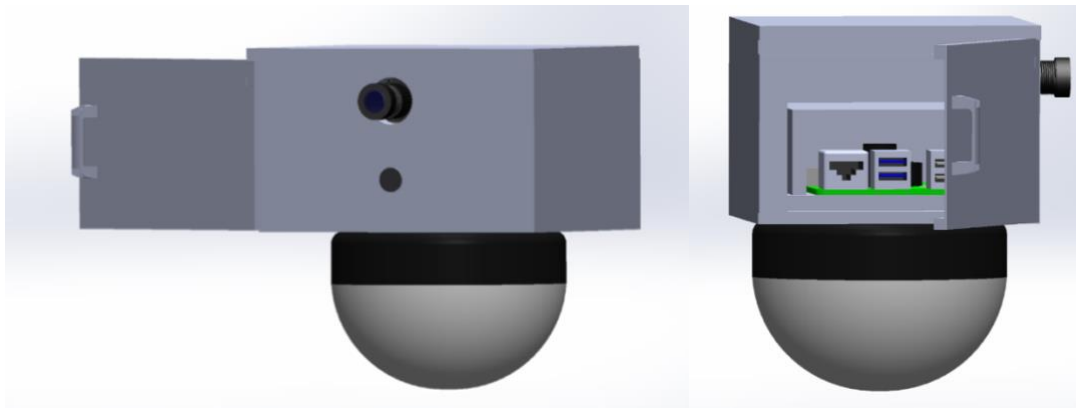


Figure 3.1.1: OpenSpot Module Model

3.2 Mounting

The module will be mounted by connecting the housing to a bracket, which is then connected to adjustable clamps. This will allow the entire system to be mounted in the air while connected to a light pole. The clamps should be adjustable and sturdy so that it is modular and fits around the diameter of different light poles. It should also be resistant to external factors to prevent corrosion. The mount will also support a mechanism to tilt the module vertically for a better angle. All these factors are considered in Table 3.2.1.



Table 3.2.1: Mounting System Design Specifications

Design ID	Design Description	Requirement ID Reference
Des 3.2.1-B	The mounting system will be able to hold the weight of the housing and components while also being modular to fit around any size pole.	Req 3.3.2-P Req 3.4.1-P Req 3.4.2-P
Des 3.2.2-B	The mounting system will connect to the housing and allow the module to tilt vertically.	N/A
Des 3.2.3-P	The mounting system should be resistant to external factors such that the entire system will not be compromised and fall from the light pole. This includes temperature changes, harsh weather, and animals.	Req 3.1.4-P Req 3.3.1-P

No mounting system is needed for the alpha phase since we will not be mounting our PoC. For the beta phase, an adequate mounting system is needed for realistic testing of the OpenSpot module. An industrial grade mount would be needed for the production phase since the module will be mounted on a light pole for the product lifespan.

4 OpenSpot Module

The OpenSpot module is made up of sensors and a microcontroller that work together to collect and distribute data. This section will detail the attributes and design choices involved with each component of our module.

4.1 Peripherals

4.1.1 Camera

The camera captures the images to be analyzed by our computer vision program. It will take a picture every 30 seconds to provide adequate parking updates. To take an image with the Raspberry Pi, the KEYESTUDIO camera module has been selected as it is compatible with our microcontroller and meets our design specs. One important characteristic for the camera was to have a focal length between 2-12mm as this threshold would provide most optimal field of view (FOV). Our selected camera has a 65° FOV and will be able to capture the parking spots with detail. The camera will be connected to the Raspberry Pi via the CSI camera connector port on the board. Table 4.1.1.1 list the camera’s specifications, Figure 4.1.1.1 shows the camera module, and

Table 4.1.1.2 list the design specifications.

Table 4.1.1.1: Camera Specifications [6]

Parameter	Value
Working Voltage	DC 3.3 V
Focal Length	3.29
Aperture	2.9
FOV	65 degrees
Dimensions	25 mm x 24mm x 9mm
Maximum Photo Resolution	5 MP
Video Resolution	1080p at 30fps

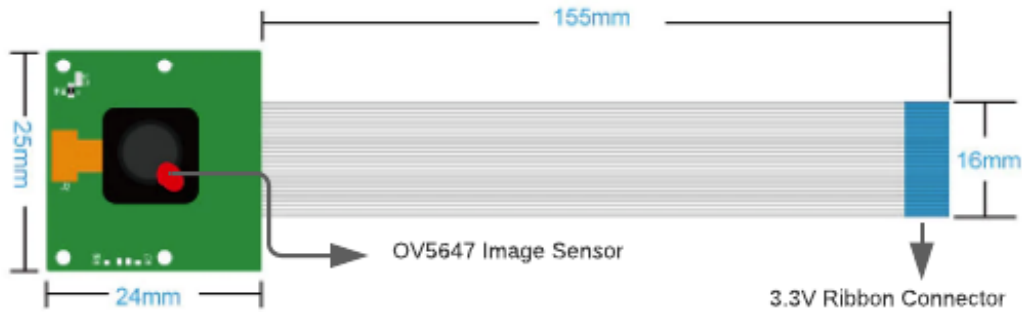


Figure 4.1.1.1: KEYESTUDIO Camera Module

Table 4.1.1.2: Camera Design Specifications

Design ID	Design Description	Requirement ID Reference
Des 4.1.1.1-A	The camera can capture a colored image during any time of day with low noise	Req 4.1.2.1-A Req 4.1.2.3-A Req 4.1.2.4-B Req 4.1.2.5-B
Des 4.1.1.2-B	The focal length will not exceed 12 mm and the aperture must not exceed f/4	Req 4.1.2.7-P Req 4.1.2.8-P:
Des 4.1.1.3-A	After taking an image, the camera will send the output back to the microcontroller which will in return forward it to our backend server for analysis	N/A

4.1.2 Microphone

The microphone will be placed in the module facing the direction as the camera to maintain consistent coverage. The microphone must have a detection radius of at least 10m away to cover an entire section of the lot, while being mounted high up on a lamp post. Furthermore, the microphone must be able to filter out background and ambient sounds so that accurate sound waves can be sent back to the audio recognition application.



Table 4.1.2.1: Microphone Design Specification

Design ID	Design Description	Requirement ID Reference
Des 4.1.2.1-A	The microphone will be able to take sounds within a 10m radius as input and function from about 3.5 meters off the ground.	Req 4.1.4.2-B Req 4.2.4.1-P
Des 4.1.2.2-A	After receiving an input sound, the microphone will send a noiseless output back to the microcontroller to be analyzed by the sound recognition application	Req 4.1.4.1-A Req 4.1.4.3-B Req 4.1.4.4-B

We will use a standard USB microphone for the Raspberry Pi 4 during the beta phase. By using Raspberry Pi specific components, it allows for seamless integration and easy configuration. Its compact size allows for easy transportation as well as the ability to fit into our enclosure designs.



Figure 4.1.2.1: USB Raspberry Pi Microphone

Table 4.1.2.2: Microphone Specification

Parameter	Value
Cable Length	1.5 meters
Size	6.5cm x 0.7 cm
Response Pattern	Omnidirectional
Connection Type	USB 2.0
Frequency Response	50Hz – 16 kHz
Sampling Rate	44.1 kHz/44.8 kHz
Sensitivity	-38 dB ± 3 dB

4.1.3 LED Indicator

The LED indicator displays real-time data that aids drivers in the parking lot. This allows drivers to focus on driving rather than using their mobile devices. The website should serve as an aid in the first step before driving or as a last resort if the entire parking lot is full. With the use of colors, the LED indicators will be able to provide different information to drivers.

- Green → more than 50% of spots are vacant.
- Yellow → less than 50% of spots are vacant.
- Red → less than 5% of spots are vacant.
- Flashing blue → Car alarm detected in the area.



At a height of 3.5m high, the LED indicator should be visible to the driver in their car and be bright enough to be seen in various weather conditions. Furthermore, the LED is also a part of the security system that OpenSpot provides with the ability to flash if a potential break in has been detected.

Table 4.1.3.1: LED Indicator Design Specification

Design ID	Design Description	Requirement ID Reference
Des 4.1.3.1-A	The ability to change the color of the LED will aid drivers in distinguishing the status of the parking spots in front of the OpenSpot module.	Req 4.1.3.1-A Req 4.1.3.2-A
Des 4.1.3.2-A	Sunlight should not impair the ability to view the color of the LED while being mounted 3.5m off the ground.	Req 4.1.3.1-A Req 4.2.3.1-P
Des 4.1.3.3-B	The LED will flash blue if a break-in is detected by the system.	Req 4.1.3.1-A Req 4.1.3.3-A

The LED indicator to be used is the RGB PixelDMX LED. This light module provides direct DMX technology which removes the needs for pixel decoders as they are already integrated within the module. It also provides all the colors we need and more. We can connect the light to our microcontroller with a DMX to USB cable and control it through the Open Lighting Project library. The specifications are shown in the following table.



Figure 4.1.3.1: PixelDMX LED

Table 4.1.3.2: PixelDMX LED Specifications

Parameter	Value
Size	133.4mm x 100mm x 82.6mm
Input Voltage	24 V
Power	3.5 W
Color Wavelength	Red (623nm), Green (522nm), Blue(472nm)
Expected Lifetime	50,000 hours
Protection Rating	IP65



4.1.4 Network Module

The Raspberry Pi 4 supports 2.4GHz and 5.0GHz wireless LAN. For our PoC and engineering prototype we plan to connect the Raspberry Pi to a mobile hotspot to facilitate the data exchange between our peripherals and backend server. For the production phase we have explored the option of using a modem kit compatible with the Raspberry Pi which would allow the microcontroller to have its own network connection. The modem kit supports the use of a sim card, allowing the Raspberry Pi to have its own 3G/4G – LTE network.

Table 4.1.4.1: Network Module Design Specifications

Design ID	Design Description	Requirement ID Reference
Des 4.1.4.1-P	The microcontroller must have its own 3G/4G -LTE network access	N/A

4.2 Hardware Design

4.2.1 Microcontroller

All components mentioned above will need to relay their information through the microcontroller so that data can be sent to and received from our servers. We have elected to use the Raspberry Pi 4, since it is reliable and supports a wide variety of component interfaces, which makes it suitable for our system. We chose to go with the 8GB ram model as this would provide for faster load times and allow us to run larger programs if needed. Furthermore, the Raspberry Pi 4 was chosen since it features a 2-lane MIPI CSI camera port and supports 2.4 GHz wireless LAN. The camera port allows for use of the Raspberry Pi camera module making the system cohesive. Wireless LAN will allow our microcontroller to successfully send data to our backend server to computation.

