Software Design Document

System: Parking Lot Availability Tracker

Software Engineering

CSCI 5330A

Professor Andrew Allen

Team Members: Taivon Watkins (Team Leader)

William Collins

Katie Lum Jonathan Roney

Jared Dean

Marvin Jenkins

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

Abstract

Table of Contents

Chapter 1: Introduction

1.1 Purpose of System

1.2 Design Methodology

1.3 Definitions, Acronyms, and Abbreviations

1.4 Overview

Chapter 2: Proposed Software Architecture

2.1 Overview

2.2 Subsystem Decomposition

2.3 Persistent Data Management

Chapter 3: Object Design

3.1 Overview

3.2 Object Interaction

3.3 Detailed Class Design

Chapter 4: Glossary

Chapter 5: Appendix

Appendix A - Use Case Diagram

Appendix B - Diary of Meetings

Chapter 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**

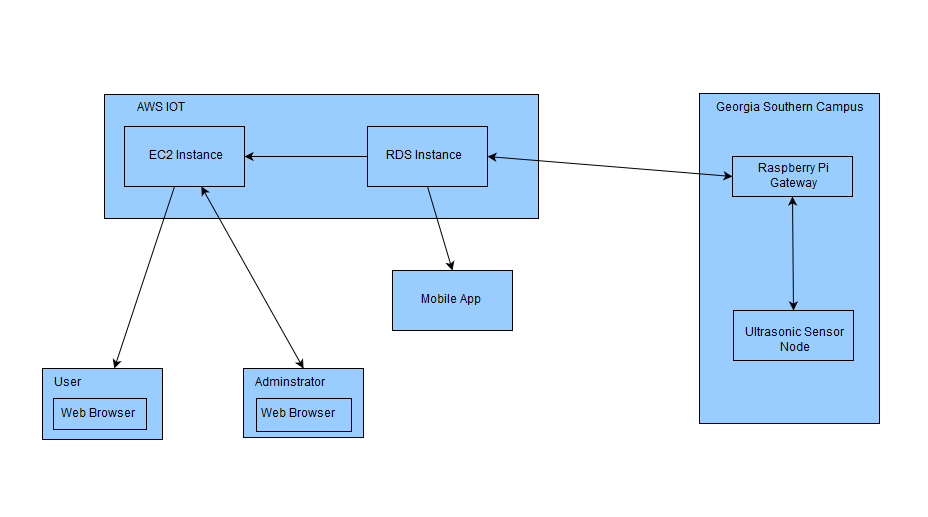
In subsequent chapters a detailed understanding of the GSU Parking Availability Tracking system will be obtained. In Chapter 2, Proposed Software Architecture, the architecture,relational database schema, and diagram will be discussed and explained. In Chapter 3, Object Design, the class diagrams will 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.

Chapter 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**

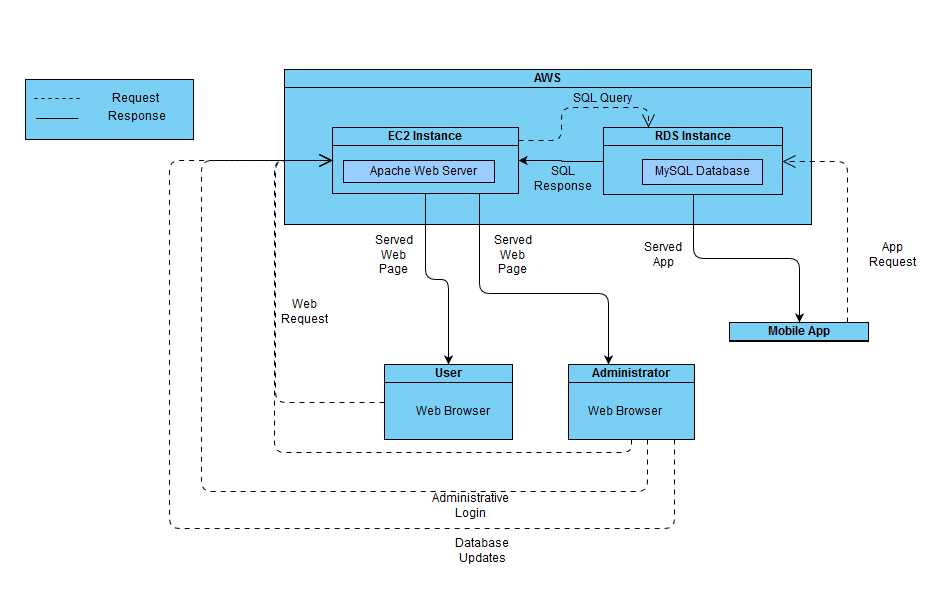


*Figure 2.1.0 - General System with Information flow*

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.

