Software Requirements Specification

Parking Lot Availability

Version 2.0

Georgia Southern University

September 13th, 2018

Prepared By:

Taivon Watkins

William Collins

Katie Lum

Jared Dean

Marvin Jenkins

Jonathan Roney

**Table of Contents**

**1. Introduction**

* 1. Purpose
  2. Project Scope
  3. Defined Terms

1. **Description**
   1. Project Perspective
   2. Constraints
   3. Users
2. **Requirements**
   1. Product Requirements
   2. Functional Requirements
   3. Product Documentation Requirements
   4. External Interface Requirements

3.4.1 User Interface

3.4.2 Hardware Interface

3.4.3 Network Interface

3.4.4 Web Interface

* 1. Data Flow Diagram

1. **Core Features**

4.1. Use Case Diagram

4.1.1 Student/Faculty Use Case Details

4.1.2 Admin Use Case Details

**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 an 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.

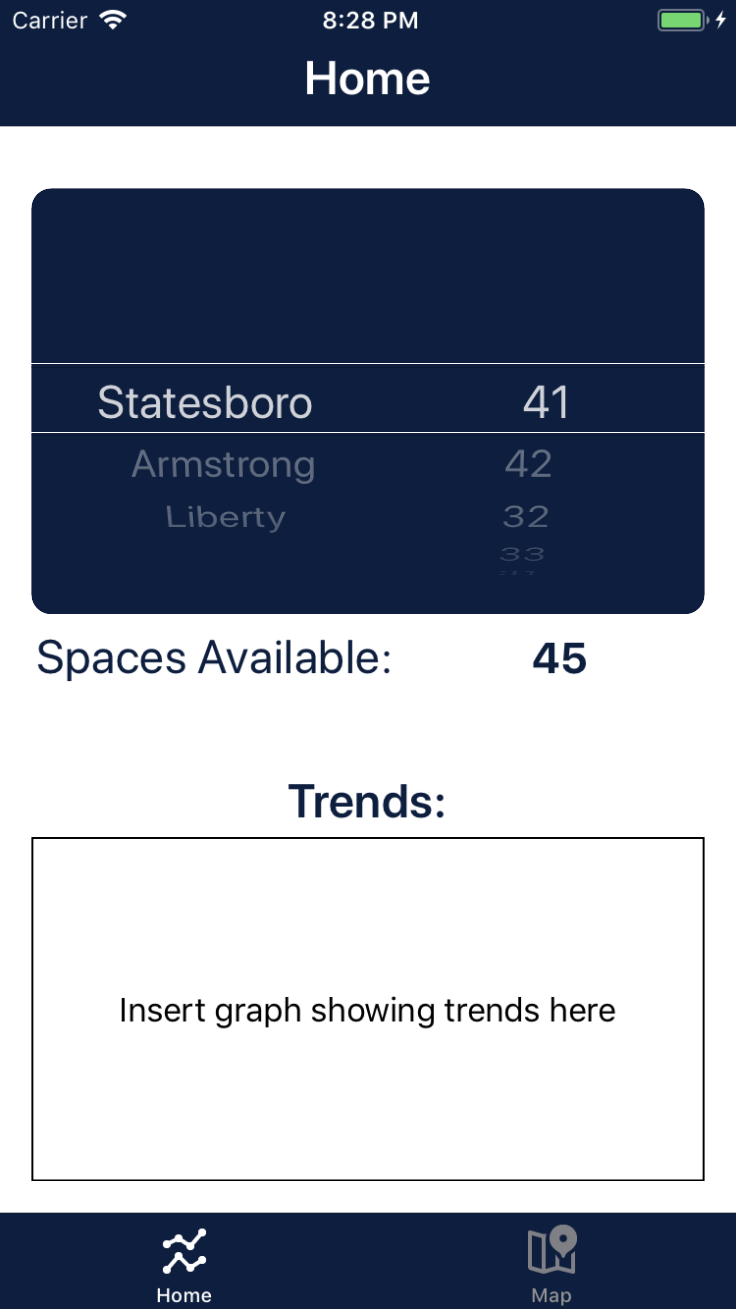
* Database
  + Relational Schema
  + ER Diagram
* AWS
  + Guides to instance creation and connection for all AWS instances
  + Guides on installing and running computational software and files to EC2
* Coding
  + Detailed descriptions in the form of comments shall be documented across all code written
* 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 nodes 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**

