Quest

Team 11

IoT Challenge Taller Vertical 2016

Abstract

"X Marks The Spot

Software Requirement Specification

1 Introduction

1.1 Purpose

As cities grow larger, the need for parking space diminishes with time. Families that used to own one vehicle have added, in average, one more of them. Every new vehicle takes up a parking space, and having thousands and even millions of cars provoke a parking problem in all major cities. People have nowhere to park their vehicles making it very stressful for the drivers. Quest is a program that will focus on facilitating parking by knowing the zone in which the driver will park, reducing the time spent looking for a spot. Using minimal resources, Quest will be an affordable yet efficient program. With range traces, it will identify which zones have space and which do not. Instead of spending precious minutes finding a space, users will know beforehand where it is convenient to park. This application is focused on parking lots of major cities in which population is incredibly dense and people live in continuous stress.

1.2 Scope

Quest is a new application that is responsible for locating the places available in the best area to park your car in the parking lot. The main purpose of our app is that the user time to find a place available and a good area to park your car. The objective of implementing such program is to facilitate parking and make life much easier for everyone that owns a vehicle and needs to find a place to park.

1.3 References

- Doukas, C. (2012). Building internet of things with the Arduino. North Charleston, SC: CreateSpace.
- Müller, B., Reinhardt, J., & Strickland, M. T. (1995). Neural networks: An introduction. Berlin: Springer.
- Wilamowski, B. M., & Irwin, J. D. (2011). Intelligent systems. Boca Raton (Fla.): CRC Press.

2 Overall Description

2.1 Product Perspective

Quest is a program that will implement a base intelligent system that will be able to determine the best possible solution in order to find an adequate parking space.

2.2 Product Functions

With Quest's algorithm, we will be able to provide an efficient way to find an available parking space for our users. Dividing parking lots by zones and matching those zones with what the user is searching for will make it efficient and simple to find a place to park your vehicle.

2.3 User Characteristics

Quest is designed to be as user friendly as possible. With a quick and intelligent design our clients will be able to find a place to park in a matter of seconds and save precious time in things that actually matter.

2.4 Constraints

Quest will require an Android system with the following hardware specifications:

- Dual-Core 1Ghz or faster processor.
- 512 GB RAM
- Android OS 4.1 jelly bean or superior.

2.5 Assumptions and Dependencies

The sensors (button, touch, LCD and beacons) are the pillars of our application, if there exist any kind of interference, the information stored in the database will be erroneous. Also if the database service collapses in some moment, the application will not be able to deploy the zones in the map.

3 Specific Requirements

3.1 User Interfaces

- A main menu to start the application with a login or sign up if it is the users first time using the application.
- A user-friendly interface that will show a map of the desired parking lot which the user is currently entering. It will display the zones and mark with colors which have available spaces in them.

3.2 Hardware Interfaces

In order for Quest to work properly we use an Intel Edison, plus some Grove Buttons, LEDs and LCDs displays from the Grove Starter Kit Plus.

3.3 Software Interfaces

To develop software for Quest it was used:

- Android Studio: Functionality and design of the app.
- Putty: Console to configure the Edison Board.
- Intel XDK: Driver for the Edison Board.
- JSON: Communication from the board to the other software interfaces.
- Python: Language used to program the Edison Board and all its functionality.
- Firebase: Database used to store information (for more detail please go to Section 3.8).

3.4 Communication Interfaces

WLAN: Used to communicate the hardware inwardly.

4 System Features

4.1 System Feature #1: Sign Up

4.1.1 Description and Priority

The app must correctly save the user's email and password in the database. By doing so, the user will then be able to log in whenever he/she likes. If the user tries to sign up with an invalid email, a popup will appear indicating to put a valid email. This is a high priority feature because if there is a problem while saving the data, the user won't be able to enter the app.

4.1.2 Stimulus / Response Sequences

- Read the user's input.
- Save it in the correct place of the database.

4.1.3 Functional Requirements

- REQ-1: The app must be connected to the database.
- REQ-2: The app has to be able to read the input.

