**UNIVERSITY OF CYPRUS**

*Undergraduate Thesis*

# Smart Parking Service and Application

Christakis Vasiliou

*Department of Computer Science*

May 2018

**UNIVERSITY OF CYPRUS**
**COMPUTER SCIENCE DEPARMENT**

# Smart Parking Service and Application

Christakis Vasiliou

<u>Supervisor</u>
Mr Vasos Vassiliou

This thesis was submitted for partial completion of the requirements for an Undergraduate Degree in Computer Science

May 2018

# *Acknowledgments*

Foremost, I would like to express my sincere gratitude to my adviser Dr. Vasos Vassiliou for the continuous support and guidance in order to provide the best result possible during this Thesis study.

Also I would like to thank my family and my friends for their support during the duration of this Thesis study as well as my whole academic career.

# *Abstract*

The purpose of this thesis is the creation of a Smart Parking system and Application that will aid the drivers in their search for available parking places when they want to go to a destination. The system will also provide functionality for the parking providers to manage their own parking places and for the system administrators to manage the whole system. To begin with the whole smart parking problem was researched for drivers and parking providers alike in order for possible solutions to be considered. Various similar applications were studied including one previously developed from a university student parts of whose work were inserted into the new system. The system consists of a website which will offer the functionality for the parking providers to manage their own parking places and an application for the normal users to be able to search for those places, based on their personal preferences. A small part was also created for the system administrators to add the new parking places into the system in order to maintain a safe way of accessing the system. The whole system architecture, design and implementation is explained throughout this thesis. For the parking search the mobile application which can be deployable on multiple mobile platforms was developed. The user sets his destination and by taking the information from the mobile phone the system can process all the parking places available and return the results in descending order based on a score. The score is calculated based on a priority of preferences set by the user by putting them on a fuzzy logic algorithm. Other minor functions were also implemented which will be discussed in this thesis along with the specific preferences mentioned before. The website provides an interface for the parking providers to change their own parking places information and make them available or unavailable in the system. For the addition of new parking places the administrator will only have access to upload them into the system through the administrator console. The whole functionality mentioned above is tested throughout this thesis and certain recommendations and additions are mentioned for the future as the time was limited.

# Table of Contents

# *List of Figures*

# *Chapter 1*

***1 INDRODUCTION***

## 1.1 Problem Statement

One of the most common everyday problems in a city is the search for suitable parking places. It is not only the search of just any parking place but also the frustration of finding a parking place that satisfies our needs and preferences. Each one of us have experienced the exhaustive research for a parking place once we arrive at a destination, searching in neighboring places for the nearest and cheapest parking we can find. Also we are stuck in traffic at times and the distance we have to drive to the desired parking place is too long. At the end in the worst case we may use an alternative method of transport to go to our destination or not go at all if the problem is too big for us.

As I mentioned before, the two most common problems that anyone can face is the discovery of a parking that is cheap and close to the destination. There are people that cannot afford paying for a very expensive parking or do not have the intention to do so. As such, they spend much too much time searching for a parking place with many times having no choice but to go to an expensive parking the initially rejected, not to mention the fuel and other costs spend while searching for the alternative. In addition, the distance they have to walk to reach their destination is one of the biggest problems. No one wants to take his car to a place only having to walk even more time to reach their destination and get tired in the process.

As some studies show, up to 75% of the traffic on the streets is caused by drivers looking for street parking and the average time it took them was between 3.5 and 40 minutes [1]. These figures help as realize the severity of the problem and the frustration is causes.

The user is only one side of the system, there are also other parties being affected from the parking problem. Firstly for businesses their location is the key and having accessible parking near them can greatly affect their business. In that case there is a need for a suitable infrastructure for parking providers. Every business or any other interested parking provider has difficulty controlling his own parking places or parking lots and there are only so many applications that give the ability to everyone to add his own parking places and manage them. At the same time, there is a need for controlling the parking places and who parks at what times with law enforcement straggling to control parking offenders. Therefore, the user needs to know what parking places are available and where he is allowed to park at what times. Moreover due to constant change of many factors those particular parking places or parking lots need to be constantly updated by their provider to present up to date information to the user.

Most of the parking applications so far don't take into account the user's preferences but instead they just present the available parking places. The preferences of each driver are not taken into account and as a result he may end up in a parking place that he is not satisfied with. Also some other applications based on crowdsourcing information have been deemed as unreliable as they leave their functionality and usefulness on the good intentions of the user. Therefore, more trusted systems using sensor and IoT technologies need to be implemented for a more real time function.

In conclusion, all the problems mentioned above have motivated the production of this thesis so that that a smart parking system can be implemented that can serve both users and providers.

## 1.2 Contribution

The purpose of this thesis is the design and development of a mobile application for both Android and IOS phones as well as a website for the parking providers to manage their own parking places. Also proper infrastructure will be designed in order for the whole parking system to run smoothly and its components to cooperate well with each other. The system must also be able to provide an easy way for the system administrators to add new parking places into the database and manage the system users.

More specifically with mobile application the users will be able to:

- Search for available parking places near a location through various ways
- Get the parking places sorted based on their personal preferences
- Select and rank those parking places with some filters

As for the parking provider's website, the parking providers will be able to:

- View their own parking places information
- Change their own parking places information
- Change if their parking places will be available or not to the users

This solution is very important for real life situations where a company or any other business wants to put her own parking places into a public system where other users will be able to search for those parking places when they want to reach that business location.

Let us take the example of a coffee shop which has 4-5 parking places behind its building. This coffee shop would like its customers to know that there are available parking places behind the coffee shop when they want to visit it. It can also give a name to those places and put a price on them. It can very easily do that by contacting the parking system administrators and provide them with its parking places information in a file. The parking administrator will then add the parking places to the system and give access to the provider (coffee shop owner) so he will be able through the provider website to access those places and manage them.

On the other side there are the customers which want to visit the coffee shop. They would be able to make a search for available parking places near the coffee shop's location through the mobile application. The application will return to them the most relevant parking places based on their preferences that will usually include those parking places added by the coffee shop owner.

Finally if the coffee shop is closed for a period, they can choose through the website to disable their parking places from appearing in the parking application results, in other words make them unavailable

## 1.3 Thesis outline

This thesis will be organized into five chapters as follows:

Chapter 2: This chapter will present some similar and related applications on smart parking along with some background concepts that were studied in order understand the smart parking problem and its solutions.

Chapter 3: In this chapter the design of the whole parking system as well as its architecture will be analyzed. Every part and role of the system is explained using use cases, component and data flow diagrams.

Chapter 4: In this chapter all the technologies and tools used to implement the parking system are explained and the methodology used to implement it.

Chapter 5: In this phase the whole system is tested that it performs correctly all his tasks through different scenarios and experiments. Proof of that is shown by a variety of screenshots showcasing the results.

Chapter 6: The final chapter consists of some conclusions produced by this thesis and some recommendations for future work.

# Chapter 2

## 2 BACKGROUND AND RELATED WORK

## 2.1 Background Concepts

### 2.1.1 Internet of things

Internet of things [2] is a computing concept on which individual objects are connected together in a network and share information with one another. They usually include sensor technologies, wireless technologies or QR codes. Each individual physical object has its identification in the network and in that way it provides the direct integration of the physical world into computer systems. The downside is that sometimes it poses risks through cyber-attacks that can affect the real word and cause real life damage.

### 2.1.2 Smart City

A smart City [3] is a municipality that uses different technologies in order to increase efficiency, share information with the public and improve the quality of the government services. A smart city mostly operates through the interconnection of multiple user devices. Many of the city's assets can be managed including local departments, information systems, schools, libraries, transportation systems, power plans, water supply networks, waste management, law enforcement, and other community services. In general a smart city aims in improving city functions and economic growth while increasing the quality of life for the citizens. In the concept of this thesis the smart parking system that will be developed can be considered as a part of a smart city.

### 2.1.3 Smart Parking

Smart Parking [4] is any parking system that helps drivers find available parking spots. It typically obtains information about parking places in a certain geographical area and usually will use sensors in order to determine if the parking place is empty or occupied.

It may benefit its users using the location information of the parking place and its goal is also the balance between the parking demand and the parking supply.

### 2.1.4 Application Program Interface (API)

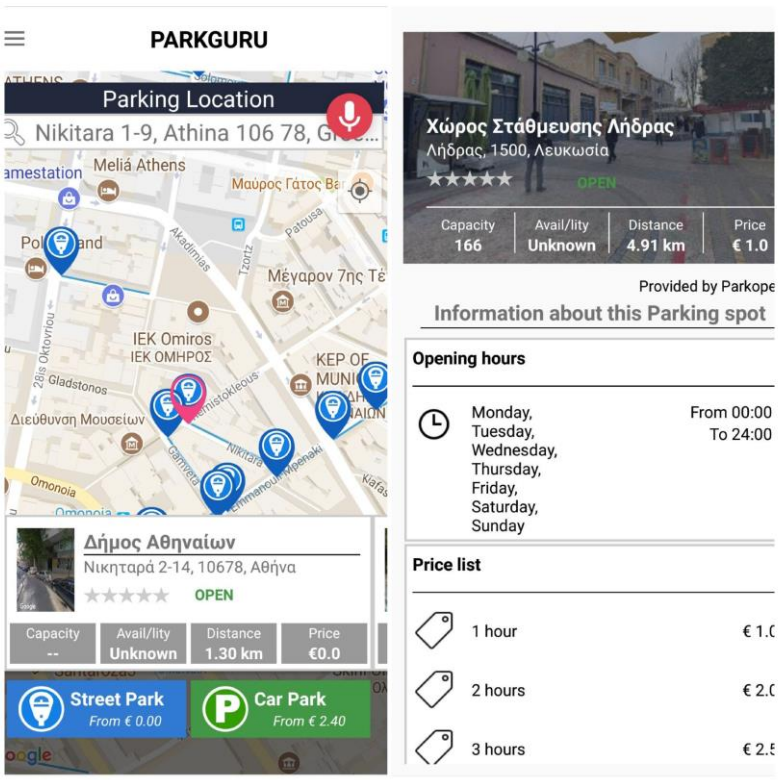
An API [5] is a set of functions and protocols that programmers will use to interact with an external system. It provides them with commands that perform certain things so they don’t have to program them from scratch. The three basic categories of APIs are local APIs, web APIs and program APIs. The local APIs are used like a middleware software inside an application, the web APIs are usually using the HTTP protocol to send requests and receive responses and the program APIs use a remote control to make a remote program appear as if it is local. In this thesis a local API will be used as well as external ones in order to request parking places information and execute other parking related functions.

### 2.1.5 Fuzzy logic

Fuzzy logic [6] is an algorithmic approach that uses levels of truth instead of the standard 0 or 1 values. In this logic the variables can take any number between 0 and 1 with them being the two extremes. Fuzzy logic is closer to our way of thinking where everything is not black or white but we know something to have more value of truth than something else. It is mainly used in different artificial intelligence applications. In this thesis a fuzzy logic algorithm will be used to sort the parking places results to the user.

## 2.2 Related Mobile Applications

### 2.2.1 ParkGuru



Δήμος Αθηναίων
Νικηταρά 2-14, 10678, Αθήνα
★★★★★ OPEN

| Capacity | Avail/lity | Distance | Price |
|---|---|---|---|
| -- | Unknown | 1.30 km | €0.0 |

Χώρος Στάθμευσης Λήδρας
Λήδρας, 1500, Λευκωσία
★★★★★ OPEN

| Capacity | Avail/lity | Distance | Price |
|---|---|---|---|
| 166 | Unknown | 4.91 km | € 1.0 |

Provided by Parkope
Information about this Parking spot

Opening hours

| | |
|---|---|
| Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday | From 00:00 To 24:00 |

Price list

| | |
|---|---|
| 1 hour | € 1.0 |
| 2 hours | € 2.0 |
| 3 hours | € 2.5 |

Figure 2.1: ParkGuru Screenshots

ParkGuru [7] is a mobile application mainly used in Greece for finding parking places including street parking and parking lots. The application allows you to search for parking places at any location you choose on the map. After that, depending on the parking's specifications you have the ability to book the parking place and pay with your credit card. The app also provides you with instructions on how to drive to your desired parking spot. It shows relevant information for each parking spot such as distance, price, availability, if it's provided, and some extra information like the parking's opening hours. Some of the many advantages that the application provides is it's availability of every parking type controlled by the municipal and its easy and safe usage as it can provide you with the option to give voice instructions while driving.