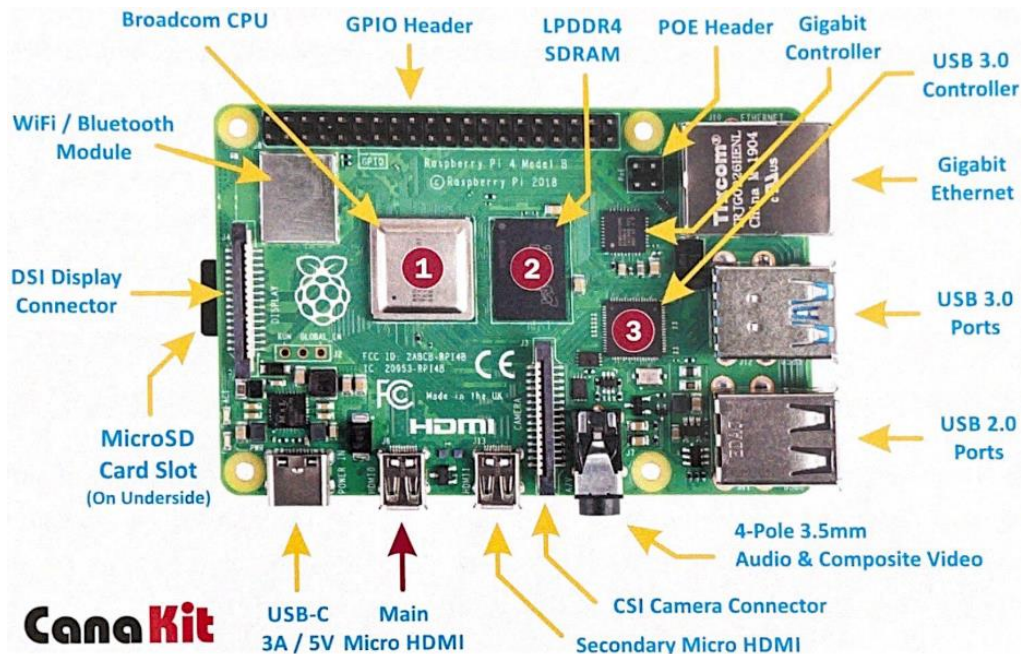


Figure 4.2.1.1 Raspberry Pi 4 Layout [7]



Table 4.2.1.1: Raspberry Pi 4 Specifications [8]

Parameter	Value
Dimensions	85mm x 56mm x 17mm
Weight	45g
Processor	Broadcom BCM2711, Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
Memory	8GB LPDDR4- 3200 SDRAM
Access	Extended 40-pin GPIO header
SD card support	Micro SD format for loading operating system and data storage
Connectivity	2.4GHz and 5.0GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE Gigabit Ethernet 2 x USB 3.0 ports and 2 x USB 2.0 ports 2 x micro-HDMI ports (up to 4kp60)
Video & sound	2-lane MIPI CSI camera port 4-pole stereo audio and composite video port
Input Power	5V DC via-USB-C connector 5V DC via GPIO header
Environment	Operating temperature, 0–50°C

Table 4.2.1.2: Microcontroller Design Specifications

Design ID	Design Description	Requirement ID Reference
Des 4.2.1.1-A	The microcontroller must be compatible with our LED light, camera, and microphone	Req 4.1.1.1-A Req 4.1.1.2-A Req 4.1.1.3-A
Des 4.2.1.2-B	The microcontroller must be able to update the LED in real-time	Req 4.1.1.2-A
Des 4.2.1.3-A	The MCU will support 2.4GHz wireless LAN.	N/A
Des 4.2.1.4-A	The microcontroller shall be able to communicate with our backend server	N/A

The Raspberry Pi features a powerful row of general-purpose input/output (GPIO) pins. There are two 5.5V pins and two 3.3V pins, as well as several ground pins 0V. Devices requiring more than 3.3V of power will not be connected to the microcontroller as it may damage the pins. Figure 4.3 shows the pin layout for the Raspberry Pi Model 4.



Table 4.2.2.1: Power Supply Design Specification

Design ID	Design Description	Requirement ID Reference
Des 4.2.2.1-A	The power supply will be portable and be able to provide the OpenSpot module up to 24 volts.	Req 3.5.1-B
Des 4.2.2.2-A	The power supply will be able to provide the OpenSpot module up to 2 amps.	Req 3.5.1-B
Des 4.2.2.3-B	The batteries will be able to last at least 6 hours before the need to recharge.	N/A
Des 4.2.1.4-B	The batteries will not charge and discharge at currents higher than 0.15C for extended periods of time to avoid damage	N/A
Des 4.2.1.5-P	The transformer will step down a 120V supply to provide 12V to the OpenSpot module.	N/A

4.2.3 Microcontroller Power

The Raspberry Pi 4 requires a 5.0V power supply with 3.0A current capacity [10]. For our testing purposes, we will be connecting the Raspberry Pi to a portable power supply then mounting it to a pole. We must ensure that the portable power supply will provide between 5 to 24V so all our sensor power requirements are met.

The camera module requires 250mA, LED requires approximately 145mA, and the microphone requires at least 2.7V. The GPIO pins can draw 50mA safely, distributed across all pins; an individual pin can safely draw only 16mA [10]. The max power output of the USB ports is 1200mA total across all ports. Since there are 4 ports each has the capacity to supply 300mA at any given time.

According to the Raspberry Pi website typical bare-board active current consumption is 600mA [10]. Thus, the power consumed by the microcontroller is shown below.

$$\text{Power (consumed by Raspberry Pi)} = V \times I = 600\text{mA} \times 5.0\text{V} = 3\text{W}$$

The portable power supply is said to have a battery capacity of 7Wh. This may vary depending on the actual environment of usage.

4.2.4 Camera Power

The camera module takes about 200 to 250mA [11]. Assuming this is measured off the 5V USB input to the Pi, we assume the following.

$$\text{Minimum Power (used by the camera module)} = V \times I = 0.20\text{A} \times 5.0\text{V} = 1.0\text{W}$$

$$\text{Maximum Power (useb by the camera module)} = V \times I = 0.25\text{A} \times 5.0\text{V} = 1.25\text{W}$$

4.2.5 LED Power

The LED dome light we have selected requires a 24 V input and consumes about 145mA as specified on their website [12]. Using this we can perform a simple power calculation.

$$\text{Power (used by the LED)} = V \times I = 0.145\text{A} \times 24.0\text{V} \approx 3.5\text{W}$$



4.2.6 Microphone Power

The microphone requires a 5V input and uses 150mA. Using this we can calculate the power consumption.

$$\text{Power (used by the microphone)} = V \times I = 0.150A \times 5.0V = 0.75W$$

5 Microcontroller Software Design

5.1 Peripheral Control

The peripherals are controlled by a Python script which runs a time-based loop. Figure 5.1.1 shows the main flow of our script. It ensures that an audio recording has started and periodically takes pictures of the parking lot. The length of each audio recording will be 30 seconds and pictures will be taken every 30 seconds¹. The indicator light only changes when a response from the server is received. Table 5.2.1 contains the design specifications for controlling the peripherals using the Python script.

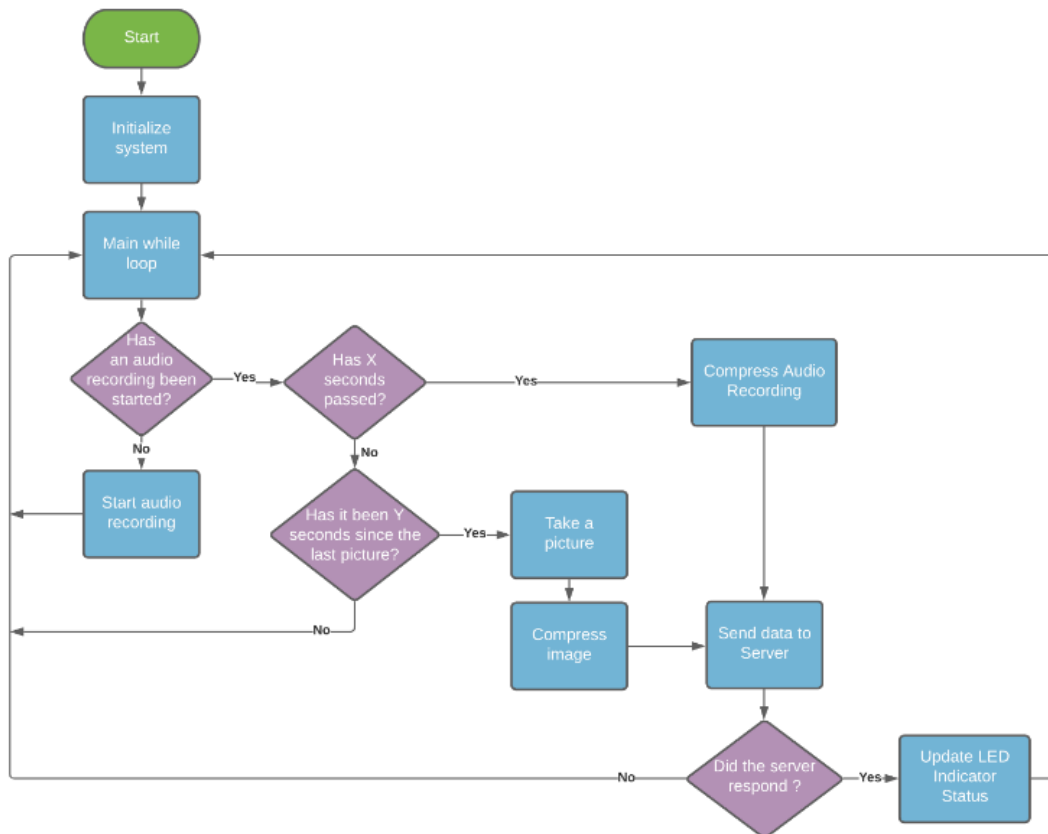


Figure 5.1.1: Module Script Flowchart

¹ Length of recording and period of pictures are subject to change from further experimentation and optimization.



5.2 Communication

Communication between the Raspberry Pi and our backend server requires a network connection. With our modules being out in parking lots, a network module (specified in 4.1.4 Network Module) is required for it to send and receive requests. Communication for images will be facilitated using imagezmq [13] as it provides an easy-to-use image transport mechanism for a distributed image processing network. Other communications will use Python's http.server library to send and receive HTTP requests with our backend webserver [14]. All communication design specifications are specified in Table 5.2.1.

Table 5.2.1: Module Script Design Specifications

Design ID	Design Description	Requirement ID Reference
Des 5.1-A	The Python script will facilitate control over the peripherals depending on the responses from the server. Using a time-based loop, audio recordings and images will be taken periodically.	Req 5.1.1-A Req 5.1.2-A Req 5.1.3-A Req 5.1.6-B Req 5.1.7-B Req 5.1.9-B Req 5.2.1-A
Des 5.2-A	The Python script running on the Raspberry Pi will run a webserver to communicate with the backend server using HTTP requests.	Req 5.1.4-A Req 5.1.5-A Req 5.1.8-B Req 5.2.2-B
Des 5.3-B	Data will be deleted locally immediately after it is sent to the server	Req 5.1.10-B Req 5.2.4-B
Des 5.4-B	The script size will be lean, and files will be cleaned up to reduce program size.	Req 5.2.3-B
Des 5.5-B	Starting or restarting the script will be simple and easy to do with minimal start up and set up required.	Req 5.2.5-B

6 Backend Server Design

6.1 Computer Vision (CV)

After receiving images from the module, the backend server will utilize Python libraries from OpenCV and TensorFlow to perform computational vision and determine instances of cars. We plan to use a Mask R-CNN model that is pretrained from the MS COCO database [1] [15]. Some preprocessing on the input images is needed to assist with accurate image segmentation. The scikit-image Python library will be used to read and perform image processing on our parking lot images as it provides a wide variety of image processing algorithms and routines. A summary of the workflow from input to output for the CV application is described in Figure 6.1.1. Figure 6.1.2 demonstrates an example output of our current CV system.

