Parking Lot Availability Tracking

Final Report

Software Engineering

CSCI 5530 A

Advisor/ Client: Dr. Andrew Allen

Advisor/ Client: Dr. Pradipta De

Team Members:

Taivon Watkins (Team Lead)

William Collins

Jonathan Roney

Katie Lum

Jared Dean

Marvin Jenkins

Table of Contents

[**Project Scope and Team Roles**](#_szabmq5fzhui) **4**

[Project Scope](#_1m01g7nw22kr) 4

[Development Teams](#_nj98ppiw25vh) 4

[**Feasibility Report**](#_wd2xdb76ydy7) **5**

[**Software Requirements Specification**](#_wy2c0xv0n0ba) **11**

[1 - Introduction](#_oeiy6d96vru9) 14

[1.1 Purpose](#_tb0jhxct104b) 14

[1.2 Project Scope](#_s5pb66b0b8ls) 14

[1.3 Defined Terms](#_dmznlgi6oif0) 14

[2 - Description](#_ojs8hcykfecb) 15

[2.1 Project Perspective](#_c9qwnl5ig6op) 15

[2.2 Constraints](#_1jlsrxevjsa1) 15

[3 - Requirements](#_5xf63puckyxq) 16

[3.1 Product Requirements](#_gnkafmjyv0i4) 16

[3.2 Functional Requirements](#_mglzkt56qc0d) 16

[3.3 Product Document Requirements](#_ra5a8doejfgy) 17

[3.4 External Interface Requirements](#_vhycs7ecscvy) 18

[4 - Core Features](#_k18r0c4o19hp) 25

[4.1 Use Case Diagram](#_8ud0svtjkr0o) 25

[**Software Design Specification**](#_navx68bp3ptr) **27**

[Abstract](#_vt0om21osfsw) 28

[1 - Introduction](#_pf4ylcssxywp) 31

[1.1 Purpose of System](#_hs93g1yss9ho) 31

[1.2 Design Methodology](#_hu6vb8z1ib2g) 31

[1.3 Definitions, Acronyms, and Abbreviations](#_uxdketso3gly) 32

[1.4 Overview](#_j28c6c6zwoji) 32

[2 - Proposed Software Architecture](#_be19wtpg6yfx) 33

[2.1 Overview](#_z3gom7lhr4lm) 33

[2.2 Subsystem Decomposition](#_1bs5wj7f9pex) 34

[2.3 Persistent Data Management](#_4ks4zoqsohbk) 36

[3 - Object Design](#_2kc8n8x4bgat) 38

[3.1 Overview](#_6a9a1rg3n0qq) 38

[3.2 Object Interaction](#_anhlkskhordz) 38

[3.3 Detailed Class Design](#_banbgfg3xqkz) 39

[4 - Glossary](#_8vexhmsrjrex) 41

[5 - Appendix](#_rmiytsv0at1n) 42

[Appendix A: Use Cases](#_xx8sdgsc158b) 42

[Appendix B: Diary of Meetings](#_txyt45tvonek) 44

[**Software Testing Document**](#_hadcb961rmpy) **55**

[System Level Requirements Validation Checklist](#_cofwb5ct0rua) 56

[Software Requirements Specification Checklist](#_s526lmb66xd6) 57

[Design Model Checklist](#_9zj49wob1n2u) 58

[Data Design](#_4qjesxa94aeg) 59

[Architectural Design](#_5puf7ntbwn46) 59

[User Interface Design](#_u0sn0seslcgq) 60

[Component-Level Design](#_eesmuf4df0lr) 61

[Test Cases](#_i2gaglsqmbgt) 62

[Embedded Systems Testing](#_v250rk345iaw) 62

[Software Tests](#_jjljp9sqcmlx) 62

[Mobile Application Testing](#_o6kasr5o33qc) 63

[Website Testing](#_gitkfjihqyex) 66

[**Hardware Setup**](#_dx0y6axcvg01) **69**

[**Future Changes**](#_xu8p18jukhr0) **71**

[Nodes](#_ugwdav84bw3m) 71

[Gateway](#_b9s2225nkok) 71

[Hardware](#_q2afn4fmbfpe) 71

[Database](#_gqtfinyb6t94) 71

[Website](#_drsudojul7fp) 72

[Mobile Application](#_zi2dy6xfq76o) 72

Section 1

# Project Scope and Team Roles

## Project Scope

The Parking Lot Tracking Group is tasked with utilizing the already built low-cost devices to implement algorithms that will allow the available spots in each given lot to be shown in real time. Not knowing how many spots are available in a lot is currently an issue for the Parking Office because they are currently tracking this data via scout cars. This data is beneficial in that it will allow administrators to decide whether new parking lots are necessary and allow them to view the analytics for each given lot. When viewing the data administrators will be able to determine which time of day and which lots are busiest and will be able to deploy scout cars to the location which will help cut down on costs for the office of transportation. In addition, this system will allow non-admin users to see the trends for a lot and pick the best time to go to their lot which will reduce driver frustration and allow for a much smoother flow of traffic.

To accomplish these goals, two partner sensors will be deployed at the entrance and exit of each lot. This information from the sensor will be sent through the network through a gateway in a one hop manner, this will reduce protocol complexity and save battery life. The gateway will then send the data to a server in the cloud where a driver can use a mobile app which will access the cloud server for updates.

## Development Teams

* Hardware: Team members in this role will focus on designing the sensor waypoint, and ensure its feasibility for scaling the solution campus-wide.
* Infrastructure: Team members in this role are responsible for setting up and maintaining the networking, server, and database infrastructure necessary. They must interface this infrastructure with the Hardware and Software teams.
* Software: Team members in this role will design and implement a mobile application to serve the end users as well as create a management console, providing them with useful information about parking across Georgia Southern.

Section 2

# Feasibility Report

**Advisor / Clients:**

Dr. Andrew Allen [andrewallen@georgiasouthern.edu](mailto:andrewallen@georgiasouthern.edu)

Dr. Pradipta De [pde@georgiasouthern.edu](mailto:pde@georgiasouthern.edu)

**Members:**

Taivon Watkins (Team Lead) tw05404@georgiasouthern.edu

William Collins wc01597@georgiasouthern.edu

Katie Lum kl03256@georgiasouthern.edu

Jared Dean jd06679@georgiasouthern.edu

Marvin Jenkins mj03505@georgiasouthern.edu

Jonathan Roney jr07449@georgiasouthern.edu

**Overview:**

The goal of our project is to create a low-cost parking lot tracking application for Georgia Southern campuses that will provide its user with information on the number of available spots in a given lot. The application will use nodes and a gateway that will update the database whenever a vehicle enters or leaves a designated lot.

**Need for the product:**

Currently the Georgia Southern Parking Office sends out scout cars to physically count the amount of available spots in each parking lot. This approach can be time consuming if there are multiple lots spread across the campus. Utilizing this product will give real-time readings on the amount of available parking spots at each lot instantaneously. The Parking Office will then be able to use this data to determine whether expansion is needed as well as be able to determine the busiest times at each lot. This information will also be beneficial to students, faculty, and guest as it will help them quickly determine which lot is best for them. This will ultimately lead to a decrease in overall congestion as drivers will not waste as much time finding parking spots and it will lower the frustration that comes with the search for parking spots.

**Scope:**

The expectations from this team shall be to design a sensory network that will keep track of the amount of cars entering and exiting the campus parking lots. The provided nodes will serve as a way to connect to a gateway to share data with the server’s database. The server will then scrutinize the shared information to ensure the accuracy of the readings and display it in the end-user application.

**Technical:**

* **Amazon Web Service (AWS) -** We will use AWS for our database and storage due to its low cost, and reliability.
* **Node -** The nodes will be equipped with special sensors that will detect when an object has passed through them. The nodes will then send information to the gateway using LoRaWan.
* **Gateway -** The gateway shall be used to receive and upload information from the nodes to the database.
* **Long Range Wide Area Network (LoRaWAN) -** LoRaWAN will be used to allow the nodes to communicate with the gateway from long distances. LoRaWAN is low cost, reliable, and secure making it the best option for our project.

**Task:**

The task of the project is to continue the work of the previous team with the hopes of delivering a fully functional prototype. The prototype should be free of the errors and concerns that the previous team dealt with. Due to the fact that we will be working off of what the previous team has done we have a list of future changes that they wanted to implement but were unable to. The tasks to be done that were listed in their final report include:

1. Gateway:
   1. Change encryption from ABP to OTAA.
      1. Reason: if an attacker gains access to code for the node or gateway security will be compromised
   2. It appears that the gateway currently can only receive from the node and cannot transmit to the node.
      1. (From the final report) “If changing from single channel to multi-channel gateway using a concentrator, the way packets are captured will need to be changed. Packet processing algorithm however should still function.”
   3. Gateway should be able to run a data analysis query after some condition is met
   4. Gateway currently does not send messages back to the node
      1. (From the final report) “The gateway sending messages back to the node has yet to be implemented/tested. This is a necessary step in order to implement time synchronization between the nodes and the gateway.”
2. Node:
   1. Change encryption method
   2. Packet transmission reliability has not been tested
   3. Clock sync has not been implemented
      1. Without a synchronized clocks any meaningful analysis of parking lot data is impossible.
   4. Node values must be tweaked after testing in an actual test environment.
   5. Separate code for the node into multiple files to make code easier to read and write
   6. Test to ensure node does not run out of memory after long periods of time

**Process to be followed:**

We will be using the Agile Scrum methodology for the project. There will be monthly goals set to ensure there is progress towards completing the project. We will have weekly sprints each month to achieve these goals.

**Deliverables:**

1. A Fully functional system consisting of the nodes, gateway, and database that will be able to effectively give data on the number of available parking spots on campus.
2. An application that the faculty and students can use to identify the number of available spaces.
3. An administrative console to monitor the system.

**Plan of Action:**

Tentative Goals:

1. September

Week 1 - Project introduction, first team meeting

Week 2 - Obtain hardware and code from previous team, analyze and meet with team about code, set up cloud services and database

Week 3 - Get current code and hardware working to previous status

Week 4 - Discuss possible fixes and implementations for the project, prepare for sprints

1. October

Week 1 - Test packet reliability of the node and gateway

Week 2 - Ensure gateway can communicate with the node, adjust queries for determining whether a car has passed or not

Week 3 - Fix control algorithms

Week 4 - Prepare for demo

1. November

Week 1 - Fix encryption issue

Week 2 - Adjust and ensure the GUI is presenting the correct data

Week 3 - Final testing and QA

Week 4 - Prepare final report and demo

**Risk Analysis:**

1. Technological risks:
   1. If selected low-cost technology proves ineffective at detecting traffic events accurately or consistently, then the whole project may become completely ineffective. To combat this, the team will study data carefully and ensure all use cases are handled properly.
   2. Some types of weather can damage equipment and lead to hardware failure. The team has added specialized weatherproof hardware and housings to protect against this.
   3. The node may potentially use more power than originally intended, causing frequent maintenance on the batteries. If this becomes an issue the node may be attached to the power supply on the gate. If the gates are unavailable we will acquire a larger battery or potentially use solar panels.
   4. If the campus network is down or inaccessible, our data won’t be able to be sent to our cloud server from the gateway and our parking counts may be off depending on how long the network is down and how many cars come in and out during that period. A solution would be to use AWS Device Shadow. It allows us to retrieve the last set of data the node collected before going offline. This will let us post an accurate count up until it went offline and we can do manual count and update for the lot if the network is down too long.
2. Requirements risk:
   1. Change of requirements: It is possible that the client will change the requirements after the project begins. Depending on the new requirements the team may have to redesign already completed sections to accommodate this change. To avoid this the team will maintain communication with the client during development so it will be possible to easily adapt.
   2. Miscommunication: It is possible that while working, more than one person will be doing the same thing that another member is already working on. This can result in time being wasted or something else not being completed and could put us behind schedule. To avoid this the team will communicate with each other frequently informing the others of the progress of what they are working on, when they have completed something, any unexpected problems, and what they will be working on next. All code will be shared with other members via some collaborative application such as google drive or GitHub.
3. Resources and time risk:
   1. It is very possible that we will not have enough time to complete the full functionality and implementation during the timeframe of one semester. There is also a similar risk of falling behind schedule if there become any large impediments. The team is all taking different classes and is involved in extracurricular activities that causes our schedules to differ.
   2. The technologies and hardware we have chosen to use could be insufficient to complete the tasks. We have chosen some open-source libraries and software to help save on costs and time. So, these libraries and software could cause impediments within our project.
4. Social and legal risk:
   1. When it comes to legal aspects we will have to worry about the protection of the users’ login information. We will be dealing with LoRaWAN security. In this data transfer system there are 2 different levels of keys. First you have the network session key that is used for determining the validity of the messages between the node and network. Then you have the application session key that deals specifically with the encryption of the data between the node and the handler. These two keys are unique for the specific device and session. Using OTAA the keys will regenerate every time the device is activated. In this system the application key is used to find the 2 session keys on activation.
5. Financial risk:
   1. To the best of our knowledge, we currently have all the materials needed to complete the project.