### 2.2.2 ParkAround

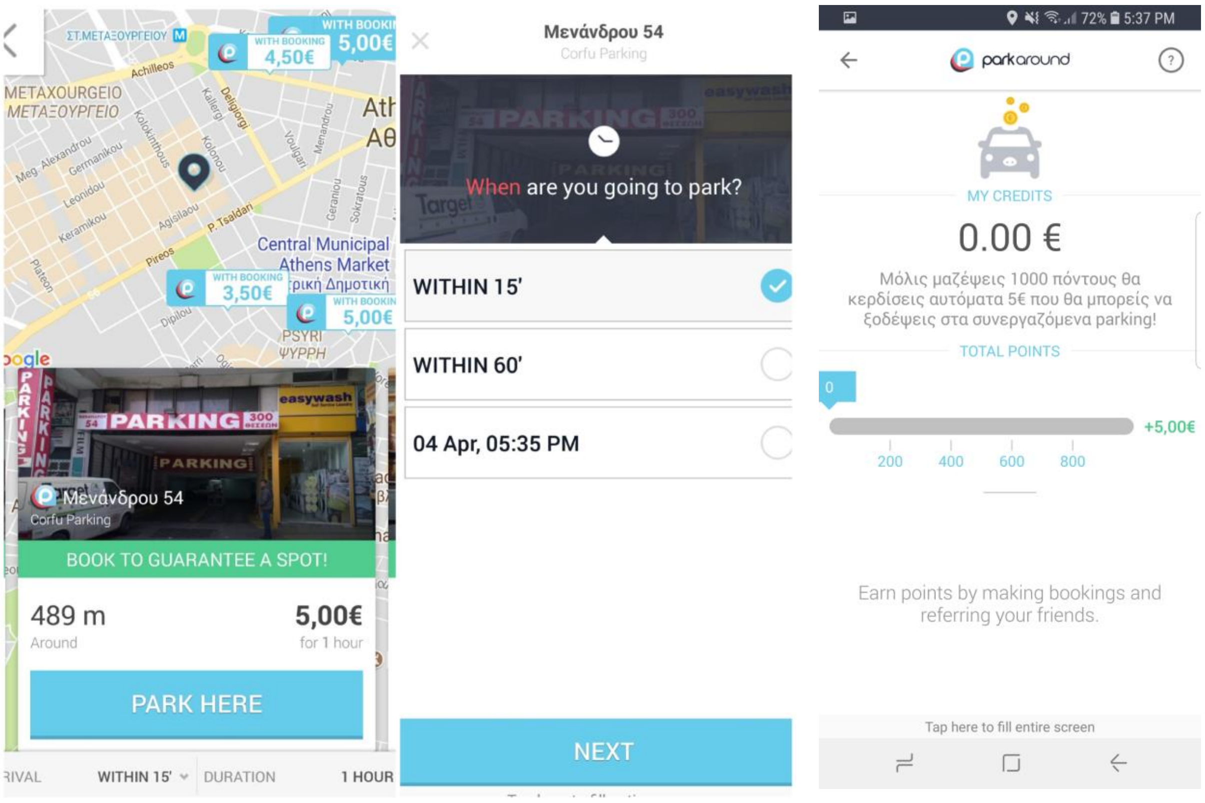
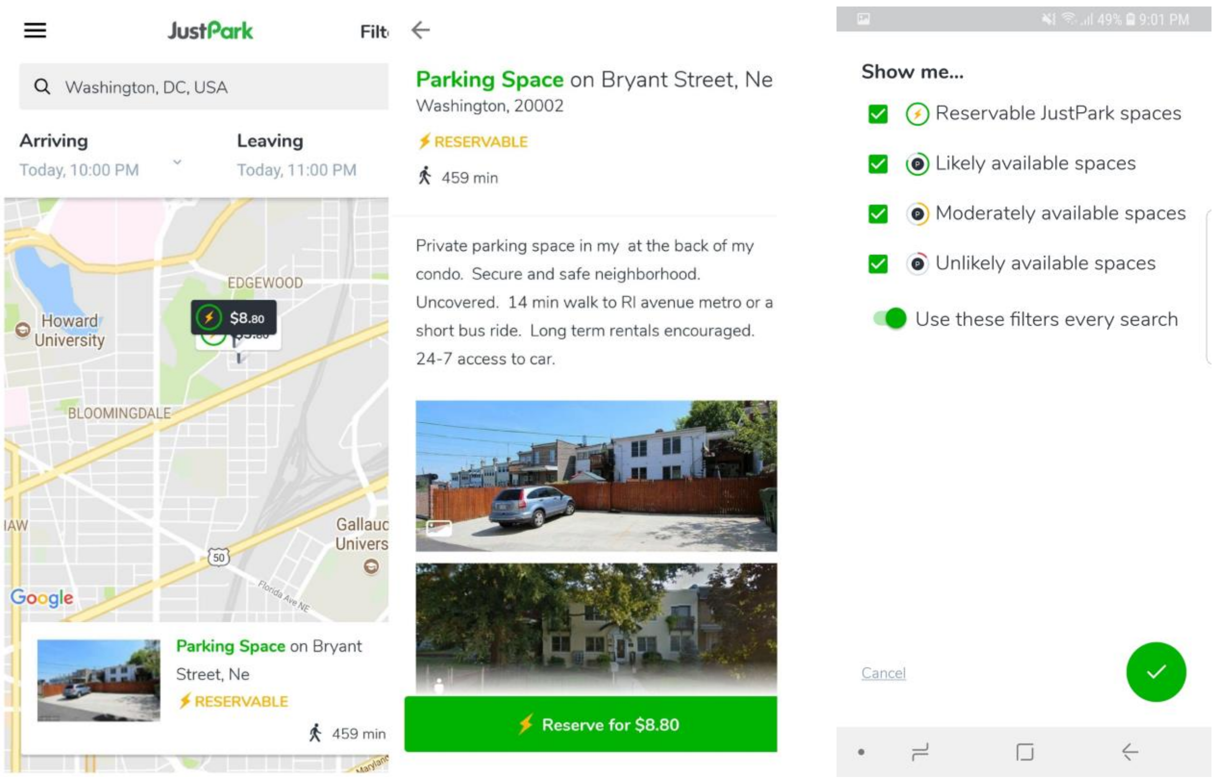

WITH BOOKING 4,50€
WITH BOOKING 5,00€
Μενάνδρου 54
Corfu Parking
BOOK TO GUARANTEE A SPOT!
489 m
Around
5,00€
for 1 hour
PARK HERE
WITHIN 15'
DURATION
1 HOUR
When are you going to park?
WITHIN 15'
WITHIN 60'
04 Apr, 05:35 PM
NEXT
parkaround
MY CREDITS
0.00 €
Μόλις μαζέψεις 1000 πόντους θα κερδίσεις αυτόματα 5€ που θα μπορείς να ξοδέψεις στα συνεργαζόμενα parking!
TOTAL POINTS
+5,00€
Earn points by making bookings and referring your friends.
Tap here to fill entire screen

Figure 2.2: ParkAround Screenshots

ParkAround [8] is another smart parking mobile application based in Greece that allows you to find parking that is usually up to 80% cheaper than normal and book it only through the app. It presents you the parking's location and relevant information providing a picture of the parking as well, so it can be easily findable. Before you book your parking you have to state to the application an approximation of the time that you will arrive at the parking, notifying you before making the booking if the parking will be actually free. The app contains a variety of different parking spots including airport parking that can be also easily accessible through their website as well. The application also provides a rewarding system for its most often users giving points depending on how many bookings you made and discounts as a reward.

### *2.2.3 JustPark*



Parking Space on Bryant Street, Ne
RESERVABLE
459 min

Parking Space on Bryant Street, Ne
Washington, 20002
RESERVABLE
459 min

Private parking space in my at the back of my condo. Secure and safe neighborhood. Uncovered. 14 min walk to RI avenue metro or a short bus ride. Long term rentals encouraged. 24-7 access to car.

Reserve for $8.80

Show me...

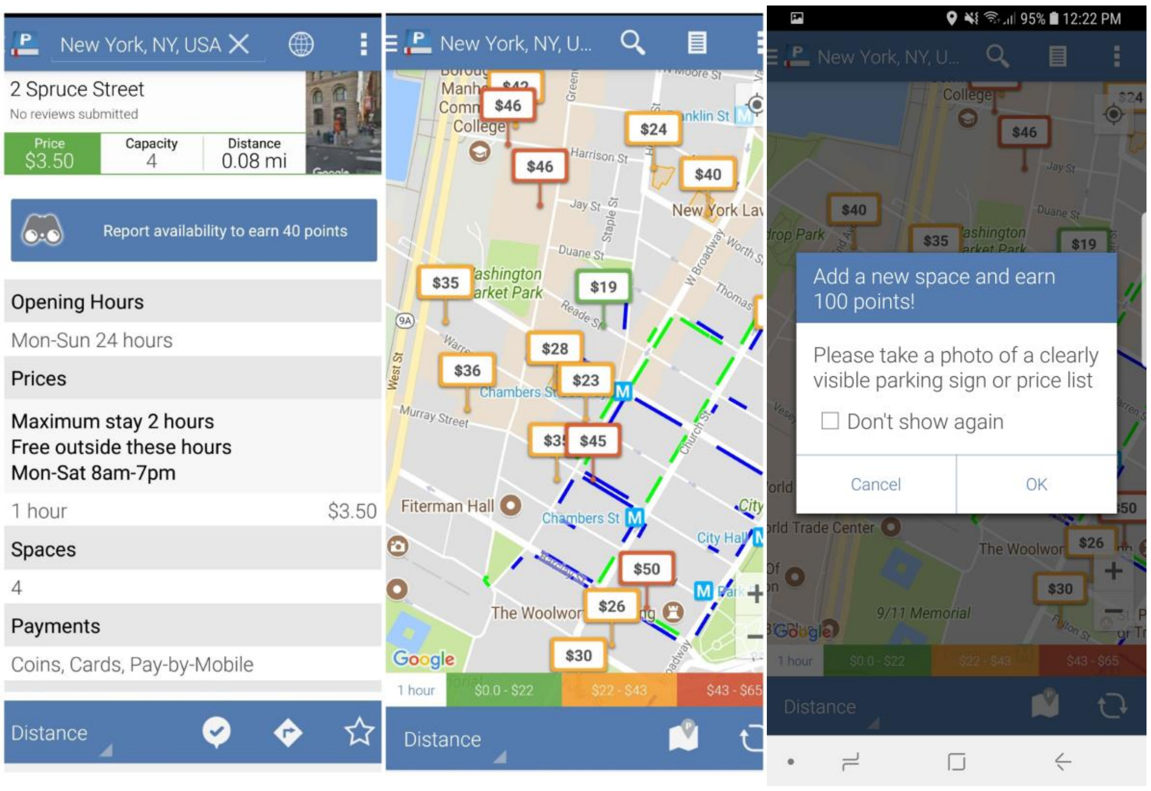
- ✓ Reservable JustPark spaces
- ✓ Likely available spaces
- ✓ Moderately available spaces
- ✓ Unlikely available spaces
- Use these filters every search: on

Cancel

Figure 2.3: JustPark Screenshots

JustPark [9] is a smart parking application that enables the efficient collaboration between parking owners and parking seekers. Through the creation of an account it gives you the ability as an owner to put your own parking spots into the system and later monitor them. As a normal user you can search for any available parking spots in a geographical area by putting you arriving and leaving times. Moreover it presents you with any amount of information uploaded by the parking owner and notifies him directly whenever you have the intention to park there so you can ask him questions and pay him directly. Finally the application provides you with some specific filters you can include in your searches such as showing you the most likely available spaces and those that are reservable.

### 2.2.4 Parkopedia Parking


New York, NY, USA

2 Spruce Street

No reviews submitted

| Price | Capacity | Distance |
|---|---|---|
| $3.50 | 4 | 0.08 mi |

Report availability to earn 40 points

Opening Hours

Mon-Sun 24 hours

Prices

Maximum stay 2 hours
Free outside these hours
Mon-Sat 8am-7pm

| | |
|---|---|
| 1 hour | $3.50 |

Spaces

4

Payments

Coins, Cards, Pay-by-Mobile

Add a new space and earn 100 points!

Please take a photo of a clearly visible parking sign or price list

☐ Don't show again

Cancel OK

Figure 2.4: Parkopedia Screenshots

Parkopedia [10] is a well-developed mobile application that allows you to find parking places either at your current location or on any address in 75 different countries. It provides a variety of filters along with account creation so you can easily manage your bookings. It provides a special feature that allows you to upload a photo of a parking sign or price list so you can add a new parking place into the system. Also it gives you the ability to report the availability of a parking place, post a new photo of it or write a review for it.

### 2.2.5 ParkEasy

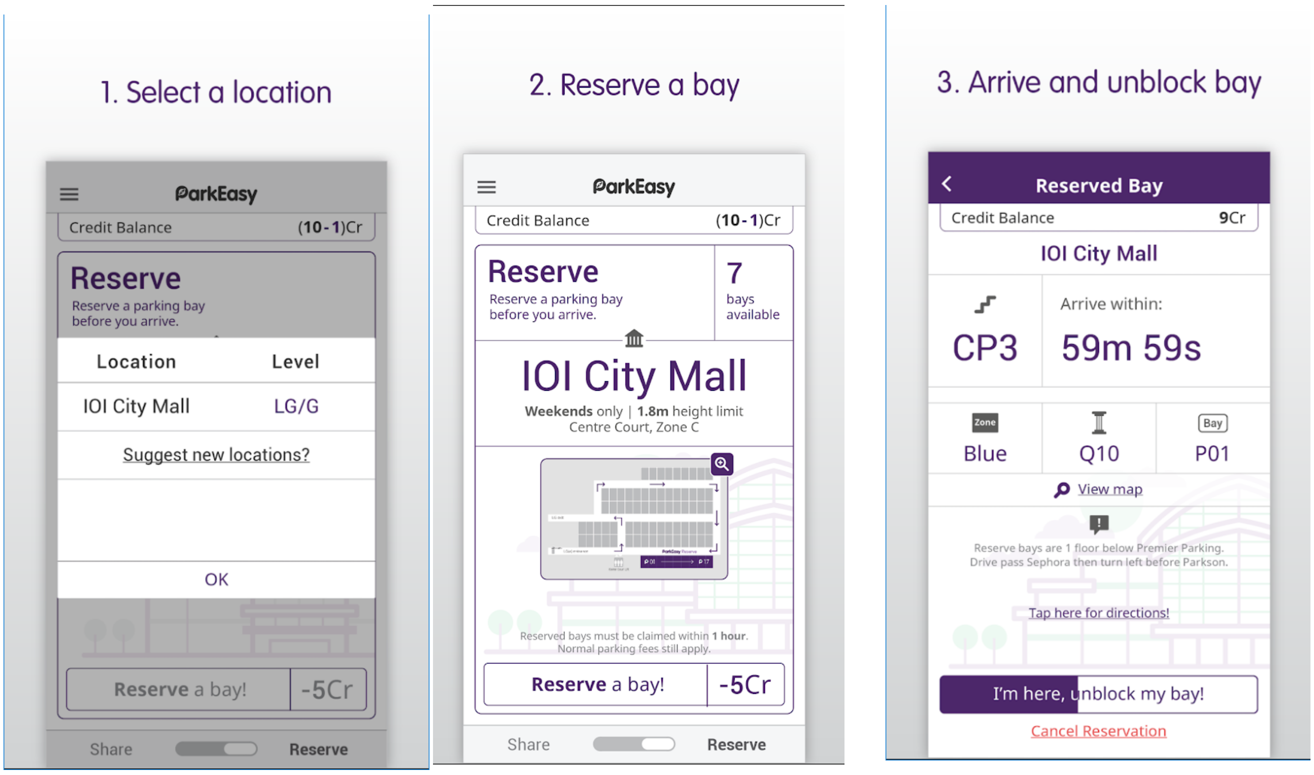


Figure 2.5: ParkEasy Screenshots

ParkEazy [11] is a mobile application that allows you to easily reserve parking spots in shopping malls within Malaysia's busy vicinity. You have the ability to reserve the parking up to one hour prior to your arrival, then the parking is reserved for you and you alone can unlock it and park once you are there. Everything is easily controlled through the application including reserving, paying and unlocking your parking spot.

## 2.3 Past Thesis

One related past thesis that was studied in detail was that of Ioanna Wyrazik, 2016[12] which was on the development of a smart parking system based on information given by the crowd of the application.

The thesis developed a mobile android application as well as a website that would help users point parking locations on the map and the rest of its users to execute a parking search based on their preferences. In particular, when the user arrived at his parking position he stated to the application that he was parked. If a parking place already existed in that place then it was marked as occupied. If there was no parking place on that spot a new one was created automatically and it was marked as a potential parking place. Each time a new user parked at that particular parking place, the place was validated more.

The user was able to search for the already existing and available parking places around a particular location using five preferences including distance and time from current position to parking, distance and time from the parking to the destination and cost. Those parameters were given in a priority by the user and based on a fuzzy algorithm a score was calculated for each individual parking place. The parking places were then presented to the user on a map along with information about them and sorted on a list based on their score.

There was also a website for the parking administrators. On that website you had the ability to create an account, login and view various statistics about the parking places including average requests to the system per hour and per each day of the week. Other kind of statistics were also visible through a statistics page including total number of places on the system, which of them are free, the current unique users of the system and information on how the parking places database is updated.

Finally except from the crowd parking places taken by the user's activity, the database was also importing parking places from various external APIs from around the world.

