RU Parking

ANDROID-BASED APPLICATION FOR SUGGESTING PARKING SPACES

http://code.google.com/p/ruparking/

REQUIREMENTS ANALYSIS

Project Members:

Cyrus Gerami
Ashwin Revo
Shweta Sagari
Samson Sequeira
Vrajesh Vyas

Contents

1. Introduction

1.1 Purpose of the System

The number or vehicles on our roads is increasing day by day. However, the city infrastructure is failing to maintain the same growth rate. In large metropolitan cities this has led to a disastrous parking situation. There is a scarcity of street parking spots and an abundance of unhappy commuters. There are navigation systems which direct users to a particular destination and even indicate traffic congestion. But there are no reliable and convenient applications which assist users in finding vacant parking spots.

Our application is intended to fill this void. It is designed to be run on mobile phones making it very accessible. This application will assist the user in finding vacant parking spaces by employing real-time assessments of the parking scenario based upon statistical analysis of current and historical data.

1.2 Scope of the System

- All users should be able to access the RUparking application on an Android OS based mobile phone that has GPS and Internet connectivity.
- The application should be able to access the Remote Server to obtain real-time parking decisions.
- This application will also be using the location framework and Google maps external library.

1.3 Objectives and success criteria

- Provide a GUI for the mobile user enabling him to find convenient vacant parking spaces.
- Equip the GUI with a map of colored streets, the color indicating the likelihood of having vacant parking spaces.
- Provide the Garage owner with data about parking availability to enable him to set pricing for accordingly.

1.4 Definitions, acronyms and abbreviations

Android OS: Android is a mobile operating system running on the Linux kernel. It allows developers to write managed code in the Java language, controlling the device via Google-developed Java libraries.

Mobile User: End user, which runs this application on a mobile phone, with personal interest.

Garage Owner: the owner of any parking garage, which uses this application for evaluation and analysis of block by block parking availability.

Remote Server: the server/Database, which provides a decision/suggestion based on raw data.

Destination Address: the address where the user would like to find out the parking availability situation. (most probably the trip destination)

Parking Space: any segment of a street offering more than three side-parking spots.

GPS: Global Positioning System, used for gathering user current coordinates

SDK: Software Development Kit, a set of development tools that allows us to create applications for a certain software package.

GUI: Graphical User Interface, a type of user interface, which allows people to interact with electronic devices such as mobile phones. A GUI offers graphical icons, and visual indicators, typed command labels or text navigation to fully represent the information and actions available to a user.

1.5 References:

1. http://developer.android.com/guide/index.html
2. http://code.google.com/apis/maps/documentation/index.html
3. http://www.parc.com/event/935/sensor-networks-in-the-big-city.html
4. http://primospot.com/

2. Current System

Currently the park space applications in the market base their decisions on static information about the parking spaces. These applications maintain a database of the legal spaces and their restrictions like parking hours, duration of parking etc. Such a system is inefficient since it is not able to suggest parking spaces based on its current state. There are applications that claim to identify vacant parking spaces in real-time by connecting sensors to the parking meters. Such a system requires large-scale changes in the existing infrastructure, which is not financially feasible. Also it is not applicable for street-side parking that provides for majority of the parking spaces in metropolitan cities. Also these parking spaces are usually metered in which case they are occupied by one vehicle for a short time like two hours. So there is a need to identify the vacancies in these parking spaces in real-time. Since these spaces do not have sensors this has not been achieved so far. Our Application tries to accomplish this using the real-time data provided by the Roadside Parking Monitoring team.

3. Proposed System

3.1 Overview

Our project aims to provide a user-friendly Graphical User Interface (GUI) that can be used on smart phones that run on the Android OS. The RU Parking Application will provide the user real-time decisions on where vacant parking spaces are present. It interacts with a Remote Server to obtain the parking information. Once this information is received by the application it will display this information in an easy-to-understand way to the user. It provides the user with 3 different levels of map detail to enable him to make different types of decisions regarding parking and commuting. This application will also provide the option to the user to be directed to a particular parking space by navigating him there turn-by-turn.

In addition to the Mobile User, this application also caters to the Garage Owners. These users are provided with the list that captures the parking situation around their garage. The Garage Owners can choose the radius of the area around them, about which they want to obtain information.

3.2 Functional Requirements:

RU parking is a mobile application that provides real time information about park space availability to the user. The application supports two types of users:

1. The mobile user should be able to access parking information for any location. This information is provided in form of maps showing colored streets. The mobile user should be able to choose between three views namely area view, block view and street view. The mobile user will also be able to select one particular parking space and obtain the routing information for that selection.

2. The garage owner should be able to access list of parking spaces in a given radius along with timing information. The user should be able to change the radius under consideration.