4.2 System Feature #2: Log In

4.2.1 Description and Priority

In this feature the user must have a registered account in the app (Sign up), and now can write down his/her email and password in order to access to it. If the email is not saved in the database or the password is wrong a popup will appear. Also if the password is forgotten, the user can click on the "Forgot password?" button to receive an email with a temporary password. This feature has high priority because if one cannot access, he/she would not be able to use the map that is the reason to implement a safe mode of recovering the password.

4.2.2 Stimulus / Response Sequences

- Read user's input.
- Search for it on the database.
- Send a recovery email if it is necessary.

4.2.3 Functional Requirements

- REQ-1: The app must be connected to the database
- REQ-2: The app has to be able to read the input.
- REQ-3: The app must be able to send the email with a correct temporary password.

4.3 System Feature #3: Capturing Data

4.3.1 Description and Priority

In this feature the sensors and the Edison board are the main actors. Using the grove button the board is going to detect if a car enter to that specific parking zone, and with the touch sensor if the vehicle leaves that area. This feature has high priority because if the data captured is not the real one, all the system is going to store and deploy false information to the user.

4.3.2 Stimulus / Response Sequences

- Read the information captured by the sensors.
- Store the data on the Firebase database.

4.3.3 Functional Requirements

- REQ-1: The Edison board must be connencted to the internet and to the Firebase.
- REQ-2: The sensors cannot fail or stop working at any time.

4.4 System Feature #4: Look Up the Zone

4.4.1 Description and Priority

The app will access the database to read the information that was put there by the Edison about the different zones, so that it can make a decision of where to tell the user to park. Then displaying, with different colors depending on how full the zone is, that parking zone. This has a high priority because it will calculate where to park, the core of the whole project.

4.4.2 Stimulus / Response Sequences

- Read the database.
- Calculate with the algorithm the best parking zone.
- Displaying the zone.

4.4.3 Functional Requirements

- REQ-1: Being connected to the database.
- REQ-2: Having an accurate algorithm.

5 Non-Functional Requirements

5.1 Performance

Will rapidly give the best zone to park, because if it takes to much time to accurate give an answer, Quest will be pointless, no time will be saved.

5.2 Reliability

Only the user can access and see the information used to create the profile and the information of the place where he/she is going to park.

5.3 Availability

The system is going to be functional in some parking lots only, and it will be necessary to have some downtime activities to upgrade the database and to add new parking space.

5.4 Security

No one will be able to see a password from another user, or access the database without previous permission.

5.5 Maintainability

Quest will easily be changed in order to meet the different kind of parking lots in which the app is going to be used. And if there is the case that in the parking lot more available spaces are created, it will also change in the app.