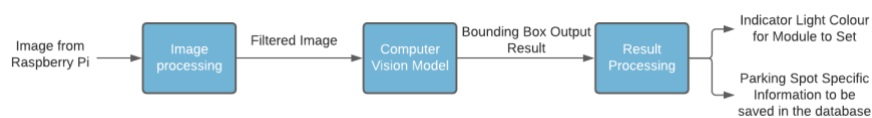


Figure 6.1.1: Computer Vision Flowchart



Figure 6.1.2: Sample Instance Segmentation, Bounding Boxes, and Classification

Our current design requires a reference photo to identify the spots in a parking lot. The reference photo is an image taken from the mounted position of the module with all the parking spots occupied in its FOV. Our algorithm then runs the reference photo through our CV model to determine the bounding boxes of each car in the photo. The bounding boxes from the reference photo represent the parking spots in the module's FOV. This reference photo is compared to the real time photos of the parking lot to determine vacancy. An overlap of over 60% of their bounding boxes will deem the parking spot to be occupied and anything less will be deemed vacant². The center location of each bounding box is also considered in the algorithm. From Figure 6.1.3, the reference photo identifies the bounding boxes of the parking spots in red (indicating the spot is occupied). The example real time photo shows the right parking spot empty and is indicated by a green bounding box.

² Percentage decision of overlap is subject change.



Figure 6.1.3: Reference Photo (left) and Example Real Time Photo (right)

Table 6.1.1: Computer Vision Design Specifications

Design ID	Design Description	Requirement ID Reference
Des 6.1.1-A	The CV application will use a pre-trained mask R-CNN model.	Req 6.1.1-A Req 6.1.3-A
Des 6.1.2-B	The CV application will run on an AWS instance with enough resources to compute the correct response efficiently.	Req 6.2.1-B

6.2 Audio Recognition

The audio recognition application will be responsible for handling audio recordings received from the microphone and analyzing it for the presence of a car alarm. This component of our system is key in handling the security aspect of OpenSpot which allows clients to be alerted if a persistent car alarm has been detected. The audio recognition software will filter and analyze recordings to detect sound waves which are consistent with impulsive sounds such as car alarms (i.e., frequency, wavelength, and amplitude) [16]. Once an alarm is detected, an alert will be relayed to the website so that it may inform the client that there is an active alarm. Furthermore, the audio recognition application will relay a request to the microcontroller to modify the state of the LED to display flashing a blue light to indicating an active alarm.

Table 6.2.1: Audio Recognition Design Specifications

Design ID	Design Description	Requirement ID Reference
Des 6.2.1-B	The car alarm audio recognition application will run on our backend server	N/A
Des 6.2.2-B	The audio recognition application will detect the presence of a car alarm by analyzing a received microphone recording	N/A
Des 6.2.3-B	The audio recognition application will relay an alert to the website upon the presence of an active car alarm	Req 7.2.4-B
Des 6.2.4-B	When an active car alarm has been detected, the audio recognition application will relay a request to the microcontroller to modify the state of the LED lights	Req 4.1.1.2-A Req 5.2.1-A



6.3 Communication

The web framework used for our website will use Django hosted on an Amazon Web Services (AWS) instance. By using cloud computing on AWS, it increases the availability and reliability of our server. In addition, AWS will allow us to have access to more powerful resources and perform computations faster to scale up our operations. The Django application will respond to web requests and communicate with our database to retrieve any necessary information.

Table 6.3.1: Server Communication Design Specifications

Design ID	Design Description	Requirement ID Reference
Des 6.3.1-B	The Django application will be hosted on an AWS server	N/A
Des 6.3.2-B	The Django application will respond to web requests and communicate with the database	Req 6.1.7-B

6.4 Database

Our database will be hosted on MongoDB Atlas which is a free to use cloud-based service. MongoDB uses a NoSQL database which will be useful for our system since we do not require usage of relational data. Further, MongoDB uses a JSON schema which allows for simplicity in structure and can be easily understood. Since it is based on a document data model, it will allow our team to store and retrieve data in an efficient manner. Some additional benefits including MongoDB's ability to secure sensitive data which will be useful as we store information regarding parking spots.

Table 6.4.1: Database Design Specifications

Design ID	Design Description	Requirement ID Reference
Des 6.4.1-B	The database will be hosted using MongoDB atlas	N/A
Des 6.4.2-B	The database will store information regarding parking spot locations, occupied/unoccupied stalls, and stall usage.	Req 6.1.2-B Req 6.1.8-B Req 6.1.9-B Req 6.1.10-B

7 Website Design

The front-end application for our smart parking module will be a website which will be compatible with browsers such as Safari, Google Chrome, and Firefox. In addition, the websites can be accessed through the latest mobile devices that have internet capabilities. The learning curve to using the website will be minimal, as it will adhere to industry usability standards. This makes our front-end application available to all age ranges and all levels of technology capabilities.

Table 7.1 highlights the website design requirements for our smart parking system. All the requirements for the website are set to be accomplished in the beta phase, with the exception of two requirements scheduled for the production phase. Our smart parking module has a lot of hardware and back-end components that need to function well together, which led to our team making the website a priority for the beta phase once our foundation is set. Hence all the design requirements you will see below are beta phase and after only.



For our website we will be using Django web framework, with REACT as our front-end. The main component of the website will be the visual display of our parking lot and the indication of availability of each individual parking spot.

Table 7.1: Website Design Specifications

Design ID	Design Description	Requirement ID Reference
Des 7.1-B	Website will have a latency of 30 seconds max for parking spot status at a respective lot	Req 7.1.1-B
Des 7.2-B	The website will display the parking lot with the appropriate status per parking spot	Req 7.1.2-B Req 7.1.3-B
Des 7.3-B	The website will allow for user input of their phone number for notification service	Req 7.1.5-B
Des 7.4-B	Website will have a tab of all parking lots that have our module installed	Req 7.1.6-B
Des 7.5-B	Parking Lot Owners (clients) will be able to Register/Log in to their account	Req 7.2.1-B Req 7.2.2-B
Des 7.6-B	Clients will receive notification if a car alarm is ringing at their parking lot	Req 7.2.4-B
Des 7.7-B	Clients will receive notifications if the module is damaged or not functioning at 100%	Req 7.2.5-B
Des 7.8-B	Busy time will be listed for each respective lot. 'i.e., parking lot X is most busy from 3-6pm'	Req 7.3.1-B
Des 7.9-P	Clients will be able to access a database of our camera feed footage	N/A
Des 7.10-P	Price information will be available on our website	Req 7.3.2-P
Des 7.11-P	Potential clients can contact our team through a support portal on the website	Req 7.3.3-P



8 Conclusion

OpenSpot provides consumers with a system which allows them to save time parking and a more efficient parking experience. We have not received any feedback regarding our requirement specification document but have received some comments during the first and second progress review. A minor area of concern that was expressed by Dr. Craig Scratchley included our ability to recognize parked cars within stalls under harsh weather conditions, such as snow or thunderstorms. After some research, our team decided that using median filtering on images captured by our camera would provide the most appropriate solution in this case.

Below is a summary regarding the sections of our design document:

1. Physical Design
 - a. Layout:
 - i. The OpenSpot module will be elevated on a light pole at an average height of approximately 3.5m
 - ii. The components within the housing unit will be laid out in a manner to mitigate any potential thermal issues
 - iii. The camera, microphone, and LED indicator will be exposed from the housing unit to observe the parking lot with the microcontroller (a Raspberry Pi 4) placed internally within the module
 - b. Mounting:
 - i. The housing will be connected to a bracket which itself will be attached to adjustable clamps to be mounted on the light pole
 - ii. The mounting system will be able to hold the weight of the housing unit, allow the module to tilt vertically, and withstand external factors including harsh weather or animals
 - iii. The mounting system itself will only be necessary for the production phase
2. OpenSpot Module
 - a. Camera:
 - i. The KEYESTUDIO camera module will be used to take a picture of parking stalls every 30 seconds
 - ii. After taking a picture, the image will be sent to the microcontroller which in turn will forward this to our backend server for analysis through our computer vision application
 - b. Microphone:
 - i. The outdoor CCTV microphone used will be able to detect sounds within a 10m radius
 - ii. The microphone will be used to capture an audio recording will in turn be relayed by the microcontroller to our backend server for analysis in detecting the presence of a car alarm using our audio recognition application
 - c. LED indicator:
 - i. The LED indicator will be used to indicate the following states to drivers within parking lots:
 1. Green: more than 50% of spots are vacant



2. Yellow: less than 50% of spots are vacant
 3. Red: less than 5% of spots are vacant
 4. Flashing blue: car alarm detected
 - ii. The LED indicator used will be an RGB PixelDMX light
 - d. Network Module:
 - i. The Raspberry Pi 4 used will be able to have access to a 3G/4G – LTE network for the production phase
 - e. Microcontroller:
 - i. The Raspberry Pi 4 used as our microcontroller will be compatible with our LED light, camera, and microphone
 - ii. It will be able to update our LED in real-time and have two-way communication with our backend server
 - f. Power:
 - i. For the alpha and beta phase, the Raspberry Pi 4 will be connected to a portable power supply
3. Microcontroller Software Design:
 - a. A python script will be used to control the peripherals and using time-based loops will capture audio recording and images
 - b. The python script will also run a webserver to communicate with the backend server using HTTP protocol
4. Backend Server Design:
 - a. Computer Vision:
 - i. The computer vision application will use a pre-trained Mask R-CNN model to analyze images of parking stalls
 - b. Communication:
 - i. A Django application hosted on an AWS server will be used to facilitate communication between our website and backend server
 - c. Database:
 - i. We will be using MongoDB Atlas as our database to store parking lot information
5. Website Design:
 - a. Our website will be built using the Django web framework and REACT as our front-end
 - b. The website will be used as the primary interface for users and clients to access information regarding parking lots such as the state of each stall, notification of an active alarm, and busy times



