An Automated Parking System

A dissertation submitted for the Degree of Master of Information Technology

T. R. Singhara

University of Colombo School of Computing

2020

Declaration

The thesis is my original work and has not been submitted previously for a degree at this or any other university/institute.

To the best of my knowledge it does not contain any material published or written by another person, except as acknowledged in the text.

Student Name: Thamali Randula Singhara

Registration Number: 2017/MIT/077

Index Number: 17550773

____

Signature: Date: 19th November 2020

This is to certify that this thesis is based on the work of Mr. Malik Silva under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Certified by:

Supervisor Name: Mr. K.P.M. K Silva

____

Signature: Date:

Abstract

Drivers become frustrated and got a challenging task when finding out a free and suitable parking area during rush hours. Traffic congestions will also get increased when drivers make loops to move through the road to find out a free parking area near the destination. This will cause accidents and waste the valuable time of drivers. To address all these issues developing a parking application to find out the nearby parking location areas will become a good solution.

In Sri Lanka, now a days the government will allow drivers to park the vehicle on open street areas where the payment can be made on a parking meter machine which is monitored by a parking warden. This project will give an application solution to find out available parking slots on open parking areas near Colombo city limits. The process of finding free parking areas will be easy and fast for drivers by using this application.

The aim of this project is to develop a web-based application and a mobile application, respectively for admin and for the driver and parking warden. The admin has the ability to manage the parking locations and wardens by assigning particular locations for the wardens through the web application. At the same time mobile application gives two user views for parking warden and driver. Parking warden can update the availability or unavailability of slots which are assigned for the warden. According to the availability of slot, they will show up in the driver's application. Once driver opens the application, current location of the driver's vehicle is tracked and available parking locations will appears on the map. Driver can get an idea about the details of parking location by selecting a particular location which gives location name and the number of available parking slots in that location.

As a user can get a clear idea about how many slots are free out of all the available slots, since in a situation like moving into a location where having 2 free slots out of 5 available slots has a low probability of change getting lost of the free slot. As a solution for this, the driver's application provides a function to view how many slots are free out of all available slots.

As a future work location update will be automated as driver has to book a particular parking slot before enter to the location. And when a situation like a vehicle accidently park in a free slot without knowing, then it take the geo location of the parking slot and using sensors it will automatically update the slot availability in the system. Then the drivers who use the application will get more accurate results by automatically updating the free available parking slots in a particular location.

Acknowledgement

This would not have been successful without the support of many people. This is a good opportunity to give sincere thanks of the people who gave their maximum support to make this effort in success.

I gratefully acknowledge Mr. K.P.M. K Silva, supervisor of the project, for his supervision, advice, and guidance from the very early stage of this project. Above all and the most needed, he provided me unflinching encouragement and support in various ways. His truly scientist intuition has made him as a constant oasis of ideas and passions in teaching, which exceptionally inspire and enrich our growth as students, and as professionals want to be.

And also, I would like to express my sincere gratitude to University of Colombo School of Computing for offering this master’s degree program to individuals like myself whom seeking for knowledge in the IT industry, and also to all the staff members who guide students from the beginning of the program.

Last, but not least, I express my heartfelt gratitude to my parents, sisters and my office colleagues and friend who were with me during the project period by providing advice, guidance and valuable support to make this effort success.

Table of Contents

List of Figures

List of Tables

List of Acronyms

API	Application Programming Interface
DB	Database
GPS	Global Positioning System
HTTP	HyperText Transfer Protocol
IDE	Integrated Development Environment
SDK	Software Development Kit
UI	User Interface

1. Introduction

1.1 Motivation

Due to increase of individual personal vehicle usage and urbanization, the vehicle parking is an important issue and its necessity is increasing day by day. Now a days mainly because of people tending to use vehicle as individuals. Upon considering this serious issue in Sri Lanka, this topic could be taken and giving a solution for this is the main motivation of this project which is a good enough to find a free parking space in metropolitan areas like Colombo, especially during rush hours. One of the best solutions available for this serious issue is developing an automated parking system which reduces the traffic congestion and reduces the time of finding a vacant parking slot where the application should give the information about occupancy status of free parking slots [1].

Currently Sri Lankan drivers follow the common manual method to finding a free parking space. Drivers find a space on the available area through luck. Traffic congestions will increase on rush hours Due to limited number of paid parking areas on busy cities the drivers are compelled to park the vehicle in open streets.

The Sri Lankan government has allowed drivers to park in open on-street parking areas near Colombo city limits. So, this application mainly focuses on the open street parking locations.

The prime aim of this project is to develop an application which consists of a web application for admin and a mobile based client for parking warden and for the driver. Admin is having the functionality of managing parking locations and wardens along with assigning the locations for parking wardens. Updating the slot availability or unavailability is the primary job of the parking warden where the app allows the functionality to change the availability for all the assigned parking slots. By getting the current geo location of the driver's vehicle, the mobile application provides with all the nearest available parking slots.

1.2 Aims and Objectives of the Project

The aim of this project is to develop an automated parking system to address the mentioned problems. This system will consist of admin module which allows to manage locations and parking wardens. It also includes a mobile based client for drivers and parking wardens which would be used to allow parking wardens to update availability and unavailability of parking slots and allow drivers to search for and locate nearby available parking slots [2].

The project objectives for an automated parking system are as follows;

- Better parking experience due to the saving of fuel and time.
- Develop for street parking specially near cities of Colombo area.
- Reduce traffic congestion in busy cities and minimize driver's time using an intelligent, user friendly automated parking application.
- Help the drivers to find out the nearby and available parking slots.
- Giving detailed information of the availability of number of slots.
- Tracking the vehicle details so the drivers cannot cheat out for paying the parking fees.
- Making a set job for parking warden to manage the availability of parking slots and track vehicle details.
- Add new locations, edit, delete and disable existing location details by the admin.
- Add new parking wardens, edit, delete and assign/ reassign locations for parking wardens by the admin.
- Guide through google maps where guides driver to the available parking slot.
- Be Scalable, robust and reliable.

1.3 Scope

This system is developed for an Android smart phone and supported by a back-end web service.

The scope of the system is defined as follows:

- Find out the benefits of developing an application for assisting drivers the need of a free parking slot.
- Investigate different requirements and options to enhance the driver’s experience and efficiency to find out a suitable free parking slot.
- Learn the development process on backend web services and smart phones using latest technologies and various software development techniques.
- Develop a mobile application which helps driver to find out a vacant parking spot quickly and easily by giving clear and simple directions.
- Develop the option to search available slots in nearby areas.
- Develop another mobile application view for parking warden to update the availability and unavailability of parking slots and track vehicles via simple and comprehensive interface.
- Develop a web-based application for admin user to manage locations and parking wardens.
- Use location-based technology in the application.
- Evaluate and review the strengths and weaknesses of the project, seek and identify for further improvements.

2. Background

2.1 Requirement Analysis

Once started working on the requirements for the implementation of the project, the idea was to develop a web service that can retrieve information of the availability status of the parking slot and post that information to the mobile application and also send information to the database through a web service URL and update those changes into the system [3].

Following are the technical requirements to develop an automated parking system.

Software Requirement

Operating system : Windows 10

IDE : IntelliJ IDEA

Web service : RESTful web services

Frameworks and APIs : Java Spring boot and JSON

Frontend : REACT, JavaScript

Database : MySQL

Browsers : Chrome, Firefox, IE

Hardware Requirement

Processor : Intel® Core™ i5-10210U CPU @ 1.60GHz 2.11GHz

RAM : 8 GB

Functional Requirements

The proposed system comprises of a web application for parking area admins, a mobile application for parking wardens and a mobile application for users.

Each of their requirements are stated below.

- Mobile application for drivers

  Login to the parking app
  Search a parking area that need for an available parking spot

Find out all available parking slots
Navigate into google map directions once find out a suitable location

- Mobile application for parking wardens

  Register into the application
  Login after successful registration
  Manage parking slots (Mark availability and unavailability of slots)
  Check slot availability
  Track vehicle details

- Web application for parking admin

  Define new parking locations and specify number of parking lots
  Modify data of existing parking areas
  Remove existing parking areas
  View the data of all registered parking areas

Non functional requirements

The non-functional requirements that should be considered throughout the application is as follows [4].

- The server should be able to handle different concurrent user requests from different users

- The application should be user friendly

- Application should be able to handle high traffic due to huge number of users may login at the same time.

- The android application provides high accuracy and availability on finding the locations.

The following figure shows the system overview using a use case diagram

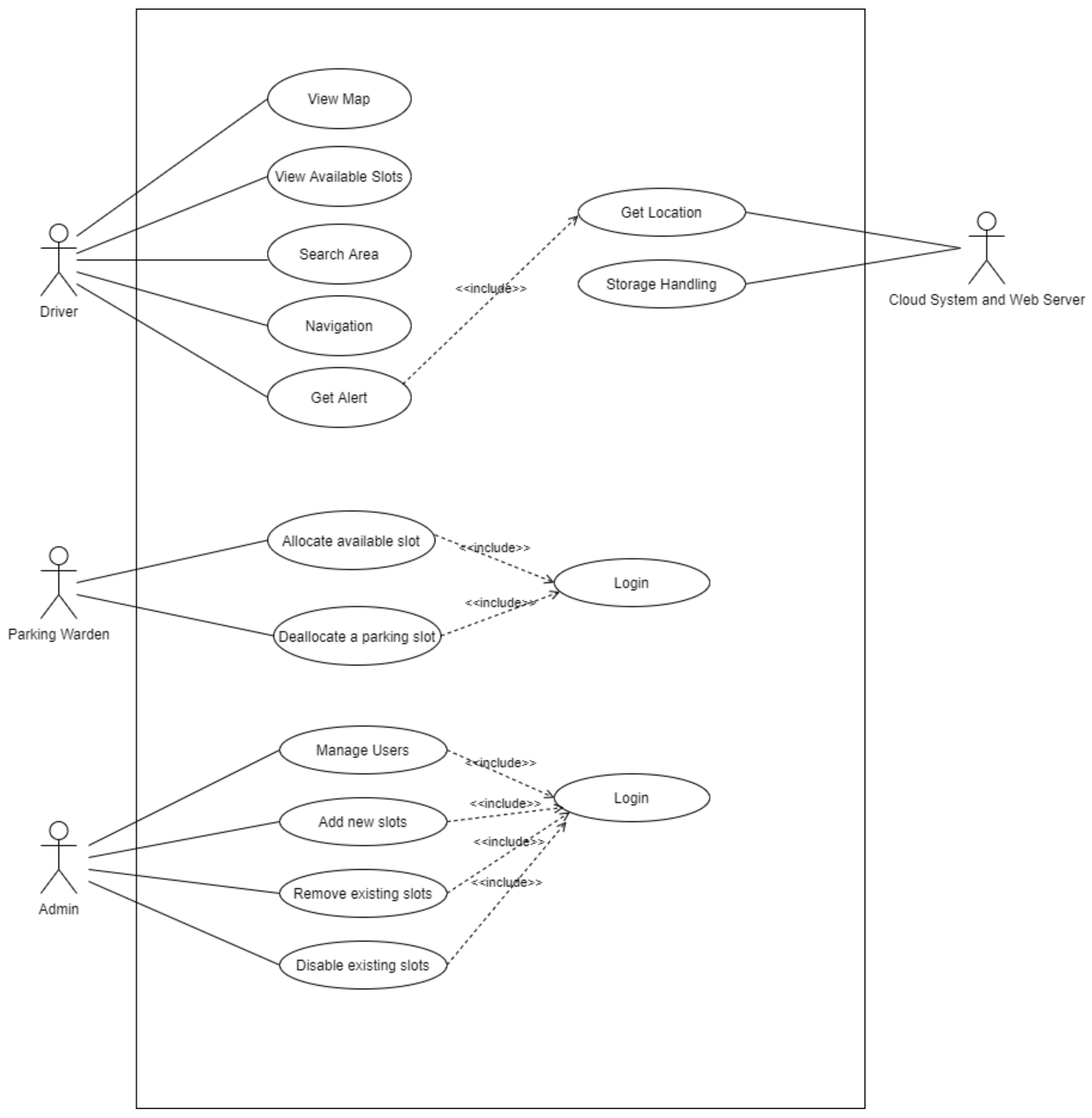


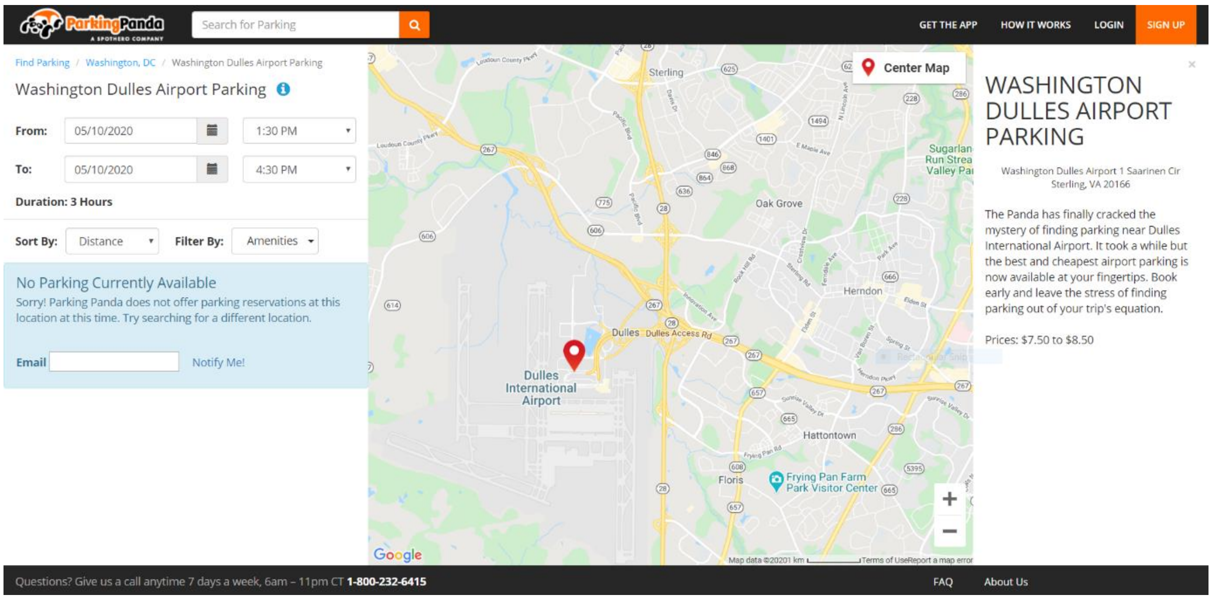
Figure 1 : Use Case Diagram

2.2 Review of similar systems

Following displays the apps used in world wide which help to find a parking spot.

Parking Panda

Possibility of Finding a guaranteed parking spot is the main advantage of this service. When the user searches for a parking space, as displayed in Figure 1, the results are displayed in a map with list format on both the website and the mobile. Using this system drivers will be able to reserve a parking space in a more convenient way. Until the driver find a specific location user has to spend more time to navigate and compare between different options. Real time prices and the distance shows the results of the application; the application doesn't allow sorting by price or by any other attribute. That is fixed [5].



Find Parking / Washington, DC / Washington Dulles Airport Parking

Washington Dulles Airport Parking

From: 05/10/2020 1:30 PM

To: 05/10/2020 4:30 PM

Duration: 3 Hours

Sort By: Distance Filter By: Amenities

No Parking Currently Available

Sorry! Parking Panda does not offer parking reservations at this location at this time. Try searching for a different location.

Email Notify Me!

WASHINGTON DULLES AIRPORT PARKING

Washington Dulles Airport 1 Saarinen Cir
Sterling, VA 20166

The Panda has finally cracked the mystery of finding parking near Dulles International Airport. It took a while but the best and cheapest airport parking is now available at your fingertips. Book early and leave the stress of finding parking out of your trip's equation.

Prices: $7.50 to $8.50

Questions? Give us a call anytime 7 days a week, 6am – 11pm CT 1-800-232-6415

Figure 2 : Parking Panda map

Spot Hero

Spot Hero is another informative parking reservation system with a similar concept, where users can find a parking spot by address and get the results on a map. the left-hand side of the parking panda application, they have a filter to prioritize the list of results. Although this feature gives more focus to the user, it can only filter by price or by distance from the searched address. Figure 2 shows Spot Hero's map and list view [6] where the other options also can be viewed that user might simply took to the map.

