# **ParkMyCar**

A Report

Submitted to

The School of Engineering and Computing

National University

In Partial Fulfillment of the Requirements

For the Degree of Master of Science in Computer Science

By

VijayaLakshmi Ammineni, Bhavya Kolakaluru, Sirisha Gummadi, Radha Kiranmai Gullapalli

March 2015

# **ParkMyCar**

A report submitted to the School of Engineering and Computing, National University, in partial fulfillment of the requirements for the Degree of Master of Science in Computer Science by:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

VijayaLakshmi Ammineni

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Bhavya Kolakaluru

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Sirisha Gummadi

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Radha Kiranmai Gullapalli

The report is hereby approved by:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Pradip Peter Dey, Ph.D.

Professor, School of Engineering and Computing

National University

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Bhaskar Raj Sinha, Ph.D.

Professor, School of Engineering and Computing

National University

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date

**Abstract**

The goal of the project is to develop a mobile application to make parking easy in downtowns especially paid parking. The app would also provide other features like timing the park and providing directions back to the parking spot. Crowdsourcing techniques will be used to improve the quality of the search results. To test this application we will look for 10 to 20 empty parking lots in San Diego downtown and then test the app by driving around San Diego downtown, search for parking lots, park the car, provide feedback and rating. This would also help us test the crowdsourcing feature. This information can also be used to provide real time approximate number of people who parked their car at different parking lots in last one hour.

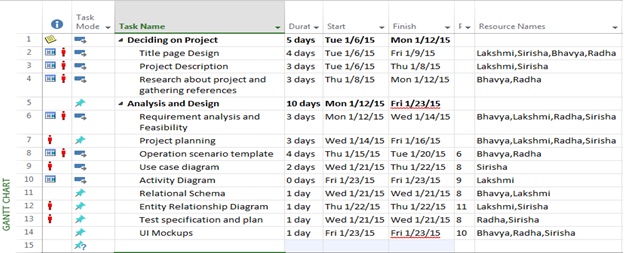
**Main Features**

1. Show nearest parking lots when provided with an address
2. This feature aims to show public/private/paid parking lots near a specified address by filtering results based on crowdsourcing feedback. It will also provide statistics like how many app users parked their vehicle in a parking lot in last one hour and this will be shown on google maps
3. It also includes navigating the user to the selected parking lot using google directions API
4. Provide directions back to your parked location
5. Store the parked location of the vehicle when parking is done.
6. Use google maps direction API to provide directions back to the parking lot
7. Parking time reminder
8. Start timer when parked, which always shows how long the car has been parked, optionally user can opt for reminders before the parking end time.
9. Admin account to add/remove/edit parking locations and change prices

**Similar mobile applications available in market and their features:**

1. **Parkdroid [2] & Car Locator [3] apps**: These apps provide features to navigate back to the parked car and also the parking timer but lack the feature to search for nearby parking lots.
2. **Park Me Right: Free Car Locator [4] app:** This app provides search for parking and uses Yelp ratings which are not real time which we would like to add to our application.