9 References

- [1] G. G. P. D. R. G. Kaiming He, "Mask R-CNN," 24 Jan 2018. [Online]. Available: <https://arxiv.org/abs/1703.06870>. [Accessed June 2021].
- [2] "MongoDB Atlas," [Online]. Available: <https://www.mongodb.com/cloud/atlas>. [Accessed July 2021].
- [3] IBM, "IBM Global Parking Survey: Drivers Share Worldwide Parking Woes," IBM, 2011. [Online]. Available: <https://newsroom.ibm.com/2011-09-28-IBM-Global-Parking-Survey-Drivers-Share-Worldwide-Parking-Woes-1>. [Accessed 2 July 2021].
- [4] S. Varrella, "Number of motor vehicle thefts in Canada in 2019, by province or territory," Statista, 9 March 2021. [Online]. Available: <https://www.statista.com/statistics/524802/canada-number-of-motor-vehicle-thefts-by-province-or-territory/>. [Accessed 3 July 2021].
- [5] "6 Things You Should Know About Street Light Poles," 14 February 2019. [Online]. Available: <https://www.greatbasinlighting.com/blog/6-facts-about-decorative-lighting-poles-california/>. [Accessed 7 July 2021].
- [6] KEYESTUDIO, "KEYESTUDIO," [Online]. Available: <https://www.keyestudio.com/products/keyestudio-5-megapixels-1080p-mini-camera-video-module-for-raspberry-pi-model-a-b-b-pi-2-and-raspberry-pi-3>. [Accessed 2021 7 July].
- [7] Canakit, "Getting Started," in *Canakit Raspberry Pi 4 Quick-Start Guide*, North Vancouver, p. 3.
- [8] Raspberry Pi, "Raspberry Pi," [Online]. Available: <https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/>. [Accessed 7 July 2021].
- [9] Raspberry Pi, "Raspberry Pi," [Online]. Available: <https://www.raspberrypi.org/documentation/usage/gpio/>. [Accessed 8 July 2021].
- [10] Raspberry Pi, "Raspberry Pi," [Online]. Available: <https://www.raspberrypi.org/documentation/hardware/raspberrypi/power/README.md>. [Accessed 8 July 2021].
- [11] Jamesh, "Raspberry Pi," 30 July 2011. [Online]. Available: <https://www.raspberrypi.org/forums/viewtopic.php?t=152864>. [Accessed 9 July 2021].
- [12] "RGB PixelDMX LED Dome Module," Environmental Lights, [Online]. Available: <https://www.environmentallights.com/16037-rgb-pixel-dmx-dome-100.html>. [Accessed 10 July 2021].



- [13] J. Bass, "Transporting OpenCV Images via ZMQ," [Online]. Available: <https://pypi.org/project/imagezmq/>. [Accessed June 2021].
- [14] "http.server — HTTP servers," [Online]. Available: <https://docs.python.org/3/library/http.server.html>. [Accessed July 2021].
- [15] "Mask R-CNN for Object Detection and Segmentation," [Online]. Available: https://github.com/matterport/Mask_RCNN/releases. [Accessed June 2021].
- [16] Y. Arslan, "A New Approach to Real Time Impulsive Sound Detection for Surveillance Applications," June 2019. [Online]. Available: https://www.researchgate.net/publication/333841929_A_New_Approach_to_Real_Time_Impulsive_Sound_Detection_for_Surveillance_Applications. [Accessed July 2021].
- [17] "5 of the Best Materials for Outdoor Projects," Azure, [Online]. Available: <https://www.azuremagazine.com/article/best-materials-for-outdoor-projects/>. [Accessed 10 July 2021].
- [18] C. P. SOLUTIONS, "3 Thermal Management Approaches for Your Smartphone," [Online]. Available: <https://resources.pcb.cadence.com/blog/3-thermal-management-approaches-for-your-smartphone-2>. [Accessed 10 July 2021].
- [19] "Security Camera Pole Mount Bracket," CCTV Camera World, [Online]. Available: <https://www.cctvcameraworld.com/pole-mount-bracket-ptz.html>. [Accessed 10 July 2021].
- [20] "2-3/8" Round Post Sign Bracket For 1 Sign," SafetySign, [Online]. Available: <https://www.safetysign.com/products/8698/round-post-bracket>. [Accessed 10 July 2021].
- [21] "LED Andon Stack Light, Steady," LED Andon, [Online]. Available: https://www.ledon.com/stack_lights/LD-52XX-100.html?gclid=CjwKCAjw55-HBhAHEiwARMCszsK-6iRc_yc3pO1c7TWK0jKa2y2Edo9dShv21Yuk5oOEVQ5uDb7xuBoCeVoQAvD_BwE. [Accessed 10 July 2021].
- [22] "7-Segment Display - 6.5" (Red)," SparkFun, [Online]. Available: <https://www.sparkfun.com/products/8530>. [Accessed 10 July 2021].
- [23] "AOM-4544P-2-R," Digi-Key, [Online]. Available: <https://www.digikey.ca/en/products/detail/pui-audio/AOM-4544P-2-R/1745492>. [Accessed 10 July 2021].
- [24] "Tonton High Sensitive Weatherproof Preamp Microphone Audio Pickup Device Sound Voice Pickup Kit with 60 Feet Cable and Power Supply for CCTV Surveillance Camera System," Amazon, [Online]. Available: https://www.amazon.ca/Tonton-Sensitive-Weatherproof-Preamp-Microphone/dp/B07D6J4XJZ/ref=cm_cr_arp_d_product_top?ie=UTF8&th=1. [Accessed 10 July 2021].



- [25] "Rode VideoMic GO Camera-Mount Shotgun Microphone," B&H, [Online]. Available: https://www.bhphotovideo.com/c/product/1012003-REG/rode_videomic_go_videomic_go_on_camera_shotgun.html/overview. [Accessed 10 July 2021].
- [26] "Databases on AWS," [Online]. Available: https://aws.amazon.com/products/databases/?nc2=h_ql_prod_db. [Accessed 10 July 2021].
- [27] "Cloud computing with AWS," Amazon Web Services, [Online]. Available: https://aws.amazon.com/what-is-aws/?nc1=f_cc. [Accessed 10 July 2021].
- [28] "Google cloud overview," Google, [Online]. Available: <https://cloud.google.com/docs/overview>. [Accessed 10 July 2021].
- [29] "What is Azure?," Microsoft, [Online]. Available: <https://azure.microsoft.com/en-us/overview/what-is-azure/>. [Accessed 10 July 2021].
- [30] B. Johnson, "Smart Parking Testimonials," [Online]. Available: <https://www.featuredcustomers.com/vendor/smart-parking/testimonials#read%20testimonial>. [Accessed 07 July 2021].
- [31] "Trustpilot," Smart Parking Technology, [Online]. Available: <https://uk.trustpilot.com/review/smartparking.com>. [Accessed 10 July 2021].
- [32] "ISO 9241-161:2016 Ergonomics of human-system interaction — Part 161: Guidance on visual user-interface elements," February 2016. [Online]. Available: <https://www.iso.org/standard/60476.html>. [Accessed June 2021].
- [33] "ISO 9241-220:2019 Ergonomics of human-system interaction — Part 220: Processes for enabling, executing and assessing human-centred design within organizations," March 2019. [Online]. Available: <https://www.iso.org/standard/63462.html>. [Accessed June 2021].
- [34] "IEEE/ISO/IEC 23026-2015 - ISO/IEC/IEEE International Standard - Systems and software engineering - Engineering and management of websites for systems, software, and services information," Mar 2015. [Online]. Available: <https://standards.ieee.org/standard/23026-2015.html>. [Accessed July 2021].
- [35] "Distracted Driving Laws & Penalties Per Province," 3 April 2021. [Online]. Available: <https://www.canadadrives.ca/blog/driving-tips/distracted-driving-laws-penalties-canada>. [Accessed June 2021].
- [36] "Distracted driving," [Online]. Available: <https://www.icbc.com/road-safety/crashes-happen/Distracted-driving/Pages/default.aspx>. [Accessed June 2021].
- [37] "MOTOR VEHICLE ACT [RSBC 1996] CHAPTER 318 Part 3.1 — Use of Electronic Devices while Driving," 16 June 2021. [Online]. Available:



https://www.bclaws.gov.bc.ca/civix/document/id/complete/statreg/96318_06. [Accessed June 2021].

- [38] "What is NoSQL?," mongoDB, [Online]. Available: <https://www.mongodb.com/nosql-explained>. [Accessed June 2021].
- [39] Wikipedia, "1," 27 July 2013. [Online]. Available: https://en.wikipedia.org/wiki/Progressive_scan. [Accessed 7 July 2021].
- [40] "USB Microphone for Raspberry Pi," MakerPortal, [Online]. Available: <https://makersportal.com/shop/usb-microphone-for-raspberry-pi>. [Accessed 10 July 2021].



10 Appendices

Appendix A: Supporting Test Plans

A.1 Introduction

A.1.1 Test Purpose

The purpose of our test plans is to make sure that our Proof-of-Concept prototype meets all our requirement specifications, in addition to all the design specifications stated in this document. If our designed module covers all stated requirements, then the OpenSpot team can successfully release our product prototype, without concern of functionality or failure.

A.1.2 Test Coverage

In our test plans we aim to cover all individual components individually and test how well they function. In addition, we must test that our back-end programs are working with our hardware and to check for any bugs or failures that could potentially occur. Once we have each individual component of the module tested properly, we will need to have some integration testing to verify the system functions correctly. With that said, our test plans only cover the PoC requirements, with some additional tests intended for our product release end of ENSC 440 (December 2021).

A.1.3 Test Methods

Our testing is dependent on visual inspection and confirmation of functionality. We would need to inspect each component to make sure it is functioning as expected and that the software is performing to our standards.

A.1.4 Test Responsibilities

All members at OpenSpot are responsible for testing the module, hardware, and software. However, we will have our software team focus on the back-end features while our hardware team focuses on the system components and module. Testing will be done through our test plans stated below.