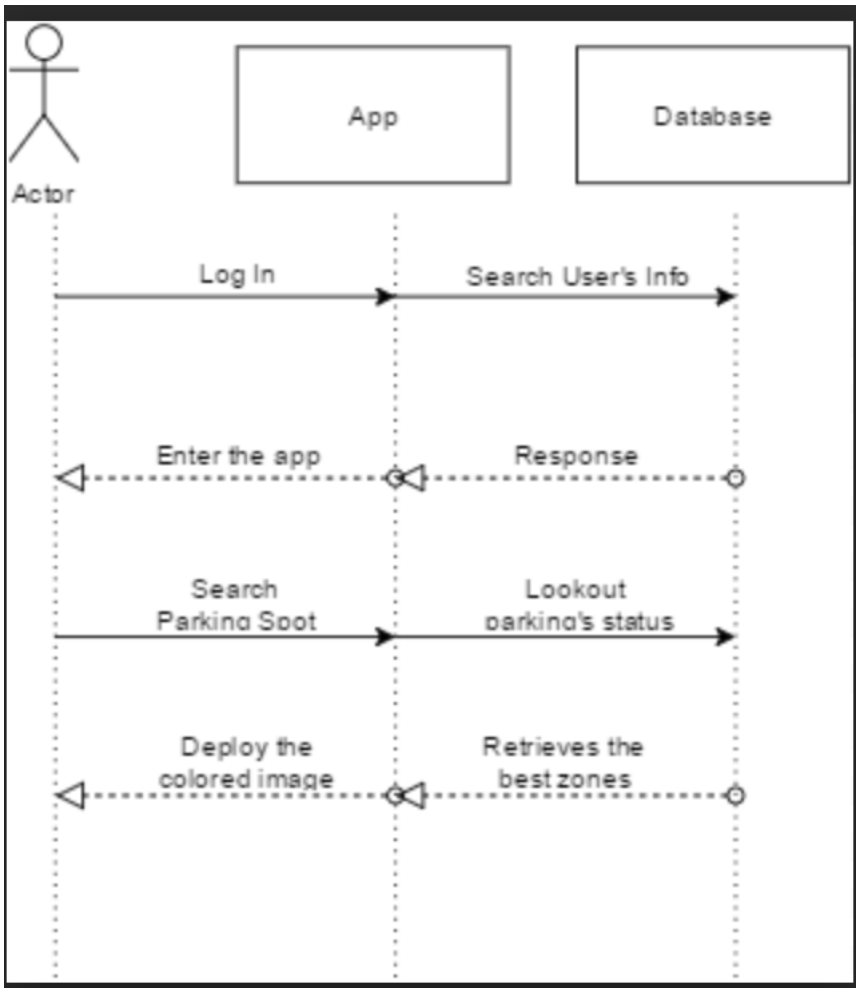
5.6 Portability

The app is developed for smartphones with Android OS, so it will not be portable for other devices with other operative systems like IOS or Windows Phone.

