|  |
| --- |
| *Comet Parking* |
| **System Requirements Specification** |
| **SE 6387 Advanced Software Engineering Project**  **Group E**    ***02/20/2014*** |

|  |
| --- |
| **Group E** |
| **Aishwarya Muthukumaran** |
| **Rizwan Noorani** |
| **Sanjitha Srinivasan** |
| **Vignesh Sridhar** |

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Description** | **Authors** |
| 1.0 | 2/19/2014 | Completed initial draft | Rizwan, Aishwarya, Vignesh, Sanjitha |
| 1.1 | 2/23/2014 | Updated SRS with additional system features | Rizwan, Aishwarya, Vignesh, Sanjitha |

Table of Contents

[Revision History 2](#_Toc380946179)

[1. Introduction 1](#_Toc380946180)

[1.1. Purpose 1](#_Toc380946181)

[1.2. Scope 1](#_Toc380946182)

[1.3. Overview 1](#_Toc380946183)

[2. Overall Description 1](#_Toc380946184)

[2.1. Product Perspective 1](#_Toc380946185)

[2.2. Product Functions 2](#_Toc380946186)

[2.3. User Characteristics 2](#_Toc380946187)

[2.4. Constraints 2](#_Toc380946188)

[2.5. Assumptions and Dependencies 2](#_Toc380946189)

[3. Hardware Specification 2](#_Toc380946190)

[3.1. IR Sensor 2](#_Toc380946191)

[3.1.1. Functionality 2](#_Toc380946192)

[3.1.2. Operational Requirements 3](#_Toc380946193)

[3.1.3. QoS Requirements 3](#_Toc380946194)

[3.1.4. Parametric Requirements 3](#_Toc380946195)

[3.1.5. Design Requirements 3](#_Toc380946196)

[3.2. Adafruit Trinket - Micro-controller 3](#_Toc380946197)

[3.2.1. Functionality 3](#_Toc380946198)

[3.2.2. Operational Requirements 3](#_Toc380946199)

[3.2.3. QoS Requirements 3](#_Toc380946200)

[3.2.4. Parametric Requirements 3](#_Toc380946201)

[3.2.5. Design Requirements 4](#_Toc380946202)

[3.3. Bluefruit EZ-Link - Bluetooth Serial Link 4](#_Toc380946203)

[3.3.1. Functionality 4](#_Toc380946204)

[3.3.2. Operational Requirements 4](#_Toc380946205)

[3.3.3. QoS Requirements 4](#_Toc380946206)

[3.3.4. Parametric Requirements 4](#_Toc380946207)

[3.3.5. Design Requirements 4](#_Toc380946208)

[3.4. Raspberry Pi 4](#_Toc380946209)

[3.4.1. Functionality 4](#_Toc380946210)

[3.4.2. Operational Requirements 5](#_Toc380946211)

[3.4.3. QoS Requirements 5](#_Toc380946212)

[3.4.4. Parametric Requirements 5](#_Toc380946213)

[3.4.5. Design Requirements 5](#_Toc380946214)

[3.5. Bluetooth 3.0 High Speed & Wi-Fi Combo USB Mini Adapter 5](#_Toc380946215)

[3.5.1. Functionality 5](#_Toc380946216)

[3.5.2. Operational Requirements 5](#_Toc380946217)

[3.5.3. QoS Requirements 5](#_Toc380946218)

[3.5.4. Parametric Requirements 6](#_Toc380946219)

[3.5.5. Design Requirements 6](#_Toc380946220)

[4. External Interface Requirements 6](#_Toc380946221)

[4. 1. User Interfaces 6](#_Toc380946222)

[4. 2. Hardware Interfaces 8](#_Toc380946223)

[4. 3. Software Interfaces 8](#_Toc380946224)

[4. 4. Communication Protocols and Interfaces 8](#_Toc380946225)

[5. System Features 8](#_Toc380946226)

[5.1 Detect a parked vehicle 8](#_Toc380946227)

[5.1.1. Description 8](#_Toc380946228)

[5.1.2. Action/Result 8](#_Toc380946229)

[5.1.3. Functional Requirements 9](#_Toc380946230)

[5.1.4 Non-Functional Requirements 9](#_Toc380946231)

[5.2 Inform the controller of a parked vehicle 9](#_Toc380946232)

[5.2.1. Description 9](#_Toc380946233)

[5.2.2. Action/Result 9](#_Toc380946234)