A.2 Physical/Mechanical Testing

Test Name: Aluminum Stress Test (Production Phase)		Date:	Result: <input type="checkbox"/> Pass <input type="checkbox"/> Fail
Test Description	An aluminum box is created and dropped from 3 meters high. Spare electrical components will be placed inside the box to see how the material absorption will affect the components.		
Expected Results	Aluminum box will sustain few damages and electrical components will still be in one piece.		
Actual Results and Comments			



Test Name: Mount Stress Test (Production Phase)		Date:	Result: <input type="checkbox"/> Pass <input type="checkbox"/> Fail
Test Description	A prototype housing will be created and mounted to a pole using the mounting system for 3 hours. Mount and pole will be agitated in ways that replicate potential external factors.		
Expected Results	Mounting system should hold up prototype housing for entire time.		
Actual Results and Comments			

Test Name: Moisture/Water Test (Beta Phase)		Date:	Result: <input type="checkbox"/> Pass <input type="checkbox"/> Fail
Test Description	A spray bottle will be sprayed onto housing to test sealing ability of housing.		
Expected Results	Housing should be able to keep all if not most of the water out of the housing.		
Actual Results and Comments			

A.3 Electronics Testing

Test Name: MCU Power Restart Test		Date:	Result: <input type="checkbox"/> Pass <input type="checkbox"/> Fail
Test Description	Microcontroller is powered on while connected to all peripherals until the power supply dies. Temperature is also taken of the core components.		
Expected Results	Microcontroller powers down and powers up successfully.		
Actual Results and Comments			

Test Name: MCU Power Supply Test		Date:	Result: <input type="checkbox"/> Pass <input type="checkbox"/> Fail
Test Description	Microcontroller is powered on while connected to all peripherals for 3 hours. Voltage measurements are taken of the voltage supplied to each peripheral in the module.		
Expected Results	Microcontroller powers down and powers up successfully.		
Actual Results and Comments			



Test Name: Camera Test		Date:	Result: <input type="checkbox"/> Pass <input type="checkbox"/> Fail
Test Description	Camera can successfully take a picture.		
Expected Results	Camera can produce a clear and precise image.		
Actual Results and Comments			

Test Name: LED Stress Test		Date:	Result: <input type="checkbox"/> Pass <input type="checkbox"/> Fail
Test Description	LED is kept on for 6 hours to check for any reduction in brightness, excessive heat, or overall failure of the light.		
Expected Results	LED should stay on for the entire duration without failing or reducing in brightness.		
Actual Results and Comments			

Test Name: Microphone Distance Test		Date:	Result: <input type="checkbox"/> Pass <input type="checkbox"/> Fail
Test Description	Microphone will be turned on in a quiet environment. A tester will clap and will continuously move backward until no sound is recognized.		
Expected Results	Microphone can pick up sounds from up to 9 meters away.		
Actual Results and Comments			

A.4 Module Software Testing

Test Name: Connectivity Test		Date:	Result: <input type="checkbox"/> Pass <input type="checkbox"/> Fail
Test Description	Turn on the module and ping the microcontroller's IP address from the remote server.		
Expected Results	Server can successfully ping the module.		



Actual Results and Comments	
------------------------------------	--

Test Name: Taking a Picture	Date:	Result: <input type="checkbox"/> Pass <input type="checkbox"/> Fail
Test Description	The time-based loop is started on the module. The module will take a picture according to the period set.	
Expected Results	Module takes a picture based on the time-based loop and can be seen locally.	
Actual Results and Comments		

Test Name: Sending an Image to the Server	Date:	Result: <input type="checkbox"/> Pass <input type="checkbox"/> Fail
Test Description	Module compresses and sends the image to the server.	
Expected Results	Server receives the image from the module.	
Actual Results and Comments		

Test Name: Deleting Local Image After Receiving a Response from the Server	Date:	Result: <input type="checkbox"/> Pass <input type="checkbox"/> Fail
Test Description	The module receives a response from the server after sending an image. The script should then delete the locally stored image.	
Expected Results	Local image is deleted after receiving a response.	
Actual Results and Comments		

Test Name: Taking an audio recording	Date:	Result: <input type="checkbox"/> Pass <input type="checkbox"/> Fail
Test Description	The time-based loop is started on the module. The module will start and end an audio recording based on the length and period set.	
Expected Results	Audio recording can be seen locally after the recording has stopped.	



Actual Results and Comments	
------------------------------------	--

Test Name: Sending an Audio Recording to the Server	Date:	Result: <input type="checkbox"/> Pass <input type="checkbox"/> Fail
Test Description	Module compresses and sends the audio recording to the server.	
Expected Results	Server receives the audio recording from the module.	
Actual Results and Comments		

Test Name: Deleting Local Audio Recording After Receiving a Response from the Server	Date:	Result: <input type="checkbox"/> Pass <input type="checkbox"/> Fail
Test Description	The module receives a response from the server after sending an audio recording. The script should then delete the locally stored audio recording.	
Expected Results	Local audio recording is deleted after receiving a response.	
Actual Results and Comments		

Test Name: Setting Appropriate Light Colour (Green)	Date:	Result: <input type="checkbox"/> Pass <input type="checkbox"/> Fail
Test Description	The module receives a response from the server after sending an image of an empty parking lot. The script will then set the indicator light to green.	
Expected Results	Indicator light is set to the colour green.	
Actual Results and Comments		

Test Name: Setting Appropriate Light Colour (Yellow)	Date:	Result: <input type="checkbox"/> Pass <input type="checkbox"/> Fail
Test Description	The module receives a response from the server after sending an image of a parking lot that has 50% of the spots empty. The script will then set the indicator light to yellow.	
Expected Results	Indicator light is set to the colour yellow.	



Actual Results and Comments	

Test Name: Setting Appropriate Light Colour (Red)	Date:	Result: <input type="checkbox"/> Pass <input type="checkbox"/> Fail
Test Description	The module receives a response from the server after sending an image of a full parking lot (no vacant spots). The script will then set the indicator light to red.	
Expected Results	Indicator light is set to the colour red.	
Actual Results and Comments		

Test Name: Setting Appropriate Light Colour (Flashing Blue)	Date:	Result: <input type="checkbox"/> Pass <input type="checkbox"/> Fail
Test Description	The module receives a response from the server after an audio recording with a car alarm going off. The script will then set the indicator light to start flashing blue.	
Expected Results	Indicator light is flashing the colour blue.	
Actual Results and Comments		

A.5 Backend Server Testing

Test Name: Connectivity Test	Date:	Result: <input type="checkbox"/> Pass <input type="checkbox"/> Fail
Test Description	From a personal laptop ping the server.	
Expected Results	Server can be pinged successfully, verifying connectivity.	
Actual Results and Comments		

Test Name: Receiving an Image	Date:	Result: <input type="checkbox"/> Pass <input type="checkbox"/> Fail
Test Description	A compressed image is sent from the module to the server.	
Expected Results	Server receives a compressed version of the image.	



Actual Results and Comments	

Test Name: Receiving an Audio Recording	Date:	Result: <input type="checkbox"/> Pass <input type="checkbox"/> Fail
Test Description	A compressed audio recording is sent from the module to the server.	
Expected Results	Server receives a compressed version of the audio recording.	
Actual Results and Comments		

Test Name: Computer Vision – Detecting vacant and occupied spots (Empty Parking Lot)	Date:	Result: <input type="checkbox"/> Pass <input type="checkbox"/> Fail
Test Description	The server receives a compressed image of an empty parking lot.	
Expected Results	The computer vision application successfully computes the number of empty parking spots and responds to the module's request. Tells the module to set the light to green. Updates the database with the status of each parking spot.	
Actual Results and Comments		

Test Name: Computer Vision – Detecting vacant and occupied spots (50% of the Spots Vacant)	Date:	Result: <input type="checkbox"/> Pass <input type="checkbox"/> Fail
Test Description	The server receives a compressed image of a parking lot with 50% of the spots vacant.	
Expected Results	The computer vision application successfully computes the number of empty parking spots and responds to the module's request. Tells the module to set the light to yellow. Updates the database with the status of each parking spot.	
Actual Results and Comments		



Test Name: Computer Vision – Detecting vacant and occupied spots (Full Parking Lot)		Date:	Result: <input type="checkbox"/> Pass <input type="checkbox"/> Fail
Test Description	The server receives a compressed image of a parking lot with all spots occupied.		
Expected Results	The computer vision application successfully computes the number of empty parking spots and responds to the module’s request. Tells the module to set the light to red. Updates the database with the status of each parking spot.		
Actual Results and Comments			

Test Name: Audio Detection – Detecting Car Alarms (ambient noise)		Date:	Result: <input type="checkbox"/> Pass <input type="checkbox"/> Fail
Test Description	The server receives a compressed audio recording of ambient noise without any active car alarms.		
Expected Results	The audio detection application does not detect any car alarms and does not tell the module to change the light setting.		
Actual Results and Comments			

Test Name: Audio Detection – Detecting Car Alarms (Car Alarm Sound)		Date:	Result: <input type="checkbox"/> Pass <input type="checkbox"/> Fail
Test Description	The server receives a compressed audio recording of a car alarm.		
Expected Results	The audio detection application successfully detects the car alarm and send a response to the module to set the lights blue and flash. The database is updated to indicate that the module that sent the audio recording hears a car alarm.		
Actual Results and Comments			



Test Name: Responding to Web Requests		Date:	Result: <input type="checkbox"/> Pass <input type="checkbox"/> Fail
Test Description	The URL of the website is accessed, sending a request to the server.		
Expected Results	Webserver responds to the web request with information and data from the database. Website loads and data is viewed from the browser.		
Actual Results and Comments			

A.6 Website Testing

Test Name: Parking Map Display		Date:	Result: <input type="checkbox"/> Pass <input type="checkbox"/> Fail
Test Description	Connect to website and see 'X' parking lot displayed with empty/full parking spots.		
Expected Results	Once you connect to our website you can see the individual parking spot status at the respective parking lot chosen.		
Actual Results and Comments			

Test Name: Different Parking Lots		Date:	Result: <input type="checkbox"/> Pass <input type="checkbox"/> Fail
Test Description	Once connected to the website you should select through the list of parking lots available for selection.		
Expected Results	Scroll through a list of parking lots and selecting the one you want brings up the correct visual representation of the parking lot.		
Actual Results and Comments			



Test Name: Text Notification	Date:	Result: <input type="checkbox"/> Pass <input type="checkbox"/> Fail
Test Description	Select a specific parking lot and input your cellphone number into a text box.	
Expected Results	Receive text messages updating the status of the parking lot. If the lot is 75% full and/or if the parking lot is completely full.	
Actual Results and Comments		

Test Name: Client Log in/out	Date:	Result: <input type="checkbox"/> Pass <input type="checkbox"/> Fail
Test Description	Clients can log into the website and have their designated parking lots assigned to them. They can logout after and return to the normal webpage.	
Expected Results	Client can select the log in option and enter their username and password. Logging into the website with their unique information. After they can log out of the website and return to regular view	
Actual Results and Comments		

Test Name: Statistical View	Date:	Result: <input type="checkbox"/> Pass <input type="checkbox"/> Fail
Test Description	Clients can see stats about their parking lot, 'busy hours', 'how many cars are parked'	
Expected Results	Once a client logs in they can view the statistics offered by OpenSpot about their specific parking lot	
Actual Results and Comments		



Appendix B: Supporting Design Options

B.1 Physical

B.1.1 Body and Housing Structure

B.1.1.1 Exterior

Table B.1.1.1.1: Exterior Body and Housing Design Options

Design Options	Specification
PLA	PLA is one of the most popular materials for 3D printing. This is most likely what we would use for our Beta phase housing since it is cheap, and we will be able to test many different types of housing design. We will be able to create dimensionally accurate housing since we will be 3D printing. It is good for the purpose of creating a rigid housing module that can be moved around. For long term use however, it may not be preferable from outdoor use since they are sensitive to sun exposure.
Aluminum	Most weatherproof metal boxes for outdoors are usually made of aluminum. Aluminum is highly durable, has corrosion resistance, light in weight, inexpensive and low maintenance. It is also very easy to cut and form. However, aluminum retains heat easily so including proper thermal management is important to consider. Aluminum also contains highly recycled content and is generally recyclable [17].
Stainless Steel	Like aluminum, stainless steel is also very applicable in outdoor use. It has even better corrosion resistance when compared to aluminum, has a long lifecycle, very resistant to extreme temperature and is also very strong. However, stainless steel requires maintenance for the steel to maintain its appearance and structure. It is also more expensive and heavier when compared to aluminum. Stainless steel is also completely recyclable [17].
Plastic	Plastic is used for electrical boxes because they are nonconductive which means they do not have to be grounded. They can withstand the UV light, is durable, fade resistant, and lightweight. Plastic is also the cheapest when compared to the metals but is not as strong and durable. The main drawback is that plastic is not as recyclable when compared to the metals.