3.3 Nonfunctional Requirements:

- **Usability:** The user must be able to utilize this application without any prior knowledge about the user interface. The interface must allow the user to work with the different features of application with a maximum of four clicks. In case of any issues, the user must be able to find the solution by referring to the user manual.

- **Reliability:** This application depends on the Internet and GPS connectivity. In case of loss of Internet connectivity this application will use the last information fetched from the remote server to display parking information. In case of GPS failure the user can input the address manually, in the destination address text box. In the event of application crashing it should be able to restart automatically displaying last confirmed output to the user.

- **Performance:** The application will support one instance per phone. After the user inputs the destination address, the output should be given within 30 seconds.

- **Supportability:** The application should be compatible with any future changes in the remote server provided the data format remains the same. The user should be able to upgrade the maps on the phone without re-installing the software. This procedure will require restarting of application.

- **Implementation:** All users should be able to access RU parking on their android based mobile phones supporting GPS and internet. Additionally users having an Internet connection should also be able to access RU parking through a web interface.

- **Interface**: The application interfaces with a remote server to access real time parking space information. The data in the remote server is stored in a tabular form having the latitude, longitude, the current state of parking space and the time of reading as attributes. This information is exchanged over the Internet.

- **Operation:** The MobileUser and GarageOwner should be able to get valid parking information provided they have access to the Internet.

- **Legal:** The developers are not responsible for incorrect parking information since this information is sourced from third party vendors. The connection between the application and the server is encrypted to safeguard user privacy. No user information is stored on the remote servers and information regarding the user location is not provided to any third party vendors.

3.4 System models:

3.4.1 Scenarios:

Scenario name	currentLocationParking
Participating actor instances	bob: MobileUser
Flow of event	1. Bob is driving towards his office to attend an important meeting, when he reaches his destination he is unable to find a parking space. 2. He activates RU parking application on his phone. 3. He selects "current location" option in the application. 4. This request is sent to the server that responds with an updated map showing colored streets with available parking spaces. 5. Bob refers to this map and can observe the nearest parking spaces.

Figure 3-1 The currentLocationParking scenario for the ViewMap use case.

Scenario name	destinationParking
Participating actor instances	bob: MobileUser
Flow of event	1. Bob is about to leave for a football match. There are 3 streets through which he can reach the stadium. He wants to know beforehand on which street there is a higher probability of finding a parking space. 2. He activates RU parking application on his phone. 3. He enters "destination address" in the application. 4. This request is sent to the server which responds with an updated map showing colored streets with available parking places. 5. Bob refers this map and can observe the street which has higher probability of having parking places.

Figure 3-2 The destinationParking scenario for the ViewMap use case.

Scenario name	routeProvider
Participating actor instances	bob:MobileUser
Flow of event	1. Bob is visiting his friend after a long time and does not know the way; moreover his friend informed him that there are no parking spaces around his apartment. 2. He activates RU parking application on his phone. 3. He enters "destination address" in the application. 4. This request is sent to the server which responds with an updated map showing colored streets with available parking places near that area. 5. Bob refers this map, selects a particular street and also clicks on the route option, which guides Bob to a parking space near his friend's place.

Figure 3-3 The routeProvider scenario for the SelectNavigation use case.

Scenario name	parkingRates
Participating actor instances	tom:GarageOwner
Flow of event	1. Tom is an intelligent garage owner who changes the parking rates as per the parking space availability. But he is unaware of the parking space availability in neighboring streets. 2. He activates RU parking application on his phone. 3. He enters "garage address" in the application. 4. This request is sent to the server, which responds with an updated list of streets with parking space availability and the timings associated with that information. 5. Tom refers to this map and decides on the parking rates.

Figure 3-4 The parkingRates scenario for the ViewStatistics use case.