[5.2.3. Functional Requirements 9](#_Toc380946235)

[5.2.4 Non-Functional Requirements 9](#_Toc380946236)

[5.3 Real-time update to the server 9](#_Toc380946237)

[5.3.1. Description 9](#_Toc380946238)

[5.3.2. Action/Result 9](#_Toc380946239)

[5.3.3. Functional Requirements 10](#_Toc380946240)

[5.3.4 Non-Functional Requirements 10](#_Toc380946241)

[5.4 Display available parking spots 10](#_Toc380946242)

[5.4.1. Description 10](#_Toc380946243)

[5.4.2. Action/Result 10](#_Toc380946244)

[5.4.3. Functional Requirements 10](#_Toc380946245)

[5.4.4. Non-Functional Requirements 10](#_Toc380946246)

[5.5 Admin Web-Interface 10](#_Toc380946247)

[5.5.1. Description 10](#_Toc380946248)

[5.5.2. Action/Result 11](#_Toc380946249)

[5.5.3. Functional Requirements 11](#_Toc380946250)

[5.5.4. Non-Functional Requirements 11](#_Toc380946251)

[6. Non-functional Requirements 11](#_Toc380946252)

[6.1 Product NFR 11](#_Toc380946253)

[6.2 Process NFR 11](#_Toc380946254)

[6.3 External NFR 11](#_Toc380946255)

[Appendix A: Glossary 12](#_Toc380946256)

[Appendix B: References 14](#_Toc380946257)

# 1. Introduction

## 1.1. Purpose

The purpose of this document is to specify the requirements for the Comet Parking System. It contains the various features, interfaces and its interactions with the external systems. This document is primarily intended to be proposed to the UTD Parking Department for its approval.

## 1.2. Scope

The Comet Parking System is a mobile/web-based application designed to assist students, faculty members and employees of UTD in finding parking spots tailored to their preferences which saves users’ valuable time.

The sensors on each parking spot will be sending information to the centralized location. The users can then view the information via their Smartphone in the form of an interactive dashboard. The system assumes that the users own a smartphone with Wi-Fi capabilities. The system has capabilities to communicate with the UTD Parking Department’s database.

## 1.3. Overview

The document contains functional requirements, non-functional requirements, constraints and the various dependencies of the system.

# 2. Overall Description

## 2.1. Product Perspective

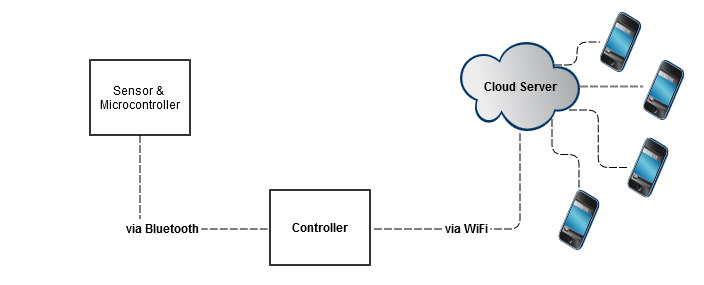


Figure 1

## 2.2. Product Functions

With the smartphone application, the user will be able to view vacant parking spots according to their permit type and preferred area.

## 2.3. User Characteristics

There are two types of users of the system. The first type includes students, faculty members and employees of UTD (not including the UTD Parking Department Administrators). The second type includes the UTD Parking Department Administrators.

The first type of users can only access Smartphone application to view the vacant parking spots.

The second type of user accesses the application via web for analyzing the data and generating reports.

## 2.4. Constraints

The following constraints were identified with regards to the project:

* Since the components of the parking system are being ordered from several third party vendors, there is possibility that they might not arrive at the same time and can stall the progress of the project.

## 2.5. Assumptions and Dependencies

The following assumptions were made with regards to the project:

* UTD will authorize the installation and operation of the parking system.
* The team members of the project will adhere to the guidelines in the Project Plan and ensure regular participation in the execution of the Project Plan.

# 3. Hardware Specification

## 3.1. IR Sensor

### 3.1.1. Functionality

The IR sensor module light is adaptable to the environment; it has a pair of infrared transmitting and receiving tube. The Transmitting tube infrared emits a certain frequency continuously. When the sensor meets with obstacles (reflecting surface), the frequency is reflected back to the receiving tube. The comparator circuit processes the reflected frequency, as a result of which, the green indicator will light up and outputs a digital signal (a low level signal).