The exterior material is essential for the structural integrity of the module. However, we also must consider requirements 3.1.1-B, 3.1.4-P and 3.4.1-B when narrowing down our three main material options: aluminum, stainless steel, and plastic. We have decided to go with aluminum because not only is it highly accessible, but it is also relatively easy to mold and satisfies all three requirements. We choose aluminum over plastic because it is recyclable since sustainability is one of OpenSpot's priorities. The drawback of retaining heat can also be managed by heat management options which will be discussed in the next section.

B.1.1.2 Interior

The temperature within the interior of the OpenSpot module will need to be controlled since the system will be constantly on. We will most likely use the method of heat spreaders instead of heat sinks to reduce the number of components inside of the module. A plate of graphite will be used to draw heat away from the source and route it to the external chassis of the module. Since we will be using



aluminum for the exterior housing, it will be able to conduct the routed heat to the atmosphere. This design is used for most modern smartphone thermal management [18].

B.1.2 Mounting

Table B.1.2.1: Mounting Design Options

Design Options	Specification
Pole Mount Bracket [19]	The pole mount bracket is a straightforward mounting mechanism for the OpenSpot module. The module would have to be connected to the mounting bracket which can then be mounted around the pole using two steel straps. The module can be connected by drilling screws into the housing through the bracket holes.
Sign Mount Brackets [20]	Sign mount brackets can be used to essentially recreate pole mount brackets except with a separate mounting bracket. Using the specified models, multiple clamps are needed to secure itself around the pole to connect to a bracket that can hold up the module. Also, since it is made of aluminum, the initial bracket may be able to become drilled through to mount the housing.

A proper mounting system is required to secure the OpenSpot module safely in place. It should also be made of one of the specified materials for the housing. Considering the requirements 3.3.1-P, 3.3.2-P and 3.4.1-B, the pole mount bracket would be the best fit since it satisfies all three requirements. The straps are made of aluminum and as we discussed earlier, are optimal for outdoor use to ensure the safety of the module. Using the pole mount bracket, the mounting system will be straightforward.

B.2 Hardware

B.2.1 Microcontroller

Table B.2.1.1: Microcontroller Design Options

Design Options	Specification
Arduino Uno	This microcontroller is cost effective and satisfied most of our requirements, however, it did not have some key features. The Uno did not have MIPI CSI camera port which is needed for our camera to work. Additionally, we need to update our LED in real time to indicate parking density and run multiple programs at a time, with the Arduino's smaller processor this would be tough.
Raspberry Pi Pico	This microcontroller is tiny, fast, and portable; however, it does not support many modules such as the wireless LAN and MIPI CSI. For that reason alone, we voted against this option.
Raspberry Pi 4	This microcontroller satisfies all the requirements and provided supported for the camera module and all our additional peripherals.

From the 3 options listed above we chose to go with the Raspberry Pi 4 since it met all our system requirements. It is configurable since you can choose from various RAM storage options and supports a vast number of sensors and ports. Since we plan on running multiple programs at a time, the 64-bit quad-core processor paired with the 8GB of DDR4 ram will provide us the reliability and speed to perform computations.



B.2.2 Camera

Table B.2.2.1: Camera Design Options

Design Options	Specification
GoPro Hero 9 Black	This camera provides great image and video quality, while meeting all our environmental constraints, however, it is expensive and lacks compatibility with our microcontroller.
Logitech C270 Webcam	This camera has nice form factor but did not meet our image and video quality standards. It does not support 1080p, so it was voted against.
KEYESTUDIO Raspberry Pi Camera Module	The Raspberry Pi Camera Module is compatible with our microcontroller and supports 1080p quality. Seamless integration with our other components.

From the 3 options listed above, we chose to go with the KEYESTUDIO Raspberry Pi Camera Module. The KEYESTUDIO Raspberry Pi is compatible with the Raspberry Pi 4 and met our image quality standards. Since the module is supported by the Raspberry Pi 4, developing a program using it will be easy.

B.2.3 LED Indicator

Table B.2.3.1: Indicator Light Design Options

Design Options	Specification
RGB PixelDMX LED Module [12]	A RGB LED module would allow us to use three colors to represent real-time data of the parking lot. This model specifically has a long-life expectancy while also being rated IP65 waterproof which is ideal for outdoor use. It is also 100mm in width which when compared to other similar modules, is relatively large which is important for driver visibility. Furthermore, the lights are configurable which allows us to change the function of the LEDs if the security system detects a break-in.
3 Color LED Stack [21]	A LED three color stack light also allows us to represent real-time data of the parking lot. However, stack lights are much more obtrusive when compared to the LED module which may also be a benefit for driver visibility. This specific model is also long lasting as well as rated IP65. These stack lights are also configurable to allow us to change the function of the LEDs if the security system detects a break-in.
7 Segment Display [22]	A seven-segment display allows us to represent real-time data of the parking lot without the need for colors. It allows us to provide the exact number of spots in the area. This model is also directly configurable through the microcontroller. It is stated that it is visible from up to 100 feet away, perfect for our intended usage. However, the only downside is that not only is it obtrusive, but it may also not be able to be seen in certain angles of the parking lot.

The main purpose of the LED indicator is to show the real-time status for the area of the parking lot that the module is watching over. From this, we consider requirements 4.1.3.1-A and 4.2.3.1-P while selecting between our three choices: RGB LED Module, three color LED stack, and a seven-segment display. We have decided to choose the RGB PixelDMX LED Module for our indicator because not only is the light able to be viewed from all angles, but it is also suitable for outdoor use. Its more compact frame also allows ease of mounting and maintenance. The LED stack and the seven-segment display



would both require additional mounting. Overall, the RGB LED Module meets all the necessary requirements needed for its purpose and excels in areas that the other options do not.

B.2.4 Microphone

Table B.2.4.1: Microphone Design Options

Design Options	Specification
USB Microphone for Raspberry Pi	This microphone is directly compatible with the microcontroller we chose for alpha and beta phase. Therefore, we will most likely use this microphone to test the audio recognition of our security system. This model is also compact in size.
Electret Microphone [23]	The electret microphone is electrostatic capacitor-based microphone. This type of microphone is extremely compact and allows for a lot of space in the module and is extremely simple to set up. With this simplicity, there are not a lot of features that add to the quality of the audio taken. There is only a screen of fabric on the top of the front plate of the microphone to make it dust proof.
Ton Ton Outdoor CCTV Microphone [24]	The only purpose of the microphone in the module is to be used for the surveillance and security aspect OpenSpot can provide. It only makes sense that we investigate CCTV microphones. This model has high gain which allows audio to be picked up from about 9 metres away. It is also waterproof which makes it suitable for outdoor use as well.
Rode VideoMic Shotgun Microphone [25]	Shotgun microphones are ideal for capturing audio in open areas which makes it suitable for our purposes. This specific model is a simpler shotgun microphone which is plug and play for easy installation. Cheaper versions of shotgun microphones, such as this one, is not fully weatherproof and do not have many features that allow for greater quality of audio. However, they are of greater quality than the other two options.

The focus of the microphone is to capture audio outdoor in a section of the parking lot to detect possible car alarms. We consider requirements 4.1.4.2-B and 4.2.4.1-P for deciding on which microphone to use and we narrowed it down to three options: an electret microphone, a CCTV microphone, and a shotgun microphone. We decided upon choosing the outdoor CCTV microphone because it provides the necessary capture distance for the audio as well as is durable outdoors. It is also quite compact which allows placement into the module quite easily compared to the shotgun microphone, which is usually larger in size. We feel that the audio quality does not need to be quite as refined for the purpose of audio detection. Unnecessary noise may be able to be filtered out once the audio is sent to the server as well.

As stated, the USB microphone for the Raspberry Pi will most likely be used for the alpha and beta phase. This is so we do not have to worry about unnecessary configuration and can get straight into implementing and testing the module. Its compact size allows us to move the components around with ease for the alpha phase and is easy to fit inside prototype housing designs.



B.2.5 Power Source

Table B.2.5.1: Power Source Design Options

Design Options	Specification
MEGO Portable Power Supply	This power supply provides 4 to 24V output, has a 7Wh battery capacity, and is portable. It is also, cost effective and features a rechargeable battery making it an excellent choice for on the go use.
USB-C Adapter Power adapter	This method requires the need of a wall outlet, and you need more than one adapter. This because some of our peripherals require a 24V input and the Raspberry Pi will not provide more than 5V. Additionally, having to use a wall outlet is not an option for our system
Lithium-ion rechargeable battery	Lithium-ion batteries are lightweight and have high capacity. However, the battery is not a viable option for our system since the size of the battery would too large and it is cost a lot more than the other options.
Tapping into light post power	For future iteration, this method would be used since it would provide a constant supply of power and eliminate the need for recharging. More research and testing must be done with this method before implemented since we lack the knowledge of how commercial electric infrastructure works.

We have chosen to go with the MEGO portable power supply because it satisfies our systems power requirements while also being portable. This option is cost effective and gives us the freedom to test our system in a variety of environments without worrying about the need of an outlet.

The MEGO is an option that will be used for the alpha and beta phases; however, in the production phase the module will be connected directly to line power. A transformer would then be used to step down the supplied 120 volts to 12 volts for the MCU and other peripherals in the module.

B.3 Software

B.3.1 Module Communication and Software

Table B.3.1.1: Module Communication Design Options

Design Options	Specifications
Python Script	Writing our own script would allow our team to be in control of what is installed and added to our on-device software. It can be tailored and designed to be lightweight and efficient while providing all the necessary functions to meet our requirements.
Webserver Framework	Using a webserver framework would allow for future endeavors with new technologies. It is much larger in size as it requires installation of libraries and functionality that may not be used for our purpose.

To facilitate the communication of the module we have decided to go with running a Python Script over using a webserver framework. Writing our own script to receive requests and send data is very simple and lightweight. It does not include any extra bloat that a webserver framework would come with and makes our code more portable to use on other microcontrollers that could be constrained on memory and storage. A Python script satisfies all the requirements with the added benefit of being fast and



efficient. In the future, we could port our script onto a webserver framework to support for greater expandability and new features if desired.

B.3.2 Computer Vision

For the computer vision application options, we will only be considering mask R-CNN models. This is because the mask R-CNN model is excellent at instance segmentation and bounding box detection while being quick and efficient.