**Visibility Plan:**

**External:** To ensure the clients are up-to-date with the project’s progress, issues that appear while working on the project, and changes to the requirements the client will be contacted as needed. For smaller updates the client will be updated by email on an as needed basis to be informed of issues that arise or the completion of a requirement. For large updates the team will have a meeting with the client so the client will be able to view overall progress and the project in its current state via deliverables.

**Internal:** The team will meet at least once per week to discuss the work that has been done, problems that have occurred, and what will be worked on next. Meetings shall be documented with a list of those who have attended to keep track of participation and so we will be able to inform those who missed the meeting on what has happened. Additionally, team members will communicate with each other via email, GroupMe, or some other application to notify others of their progress and source code will also be shared.

Section 3

# Software Requirements Specification

Parking Lot Availability Tracking

Software Engineering

CSCI 5530 A

Advisor/ Client: Dr. Andrew Allen

Advisor/ Client: Dr. Pradipta De

Prepared By:

Taivon Watkins

William Collins

Katie Lum

Jared Dean

Marvin Jenkins

Jonathan Roney

Table of Contents

[**Project Scope and Team Roles**](#_szabmq5fzhui) **4**

[Project Scope](#_1m01g7nw22kr) 4

[Development Teams](#_nj98ppiw25vh) 4

[**Feasibility Report**](#_wd2xdb76ydy7) **5**

[**Software Requirements Specification**](#_wy2c0xv0n0ba) **11**

[1 - Introduction](#_oeiy6d96vru9) 14

[1.1 Purpose](#_tb0jhxct104b) 14

[1.2 Project Scope](#_s5pb66b0b8ls) 14

[1.3 Defined Terms](#_dmznlgi6oif0) 14

[2 - Description](#_ojs8hcykfecb) 15

[2.1 Project Perspective](#_c9qwnl5ig6op) 15

[2.2 Constraints](#_1jlsrxevjsa1) 15

[3 - Requirements](#_5xf63puckyxq) 16

[3.1 Product Requirements](#_gnkafmjyv0i4) 16

[3.2 Functional Requirements](#_mglzkt56qc0d) 16

[3.3 Product Document Requirements](#_ra5a8doejfgy) 17

[3.4 External Interface Requirements](#_vhycs7ecscvy) 18

[4 - Core Features](#_k18r0c4o19hp) 25

[4.1 Use Case Diagram](#_8ud0svtjkr0o) 25

[**Software Design Specification**](#_navx68bp3ptr) **27**

[Abstract](#_vt0om21osfsw) 28

[1 - Introduction](#_pf4ylcssxywp) 31

[1.1 Purpose of System](#_hs93g1yss9ho) 31

[1.2 Design Methodology](#_hu6vb8z1ib2g) 31

[1.3 Definitions, Acronyms, and Abbreviations](#_uxdketso3gly) 32

[1.4 Overview](#_j28c6c6zwoji) 32

[2 - Proposed Software Architecture](#_be19wtpg6yfx) 33

[2.1 Overview](#_z3gom7lhr4lm) 33

[2.2 Subsystem Decomposition](#_1bs5wj7f9pex) 34

[2.3 Persistent Data Management](#_4ks4zoqsohbk) 36

[3 - Object Design](#_2kc8n8x4bgat) 38

[3.1 Overview](#_6a9a1rg3n0qq) 38

[3.2 Object Interaction](#_anhlkskhordz) 38

[3.3 Detailed Class Design](#_banbgfg3xqkz) 39

[4 - Glossary](#_8vexhmsrjrex) 41

[5 - Appendix](#_rmiytsv0at1n) 42

[Appendix A: Use Cases](#_xx8sdgsc158b) 42

[Appendix B: Diary of Meetings](#_txyt45tvonek) 44

[**Software Testing Document**](#_hadcb961rmpy) **55**

[System Level Requirements Validation Checklist](#_cofwb5ct0rua) 56

[Software Requirements Specification Checklist](#_s526lmb66xd6) 57

[Design Model Checklist](#_9zj49wob1n2u) 58

[Data Design](#_4qjesxa94aeg) 59

[Architectural Design](#_5puf7ntbwn46) 59

[User Interface Design](#_u0sn0seslcgq) 60

[Component-Level Design](#_eesmuf4df0lr) 61

[Test Cases](#_i2gaglsqmbgt) 62

[Embedded Systems Testing](#_v250rk345iaw) 62

[Software Tests](#_jjljp9sqcmlx) 62

[Mobile Application Testing](#_o6kasr5o33qc) 63

[Website Testing](#_gitkfjihqyex) 66

[**Hardware Setup**](#_dx0y6axcvg01) **69**

[**Future Changes**](#_xu8p18jukhr0) **71**

[Nodes](#_ugwdav84bw3m) 71

[Gateway](#_b9s2225nkok) 71

[Hardware](#_q2afn4fmbfpe) 71

[Database](#_gqtfinyb6t94) 71

[Website](#_drsudojul7fp) 72

[Mobile Application](#_zi2dy6xfq76o) 72

## 1 - Introduction

### 1.1 Purpose

This document was prepared to give a detailed list of requirements for the Parking Availability Project for the computer science students participating at Georgia Southern University. Constraints, dependencies, physical, and application requirements, as well as customers are all specified within this document.

### 1.2 Project Scope

The expectations from this team shall be to design a sensory network that will keep track of the amount of cars entering and exiting the campus parking lots. The provided nodes will serve as a way to connect to a gateway to share data with the server’s database. The server will then scrutinize the shared information to ensure the accuracy of the readings and display it in the end-user application.

### 1.3 Defined Terms

|  |  |
| --- | --- |
| Term | Definition |
| AWS | Amazon Web Services - A cloud services host which provides a plethora of services such as relational databases and cloud computing. |
| RDS | Relational Database - Generally referring to an instance in AWS |
| EC2 | Elastic Cloud Compute - AWS cloud computing tool |
| S3 | Simple Storage Service - AWS cloud storage tool |
| LoRaWAN | Low-Frequency radio that uses ALOHA to transmit data |
| Raspberry Pi | General purpose tool used for affordable computational power. In this project it will be used as the gateway to the cloud servers |
| Adafruit Feather | An Arduino-based microprocessor. One Adafruit Feather will be used in each node. |
| Module | A component, such as a sensor or a LoRa Radio, that is added to the gateway and node |
| Relational Schema | Simple diagram for how database objects relate to each other |
| ER Diagram | Advanced diagram to show how database objects connect |

## 2 - Description

### 2.1 Project Perspective

This project aims to bring an innovative and cost effective solution to parking capacity and future traffic tracking. While a simple application will be designed for this project, it is easily scalable to suit a wide variety of needs. A relational database, cloud server, and aforementioned application shall be implemented based on the scope of this project.

### 2.2 Constraints

* The ultrasonic sensors used to detect cars shall be a large constraint. There are a wide variety of sensors that are capable of doing similar functions, but all have different embedded systems
* Easily accessible internet connected Wi-Fi is crucial. This project makes use of the internet frequently, it is not feasible to attempt this project without an Internet connection.
* The gateway could build up a large queue to send off to the RDS, slowing down the rate at which our information travels. Furthermore, the time it takes for the information to travel without a queue is also a constraint. This could require a more robust gateway computer in the future as the project scales up.

## 3 - Requirements

### 3.1 Product Requirements

The final product will consist of the following three layers: physical, application, server.

* Physical
  + The physical layer will consist of paired nodes that shall obtain information from the ultrasonic sensors and transmit the data to a Raspberry Pi gateway via a low frequency radio known as LoRaWAN. The gateway will occasionally sync clocks with the nodes to minimize clock drift and upload the data to the cloud database.
* Application
  + The application layer will consist of a mobile app as well as a website that shall provide users with the ability to view the capacity of each parking lot. The information shall be pulled from the cloud servers and an alternative copy will be given to the administrators at the parking office for updating information if necessary.
* Server
  + The server layer shall be responsible for storing and hosting data. It will consist of a database instance for public data, a web hosting instance for computation and web hosting, and a storage instance for backups and miscellaneous storage.

### 3.2 Functional Requirements

In this section specific requirements are listed individually.

* Ultrasonic Sensor Object Detection
  + When an object passes in front of a sensor, the sensor shall be able to detect that and give the response to the LoRaWAN radio.
* LoRaWAN Radio Communication
  + When information is received, the LoRaWAN radio shall transmit that data to the designated Raspberry Pi device
* Raspberry Pi Gateway Upload
  + The Raspberry Pi shall be responsible for receiving data packets from the LoRaWAN radio. It shall then take this data and upload it to the database hosted by AWS RDS.
* Relational Database
  + This project will utilize a database for data storage and processing. This database shall provide the system with public data upon request.
* Cloud Computation and Web Hosting
  + This layer shall provide the users with a means of accessing the website by using a web hosting instance for web hosting. Furthermore, any computations shall be done on this instance as well.
* Backup Storage
  + To ensure reliable data storage, periodic backups shall be stored in a storage instance.
* Application Website
  + To provide users useful parking information, a website shall have a page for viewing up to date parking availability information
* Mobile Application
  + This project shall include a mobile app for viewing real time readings of each individual lot’s capacity.
* Maintainability
  + To Ensure maintainability, all written code will be well commented and all documentation well organized, to ensure that future teams will be able to easily understand and extend the code, as well as fix any bugs that come up. For more information, refer to Product Documentation Requirements below.

### 3.3 Product Document Requirements

To ensure scalability, documentation for the usage/steps and relevant information shall be documented for future reference.

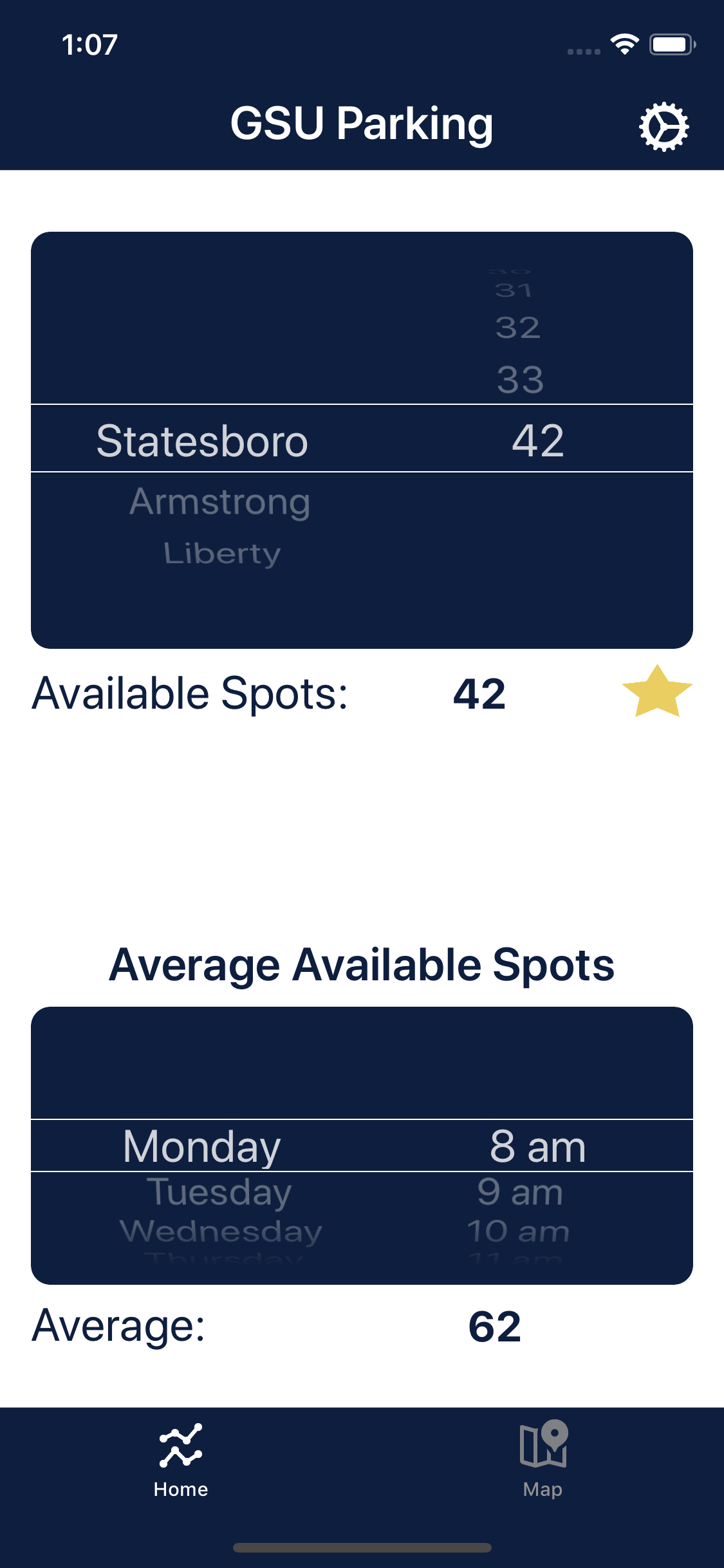
* Database
  + Relational Schema
  + ER Diagram
* AWS
  + Guides to instance creation and connection for all AWS instances
  + Guides on installing and running computational software and files to EC2
* Coding
  + Detailed descriptions in the form of comments shall be documented across all code written
* Physical
  + Hardware Information, such as board type, manufacturer, and specifications shall be documented
  + Schematics for node and gateway shall be documented
  + Wire diagrams shall be documented