Bellow you can see the general architecture diagram of that system

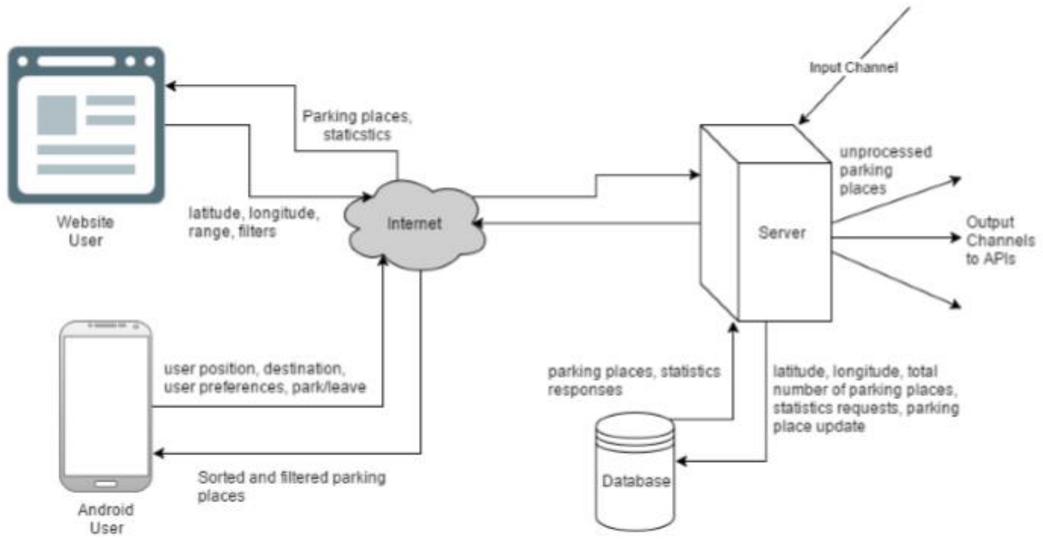

Input Channel
Parking places, staticstics
Website User
latitude, longitude, range, filters
Internet
Server
unprocessed parking places
Output Channels to APIs
user position, destination, user preferences, park/leave
parking places, statistics responses
latitude, longitude, total number of parking places, statistics requests, parking place update
Database
Sorted and filtered parking places
Android User

Figure 2.6: Smart Parking System Architecture of [12]

In general that thesis succeeded its goal, which was to aid the driver in his choice of the best parking place based on his preferences. Assuming that the driver had a mobile phone he could easily find his most preferred parking place and at the same time park and keep a record of where he parked. Also, the crowd was successfully helping in the identification and validation of new parking places. However there is room for improvement in the future in terms of a mobile application that has more preferences for the user and also a more reliable system with better administrative support for the reason that the current system was mainly based on the good intentions of its users.

# *Chapter 3*

## *3 SYSTEM ARCHITECTURE AND DESIGN*

## 3.1 System Design

### *3.1.1 Use Cases*

The Smart Parking System will have three main types of users. The fist type of user is the normal user that will use the system to find available parking places. The second type is the parking provider, which is anyone that will provide the parking system with his parking places and access will be given to him to view and manage his own parking places. The third and final user will be the administrator that will have full control over the system and he will be able to easily upload new parking places into the system that are provided to him by the parking providers. In the figure bellow there is the general use case diagram for those three types of users and their interaction with the system.

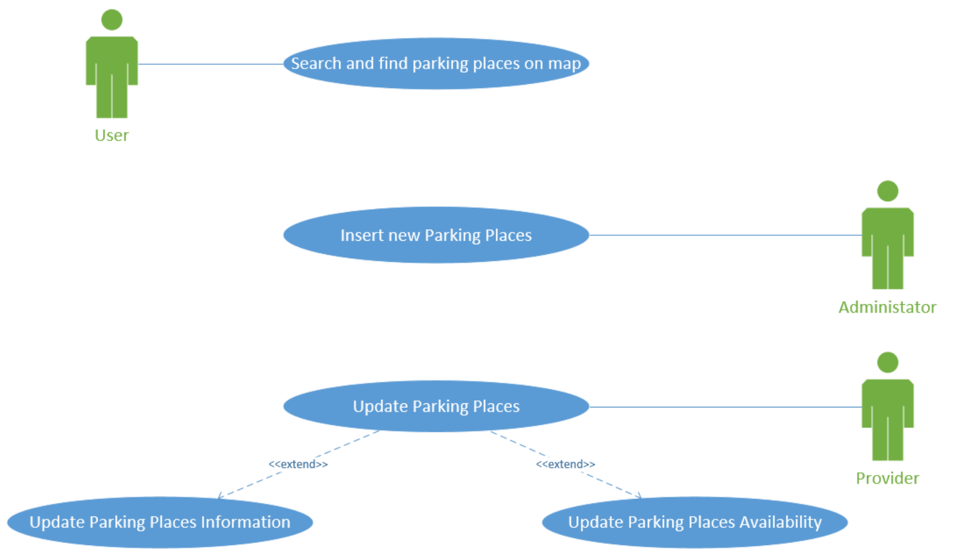

User
Search and find parking places on map
Insert new Parking Places
Administator
Update Parking Places
Provider
<<extend>>
<<extend>>
Update Parking Places Information
Update Parking Places Availability

Figure 3.1: System Use Case Diagram

You can see the five core activities provided by the Smart Parking System. Those activities are executed by the two parts that comprise the system, the mobile web application and the website. In general a user can search and find the available parking places using the mobile application and the administrators and providers and using the website to add and update parking places.

In the figures bellow we see in detail what actions each type of user can perform on the system:

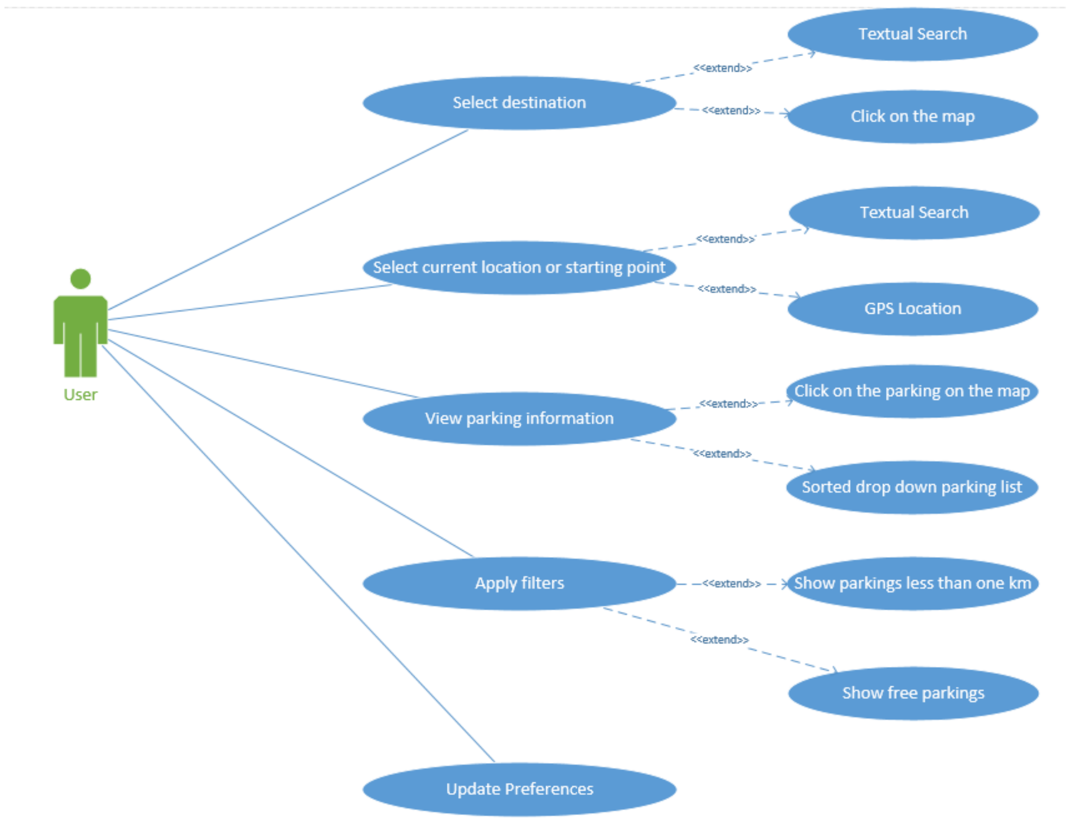


Figure 3.2: Mobile Application Use Case Diagram

In figure 3.2 are all the actions that can be executed by the normal user using the mobile application.

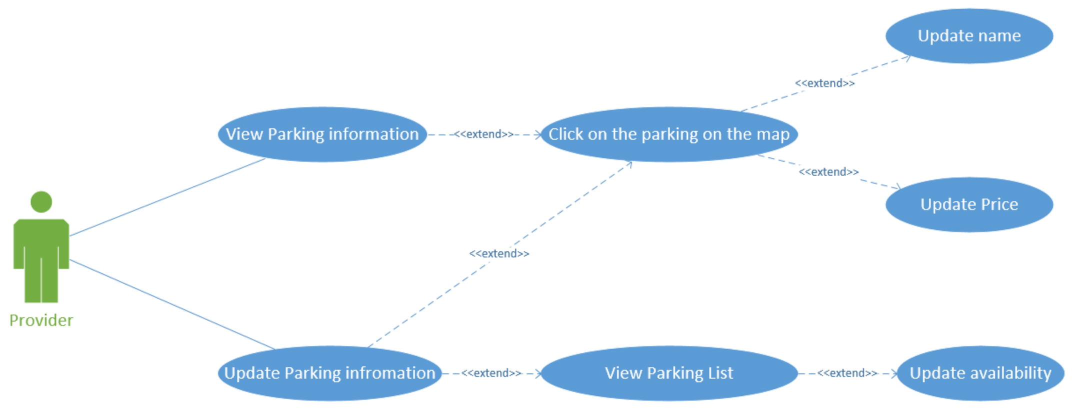


Figure 3.3: Parking Provider Website Use Case Diagram

In the figure above are the actions that can be performed by the parking provider using the Website. Please note that the parking provider cannot add new parking places to the system and that will be done only by the system administrators.

### 3.1.2 Database ER Diagram

For the parking system to work two separated databases are needed to be maintained. One database that will hold all the parking places and lots information and one for the user's information. The users can either be normal users, parking providers or system administrators. The reason their information is kept separate is for more security as they contain sensitive information.

The main database of the system is the parking database that holds all the parking information. The database is hosted in a No SQL database so no relationships between the collections are forced. However using some common fields there is a connection between the parking collections and the Users database. In the figure bellow we present both databases as one database and the relationships between them.

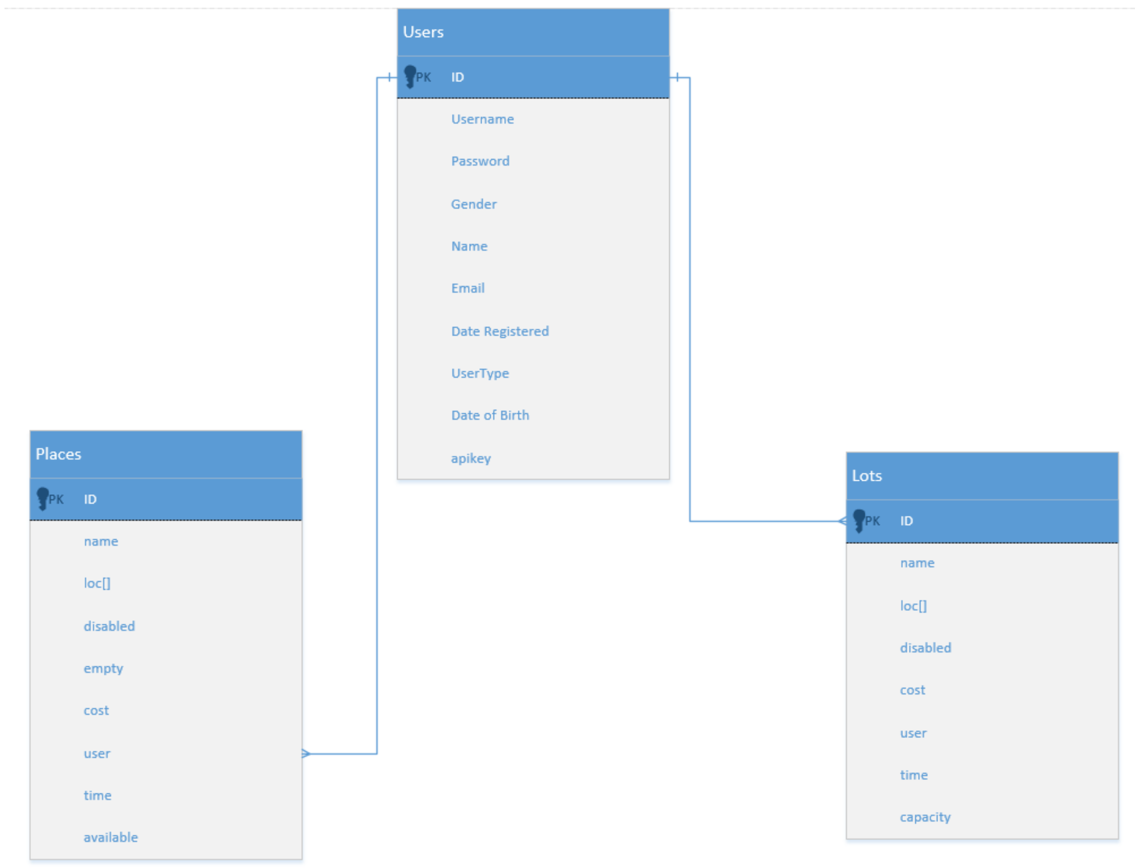


Figure 3.4: Database ER Diagram

There are three main tables/collections in the parking system. The places collection holds the information about the parking places and the lots collection the information about the parking lots. Each of these two collections has a user field, which is the user id of the user owning them. The user id is part of the Users table which contains all the information about the users of the system. The relationship is not forced meaning that a user may or may not have any parking places of his own.

## 3.2 System architecture

### 3.2.1 Overall Flow

In this section the whole system architecture is explained as well as the overall flow of information from one part of the system to another.

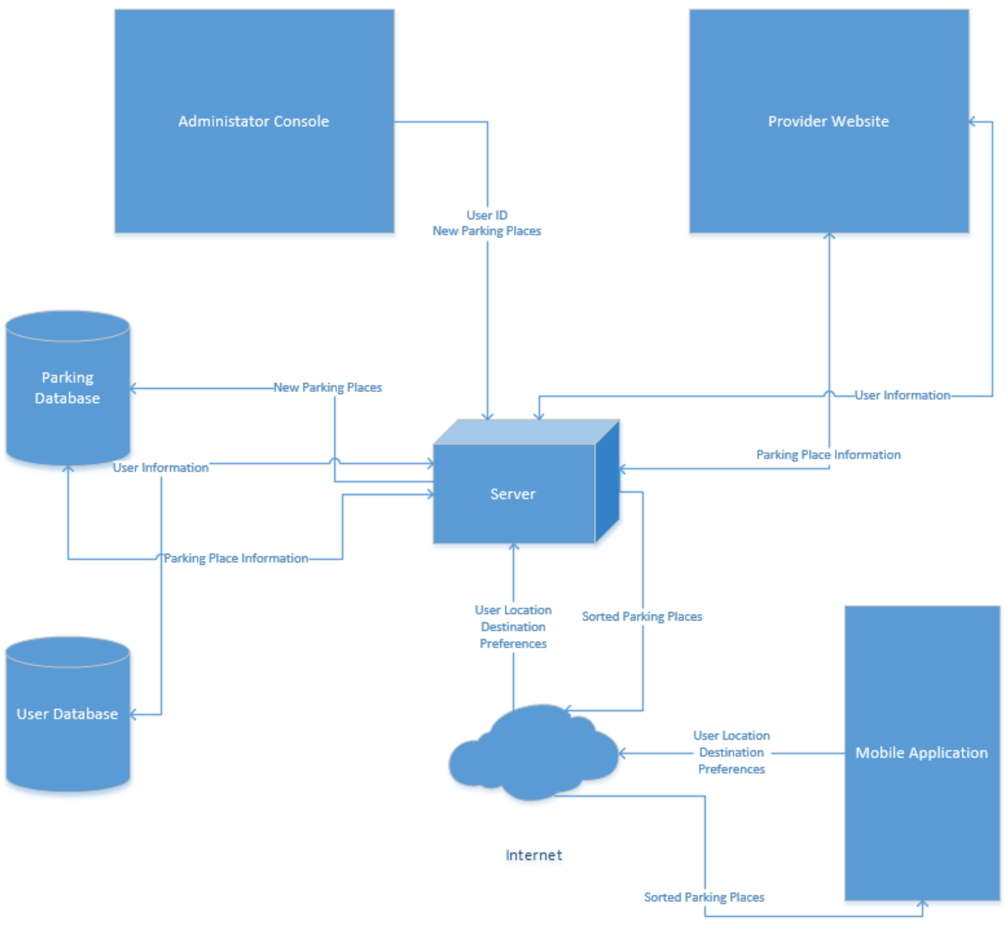


Figure 3.5: System Component Diagram

In the figure above we have the following components that comprise the Smart Parking System:

**Administrator Console**

The Administrator Console will be a simple form that will allow the administrator to add new parking places to the system. The parking places will be given to him manually by the parking provider in a CSV file which will be uploaded through the form to add the new parking places into the database.