3.4.2 Use Cases

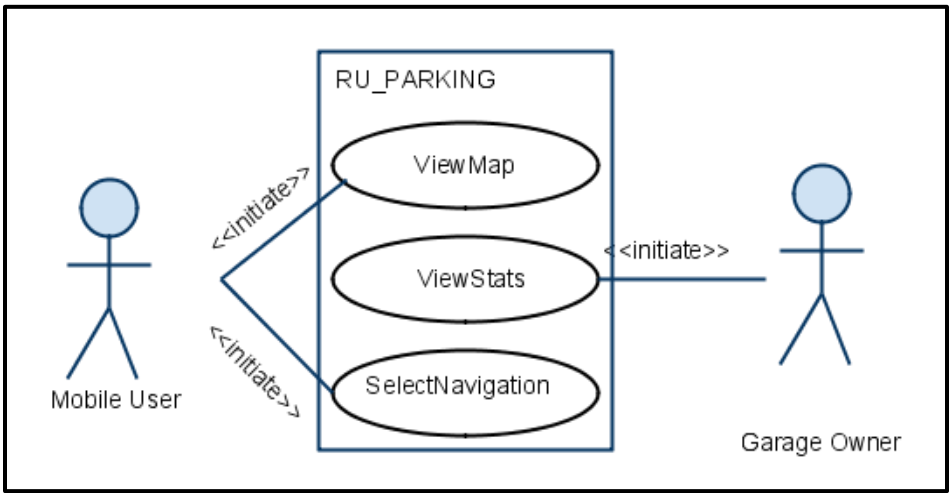

Figure 3-5 High-level use cases for RU Parking

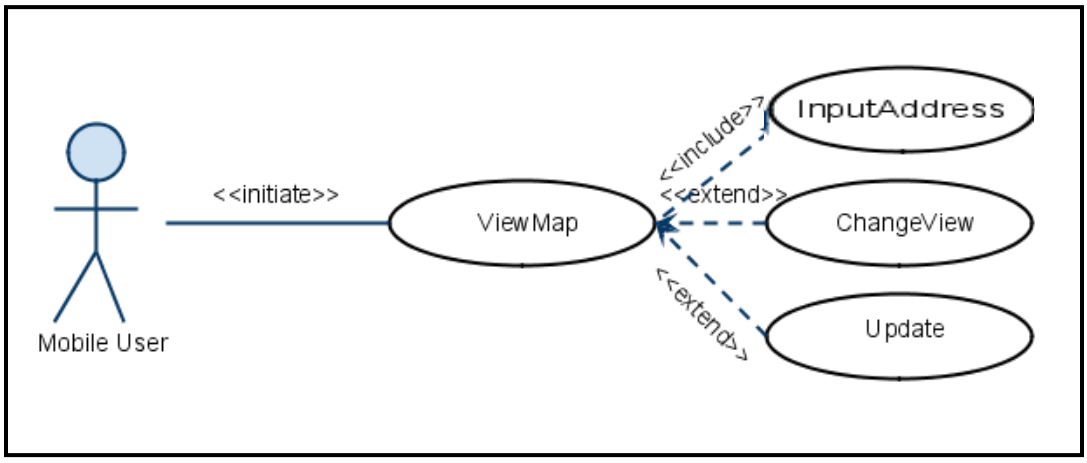

Figure 3-6 Detailed use cases refining the ViewMap high-level use case.

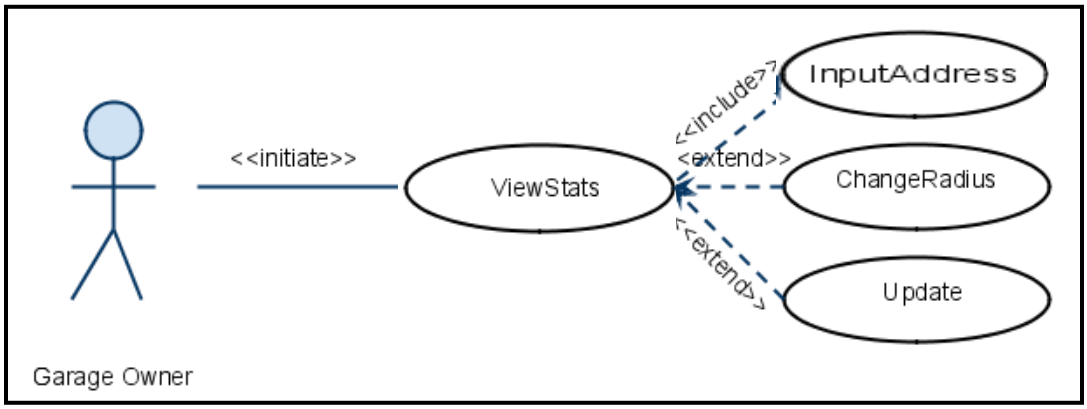

Figure 3-7 Detailed use cases refining the ViewStatistics high-level use case.

Use case Name	ViewMap
Participating Actors	Initiated by MobileUser
Flow of Events	1. MobileUser presses the “OpenApplication” button which creates an “InputAddress” form. 2. In this form the user can type in the destination address or select the current location as his input. Once the form is completed the MobileUser submits the form by pressing the “Submit” button, at which point the Server is notified. 3. The Server parses the information submitted by the MobileUser and submits parking availability data to the Application. 4. The Application uses this data to create a map which is then displayed on the LCD.
Entry Condition	MobileUser activates the “ViewMap” function of his phone.
Exit Condition	• The MobileUser presses the “ExitApplication” button. • The MobileUser presses the “ChangeView” button. • The MobileUser presses the “SelectStreet” button. • The MobileUser presses the “Update” button.
Quality Requirements	• Parking suggestion should be in real-time and accurate. • The colors in the map should be easily identifiable.

Figure 3-8 The ViewMap use case description.