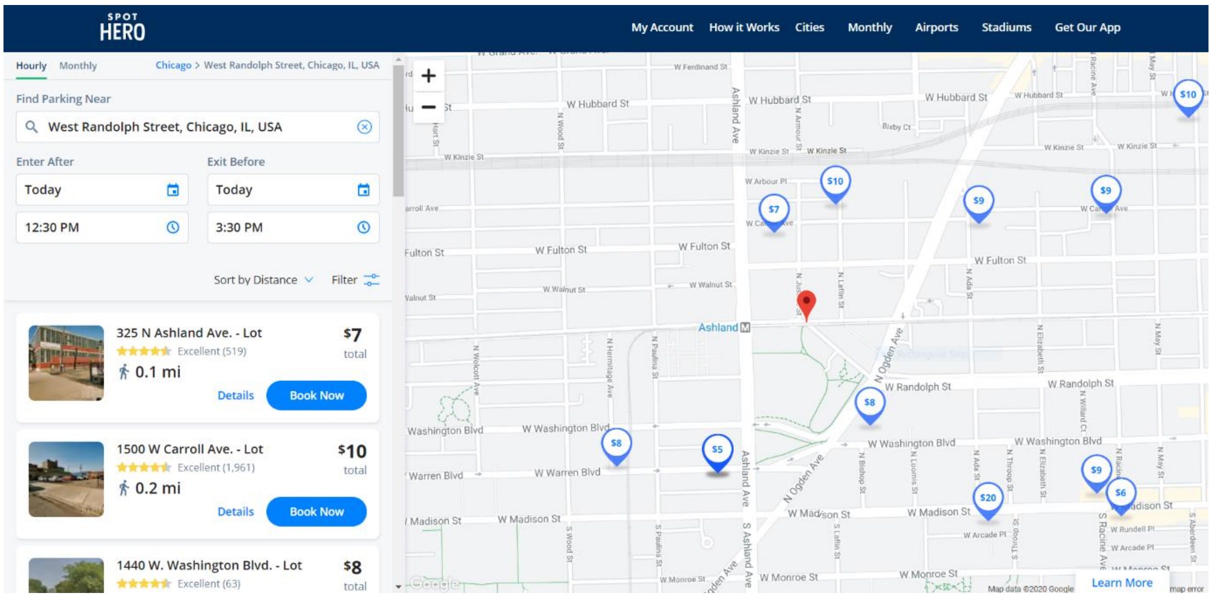


Figure 3 : Spot Hero map

Best Parking

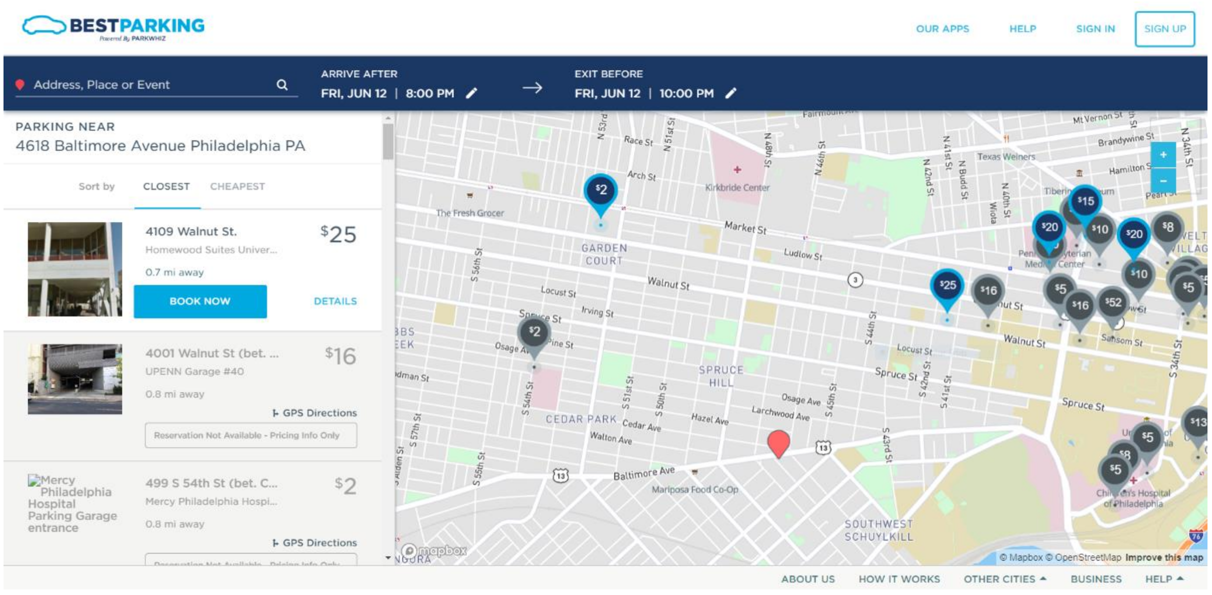


Figure 4 : Best Parking map

Best Parking is an online-based parking search engine. By using the online web application or the mobile application drivers can use the system which provides rates with hourly, daily, weekly, and monthly. The application can search for parking by using current location, specified address, cross street, or attraction site. Users have to select from the drop down lists the city, location, parking type, and duration of the parking. The available parking locations are appeared with priced pins in the map as shown in Figure 3. By clicking on a particular pin point, it gives the information about the location where the calculations are accurate and satisfy the user’s need. In addition, the results are informative and do not reserve a parking spot. Furthermore, clicking on drop down lists and choosing arrival and departure [7].

ParkMobile

The work procedures of this app are basically divided into three steps. First, look for the parking signs or stickers at the place users want to park later. Second, once users are registered, use the app to enter in the zone number listed on the parking signs found in previous step to start a parking session. The final step includes reserving the selected parking spot and paying the parking fees. After the reservation processes of parking have finished, and users will receive a receipt as their parking confirmation. Also, one feature of this app is that users can open the parking notification which will remind them 15 minutes before their parking session is set to expire [8].

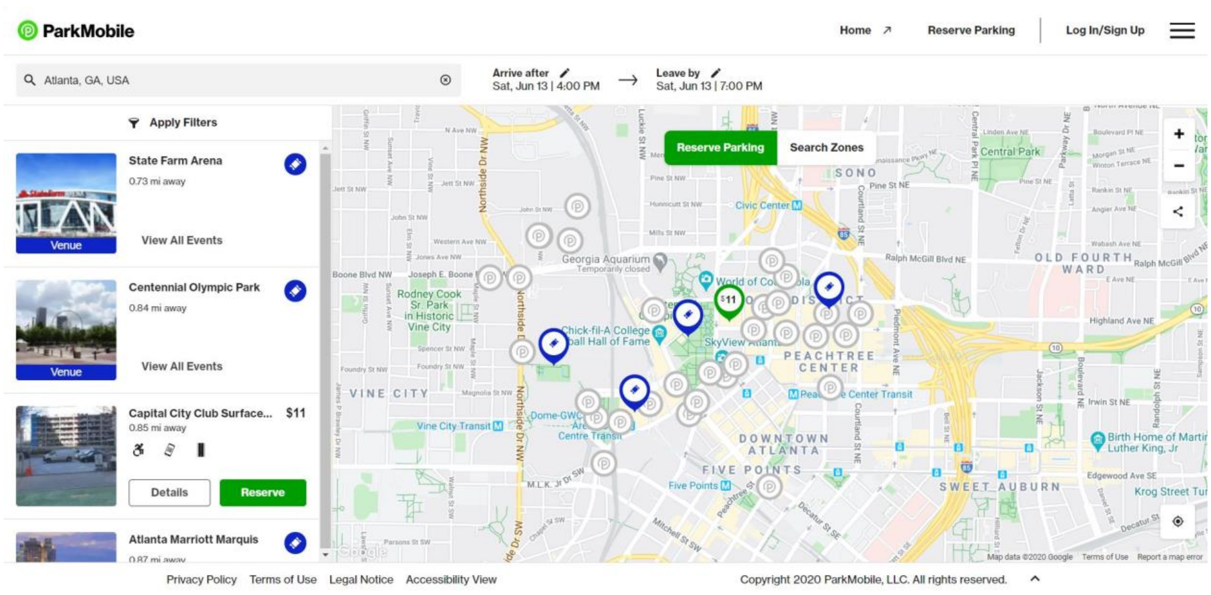


Figure 5 : ParkMobile map

2.3 Comparison of alternative tools and technologies

Smart parking technology	Suitability on open parking lots	Strength	Drawback
Active/Passive Infrared sensor	✗	Provides parking occupancy status	Due to varying environmental conditions, it is not suitable for open parking lot.
Ultrasonic sensor	✗	Provides parking occupancy status	Due to varying environmental conditions, it is not suitable for open parking lot.
Inductive loop detectors	✓	Facilitates in getting the count of vehicles in a parking lot	Not suitable for individual parking occupancy status.
Parking guidance systems	✓	Guide driver to a parking lot based on count of available parking spaces	Do not solve congestion while cruising for parking space
Radio frequency tags	✗	Facilitates in authorizing vehicles in a parking lot.	Due to varying environmental conditions, it is not suitable for open parking lot.
GPS	✓	Provides navigational directions to the parking space	Does not provide parking occupancy status and is subjective to errors
Machine vision	✓	Provides parking occupancy status or facilitates in authorizing vehicles	Challenges with low lightning conditions
Multi-agent systems	✓	Provides parking occupancy status	Challenges will be based on the technology used
Neural Networks	✓	Facilitates efficient parking occupancy detection	Is used only as a supporting tool in parking occupancy detection
Fuzzy logic	✓	Based on historical data gives the parking availability status	Do not provide real time parking occupancy

Table 1 : Existing smart parking technologies

3. Methodology

3.1 Description of Design

This application provides by integrating user-friendly features in it that can help to navigate through the crowded parking places and find a parking space easily. This application is developed using 3 components as,

- Driver Panel
- Parking Warden Panel
- Admin Panel

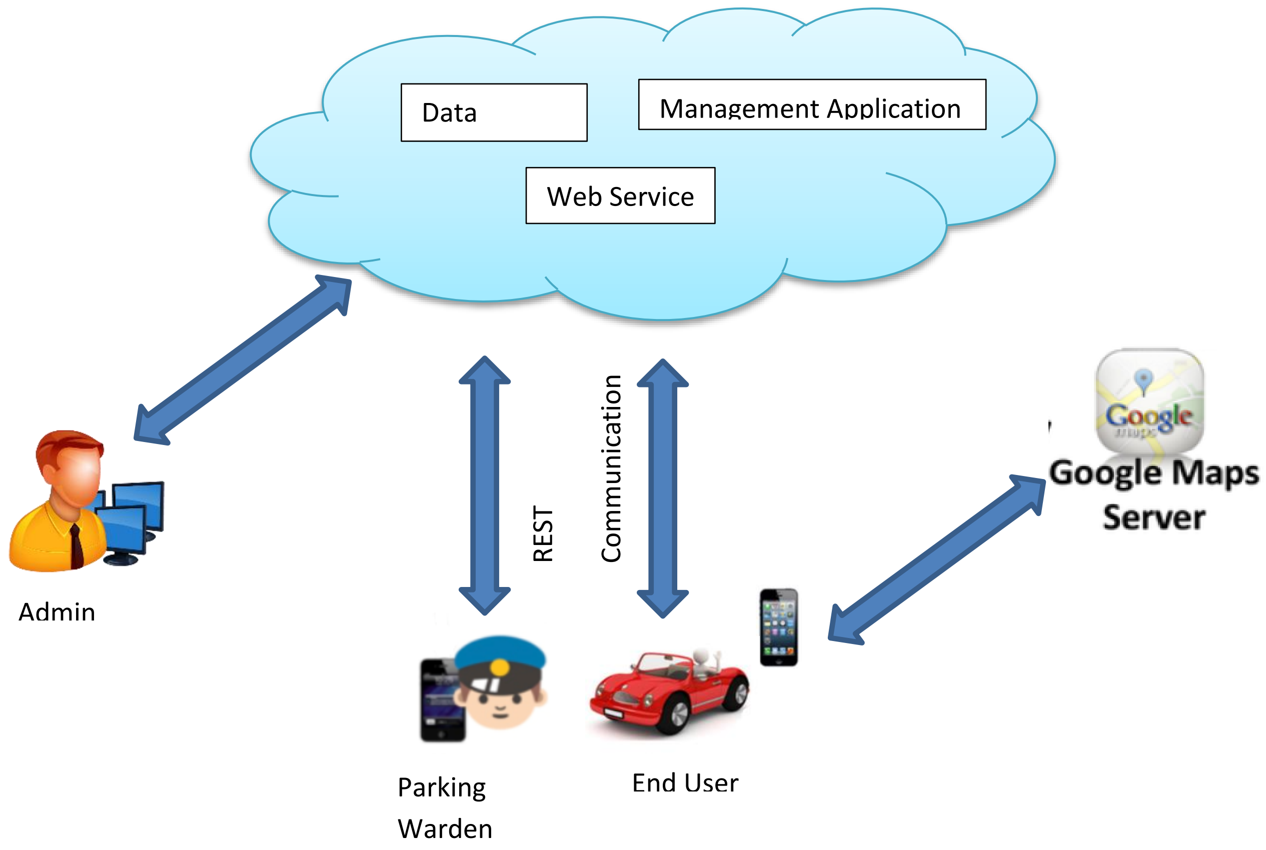


Figure 6 : System Architecture

Basically, the parking system is designed in a manner, which is applicable to cover open parking areas, and street side parking. Going through the technical side, this automated parking system consists of the main components like Cloud system along with Database and Web Service, the Navigation system and User Device. The above image will show the cloud-based system architecture for the parking system.

System architecture shows the components in the parking system, including parking warden, driver, admin, cloud server along with a web service, front end application and a database. The parking warden has the ability to mark availability/ unavailability of parking slots. The main application sever database keeps the data of all parking locations which are assigned to a parking warden. The driver first has to register for the app using internet. The application server keeps track of GPS location where the vehicle exist and display the nearest parking slots according to the current location. All parking slots in an area and their location on GPS are kept in the main database server.

In this system, REST communication is available to call through web services and mobile device. Spring Boot use to build RESTful web services and DB side is develop using MySQL. Functionality and data are considered resources and are accessed using Uniform Resource Identifiers (URIs), specially links on the web in the REST architectural style. RESTful web services are developed to work on the Web at best version. This architectural style contains a client/server architecture and is designed to use a stateless communication protocol, typically HTTP. By using a standardized protocol and interface, clients and servers exchange resource representations in the REST architecture style. Using a set of structured well-defined and simple operations, the resources of REST are acted. Constraints such as performance, scalability, and modifiability specify Representational State Transfer (REST) in an architectural style.

- Driver Panel

The driver panel of your parking mobile app will help users to view for a free parking location nearby with just a click on the mobile application.

Track Location- With the in-built GPS navigation functionality in the app, application can track the current location of the driver’s vehicle.

View available slots with the details- The search feature in a parking app lets its users find available parking in a particular location, city or a particular parking area and select the same as per their convenience.

- Parking Warden Panel

The parking warden panel allows to allocate a particular parking lot once a vehicle comes in. and removes it once the vehicle goes out.

Manage Parking Slots- Through this feature a parking warden can update the availability or unavailability of a particular slot in the location once a vehicle comes in and goes out from the slot.

Check available Slots- The parking warden can check how many free slots in their parking area. This way they can decide on accepting or declining a parking request.

- Admin

Add/Remove/Edit/Disable Parking Slots- Through this feature admin can add new parking slots and remove/disable existing parking slots.

Add/Remove/Edit/Assign wardens to the Parking Locations- Through this feature admin can add new locations, edit existing parking locations, delete the locations and assign the parking locations to a selected warden.

Backend Management System

- Authenticate users and admins before modifying any sensitive data.
- Allow modification of parking lot status by operators.

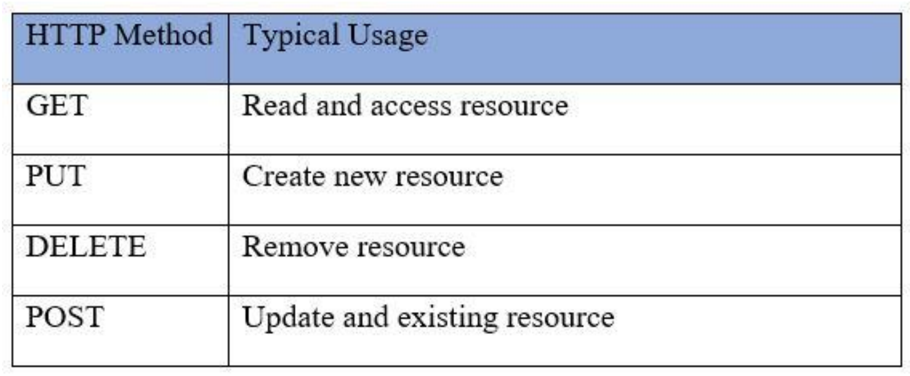
3.2 Implementation Details

This section describes about the developing tools and techniques used in implementing this system and the details of the implementation part of this system. The implementation can be categorized into three main sections as the implementation of the mobile application phase, web service phase and web application phase. Following section demonstrates the implementation details of each section of this system with their processes after discussing the tools and techniques adapted.