Table B.3.2.1: Computer Vision Application Design Options

Design Options	Specifications
Pretrained Mask R-CNN	In section 6.1 Computer Vision (CV) it describes the usage of a pretrained mask R-CNN model that was trained on the MS COCO database. This pre-trained model is tested and proved to be accurate with many researchers. However, the pretrained model is not specifically trained to detect cars and objects frequent to a parking lot. This could result in some inaccuracies due to weather, time of day and incorrectly classifying object.
Proprietary Trained Model	Training our own model would require a labeled dataset and computational power to train a model. Labeled data can be obtained from opensource forums. Training would need to take place through cloud computing platforms such as, Google Collab, which has limited resources for free users. This would result in a slow turn over in acquiring data, training a model, and proper testing.
Use both pre-trained and proprietary trained model	Use both pre-trained model and proprietary trained model. This is to allow a model for our team to fall back on while we continue to develop, optimize, and train our model.

Among these options, the mixture of a pre-trained model and proprietary trained model will be selected because it meets our requirements and has little entry to barrier. By using two different models, it allows a safety net and model we can use while on our system as we team develop and train our own model. We have also considered to train a specific model for tricky weather conditions. For an example, during the winter and snow fall, the system could switch from the generic pre-trained model to the model specifically for snow covered cars. This is a way to combat the need and reliance on reference photos that requires a full parking lot in each module's FOV. Thus, making our system more resilient and adaptable to different environments which would unlock new markets and have as an added differentiator.



B.3.3 Database

Table B.3.3.1: Design Options for Database

Design Options	Specification
MongoDB	MongoDB provides our team with a cloud database service that provides great flexibility. It makes use of JSON-like documents and is free to use which is advantageous [2]. As our group does not need to store relational data for our system, MongoDB is quite a suitable option. Since a member of our group has had previous experience using MongoDB there would not be a need to spend as much time learning about the service.
SQL database hosted on an AWS server	A database hosted on an AWS instance would provide our team a broad selection of choices. This option would be fully managed, secure, and highly available. Further, AWS has the capability to provide several database types including document, key-value, and relational if needed [26]. However, another consideration would be that AWS databases are not free to use and have an associated cost to them. None of our group members have any previous experience using a SQL database hosted on an AWS server as well.

Following some internal discussions, our team decided that using MongoDB for our database would be the best approach. It is free to use, cloud hosted, and a few of our members have worked with it in the past which adds an aspect of familiarity. Another aspect we considered was that MongoDB uses JSON-like documents and since our group does not require storage of relational data it provides an appropriate fit. Further, using documents are easier to use and understand which provides an advantage as we store information relating to parking spaces.

B.3.4 Server Hosting

Table B.3.4.1: Design Options for Server Hosting

Design Options	Specification
AWS (Amazon Web Services)	AWS is an industry leader in providing a large variety of services on a cloud platform. Since AWS is highly recognized globally and provides our group with the most flexibility in terms of providing services in areas such as compute, storage, and application integration it is a strong option to consider [27]. In addition, AWS provides free credits to users and members of our group have expressed a desire to work with it in the past.
Google Cloud	Google Cloud is another cloud computing option which allows users to access resources such as virtual machines with computers and hard disk drives distributed around the world [28]. Like AWS, Google Cloud provides users with several free credits and is simple to interact with. Another aspect to consider is the fact that several our members have had previous experience using Google Cloud.
Microsoft Azure	Microsoft Azure is another cloud platform that provides us with the capability to have access to a variety of services including virtual machines and databases. It also provides our team with a hybrid cloud environment and is known for having multilayered security [29].



After some careful consideration, our team collectively decided that the best approach would use AWS to host our server. AWS provides our team with the widest variety of services and our members expressed the most interest in choosing this option. Since several members have previous experience working with AWS, our team will not have as steep of a learning curve which will ultimately lead to more productivity in implementing our architecture.

B.3.5 Client and User Interface

Table B.3.5.1: Design Options for Client & User Interface

Design Options	Specification
iOS App / Android Application	When developing a mobile application, we would need to choose between iOS or Android. Also, creating an application for both platforms would cost the group a lot of available time, since no members of OpenSpot have proper experience with app development. However, mobile applications would allow for easy notifications and could be compatible with various car systems such as Apple Car Play or Android Auto. Allowing for in car display.
Website	Website development is mostly done in HTML, CSSS, JavaScript and some sort of Web Framework for easier development process. The website can be designed to be compatible with all internet-capable devices that allow the functionality of a modern web browser (Safari, Google Chrome, Firefox). Majority of our group has some sort of industry or classroom experience with web development. Costing the group much less time.

After consultation between the software developers in our team, a website was chosen to be our main source of Client & User interface. We believe that a website meets all our software and business requirements, while not costing us an extensive amount of time for ramping and learning. With a website we can provide a pleasant GUI to the user and make it friendly so that all range of technically capable users can function use it. In addition, with the feature of text notification, we will be able to provide the user with real-time information about a parking-lot they subscribed to. Built in car systems can read out the text messages, covering the real benefit that an app would provide over a website.



B.3.6 Web Server Framework

Table B.3.6.1: Design Options for Web Server Framework

Design Options	Specification
Django	Django is a python web framework that the software lead at OpenSpot has experience working with before. It allows us to host our computer vision application, allowing for an easy integration into our website. In addition, it can easily work with MongoDB as it has a dedicated database section. Majority of us at OpenSpot are familiar with Python allowing for easy integration if chosen.
Express	Express is increasing in popularity since it is a Node.js web app framework. This would allow the team to easily host our computer vision application and display it appropriately on our front-end. Only a single member of our team has experience working with JavaScript and Node.js in specific.
Ruby on Rails	Ruby on Rails or known as Rails is another server-side web application framework. You can develop your website with HTML, CSSS, and JavaScript. No team member has any experience with this framework and the time required to learn is unnecessary as all other frameworks mentioned above provide similar functionality

Django is chosen as our web framework, as it is written in Python. Majority of our members at OpenSpot have experience working with Python, allowing the integration of this framework to be easy. In addition, majority of the tutorials and guides that we have researched for hosting our computer vision algorithm use Django as it is easy to integrate applications. Aside that, the front-end can be done in HTML, CSSS, and JavaScript, which most people who have taken a web development course have experience with. With experience, comes time saved. Hence, OpenSpot has decided to go with Django as our web framework.



Appendix C: User Interface and Appearance-Prototype Design

C.1 Introduction and Background

In the design of our user interface, we have collectively strived to create a mobile-friendly website that offers users and clients an elegant design that is easy to use. We have designed our website with the aspect of interactivity in mind as this is crucial in users accessing information on the availability of parking stalls within a particular lot before their arrival.

C.1.1 Purpose

The purpose of this appendix will be to give an overview of the design process behind our user interface. It will outline details behind our decisions including justifications while considering the guidelines behind an effective user interface and design.

C.1.2 Scope

There will be five topics which will be discussed in our user interface and appearance-prototype design:

1. **User Analysis:** This section will outline required user knowledge and restrictions with respect to prior experience while using OpenSpot.
2. **Technical Analysis:** Describes how the design of OpenSpot considers the “Seven Elements of UI Interaction” (discoverability, feedback, conceptual models, affordances, signifiers, mappings, constraints).
3. **Engineering Standards:** Outlines the engineering standards which are relevant to the design of our user interface.
4. **Analytical Usability Testing:** This section details the analytical usability testing which will be conducted by our team.
5. **Empirical Usability Testing:** Describes empirical usability testing completed with users while considering safe and reliable use of our system to minimize any potential error along with effective recovery.

C.2 User Analysis

The OpenSpot solution is designed to be a simple and accessible solution that anyone can use. Change is something that can take people some time to get used to. That is why we believe in making the transition from their normal parking experience into a smart parking experience as effortless and as convenient as possible. We have dedicated our time and effort into simplifying each component of the OpenSpot solution as much as possible.

For the direct solution, drivers would only have to know what the different colors of the LED indicator represent. By potentially adding signage that directs drivers to our website (potentially with a QR code), we can easily inform them that a green light represents more than 50% of available spots in the area are open, a yellow light represents less than 50% of spots in the area are open, and that a red light means less than 5% of the spots in the area are available. Even if drivers did not know what the lights meant, by using common color patterns, drivers will be able to infer what the different color lights mean. A green light usually is generally associated with availability while red is associated with unavailability.

For the indirect solution, making the website UI clean and simple is essential in providing a welcoming experience online. Websites are something almost all users will have experience navigating. If we can



implement a traditional user-friendly UI, users should be able to navigate through the website with ease.

Looking at user's prior experience with similar systems, we will study the smart parking solution, Smart Parking Ltd, a UK based company. The client testimonials are all positive where they have an overall rating of 4.7/5. Most testimonials have the same testament that the solution provides a useful aid in parking that not only helps congestion and thus pollution, but also that the information gained from the system is overall beneficial for transport needs. A testimonial that sticks out mentions how "mobile devices and new sources of data are making it ever-easier to help people avoid delay and manage their journeys." [30] This is a key design feature that we thought would also help the travelling experience and we now have confirmation in the benefit it can potentially provide. The user experience on the other hand is a different story. On Trustpilot, Smart Parking has a 1.1/5 rating [31]. Most of the issues stem from car parks where a charge is applicable.

C.3 Technical Analysis

The main criteria outlined in the 405W lectures and Don Norman's text, 'The Design of Everyday Things', are discoverability, feedback, conceptual models, affordances, signifiers, mappings, and constraints.

C.3.1 Discoverability

To make sure we oblige by this our website will have well labeled tabs for easy navigation leaving causing no confusion for the user.

C.3.2 Feedback

We will be using a LED beacon mounted on our system to indicate the parking density in different lots. Red will indicate that less than 5% of spots are available, yellow will indicate less than 50%, and green will indicate over 50%. This way users are aware of the parking availability. Users can also opt into text message notifications which will keep them updated on parking available for their specified lots. Along with the LED's our website will also show a heat map which will show high traffic areas.

C.3.3 Conceptual Model

By showing a map, a compass has been added to indicate the direction. Also, surrounding land marks and parking lot lines have been added to guide the user in the orientation

C.3.4 Affordance

Our users will use our system via our website. Each option will have a low level of physical interactions but have well labeled navigation buttons which users can tap to explore the system.

C.3.5 Signifiers

Signifiers are like feedback. The LED mounted to our system would be the signifier in our case since it displays a system change/update. The color of the LED would be the feedback, the actual information relayed to the user.

C.3.6 Mappings

Our system does not use physical mapping as user do not have to physically interact with our module. However, our digital mappings on the website will have large buttons that follow general conventions and will be well labeled.



C.3.7 Constraints

To access our website user's will need a mobile device and an internet connection. The map will be constrained to only show areas of the parking lot and nothing more. The user shall not be able to view outside a certain radius of the specified parking lot.

C.4 Engineering Standards and Laws

Following user interface engineering standards is imperative to our system. The main interfaces of our system are the indicator lights and the website. The following standards OpenSpot will follow are published by ISO. We plan to adhere to these well-defined industry standards to maintain consistent usability and follow general conventions.