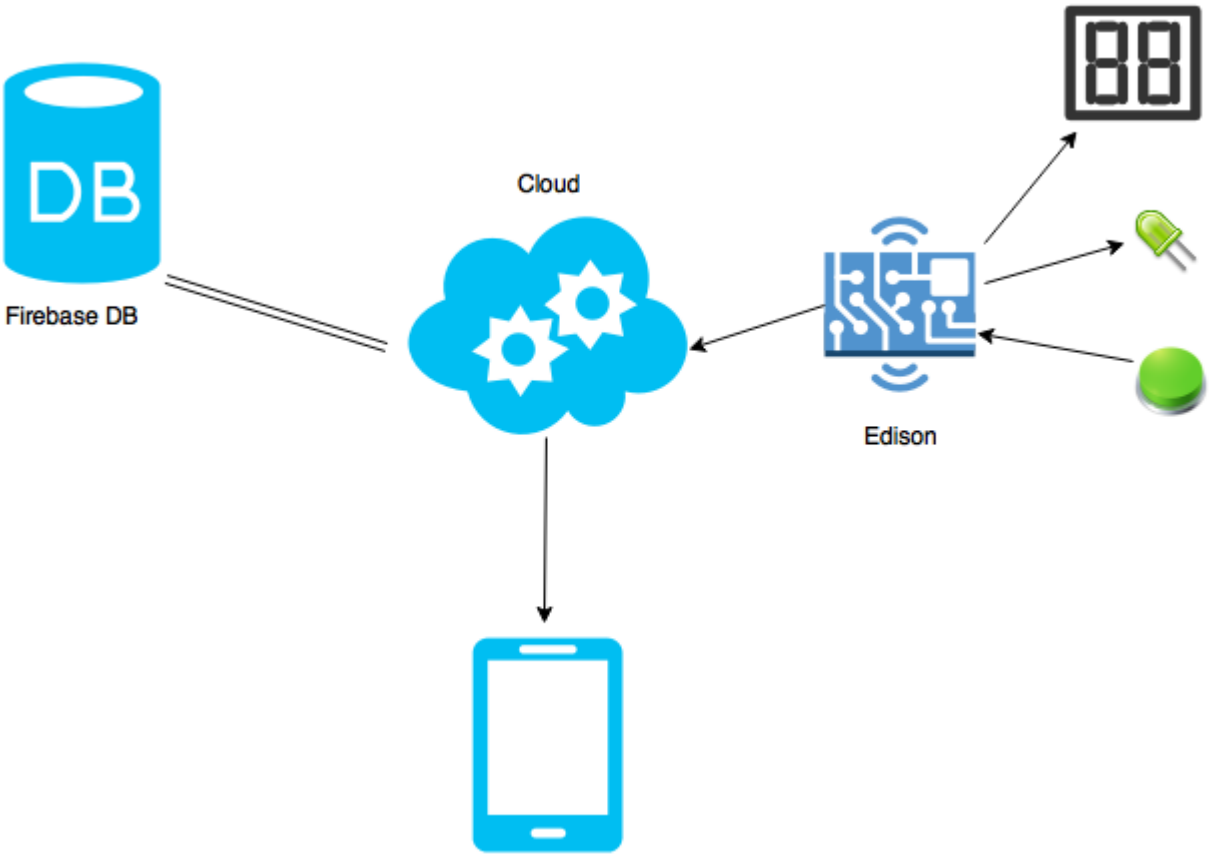
6 Logical database requirements

The system must store all the user account information as well as the data collected by the sensors in the Firebase Database. For each user account the login ID, name, password, email address shall be stored in one file. The email address gives the user option to receive further information about updates and new parking slots added.

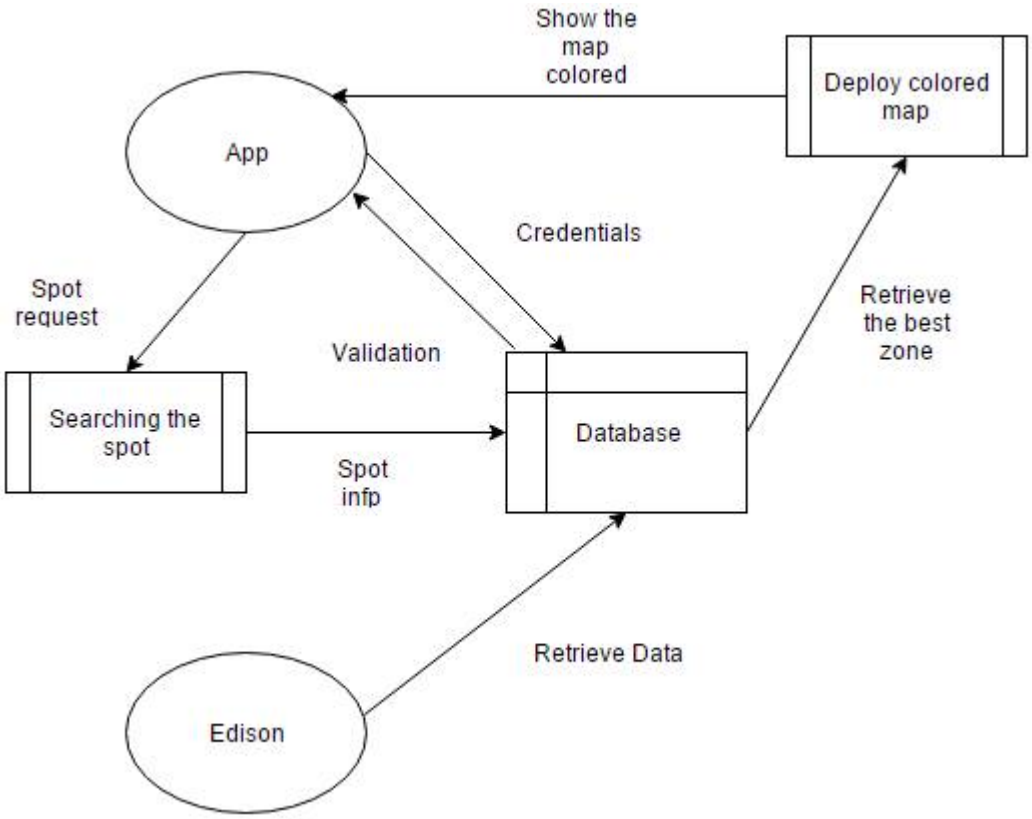
System Sequence Diagram



Architecture Diagram



Data Flow Diagram



Activity Model