|  |
| --- |
| *CometPark* |
| **System Requirements Specification** |
| **SE 6387 Advanced Software Engineering Project**  **R.Z. Wenkstern**    ***02/20/2014*** |

|  |
| --- |
| **Group B 002** |
| **Arunkumar Manickam** |
| **Hariprasad Natarajan** |
| **Prasanna Venkatesh Venkitasamy** |
| **Rekha Muthulakshmi Nachadalingam** |

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Decription** | **Authors** |
| 1.0 | 02/20/2014 | Completed initial draft | Arunkumar,  Rekha Muthulakshmi |

Contents

[Revision History 2](#_Toc380659041)

[List of Figures 6](#_Toc380659042)

[List of tables 7](#_Toc380659043)

[1. Introduction 8](#_Toc380659044)

[1.1. Purpose 8](#_Toc380659045)

[1.2. Scope 8](#_Toc380659046)

[1.3. Overview 8](#_Toc380659047)

[2. Overall Description 8](#_Toc380659048)

[2.1. Product Perspective 8](#_Toc380659049)

[2.2. Product Functions 9](#_Toc380659050)

[2.3. User Characteristics 9](#_Toc380659051)

[2.4. Constraints 10](#_Toc380659052)

[2.5. Assumptions and Dependencies 10](#_Toc380659053)

[3. Hardware Specification 10](#_Toc380659054)

[3.1. Sensor 10](#_Toc380659055)

[3.1.1. Functionality 10](#_Toc380659056)

[3.1.2. Operational Requirements 10](#_Toc380659057)

[3.1.3. QoS Requirements 10](#_Toc380659058)

[3.1.4. Parametric Requirements 10](#_Toc380659059)

[3.1.5. Design Requirements 10](#_Toc380659060)

[3.2. XBee Module 10](#_Toc380659061)

[3.2.1. Functionality 10](#_Toc380659062)

[3.2.2. Operational Requirements 11](#_Toc380659063)

[3.2.3. QoS Requirements 11](#_Toc380659064)

[3.2.4. Parametric Requirements 11](#_Toc380659065)

[3.2.5. Design Requirements 11](#_Toc380659066)

[3.3. Solar Panel 11](#_Toc380659067)

[3.3.1. Functionality 11](#_Toc380659068)

[3.3.2. Operational Requirements 11](#_Toc380659069)

[3.3.3. QoS Requirements 11](#_Toc380659070)

[3.3.4. Parametric Requirements 11](#_Toc380659071)

[3.3.5. Design Requirements 11](#_Toc380659072)

[3.4. Battery 11](#_Toc380659073)

[3.4.1. Functionality 11](#_Toc380659074)

[3.4.2. Operational Requirements 11](#_Toc380659075)

[3.4.3. QoS Requirements 11](#_Toc380659076)

[3.4.4. Parametric Requirements 11](#_Toc380659077)

[3.5. Controller 12](#_Toc380659078)

[3.5.1. Functionality 12](#_Toc380659079)

[3.5.2. Operational Requirements 12](#_Toc380659080)

[3.5.3. QoS Requirements 12](#_Toc380659081)

[3.5.4. Parametric Requirements 12](#_Toc380659082)

[3.5.5. Design Requirements 12](#_Toc380659083)

[3.6. WiFi Dongle 12](#_Toc380659084)

[3.6.1. Functionality 12](#_Toc380659085)

[3.6.2. Operational Requirements 12](#_Toc380659086)

[3.6.3. QoS Requirements 12](#_Toc380659087)

[3.6.4. Parametric Requirements 12](#_Toc380659088)

[3.6.5. Design Requirements 12](#_Toc380659089)

[4. External Interface Requirements 12](#_Toc380659090)

[4. 1. User Interfaces 12](#_Toc380659091)

[4. 2. Hardware Interfaces 13](#_Toc380659092)

[4. 3. Software Interfaces 13](#_Toc380659093)

[4. 4. Communication Protocols and Interfaces 13](#_Toc380659094)

[5. System Features 13](#_Toc380659095)

[5.1. System Feature A 13](#_Toc380659096)

[5.1.1. Description 13](#_Toc380659097)

[5.1.2. Action/result 13](#_Toc380659098)

[5.1.3. Functional Requirements 14](#_Toc380659099)

[5.1.4. NFR 14](#_Toc380659100)

[5.2. System feature B 14](#_Toc380659101)

[5.2.1. Description 14](#_Toc380659102)