**Provider Website**

The provider website will mainly serve as a place for the parking provider to manage his own parking places. The provider will be able to view his parking places information that is send to the website from the database. He will be also able to change his parking place's information with the updated information being send back to the server to update the database.

**Mobile Application**

The mobile application will interact with the normal user that will use the system to find available parking places. The application will be able to send to the system the user's position, desired destination and preferences for processing. The system will process this information on the server with data from the database and will send back to the application the sorted parking places with their information.

**Server**

The server will mainly be used for the communication between the different parts of the system as well as the processing of information. The databases will also be hosted on the server and that will allow them to send their information for processing and get updated by the server. The server will also receive the user's real time information by the mobile application and make the processing necessary to send results back to the user. Finally the server will serve the back-side communication between the provider website, administrator console and the databases.

**Parking Database**

This database will be used to store the parking place's information and the requests for parking made to the server.

**User Database**

This database contains all the system user's information.

Overall the flow of information will go between the mobile application, website and the databases using the server as medium and as a processing component. This is done through the API running on the server that will provide the services to those two components.

### 3.2.2 *Mobile Application*

As mentioned before the mobile application will send real time data to the server along with the user's current location, desired destination and preferences. As to the location the phone's GPS will be used by the application and an option will be available for the user to choose a different location than his current as a starting point.

The user will be able to give his own priority on the following preferences:

- Distance from current location to parking
- Distance from parking to destination
- Time from current location to parking
- Time from parking to destination
- Price

This preferences will each be given a normalized weight based on their priority then send to the server and used in a fuzzy logic algorithm to calculate a score for each individual free parking place near the destination location. The API on the server will then send back the first 50 parking results along with their information sorted based on the score calculated. The information send back to the user for each parking place will include the distance from current location to parking, the distance from parking to destination and the parking price. Details on the figure bellow.

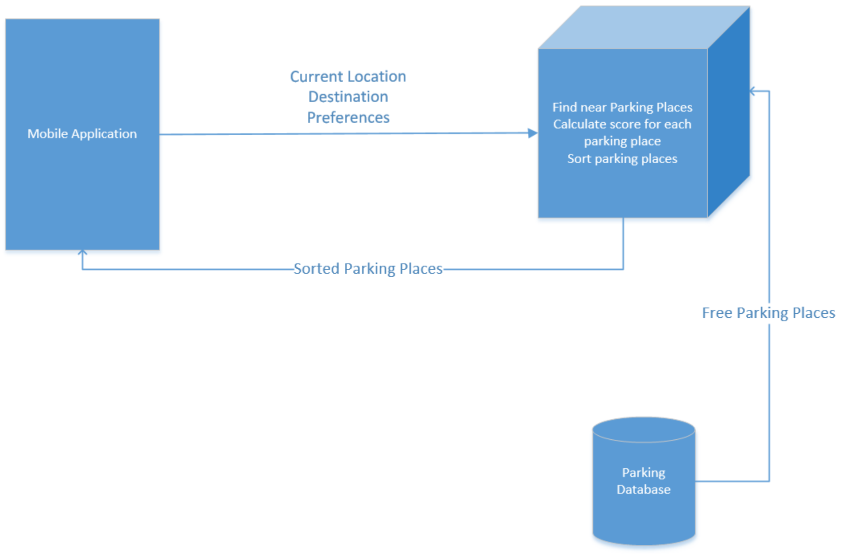


Figure 3.6: Mobile Service Component Diagram

Finally after the user receives the parking places there are two options to filter the parking places based on the nearest to the destination and those that are free of charge.

### *3.2.3 Parking Provider Website*

The parking provider website will give to each parking provider access of his own parking places information so he may change it if he likes. More specifically he will be able to change their name, price and their availability on the parking system. If one parking place is marked as unavailable by its provider then it will not be available to the normal users through the mobile application. The website will use the back-end API in order to communicate with the parking database and access will be given only to the specific provider's parking places. The parking provider will not be able to add his own parking places to the system but that will be done only by the system administrators.

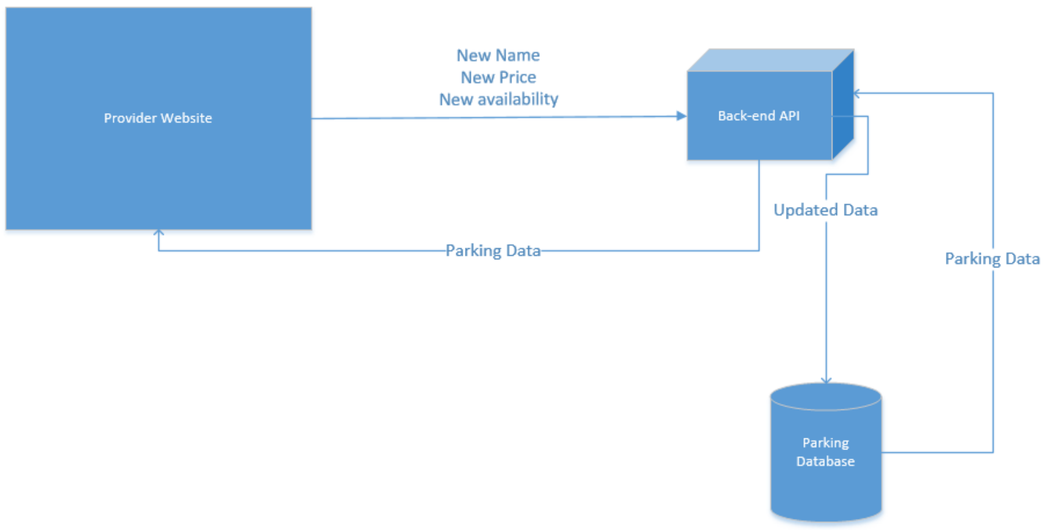


Figure 3.7: Parking Provider Service Component Diagram

As we can see from the figure the back-end API will be used as a mediator for the communication between the parking provider and the parking database and will control which provider has access to which parking places.

### 3.2.4 Administrator Console

Administrator console will give the option to the administrator to upload a file containing the new parking information. By specifying which user, this information will be processed on the server and update the parking database with the new parking places. Only the system administrators will be able to update the database with new parking places and only then the parking providers will be able to access them.

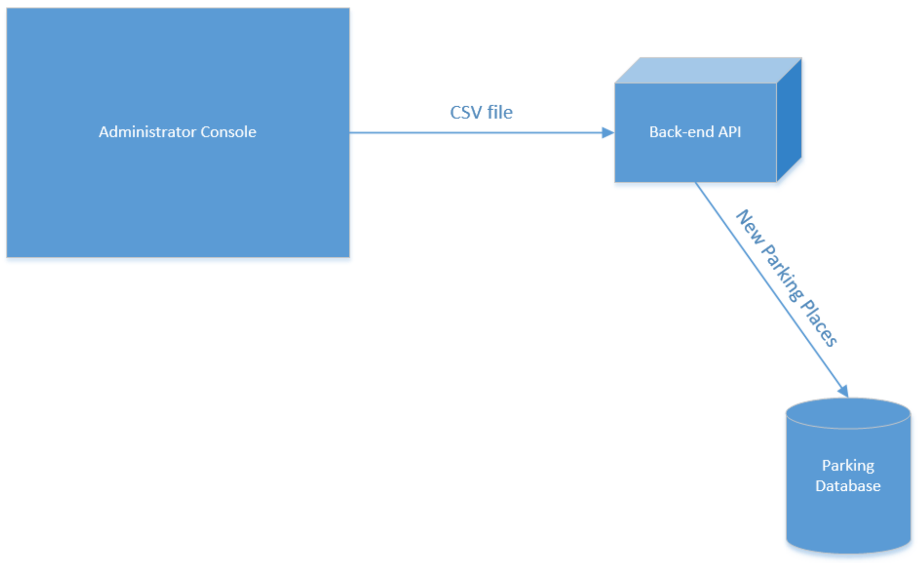


Figure 3.8: Administrator Console Component Diagram

The data flow is really simple, the administrator will upload the csv file containing the new parking places into the back-end API which is located on the server. From there the file will be processed and the new parking places will be added to the parking database.

# Chapter 4

## 4 IMPLEMENTATION

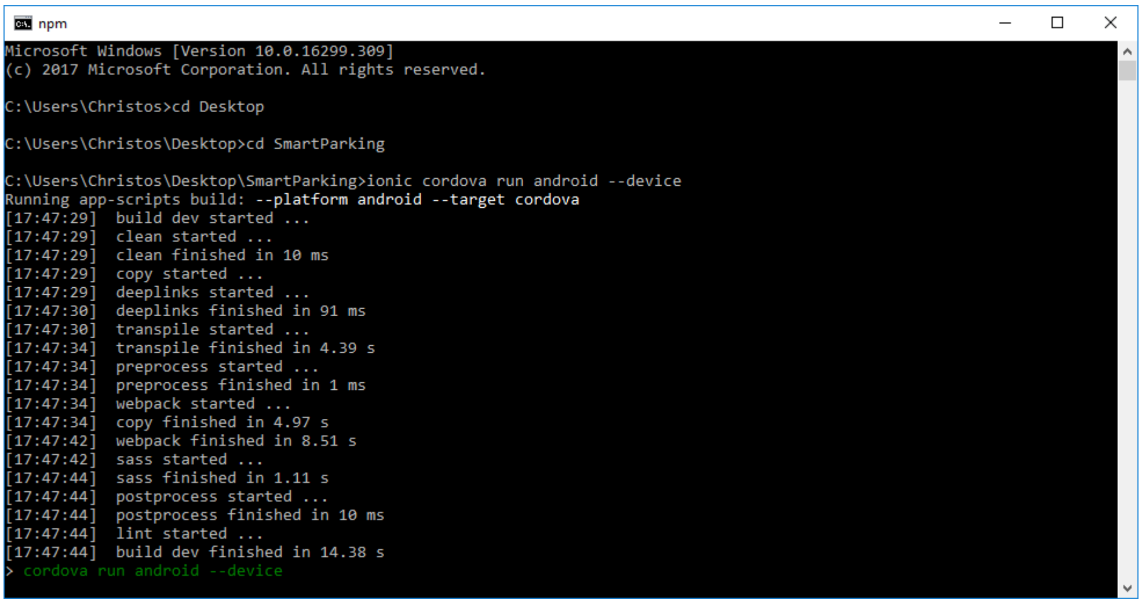
## 4.1 Implementation tools and technologies

In order to implement the whole system various tools and technologies were used and combined together to archive the desired functionality. For the mobile application, Ionic a hybrid mobile development framework was used on which the application was written using web languages. The mobile application was also aided by different APIs like the google maps API in order to provide location and map services to the user. A previously developed fuzzy algorithm API which was written in c# and the .NET framework was used in order to provide the sorted parking places to the mobile user. Also all the parking information is stored in a MongoDb database and the user data on a simple MySQL LocalDB instance. Finally for the development of the provider's website and administrator console Html with Bootstrap was used along with PHP in the back for the communication with the database.

In more detail the following tools were used:

### 4.1.1 Ionic Framework

Ionic [13] is a tool that provides you with the ability to develop hybrid mobile applications using web languages. It uses the node.js framework as its platform for installation, development and debugging on your local system. Its main language is Angular but it also allows you to combine pure JavaScript. Also it has its own libraries for html in the front so that the GUI can have the same appearance as a mobile phone. After the development it gives you options to export your application to any of the supported mobile platforms by creating a web view and later wrapping it with the desired kernel. The reason this framework is used is to give the ability to simultaneously provide a fully functional mobile application for both android and iOS without any difference between them.



```
npm
Microsoft Windows [Version 10.0.16299.309]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\Christos>cd Desktop

C:\Users\Christos\Desktop>cd SmartParking

C:\Users\Christos\Desktop\SmartParking>ionic cordova run android --device
Running app-scripts build: --platform android --target cordova
[17:47:29]  build dev started ...
[17:47:29]  clean started ...
[17:47:29]  clean finished in 10 ms
[17:47:29]  copy started ...
[17:47:29]  deeplinks started ...
[17:47:30]  deeplinks finished in 91 ms
[17:47:30]  transpile started ...
[17:47:34]  transpile finished in 4.39 s
[17:47:34]  preprocess started ...
[17:47:34]  preprocess finished in 1 ms
[17:47:34]  webpack started ...
[17:47:34]  copy finished in 4.97 s
[17:47:42]  webpack finished in 8.51 s
[17:47:42]  sass started ...
[17:47:44]  sass finished in 1.11 s
[17:47:44]  postprocess started ...
[17:47:44]  postprocess finished in 10 ms
[17:47:44]  lint started ...
[17:47:44]  build dev finished in 14.38 s
> cordova run android --device
```

Figure 4.1: Ionic Framework Mobile Phone deploy screenshot

In the above screenshot you can see how the application is compiled and exported to the desired mobile opereting system which in this case is android. In this example the mobile phone is connected to the pc and the application is  autmaticaly deployed on that phone.

### *4.1.2 Google Maps JavaScript API*

Google Maps JavaScript API [14] provides you with many of the services a google map can provide so that the developers can use them on their own websites and applications. It contains very easy documentation and you only have to get your own API key for free in order to use the supported services up to specific frequency. In this system the google maps JavaScript API is used by the mobile application in order to provide the users with an interactive map containing the various parking places as markers and allow the user to interact with them and state his destination preference on the map.

### 4.1.3 Google Places API

Google Places API [15] can be combined with Google Maps to give access of a large database with different places all around the world to developers. A request can be made on a specific location with a certain radius and return back the results based on a text query. The results include the places associated with the text query along with extra information about them. In the mobile application Google Places API is used to give the ability to the user to search for any particular place on the map to use it as a destination location or as a starting location.

### 4.1.4 PHP with MongoDB Library

PHP is a very common server-side programming language that allows the communication of the front end with back end Database while providing various functionality that can be carried out on the back server-side. PHP with a MongoDB library imported is used on the server. With the library you gain the functionality to connect to MongoDB and execute operations like adding, reading or changing records, then PHP is communicating with the front end using GET and POST commands to execute the desired operations. PHP is specifically used in the provider's website to give providers access to their own parking places on the database. PHP is also used on the administrator console to import the parking places into the database using a csv file.