3.2.1 Development Tools and Technologies

- Restful Web Service

  Representational State Transfer (REST) is a style of software architecture for distributed hypermedia system which is using the web standards and HTTP protocol, and it is one of the current popular web service implementations. Its basic idea is to treat everything as resource identified by universal resource identifiers (URIs) and supported by HTTP common operations. POST, GET, PUT and DELETE are the mainly used HTTP methods. HTTP and the main principles concepts of REST are developed using RESTful web services [9].



HTTP Method	Typical Usage
GET	Read and access resource
PUT	Create new resource
DELETE	Remove resource
POST	Update and existing resource

Table 2 : HTTP Methods and Usage

The system has implemented to use RESTful style to represent a resource. The following table illustrate some example URI used in the system.

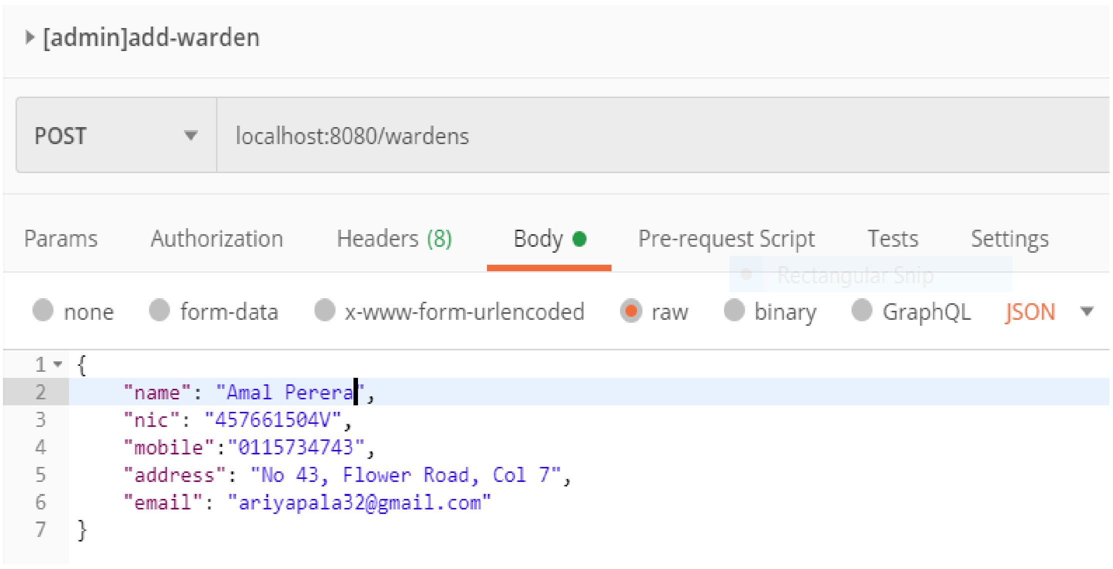
URI	Description
localhost:8080/locations	Add new parking locations
localhost:8080/wardens	Add new parking wardens
localhost:8080/locations/1	Request on locations where the location id=1 pass as a path variable
localhost:8080/wardens	Request to list all the parking wardens
localhost:8080/assign?warden-id=2	Assign locations to parking warden where warden-id is passed as request parameter

localhost:8080/wardens/5	Update parking warden whose id=5 pass as a path variable
localhost:8080/locations/1	Delete location which id=1 pass as a path variable
localhost:8080/wardens	Delete all parking wardens

Table 3 : URIs used in the system

The current system design has exposed JSON interface for the mobile client to retrieve data for available parking spots and update available parking locations via Restful interface from the main application server.

JSON is a lightweight data-interchange format. It is natively support by JavaScript engine as well as many mobile clients, it is commonly used in web service, thus, easier to process. Through RESTful interface, main application server and mobile client communication can be happened. For manipulation in the mobile based client for driver and parking warden, data received by the mobile client will further processed and stored in the data store in memory [10].


[admin]add-warden

POST localhost:8080/wardens

```
{
    "name": "Amal Perera",
    "nic": "457661504V",
    "mobile":"0115734743",
    "address": "No 43, Flower Road, Col 7",
    "email": "ariyapala32@gmail.com"
}
```

Figure 7 : JSON Body

Below have mentioned some API End Points used in the system.

- Add wardens

```
@POST localhost:8080/wardens
@
@Body
{
        "name": "Amal Perera",
        "nic": "457661504V",
        "mobile":"0115734743",
        "address": "No 43, Flower Road, Col 7",
        "email": "ariyapala32@gmail.com"
}
```

- Get a particular warden whose id passed as a path variable

  ```
  @GET localhost:8080/wardens/4
  @path-param 4
  ```

- Request to list all parking wardens

  ```
  @GET localhost:8080/wardens
  ```

- Add locations

  ```
  @POST localhost:8080/locations
  @Body
  {
          "lon":4676.34,
          "lat":1323.378,
          "lname":"Majestic City",
          "city":"Colombo 4",
          "slots": 10
  }
  ```

- Update parking warden data whose id=5 passed as path variable

  ```
  @PUT localhost:8080/wardens/5
  @path-param 5
  @Body
  {
      "slots": [],
      "name": "sumanapala",
      "nic": "762423050v",
      "mobile": "0879372908",
      "address": "moratuwa",
      "email": "sumana@gmail.com"
    }
  ```

- Assign location to a particular warden passed as request parameter

  ```
  @PUT localhost:8080/assign?warden-id=2
  @param warden-id=2
  @Body
  {
          "id":1,
          "lname":"Majestic City"
  }
  ```

- Delete all wardens

  ```
  @DELETE localhost:8080/wardens
  ```

- Delete a particular warden whose id = 6 passed as path variable

@DELETE localhost:8080/wardens/6
@path-param 6

- Spring Framework

Open source application framework for the java platform can be considered as the Spring Framework. The framework avails a complete programing and configuration model for present-day java-based enterprise applications and supports RESTful web services development. Web applications can build on top of Java EE platform with some extensions where spring framework not only supports the development of Java applications. All the tiers including one tier- stand-alone java application, web tier- in web application and enterprise tier- in Enterprise Java Beans. The above-mentioned tiers are a single place for Java based application on all layers [11].

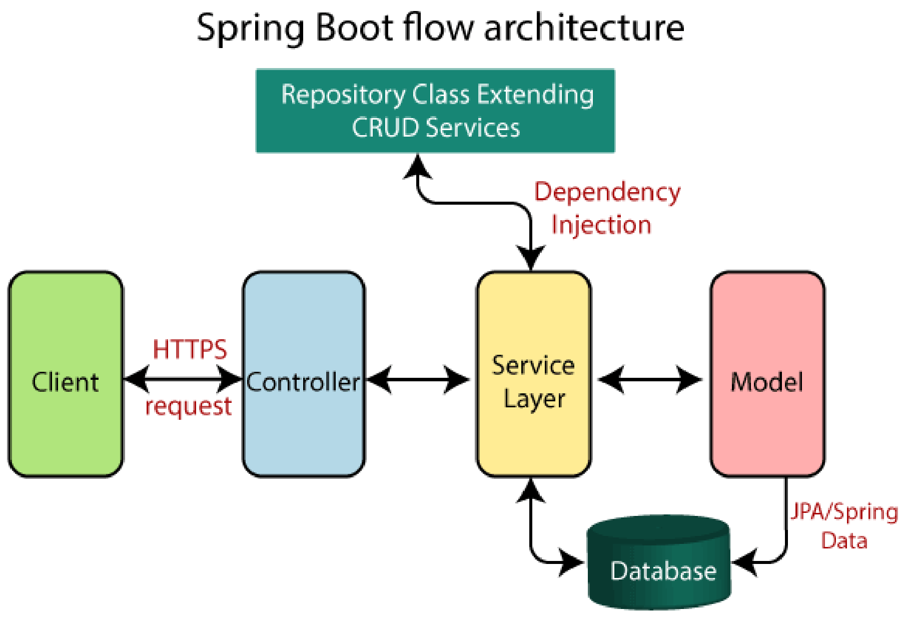


Figure 8 : Spring Boot flow architecture

- Android Studio

Android studio allows to download in the operating systems like Windows, macOS and Linux which allows to do developments on JetBrains' IntelliJ IDEA software. For operating system of google’s android integration of development environment can be used. Using the new constraint layouts along with a conversion of PNG, BMP, GIF, JPEG to WEbP format used the current android studio version v2.3.0 for this project. As the primary IDE for native Android application development and for Eclipse Android Development Tools, the replacement can happen. To give a better compression, WebP file format is used which provides transparency (like PNG) and a glossy compression (like JPEG) [12]

- Android SDK

  Comprehensive set of development tools include the Android software development kit (SDK) includes a. Sample code, tutorials, libraries, debuggers and handset emulators are included on this. SDK version v29 is used for this project implementation. Currently Windows, Mac OS, Linux are the supported development platforms for Android SDK [13].

- Third party libraries and APIs

  The Google Maps Android API supports to add Google Maps data to the application. The map features give functionalities like access to Google Maps servers, data downloading, map display, and response to map gestures. Allowing the direct interaction on user with the map when load the application and display the information for free slots of locations are the additional features [14]. Adding markers, polygons, overlays in the map can be done using API calls, and to set up the user's current view in a particular map area. Adding following features into the map will be allowed by the third-party libraries and APIs:

  Bitmap graphics anchored to specific positions on the map.
  Sets of line segments.
  Enclosed segments.
  Icons anchored to specific positions on the map.
  Sets of images which are displayed on top of the base map tiles.

  The parking slots displayed according to the current location of driver's vehicle, have used third party APIs and libraries for this [15].

The system can be divided into the sections as a database server, a main server application (web service), administrative front end as well as a mobile based front end. The web service is implemented by using Java Spring Boot. It handles all the business logic and process user's request and communication with the web management front end, mobile client via HTTP interface. It will extract and validate user's submitted request according to the validation rules against the actual data stored in the database, received requests which failure to validate will be rejected by the system while valid input will be stored and further processed by the server.

For data communication between client and server, standard HTTP request, JSON, will be used and Restful Web service is implemented to enable easy communication [16].

The MySQL database will store all the data processed by the system and all data will accessed and manipulated by the back-end system deployed at the application server directly.

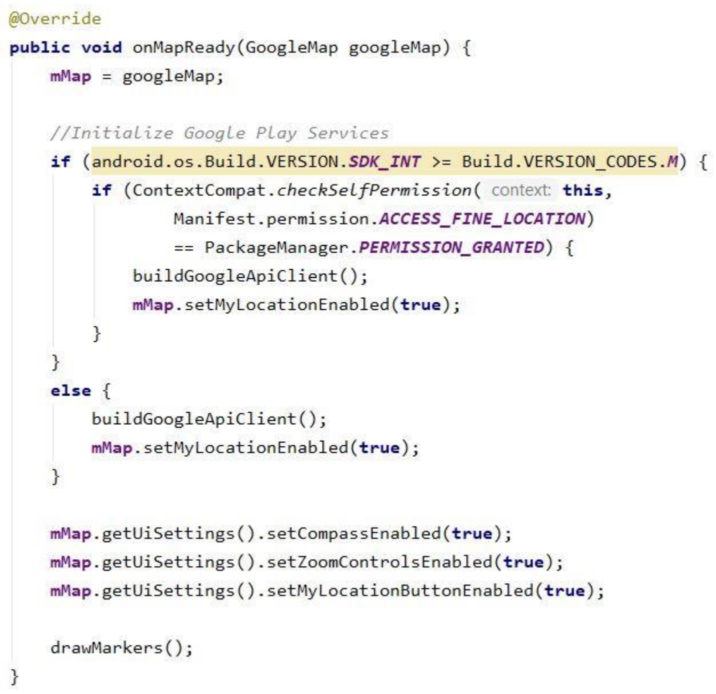
The management front-end will be used by the admin user. It is implemented using React along with HTML, JavaScript and Bootstrap. The major purpose is to interpret user's input and submit request to the application server for further processing. It also includes basic validation of user input, so that all operational logic can remain at the server and prevent security issues, this also make the system more maintainable.

The mobile based client will be used by the driver and the parking warden. As a front-end client, it also interprets user’s input and submits requests to the application server for further processing. It allows driver to view the map with available parking slots, search parking locations and update available parking slots by the warden. As integrated with google map, the mobile client will also communicate with google map server to retrieve geo location information as well as address querying information.

3.2.2 System Implementation

Some important code sections of the system are included below.

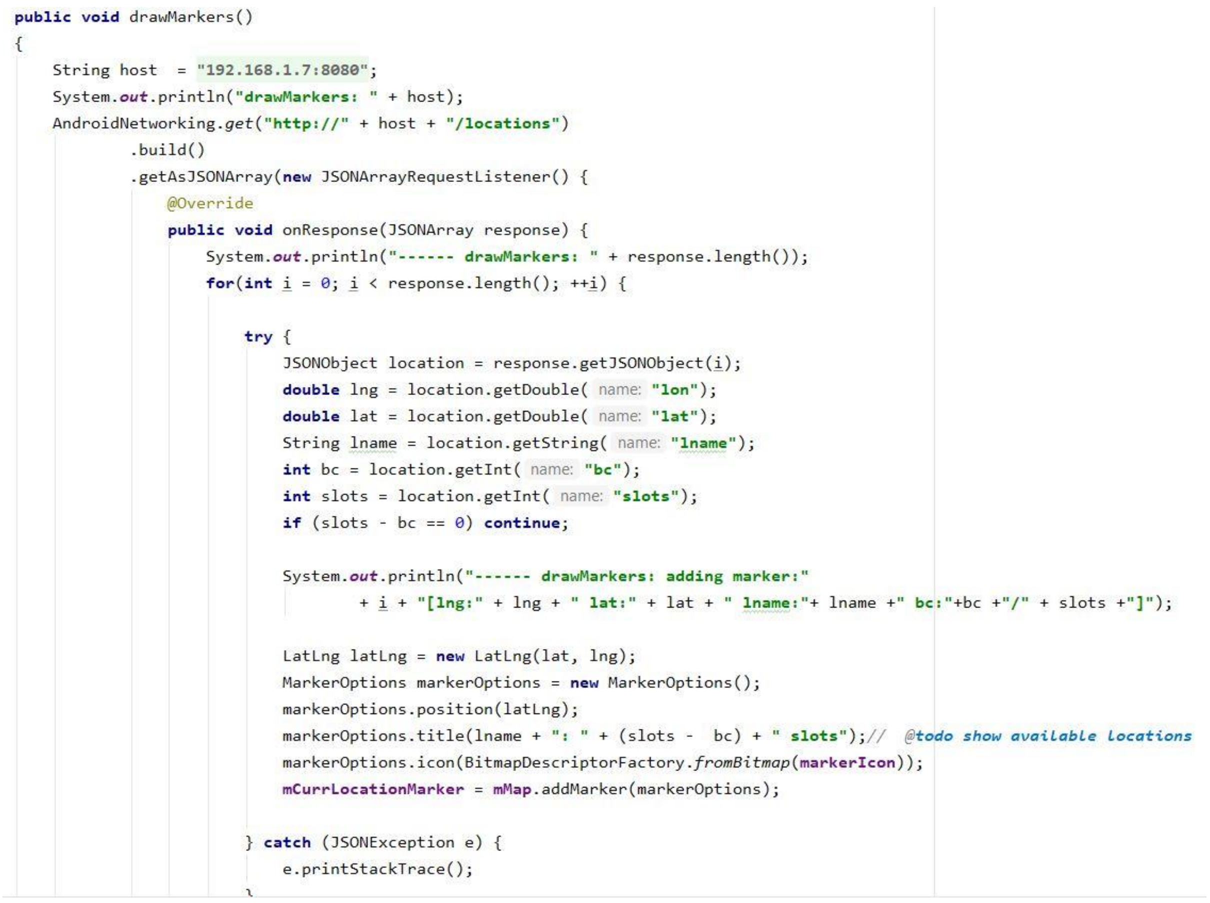
The callback is given to the map object once after the map is ready. The map is stored for later usage. Basically map related UI settings, like location-enable facility, compass-enable facility, zoom controllers and enable mylocations button are added. Finally the marker drawing function is called.



```java
@Override
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;

    //Initialize Google Play Services
    if (android.os.Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
        if (ContextCompat.checkSelfPermission( context: this,
                Manifest.permission.ACCESS_FINE_LOCATION)
                == PackageManager.PERMISSION_GRANTED) {
            buildGoogleApiClient();
            mMap.setMyLocationEnabled(true);
        }
    }
    else {
        buildGoogleApiClient();
        mMap.setMyLocationEnabled(true);
    }

    mMap.getUiSettings().setCompassEnabled(true);
    mMap.getUiSettings().setZoomControlsEnabled(true);
    mMap.getUiSettings().setMyLocationButtonEnabled(true);

    drawMarkers();
}
```