Use case Name	ViewStats
Participating Actors	Initiated by GarageOwner
Flow of Events	1. GarageOwner presses the “OpenApplication” button which creates an “InputAddress” form. 2. In this form the user can type in the destination address or select the current location as his input. Once the form is completed the GarageOwner submits the form by pressing the “Submit” button, at which point the Server is notified. 3. The Server parses the information submitted by the GarageOwner and submits parking availability data to the Application. 4. The Application uses this data to create a list which is then displayed on the LCD.
Entry Condition	GarageOwner activates the “ViewStats” function of his phone.
Exit Condition	• The GarageOwner presses the “ExitApplication”

	button. • The GarageOwner presses the “ChangeRadius” button. • The GarageOwner presses the “Update” button.
Quality Requirements	1. Parking suggestion should be in real-time and accurate.

Figure 3-8 The ViewStatistics use case description.

Use case Name	SelectNavigation
Participating Actors	Initiated by Mobile User
Flow of Events	1. The MobileUser presses the “SelectStreet” button. 2. Now the MobileUser is asked to confirm this by pressing a “Navigate” button. 3. A navigate request is sent to the Application. 4. The Application calls the “AddRouteToMap” function which generates the routing information. 5. This information is then sent to the LCD to be displayed.
Entry Condition	The MobileUser uses the “ViewMap” function.
Exit Condition	• The MobileUser presses the “ExitApplication” button. • The MobileUser pressing the “Cancel” button when asked for confirmation.
Quality Requirements	• The Application should suggest the best route on the basis of shorter distance or lesser time. • The directions should be timed correctly so that the user is informed on time.

Figure 3-9 The SelectNavigation use case description.

Use case Name	InputAddress
Participating Actors	Initiated by MobileUser or GarageOwner
Flow of Events	1. MobileUser or GarageOwner presses the “OpenApplication” button which creates an “InputAddress” form. 2. In this form the user can type in the destination address or select the current location as his input. Once the form is completed the MobileUser or GarageOwner submits the form by pressing the “Submit” button, at which point the Server is notified.
Entry Condition	The MobileUser or the GarageOwner activates the Application.

Exit Condition	• The MobileUser or GarageOwner clicks on the "ExitApplication" button. • The MobileUser or GarageOwner clicks on the "Submit" button.
Quality Requirements	The form should be able to differentiate between street names, state names and ZIP codes.

Figure 3-10 The InputAddress use case description.

Use case Name	ChangeView
Participating Actors	Initiated by MobileUser
Flow of Events	1. The MobileUser presses the"ChangeView" button while viewing the map on the LCD. 2. This sends a "ChangeView" request to the Application. 3. The Application then creates a new map having a different detail level. 4. This information is then sent to be displayed on the LCD.
Entry Condition	The MobileUser uses the "ViewMap" function.
Exit Condition	The Application displays the new map on the LCD.
Quality Requirements	The "ChangeView" function should have 3 distinct levels of detail to provide varying amounts of information to the user.

Figure 3-11 The ChangeView use case description.

Use case Name	ChangeRadius
Participating Actors	Initiated by GarageOwner
Flow of Events	1. The GarageOwner presses the"ChangeRadius" button while viewing the list on the LCD. 2. This sends a "ChangeRadius" request to the Application. 3. The Application then creates a new list having a different amount of information. 4. This information is then sent to be displayed on the LCD.
Entry Condition	The GarageOwner uses the "ViewStats" function.
Exit Condition	The Application displays the new list on the LCD.
Quality Requirements	The "ChangeRadius" function should have 3 distinct details of information to provide to the user.

Figure 3-12 The ChangeRadius use case description.

Use case Name	Update
Participating Actors	Initiated by Mobile User and Garage Owner
Flow of Events	1. The MobileUser or GarageOwner presses the "Update" button while viewing the map or list. 2. The request for update is sent to the Application. 3. The Application then submits the address to the Server. 4. The Server parses this information and sends relevant parking availability information to the Application. 5. The Application then creates a new map or list which is displayed on the LCD.
Entry Condition	The MobileUser uses the "ViewMap" function.
Exit Condition	The Application displays the new map or list on the LCD.
Quality Requirements	While updating the previous destination address of the user should be used.

Figure 3-13 The Update use case description.

3.4.3 User Interface—navigational paths and screen mock-ups:

Figure 3-14 The Screen mock-ups for RU Parking.

The RU Parking Application users are provided with a user-friendly and convenient Graphical User Interface. It is important that the GUI is easy to use and gives only relevant details so as to not distract the user or take more time to understand.

The first screen of the GUI gives the user the option of choosing where he wants to park; at his current location or some other address.Once the application has figured out the user's required parking location, the GUI shows the map of that location and relevant parking information. The availability of parking is illustrated using colors which makes it very readable. The GUI also provides options for changing the level of detail in the map. It also offers visual directions when the user opts for the Navigation mode. The Garage Owner will be provided with a list of statistics and the option to choose the area about which he needs information.