### 4.1.5 MongoDB

MongoDB is a free document-orientated database that uses JSON format for its data representation. That makes it extremely convenient to exchange information with a web service. It is also classified as a NoSQL Database which means that is very flexible allowing for flexible records and collections. In this smart parking system is used to store all the parking places and parking lots information along with the requests that are made to the system. The reason MongoDB is used instead of an SQL Server is because of its easy communication with the web services and also because it provides various libraries that let you perform geographical searches. For example by having the coordinates of the parking places you can execute a query on a specific geographical area using the coordinate’s data from the parking places records.

#### *4.1.6 SQL Server LocalDB*

LocalDB is a lightweight SQL server that uses only the minimal files necessary to operate. The LocalDB is only used to store the user’s personal information and provide simple login services to them. The user’s personal information is not stored in MongoDB for the reason that an SQL server is more secure and it’s needed for sensitive information.

#### *4.1.7 Windows Server 2012R2*

For the hosting of the web services and the system databases a machine with Windows Server 2012R2 64bit operating system is used. In order to host those services the system operates an IIS Web Server. The reason Windows Server is used is the hosting of the .NET web services that require IIS which can only be installed on a Windows Server.

## 4.2 Implementation Process

Using the various tools and technologies, the website and mobile application were developed as well as the API. The final product was a combination of newly developed systems as well reuse of some already developed parts. The system was developed in such way that all the components communicate well with each other and are combined to produce results.

### 4.2.1 Server and API

As mentioned before the server runs on Windows Server 2012R2. That system was already installed along with a running IIS web server. Internet information technologies (IIS) is a flexible, secure and manageable web server for hosting anything on the web developed by Microsoft. The whole website as well as the API will be hosted on that server. A previously developed API taken as a whole is used to process the information coming from the mobile application as discussed before. That API was part of a previews website written in .net C# MVC and only its back API services are used through POST and GET requests from the mobile application
PHP with MongoDB library is installed on the server and four PHP files were developed in order to receive requests from the parking provider website and administrator console. Specifically the PHP files are used for the reasons bellow accordingly:

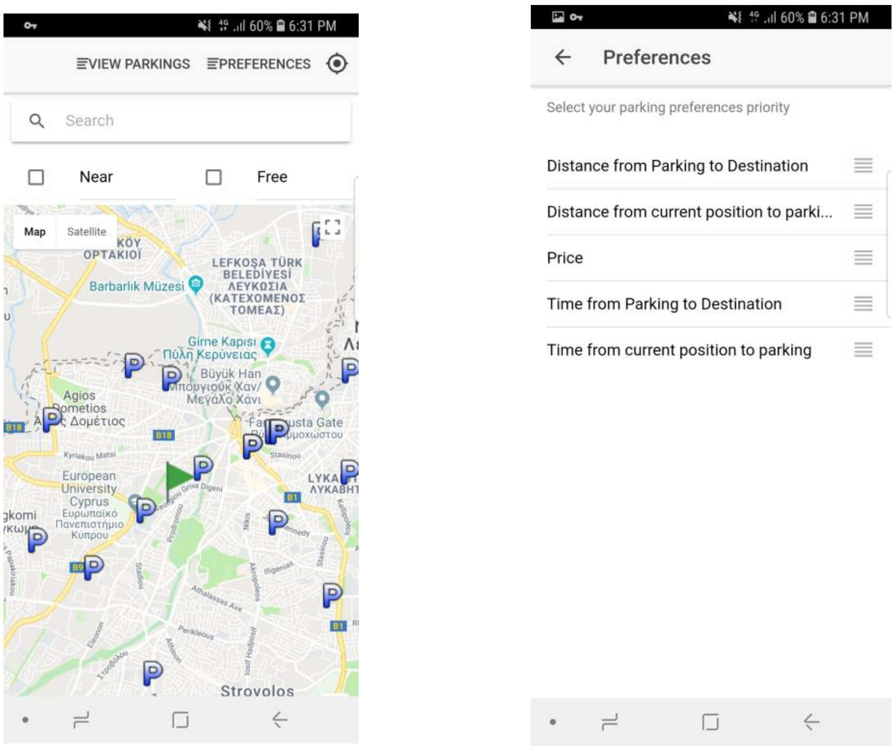
- Receive the CSV file from the administrator console and add the new parking places to the database
- Update the parking database based on the requests for change received by the parking provider website
- Update the availability of the parking places of the database based on the requests from the parking provider website
- Retrieve the parking places information from the database and send it back to parking provider website

This PHP files conclude the API used for the website and are used separately from the c# API serving the mobile application.

### 4.2.2 Mobile Application

For the mobile application as mentioned before, IONIC, a framework for hybrid mobile application development using web languages is used. The application was developed and tested using node.js and exported into the desired mobile phone operating system. For the dynamic content we mainly have anglular.js a JavaScript framework as described before and for the view and feel of the mobile application HTML is used formatted in the correct way.

Basically the mobile application consist of two pages on which each one has its own HTML, typescript and CSS file. Also the application executes various GET and POST requests in order to communicate with the API located on the server and receive information. Google maps is imported into the application using an API key and various services from the Google Maps JavaScript API and Google Places API are used. Bellow you can see the two main pages developed on the mobile application.



Select your parking preferences priority

Distance from Parking to Destination
Distance from current position to parki...
Price
Time from Parking to Destination
Time from current position to parking

Figure 4.2: Mobile Application Look Screenshots

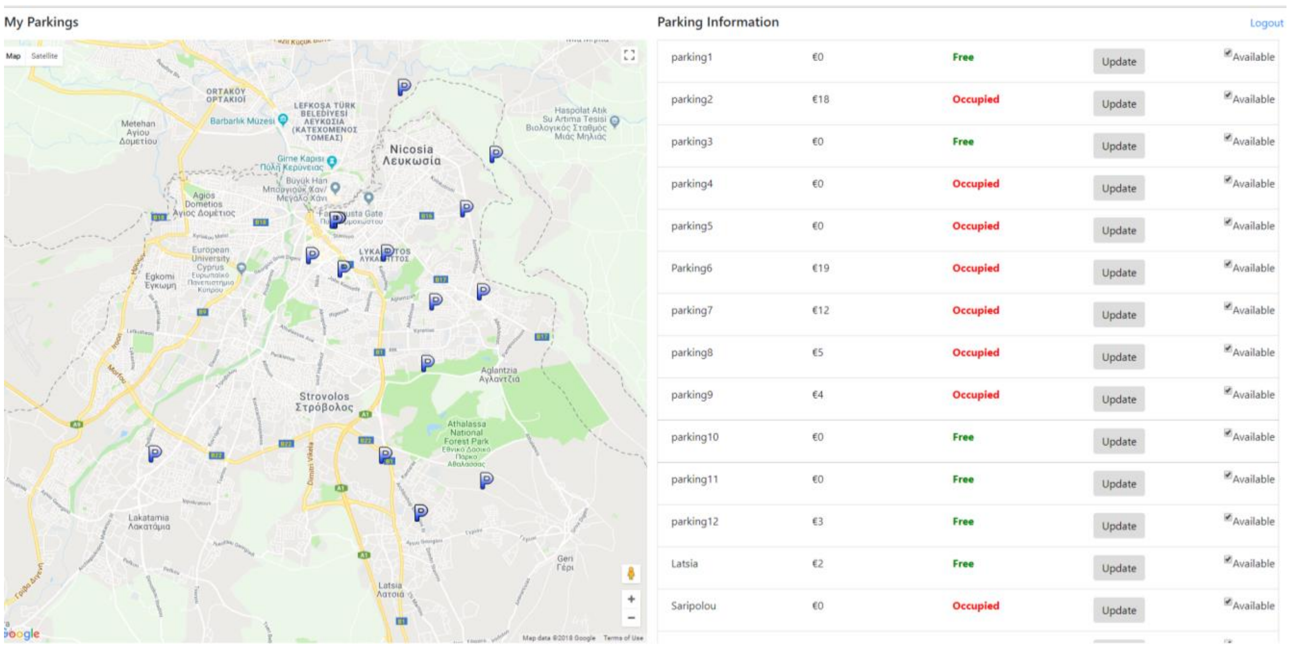
The first page is the main map page developed. From here the user can view the parking places on the map after he completes its search with the green flag pointing to the destination spot inputted by the user. The three buttons on the top of the page are for displaying a sorted list of the parking places, entering the preferences page showing on the right and set a different current location respectively.

The second page is the page where the user sets his preferences priority with which the parking place's score will be calculated.

More details about the functionality will be explained in the scenarios section later.

### 4.2.3 Website

For the website front end we used HTML with pure JavaScript for the developing of the login screen, the provider's parking page and administrator console. Those files are sending POST and GET requests to the back-end API discussed before for communication with the database and extra processing. A small amount of the Bootstrap framework is used to make the website pages appear friendly to the user. Bellow you can see the two main pages of the website, one for the parking providers to manage their parking places and one for the administrator console

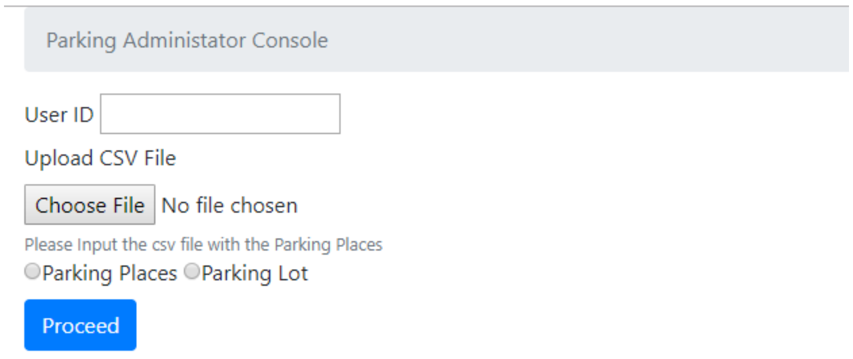


Parking Information

| | | | | |
|---|---|---|---|---|
| parking1 | €0 | Free | Update | ✓ Available |
| parking2 | €18 | Occupied | Update | ✓ Available |
| parking3 | €0 | Free | Update | ✓ Available |
| parking4 | €0 | Occupied | Update | ✓ Available |
| parking5 | €0 | Occupied | Update | ✓ Available |
| Parking6 | €19 | Occupied | Update | ✓ Available |
| parking7 | €12 | Occupied | Update | ✓ Available |
| parking8 | €5 | Occupied | Update | ✓ Available |
| parking9 | €4 | Occupied | Update | ✓ Available |
| parking10 | €0 | Free | Update | ✓ Available |
| parking11 | €0 | Free | Update | ✓ Available |
| parking12 | €3 | Free | Update | ✓ Available |
| Latsia | €2 | Free | Update | ✓ Available |
| Saripolou | €0 | Occupied | Update | ✓ Available |

Figure 4.3: Parking Provider Website Look Screenshot

On the left the parking provider can see his own parking places on the map along with their information on the right. From the parking list he can choose any parking place and update its information or make it unavailable


Parking Administator Console

User ID

Upload CSV File

Please Input the csv file with the Parking Places

Figure 4.4: Administrator Console Look Screenshot

In the figure above how the administrator console used by the parking system administrators to add new parking places to the system looks like. The administrator can put the user's id and upload the file containing his new parking places or parking lots.

### 4.2.4 Databases

As mentioned before the system consists of two databases which are both hosted on the server. The first one is the parking database which is hosted on a MongoDB instance that was already installed into the server. It is accessed from the API on server using a specific portal. The second one is an SQL LocalDB which as mentioned before is a lightweight SQL Server. The whole database is stored in an MDF file which is imported into the SQL Server and it contains all the user information in one table. This database is only accessed through the .Net API mentioned before in order to be really closed and safe as it contains sensitive information.

# Chapter 5

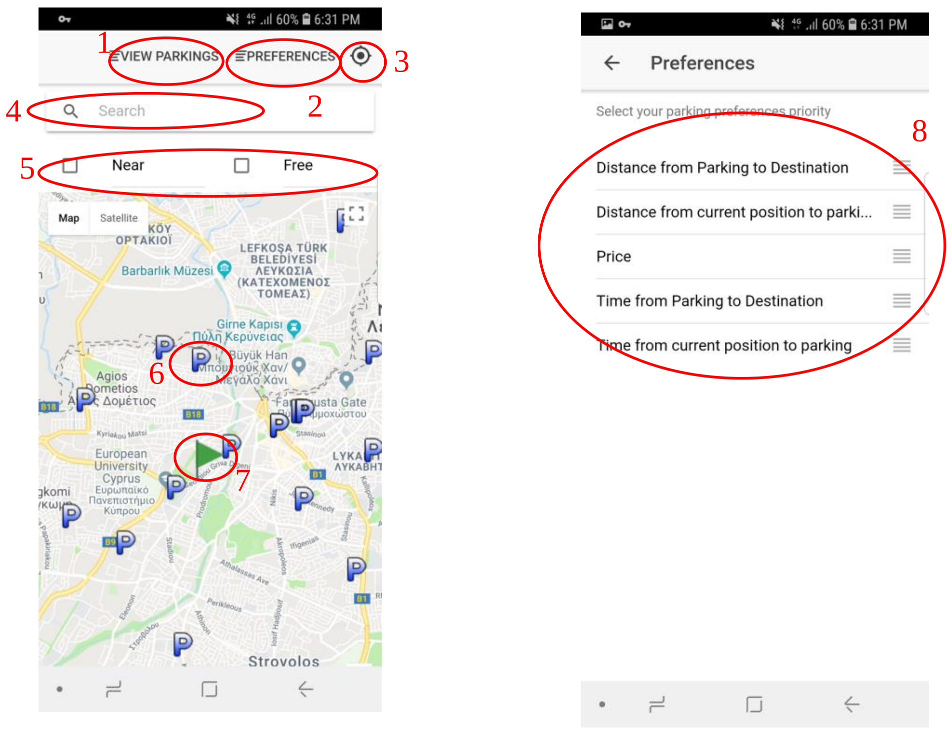
## 5 OPERATION AND EVALUATION

### 5.1 User Interface Overview

In this section the whole Smart Parking System and its functionality is discussed by giving a variety of screen shots that showcase the results. The screenshots will show in detail both the mobile application and the parking provider website and how their features look like to the user.

#### 5.1.1 Mobile Application

As discussed earlier the Mobile Application has two main screens, one is the main map screen with the parking information and the other is the preferences screen where the user sets its own personal preferences. Bellow will have an overview of each screen and its functionality.



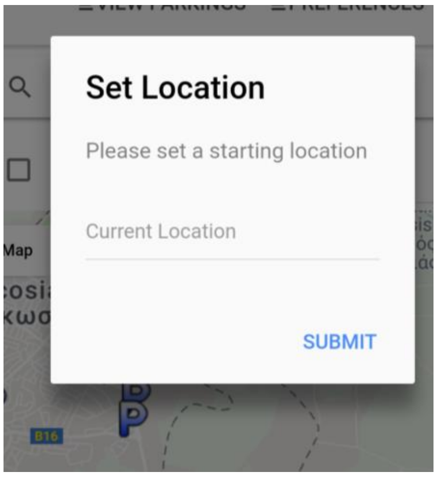
VIEW PARKINGS
PREFERENCES
Search
Near
Free
Map
Satellite
1
2
3
4
5
6
7
8
Preferences
Select your parking preferences priority
Distance from Parking to Destination
Distance from current position to parki...
Price
Time from Parking to Destination
Time from current position to parking

Figure 5.1: Mobile Application Functionality screenshots

1. This button will show all the results with the parking places that came back to the Mobile application and are now appearing on the map. The results are sorted based on the score of each parking place. The list appearing is shown below and provides a quick way for the user to select any parking place on the map.