**Project Plan:**

****

****

**Usecase Diagram:**

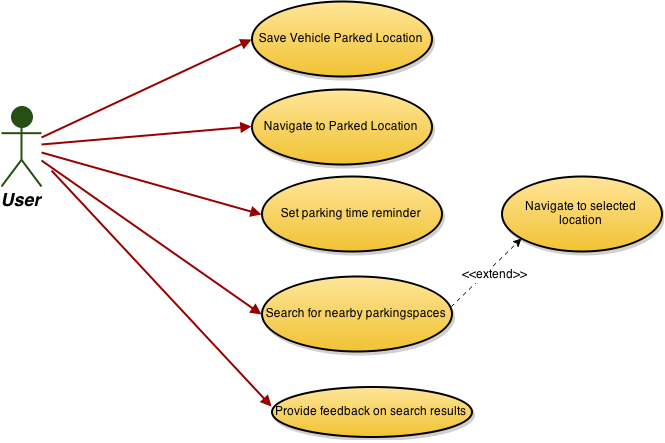


Figure 1: Use case Diagram

**Administrator Use case:**

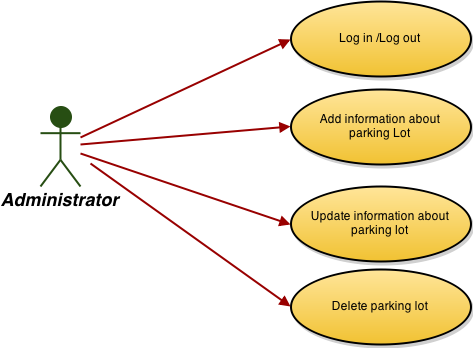


Figure 2: Admin Use case Diagram

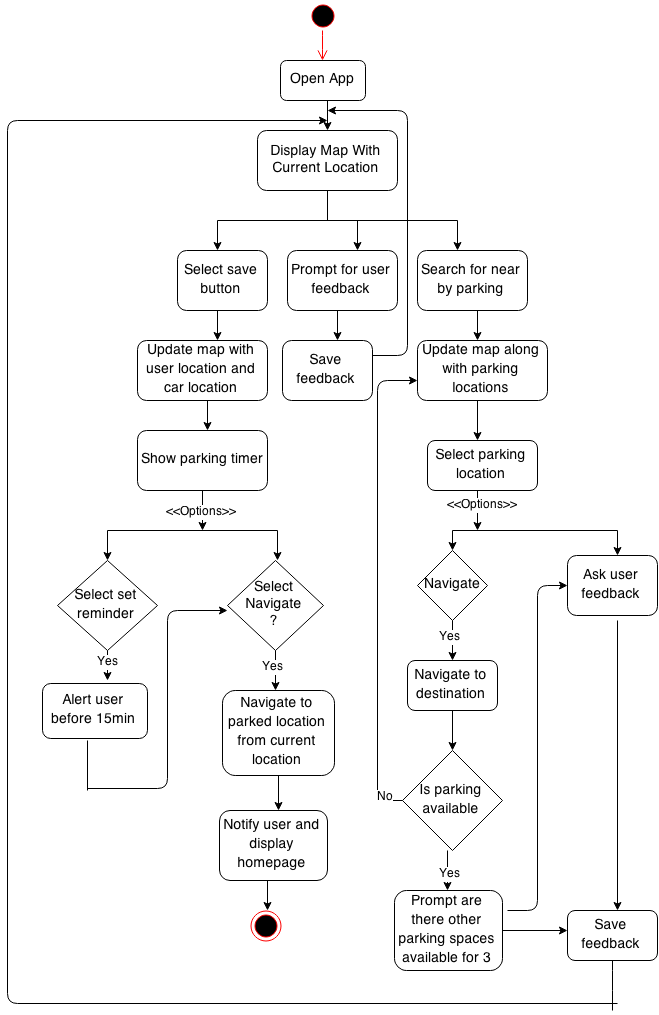
**ActivityDiagram:**

Figure 3: Activity Diagram

**Relational Schema:**

Relational schema is a set of formulas (sentences) called integrity constraints imposed on a database. These integrity constraints ensure compatibility between parts of the schema. All constraints are expressible in the same language. A database can be considered a structure in realization of the database language.[ The states of a created conceptual schema are transformed into an explicit mapping, the database schema. This describes how real world entities are modeled in the database.[5]

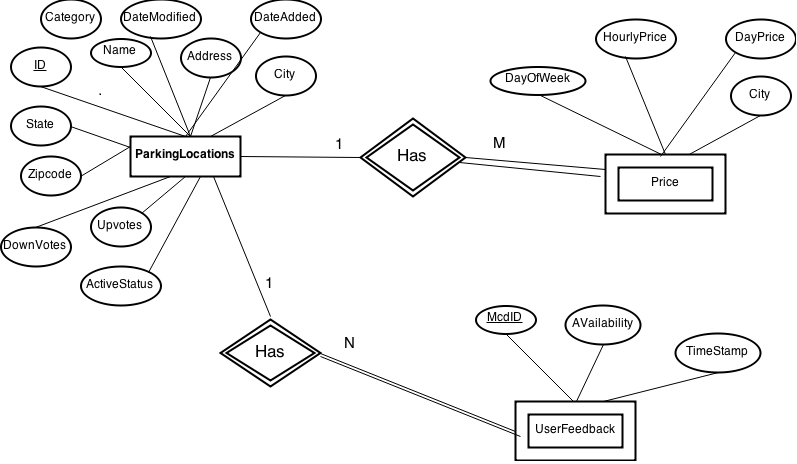
****

Figure : Relational Schema

**Entity Relationship Diagram:**

Database design can be explained clearly with the help of ER Diagram

ER Diagram that is Entity-Relation diagram represented in the form of classes and their dependencies with the other classes. A class is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. In the class diagram these classes are represented with boxes which contain three parts:

* The upper part holds the name of the class
* The middle part contains the attributes of the class
* The bottom part gives the methods or operations the class can take or undertake

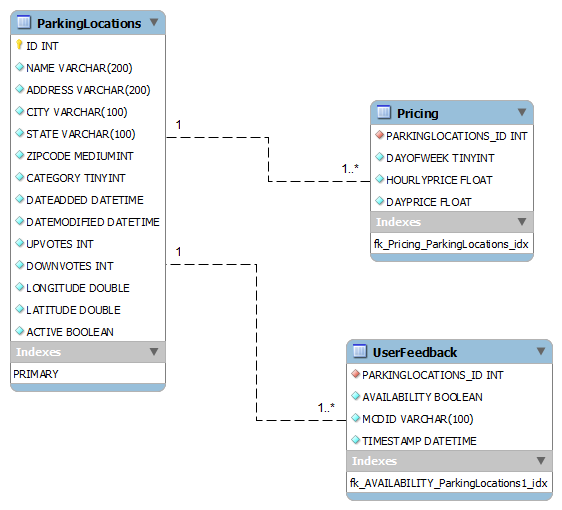


Figure : Entity Relationship Diagram

**Requirement Analysis and Feasibility**

ParkMyCar application helps locating the parking lots in downtowns when searched with an address and make parking easier to the user. Once developed it will work based on the users feedback coupled with inputs from ParkMyCar administrator. The mobile application requires that the user turns on location and internet services. The parking lots shown in the application can be public/paid and the lot results are filtered based on crowdsourcing feedback. The app will prompt for user feedback regarding the parking lot availability and the feedback will be saved. The application also includes an option for user to update the car location which can help in navigating the user once he has to return back to the car. This will involve using a google maps direction API to provide directions back to the parking lot.

This application has a client server architecture. The Java web application acts as backend. The client is an Android application. The technologies used are Java, spring framework, MySQL database and Apache tomcat container, Google Maps geocoding/reverse geocoding APIs, Android Development Toolkit (ADT), Google Maps (Android) API, Git Hub (as Source repository) and AWS (for deployment).

**Operational Scenario Template:**

Construct operational scenarios to cover the normal and abnormal program uses:

Table 1: User Scenario1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Scenario # | | | User Objective: Select menu and select any one of options. | |
| Scenario Objective: To identify program response to user actions | | | | |
| Source: User | Step | Action: | | Comments |
| user |  | Open Mobile Application | |  |
| program |  | Display map with current location | | Display menu |
| user |  | Select the user choice among save functionality, user feedback, search for nearby parking | | User can able to select his/her desired option |

Table 2: User Scenario2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Scenario # | | | User Objective: User select save functionality to save current parking spot from the menu options. | |
| Scenario Objective: To identify program response to user actions | | | | |
| Source: User | Step | Action: | | Comments |
| user |  | user select save functionality | | User can update |
| user |  | User update map with user location and car location | |  |
| program |  | Displays parking timer | | It display option |
| user |  | Select options among set reminder and navigate | |  |
| program |  | Show parking timer | | It gives options. |
| user |  | Select options among set reminder and navigate | |  |
| user |  | If user select set reminder | | Alert user before 15 min |
| program |  | It displays navigate option | |  |
| user |  | User navigate to park location from current location | |  |
| program |  | Notify user and display home page | |  |

Table 3: User Scenario3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Scenario # | | | User Objective: Select user feedback functionality from the menu. | |
| Scenario Objective: To identify program response to user actions | | | | |
| Source: User | Step | Action: | | Comments |
| user |  | user selects user feedback from the menu | |  |
| program |  | It shows feedback given by other users or it allows to enter the feedback | | User give their own feedback |
| user |  | Saves the feedback | | Other users can see that feedback |
| Program |  | After saving it display homepage | | Exit from that menu |