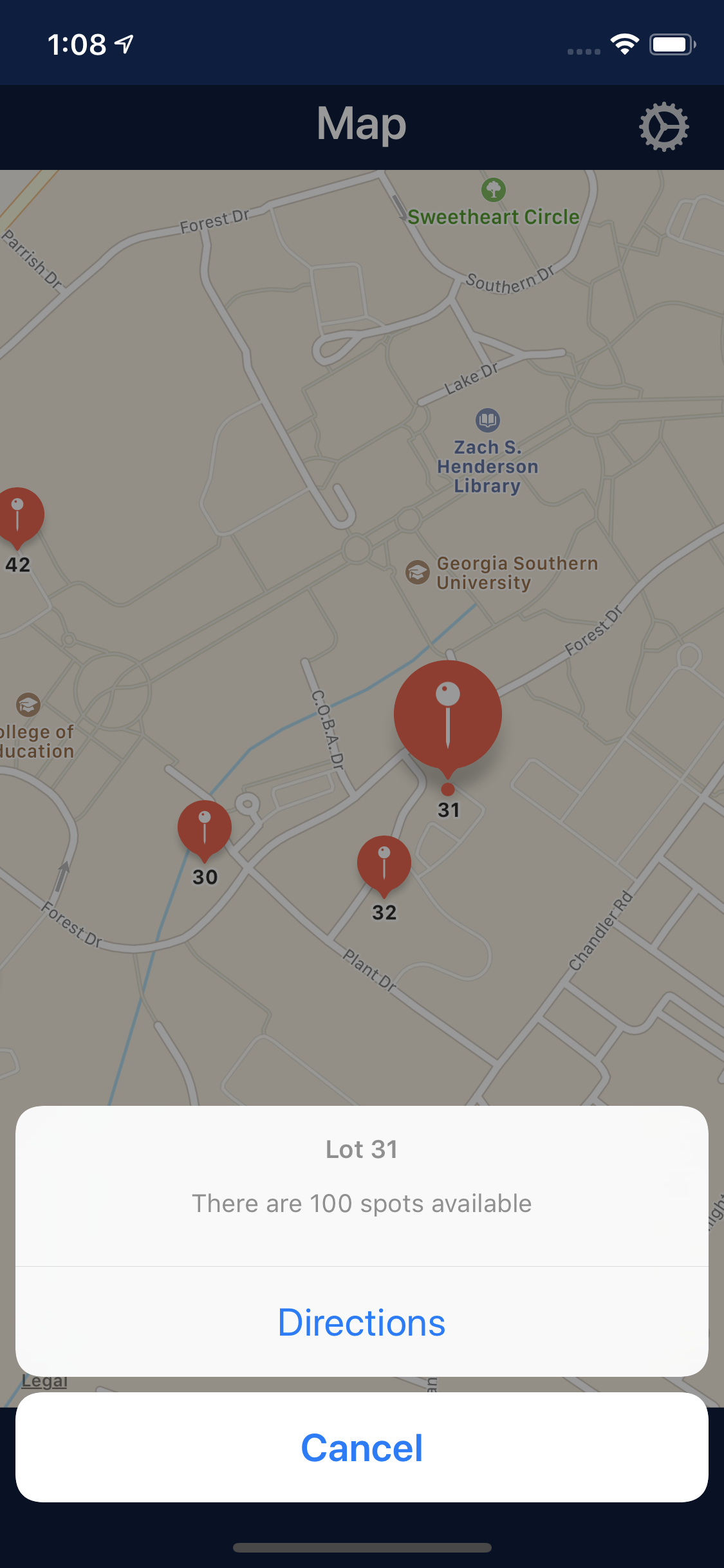
### 3.4 External Interface Requirements

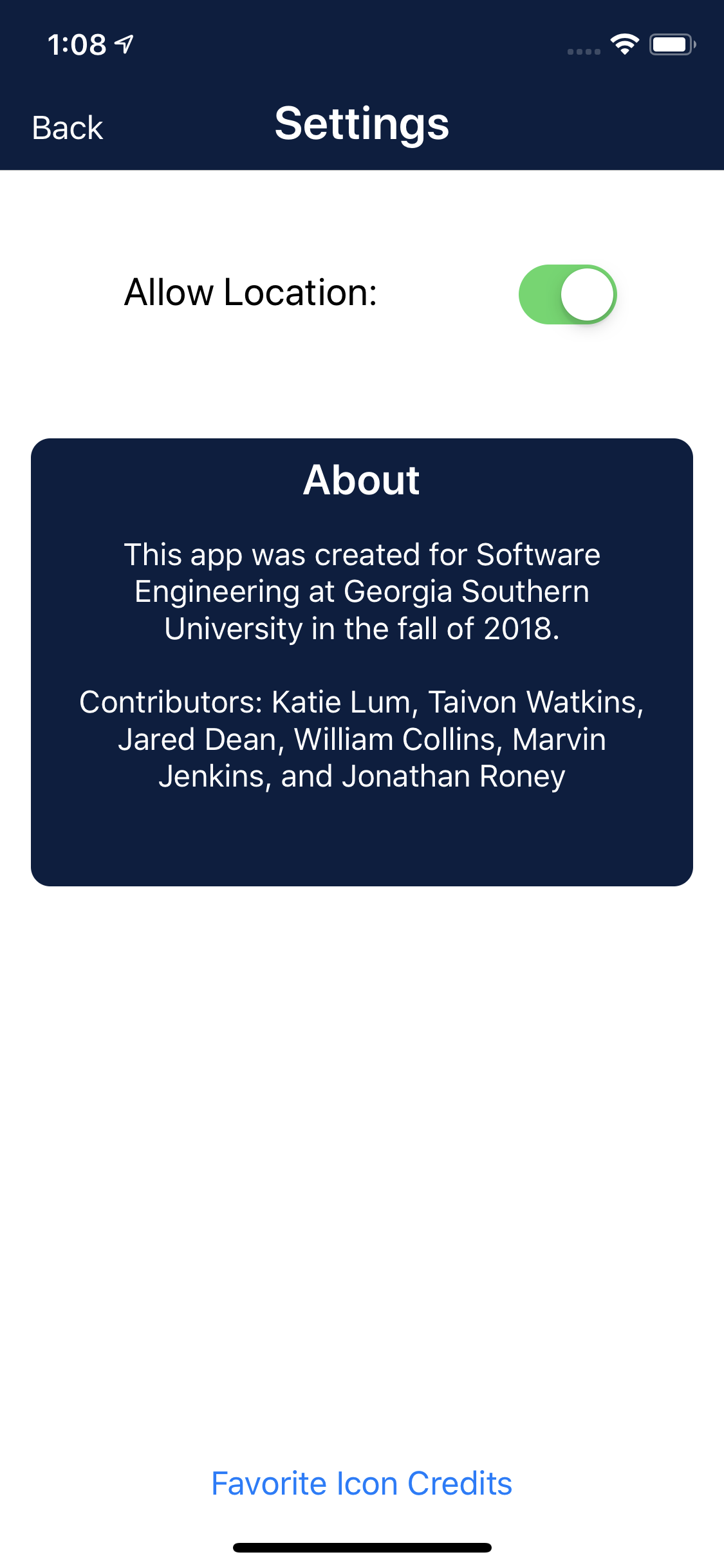
**3.4.1 User Interface**

General Application Layout

* Every user will have one screen that will allow them to select the lot they are trying to check and it will display the lot number and the available spots in that corresponding lot. This tab will also show the trends for that lot. There will be another screen where the application shall have a map and show nearby lots.

****





* Users will be able to access the site by going to the official Georgia Southern App and selecting our icon.
* The application will also include a geolocation feature that will allow the user to use a map to select and view their desired lot.



* Every user will be able to view all available lots with their corresponding capacity based on their role and their campus.
* The website shall be accessible to all users utilizing a web browser.
* The web app will also include a geolocation feature that will allow the user to view a map of the campus lots. The feature will also allow the user to click on each individual lot and display the capacity of that particular lot.

**3.4.2 Hardware Interfaces**

Parking Lot Nodes

* Each node will consist of an ultrasonic sensor and an Arduino-based microcontroller to detect passing cars.
* Once the information is collected, it will send the data to the gateway via LoRaWAN
* The node will be inside of a weather case to protect it against all elements
* Two nodes will be required at each entrance and exit in order to determine if a car has passed or not
* The nodes will periodically synchronize with each other and the gateway to ensure their times are the same

Gateway

* The gateway will consist of a Raspberry Pi microprocessor outfitted with LoRaWAN module
* The gateway shall use this module to receive the raw sensor data from each node and then upload that data to the AWS relational database

**3.4.3 Network Interface**

LoRaWAN

* LoRaWAN is a communication protocol that is designed to allow low-powered devices to communicate across long distances
* It will be used to establish the network needed for the nodes and the gateways to communicate and transmit data between each other
* LoRaWAN comes encrypted end-to-end so data is protected while being sent from the node to the gateway to the server

AWS

* AWS allows for the nodes to be hosted on the server and connect all hardware devices to one network
* AWS shall keep track of all nodes and gateways, when they are on and off the network and deploy updates to the software on the nodes and gateways
* AWS will host the final application and allow for users to access the site to view parking lot availability. It will also be used for any needed computations using the EC2.

**3.4.4 Web Interface**

User View

* Students and faculty will see the user view. This page will only show the geolocation feature, the number of available slots in their selected lot, and a drop down menu to select their desire lot.

Administrator View

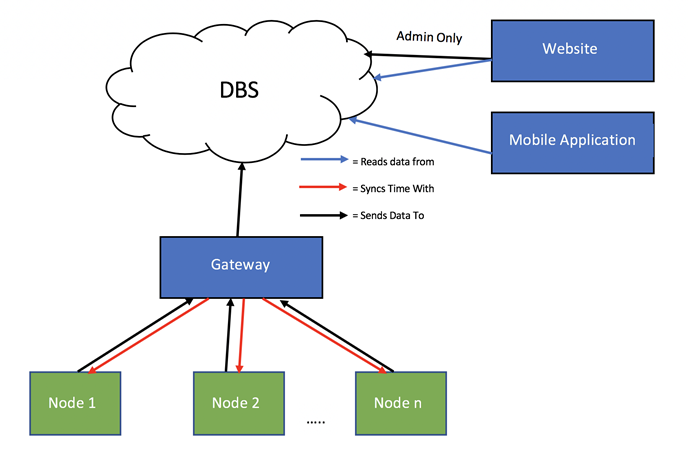
* Administrators will have a separate view that will give them the ability to view the network traffic and also remove/add users.