Figure 9 : Code section for map ready functionality

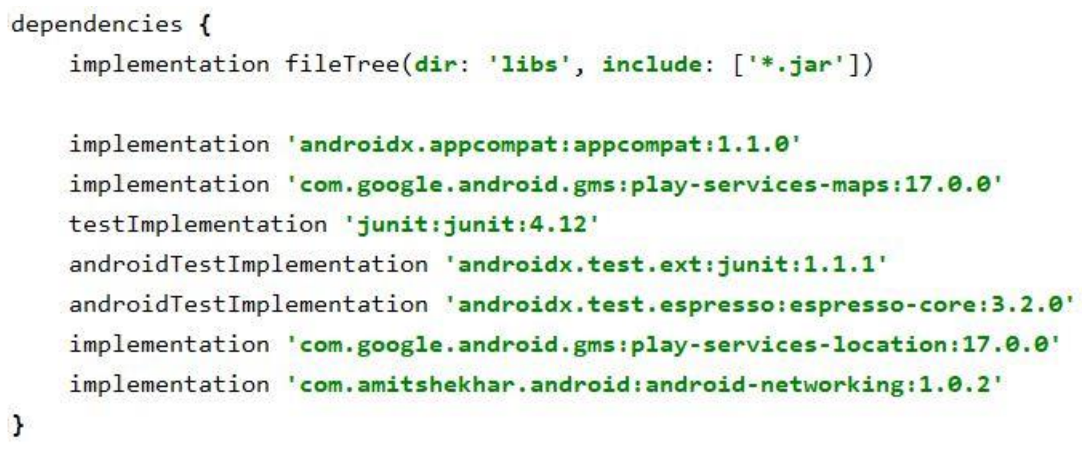
Code snippet to load all the available parking slots on the map near the current location where the driver exist.



```
public void drawMarkers()
{
    String host  = "192.168.1.7:8080";
    System.out.println("drawMarkers: " + host);
    AndroidNetworking.get("http://" + host + "/locations")
            .build()
            .getAsJSONArray(new JSONArrayRequestListener() {
                @Override
                public void onResponse(JSONArray response) {
                    System.out.println("------ drawMarkers: " + response.length());
                    for(int i = 0; i < response.length(); ++i) {

                        try {
                            JSONObject location = response.getJSONObject(i);
                            double lng = location.getDouble( name: "lon");
                            double lat = location.getDouble( name: "lat");
                            String lname = location.getString( name: "lname");
                            int bc = location.getInt( name: "bc");
                            int slots = location.getInt( name: "slots");
                            if (slots - bc == 0) continue;

                            System.out.println("------ drawMarkers: adding marker:"
                                    + i + "[lng:" + lng + " lat:" + lat + " lname:"+ lname +" bc:"+bc +"/" + slots +"]");

                            LatLng latLng = new LatLng(lat, lng);
                            MarkerOptions markerOptions = new MarkerOptions();
                            markerOptions.position(latLng);
                            markerOptions.title(lname + ": " + (slots -  bc) + " slots");//  @todo show available locations
                            markerOptions.icon(BitmapDescriptorFactory.fromBitmap(markerIcon));
                            mCurrLocationMarker = mMap.addMarker(markerOptions);

                        } catch (JSONException e) {
                            e.printStackTrace();
```

Figure 10 : Code section to load all available parking slot and draw a marker

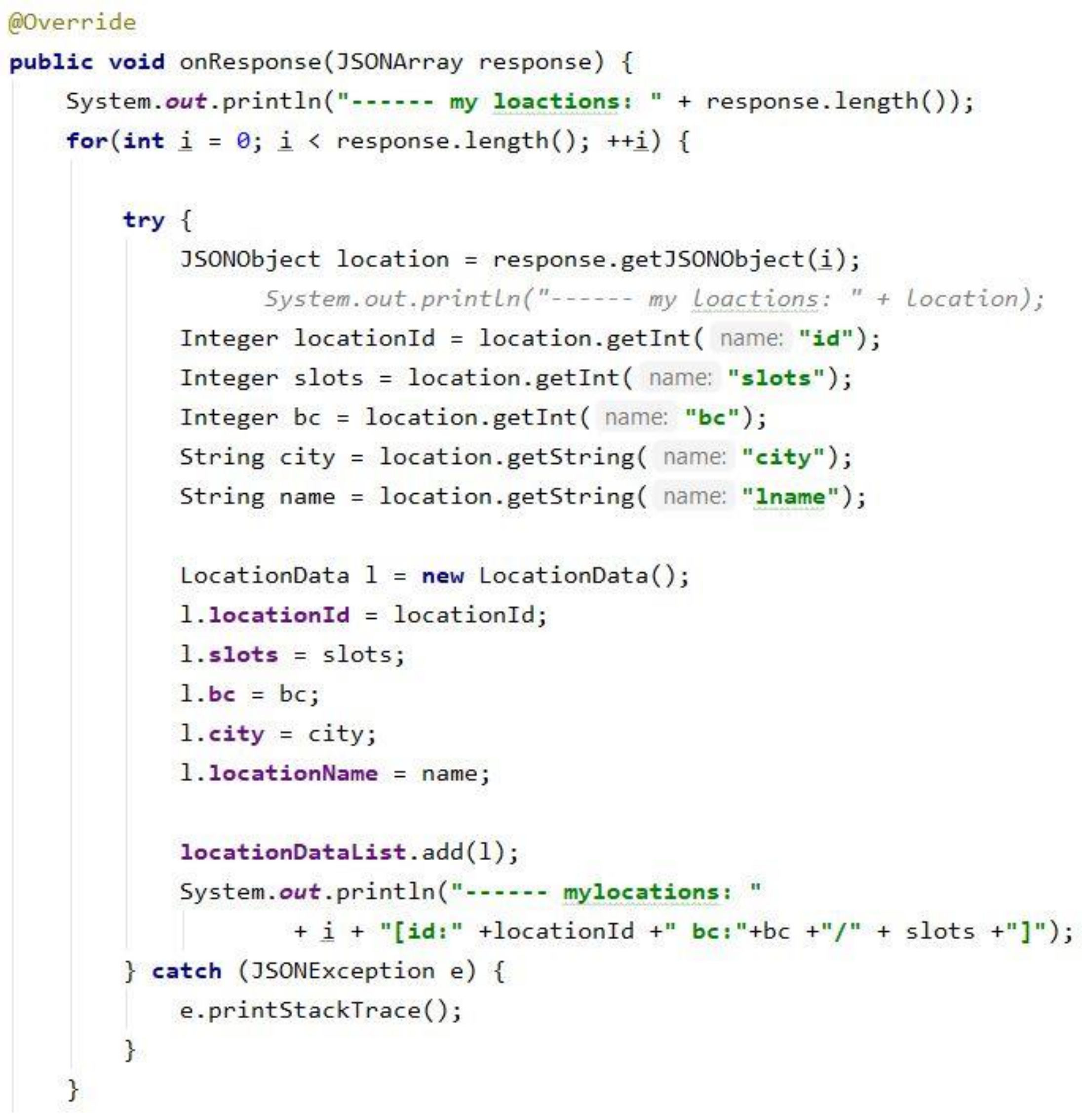
All the dependencies that used to build the application.



```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])

    implementation 'androidx.appcompat:appcompat:1.1.0'
    implementation 'com.google.android.gms:play-services-maps:17.0.0'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.1'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0'
    implementation 'com.google.android.gms:play-services-location:17.0.0'
    implementation 'com.amitshekhar.android:android-networking:1.0.2'
}
```

Figure 11 : Code section for added dependencies

The assigned locations for the parking warden can be tracked using the following code snippet.

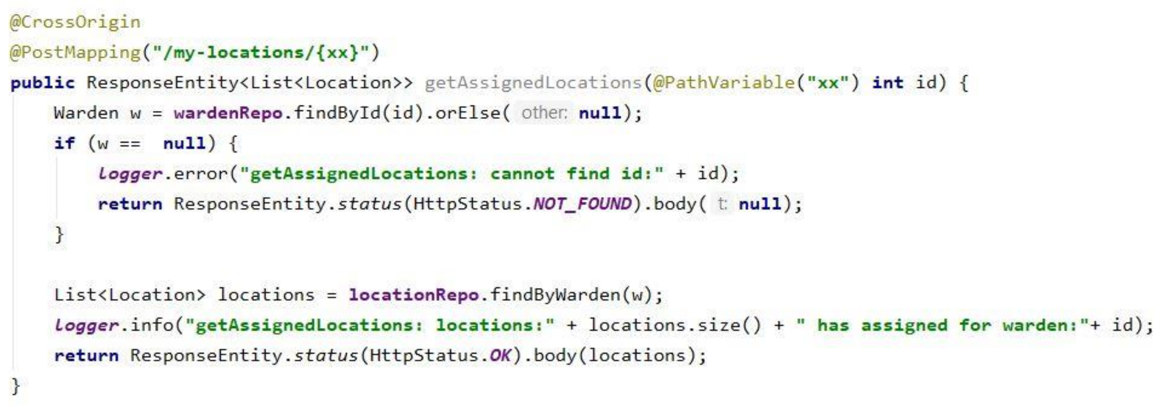


```
@Override
public void onResponse(JSONArray response) {
    System.out.println("------ my loactions: " + response.length());
    for(int i = 0; i < response.length(); ++i) {

        try {
            JSONObject location = response.getJSONObject(i);
                  System.out.println("------ my loactions: " + location);
            Integer locationId = location.getInt( name: "id");
            Integer slots = location.getInt( name: "slots");
            Integer bc = location.getInt( name: "bc");
            String city = location.getString( name: "city");
            String name = location.getString( name: "lname");

            LocationData l = new LocationData();
            l.locationId = locationId;
            l.slots = slots;
            l.bc = bc;
            l.city = city;
            l.locationName = name;

            locationDataList.add(l);
            System.out.println("------ mylocations: "
                    + i + "[id:" +locationId +" bc:"+bc +"/" + slots +"]");
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }
```

Figure 12 : Code section to track all assigned locations for parking warden

The web service is implemented using java spring boot which mainly consist of AdminController.java, WardenController.java and DriverController.java.

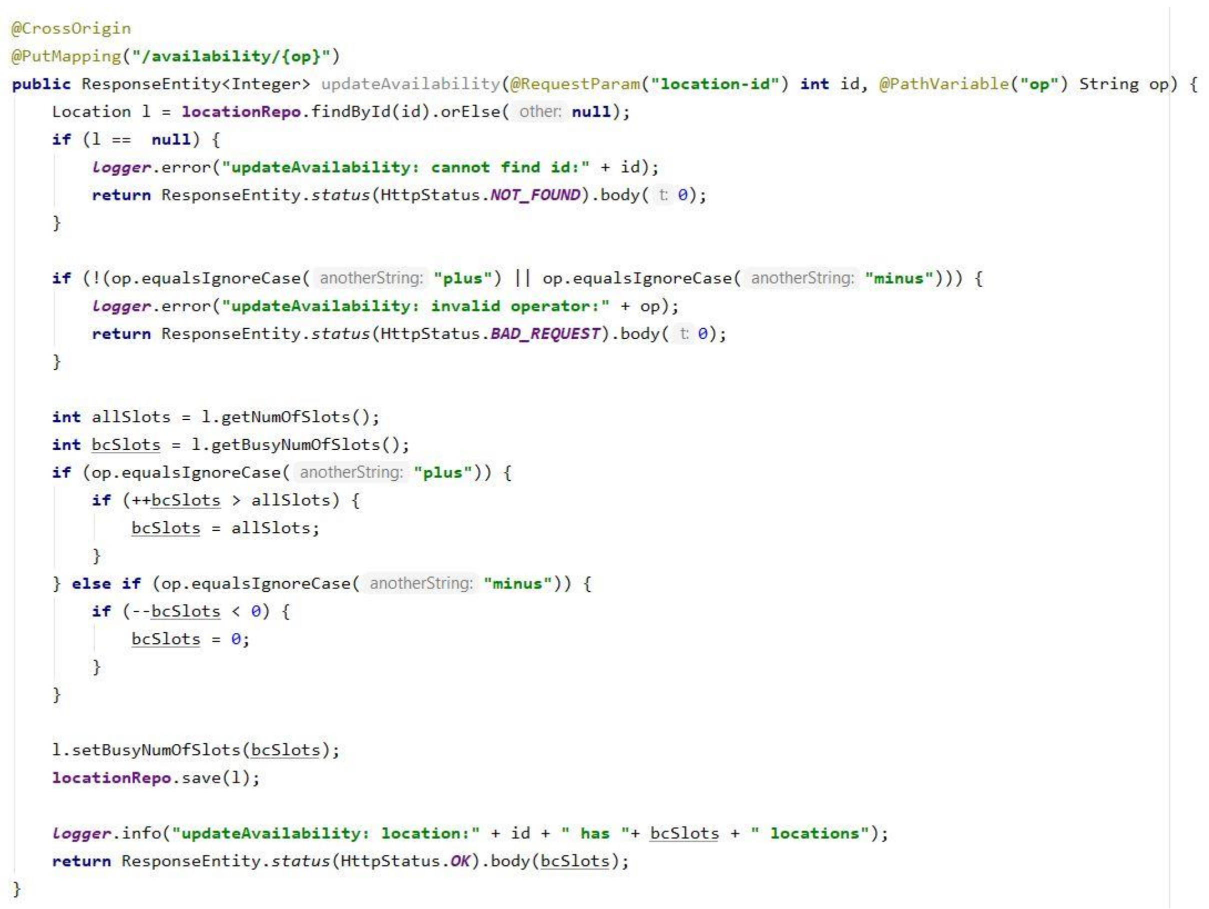
The following code snippets displays Admin and Warden functionality methods.



```java
@CrossOrigin
@PostMapping("/my-locations/{xx}")
public ResponseEntity<List<Location>> getAssignedLocations(@PathVariable("xx") int id) {
    Warden w = wardenRepo.findById(id).orElse( other: null);
    if (w ==  null) {
        logger.error("getAssignedLocations: cannot find id:" + id);
        return ResponseEntity.status(HttpStatus.NOT_FOUND).body( t: null);
    }

    List<Location> locations = locationRepo.findByWarden(w);
    logger.info("getAssignedLocations: locations:" + locations.size() + " has assigned for warden:"+ id);
    return ResponseEntity.status(HttpStatus.OK).body(locations);
}
```

Figure 13 : Code section to get the assigned locations

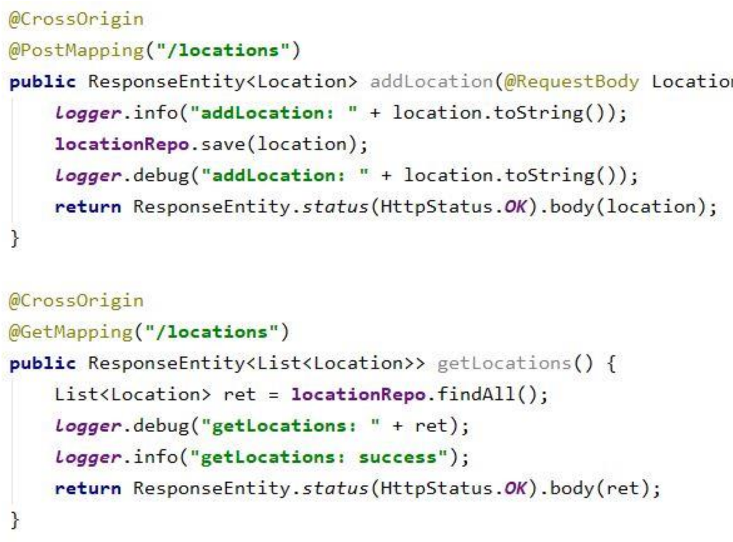
This function checks for own locations assigned by admin and return data to the front-end. All REST functions as well as this function enable cross-origin to grant access from outside.



```java
@CrossOrigin
@PutMapping("/availability/{op}")
public ResponseEntity<Integer> updateAvailability(@RequestParam("location-id") int id, @PathVariable("op") String op) {
    Location l = locationRepo.findById(id).orElse( other: null);
    if (l ==  null) {
        logger.error("updateAvailability: cannot find id:" + id);
        return ResponseEntity.status(HttpStatus.NOT_FOUND).body( t: 0);
    }

    if (!(op.equalsIgnoreCase( anotherString: "plus") || op.equalsIgnoreCase( anotherString: "minus"))) {
        logger.error("updateAvailability: invalid operator:" + op);
        return ResponseEntity.status(HttpStatus.BAD_REQUEST).body( t: 0);
    }

    int allSlots = l.getNumOfSlots();
    int bcSlots = l.getBusyNumOfSlots();
    if (op.equalsIgnoreCase( anotherString: "plus")) {
        if (++bcSlots > allSlots) {
            bcSlots = allSlots;
        }
    } else if (op.equalsIgnoreCase( anotherString: "minus")) {
        if (--bcSlots < 0) {
            bcSlots = 0;
        }
    }

    l.setBusyNumOfSlots(bcSlots);
    locationRepo.save(l);

    logger.info("updateAvailability: location:" + id + " has "+ bcSlots + " locations");
    return ResponseEntity.status(HttpStatus.OK).body(bcSlots);
}
```

Figure 14 : Code section to update the availability/unavailability of parking slots

The function responsible to update the availability of free slots in a particular place. Additionally it has the ability to handle the free slots not to exceed the maximum number of slots and not to decrease that less than zero in the place.