### 3.1.2. Operational Requirements

The IR sensor is connected with the Adafruit trinket in a breadboard. The trinket, which is a microcontroller, triggers the sensor to send/receive IR signals at a particular interval and detects the voltage difference, which occurs when a car is parked. The micro controller will be able to power the IR sensor.

### 3.1.3. QoS Requirements

* The IR sensor shall not exceed the voltage limit of 5V.
* The sensor shall be able to detect the presence of a vehicle between 2 to 30 cm and a detection angle of 35 °.

### 3.1.4. Parametric Requirements

* The sensor must be fabricated within 3.2cm\*1.4cm and should weigh less than 2 grams.

### 3.1.5. Design Requirements

* The sensor shall be designed to be able to connect to the micro-controller.

## 3.2. Adafruit Trinket - Micro-controller

### 3.2.1. Functionality

The trinket is a micro-controller board, built around the ATMEL ATtiny 85 chip and includes a [CPU](http://www.wisegeek.com/how-does-a-cpu-work.htm), RAM, ROM and I/O ports. This device is self-contained and independent.

### 3.2.2. Operational Requirements

The micro-controller powers and triggers the IR sensor at regular intervals and then detects the difference in voltage when a car is parked. The micro-controller then instructs the Bluefruit EZ-Link shield to transfer the change in state information to the Raspberry Pi, which uploads the information in the cloud.

### 3.2.3. QoS Requirements

* The micro-controller shall be able to withstand up to 16V input.
* The micro-controller shall include safety features including reverse-polarity protection, thermal and current-limit protection.

### 3.2.4. Parametric Requirements

* The micro-controller shall not exceed 1.2" x 0.6" x 0.2" / 31mm x 15.5 x 5mm.
* The micro-controller shall not exceed 2 grams in weight.

### 3.2.5. Design Requirements

* The micro-controller shall be equipped with hardware capability for sensor interfacing.

## 3.3. Bluefruit EZ-Link - Bluetooth Serial Link

### 3.3.1. Functionality

The Bluefruit EZ-Link is a 'SPP' serial link client device, which can pair with a computer or a tablet and be detected as a serial/COM port. It transmits/receives data from source to destination wirelessly via Bluetooth protocol.

### 3.3.2. Operational Requirements

The Bluefruit EZ-Link is connected with the trinket (microcontroller). When the micro-controller detects a change in state, it instructs the EZ-Link to wirelessly send the data (sensor ID, timestamp, and flag status.) via Bluetooth to the Raspberry Pi.

### 3.3.3. QoS Requirements

* The module shall automatically detect and change the serial baud rate.
* The DTR/RTS/DSR flow control pins shall automatically sync to the computer serial port.

### 3.3.4. Parametric Requirements

* The module must be built with dimensions of 20.4mm / 0.8" x 41mm / 1.6" x 4mm / 0.2".
* The module shall not exceed 5 grams in weight.

### 3.3.5. Design Requirements

* The module shall notify the computer when the DSR input pin changes its states.
* The module shall indicate when the data is sent & received.

## 3.4. Raspberry Pi

### 3.4.1. Functionality

The Raspberry Pi is a computer that functions as a main controller and can communicate with the micro-controller via Bluetooth dongle.

### 3.4.2. Operational Requirements

The Raspberry Pi acts as the main controller. It receives data from the micro-controller when a change is detected. The data is uploaded in the cloud via Wi-Fi/Bluetooth module periodically.

### 3.4.3. QoS Requirements

* The Raspberry Pi shall be able to work at turbo preset of 1000 MHz ARM, 500 MHz core, 600 MHz SDRAM, 6 overvolt for advanced processing operations.

### 3.4.4. Parametric Requirements

* The Raspberry Pi should not exceed the size of 100mm x 60 mm in dimension.
* The Raspberry Pi should weigh less than 50 grams.

### 3.4.5. Design Requirements

* The Raspberry Pi shall be powered via the USB data ports.
* The Raspberry Pi shall include SD card slot, Ethernet adapter and peripherals like 8 × GPIO, UART, I²C bus, SPI bus with two chip selects, I²S audio, +3.3 V, +5 V and ground.

## 3.5. Bluetooth 3.0 High Speed & Wi-Fi Combo USB Mini Adapter

### 3.5.1. Functionality

The Bluetooth 3.0 High Speed & Wi-Fi Combo USB Mini Adapter enables sending and receiving data back and forth the source and destination wirelessly using the Bluetooth and Wi-Fi protocol simultaneously.