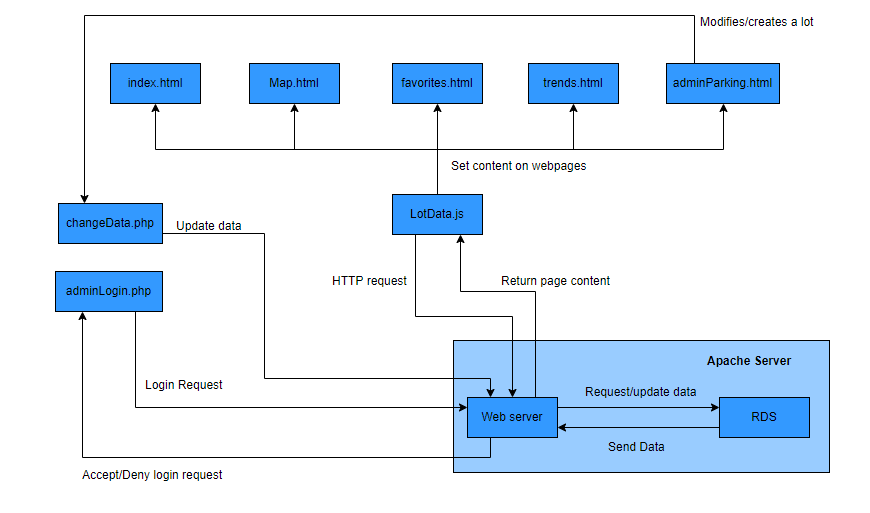
2.2 Subsystem Decomposition



*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.

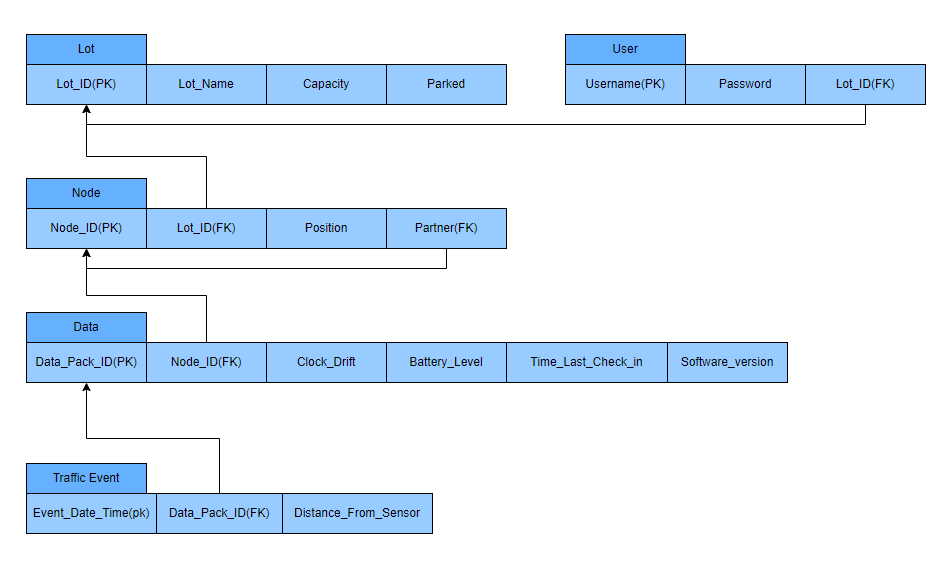


*Figure 2.2.1: diagram showing the interaction between web pages, the server, and other components*

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 at which 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. |