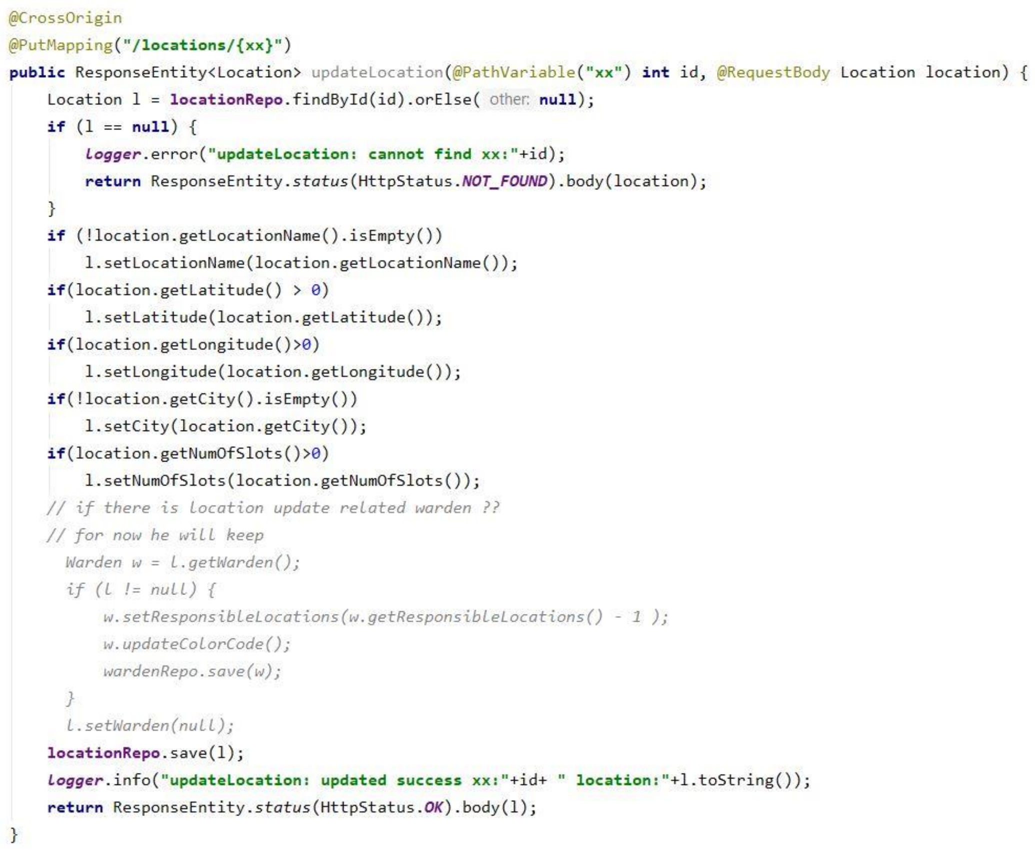


```java
@CrossOrigin
@PostMapping("/locations")
public ResponseEntity<Location> addLocation(@RequestBody Location location) {
    logger.info("addLocation: " + location.toString());
    locationRepo.save(location);
    logger.debug("addLocation: " + location.toString());
    return ResponseEntity.status(HttpStatus.OK).body(location);
}

@CrossOrigin
@GetMapping("/locations")
public ResponseEntity<List<Location>> getLocations() {
    List<Location> ret = locationRepo.findAll();
    logger.debug("getLocations: " + ret);
    logger.info("getLocations: success");
    return ResponseEntity.status(HttpStatus.OK).body(ret);
}
```

Figure 15 : Code section to POST locations and GET locations

Add function insert location data to database table and the get function can be used to retrieve all locations data to outside.

The following function has been implemented to update selected location details and it has bound to locate the edit feature in admin-ui.



```java
@CrossOrigin
@PutMapping("/locations/{xx}")
public ResponseEntity<Location> updateLocation(@PathVariable("xx") int id, @RequestBody Location location) {
    Location l = locationRepo.findById(id).orElse( other: null);
    if (l == null) {
        logger.error("updateLocation: cannot find xx:"+id);
        return ResponseEntity.status(HttpStatus.NOT_FOUND).body(location);
    }
    if (!location.getLocationName().isEmpty())
        l.setLocationName(location.getLocationName());
    if(location.getLatitude() > 0)
        l.setLatitude(location.getLatitude());
    if(location.getLongitude()>0)
        l.setLongitude(location.getLongitude());
    if(!location.getCity().isEmpty())
        l.setCity(location.getCity());
    if(location.getNumOfSlots()>0)
        l.setNumOfSlots(location.getNumOfSlots());
    // if there is location update related warden ??
    // for now he will keep
      Warden w = l.getWarden();
      if (l != null) {
          w.setResponsibleLocations(w.getResponsibleLocations() - 1 );
          w.updateColorCode();
          wardenRepo.save(w);
      }
      l.setWarden(null);
    locationRepo.save(l);
    logger.info("updateLocation: updated success xx:"+id+ " location:"+l.toString());
    return ResponseEntity.status(HttpStatus.OK).body(l);
}
```

Figure 16 : Code section to update locations

3.2.3 Database Design

Figure 17 is the ERD for the parking system. Fields in Location table has mapped with location class in the web service. Each location has primary key that mapped to location id and the foreign key is wid mapped to warden table primary key. Parking detail table is an extended table for gather vehicle details that will be used to identify correct parking locations areas' requirements.

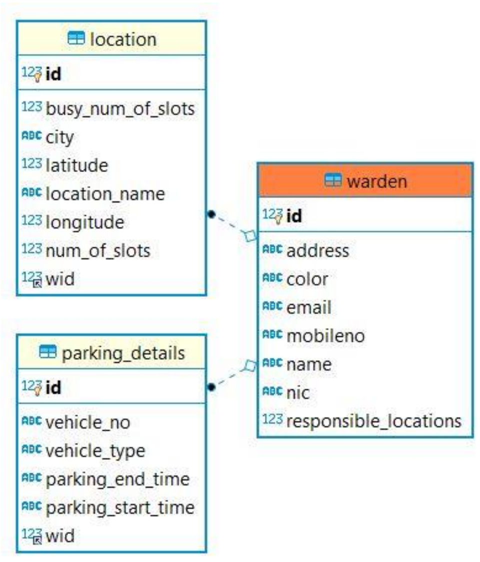


Figure 17 : Database table design for Location, Warden and Parking Details

Table views on DBeaver are displayed on following figures.

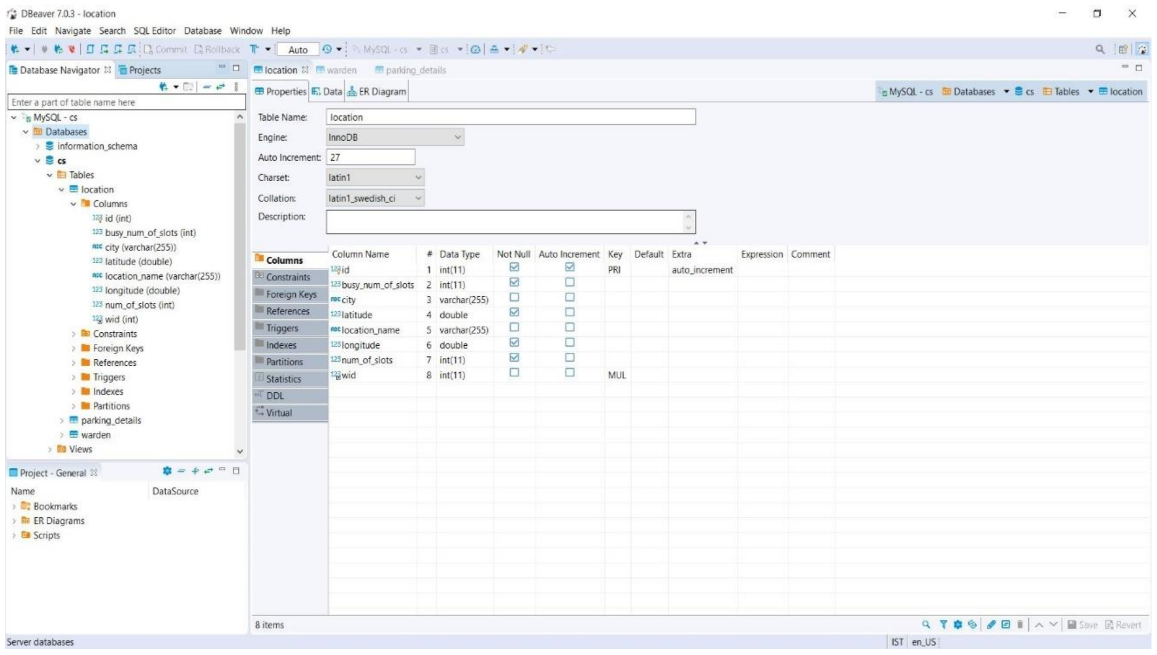


Table Name: location
Engine: InnoDB
Auto Increment: 27
Charset: latin1
Collation: latin1_swedish_ci
Description:

Column Name	#	Data Type	Not Null	Auto Increment	Key	Default	Extra	Expression	Comment
id	1	int(11)	✓	✓	PRI		auto_increment		
busy_num_of_slots	2	int(11)	✓	☐					
city	3	varchar(255)	☐	☐					
latitude	4	double	✓	☐					
location_name	5	varchar(255)	☐	☐					
longitude	6	double	✓	☐					
num_of_slots	7	int(11)	✓	☐					
wid	8	int(11)	☐	☐	MUL				

Figure 18 : Location table details

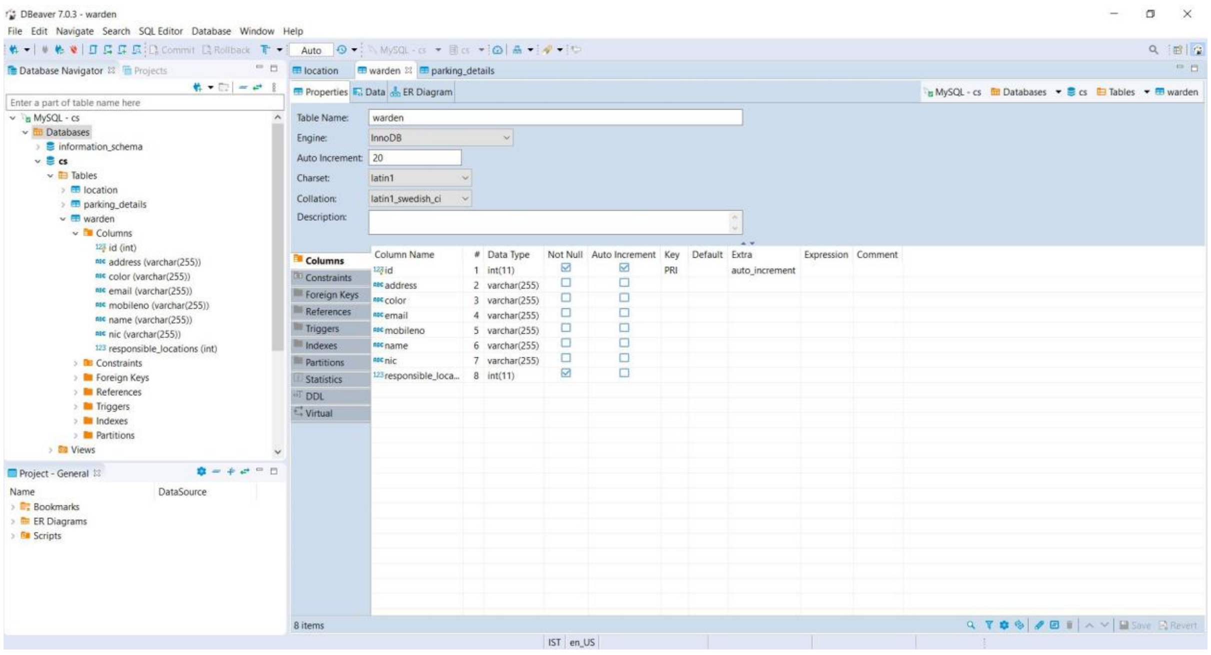


Table Name:	warden
Engine:	InnoDB
Auto Increment:	20
Charset:	latin1
Collation:	latin1_swedish_ci
Description:	

Column Name	#	Data Type	Not Null	Auto Increment	Key	Default	Extra	Expression	Comment
id	1	int(11)	✓	✓	PRI		auto_increment		
address	2	varchar(255)	☐	☐					
color	3	varchar(255)	☐	☐					
email	4	varchar(255)	☐	☐					
mobileno	5	varchar(255)	☐	☐					
name	6	varchar(255)	☐	☐					
nic	7	varchar(255)	☐	☐					
responsible_loca...	8	int(11)	✓	☐					

Figure 19 : Warden table details

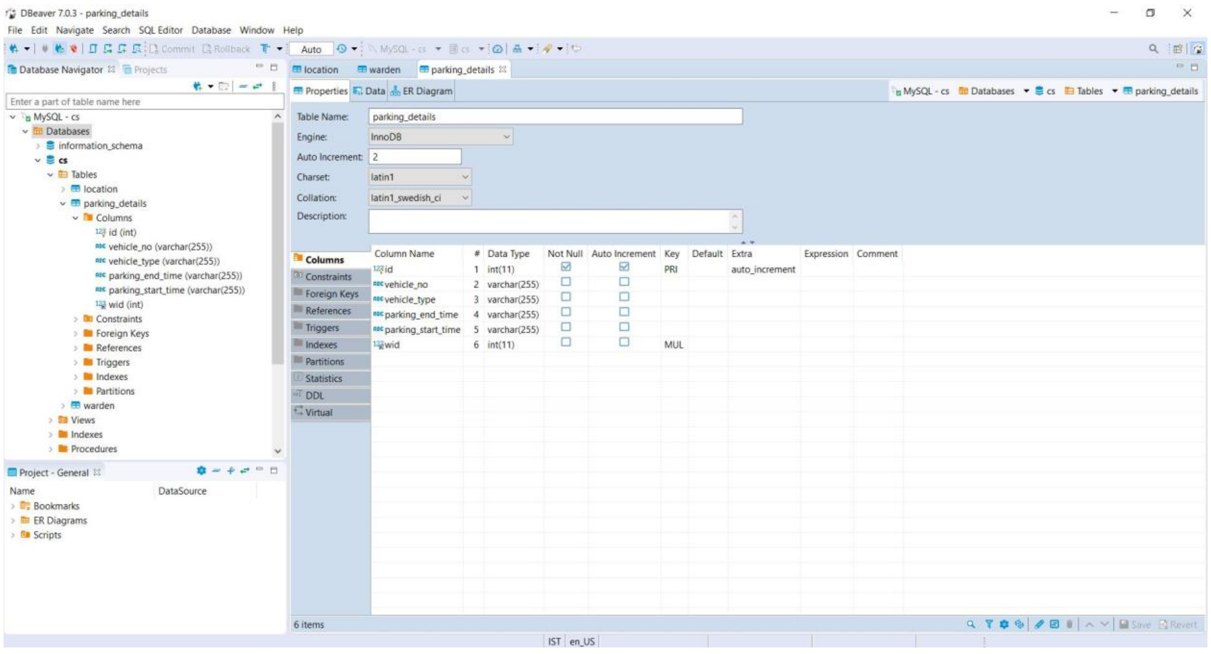


Table Name:	parking_details
Engine:	InnoDB
Auto Increment:	2
Charset:	latin1
Collation:	latin1_swedish_ci
Description:	

Column Name	#	Data Type	Not Null	Auto Increment	Key	Default	Extra	Expression	Comment
id	1	int(11)	✓	✓	PRI		auto_increment		
vehicle_no	2	varchar(255)	☐	☐					
vehicle_type	3	varchar(255)	☐	☐					
parking_end_time	4	varchar(255)	☐	☐					
parking_start_time	5	varchar(255)	☐	☐					
wid	6	int(11)	☐	☐	MUL				

Figure 20 : Parking table details

3.2.4 Mobile Client Interface

The system provides a mobile client for parking warden and driver. It includes functions of view available parking slots, search parking locations and update parking slot availability. The mobile client is another front-end interface of the main application, it will process user input, issues requests to main server and render the results returned by the server with minimum business logic processing.