[5.2.2. Action/result 15](#_Toc380659103)

[5.2.3. Functional Requirements 15](#_Toc380659104)

[5.2.4. NFR 15](#_Toc380659105)

[5.3. System feature C 15](#_Toc380659106)

[5.3.1. Description 15](#_Toc380659107)

[5.3.2. Action/result 15](#_Toc380659108)

[5.3.3. Functional Requirements 16](#_Toc380659109)

[5.3.4. NFR 16](#_Toc380659110)

[5.4. System feature D 16](#_Toc380659111)

[5.4.1. Description 16](#_Toc380659112)

[5.4.2. Action/result 16](#_Toc380659113)

[5.4.3. Functional Requirements 16](#_Toc380659114)

[6. Non Functional Requirements 16](#_Toc380659115)

[6.1 Product NFR 16](#_Toc380659116)

[6.1.1 Usability 16](#_Toc380659117)

[6.1.2 Availability 16](#_Toc380659118)

[6.2 Process NFR 16](#_Toc380659119)

[6.2.1 Development 16](#_Toc380659120)

[6.2.2 Case tools 17](#_Toc380659121)

[6.2.3 Documentation 17](#_Toc380659122)

[6.3 External NFR 17](#_Toc380659123)

[6.3.1 University standard 17](#_Toc380659124)

[6.3.2 Device Standard 17](#_Toc380659125)

[6.3.3 Server Standard 17](#_Toc380659126)

[Appendix A: Glossary 18](#_Toc380659127)

[Appendix B: References 19](#_Toc380659128)

# List of Figures

|  |  |  |
| --- | --- | --- |
| **Figure Number** | **Name** | **Page** |
| Figure 1 | Block diagram of the CometPark System | 9 |

# List of tables

|  |  |  |
| --- | --- | --- |
| **Figure Number** | **Name** | **Page** |
| Table 1 | Color Codes of UTD Parking Department | 15 |

## 1. Introduction

## 1.1. Purpose

The purpose of this document is to give a detailed explanation of the requirements of the parking spot locating system, “CometPark”. The document will also cover each of the system’s features. This document is primarily intended to be proposed to the development team as a reference for developing the first version of the system.

## 1.2. Scope

The “Comet Park” is a parking spot locating system which helps people to find an empty parking spot in The University of Texas at Dallas parking lots using a mobile application.

The goal of the CometPark system is to help the student and staffs of UTD who commute to the campus by car in locating vacant parking spaces on-campus without difficulty. The system will be made available to the users as a mobile application that the users can access from their smart phones. The objective of the system is to provide a solution to the problem of finding vacant parking spots and to achieve maximum utilization of the lots.

## 1.3. Overview

The SRS document describes the “CometPark” parking spot locating system. The remainder of this document includes seven chapters.

The second section provides an overview of the overall system. This chapter also introduces product perspective, functions and user characteristics. Further, the chapter also mentions the system constraints and assumptions about the product.

The third chapter provides the hardware specification in detail and design requirements for each hardware component. The fourth chapter deals with the description of the different system interfaces. The fifth chapter provides system features and sixth chapter discuss the non-functional requirements. The seventh chapter discusses the other requirements.

The Appendixes in the end of the document includes the all references and glossary.

# 2. Overall Description

## 2.1. Product Perspective

This system will consist of Sensors, Controllers, Server and Mobile application. The application communicates with the cloud server which will have the updated status information by communicating with the controller .The communication between the controller and server will be through the internet. The controller receives the status of each parking spot through the XBee transceiver module present on each spot and on the controller. Below is the diagram of the CometPark system which illustrates the interactions between the sensor, controller, server and mobile application.

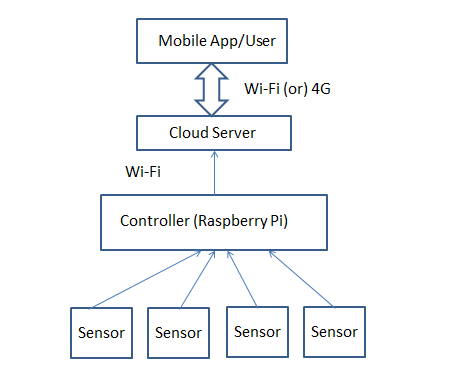


Figure 1 Block diagram of the CometPark system

## 2.2. Product Functions

The users of the application will use their smart phone to see the vacant parking spots that are close to them. The number of results to be shown can be set by the user. Since the parking system of UTD is based on color coded permits, the system will allow the user to set their color preference. The administrator of the system will manage the closure of some or all parking lots in case of events that happen in the campus.