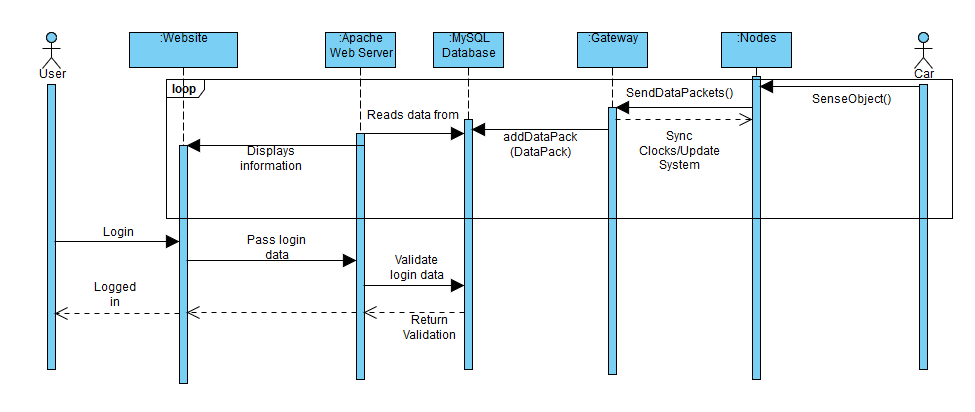
Chapter 3

Object Design

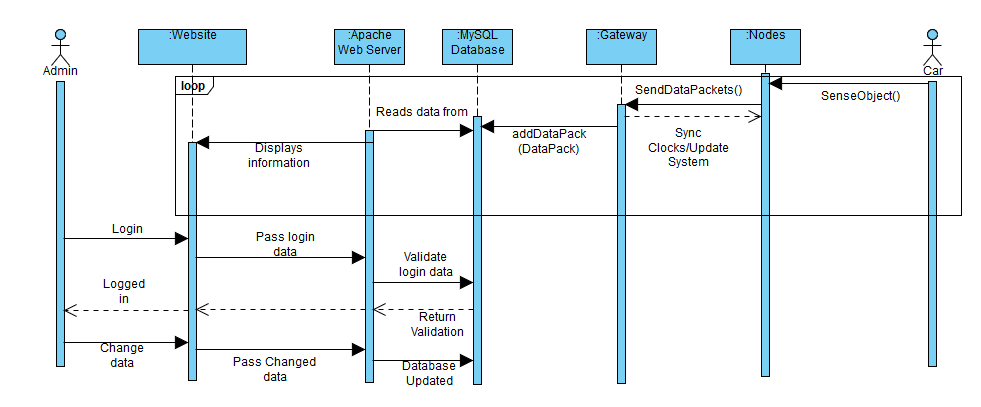
3.1 Overview

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.

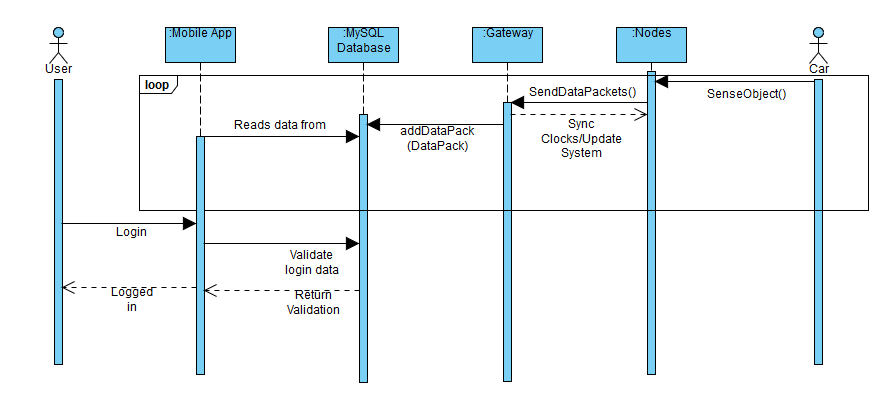
3.2 Object Interaction



*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.

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 and 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.

Chapter 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.

Chapter 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 - 7:00pm

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

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
* Dr. Allen says our app must be geolocation enabled such that it can tell the user which lot(s) are closest to that user
* Dr. Allen says we need an explicit list of everything we would like to try to accomplish by the end of the semester
  + This needs to include things we are not confident we can complete this semester
* Software Requirements Document
* Adding Key Features list to Software Requirements Doc
* 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
* Getting maps on app and website working

Date: 10/7/2018

TIme: 3:10pm - 4:30pm

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)