### 3.5.2. Operational Requirements

The module is attached to the Raspberry Pi to make it Bluetooth-enabled, through which it can communicate with the micro controllers. It also provides a dual purpose of transmitting the data via Wi-Fi to the cloud.

### 3.5.3. QoS Requirements

* The module shall be able to implement the latest Bluetooth 3.0 protocol with the range of 10 Meters (33 feet).
* The module shall support 802.11n Wi-Fi capabilities and will operate in the 2.4GHz~2.4835GHz ISM Band.
* The module shall be designed to support Wi-Fi transfer rate of up to 150 Mbps and Bluetooth of up to 24 Mbps without interference between the two signals.

### 3.5.4. Parametric Requirements

* The module must be constructed with the dimensions of 6 x 1 x 7 inches
* The module shall not exceed 15 grams.

### 3.5.5. Design Requirements

* The module shall support data transfer through Bluetooth 3.0 and Wi-Fi 802.11n simultaneously.
* The module shall support backward compatibility with Bluetooth 2.0.
* The module shall be compatible with Windows 7.

# 4. External Interface Requirements

## 4. 1. User Interfaces

First time users of the comet parking application will see the Register page when he/she opens the application (Figure 2) via their smartphone. The users will have to register using the following registration form.

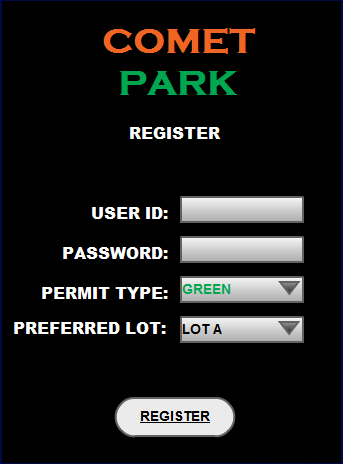


Figure 2

After registering successfully, the users can view their preferred parking lot structure. The available spots are displayed in green and unavailable spots are in orange. The grey spots display the area where the users are not permitted to park their car (Figure 3).

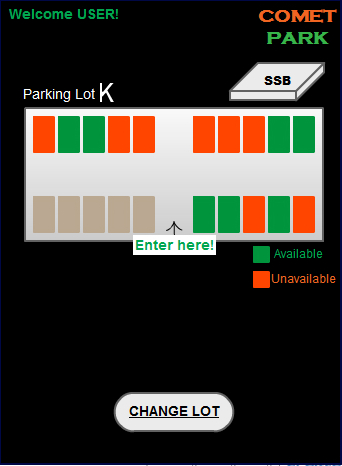


Figure 3

The users can also view different parking lots by clicking on the Change Parking Lot button (Figure 4).

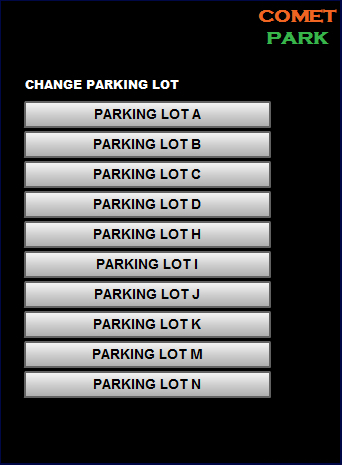


Figure 4

## 4. 2. Hardware Interfaces

The major hardware devices used are the IR sensor, Adafruit Trinket micro-controller, Raspberry Pi controller and a Smartphone. The IR sensors and the micro-controller are placed in each parking spot, which detect the vehicle presence. The Raspberry Pi receives the signal from the micro-controller and updates the database server.

## 4. 3. Software Interfaces

The smartphone application requests the recently updated parking spot information from the server. The smartphone can only view information from the server.

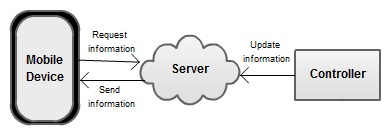


Figure 5

## 4. 4. Communication Protocols and Interfaces

The communication between various parts of the system is very essential since they are inter-dependent. The sensor and microcontroller communicate with the Raspberry Pi via Bluetooth 3.0. The Raspberry Pi updates the server via Wi-Fi using TCP/IP protocol.