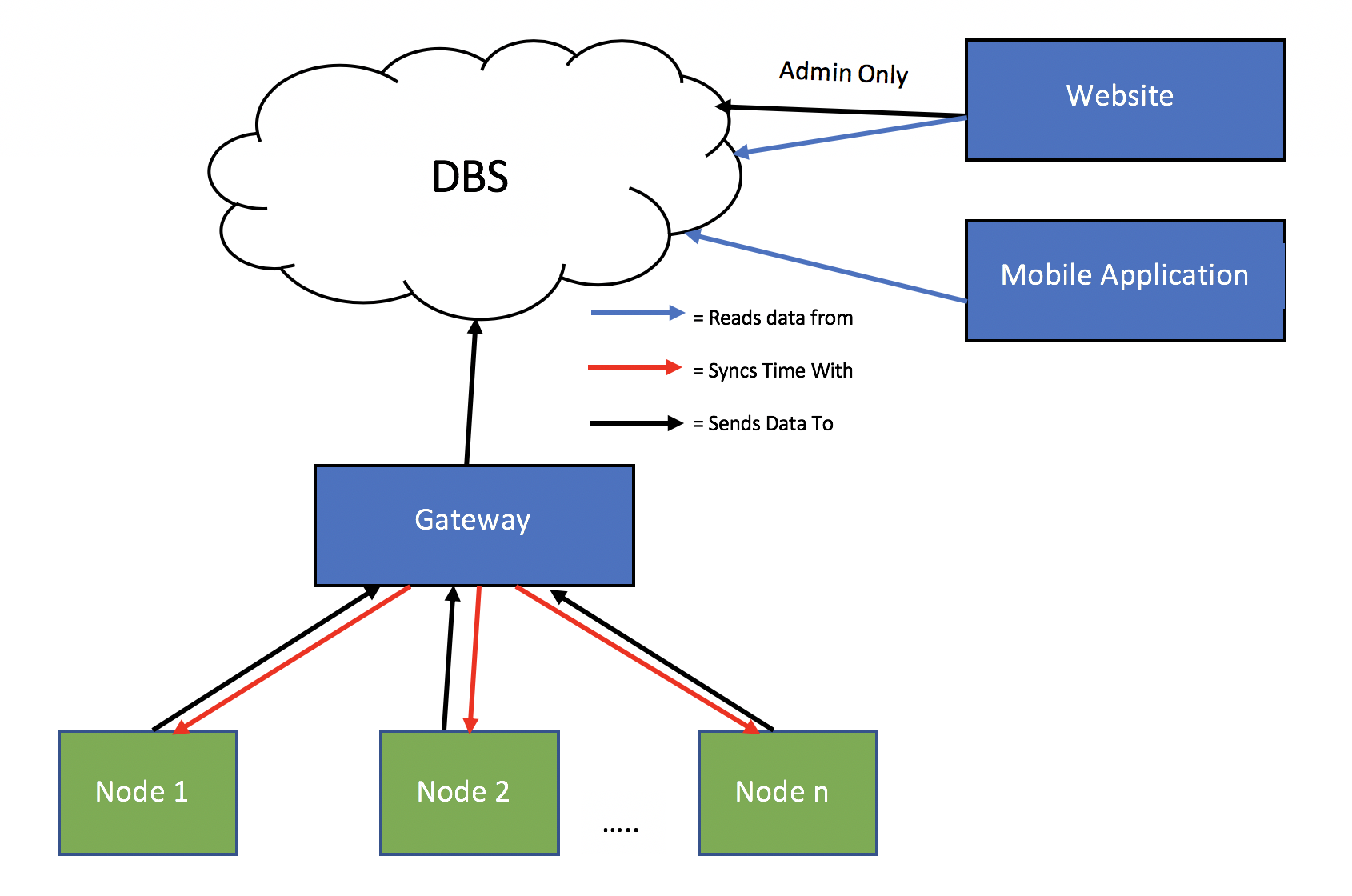
* LoRaWAN
  + LoRaWAN is a communication protocol that is designed to allow low-powered devices to communicate across long distances
  + It will be used to establish the network needed for the nodes and the gateways to communicate and transmit data between each other
  + LoRaWAN comes encrypted end-to-end 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 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.

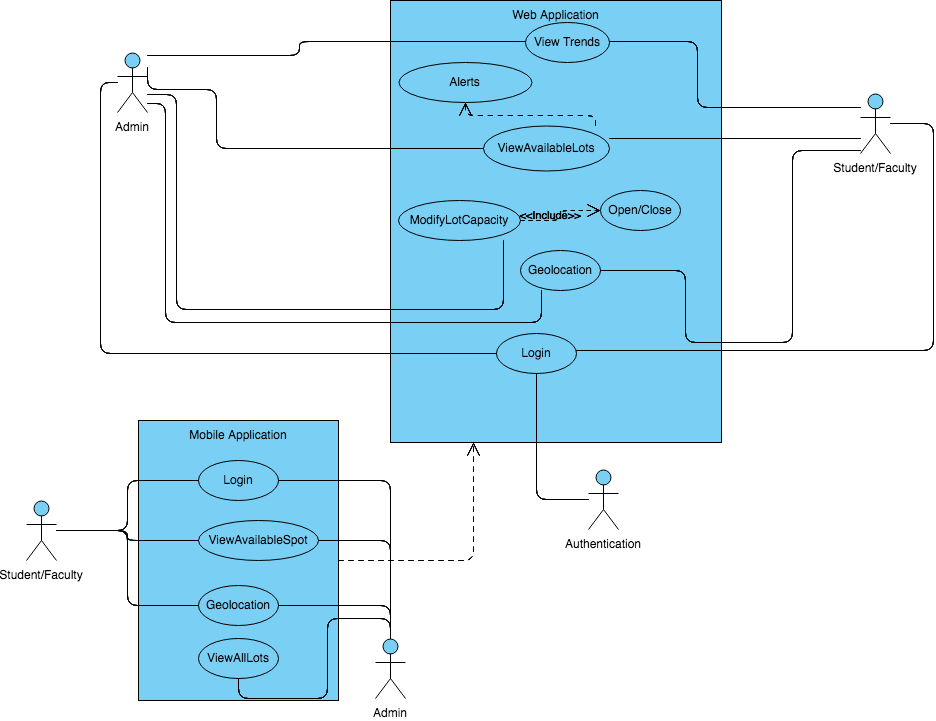
**3.5 Data Flow Diagram**

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 the student and faculty members but they’re combined. The system shall allow the admin to view all lots.