## 2.3. User Characteristics

There are two types of users that interact with the system: The mobile app users, namely the students and staff of UTD who will interact via the mobile application and the administrator who will manage the system via a web application. Each of these users has different use of the system and so each of them has their own requirements.

The mobile application users will use only the mobile application to find the empty parking spots based on the color criteria they want. The admin will interact with the web application to manage the information about the parking spots and to inform the closure of partial/all parking lots in case of events or emergencies.

## 2.4. Constraints

The mobile application must run on the Android OS. The software must also use the language supported by the Android development environment, java and the Android SDK.

## 2.5. Assumptions and Dependencies

* The application will always be used on smart phones that have GPS capability and a data/wireless connection.
* The CometPark system will use wireless network connection to fetch data from the controllers, hence it is essential that there is a non-stop CometNet wifi connection for the controllers and server to function.
* The parking lots have access to a power source, preferably a renewable energy source.
* The application will be hosted on a cloud server(AWS) and the availability of the application is dependent on the availability of the web server.
* The cost and timelines described for this project are based on the estimates for a small number of parking spaces in a single parking lot. Implementing the system on a large scale will have a different budget and timeline.

# 3. Hardware Specification

## 3.1. Sensor

### 3.1.1. Functionality

The passive Infrared sensor will detect the presence of vehicle without any physical contact.

### 3.1.2. Operational Requirements

It will send the status to the controller via xBee when there is any change in parking spot availability.

### 3.1.3. QoS Requirements

### 3.1.4. Parametric Requirements

Operates with 5 to 20 V, power of 65mA, detection range 2 M and temperature range of -15 ~ to 70 F

### 3.1.5. Design Requirements

It must be placed in close proximity (Less than 2 m) to the parking spot so that it will spot the vehicle.

## 3.2. XBee Module

### 3.2.1. Functionality

The XBee module allows for communication between the sensor and the Controller using Radio Frequency.

### 3.2.2. Operational Requirements

XBee transmitter will send the parking spot status message from the sensor to the XBee receiver in the controller side.

### 3.2.3. QoS Requirements

### 3.2.4. Parametric Requirements

Operates with 3.3V and consumes 50 mA power.

### 3.2.5. Design Requirements

The XBee receiver must be within the range of (300 ft) the XBee transmitter to be able to receive status signal.

## 3.3. Solar Panel

### 3.3.1. Functionality

The function of solar panel is to generate electricity from the sun light.

### 3.3.2. Operational Requirements

Solar panel will charge up the battery which is then used to power the sensor and XBee.

### 3.3.3. QoS Requirements

### 3.3.4. Parametric Requirements

Generates 6V to the battery.

### 3.3.5. Design Requirements

It must be placed in an open space with direct exposure to sunlight.

## 3.4. Battery

### 3.4.1. Functionality

The function of battery is to store and supply power to XBee and Sensor.

### 3.4.2. Operational Requirements

It stores the power from the solar panel and supplies power to the XBee and sensors.

### 3.4.3. QoS Requirements

### 3.4.4. Parametric Requirements

The input voltage should be 6V and the temperature should be – 10 ~ +60 degree C.

**3.4.5. Design Requirements**

None

## 3.5. Controller

### 3.5.1. Functionality

The functionality of the controller is to send the parking spots status to the cloud server.

### 3.5.2. Operational Requirements

The XBee receiver attached to the Raspberry Pi receives the signal sent from various XBee transmitters about the parking spot availability and will process the data and update the status to the cloud server.

### 3.5.3. QoS Requirements

### 3.5.4. Parametric Requirements

Operates with 5 V and consumes 700 mA power.

### 3.5.5. Design Requirements

It should be within the Wi-Fi range.

It should have an uninterrupted power supply.

## 3.6. WiFi Dongle

### 3.6.1. Functionality

The WI-FI dongle allows the Raspberry Pi controller to connect to a wireless network.

### 3.6.2. Operational Requirements

It will allow the Raspberry Pi to send the status of parking spots to the cloud server via WI-FI network.

### 3.6.3. QoS Requirements

### 3.6.4. Parametric Requirements

Operates with 5V powered USB port.

### 3.6.5. Design Requirements

There should not be any interference at 2.4GHz wireless network.