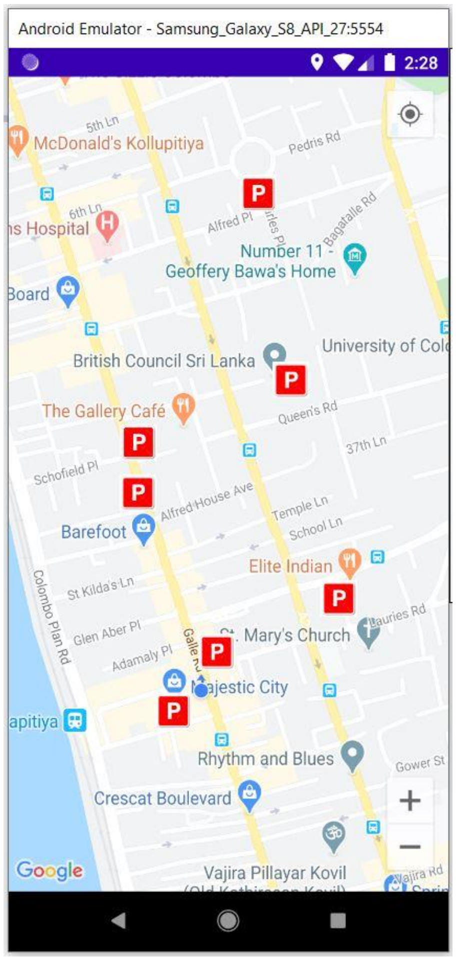


Figure 21 : Map load view with free parking slot

To provide an easy to use and user-friendly parking slot finding experience, the app has integrated those functionalities with google map.

Once open the mobile application, the map will open up with the current location and also request to the application server to perform loading the location area which have free parking slots, which is within the current location area. Once server returns the matching results, the mobile client will display the free parking spot locations. Once the user clicks on a particular location it will show the location details by giving the location name and the number of slots available in that particular location as displayed on Figure 22.

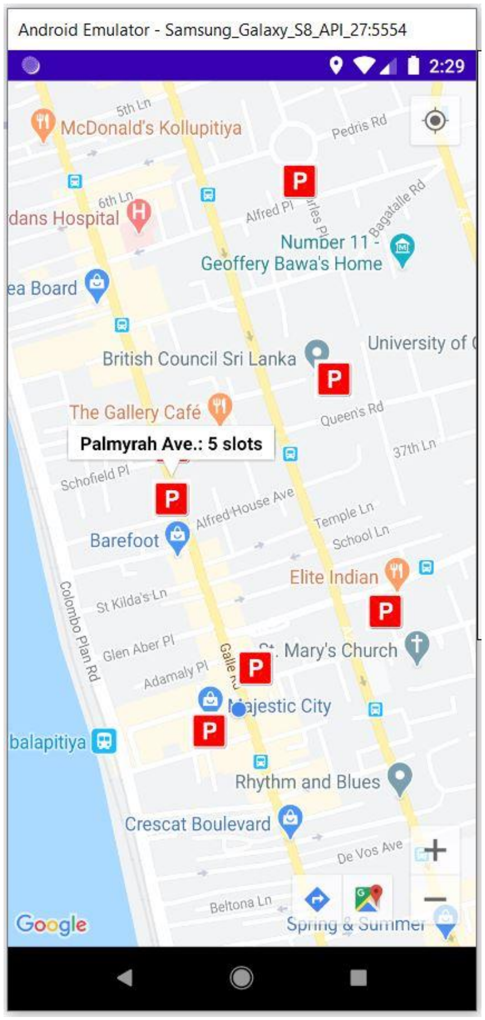


Figure 22 : Map load view with location details

Then at the bottom of the app which shown in Figure 22, the option will give the user to redirect to the google map to find the direction to the location. When clicking on that, google map will load as shown in Figure 23.

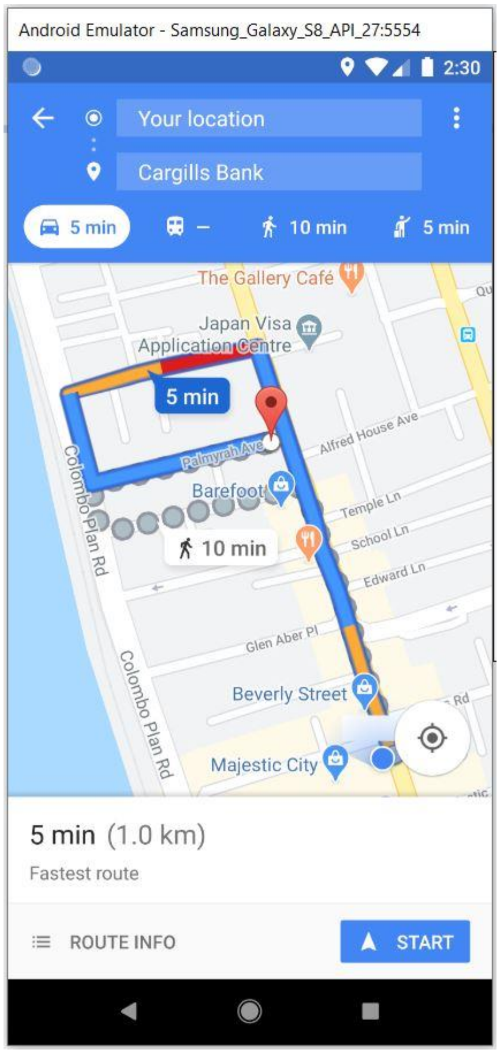


Figure 23 : Google Map View with the direction

Once a particular vehicle reserves a slot on the location, parking warden will log into the mobile client app which will gives the separate user view and will update the available slots on that location. And at the same time when a particular vehicle leaves the location, similarly warden can update the available slots. Figure 24 shows the parking warden panel where at the top, warden can swipe through the assigned locations. From the bottom controller changing slots for the selected location can be done. The app functionality gives the warden to add up the slots to the maximum point and reduce the slots up to 0.

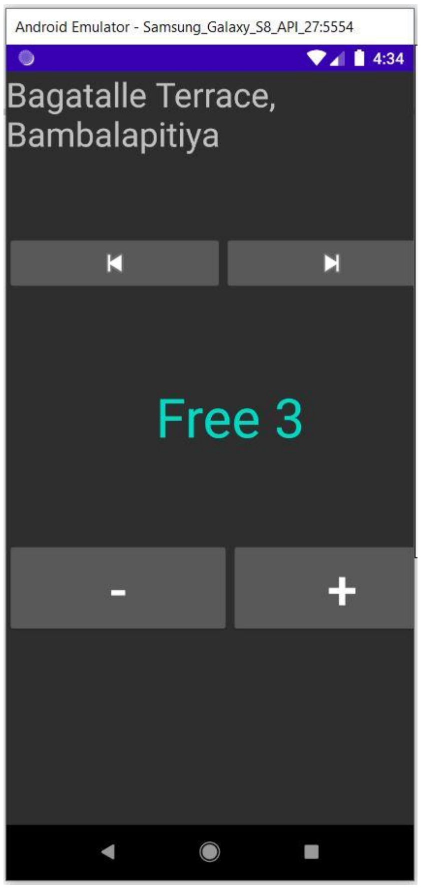

Android Emulator - Samsung_Galaxy_S8_API_27:5554
4:34
Bagatalle Terrace, Bambalapitiya
Free 3
-
+

Figure 24 : Warden UI

3.2.5 Web Client Interface

Through the web management front-end, the following figure shows all the parking location and parking warden setting functionalities. First admin should login to the system using the screen shown in Figure 25.

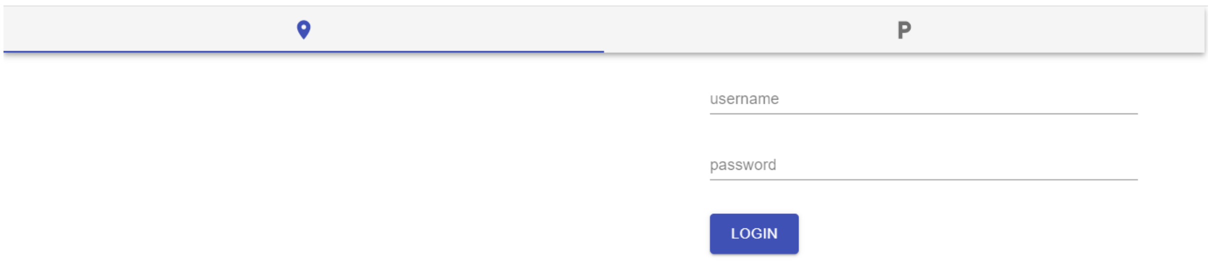
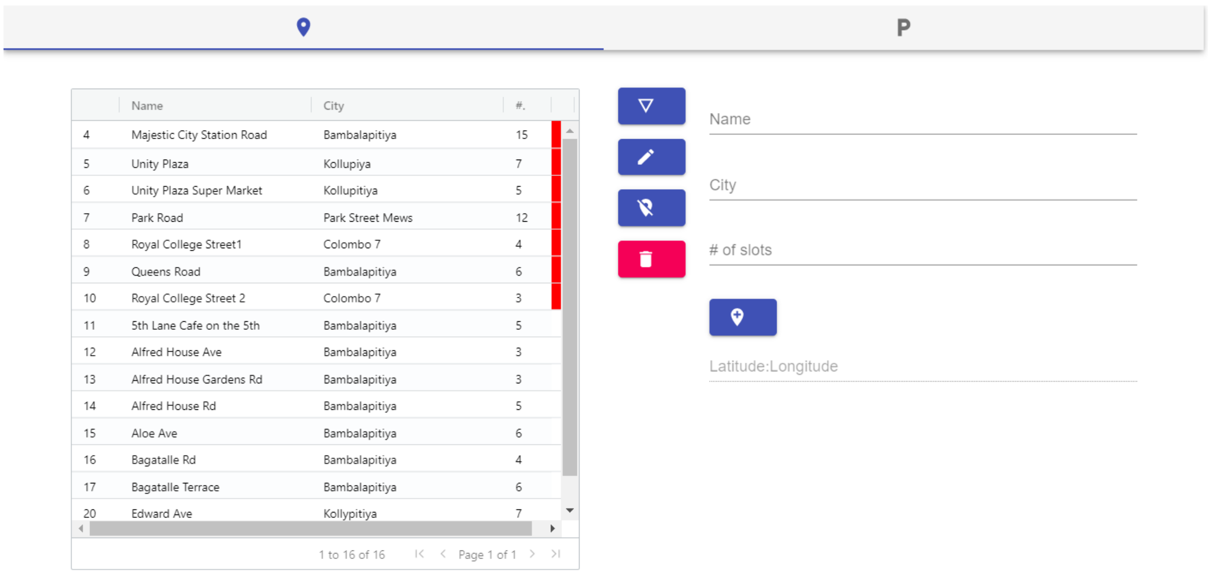


Figure 25 : Admin Login UI

Using the below interface locations can be setup. A new location can be saved into the system by giving the Name, City, No of Slots, Longitude and Latitude. Longitude and Latitude can be entered by selecting location on the map which is displayed when clicking on the location add button on the interface as shown Figure 26.



	Name	City	#.
4	Majestic City Station Road	Bambalapitiya	15
5	Unity Plaza	Kollupiya	7
6	Unity Plaza Super Market	Kollupitiya	5
7	Park Road	Park Street Mews	12
8	Royal College Street1	Colombo 7	4
9	Queens Road	Bambalapitiya	6
10	Royal College Street 2	Colombo 7	3
11	5th Lane Cafe on the 5th	Bambalapitiya	5
12	Alfred House Ave	Bambalapitiya	3
13	Alfred House Gardens Rd	Bambalapitiya	3
14	Alfred House Rd	Bambalapitiya	5
15	Aloe Ave	Bambalapitiya	6
16	Bagatalle Rd	Bambalapitiya	4
17	Bagatalle Terrace	Bambalapitiya	6
20	Edward Ave	Kollypitiya	7

1 to 16 of 16 Page 1 of 1

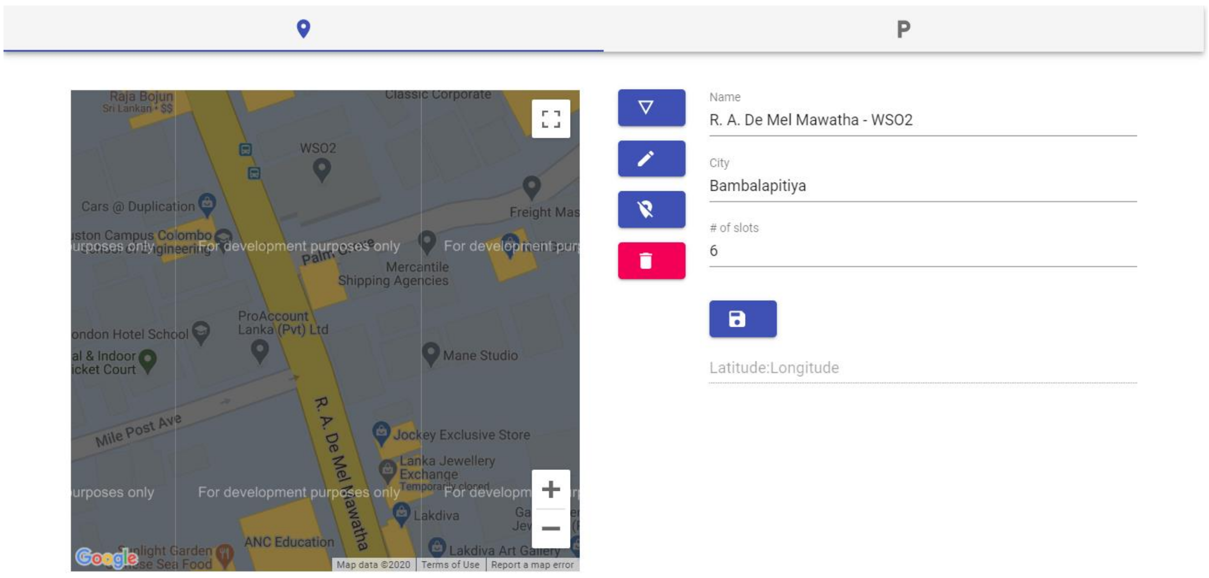
Name

City

# of slots

Latitude:Longitude

Figure 26 : Parking Location UI



Name
R. A. De Mel Mawatha - WSO2

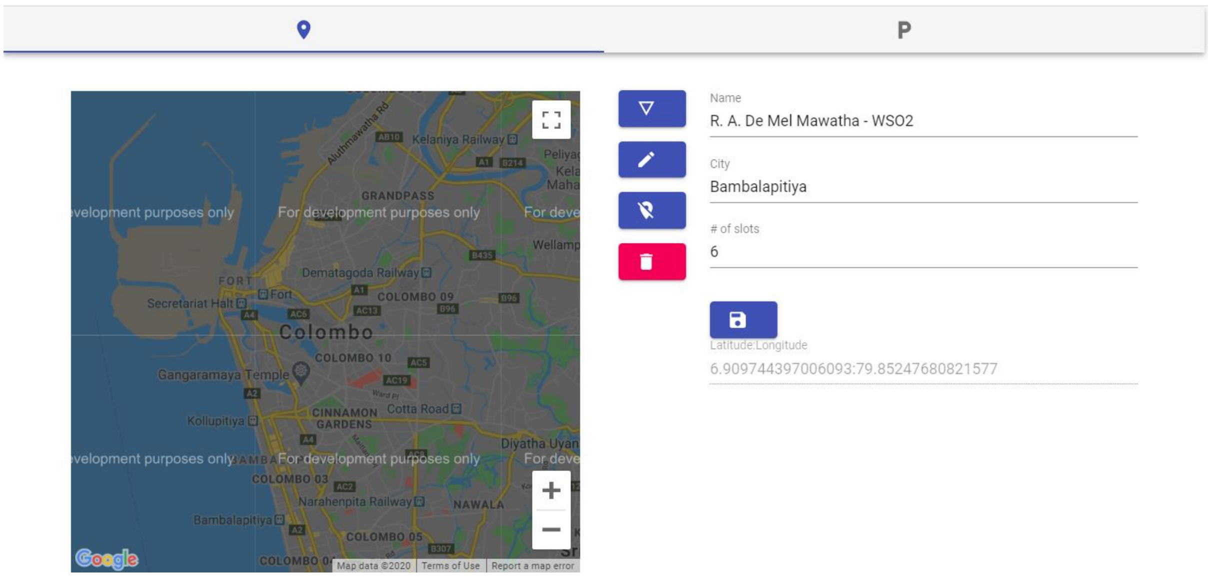
City
Bambalapitiya

# of slots
6

Latitude:Longitude

Figure 27 : Map Load View to select Longitude and Latitude

Once select the exact location from the map, the relevant Longitude and Latitude is auto selected into the field as shown in Figure 28.