Geolocation

* The website will include a geolocation feature that will be used to select desired lots from a map. The user will be able to view available lots in their area or scroll to a different area.

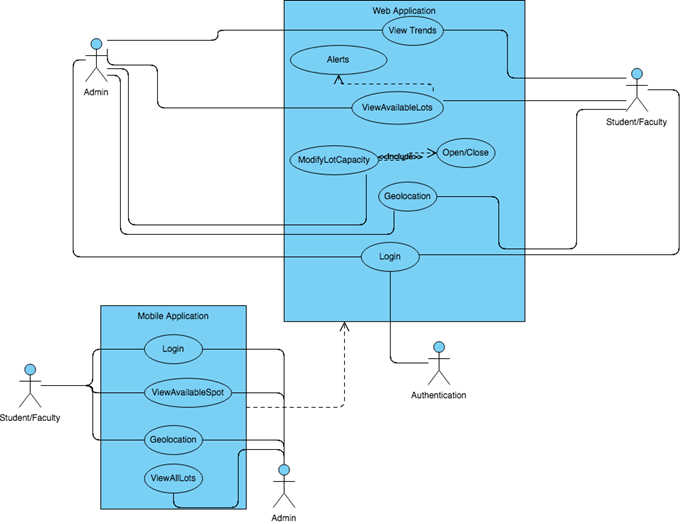
**3.5 Data Flow Diagram**

Below is a figure that shows how the data will flow in the system, from sensor to user



## 4 - Core Features

### 4.1 Use Case Diagram



**4.1.1 Student/Faculty**

Website

* **Login** - The system shall allow the user to login using their MyGeorgiaSouthern credentials.
* **View Available Spots** - The system shall allow the user to view a table of all the lots and spaces available on one page.
* **View Trends** - The system shall allow the user to view parking lot trends by clicking on the parking lot tab and showing the user a data graph
* **Geolocation** - The system shall allow the user to view close by parking lots on a map
* **Favorite Lot** - The system shall allow a user to save a lot under favorites for easy navigation whenever they open the favorites tab their favorite parking lot on campus

App

* **Login** - The system shall allow the user to log in using their MyGeorgiaSouthern credentials.
* **View Available Spots** - The system shall allow the user to view available spots by picking the campus and lot
* **View Trends** - The system shall allow the user to view parking lot trends by clicking on the parking lot tab and showing the user a data graph
* **Geolocation** - The system shall allow the user to view close by parking lots on a map

**4.1.2 Admin**

Website

* **Login** - The system shall allow the user to login using their credentials, in which the credentials will immediately be authenticated.
* **Geolocation** - The system shall provide the nearest lots and their capacities based on the user’s location.
* **Modify Lot Capacity** - The system shall give the user modification rights and update its current data based on changes made.

The system shall give the user the right to open and close lots as necessary.

* **View Available Lots** - The system shall give readings on the status of the capacity in each lot.

Based on the capacity in each individual lot, the system shall periodically send alerts to notify the user when a lot has reached its capacity limit.

* **View Trends** - The system shall provide a graph containing the status of each lots capacity throughout the day.

App

* **View All Lots** - The system shall allow the user to have the same rights as the student and faculty members but they’re combined. The system shall allow the admin to view all lots.

Section 4

# Software Design Specification

Parking Lot Availability Tracking

Software Engineering

CSCI 5530 A

Advisor/ Client: Dr. Andrew Allen

Advisor/ Client: Dr. Pradipta De

Prepared By:

Taivon Watkins

William Collins

Katie Lum

Jared Dean

Marvin Jenkins

Jonathan Roney

## Abstract

The Georgia Southern Parking Office is currently looking for an efficient low-cost way to track each individual lots parking activity throughout the day. Doing so will ensure that each lot around campus is being utilized to its fullest capability. One way to provide this service is to maintain an accurate and easily accessible database that will contain information on each individual lot across campus. This will help determine plans of action for the future such as more parking passes being issued or possible lot expansions. A system like this will not only benefit the Parking Office, but it will benefit students/faculty and staff as well as it will open up their parking options and reduce parking congestion. This system is now in progress and is being called the GSU Parking Availability Tracker. The project will provide the Parking Office with a website as well as a mobile application connected to a database which would receive its information from a single lot. Based on the information provided in this document, the tools provided can be utilized throughout the entire campus.

Table of Contents

[**Project Scope and Team Roles**](#_szabmq5fzhui) **4**

[Project Scope](#_1m01g7nw22kr) 4

[Development Teams](#_nj98ppiw25vh) 4

[**Feasibility Report**](#_wd2xdb76ydy7) **5**

[**Software Requirements Specification**](#_wy2c0xv0n0ba) **11**

[1 - Introduction](#_oeiy6d96vru9) 14

[1.1 Purpose](#_tb0jhxct104b) 14

[1.2 Project Scope](#_s5pb66b0b8ls) 14

[1.3 Defined Terms](#_dmznlgi6oif0) 14

[2 - Description](#_ojs8hcykfecb) 15

[2.1 Project Perspective](#_c9qwnl5ig6op) 15

[2.2 Constraints](#_1jlsrxevjsa1) 15

[3 - Requirements](#_5xf63puckyxq) 16

[3.1 Product Requirements](#_gnkafmjyv0i4) 16

[3.2 Functional Requirements](#_mglzkt56qc0d) 16

[3.3 Product Document Requirements](#_ra5a8doejfgy) 17

[3.4 External Interface Requirements](#_vhycs7ecscvy) 18

[4 - Core Features](#_k18r0c4o19hp) 25

[4.1 Use Case Diagram](#_8ud0svtjkr0o) 25

[**Software Design Specification**](#_navx68bp3ptr) **27**

[Abstract](#_vt0om21osfsw) 28

[1 - Introduction](#_pf4ylcssxywp) 31

[1.1 Purpose of System](#_hs93g1yss9ho) 31

[1.2 Design Methodology](#_hu6vb8z1ib2g) 31

[1.3 Definitions, Acronyms, and Abbreviations](#_uxdketso3gly) 32

[1.4 Overview](#_j28c6c6zwoji) 32

[2 - Proposed Software Architecture](#_be19wtpg6yfx) 33

[2.1 Overview](#_z3gom7lhr4lm) 33

[2.2 Subsystem Decomposition](#_1bs5wj7f9pex) 34

[2.3 Persistent Data Management](#_4ks4zoqsohbk) 36

[3 - Object Design](#_2kc8n8x4bgat) 38

[3.1 Overview](#_6a9a1rg3n0qq) 38

[3.2 Object Interaction](#_anhlkskhordz) 38

[3.3 Detailed Class Design](#_banbgfg3xqkz) 39

[4 - Glossary](#_8vexhmsrjrex) 41

[5 - Appendix](#_rmiytsv0at1n) 42

[Appendix A: Use Cases](#_xx8sdgsc158b) 42

[Appendix B: Diary of Meetings](#_txyt45tvonek) 44

[**Software Testing Document**](#_hadcb961rmpy) **55**

[System Level Requirements Validation Checklist](#_cofwb5ct0rua) 56

[Software Requirements Specification Checklist](#_s526lmb66xd6) 57

[Design Model Checklist](#_9zj49wob1n2u) 58

[Data Design](#_4qjesxa94aeg) 59

[Architectural Design](#_5puf7ntbwn46) 59

[User Interface Design](#_u0sn0seslcgq) 60

[Component-Level Design](#_eesmuf4df0lr) 61

[Test Cases](#_i2gaglsqmbgt) 62

[Embedded Systems Testing](#_v250rk345iaw) 62

[Software Tests](#_jjljp9sqcmlx) 62

[Mobile Application Testing](#_o6kasr5o33qc) 63

[Website Testing](#_gitkfjihqyex) 66

[**Hardware Setup**](#_dx0y6axcvg01) **69**

[**Future Changes**](#_xu8p18jukhr0) **71**

[Nodes](#_ugwdav84bw3m) 71

[Gateway](#_b9s2225nkok) 71

[Hardware](#_q2afn4fmbfpe) 71

[Database](#_gqtfinyb6t94) 71

[Website](#_drsudojul7fp) 72

[Mobile Application](#_zi2dy6xfq76o) 72

## 

## 1 - Introduction

The objective of this project is to create a system that will accurately track the amount of available spots in each individual lot. Although this project ensures an improvement in the overall Parking Office efficiency, in order to do so, the design of the system must be laid out and clearly specified. To express these specifications, this chapter will introduce the Parking Project system, design methods, common and uncommon vocabulary used, and a brief overview of the document itself.

### 1.1 Purpose of System

The reason behind the creation of the Parking Availability Tracker for Georgia Southern’s Parking Office derives from the office wanting a low-cost and reliable solution to its current tracking system. Ultrasonic sensors will be deployed on campus at the entrances and exits of each parking lot to track the number of cars in a lot at any given time. Based on the information from the sensors, analytics will be compiled and used to determine plan of actions for the future. The electronics used in this project shall be hand-assembled, making this a low-cost project.

### 1.2 Design Methodology

* The Parking Availability Tracker development is based on the Agile methodology. Agile was chosen because testing is integrated during the cycle meaning that there are regular checkups to ensure that the system is working and because of this there is a reduced risk of absolute failure in the final deliverance of the project.
* After the completion of the Software Requirements Document, identifying sections of the project that would consume majority of the time became apparent. This document also produced a use case model that would clearly explain the use of the system by a regular user and the Parking Office administration.
* This document will address Data Flow diagrams, UML diagrams, Relational Schemas, EER diagrams, and Use Cases. These diagrams portray how each system/subsystem are incorporated into the final design of the project.

### 1.3 Definitions, Acronyms, and Abbreviations

**Adafruit Feather:** An Arduino-based microprocessor. One Adafruit Feather will be used in each node.

**Adafruit FeatherWing:** An add-on component to the Feather that extends functionality.

**AWS:** Amazon Web Services - A cloud service that provides on-demand cloud computing platforms such as relational databases and cloud computing/web hosting.

**EC2:** Elastic Cloud Compute - AWS cloud computing/web hosting tool.

**ER Diagram:**  Advanced diagram to show how database objects connect.

**I2C:** A two wire communications protocol for integrated circuits.

**Header Pin:** A section of pins either male or female to connect electrical components together.

**Logic-Level Shifter:** Adapts the logical voltage from one level to another so that electrical components that work at different levels can communicate with one another.

**LoRaWAN:** Low-Frequency radio that uses ALOHA to transmit data.

**MySQL Server:** SQL server

**Module:** A component, such as a sensor or a LoRa Radio, that is added to the gateway and node.

**RDS:** Relational Database - Generally referring to an instance in AWS.

**RTC:** Real-Time Clock - Keeps time even when a node or gateway is powered off.

**S3:** Simple Storage Service - AWS cloud storage tool.

**Relational Schema:** Simple diagram for how database objects relate to each other.

**Voltage Step-Up Regulator:** Steps the input voltage up to a higher voltage to power electrical components which require that voltage to operate.

**SPI:** Serial Peripheral Interface - A communications protocol for electronic chips.

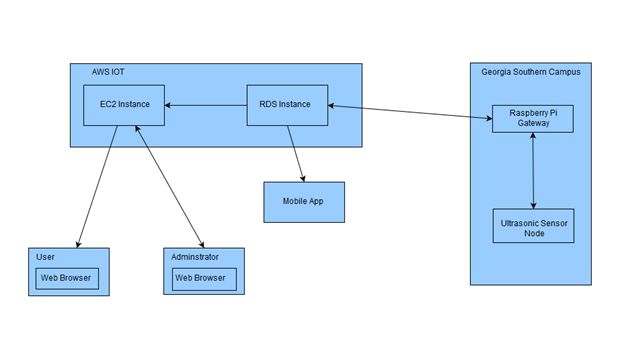
### 1.4 Overview

In subsequent chapters a detailed understanding of the GSU Parking Availability Tracking system will be obtained. In Chapter 2, Proposed Software Architecture, the architecture, relational database schema, and diagram will be discussed and explained. In Chapter 3, Object Design, the class diagrams will be discussed as well as the purpose of each diagram. Chapter 4, Glossary, will define the terms being used throughout this document. This chapter will actually be useful for readers who encounter unfamiliar terms. Chapter 5, Appendix, will discuss the use cases, class diagrams and a diary of meetings. By the end of this document, the reader will have an understanding of the design and its purpose as well as possible implementation of the Parking Lot system.