# 5. System Features

## 5.1 Detect a parked vehicle

### 5.1.1. Description

The sensor and the micro-controller on each parking spot will identify the presence of a vehicle.

### 5.1.2. Action/Result

The sensor senses the presence of an object and the micro-controller identifies whether the object is a parked vehicle or not.

### 5.1.3. Functional Requirements

* The system shall be able to detect a parked vehicle.

### 5.1.4 Non-Functional Requirements

* The system shall be able to detect the presence of a vehicle between 2 to 30 cm and at a detection angle of 35 °.

## 5.2 Inform the controller of a parked vehicle

### 5.2.1. Description

When the micro-controller detects a parked vehicle, it informs the controller connected to it via Bluetooth.

### 5.2.2. Action/Result

The micro-controller sends its ID and a set parked flag to the controller.

### 5.2.3. Functional Requirements

* The system shall be able to establish communication between the micro-controller on each parking spot to the controller.

### 5.2.4 Non-Functional Requirements

* The micro-controller shall be able to communicate with the controller in a reliable manner.

## 5.3 Real-time update to the server

### 5.3.1. Description

The controller will perform a real-time update on the server with the parking spot occupancy information.

### 5.3.2. Action/Result

The controller updates the server with the controller ID, its parking lot status (includes each parking spot and its availability status) each time a car gets parked.

### 5.3.3. Functional Requirements

* The system shall be able to receive real-time information from each parking lot.
* The controller shall be able to establish communication with the server via Wi-Fi

### 5.3.4 Non-Functional Requirements

* The controller shall be able to establish communication with the server via Wi-Fi with a minimum speed of 100Mbps.

## 5.4 Display available parking spots

### 5.4.1. Description

This feature enables the user to view the vacant parking spots based on the parking lot of the user’s choice and the parking permit owned by the user. The parking spots are displayed on the smartphone application for each parking lot. The vacant parking spots are highlighted in green and the occupied parking spots are highlighted in red.

### 5.4.2. Action/Result

The available parking spots for a specific parking lot are displayed every time the user accesses the smartphone application of the parking system. The user can change the parking lot by clicking the Change Parking Lot Button.

### 5.4.3. Functional Requirements

* The system shall display the available parking spots in green and the occupied parking spots in red for every parking lot.
* The system shall enable the users to change the parking lot in which the parking spots are displayed by providing a list of available parking lots.

### 5.4.4. Non-Functional Requirements

* The system shall be able to change the parking lot and display the available parking spots in that lot within ten milliseconds.

## 5.5 Admin Web-Interface

### 5.5.1. Description

This feature enables the customer to view and generate customized reports from the parking information available on the cloud.

### 5.5.2. Action/Result

The customer requests the system for a customized report. The system provides the requested report.

### 5.5.3. Functional Requirements

* The system shall enable the customer to request a customized report from the parking data available on the cloud.
* The system shall display the report requested by the customer.

### 5.5.4. Non-Functional Requirements

* The system shall not exceed a maximum of ten seconds to provide the customized report to the customer.

# 6. Non-functional Requirements

## 6.1 Product NFR

* The system shall not experience a downtime during the following peak hours:
  + 8AM-10AM, 12PM-1PM, 4PM-5PM.
* The system shall be reliable. The MTBF (Mean Time between Failures) shall be a maximum of 10 minutes.
* The system shall be scalable to accommodate a maximum of 100 parking lots.
* The system shall not occupy more than 5GB of space on the server for the application and the database combined.

## 6.2 Process NFR

* The system shall be programmed using Java and Python.
* The system shall be designed using Harmony process.
* The operating system for the programmable hardware shall be Windows.
* The operating system for the mobile device shall be Android.

## 6.3 External NFR

* The system shall be able to connect to external systems such as UTD Parking department database.

# Appendix A: Glossary

|  |  |
| --- | --- |
| **Term** | **Definition** |
| IR Sensor | Infrared Sensor |
| Adafruit Trinket | Micro-controller which is connected to IR sensor in bread board |
| ATMEL ATtiny 85 chip | Part of Raspberry PI controller CPU |
| GPIO | General-Purpose input/output |
| DTR | Data Terminal Ready |
| RTS | Request To Send |
| DSR | Data Set Ready |
| SD | Secure Digital |
| SPI | Serial Peripheral Interface |
| ISM | The industrial, scientific and medical radio |
| Mbps | Mega bits per second |

# Appendix B: References

1. Rym Z. Wenkstern, SRS Template, P 1-9