Name
R. A. De Mel Mawatha - WSO2

City
Bambalapitiya

# of slots
6

Latitude:Longitude
6.909744397006093:79.85247680821577

Figure 28 : Logitude and Latitude of the selected location

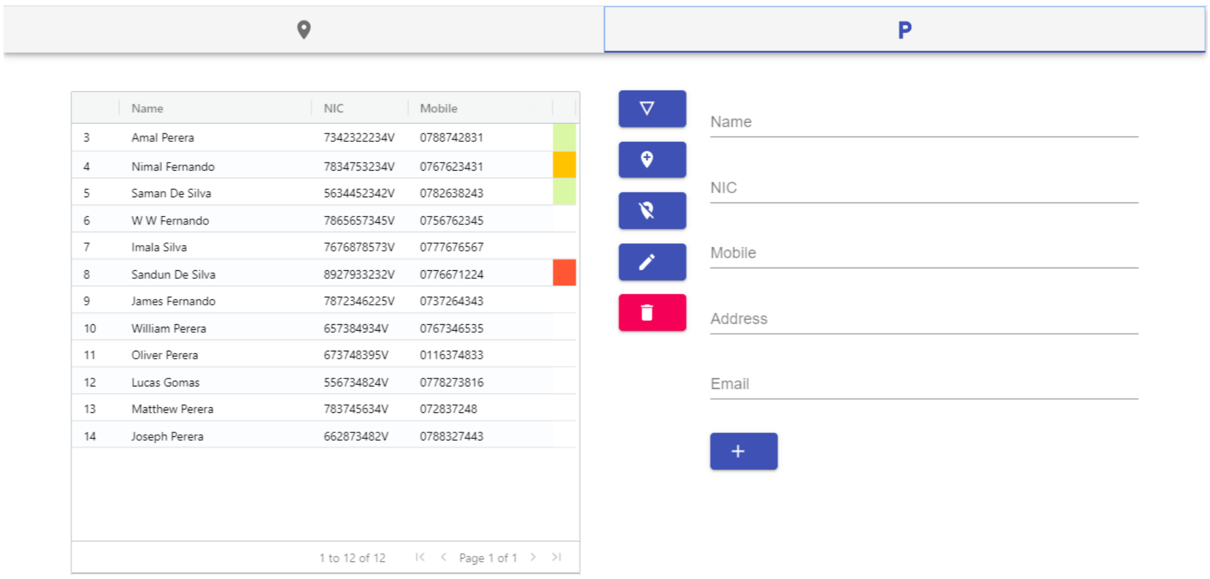
The functionalities exist to View, Edit and Delete an existing parking location. The added parking locations can be view on the grid on left side. When a particular location is not assigned to a parking warden it is by default it displayed using color code White. Once assign a particular warden to the location, it displayed the color code as Red.

The admin is able to add parking wardens into the system by giving valid data for Name, NIC, Mobile, Address and Email. The added parking wardens are displayed on the grid at left hand side of the UI. View, Edit and Delete functionalities also exist for existing parking wardens in the system. Apart from that admin can assign parking wardens to a location using the same interface displayed on Figure 29. By default, the added parking wardens are displaying using the color code White. When assigning a location to a warden, first admin has to select the parking warden and clicking on Add Location icon, the location which are not assigned a warden are list down on the grid below the parking warden grid list as shown in Figure 30.

In the system a particular warden can be assigned to many locations. Once assign a particular location to a parking warden, system gives the color code as Green if the warden has been assigned to only one location. If there are 2 locations assigned, gives the color code as Orange and if there are more than 3, gives the color code as Red. Figure 29 displayed all three color codes.

And also, there is a search implemented on the grid. Admin can search parking wardens according to the Name, NIC and Mobile of the warden.

Figure 29 shows the parking warden setting interface.

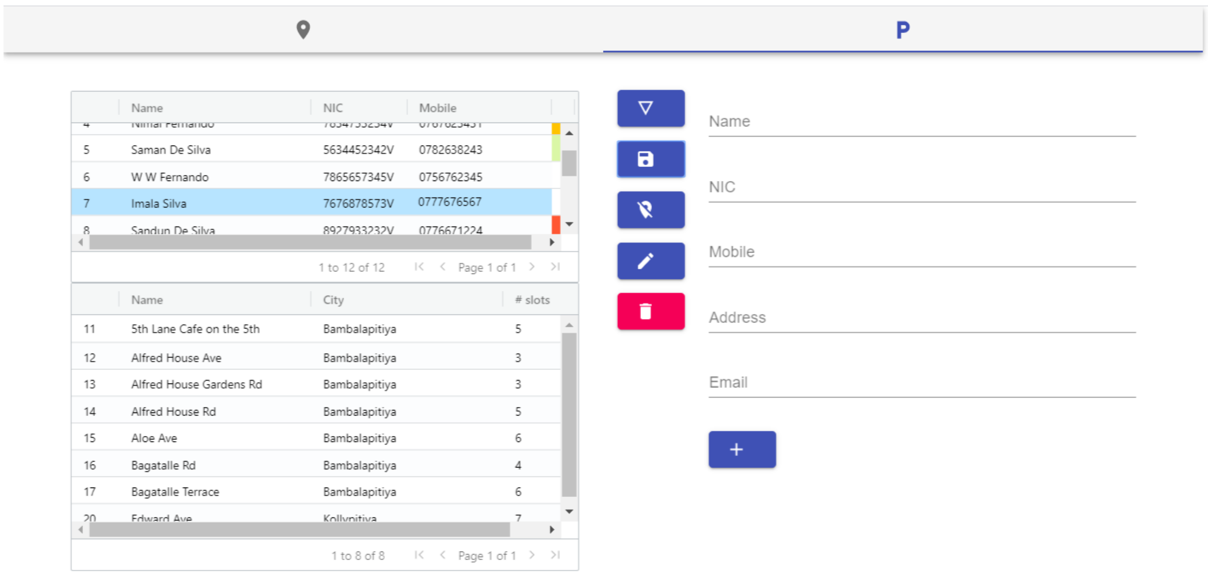


	Name	NIC	Mobile	
3	Amal Perera	7342322234V	0788742831	🟩
4	Nimal Fernando	7834753234V	0767623431	🟧
5	Saman De Silva	5634452342V	0782638243	🟩
6	W W Fernando	7865657345V	0756762345	
7	Imala Silva	7676878573V	0777676567	
8	Sandun De Silva	8927933232V	0776671224	🟥
9	James Fernando	7872346225V	0737264343	
10	William Perera	657384934V	0767346535	
11	Oliver Perera	673748395V	0116374833	
12	Lucas Gomas	556734824V	0778273816	
13	Matthew Perera	783745634V	072837248	
14	Joseph Perera	662873482V	0788327443	

1 to 12 of 12 Page 1 of 1

Figure 29 : Parking Warden UI

Following Figure 30 shows the interface when admin select a warden to assign a particular location/s. The grid below the warden show the available parking locations which are not assigned to a particular warden



	Name	NIC	Mobile	
4	Nimal Fernando	7834753234V	0767623431	🟧
5	Saman De Silva	5634452342V	0782638243	🟩
6	W W Fernando	7865657345V	0756762345	
7	Imala Silva	7676878573V	0777676567	
8	Sandun De Silva	8927933232V	0776671224	🟥

1 to 12 of 12 Page 1 of 1

	Name	City	# slots
11	5th Lane Cafe on the 5th	Bambalapitiya	5
12	Alfred House Ave	Bambalapitiya	3
13	Alfred House Gardens Rd	Bambalapitiya	3
14	Alfred House Rd	Bambalapitiya	5
15	Aloe Ave	Bambalapitiya	6
16	Bagatalle Rd	Bambalapitiya	4
17	Bagatalle Terrace	Bambalapitiya	6
20	Edward Ave	Kollupitiya	7

1 to 8 of 8 Page 1 of 1

Figure 30 : Parking warden UI to assign Locations to the selected warden

4. Evaluation

4.1 User evaluation, Testing and Results

To evaluate how the system, fulfill the objectives, there can be different types of evaluation methods need to be considered. A user-based evaluation is considered in evaluating the system.

What is evaluated?

How accurate the parking app facilitates the user to find a free parking slot?

How it gives the search results for the given parking location area?

What is the purpose of the evaluation?

Ultimately achieve the project goals and the objectives is the purpose of the evaluation. People who are searching for a free parking space in busy cities is becoming a major problem in rush hours. Therefore, it should be needed that the goals are evaluated and achieved properly or not.

Who is interested in the evaluation?

People who interested in using the developed application are the relevant parties who would like for the evaluation. It can be considered that the project can be extended to deliver, If the project is successfully achieving the objectives and the goal.

How will they use the findings?

People who find free parking slots can use the findings of the evaluation result in finding the parking location. How a particular parking user finds a parking area with free slots.

What questions should be answered?

- Find free parking spaces from the user‘s current location in terms of distance and travel time
- Guide vehicles to the selected parking space through the shortest possible path by navigating the application for google maps.
- Save time of finding free parking slots
- What kinds of enhancements can be applied to improve the efficiency and other non-functional factors of the developed application.

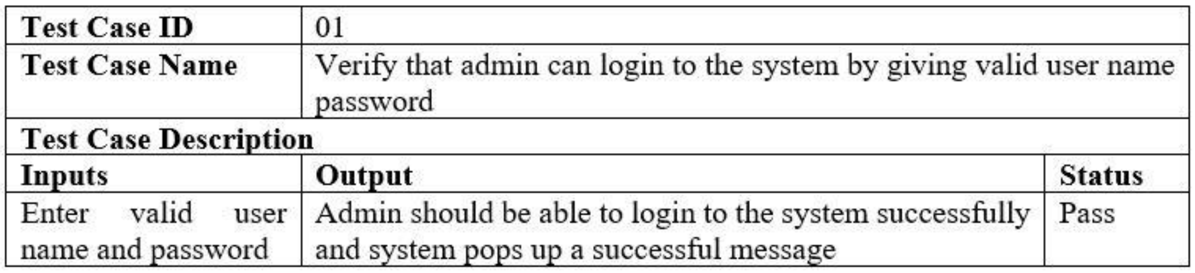
Software testing is a critical component of software quality assurance that represents the ultimate analysis of specification, design, and code generation of software product. The testing method is basically combined with Verification and Validation. Validation refers to testing whether the system satisfies the requirements while verification refers to whether the system implements the specified functions properly. Basic goals of test evaluation are determining whether the promises about the invention by the supplier and the requirements of the customer are met on an acceptable level [17].

Testing begins with the implementation; code is reviewed while developing stage for testing. Test plan included all phases of testing and also used as a guide for the overall testing process. Before the system implementation, the test plan was designed. A test plan includes: test objectives, schedule and logistics, test strategies and especially test cases.

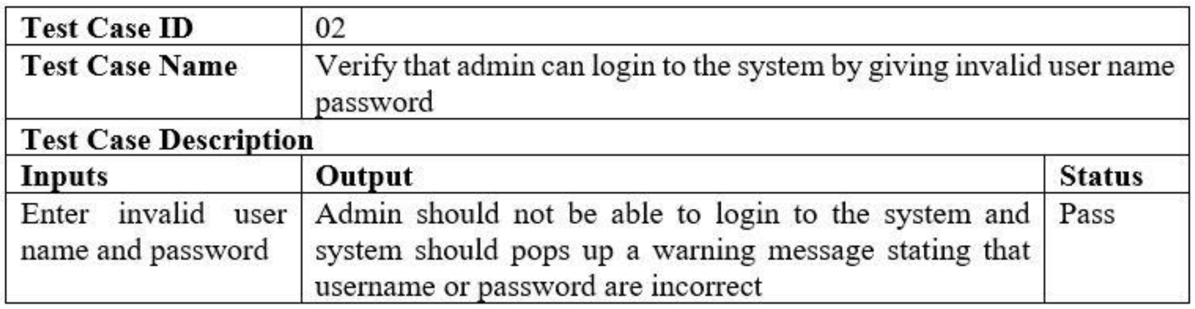
Test cases were created according to the designed test plan. That contains data, procedure, and expected result and represents which use to system or part of the system run. To reduce complexity of the testing process test cases were designed for each module independently. The following tables specify some test cases.

Manual testing method and procedures used for testing rather than automation tools and technologies.

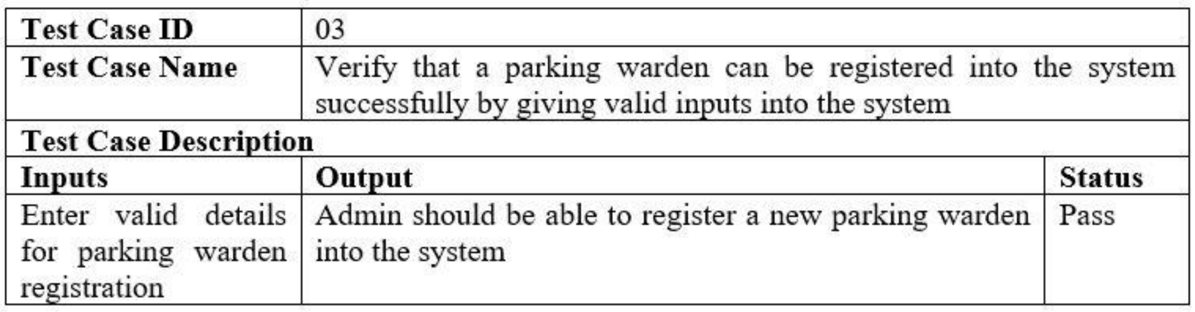
Test cases for web client functions.



Test Case ID	01	
Test Case Name	Verify that admin can login to the system by giving valid user name password	
Test Case Description		
Inputs	**Output**	**Status**
Enter valid user name and password	Admin should be able to login to the system successfully and system pops up a successful message	Pass



Test Case ID	02	
Test Case Name	Verify that admin can login to the system by giving invalid user name password	
Test Case Description		
Inputs	**Output**	**Status**
Enter invalid user name and password	Admin should not be able to login to the system and system should pops up a warning message stating that username or password are incorrect	Pass



Test Case ID	03	
Test Case Name	Verify that a parking warden can be registered into the system successfully by giving valid inputs into the system	
Test Case Description		
Inputs	**Output**	**Status**
Enter valid details for parking warden registration	Admin should be able to register a new parking warden into the system	Pass

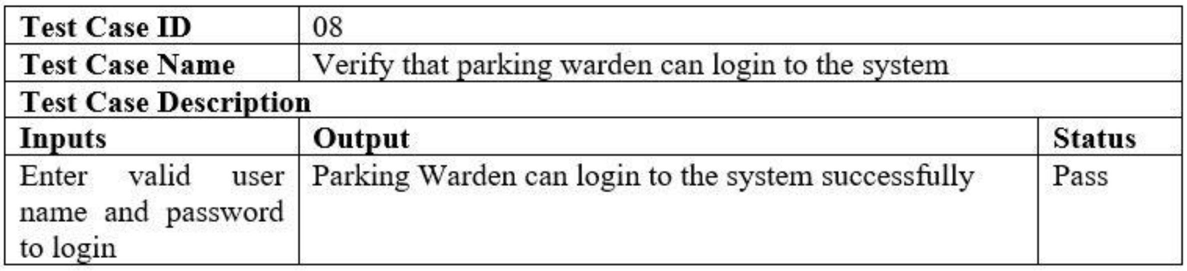
Test Case ID	04	
Test Case Name	Verify that a parking warden can be registered into the system successfully by giving invalid inputs into the system	
Test Case Description		
Inputs	**Output**	**Status**
Enter invalid details for parking warden registration	Admin should not be able to register a new parking warden and system should validate the fields	Pass

Test Case ID	05	
Test Case Name	Verify that admin can add new parking locations into the system	
Test Case Description		
Inputs	**Output**	**Status**
Enter valid details for parking location registration	Admin should be able to add new parking locations and system should validate the data on fields	Pass

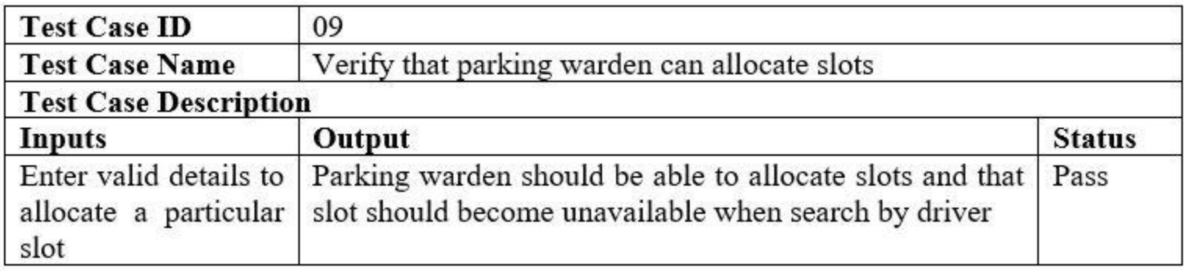
Test Case ID	06	
Test Case Name	Verify the color codes according to the number of locations assigned to the parking warden	
Test Case Description		
Inputs	**Output**	**Status**
Enter valid details for parking location registration	Admin should be able to view the color codes displayed according to the number of locations assigned	Pass