# 4. External Interface Requirements

4. 1. User Interfaces

The CometPark system will be presented as a mobile application to the users and it will be available as a web interface for the administrator. The layout of the screens of the mobile application and the web interface will be determined in the later stages as the project progresses.

4. 2. Hardware Interfaces

The CometPark system has a number of hardware components that will interact with each other to make the parking space information available to the user. The following are the hardware interfaces between the various components of the system.

1. XBee Explorer Dongle – To interface the XBee module with the Raspberry Controller

2. Wifi Adapter – For wireless network connection between the Controller and the Cloud server

3. Resistors to manage voltage differences between the sensors and power source.

4. 3. Software Interfaces

The user will be able to access the CometPark system via a mobile interface and the administrator will access the system through a web interface. The system will use REST Web service to process the user request to get the vacant parking spot information.

4. 4. Communication Protocols and Interfaces

The communication between the sensors and the Raspberry Pi controller is through XBee module which uses the IEEE 802.15.4 networking protocol and operates on ISM 2.4 GHz. The Raspberry Pi controller communicates with the cloud server through Wi-Fi. The Wi-Fi adapter provides this wireless capability to the controller. The Wi-Fi adapter uses the 802.11n standard and operates on 2.4GHz.

# 5. System Features

## 5.1. System Feature A

Locate the vacant parking spots that are nearest to the user.

### 5.1.1. Description

The primary goal of this project is to help the users locate a vacant spot close to them to park their vehicle. When the user arrives at a location and refreshes the application, the application shall display the unoccupied spots in the parking lots close to him/her.

### 5.1.2. Action/result

When the user is near a building in the campus and wants to park his/her vehicle in a nearby parking lot, refreshing the mobile application will display the vacant parking spot that is nearest to him/her.

This feature will save a lot of time and fuel to the users who commute to the UTD campus via car. The application will prevent frustration among the students and staff in trying to find a vacant spot during peak hours.

### 

### 5.1.3. Functional Requirements

FR 1.1 -The CometPark system shall find the vacant parking spots that are of shortest distance from the user’s location.

FR 1.2 - The system shall be available to the user as a mobile application.

FR 1.3– The system shall be able to determine the status of the individual parking spots.

FR 1.4 – The system will allow the user to choose the number of nearest vacant spots (between 1 and 5) to be displayed to him/her.

FR 1.5 – The system shall identify whether or not there are any vacant parking spots in the lot.

### 5.1.4. NFR

NFR 1.1 – Adaptability – The system shall allow the users to set the number of spots to be displayed according to his/her needs.

## 5.2. System feature B

Provide options to the user to find parking spots with the preferred color code.

### 5.2.1. Description

According to the current Parking system at UTD, the users purchase parking permit of a particular color and must park their vehicles only in the spots allocated for that color. To meet this requirement, the application will provide the user with the option to set their preferred color and view only the vacant spots associated with that color.

The following are the options that will be available for the user to choose from:

1. Green

2. Gold

3. Orange

4. Evening Orange

5. Purple

6. Handicap

According to the color code chosen, the available parking spaces in the lots will be determined according to the following table specified by the UTD Parking Department in their website.

|  |  |
| --- | --- |
| **Permit Type** | **Permissible Parking Areas** |
| Green | Green and Extended parking areas |
| Gold | Gold, green and Extended parking areas |
| Evening Orange | Orange after 5.P.M ONLY- gold, green and extended parking areas anytime |
| Orange | Orange, Gold and Green and Extended parking areas |
| Purple | Purple, Orange, Gold, Green and Extended parking areas |
| Handicap | Areas designated for Handicap parking |

TABLE 1: COLOR CODES OF THE UTD PARKING DEPARTMENT

### 5.2.2. Action/result

The user can set the color code of their choice from a menu option that the application will provide. The application will locate the nearest available parking spaces that are allocated for the color set by the user.

This feature will allow the users to see only the vacant spots that correspond to the user’s parking permit.

### 5.2.3. Functional Requirements

FR 2.1-The application shall provide the user with the option to set the color code of their permit.

FR 2.2- The application shall display only the vacant parking spaces associated with the color chosen by the user if the user sets a color.

### 5.2.4. NFR

NFR 2.1 – Adaptability – The system shall allow the users to set their preferred color parking spaces that are to be searched for.

NFR 2.2 – Extensibility – The system shall allow the addition of new color codes for the parking spaces.

## 5.3. System feature C

Manage the closure of some of the parking lots in case of events and occasions.