Table 4: User Scenario4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Scenario # | | | User Objective: User select search for nearby parking functionality from the menu. | |
| Scenario Objective: To identify program response to user actions | | | | |
| Source: User | Step | Action: | | Comments |
| user |  | Select search for nearby parking | |  |
| program |  | Update map along with parking location | | Displays available parking lots |
| user |  | Select parking location | |  |
| Program |  | It displays navigate option and user feedback option | | User selects one option |
| user | 5. | If user selects navigate option then it navigates to destination or it ask for feedback and saves feedback | |  |
| program | 6. | It checks for available parking | |  |
| user | 7. | If parking is available user prompts for other parking space available | | Sends feedback if parking space available or ask for feedback |
| program | 8. | If parking space not available then it update map along with parking location | | It checks and display the result |
| program | 9. | After finding the location it exits from the menu | | Display home page |

Table 5: User Scenario5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Scenario # | | | User Objective: User actions | |
| Scenario Objective: To identify user actions in the application. | | | | |
| Source: User | Step | Action: | | Comments |
| user |  | Open mobile application | |  |
| program |  | Displays menu | |  |
| user |  | User select save vehicle parked location. | |  |
| program |  | Display navigate option | |  |
| user |  | Navigate to parked location | |  |
| Program |  | It shows parking timer | | It allows to set reminder. |
| user |  | Set parking time reminder. | | Alert user before 15 min |
| user |  | Search for nearby parking locations. | | It displays nearby parking lots. |
| program |  | Displays nearby parking locations | | Select one parking location. |
| user |  | Navigate to selected location | |  |
| user |  | Provide feedback on search results | | He can provide feedback. |
| Program |  | It saves feedback and display homepage | |  |

Table 6: Administrator Scenario6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Scenario # | | | User Objective: Administrator actions. | |
| Scenario Objective: To identify administrator actions in the application. | | | | |
| Source: User | Step | Action: | | Comments |
| administrator |  | Login in to the application | |  |
| Program |  | Opens the application | |  |
| administrator |  | Add information about parking lot | | Update the information |
| administrator |  | Update the information about parking lot | |  |
| program |  | Saves the updated information about parking lot. | |  |
| administrator |  | Delete parking lot which are not required or already filled. | | Remove the parking lots. |
| administrator |  | Logout from the application. | |  |

**Test Specification/Test Plan:**

1. **Software to be tested:**

The software applications being tested are android and web applications.

1. **Testing strategy**

1. **Unit testing**

Unit testing will be done on the lowest level. As an individual module is being written, testing will be conducted for each and every piece of functional code by compiling, reviewing and fixing compiler errors. When the coding stage of the module is complete, the module will then be tested according to its purpose. In this stage, the module will be tested with all reasonable and expected inputs to see if it does its prescribed task. Finally, the module will be tested for security issues. [8]

The following are the main modules and features that are tested in the application prototype

1. Web application module:

* Login feature
* Verify home screen view and database connection
* Adding parking lot information feature

1. Android application module:

* Verify home screen – The home screen should display map with user current location.
* Verify database connection from android application.
* Verify search functionality for nearby parking lots given the current location as address.

**References:**

[1]Pressman, Roger S, McGraw Hill, " Software Engineering: A Practitioner's Approach,"(7th Edition)

[2] “Parkdroid” <https://play.google.com/store/apps/details?id=com.parkdroid&hl=en>,

[3]EdwardKim,“CarLocator”<https://play.google.com/store/apps/details?id=com.edwardkim.android.carlocatorfull&hl=en>

[4]”ParkMeRight:FreeCarLocator”<https://play.google.com/store/apps/details?id=com.parkmeright&hl=en>

[5] Carlos Coronel , Peter Rob ,Steven Morris, ”Database Systems: Design, Implementation and Management”(9th Edition)

[6] Johannes Gehrke, Raghu Ramakrishnan,”Database Management Systems”(3rd Edition)

[7] Rybinski, H. (1987), ACM Transactions on Database Systems ,"On First-Order-Logic Databases".

[8] Bret Pettichord, Cem Kaner, James Bach, “Lessons Learned in Software Testing: A Context-Driven Approach” (1st edition)

[9] Grgurina, R.; Brestovac, G.; Grbac, T.G., "Development environment for Android application development: An experience report," MIPRO, 2011 Proceedings of the 34th International Convention, vol., no., pp.1693,1698, 23-27 May 2011

[10] Chatzimilioudis, G.; Konstantinidis, A.; Laoudias, C.; Zeinalipour-Yazti, D., "Crowdsourcing with Smartphones," Internet Computing, IEEE , vol.16, no.5, pp.36,44,Sept.-Oct.2012doi: 10.1109/MIC.2012.70