2. The button takes you to the screen on the right where the user can choose his own preferences priority, more on that later.

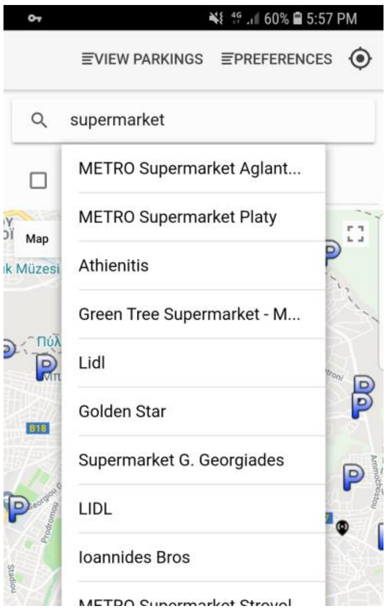
3. By pressing the button the below search bar will appear to the user. Through this search bar he can search for a different place to set as its current location. From that point on all the distances will be calculated taking into account that position as his current location and starting point. When he wishes to switch back to his real current location provided by the GPS he can do so by clicking the button again and leaving the search bar textbox empty.


Set Location

Please set a starting location

Figure 5.2: Set Location Textbox Screenshot

4. The search bar is used by the user to select a destination by searching for a specific location. A list of different places will appear as shown below and the user can select one of them as its final destination.



METRO Supermarket Aglant...
METRO Supermarket Platy
Athienitis
Green Tree Supermarket - M...
Lidl
Golden Star
Supermarket G. Georgiades
LIDL
Ioannides Bros

Figure 5.3: Destination Textual Search Screenshot

5. These are some filters that the user can put after the results are being shown on the map. He can choose to display only parking places within 1 km of his destination, parking places that are free of charge or both at the same time. The parking places will be filtered and only those satisfying the filters will appear on the map at each time. The user can deselect any of the two filters and the parking places will appear again on the map automatically.

6. A marker corresponds to a specific parking place. By clicking on the marker you can view information about that parking place as shown below

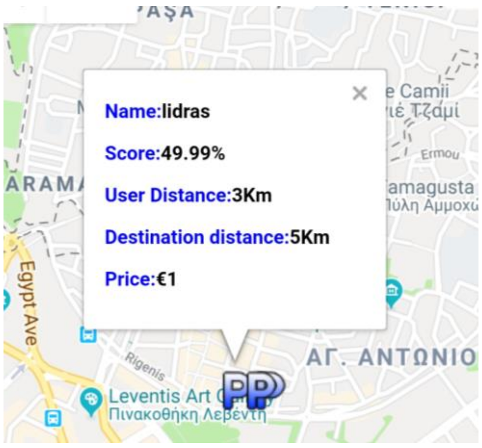

Name:lidras
Score:49.99%
User Distance:3Km
Destination distance:5Km
Price:€1

Figure 5.4: Parking Place Information Screenshot

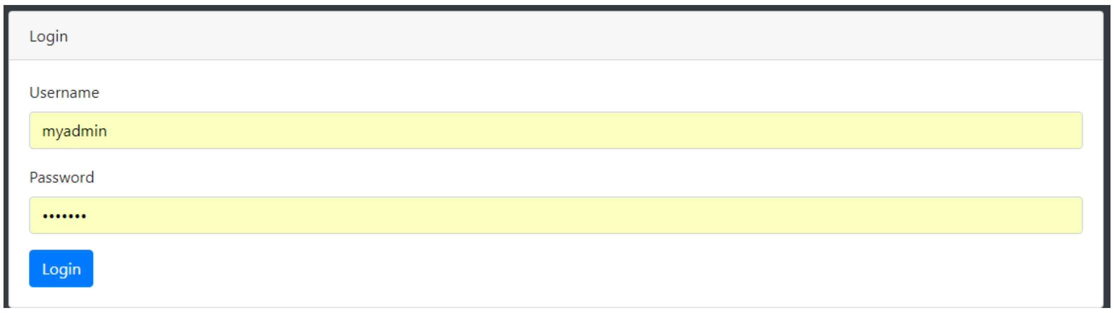
You can view information such as the name of the parking place, its score, which is calculated with the fuzzy algorithm mentioned before, and the parking price. Various distances are also shown such as the distance of the parking place from your current position and the distance of the parking from the destination.

7. The green flag represents the final destination the user has selected. He can select that destination either by clicking on the map or by using the search bar. All the result parking places will then appear using that spot as destination.

8. These are the preferences the user can set a priority on. He can do this very easily as each preference is a sliding item so he can slide them to the correct position and set their priority. After he exits the screen, this priority will be saved and any parking search he performs later will be based on that priority.

### 5.1.2 Website

The website mainly consists of the screen where the parking provider can view and manage his own parking places and the administrator console which is for the parking administrator to add new parking places into the system. Depending on what type of user you are, the website login page will direct you to the correct screen. First let's see how the login page looks like.



Login

Username

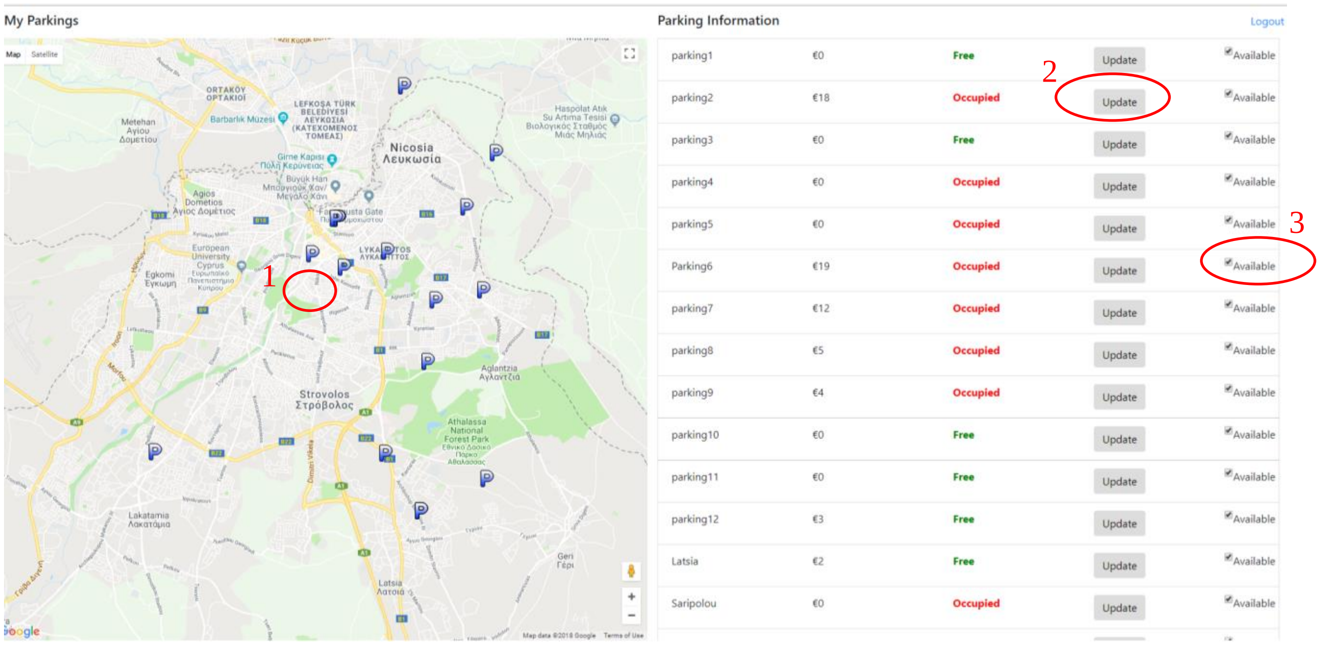
myadmin

Password

•••••••

Figure 5.5: Website Login Screenshot

If you log in as a parking provider the below screen will appear with the following functions.



Parking Information

| | | | | |
|---|---|---|---|---|
| parking1 | €0 | Free | Update | ✓ Available |
| parking2 | €18 | Occupied | Update | ✓ Available |
| parking3 | €0 | Free | Update | ✓ Available |
| parking4 | €0 | Occupied | Update | ✓ Available |
| parking5 | €0 | Occupied | Update | ✓ Available |
| Parking6 | €19 | Occupied | Update | ✓ Available |
| parking7 | €12 | Occupied | Update | ✓ Available |
| parking8 | €5 | Occupied | Update | ✓ Available |
| parking9 | €4 | Occupied | Update | ✓ Available |
| parking10 | €0 | Free | Update | ✓ Available |
| parking11 | €0 | Free | Update | ✓ Available |
| parking12 | €3 | Free | Update | ✓ Available |
| Latsia | €2 | Free | Update | ✓ Available |
| Saripolou | €0 | Occupied | Update | ✓ Available |

Figure 5.6: Parking Provider Website Functionality Screenshot

This is the personalized page for each parking provider to view his own parking places. He can view them on the map to the left and on the list on the right. On the list the provider can see the name, price and availability of each of his parking places.

1. A marker corresponds to a specific parking place. By clicking on the marker the provider can view the name and price of his parking place and alter them as he likes as shown below.

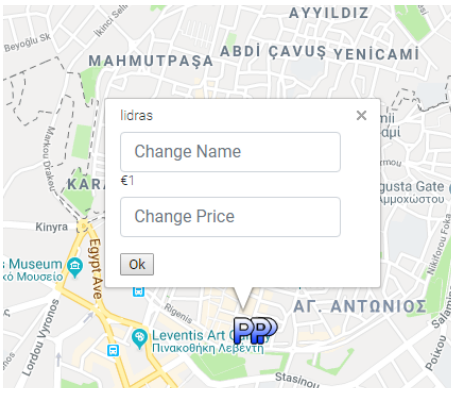


Figure 5.7: Update Parking Place Information Screenshot

2. This button is used to trigger the update window shown above to the selected parking place. In this way the user can easily select which parking place to edit from his list rather than looking for it on the map.

3. This checkbox will show if the parking place is available to the system or not. If the parking place is unavailable the list item will appear grey. The provider can anytime check and uncheck the checkbox to change the availability.

If you log in as a parking system administrator the bellow simple form will appear for you to add new parking places into the system.

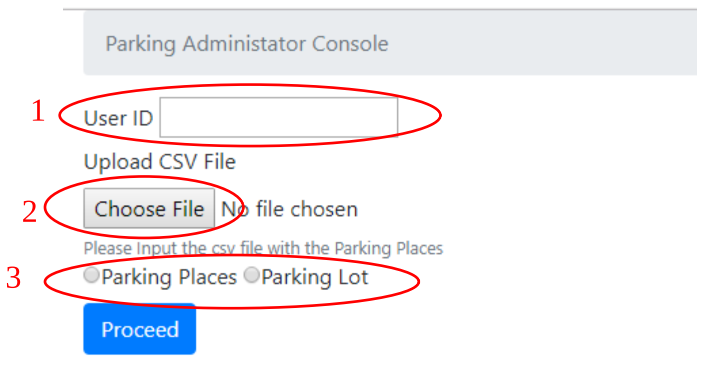


Figure 5.8: Administrator Console Functionality Screenshot

1. In this textbox the administrator will put the user Id which owns the new parking places he is about to insert. For security reasons the administrator will view the user id manually from the database which he only has access.

2. From here the administrator will upload a CSV file containing the new parking places information. The CSV file is given offline to the administrator by the parking provider in a predetermine format.

3. The parking provider has the option to provide the administrator with either alone parking places or parking lots that contain many parking places. The new places will be saved as parking lots if the administrator chooses the parking lot option.

## 5.2 Scenarios and experiments

### 5.2.1 *Validation of parking results*

As discussed before, the mobile application sends back the parking places results sorted based on their individual score. Due to the fact that there are five different preferences you can have a large number of different priority sequences between them. In order to test those results we tested 5 different combinations to see the effect they have on the ordering of the results.

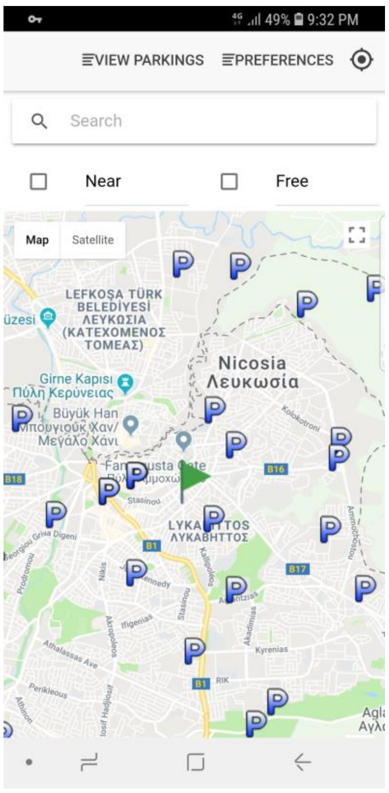


Figure 5.9: Parking Results Experiment Screenshot

We executed the above parking search five times using the same user location and destination (green flag) each time and only changing the priority of the user's preferences. With those results we produced the following tables each containing the parking places relevant information as it is extracted from the API. In the experiments we only specifically stated the ordering of only the first three preferences as there are the only ones that are taken into account in the fuzzy logic algorithm.

| No | Distance from destination(Km) | Distance from user location(km) | Time from destination | Time from user location | Cost |
|---|---|---|---|---|---|
| 1 | 1.116 | 2.209 | 14.58 | 5.21 | 19 |
| 2 | 1.633 | 3.629 | 20.46 | 10.16 | 0 |
| 3 | 0.642 | 3.144 | 7.3 | 7.68 | 19 |
| 4 | 2.31 | 5.358 | 29.46 | 14.76 | 0 |
| 5 | 1.85 | 4.677 | 22.116 | 12.55 | 0 |
| 6 | 1.21 | 3.09 | 15.18 | 8.08 | 1 |
| 7 | 1.148 | 3.028 | 14.51 | 7.83 | 2 |
| 8 | 1.126 | 3.006 | 14.26 | 7.76 | 0 |
| 9 | 1.62 | 3.63 | 20.38 | 10.18 | 0 |
| 10 | 1.472 | 1.22 | 18.98 | 3.16 | 0 |

Table 5.1: Experiment Parking Places

This first table contains in a random order all the parking places results we will use in the following experiments. The number on the first column is just a reference to the specific parking place and that parking place will carry that number in all experiments in order to show how their ordering changes depending on the user preferences.

**Cost, Distance from parking to destination, Distance from user location to parking**

| No | Cost | Distance from destination(Km) | Distance from user location(km) | Score |
|---|---|---|---|---|
| 8 | 0 | 1.126 | 3.006 | 93.91 |
| 6 | 1 | 1.21 | 3.09 | 93.44 |
| 7 | 2 | 1.148 | 3.028 | 92.98 |
| 10 | 0 | 1.472 | 1.22 | 92.05 |
| 9 | 0 | 1.62 | 3.63 | 89.67 |
| 2 | 0 | 1.633 | 3.629 | 89.58 |
| 5 | 0 | 1.85 | 4.677 | 86.11 |
| 4 | 0 | 2.31 | 5.358 | 81.92 |
| 3 | 19 | 0.642 | 3.144 | 19.98 |
| 1 | 19 | 1.116 | 2.209 | 19.96 |

Table 5.2: Results for Cost, Destination Distance, User Distance

The two parking places that are much more expensive from the other have the lowest score as it was expected, with the exception of parking places 6 and 7 due to the fact that their cost difference is too low compare to the distance difference. As a result the second priority which is the distance from user location out weights the first priority which is cost.