Test Case ID	07	
Test Case Name	Verify that admin can edit no of slots assigned to a particular location	
Test Case Description		
Inputs	**Output**	**Status**
Enter valid details for parking location registration	Admin should be able to view the color codes displayed according to the number of locations assigned	Pass

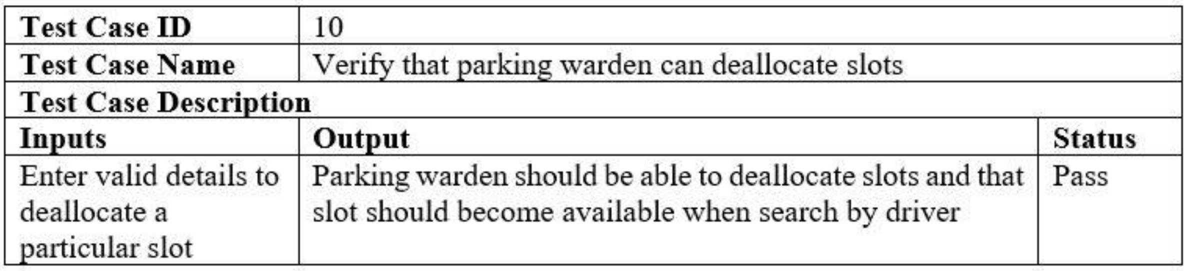
Test Cases for Mobile Client



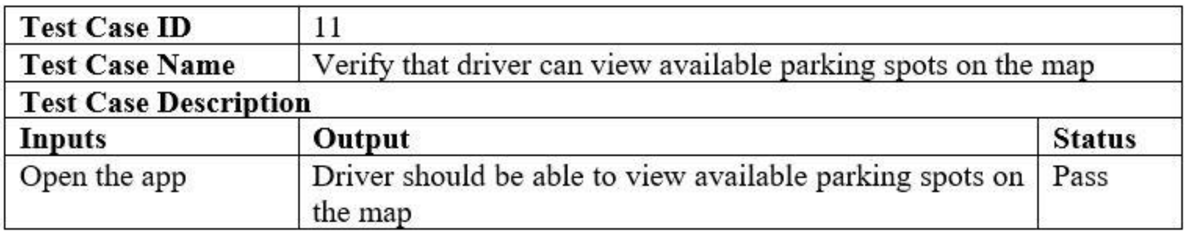
Test Case ID	08	
Test Case Name	Verify that parking warden can login to the system	
Test Case Description		
Inputs	**Output**	**Status**
Enter valid user name and password to login	Parking Warden can login to the system successfully	Pass



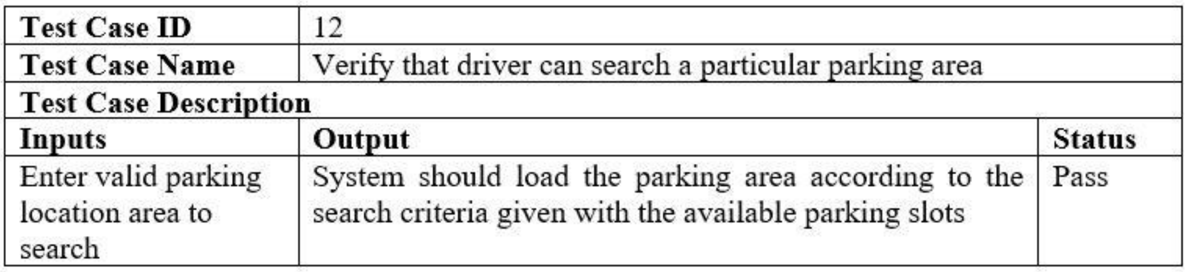
Test Case ID	09	
Test Case Name	Verify that parking warden can allocate slots	
Test Case Description		
Inputs	**Output**	**Status**
Enter valid details to allocate a particular slot	Parking warden should be able to allocate slots and that slot should become unavailable when search by driver	Pass



Test Case ID	10	
Test Case Name	Verify that parking warden can deallocate slots	
Test Case Description		
Inputs	**Output**	**Status**
Enter valid details to deallocate a particular slot	Parking warden should be able to deallocate slots and that slot should become available when search by driver	Pass



Test Case ID	11	
Test Case Name	Verify that driver can view available parking spots on the map	
Test Case Description		
Inputs	**Output**	**Status**
Open the app	Driver should be able to view available parking spots on the map	Pass



Test Case ID	12	
Test Case Name	Verify that driver can search a particular parking area	
Test Case Description		
Inputs	**Output**	**Status**
Enter valid parking location area to search	System should load the parking area according to the search criteria given with the available parking slots	Pass

This section presents the final outcome results of the application where according to the slot allocation of the parking availability by the warden, how driver can view the available parking slots in the map. In Figure 31 shows available parking locations in the area where the user's current location shows a blue color dot in the map.

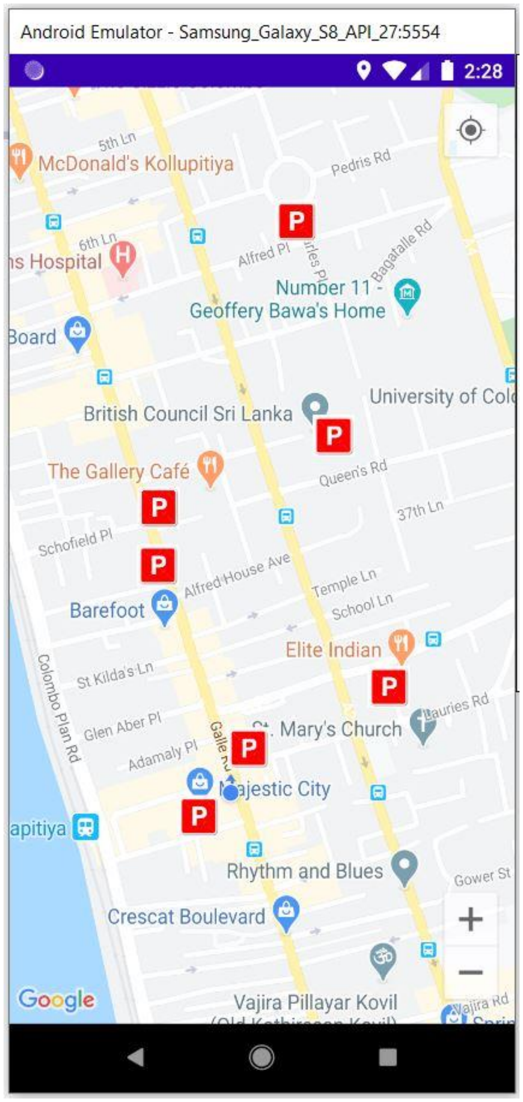


Figure 31 : Load map with all available parking slots nearby

Driver can view how many slots are available in a particular location by selecting the location. Application shows the location name and the number of slots available in that selected location. At that time if the parking warden updated the slot availability, the data should be synced on the driver app accordingly. Using the number of parking slots available, the user can get an idea about the probability of getting loss of a available parking slot in that location.

The Figure 32 the details of the selected parking location by driver.

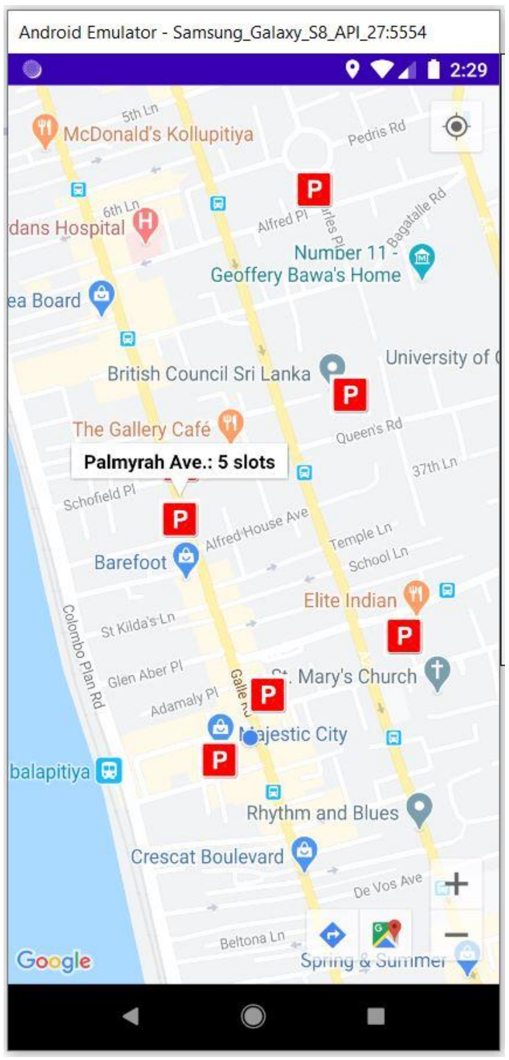


Figure 32 : Details on selected location

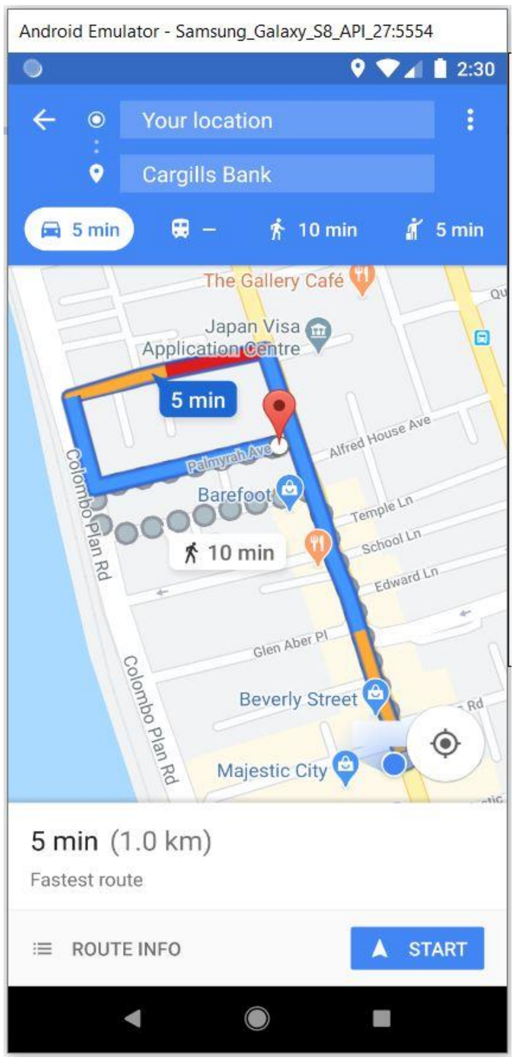


Figure 33 : Directed to google maps

Once user click the direction, application redirects to the google maps and driver can get the fastest route from the map along with the distance and the time to get into that location.

4.2 Critical analysis of work

When analyze the work carrying out during this project there were some challenges identified.

Some of the challenges faced during the system development phases are listed below:

- Design

  Different technologies were used to meet the requirements of the project since there is no one technology that addressed all the needs of the system. Discussions made with the supervisor when narrow down the earlier project requirements.

- Implementation

  The implementation of the parking application requires huge knowledge of back end development languages like java spring boot, Android and database development side of MySQL, front end development languages like REACT, JavaScript. And send API calls through JSON requests.

  The time to get familiar into these programming languages is usually high since the time taken to learn the real time communication between those languages is quite deep.

- Deployments/Installation

  When setting up the development environment which matches the above mentioned programming languages changing PostgreSQL to MySQL was suitable.

- Testing and Evaluation

  Bug fixes and verifying them on the development environment sequentially.

  Getting feedback from the end users about the application when searching for a free parking slots during rush hours.

4.3 Conclusion

The developed automated parking application is a solution to reduce driver's frustration by providing accurate information about the available free parking slots in current area. In Sri Lanka there are lot of private parking areas where the management of those parking areas are doing manually.

The primary objectives through interviews identified that there are main problems associated with parking areas specially in Colombo area,

- ❖ Finding free parking slots in open parking streets takes a lot of time
- ❖ Cost of parking fees in private parking areas are high, due to that people tend to move to open street parking in Colombo city limits
- ❖ Traffic congestion is high during rush hours

The system reduces the work of manual parking effort and reduces the searching time taken by drivers to find free parking slots. This will automatically reduce environmental pollution, traffic congestion and fuel consumption. Then the efficiency of transportation will increase.

Understanding the requirements and finding the best technologies to implement was a challenging task. The system consists of three main components for Admin user, Driver and Parking Warden.

Admin user can manage parking locations and wardens through a web application. At the same time android based mobile applications are developed separately for parking warden and for the driver. Parking warden can update slot availability according to that the available slot data are synced to the driver app accordingly. By clicking on a particular location in the map, the driver can view the location details along with the available number of slots. Selecting the required location in the map driver can get the directions by navigating to the google maps.

A successful implementation of this project would result in less traffic in crowded parking spaces specially like open street parking areas in Colombo city limits. It provides drivers, as it would reduce the wasting time, long queues, tension, stress and increase the efficiency of the parking system. And also using this parking application can track out who has paid for the parking meter which are maintained by the government that would be easy for the work carried out by the parking warden.

4.4 Future work

There is a possibility to do more improvements even though the project objectives were achieved. This automated parking system can be further upgraded by including features like,

- From the driver's point of view, allow an option to reserve a parking slot before reach into the location.
- Making a payment for the reservation will reduce the unnecessary bookings.
- Make payments charge hourly.
- Tracking vehicle details will allow to find out what are the vehicles that reached in/ out to the location
- Scanning the vehicle which are coming in/out from the location and automatically update the slot availability which reduce the manual work done by the parking warden who is allocated for a particular location.
- Automate the slot allocation by getting GPS location.
- Analyze data and make some predictions using history data through which user can get suggestions and recommendations on parking spaces and the availability trends and make available more parking slots on popular parking areas.
- Send alerts when adding new parking locations into the system.
- Make notify and send SMSs to the user once payments are done for parking reservations.
- Notify driver about the occupancy status of the parking spaces.

5. References

[1] K. D. P. P. D. Abhishek Mitra, "Android Based Smart Parking Reservation," *International Journal of Innovative Research in Computer,* 2016.

[2] R. A. N. G. A. B. Supriya Gatalwar, "ParkSmart: Android Application for Parking System," *IJCSN International Journal of Computer Science and Network,* 2016.

[3] S. D. Renuka R., "ANDROID BASED SMART PARKING SYSTEM USING SLOT," *ARPN Journal of Engineering and Applied Sciences* , vol. 10, no. 7, 2015.

[4] J. Liang, "“ParkinVT”, A Concept Mobile Application," *Thesis submitted to the faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of Master of Fine Arts in Creative Technologies,* 2017.

[5] L. K. Mark Lawrence, "Parking Panda," a SpotHero Company, 2011. [Online]. Available: https://www.parkingpanda.com/.

[6] L. K. Mark Lawrence, "SPOT HERO," 2011. [Online]. Available: https://spothero.com/.

[7] A. D. D. R. J. J. Yona Shtern, "Best Parking powered by PARKWHIZ," ParkWhiz company, 2007. [Online]. Available: https://www.bestparking.com/.

[8] M. d. Vries, "ParkMobile," Parkmobile Group, 2000. [Online]. Available: https://parkmobile.io/.

[9] R. T. Fielding, "Representational State Transfer (REST)," *Architectural Styles and the Design of Network-based Software Architectures,* 2000.

[10] J. Freeman, "What is JSON? A better format for data exchange," 2019. [Online]. Available: https://www.infoworld.com/article/3222851/what-is-json-a-better-format-for-data-exchange.html.

[11] K. C. N. O. Dashrath Mane, "The Spring Framework: An Open Source Java Platform for Developing Robust Java Applications," *International Journal of Innovative Technology and Exploring Engineering (IJITEE),* vol. 3, no. 2, 2013.

[12] V. S. Kharchenko, "Mobile and hybrid Internet of Things based computing," *ERASMUS+ ALIOT “Modernization Internet of Things: Emerging Curriculum for Industry and Human Applications Domains” (573818-EPP-1-2016-1-UK-EPPKA2-CBHE-JP),* 2019.

[13] "Getting Started with the NDK," 2006. [Online]. Available: https://developer.android.com/ndk/guides.

[14] "Maps SDK for Android," 2018. [Online]. Available: https://developers.google.com/maps/documentation/android-sdk/intro.

[15] M. C. A. M. O. A. Y. Tejal Lotlikar, "Smart Parking Application," *International Journal of Computer Applications* , vol. 149, no. 9, 2016.

[16] S. C. YADAVALLI, "SMART PARKING SYSTEM," *B-Tech, Jawaharlal Nehru Technological University, India, 2014,* 2016.

[17] C. W. Sewagudde, "The design and implementation of a smart-parking system," *Degree Programme in Computer Science and Engineering*, 2016.