## 2 - Proposed Software Architecture

This project’s software architecture was carefully selected as it is the most essential design decision that will determine the success, understanding, and repeatability of the project. Due to the importance of the system architecture it is paramount that the design is efficiently done and understood completely so that future engineers can easily reiterate the design and possibly add more analytics to further the system’s efficiency. To achieve the goal of understanding the design completely, this chapter will address the overall software architecture design.

### 2.1 Overview

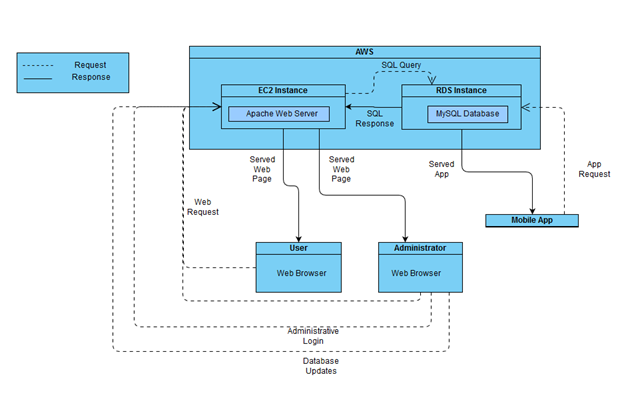


*Figure 2.1.0 - General System with Information flow*

Purpose of above architecture

* The above architecture was designed to be easily understandable so that future engineers would be able to reiterate the design and improve its overall efficiency.
* The system goes through three tiers in which data is being passed from the physical nodes and gateway to the AWS IoT virtually consisting of the relational database and cloud computation to the actual display which is the website and mobile app.
* The system will also contain two-way communication for update purposes.

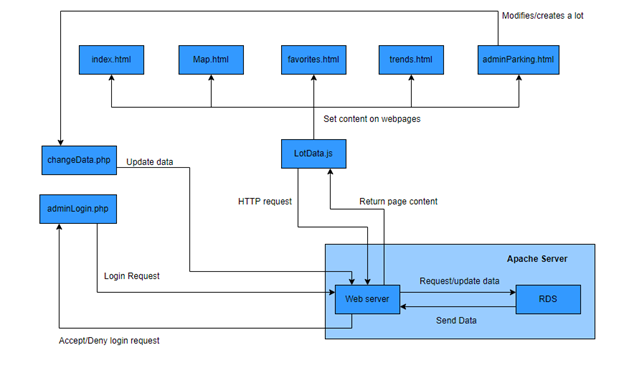
### 2.2 Subsystem Decomposition



*Figure 2.2.0: Diagram showing the interaction between systems*

Components

* **EC2 instance** – Runs apache2 to host a web server, communicates with the database and displays the data pulled from the database to the webpage. (MySQL -> PHP -> JavaScript -> HTML)
* **RDS Instance** – a MySQL database that receives information from the gateway, lot information is updated based on what is returned.
* **User** –The user will be able to submit a request to the database which will return all relevant information. Specifically, the user will be able to view the trends for a lot, view a map of the selected campus where lot information will be displayed, and select a favorite lot where compiled information regarding selected lots will be displayed.
* **Admin** – After logging in administrators will have the same functionality as the user as well as the ability to manipulate lot data and database.
* **Mobile Application** – Has the same functionality as the user webpage. The mobile application uses geolocation to determine the status of lots and can provide users with directions to a specific lot.

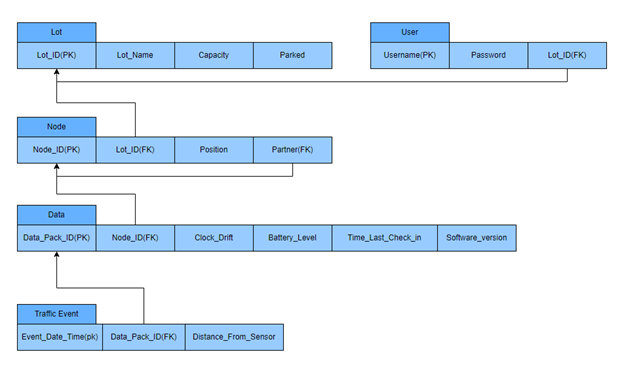


*Figure 2.2.1: diagram showing the interaction between web pages, the server, and other components*

Components

* **Apache2 Server** – A web server that hosts the MySQL database.
* **index.html** – This will be the main page of the website, the user will be able to select their campus and the webpage will make a request to the server for information regarding lots related to that campus and display all results.
* **trends.html** – Will display data trends that have been collected such as peak hours, hours when the lot is least used, etc.
* **favorites.html** – Webpage that will display compiled information regarding a lot the user has designated as a favorite.
* **map.html** – Using google maps the webpage will display a map of the selected campus where users will be able to click on a lot and have it display its related information such as total spaces and spaces available.
* **adminParking.html** – webpage for administrators where they will be able modify lot information and create new lots. From this page administrators
* **lotData.js** – JavaScript file that is responsible for setting the content of a webpage and updating content as it changes.
* **adminLogin.php**– sends a login request to server where the database will then check to see if the username and password are registered. After the user has been verified as an administrator they will be redirected to adminParking.html. If the credentials entered do not exist within the database, the user will be redirected back to the homepage.
* **changeData.php** – Responsible for updating the database after an administrator modifies an existing lot or creates a new lot.

### 2.3 Persistent Data Management



*Figure 2.3.0: Database relational schema*

Table attributes and Definitions

|  |  |
| --- | --- |
| **Lot Table** | |
| Lot\_ID | A unique ID for each lot |
| Lot\_Name | A name given to the lot by the Parking and Transportation office. |
| Capacity | The maximum number of cars that can park in a lot |
| Parked | The amount of cars currently parked in a lot |

|  |  |
| --- | --- |
| **User Table** | |
| Username | Login Username |
| Password | Login Password |
| Lot\_ID | Included in the table so a user can mark specific lots |

|  |  |
| --- | --- |
| **Node Table** | |
| Node\_ID | A unique ID for the node. Will be used to identify which node is being accessed |
| Lot\_ID | Links the deployed node to its lot |
| Position | Used to determine whether a car is entering or exiting a lot |
| Partner | Links two nodes together, each node will have a partner so the pair will be able to determine the direction, size, and speed of an object |

|  |  |
| --- | --- |
| **Data Table** | |
| Data\_Pack\_ID | A unique ID to determine a packet |
| Node\_ID | Used to link a data packet to a particular node |
| Clock\_Drift | Used to account for the natural variations in the clocks |
| Battery\_Level | Determines the battery level of the node |
| Time\_Last\_Check\_In | Specifies the last time a node has communicated with the gateway. Used for analytics |
| Software\_Version | Software information about a node |

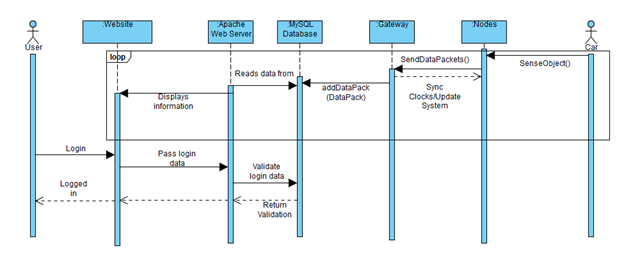
|  |  |
| --- | --- |
| **Traffic Event Table** | |
| Event\_Date\_Time | Date and time the traffic even was detected. Used in analytics. |
| Data\_Pack\_ID | Used to link a traffic event to a data packet. |
| Distance\_From\_Sensor | Distance from the sensor. Used in analytics. |

## 3 - Object Design

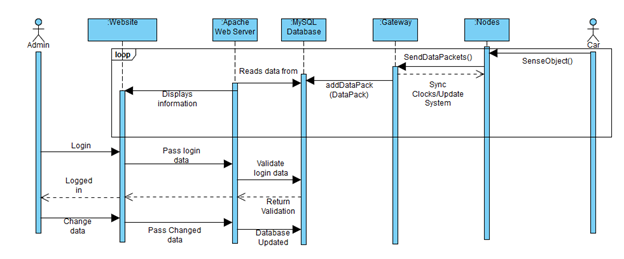
### 3.1 Overview

This section goes into detail about how the nodes and gateway interact with one another at the programming object level. Using diagrams and detail, it will also include how the nodes and gateway communicate with the network infrastructure, website, and mobile app.

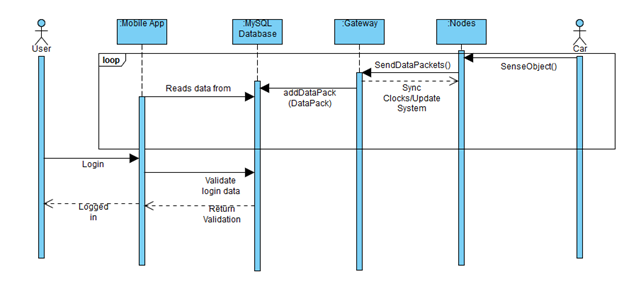
### 3.2 Object Interaction



*Figure 3.2.0 Object Interaction for website (User)*



*Figure 3.2.1 Object Interaction for website (Administrator)*



*Figure 3.2.2 Object Interaction for mobile app (User)*

### 3.3 Detailed Class Design

**3.3.1 Node**

* **Node:** The Node class represents the microcontroller that connects the sensor, LoRa module, and Clock module. The node is responsible for compiling the sensor information into a data packet with other information (nodeID, battery percentage, etc.) and sending the data packet to the gateway using the LoRa module. The node will use the clock module to timestamp the sensor data, and will frequently sync the clock with the gateway. The node should update its software when it receives an update from the gateway.
* **LoRa:** The LoRa class represents the LoRa radio module which is connected to the microcontroller. The LoRa module is responsible for sending data packets to the gateway, receiving clock sync packets from the gateway, and receiving software update packets from the gateway.
* **Sensor:** The Sensor class represents the ultrasonic sensor which is connected to the microcontroller. The Sensor is responsible for sensing when an object passes in front of it and for calculating the distance between the sensor and that object.
* **Clock:** The Clock class represents the clock module which is connected to the microcontroller. The clock is responsible for keeping an accurate time of day, and will be used to timestamp the sensor data. The clock will be periodically synced with the clock on the gateway.

**3.3.2 Gateway**

* **Gateway:** The Gateway class represents the Raspberry Pi which connects to the LoRa module and Clock module, and has connection to the internet. The gateway will receive data packets from the nodes and upload that data via a query to the database. The gateway will also periodically send clock sync packets to the nodes, which will sync the nodes’ clocks with the gateway’s clock. The gateway should be able to send system update packets to the nodes, causing the nodes to update their software, when necessary.
* **LoRa:** The LoRa class represents the LoRa radio module which is connected to the Raspberry Pi. The LoRa module will receive data packets from the nodes and forward them to the gateway. The LoRa module will also send clock sync packets to the nodes when necessary. The LoRa module should be able to send system update packets to the nodes when necessary.
* **Clock**: The Clock class represents the clock module which is connected to the Raspberry Pi. The clock will keep accurate time and serve as a reference to what the nodes’ clocks will sync to.

**3.3.3 Database**

* Database: The Database class represents the relational database which holds the parking sensor data. The database will allow queries to be made in order to modify or access the data inside the database. The database will process the sensor data in order to calculate the amount of available parking spaces at each parking lot. It also stores data for the statistics that are used for analytical purposes.

**3.3.4 Website**

* The Website class represents the website where users can access the parking lot data. The website shall read the amount of parking spaces available in a particular parking lot from the database as well as trend data and then display this information to the user.

**3.3.5 Mobile App**

* The Mobile App class represents the mobile app that users will use to access the parking lot data. The mobile app shall read the parking data in a particular parking lot from the database and display this information to the user.

## 4 - Glossary

**Adafruit Feather:** An Arduino-based microprocessor. One Adafruit Feather will be used in each node.

**Adafruit FeatherWing:** An add-on component to the Feather that extends functionality.

**AWS:** Amazon Web Services - A cloud service that provides on-demand cloud computing platforms such as relational databases and cloud computing/web hosting.