**Time from parking to destination, Time from user location to parking, Cost**

| No | Time from destination | Time from user location | Cost | Score |
|---|---|---|---|---|
| 8 | 14.26 | 7.76 | 0 | 92.83 |
| 6 | 15.18 | 8.08 | 1 | 92.12 |
| 7 | 14.51 | 7.83 | 2 | 91.96 |
| 10 | 18.98 | 3.16 | 0 | 90.56 |
| 3 | 7.3 | 7.68 | 19 | 79.91 |
| 1 | 14.58 | 5.21 | 19 | 79.19 |
| 9 | 20.38 | 10.18 | 0 | 73.78 |
| 2 | 20.46 | 10.16 | 0 | 73.25 |
| 5 | 22.116 | 12.55 | 0 | 62.72 |
| 4 | 29.46 | 14.76 | 0 | 52.07 |

Table 5.3: Results for Destination Time, User Time, Cost

The score is relatively decreasing as the time from parking to destination increases with some exceptions like parking place 3 which drops down due to the large difference in cost.

**Distance from user location to parking, time from parking to destination, Distance from parking to destination**

| No | Distance from user location(km) | Time from destination | Distance from destination(Km) | Score |
|---|---|---|---|---|
| 3 | 3.144 | 7.3 | 0.642 | 94.41 |
| 1 | 2.209 | 14.58 | 1.116 | 93.05 |
| 8 | 3.006 | 14.26 | 1.126 | 92.40 |
| 7 | 3.028 | 14.51 | 1.148 | 92.21 |
| 6 | 3.09 | 15.18 | 1.21 | 91.64 |
| 10 | 1.22 | 18.98 | 1.472 | 89.78 |
| 9 | 3.63 | 20.38 | 1.62 | 84.60 |
| 2 | 3.639 | 20.46 | 1.633 | 84.56 |

| | | | | |
|---|---|---|---|---|
| 5 | 4.677 | 22.116 | 1.85 | 76.23 |
| 4 | 5.358 | 29.46 | 2.31 | 72.06 |

Table 5.4: Results for User Distance, Destination Time, Destination Distance

In this table you can notice that parking place 3 although it has a larger distance from the user location than most of the other parking places it has the higher score as it has much less time from parking to destination.

**Time from user location to parking, Cost, Distance from parking to destination**

| No | Time from user location | Cost | Distance from destination(Km) | Score |
|---|---|---|---|---|
| 10 | 3.16 | 0 | 1.472 | 93.37 |
| 8 | 7.76 | 0 | 1.126 | 93.28 |
| 6 | 8.08 | 1 | 1.21 | 92.76 |
| 7 | 7.83 | 2 | 1.148 | 92.37 |
| 2 | 10.16 | 0 | 1.633 | 86.61 |
| 9 | 10.18 | 0 | 1.62 | 86.61 |
| 1 | 5.21 | 19 | 1.116 | 79.74 |
| 3 | 7.68 | 19 | 0.642 | 79.35 |
| 5 | 12.55 | 0 | 1.85 | 57.80 |
| 4 | 14.76 | 0 | 2.31 | 52.73 |

Table 5.5: Results for User Time, Cost, Destination Distance

Parking places 1 and 3 fall down due to their high cost, otherwise the score normally comes down as the time from user location increases.

**Distance from parking to destination, distance from user location to parking, cost**

| No | Distance from destination | Distance from user location(km) | Cost | Score |
|---|---|---|---|---|
| 8 | 1.126 | 3.006 | 0 | 93.47 |
| 6 | 1.21 | 3.09 | 1 | 92.94 |
| 7 | 1.14 | 3.028 | 2 | 92.66 |
| 10 | 1.472 | 1.22 | 0 | 90.59 |
| 9 | 1.62 | 3.63 | 0 | 82.40 |
| 2 | 1.633 | 3.629 | 0 | 82.88 |
| 3 | 0.642 | 3.144 | 19 | 79.92 |
| 1 | 1.116 | 2.209 | 19 | 79.24 |
| 5 | 1.85 | 4.677 | 0 | 62.13 |
| 4 | 2.31 | 5.358 | 0 | 52.11 |

Table 5.6: Results for Destination Distance, User Distance, Cost

We can conclude from all the tables that changing the preferences ordering does affect the results as different scores are calculated for each parking place with each experiment. Also we can proof that the second and third priority preferences can also affect that score especially if they have big difference compare to the first priority.

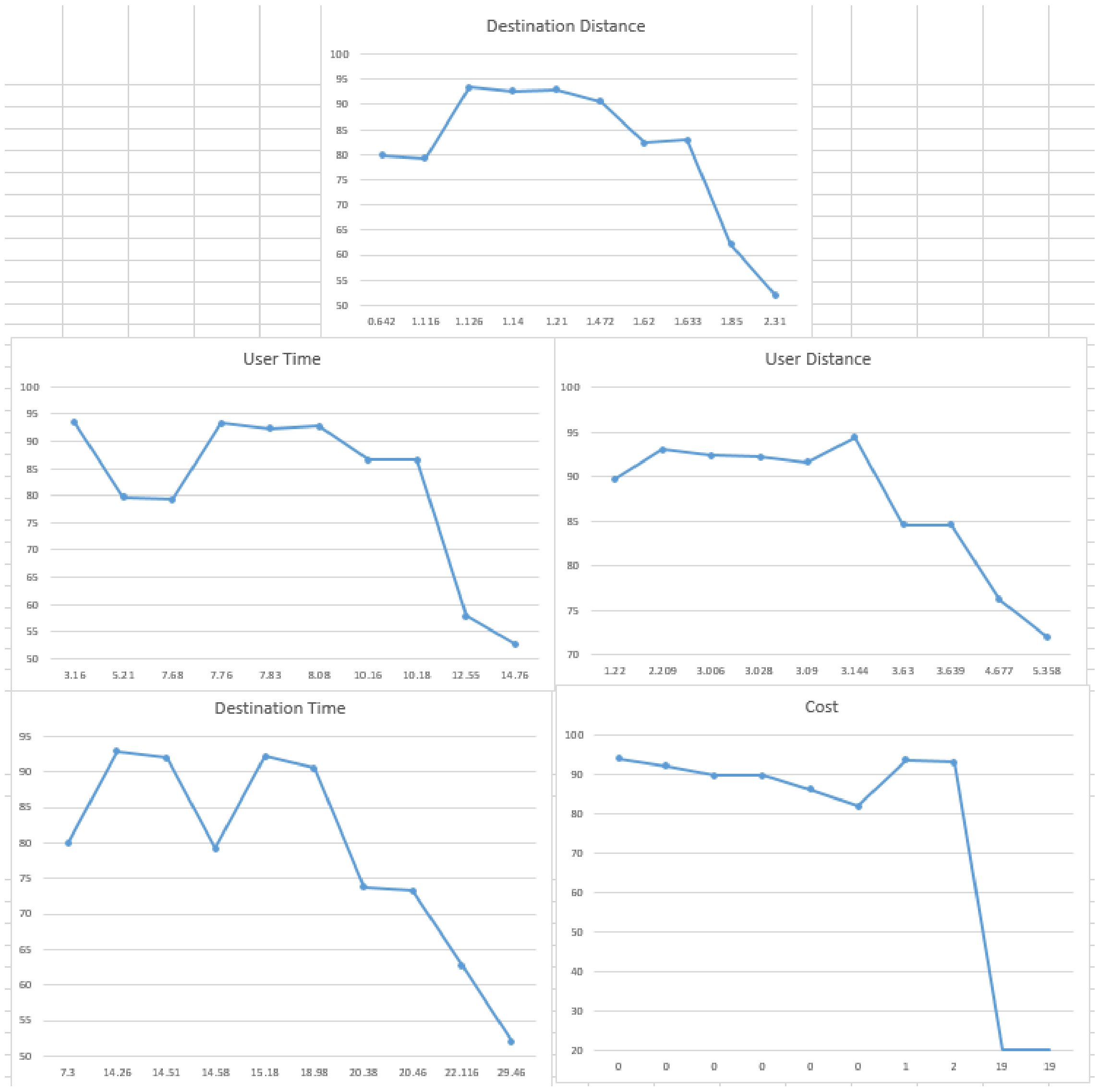


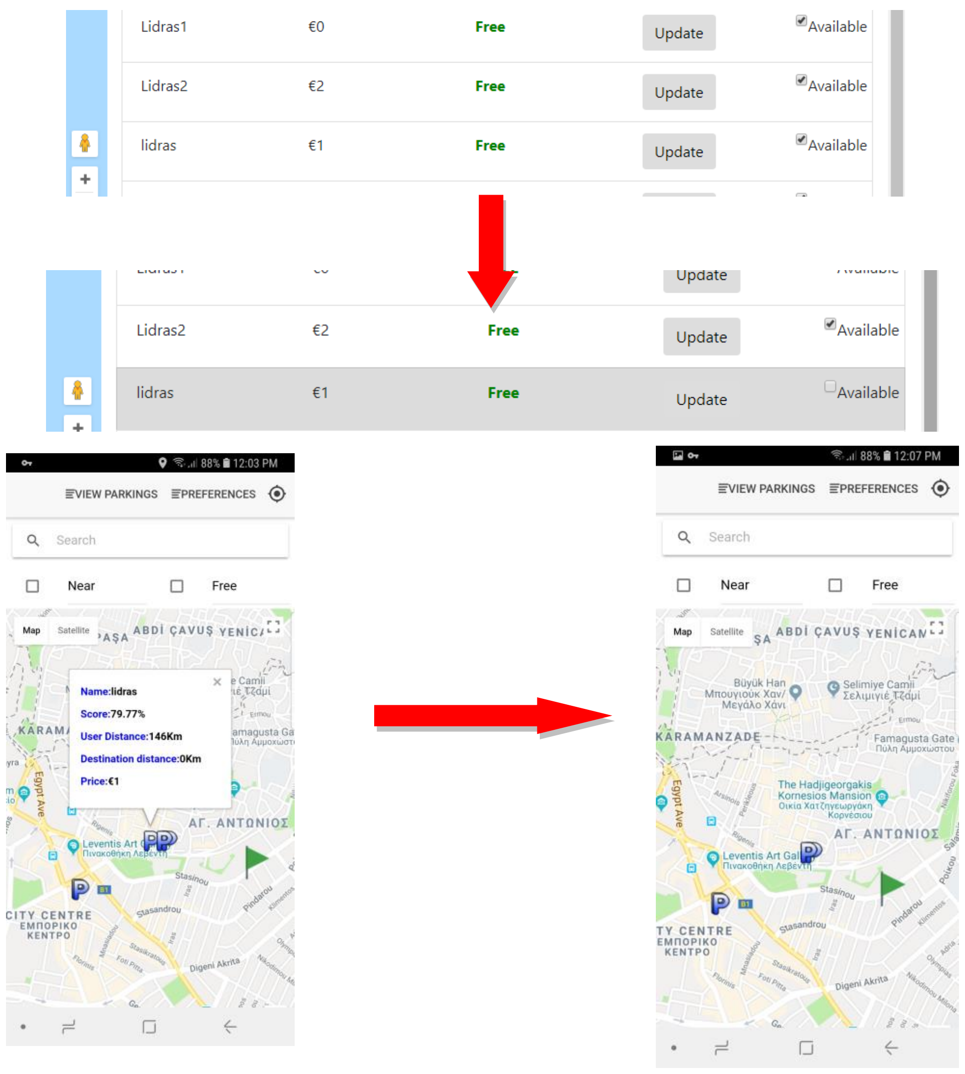
Figure 5.10: Parking Results Experiment Graphs

Finally from the experiment tables we extracted the charts that show the relationship between the first priority metric and the score. The main conclusion here is that although they have some fluctuations due to the second and third priority preference, the scores go down in the long run as the metric number goes bigger.

### 5.2.2 Scenarios

Some scenarios will be executed in order to examine the whole system that it functions properly and delivers the desired results. The scenarios will test the mobile application and website as well as the communication between them through the server.

A parking provider for some reason wants make a parking place he owns unavailable. That means that the parking place will no longer be available in the system and it would not come back as a result when a user executes a search for parking places through the mobile application. In the example bellow we show a case where a parking provider makes a parking place unavailable and the consequences it has when a user executes a parking search from the mobile application.



| | | | | |
|---|---|---|---|---|
| Lidras1 | €0 | Free | Update | ✓ Available |
| Lidras2 | €2 | Free | Update | ✓ Available |
| lidras | €1 | Free | Update | ✓ Available |

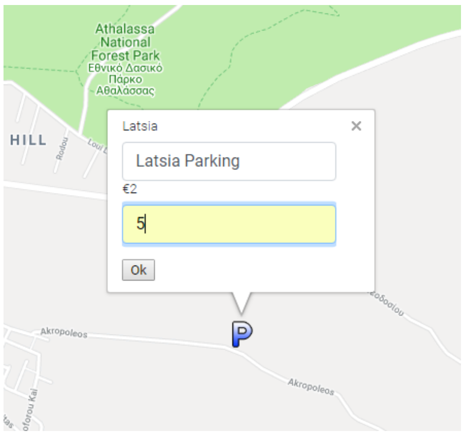
| | | | | |
|---|---|---|---|---|
| Lidras2 | €2 | Free | Update | ✓ Available |
| lidras | €1 | Free | Update | ☐ Available |

Name:lidras
Score:79.77%
User Distance:146Km
Destination distance:0Km
Price:€1

Figure 5.11: Availability Scenario Screenshot

You can see in the figure the effects on the mobile application when a user searches for parking places before and after the parking provider makes it unavailable. The parking place with the name “Lidras” no longer appears when the user searches for parking places after the provider made it unavailable through the website.

Similarly, the parking provider wants to change the name and price of one of his own parking places. Bellow you can again see the effects of it on the mobile application.