C.4.1 UI Design Standards

ISO 9241-161:2016	Ergonomics of human-system interaction — Part 161: Guidance on visual user-interface elements [32]
ISO 9241-220:2019	Ergonomics of human-system interaction — Part 220: Processes for enabling, executing, and assessing human-centered design within organizations [33]
IEEE/ISO/IEC 23026-2015	ISO/IEC/IEEE International Standard - Systems and software engineering - Engineering and management of websites for systems, software, and services information. Goal is to improve the usability of informational websites and ease of maintenance of managed web operations. [34]

C.4.2 Driving Safety Laws and Regulations

The following safety laws and regulations are stated here to acknowledge the possible misuse of our system that could contribute to distracted driving. A confirmation will be added upon accessing our website to discourage unlawful behaviour. The following are in reference to Canadian and Provincial rules, guidelines, and laws.

Distracted Driving Laws	Distracted Driving Laws & Penalties Per Province [35] [36]
MOTOR VEHICLE ACT [RSBC 1996] CHAPTER 318 Part 3.1	Use of electronic devices while driving for British Columbia [37]

C.5 Analytical Usability Testing

In this section we aim to outline the analytical testing procedures that members of OpenSpot Team will be performing. We aim to cover problems that are overlooked in our interface design. The two main UI-components will be our website and the LED lighting at the parking lot. The test guide is written below.

C.5.1 Website

1. Website has an intuitive look to it and is easy to navigate to the respective parking lot.
2. Visual image of the parking lot is easy to follow and can easily find which parking spots are taken or not.
3. Website has an easy text input for signing up for text notifications.



- a. Process to unsubscribe is easy to perform as well.
 - i. Respond with 'STOP'.
4. Navigating into the login page for clients is easy to follow and sign up.
5. Statistical view of the parking lot for clients is pleasant to look at and provides information in an informative manner. Not overly complicated.
6. Website displays appropriately on various devices and sizing does not get ruined based on laptop versus mobile device.
7. Notifications sent to clients when a car alarm is ringing is intuitive and tells in which parking lot it is happening.

C.5.2 Module

1. Viewing free spots at parking lot.
 - a. The LED lights at each module should be clearly visible once within its vicinity.
 - b. The LED lights colour should be easily distinguishable at night or early morning.

C.6 Empirical Usability Testing

In this section we will be detailing the empirical testing that will be conducted by end users. In this case, the users will have no prior knowledge of OpenSpot and will provide their feedback appropriately. We will examine their thoughts and comments regarding our design which will be used to determine the usability of OpenSpot. The testing will provide the inclusion of both clients and users in this case since both interact with the user interface through the website to view information regarding parking lots. Since end users will be interacting with the system through the website, the section below will focus on their interaction while using the interface.

C.6.1 End User Activity 1: Navigating to the Website and Signing up for Text Notifications

Feedback Questions:

1. Were you able to successfully access and the website and navigate to the text notifications sign up page?
2. Can you enter your phone number and submit it on this page?
3. Did you receive a text notification alerting you that a parking lot was full?

C.6.2 End User Activity 2: Viewing Parking Lots on the Website

Feedback Questions

1. Were you able to navigate to the website and find the parking lot you were looking for?
2. Once selected, were you able to observe the state of each parking stall? (i.e., green for unoccupied, red for occupied)
3. Did this parking lot map provide a helpful view in advance of your arrival?

C.6.3 End User Activity 3: Client notification of an active car alarm within a lot

Feedback Questions:



1. Were you able to successfully receive a notification that an active car alarm was occurring within a parking lot via the website?
2. Was this helpful in efficiently determining which car had a car alarm which was set off?
3. Were you able to see the LED light flashing (signifying a car alarm) upon arrival at the affected parking lot?
4. Did this save you a significant amount of time in determining the presence of a potential crime occurring?

C.6.4 End User Activity 4 Clients Able to Register and Login to the Website

Feedback Questions:

1. Were you able to navigate to the website and able to register for an account?
2. Once registered, did you successfully login to the website and were able to view the different pages?
3. Did you encounter any difficulties during the signup and/or login process?
4. Were you able to recover your account if you forgot your password/username?

C.6.5 End User Activity 5: Obtaining help

Feedback Questions:

1. Were you able to find help in navigating the site?
2. Were you able to find the answer to any question you may have?
3. Were you able to find the contact information for customer service?

C.6.6 End User Activity 6: Account Management

Feedback Questions:

1. Were you able to navigate through the account management menu?
2. Are you able to find and configure your privacy settings?
3. Are you able to find the option to delete your account?

C.7 Graphical Presentation

The interface that user and clients will interact with is the home (main) page of the website. Shown in Figure C.7.1 and Figure C.7.2, the user is greeted by a map of the destination's parking lot. Green and red rectangles fill spots to respectively indicate vacant and occupied. There are minimal and recognizable monuments of the parking lot, as well as a compass to help the user identify the section of the parking lot they are viewing. The users can also zoom in or out to view more of spots. Other parking lots at the destination can be search, selected, and viewed through the side bar. This provides the user with the most critical information (the availability of parking stalls) centered directly on the screen.

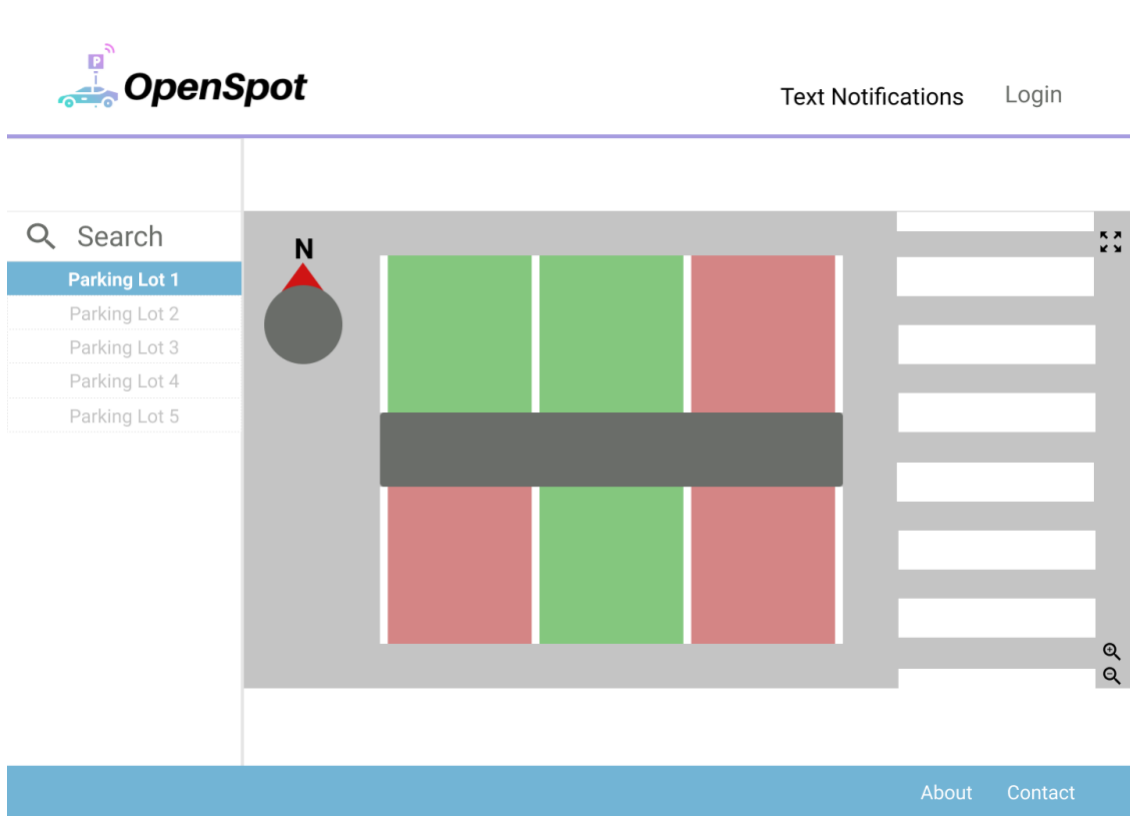


Figure C.7.1: Landing (main) Page of the Website – Desktop View

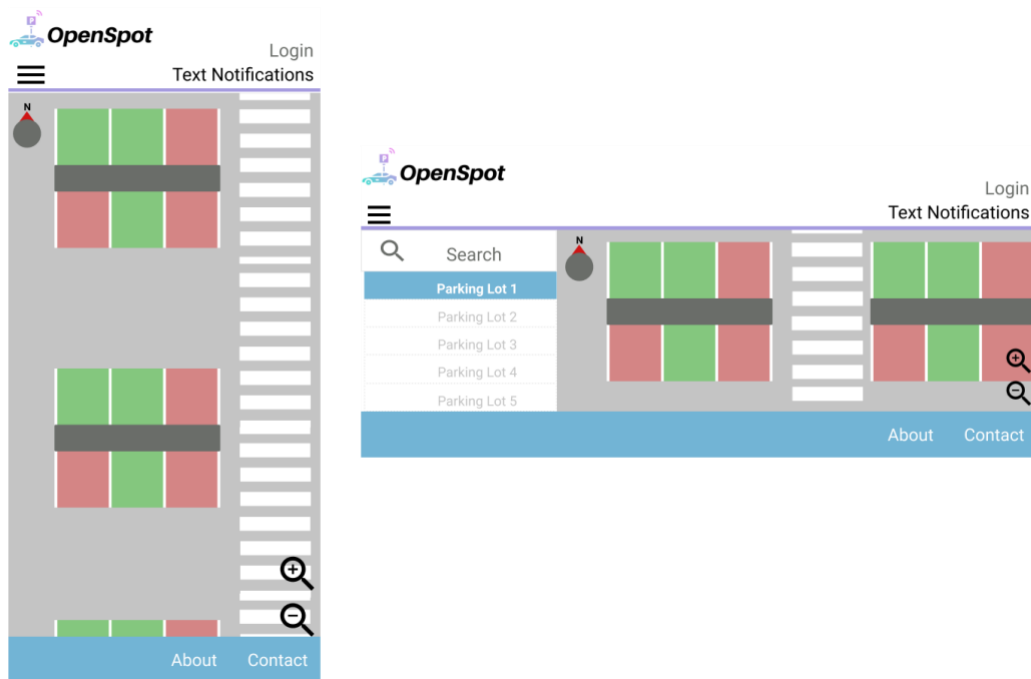


Figure C.7.2: Mobile Landing Page of Website - Portrait (Left) and Landscape (Right)



C.8 Conclusion

The user interface design requirements are essential to making sure our product is accepted by the users. The way the users interact with our product is through the website and indicator lights. We have designed it in accordance with the guidelines from Don Norman's book to provide an interface that is simple and easy to interact with. Through the use of analytical and empirical usability testing our team will be able to gain valuable insight into determining the effectiveness of our design.

Our team has done the appropriate research in determining the design and overall architecture of our website. A low fidelity prototype of our website has been designed in Figma, as seen in Figure C.7.1 and Figure C.7.2. Appropriate engineering standards have been identified and we will be following them closely as we implement the website in the beta phase. The colours of the indicator lights have been selected appropriately to be ubiquitous with their generally associated meanings (e.g., red meaning unavailable, green meaning vacant).