**EC2:** Elastic Cloud Compute - AWS cloud computing/web hosting tool.

**ER Diagram:**  Advanced diagram to show how database objects connect.

**I2C:** A two wire communications protocol for integrated circuits.

**Header Pin:** A section of pins either male or female to connect electrical components together.

**Logic-Level Shifter:** Adapts the logical voltage from one level to another so that electrical components that work at different levels can communicate with one another.

**LoRaWAN:** Low-Frequency radio that uses ALOHA to transmit data.

**MySQL Server:** SQL server

**Module:** A component, such as a sensor or a LoRa Radio, that is added to the gateway and node.

**RDS:** Relational Database - Generally referring to an instance in AWS.

**RTC:** Real-Time Clock - Keeps time even when a node or gateway is powered off.

**S3:** Simple Storage Service - AWS cloud storage tool.

**Relational Schema:** Simple diagram for how database objects relate to each other.

**Voltage Step-Up Regulator:** Steps the input voltage up to a higher voltage to power electrical components which require that voltage to operate.

**SPI:** Serial Peripheral Interface - A communications protocol for electronic chips.

## 5 - Appendix

### Appendix A: Use Cases

|  |  |
| --- | --- |
| **Table A.0 - Use Case - Student (Parking Lot Lookup)** | |
| User | Georgia Southern University Student |
| Case | A student is getting ready to go to class. They use the website or mobile application to check the parking lot of their choice to see if there are any available spots. If the lot is full, then the student can find an alternative way to get to class. |
| Benefit | The student was able to save time by checking whether or not the lot was full. If the lot was full, then it would save the student time to make a different choice. |

|  |  |
| --- | --- |
| **Table A.1 - Use Case - Student (Parking Lot Lookup and Geolocation)** | |
| User | Georgia Southern University Student |
| Case | It is finals week and a student is getting ready to go do some late night studying at the library. The student uses the website or mobile application to see which lots are around the library. The student then selects the lot right behind the library and the application shows that it is full. The student then selects a lot that is a little farther away and sees that there are spots available. The student then uses the mobile application to get driving directions to the chosen lot. |
| Benefit | The student saves time and avoids frustration of after-hours parking. The student also gets directions to the lot of their choice. |

|  |  |
| --- | --- |
| **Table A.2 - Use Case - Faculty Member (Parking Lot Lookup and Favorites)** | |
| User | Georgia Southern University Faculty Member |
| Case | Before arriving at work, the faculty member checks the website or mobile application to see if their favorite lot is full or closed. The application automatically loads the faculty member’s favorite lot so they can quickly check it. |
| Benefit | It saves the faculty member time to have his favorite lot already selected when checking the application. |

|  |  |
| --- | --- |
| **Table A.3 - Use Case - Administrator (Alter parking database)** | |
| User | Georgia Southern University Parking and Transportation Administrator |
| Case | Georgia Southern has acquired a new parking lot. An administrator accesses the administrator dashboard on the website and adds the new lot with the appropriate inputs and specifications. The administrator also changes Lot 42 to be closed on October 5th for the Homecoming Parade. |
| Benefit | The administrator can quickly manipulate the parking database to account for new lots and changes for other lots. |

|  |  |
| --- | --- |
| **Table A.4 - Use Case - Student (Trends)** | |
| User | Georgia Southern University Student |
| Case | A student is trying to find out when is the best time to come to campus. The student uses the website or mobile application to view parking trends for a particular lot. The student now knows when their parking lot is the busiest. |
| Benefit | The student can see when the parking lot is the busiest and the slowest so they can plan their day around when to arrive. This saves them time and eliminates frustrations. |

### Appendix B: Diary of Meetings

Date: 9/06/18

Time: 3:00 - 3:35 pm

Location: IT Building - Room 2206

Attendance: Katherine (Katie) Lum, Taivon Watkins, Jonathan (Adam) Roney, Jared Dean, Marvin Jenkins, William (Chance) Collins

Points Discussed:

* Skills of team members
* General project discussions after reading previous report.
* High level discussions of tools and services that will be used.
* Discussion of obtaining current software and hardware.

Date: 9/09/18

Time: 6:00 pm - 7:00 pm

Location: IT Building - Room 2206

Attendance: Taivon Watkins, Jonathan (Adam) Roney, Jared Dean, Marvin Jenkins, William (Chance) Collins

Absence: Katherine (Katie) Lum

Points Discussed:

* Obtaining code from previous team.
* Structuring of the Feasibility Report
* Structuring of weekly report
* Continued discussion of team roles

Date: 9/13/18

Time: 3:30pm - 5:15pm

Location: IT Building - Room 2206

Attendance: Katherine (Katie) Lum, Taivon Watkins, Jonathan (Adam) Roney, Jared Dean, Marvin Jenkins, William (Chance) Collins

Points Discussed:

* Discussed AWS and created accounts
* Discussion of code that was received earlier this week
* Discussed placement of sensors and what kind of algorithm is needed to ensure pedestrians and cyclists are not counted
  + Distance apart, angle, distance measured by sensors
* Do we want to use AWS or Azure?
  + We settled on AWS
* Starting Software Requirements which is due September 21st

Date: 9/18/18

Time: 4:50 pm - 6:30 pm

Location: IT Building - Room 2206

Attendance: Taivon Watkins, Jonathan (Adam) Roney, Jared Dean, Marvin Jenkins, William (Chance) Collins, Katherine (Katie) Lum

Points Discussed:

* Discussed VPC, S3, RDS, EC2
* Selected a project management tool (Trello), made accounts
* Made personal accounts on AWS
* Worked on getting PuTTY installed, setup and running

Date: 9/19/18

Time: 5:30 pm - 6:45

Location: IT Building - Room 2206

Attendance: Taivon Watkins, Jonathan (Adam) Roney, Jared Dean, Marvin Jenkins, Katherine (Katie) Lum, William (Chance) Collins

Points Discussed:

* Software Requirements Document
* Adding Key Features list to Software Requirements Doc
* Meeting tomorrow to begin getting the existing hardware back up and running, as well as beginning to build approximately 3-4 additional nodes ( outside of this semester’s scope 11/30/2018)

Date: 9/20/18

Time: 3:30 - 6:30

Location: IT Building - Room 2206

Attendance: Taivon Watkins, Jonathan (Adam) Roney, Jared Dean, Marvin Jenkins, Katherine (Katie) Lum, William (Chance) Collins

Points Discussed:

* Discussed our need of proper organization as a team with Dr. Allen
* Discussed need to use Agile/SCRUM methodologies in order to make new and forward progress on this project with Dr. Allen
* Decided to implement geolocation on app and website
* Software Requirements Document
* Adding Key Features list to Software Requirements Document
* Constructing use case diagrams

Date: 9/21/18

Time: 1:30 - 3:45

Location: IT Building - Room 2206

Attendance: Taivon Watkins, Jonathan (Adam) Roney, Jared Dean

Points Discussed:

* Discussed AWS
* Finished up SRS
* Adjusted tentative GUI

Date: 9/23/2018

Time: 5:00 pm - 7:45

Location: IT Building - Room 2212

Attendance: Taivon Watkins, Jonathan (Adam) Roney, Jared Dean, Katie Lum, William (Chance) Collins

Points Discussed:

* Discussed layout of website and mobile app
* Spoke about the features in the applications
* Established roles
* Discussed what information needs to be stored within the trends graph

Date: 9/25/2018

Time: 4:00 pm - 6:00pm

Location: IT Building - Room 2212

Attendance: Taivon Watkins, Jonathan (Adam) Roney, Jared Dean, Katherine (Katie) Lum, William (Chance) Collins, Marvin Jenkins

Points Discussed:

* Set up AWS Instances
* Looking at hardware
* Discussing possible solutions for clock timing issue

Date: 9/27/2018

Time: 4:00pm - 5:45pm

Location: IT Building - Room 2206

Attendance: Taivon Watkins, Jared Dean, Katherine (Katie) Lum, William (Chance) Collins, Marvin Jenkins

Points Discussed:

* Connected to Pi gateway
* Turned on ultrasonic sensors to check battery levels
* Set up database
* Continuing website design

Date: 10/2/2018

Time: 4:00pm - 6:30pm

Location: IT Building - Room 2206

Attendance: Taivon Watkins, Jared Dean, Katherine (Katie) Lum, William (Chance) Collins, Marvin Jenkins, Jonathan (Adam) Roney

Points Discussed:

* Got both nodes and the gateway running
* Discussed clock drift issue and ways to possibly fix it
* Both website and app are on track to have the map system up and running by the end of this week

Date: 10/3/2018

Time: 10:30am - 11:00am

Location: Dr. Andrew Allen’s Office

Attendance: Taivon Watkins, Jared Dean, Katherine (Katie) Lum

Points Discussed:

* More important to get clock drift corrected than to optimize battery usage
* Send 4 packets per vehicle instead of 2
  + 1 and 3 from first node, 2 and 4 from second node

Date: 10/4/2018

Time: 4:00pm - 6:05pm

Location: IT Building - Room 2206

Attendance: Taivon Watkins, Jared Dean, Katherine (Katie) Lum, Marvin Jenkins, Jonathan (Adam) Roney

Points Discussed:

* Working to get nodes to immediately send packets rather than holding x amount of packets for y amount of time (switched back 11/30/2018)
* Getting maps on app and website working

Date: 10/7/2018

Time: 3:10 pm - 4:30 pm

Location: IT Building - Room 2206

Attendance: Taivon Watkins, Marvin Jenkins, Jonathan (Adam) Roney, William(Chance) Collins

Points Discussed:

* Software Design Document
* Getting gateway to communicate with the RDS instance (phone call to Kyle)

Date: 10/9/2018

Time: 4:00pm - 5:30pm

Location: IT Building - Room 2206

Attendance: Taivon Watkins, Jared Dean, Katherine (Katie) Lum, Marvin Jenkins, Jonathan (Adam) Roney, William(Chance) Collins

Points Discussed:

* Trying to get mobile app connected to the database
* Sending updates from gateway to nodes

Date: 10/14/2018

Time: 2:30pm - 3:30pm

Location: IT Building - Room 2212

Attendance: Taivon Watkins, Jared Dean, Jonathan(Adam) Roney

Points Discussed:

* Splitting the traffic event packets into entrance packets and exit packets
* Getting website to pull data from database

Upcoming week

• Get gateway to send updates to the nodes

• Connect mobile app to the database

• Send packets from node and have gateway determine whether it’s a car or not

• Continue development of website

Past Week:

Date: 10/16/2018

Time: 4:00pm -

Location: IT Building - Room 2206

Attendance: Taivon Watkins, Jared Dean, Katherine (Katie) Lum, Marvin Jenkins, Jonathan (Adam) Roney, William(Chance) Collins

Points Discussed:

* Trying to get nodes to send two packets per event instead of one
* Created “About” box in the mobile app under settings
* Cleaning up formatting on website
* Need to implement time sync for nodes
* Nodes need to be able to receive updates

Date: 10/18/2018

Time: 4:45pm - 6:00pm

Location: IT Building - Room 2206

Attendance: Taivon Watkins, Jared Dean, Katherine (Katie) Lum, Marvin Jenkins, Jonathan (Adam) Roney, William(Chance) Collins

Points Discussed:

* Future goal: use hardware’s unique MAC address as node ID instead of having it hard coded into the code
* Changes for Node:
  + If node 2 entrance time falls between entrance and exit times of node 1, then it = a car
    - I.e. if enter 2 > enter 1 && < exit 1
    - Exit 1 must also be > enter 1 && exit 2 must be > exit 1
    - Going to need to create a lot of exceptions to get any kind of accuracy
    - Currently having gateway to process the times and determine whether the data packet needs to be thrown out or accepted.
  + Need to make sure enter and exit times from same car can’t be sent in 2 separate “1-minute aggregate” packets
  + What is max delay between sent packets? -5secs? Longest between packets should be no more than a minute, but ideal would be no delay or time between packets
  + Need to be able to pair enter and exit times both within a node, between a node pair, and between many nodes (such as full scale implementation)
  + Need to time sync between nodes and gateway
    - Need to know how long the nodes go before the time drift is significant
      * This will let us know how often we need to force a re-sync
    - Will temp change be problematic/cause additional clock shift?