### 5.3.1. Description

In the event of a special occasion, the Parking Management at UTD closes some of the parking lots to the staff and students. During such an event, the application must not display the spots in those lots even though they are unoccupied.

### 5.3.2. Action/result

The administrator of the application marks the parking lots that are closed to the staff and students . As a result of this feature, when the user searches for a parking spot, the application will not display the spaces in the lots that are marked as closed.

### 5.3.3. Functional Requirements

FR 3.1 – The system shall authenticate the administrator by validating the user credentials.

FR 3.2 – The system shall manage the partial closure of parking lots by not displaying the spaces in the lots indicated as closed by the administrator.

### 5.3.4. NFR

NFR 3.1 –Configurability– The system shall be configurable by allowing the administrator to handle the closure status of parking lots.

## 5.4. System feature D

Manage the closure of all parking lots in case of emergencies.

### 5.4.1. Description

In the event of an emergency, the application will display an alert message on its home page informing the closure of all the parking lots in the campus.

### 5.4.2. Action/result

The administrator of the application sets all the parking lots as closed and sets an emergency alert message to be displayed on the application’s homepage.

### 5.4.3. Functional Requirements

FR 4.1 –The application shall inform the users of the closure of all parking lots in the event of an emergency.

# 6. Non Functional Requirements

## 6.1 Product NFR

### 6.1.1 Usability

The system shall have an application that runs on the user’s smart phone and the admin of the system shall have a web interface to control the entire system, thus making the system easy to use for the respective users.

### 6.1.2 Availability

The system will be in running state 24x7. The user will have the option of finding the vacant spot any time.

## 6.2 Process NFR

### 6.2.1 Development

The development process to be used must be explicitly defined and must be based on harmony process for System Engineering.

### 6.2.2 Case tools

The Comet Park system shall use several CASE tools during various phase of the project lifecycle. This includes VERSIONONE for project planning management, Astah for UML diagram generation, Github for source code version control, IBM Rational Doors for documenting the requirements using IEEE STD 830, IBM Rhapsody for designing the architecture, and JUnit for testing.

### 6.2.3 Documentation

The CometPark system shall have documents that clearly define its functionality, use cases and architecture. All documentation will be reviewed and updated with every change that is implemented in the system. The documentation shall be maintained in such a way that they would be traceable and straight forward.

## 6.3 External NFR

### 6.3.1 University standard

The CometPark system shall comply with the standards imposed by UT-Dallas system for the Parking and Transportation department, CometNet Wi-Fi.

### 6.3.2 Device Standard

The devices used should follow certain standards that are set such that it doesn’t harm the environment. The XBee wireless sensor network Standard ISM 2.4 GHz shall be used by the devices for communication. The Wi-Fi wireless network Standard 2.4 GHz shall be used by the devices for communication. Also, the products shall follow FCC(Federal Communication Commission) radio frequency guidelines.

### 6.3.3 Server Standard

The CometPark system should comply with the policies that are mentioned in Amazon Web Service (AWS) service agreement. The service agreement can be accessed on their website : <http://aws.amazon.com/agreement/>

# Appendix A: Glossary

|  |  |
| --- | --- |
| **Term** | **Definition** |
| **UTD** | University of Texas at Dallas |
| **AWS** | Amazon Web Services |
| **IEEE 802.15.4** | IEEE 802.15.4 is a standard which specifies the physical layer and media access control |
| **XBee** | The XBee product is a modular product that make deploying wireless technology easy and cost-effective |
| **FCC** | The Federal Communications Commission (FCC) regulates interstate and international communications by radio, television, wire, satellite and cable. |
| **RF** | Radio frequency (RF) in the range of around 3 kHz to 300 GHz |
| **Astah** | A UML modeling tool |
| **CASE** | Computer Aided Software Engineering |
| **REST** | Representational state transfer (REST) is an architectural style consisting of a coordinated set of constraints applied to components, within a distributed hypermedia system. |
| **Raspberry Pi** | The Raspberry Pi is a credit-card-sized single-board computer for processing the parking data. |
| **IR Sensor** | A sensor which uses infrared rays and measures a physical quantity and converts it into a signal which can be read by an instrument. |
| **Cloud Server** | Cloud Servers are on-demand virtual machines engineered to deliver performance and reliability. |
| **Wi-Fi Dongle** | A Wi-Fi dongle is USB adapter that can be connected to a personal computer or a laptop, to enable it join or connect to a wireless network. |

# Appendix B: References

1. <https://www.utdallas.edu/parking/permits.html>