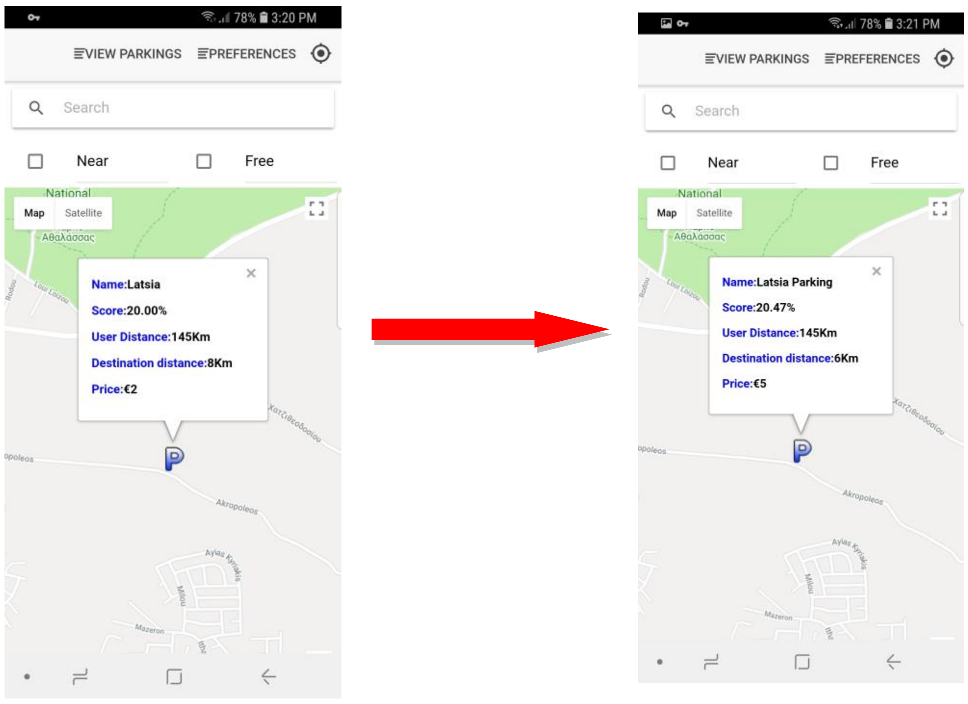

Latsia

Latsia Parking

€2

5

Figure 5.12: Update parking place box screenshot

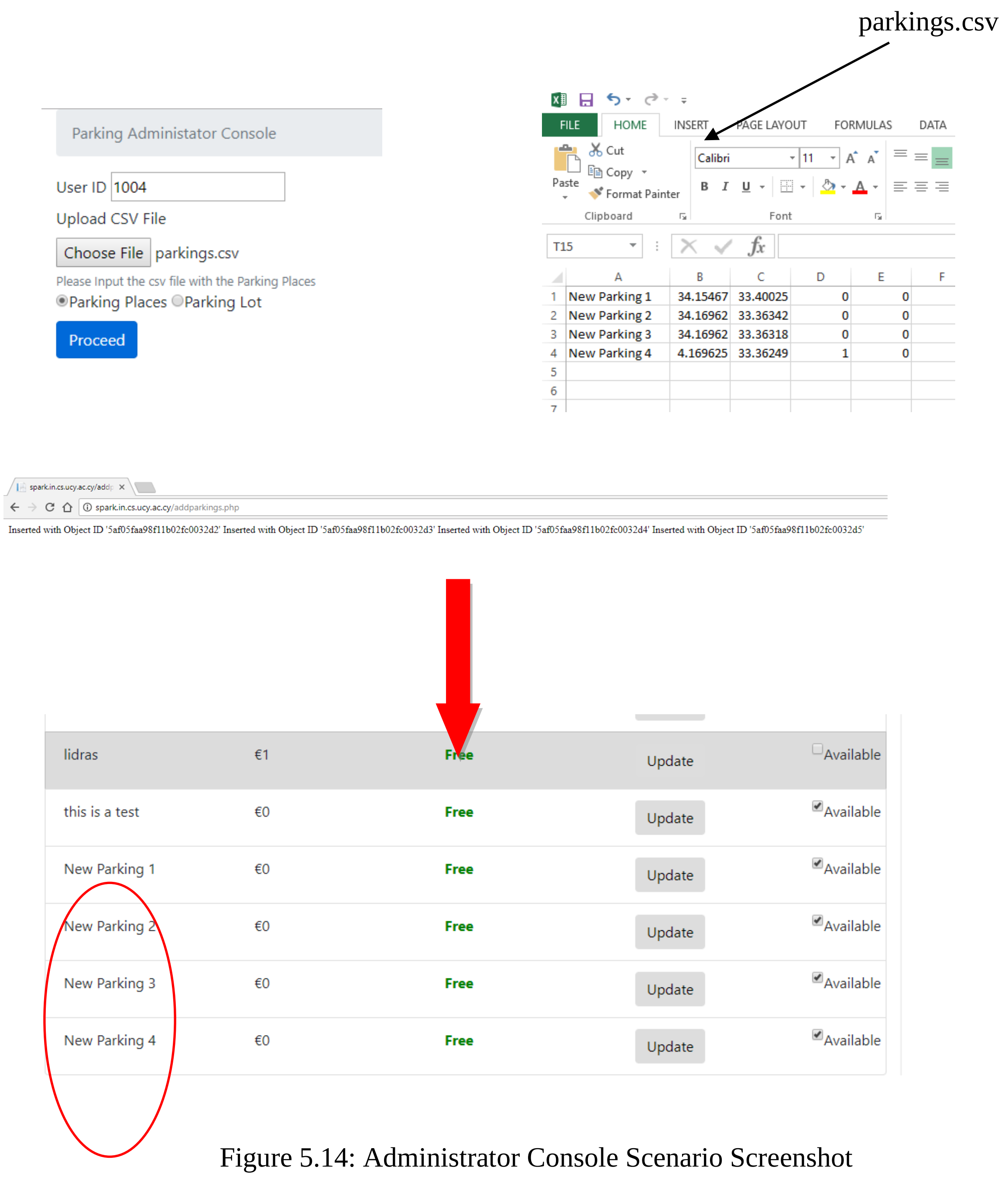

Name:Latsia
Score:20.00%
User Distance:145Km
Destination distance:8Km
Price:€2

Name:Latsia Parking
Score:20.47%
User Distance:145Km
Destination distance:6Km
Price:€5

Figure 5.13: Update parking place Scenario Screenshot

You can see from the mobile application that both the name and price of the selected parking place is changed. Please note that to see the different you execute a parking search two times, one before the change at the provider website and one after.

Now we will test how the parking administrator can put new parking places into the system through the administrator console. In this example an existing parking provider wants to insert 3 new parking places.



User ID 1004
Upload CSV File
Choose File parkings.csv
Please Input the csv file with the Parking Places
[✓ Parking Places] Parking Lot

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | New Parking 1 | 34.15467 | 33.40025 | 0 | 0 | |
| 2 | New Parking 2 | 34.16962 | 33.36342 | 0 | 0 | |
| 3 | New Parking 3 | 34.16962 | 33.36318 | 0 | 0 | |
| 4 | New Parking 4 | 4.169625 | 33.36249 | 1 | 0 | |
| 5 | | | | | | |
| 6 | | | | | | |
| 7 | | | | | | |

spark.in.cs.ucy.ac.cy/addparkings.php

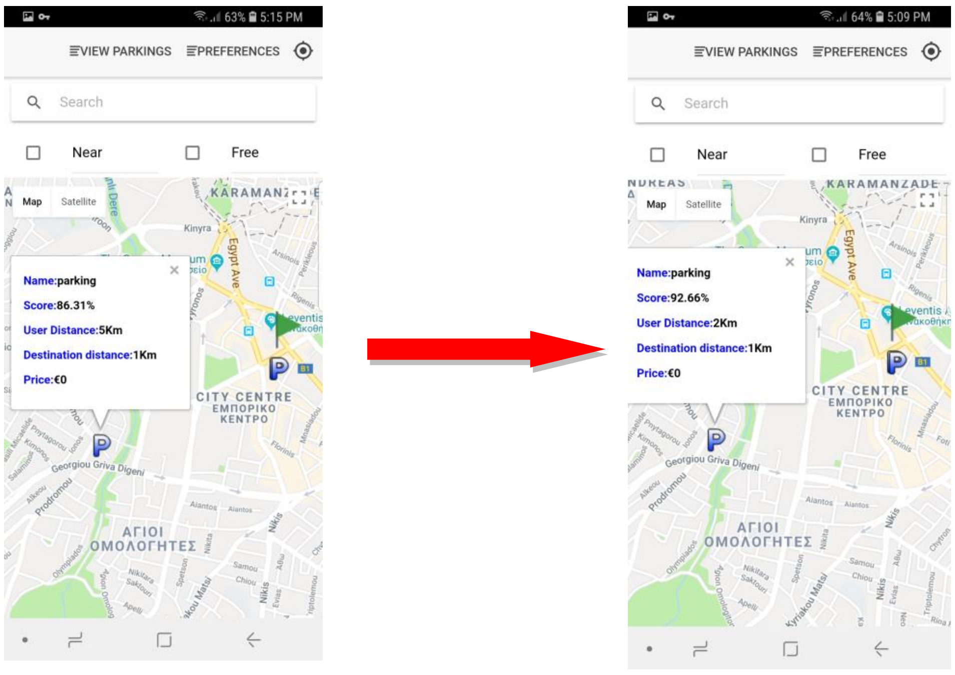
```
Inserted with Object ID '5af05faa98f11b02fc0032d2' Inserted with Object ID '5af05faa98f11b02fc0032d3' Inserted with Object ID '5af05faa98f11b02fc0032d4' Inserted with Object ID '5af05faa98f11b02fc0032d5'
```

| | | | | |
|---|---|---|---|---|
| lidras | €1 | Free | Update | ☐ Available |
| this is a test | €0 | Free | Update | ✓ Available |
| New Parking 1 | €0 | Free | Update | ✓ Available |
| New Parking 2 | €0 | Free | Update | ✓ Available |
| New Parking 3 | €0 | Free | Update | ✓ Available |
| New Parking 4 | €0 | Free | Update | ✓ Available |

Figure 5.14: Administrator Console Scenario Screenshot

As you can see the parking administrator uses the administrator console to upload a CSV file containing the new parking places information. You can see the CSV file on the top right. Its format is predetermined and has the name, coordinates and two true or false values the first specifying if the parking place is for disabled persons and the second if the parking is free. The system will then return back a message to the user that the new objects have been inserted successfully into the database. You can see the results of this thought the parking provider website.

One last small scenario, in the case that a mobile application user wants to set a different starting point other than his current location he can very easily change it through the mobile application. For this experiment we execute the same search on the same destination before and after the user changes his current location.



Name:parking
Score:86.31%
User Distance:5Km
Destination distance:1Km
Price:€0

Name:parking
Score:92.66%
User Distance:2Km
Destination distance:1Km
Price:€0

Figure 5.15: Different Current Location Scenario Screenshot

In the figure you can see the same parking place information before and after the change. You can notice that the distance of the user from the same parking has changed from 5km to 3km due to the fact that the user set his current location on a place that is nearer to the parking place.

# Chapter 6

***6 CONCLUTIONS***

## 6.1 Summary and Conclusions

The problem of Smart Parking is a vital one that affects our everyday lives both as drivers and as parking providers. Many applications are available out there that solve the problem in many and different ways. Through this thesis a Smart Parking system was developed that not only serves the everyday users in their search for available parking places but also the parking providers which they want an easy way to manage their own parking places.

For the normal users which are the drivers a mobile application was developed that enabled the easy search for parking places based on their preferences. Because it is on mobile the users could use it anywhere whether they were on the road or before they would start driving towards their destination. The variety of preferences provided by the application allowed them to customize their search based on what it is more important to them and produce the results in a different order every time. They could also if they so choose set a different starting location as their current position to execute a parking search way in advance. In many cases the user would want to just look into the free parking places or the ones nearest to his destination. The application by providing filtering of results avoids flooding the user with unnecessary information. Finally the interaction with the mobile application is really easy because of the options to both click on the map and execute a textual search for destinations.

The website is developed in a way that is easily used by the parking providers and the administrators of the system. The interaction of the mobile application and website was almost instantaneous. As soon as the parking provider made a change through the website that change had effects on the mobile application’s search results. The provider could easily change his parking place’s name, price and make them available or unavailable accordingly. The system administrator could with just a CSV file containing the parking places add the new elements into the database. To view the user's information he would have to manually view the Users database records but that wasn't a problem since as an administrator of the system he had access to everything.

In general the system's components communicated well with each other and produced the desired results. The system was found to be really useful for both everyday drivers and parking providers but for sure there is room for improvement and extra functionality in the future.

## 6.2 Future Work

The parking system as a whole succeeded in deploying most of the core functionalities of such system but as time was limited there is always room for improvement and development.

The mobile application was well customized into user's needs by using five different preferences for the user but as there is a built in login feature there is much room for a more personalized application. In the future the application could contain popular features such as giving the ability to the user to save its own favorite parking places or moreover collect search history that could filter his search results. In addition the user could save extra information about him such as what car he drives or where he is parked at the moment.

At the moment the mobile application does not provide any feature for reserving and paying the parking place in advance. For that reason a billing system can be developed which will give the option to the user to pay with his credit card easy and safe. To add to this, the prices are also fixed at the moment, which is a problem if a parking provider wants to change his price with the time the car stayed at the parking place. As such the application could provide flexible time price information and charge the user accordingly either after he records the time he parked or before.

As it is the website has only very core functionality for the parking providers so in the future it can provide them with different statistics about their parking places. At the moment the system already records some information but it is not exploited yet.

Statistics may include which parking places are more popular and that could also be used by the mobile application to provide the normal users with that information. Moreover the website lacks the ability for the parking provider or the system administrators to delete a parking place. The only way to work around this problem at the moment is to make the parking place unavailable.

The mobile application only has the two main filters. Displaying only the free parking places and the ones that are less than 1km away from the destination. There is room for additional filters for the user such as the time duration both from the parking place to destination and from current position to parking place. Those times and distances can be more flexible and not limited to just one amount. There are also more specific filters that can be deployed given the necessary information from the parking providers such as displaying only parking places that are shaded, secured, on hard ground or any other given information. For that to happen the website must also give the ability to the parking providers to upload any other relevant information they want about their parking places.

Finally it would be a good idea for the mobile application to have an announcements feature which can notify users if there is an event at a particular location. The events maybe cases like when a road is closed for construction or some other reason and the parking places around it become unavailable. Also maybe there are certain events that cause particular roads to parking places to become full with traffic and as such recommend the user to avoid them.

# Bibliography

[1] Current smart parking technologies information technology essay Section 2.2 [Online] Available: https://www.uniassignment.com/essay-samples/information-technology/current-smart-parking-technologies-information-technology-essay.php

[2] Internet of things-Definition [Online] Available: https://www.techopedia.com/definition/28247/internet-of-things-iot

[3] Definition-Smart City [Online] Available: http://internetofthingsagenda.techtarget.com/definition/smart-city

[4] Definition-Smart Parking [Online] Available: https://encyclopedia2.thefreedictionary.com/smart+parking

[5] Definition- API [Online] Available: https://techterms.com/definition/api

[6] Definition-Fuzzy logic [Online] Available: http://whatis.techtarget.com/definition/fuzzy-logic

[7] ParkGuru on Google play [Online] Available: https://play.google.com/store/apps/details?id=com.parkguru&hl=en

[8] ParkAround on Google play [Online] Available: https://play.google.com/store/apps/details?id=com.parkingd2&hl=en

[9] JustPark on Google play [Online] Available: https://play.google.com/store/apps/details?id=com.justpark.jp&hl=en

[10] ParkoPedia on Google play [Online] Available: https://play.google.com/store/apps/details?id=com.parkopedia&hl=en

[11] ParkEasy on Google play [Online] Available: https://play.google.com/store/apps/details?id=com.pixelbyte.parkeasy_android&hl=en

[12] Master of Science Thesis on Design and Implementation of a smart parking system and application based on fuzzy logic algorithm, Ioanna Wyrazik University of Cyprus May 2016

[13] Ionic framework official site and documentation [Online] Available: https://ionicframework.com/

[14] Google maps JavaScript API documentation [Online] Available: https://developers.google.com/maps/documentation/javascript/tutorial

[15] Google maps Place API documentation [Online] Available: https://developers.google.com/places/web-service/intro