Date: 10/21/18

Time: 5:00pm - 7:45pm

Location: IT Building - Room 3212

Attendance: Taivon Watkins, William (Chance) Collins, Marvin Jenkins, Jared Dean

Points Discussed:

* Corrected the times on the nodes
* Decided on syncing the times on the nodes every day at midnight. (Ran gateway and nodes for two days and no signs of drift)
* Pulled information from database to website

Upcoming Week:

• Implement car-detection algorithm

• Implement time sync algorithm

• Connect mobile app to database

• Continue website development

Date: 10/23/18

Time: 3:30pm - 5:00pm

Location: IT Building - Room 2206

Attendance: Taivon Watkins, Jared Dean, Katherine (Katie) Lum, Marvin Jenkins, Jonathan (Adam) Roney, William (Chance) Collins

Points Discussed:

* Katie is switching from Hardware to Website
* Found way to assign “serial numbers” to nodes
  + These can be called in methods to identify the nodes digitally
  + Allows for one code to be used to run all nodes

Date: 10/28/18

Time: 4:30pm - 6:00pm (Hardware & App Team), 6:30pm - 8:15pm (Website Team)

Location: IT Building - Room 2206

Attendance: Taivon Watkins, Jared Dean, Katherine (Katie) Lum, Marvin Jenkins, Jonathan (Adam) Roney, William (Chance) Collins

Points Discussed:

* App having trouble connecting to database
* Still implementing time sync and car detection algorithms
* Working on Map and index pages of website

Upcoming Week:

• Continue hardware implementation

• Pull from database(App)

• Continue website design

Past Week:

Date: 10/30/18

Time: 3:30pm - 5:30pm

Location: IT Building - Room 2206

Attendance: Taivon Watkins, Jared Dean, Katherine (Katie) Lum, Marvin Jenkins, Jonathan (Adam) Roney, William(Chance) Collins

Points Discussed:

* Working on sketching out web page layouts
* Communicating with database team (Marvin) to determine what fields are already incorporated in the database and which ones we might need
  + Marvin just added latitude and longitude to the lot table
  + Already have a “Favorite Lot” type field
* Decide if we should keep map on webpage or not

Date: 11/1/18

Time: 4:45pm - 5:30pm

Location: IT Building - Room 2206

Attendance: Taivon Watkins, Jared Dean, Katherine (Katie) Lum, Marvin Jenkins, Jonathan (Adam) Roney, William(Chance) Collins

Points Discussed:

* Reverting back to pulling for single lots
* Pairing the traffic events from each node (for car-detection algorithm)
* Working on trends page once algorithm is complete

Date: 11/4/18

Time: 7:00pm - 8:15pm

Location: IT Building - Room 2206

Attendance: Taivon Watkins, Jared Dean, Marvin Jenkins, William(Chance) Collins

Points Discussed:

* Simulating car passing for car detection algorithm
* Distinguishing between admin and general user login
* Pinning the lots on map and making a lot a favorite

Upcoming Week:

• Finish up car detection algorithm and test it using actual cars

• Finish favorites selector and pins for map.

• Begin grabbing data from database to make trends page

Past Week:

Date: 11/6/18

Time: 3:45pm - 6:00pm

Location: IT Building - Room 2206

Attendance: Taivon Watkins, Jared Dean, Marvin Jenkins, William(Chance) Collins, Katherine (Katie) Lum, Jonathan (Adam) Roney

Points Discussed:

* Packet Transmission issue
* Favorite Lots
* Allow location switch in settings menu and also adding php to the ec2

Date: 11/8/18

Time: 3:45pm - 6:00pm

Location: IT Building - Room 2206

Attendance: Taivon Watkins, Jared Dean, Marvin Jenkins, William(Chance) Collins, Katherine (Katie) Lum, Jonathan (Adam) Roney

Points Discussed:

* STQA Document
* Tested car detection algorithm
* Packet Transmission working

Date: 11/11/18

Time: 5:30pm - 6:45pm

Location: IT Building - Room 2206

Attendance: Taivon Watkins, Marvin Jenkins, Jonathan (Adam) Roney

Points Discussed:

* App pulls open spots from database
* Fixing features within admin login
* Modifying design of website

Upcoming Week

- Continue modifying design of website

- Fix analytics/trends page

- Discuss whether a homepage will be used before login or have login page as index

- Continue working on app

Past Week:

Date: 11/13/18

Time: 3:30pm - 5:30pm

Location: IT Building - Room 2206

Attendance: Taivon Watkins, Jared Dean, Marvin Jenkins, William(Chance) Collins, Katherine (Katie) Lum, Jonathan (Adam) Roney

Points Discussed:

* Login button and link no longer are called “Admin Login”
* Working on design of home page
* Working on favorite lots (app)

Date: 11/15/18

Time: 4:45pm - 7:30pm

Location: IT Building - Room 2206 / Lot 42 Parking Lot

Attendance: Taivon Watkins, Jared Dean, Marvin Jenkins, William(Chance) Collins, Katherine (Katie) Lum, Jonathan (Adam) Roney

Points Discussed:

* Tested car detection algorithm with actual car (Worked!)
* Making sure RTC is adjusted before presentation (clock has drifted by seconds)
* Fixing car detection algorithm to work with cars exiting

Upcoming Week:

•Designing home page

•Designing login page

•Ensuring accuracy of analytics page

Past Week:

Date: 11/25/18

Time: 5:00pm - 7:15pm

Location: IT Building Lobby

Attendance: Taivon Watkins, Jared Dean, Marvin Jenkins, William(Chance) Collins, Jonathan (Adam) Roney

Points Discussed:

* Piecing together the website with the new home page and new login design
* Favorite Lots (website & app)
* Adding trends on userParking html page as well as trends tab
* Looking to attach sign post to nodes to test on Tuesday

Upcoming Week:

• Attach sign post

•Finish up website and app

• Correct times for video

• Presentation!!!

Date: 11/27/2018

Time: 3:30pm - 5:30pm / 6:30pm - 10:30pm

Location: IT Building 2206 / Lot 42 / PAC Lot

Attendance: Taivon Watkins, Jared Dean, Marvin Jenkins, William (Chance) Collins, Jonathan (Adam) Roney, Katherine (Katie) Lum

Points Discussed:

* Recording video (Entirely too cold for hardware)
* Implementing cars out algorithm for video
* Favorite Lots needs to automatically show on website

Date 11/28.2018

Time: 12:10pm - 1:30pm / 4:00pm - 12:00pm

Location: Stadium Parking Lot

Attendance: Taivon Watkins, Jared Dean ( 12:10 pm - 1:30 pm)

Taivon Watkins, Jared Dean, Marvin Jenkins, William (Chance) Collins, Jonathan (Adam) Roney, Katherine (Katie) Lum(4:00pm-7:00pm)

Points Discussed:

* Recording video( Warmer outside)
* Videos for entrance and exit uploaded to website

Date 11/29/.2018

Time: 11:00pm - 12:00am

Location: IT Building 2206

Attendance: Taivon Watkins, Jared Dean, Marvin Jenkins, Jonathan (Adam) Roney, William (Chance) Collins,Katherine Lum

Points Discussed:

* Practiced presentations
* Fixed small bugs in website

Section 5

# Software Testing Document

Parking Lot Availability Tracking

Software Engineering

CSCI 5530 A

Advisor/ Client: Dr. Andrew Allen

Advisor/ Client: Dr. Pradipta De

Prepared By:

Taivon Watkins

William Collins

Katie Lum

Jared Dean

Marvin Jenkins

Jonathan Roney

## System Level Requirements Validation Checklist

|  |  |  |
| --- | --- | --- |
| **Description** | **✔/✘** | **Comments** |
| Has overall feasibility been addressed? | ✔ |  |
| Have all clients been identified and polled for agreement? | ✔ |  |
| Have the overall function and behavior of the system been defined? | ✔ |  |
| Based on existing documentation/information, do you understand the system in the context of each of the views in the system engineering hierarchy? | ✔ |  |
| Have system processes been adequately (unambiguously) and consistently defined? | ✔ |  |
| Is system output and input adequately defined? | ✔ |  |
| Have system-level assumptions, simplifications, limitations, constraints and preferences been explicitly and unambiguously stated? | ✔ |  |
| Has simulation been done to demonstrate technological feasibility? | ✘ | Have not tested algorithm with an actual car |
| Has a data architecture been identified? | ✔ |  |
| Has an application (functional) architecture been defined? | ✔ |  |
| Has the required technology infrastructure for the system been adequately defined? | ✔ |  |
| Has requirements elicitation been performed at the system level? | ✘ | All requirements have not been fulfilled |
| Has business and technical feasibility been assessed? | ✔ |  |
| Have usage scenarios been created at the system level? | ✔ |  |
| Has a requirements management process been established for the system? | ✔ |  |
| Has allocation occurred for all system elements? | ✔ |  |
| Is the allocation for software reasonable and well-defined? | ✔ |  |
| Has a system model (e.g., a Hatley-Pirbhai model) been developed? | ✔ |  |

## Software Requirements Specification Checklist

|  |  |  |
| --- | --- | --- |
| **Description** | **✔/✘** | **Comments** |
| Do stated goals and objectives for software remain consistent with system goals and objectives? | ✔ |  |
| Have important interfaces to all system elements been described? | ✔ |  |
| Have all data objects been described? Have all attributes been identified? | ✔ |  |
| Do major functions remain within scope and has each been adequately described? | ✔ |  |
| Have functions been refined (elaborated) to an appropriate level of detail? | ✔ |  |
| Is information flow adequately defined for the problem domain? | ✔ |  |
| Are diagrams clear; can each stand-alone without supplementary text? | ✔ |  |
| Is the behavior of the software consistent with the information it must process and the functions it must perform? | ✔ |  |
| Have events and states been identified? | ✔ |  |
| Are design constraints realistic? | ✔ |  |
| Have technological risks been fully defined? | ✔ |  |
| Have alternative software requirements been considered? | ✔ |  |
| Have validation criteria been stated in detail; are they adequate to describe a successful system? | ✔ |  |
| Have inconsistencies, omissions or redundancy been identified and corrected? | \* | \*Clock requires upgrade to better, more consistent model, Time sync |
| Is the customer contact complete? | ✔ |  |

## Design Model Checklist

General Issues

|  |  |  |
| --- | --- | --- |
| **Description** | **✔/✘** | **Comments** |
| Does the overall design implement all explicit requirements? Has a traceability table been developed? | ✔ |  |
| Does the overall design achieve all implicit requirements? | \* | \*System requires more accurate clock, coupled with time updates from gateway to meet all requirements |
| Is the design represented in a form that is easily understood by outsiders? | ✔ |  |
| Is design notation standardized? Consistent? | ✔ |  |
| Does the overall design provide sufficient information for test case design? | ✔ |  |
| Is the design created using recognizable architectural and procedural patterns? | ✔ |  |
| Does the design strive to incorporate reusable components? | ✔ |  |
| Is the design modular? | ✔ |  |
| Has the design defined both procedural and data abstractions that can be reused? | ✔ |  |
| Has the design been defined and represented in a stepwise fashion? | ✔ |  |
| Has the resultant software architecture been partitioned for ease of implementation? Maintenance? | ✔ |  |
| Has a Design Specification been developed for the software? | ✔ |  |

### Data Design

|  |  |  |
| --- | --- | --- |
| **Description** | **✔/✘** | **Comments** |
| Have data objects defined in the analysis model been properly translated into required data structures? | ✔ |  |
| Do the data structures contain all attributes defined in the analysis model? | ✔ |  |
| Have any new data structures and/or attributes been defined at design time? | ✔\* | \*RDS instance required occasional updating to meet requirements |
| How do any new data structures and/or attributes relate to the analysis model and to overall user requirements? | ✔ |  |
| Have the simplest data structures required to do the job been chosen? | ✔ |  |
| Can the data structures be implemented directly in the programming language of choice? | ✔ |  |
| How are data communicated between software components? |  | Gateway pipes from node to RDS using Node.js |
| Do explicit data components (e.g., a database) exist? If so, what is their role? | ✔ | Houses collected data and allows for retrieval by the end user on the website |

### Architectural Design

|  |  |  |
| --- | --- | --- |
| **Description** | **✔/✘** | **Comments** |
| Has a library of architectural styles been considered prior to the definition of the resultant software architecture? | ✔ |  |
| Has architectural tradeoff analysis been performed? | ✔ | This architecture is required when creating an application that uses LoRa and an end user application. |
| Is the resultant software architecture a recognizable architectural style? | ✔ | Three-tier system |
| Has the architecture been exercised against existing usage scenarios? | ✔ |  |

### User Interface Design

|  |  |  |
| --- | --- | --- |
| **Description** | **✔/✘** | **Comments** |
| Have the results of task analysis been documented? | ✔ |  |
| Have goals for each user task been identified? | ✔ |  |
| Has an action sequence been defined for each user task? | ✔ |  |
| Have various states of the interface been documented? | ✔ |  |
| Have objects and actions that appear within the context of the interface been defined? | ✔ |  |
| Have the three "golden rules" (SEPA, 5/e, p. 402) been maintained throughout the GUI design? | ✔ |  |
| Has flexible interaction been defined as a design criterion throughout the interface? | ✔ |  |
| Have expert and novice modes of interaction been defined? | ✘ |  |
| Have technical internals been hidden from the casual user? | ✔ |  |
| Is the on-screen metaphor (if any) consistent with the overall applications? | ✔ |  |
| Are icons clear and understandable? | ✔ |  |
| Is interaction intuitive? | ✔ |  |
| Is system response time consistent across all tasks? | ✔ |  |
| Has an integrated help facility been implemented? | ✘ |  |
| Are all error messages displayed by the interface easy to understand? Do they help the user resolve the problem quickly? | ✔ |  |
| Is color being used effectively? | ✔ |  |
| Has a prototype for the interface been developed? | ✔ |  |
| Have user's impressions of the prototype been collected in an organized manner? | ✘ |  |

### Component-Level Design

|  |  |  |
| --- | --- | --- |
| **Description** | **✔/✘** | **Comments** |
| Has each algorithm been "desk-tested" to uncover errors? Is each algorithm correct? | ✔ |  |
| Is the design of the algorithm consistent with the data structured that the component manipulates? | ✔ |  |
| Have algorithmic design alternatives been considered? If yes, why was this design chosen? | ✔ | The final algorithm to detect a vehicle was selected as it was the most basic way of identifying a vehicle. More complex analyses can be built on top of this algorithm in the future. |
| Has the complexity of each algorithm been computed? | ✔\* | \*Complexity of all implemented algorithms well within acceptable levels |
| Have structured programming constructs been used throughout? | ✔ |  |

## Test Cases

### Embedded Systems Testing

#### Software Tests

|  |  |
| --- | --- |
| Test Case | E-CarDetection-0 |
| Test Preconditions | Gateway has successfully received packets from both nodes |
| Description | Send traffic events in data packet that will satisfy the carDetection algorithm |
| Purpose | To monitor the amount of cars passed |
| Methods | Have each node to send packets to the gateway containing traffic event times that signify that an object set off both of them at the same time. |
| Result | Pass |

|  |  |
| --- | --- |
| Test Case | E-CarDetection-1 |
| Test Preconditions | Gateway has successfully received packets from both nodes |
| Description | Send traffic events in data packet that will satisfy the carDetection algorithm and send traffic events that will not satisfy the algorithm |
| Purpose | To ensure the accuracy of the opening spots data being updated in the database |
| Methods | Have each node to send packets to the gateway containing traffic events and compare the start and end times of the traffic events on both nodes and should the start and end time follow the conditions of the algorithm increment the car counter variable and update the row value in database. |
| Result | Pass |

|  |  |
| --- | --- |
| Test Case | E-PacketTransmission-0 |
| Test Preconditions | Both nodes are able to send to gateway through single channel |
| Description | Send packets to gateway to test packet transmission reliability |
| Purpose | To mitigate packet loss |
| Methods | Create time slots for the nodes and have them send immediately after the previous nodes’ packet has transmitted. |
| Result | Pass |

#### Mobile Application Testing

|  |  |
| --- | --- |
| Test Case | M-Lot-0 |
| Description | To ensure the proper handling of a query from the homepage of the mobile application. There will be a picker filled with the lot number and when selected, it should return the appropriate spaces available. |
| Prerequisites | The lot must be in the database with the proper pre-verified output. |
| Input | Manual Selection of “Lot 42” in the picker |
| Output | Lot 42 should be displayed with the appropriate amount of open spots available that is in the database |
| Pass/Fail | Fail (Currently working towards) |

|  |  |
| --- | --- |
| Test Case | M-Location-0 |
| Description | To ensure the proper handling of getting permission for location services while using the app. This happens when you first install the application and navigate to the map tab. |
| Prerequisites | This must be a fresh install because it only asks this question on first launch. |
| Input | Open app and navigate to the map tab. |
| Output | Pop up message requesting location appears. |
| Pass/Fail | Pass |

|  |  |
| --- | --- |
| Test Case | M-Location-1 |
| Description | To ensure that location services will be turned off after initially allowing it. |
| Prerequisites | The user must have selected to allow location services to track them on the app’s first launch. |
| Input | Navigate to the settings page and turn the location switch off. |
| Output | Navigate back to the map page. It is no longer tracking your location |
| Pass/Fail | Pass |

|  |  |
| --- | --- |
| Test Case | M-Location-2 |
| Description | To ensure that location services can be turned back on after turning off |
| Prerequisites | You would have to have had location services turned on, then turned off in the settings page. |
| Input | Navigate to the settings page, turn the location switch back on |
| Output | Navigate back to the map tab and you will see it is tracking your location again. |
| Pass/Fail | Pass |

|  |  |
| --- | --- |
| Test Case | M-Location-3 |
| Description | To ensure that you cannot turn location services back on in the app if you initially did not allow it to track you in the initial install. You will need to go to your phone’s settings. |
| Prerequisites | You will need to have not allowed location services when prompted on the first install. |
| Input | Navigate to settings menu and notice that the location switch if off. Try to turn it on. |
| Output | Pop up message letting you know location services is disabled. (Two options: Cancel and Open Settings) |
| Pass/Fail | Pass |

|  |  |
| --- | --- |
| Test Case | M-Location-4 |
| Description | To ensure that the location switch pop up successfully takes you into your app location settings. |
| Prerequisites | You will need to have not allowed location services when prompted on the first install. Open settings and try to turn on location services. |
| Input | Select “Open Settings” from the pop up. |
| Output | App redirects to phone’s settings menu for location services. |
| Pass/Fail | Pass |

|  |  |
| --- | --- |
| Test Case | M-Map-0 |
| Description | To ensure that the map loads correctly with lot pins and zoomed in correctly. |
| Prerequisites | No prerequisites |
| Input | Open the app and navigate to the map tab. |
| Output | Map is automatically zoomed in on Georgia Southern, your location is/is not showing (depending on location services preferences) and all pins are showing parking lots. |
| Pass/Fail | Partial fail (Not all pins are showing - currently working towards) |

|  |  |
| --- | --- |
| Test Case | M-Map-1 |
| Description | To ensure that the correct number of open spots are shown when clicking on a map pin |
| Prerequisites | Open app and navigate to the map tab |
| Input | Click on lot 42 pin |
| Output | Action sheet pops up showing the correct number of available spots and two selectable options (Cancel and Directions) |
| Pass/Fail | Fail (Working towards) |

|  |  |
| --- | --- |
| Test Case | M-Map-2 |
| Description | To ensure that the Directions button opens up navigation to the parking lot |
| Prerequisites | Navigate to the map tab and click on lot 42 pin |
| Input | Click on the directions button on the action sheet |
| Output | Successfully opens Apple Maps with driving directions from your current location |
| Pass/Fail | Pass |

### Website Testing

|  |  |
| --- | --- |
| Test Case | M-Lot-0 |
| Description | To ensure that the selected lot name, capacity, and available spots displays correctly. |
| Prerequisites | Click on selector to display list of lots |
| Input | Select desired lot from the selector |
| Output | Successfully displays specific lot information |
| Pass/Fail | Pass |

|  |  |
| --- | --- |
| Test Case | M-Map-0 |
| Description | To ensure that the map is centered on the desired lot when chosen |
| Prerequisites | Lot must be chosen |
| Input | Select a lot from lot selector |
| Output | Successfully zooms and centers the selected lot location |
| Pass/Fail | Pass |

|  |  |
| --- | --- |
| Test Case | M-Login-0 |
| Description | To ensure that the admins can login successfully to the admin panel |
| Prerequisites | Must be on admin login screen |
| Input | Input credentials |
| Output | Successfully opens admin panel |
| Pass/Fail | Pass |

|  |  |
| --- | --- |
| Test Case | M-Logout-0 |
| Description | To ensure that the admins will be able to log out of the admin panel |
| Prerequisites | Must be logged into the admin panel |
| Input | Click logout button at top left corner |
| Output | Successfully logs out of admin panel |
| Pass/Fail | Pass |

|  |  |
| --- | --- |
| Test Case | M-Lot-0 |
| Description | To ensure that the admins will be able to edit any existing lot |
| Prerequisites | Click “edit lot” table to navigate to the edit lot page |
| Input | Select desired lot and input capacity and available spots |
| Output | Successfully updates database |
| Pass/Fail | Pass |

|  |  |
| --- | --- |
| Test Case | M-Lot-1 |
| Description | To ensure that the admins will be able to create new lots |
| Prerequisites | Click “create lot” at the top of the screen |
| Input | Choose campus and input new lot information |
| Output | Successfully updates the database to include the new lot |
| Pass/Fail | Pass |

|  |  |
| --- | --- |
| Test Case | M-User-0 |
| Description | To ensure that the admin will be able to change their password via admin panel |
| Prerequisites | Must be logged in and on the “user options” page |
| Input | Input credentials and new credentials |
| Output | Successfully updates password in the database |
| Pass/Fail | Pass |

|  |  |
| --- | --- |
| Test Case | M-Admin-0 |
| Description | To ensure that the admins will be able to add new admin authorizations |
| Prerequisites | Must be logged and on the “user options” page |
| Input | Input credentials and new admin information |
| Output | error |
| Pass/Fail | Failed |

# Hardware Setup

# 

# 

# 

# 

# Future Changes

## Nodes

* Time Sync - Gateway would be able to downlink updated time to node to update drifting RTC clock
* Software Update - Node should be able to receive software updates from gateway to automatically update the Arduino sketch
* Optimize Packet Transmission Algorithm - Fix algorithm to where traffic events are still recorded while the node is transmitting once a minute

## Gateway

* Downlink Messages
  + Currently the maximum amount of messages allowed to downlink in day is 10, including ack for uplink.
  + System updates
* OTAA (Over the Air Activation)
  + Currently using ABP (Activation By Personalization) which is a protocol that presets the AppSKey & NwkSKey session keys and pre-registers the end-device on the network. This approach is easier to set up, but it means that the encryption keys are comprised if a replay attack occurs.
  + OTAA is a protocol that has a join procedure that joins the gateway with the nodes during which a DevAddr is dynamically assigned and security keys are negotiated with the device.

## Hardware

* Node/ post mounting
  + Need a more secure way to mount the node to the posts

## Database

* Add Admin Table
  + Having a seperate table for admins will allow other admins to identify who the other admins are with their given admin ID. This will help with tracking what changes have been done to any lots or the distribution of admin privileges. This will also help with identifying an admin account that may have been compromised due to a successful attack.

## Website

* Map:
  + The map should eventually have the ability to drop pins on lots after a campus is selected.These pins should have a popup to display information based on lot data returned.
  + The map should have the ability to create a route from the user’s location to the lot that is selected. This was not implemented due to time constraints and the issue with markers not dropping on page load.
* Notifications:
  + For the admin section of the website, it should eventually be able to query the database at regular intervals and notify the admin if a lot is approaching maximum capacity. This functionality has yet to be implemented.
  + The favorite lots function is currently only able to set the favorite lot and is unable to retrieve it and set the selector values on start to the result.
* General:
  + Need to fix the fact that occasionally selectors will not load.
  + The trends page is currently set up so that it’s possible to load the graphs on start. Due to issues with functions not executing when the selected index is changed via JavaScript files and too many functions attempting to execute at once this feature is not implemented
* Security:
  + In the future, some security measures should be taken to prevent attacks like SQL injections and Cross-Site scripting.
  + The “add admins” feature should also write into a log to display which admins gave authorization to what users to prevent unlawful distribution of privileges.

## Mobile Application

* Cross Platform Application
  + Create a native android application or cross platform application to complement or replace the iOS application.