SMART PARKING MANAGEMENT SYSTEM FOR INDIAN CONTEXT

Submitted in partial fulfilment of the requirements

of the degree of

Bachelor of Technology

By

Anjali Tiwari (127205)

Avala Ganesh (127209)

Bishal Kumar Shrestha (127213)

SUPERVISOR

Dr. Rashmi Ranjan Rout

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY, WARANGAL

2015-2016

Approval Sheet

This project work entitled

‘**Smart Parking Management System for Indian Context**’

by Anjali Tiwari, Avala Ganesh, and Bishal Kumar Shrestha

is approved for the partial completion of the degree of

Bachelor of Technology in Computer Science and Engineering

Examiners

____________________	____________________	____________________
____________________	____________________	____________________
____________________	____________________	____________________

Supervisor

____

Dr. Rashmi Ranjan Rout

Chairman

____

Dr. Ch. Sudhakar

Date: _______________

Place: _______________

Certificate

This is to certify that the project work entitled "**Smart Parking Management System for Indian Context"** is a bonafide record of work carried out by Anjali Tiwari (bearing roll no 127205), Avala Ganesh (bearing roll no 127209), and Bishal Kumar Shrestha (bearing roll no 127213), submitted to the faculty of Computer Science and Engineering, in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering at National Institute of Technology, Warangal during the academic year 2015-2016.

Project Guide	**Head of Department**
Dr. Rashmi Ranjan Rout	**Dr Ch. Sudhakar**
Assistant Professor	Associate Professor
Department of Computer	Department of Computer
Science & Engineering	Science & Engineering
NIT Warangal	NIT Warangal

Declaration

We declare that this written submission represents our ideas in my own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

____

____

____

(Signature(s))

Anjali Tiwari (127205)

Avala Ganesh (127209)

Bishal Kumar Shrestha (127213)

Date: __________

ABSTRACT

Today, in major cities of India, parking vehicles is a big concern. The improper and unplanned parking system leads to wastage of time, money and fuel. It also is a major factor in traffic congestion and increase in air pollution. For example, the presence of 800 vehicles per hour (veh/h) and illegal parking between two closely signalized junctions for over 18 minutes determine extra CO emissions about 20%. [1]

In [2], the architecture of parking management is implemented by Intelligent Parking Assistant (IPA), which uses magnetic sensors and Tag RFID for authentication of the users and access and availability of the parking space. In [3], intelligent car park management system was designed using data acquired from 3-layer framework of Wireless Sensor Networks (WSN)-based system. Sensor mesh network of the Mote layer monitors the environment of the parking space and transfers the information to the Server layer, which in turn logs the data and also allows the final Client layer to view and analyze the data. The major drawbacks of the above discussed solutions are the use of hardware sensors which increases the cost of implementation and also the lack of prediction algorithm that enables the users to reserve the parking space in advance.

In addition, the idea is to develop a prediction algorithm using the machine learning approach to generate a linear function which forecasts the availability and occupancy of parking spaces.

Contents

1. Introduction

Ministry of Urban Development (MOUD), Government of India has initiated Smart Cities Mission and also have explained that “In the approach of the Smart Cities Mission, the objective is to promote cities that provide core infrastructure and give a decent quality of life to its citizens, a clean and sustainable environment and application of ‘Smart’ Solutions”. MOUD has also listed “efficient urban mobility and public transport” as one of the core infrastructure elements in a smart city and smart parking as a vital part of urban mobility. [4]

However, in major cities of India, parking vehicles is a big concern. Finding a parking space in a crowded area can be a hassle. Riding around to find a parking space has an adverse impact on the congestion in the area. Some estimates suggest that it amounts to around 30% of the traffic. The improper and unplanned parking system leads to wastage of time, money and fuel. It also is a major factor in traffic congestion and increase in air pollution. For example, the presence of 800 vehicles per hour (veh/h) and illegal parking between two closely signalized junctions for over 18 minutes determine extra CO emissions about 20%. [1]

It is quite evident that a smart parking solution, which caters to the India’s urban requirement is a must for the Smart Cities Mission. This report attempts to do a thorough examination of current existing solutions and propose a solution that meets the requirement and is of low cost, high accuracy and dependable.

2. Literature Review

Through this section, several papers found useful and relevant in this research have been summarized. These papers have been categorized into mainly four sections, namely, (2.1) Overview, (2.2) Parking detection & availability, (2.3) Parking forecast & assignment and (2.4) Smart phone based applications. First section gives an overview of several smart parking systems being implemented in the world. Second section deals with various technologies of vehicle detection and estimating current parking occupancy, while, the third section deals with estimation of level of parking at the time of arrival of the vehicle and parking assignment. Final section of the literature review discusses the existing smart phone based applications for parking management and their functionality.

2.1 Overview

The major cities in India are suffering from the problem of parking vehicles. The improper and unplanned parking system leads to wastage of time, money and fuel. It also is a major factor in traffic congestion and increase in air pollution. For example, the presence of 800 vehicles per hour (veh/h) and illegal parking between two closely signalized junctions for over 18 minutes determine extra CO emissions about 20%. [1] In the past year, the number of smartphone users in urban places has increased by 89% and thus this makes the development of an Android app for reservation and booking of parking place an efficient and economical solution.

2.2 Parking Detection and Availability

2.2.1 Sensor based

Sensors are one of the prominent tools used in vehicle detection for parking management. Research in [5] utilizes Fiber Braggs Grating sensors for vehicle detection. This type of sensor is embedded into the ground. As the pressure exerted on the sensor changes, the wavelength of light reflected at the sensor changes as well. With of help of this change, the presence of a vehicle can be detected. In this research, the sensor was placed approximately where the rear wheels of the vehicles would be after parking the spot. The research showed successful implementation of such technology for a prototype of two parking spaces. There are several other research studies which use combination of sensors in detecting vehicle movement and parking spot occupancy. A research by [6] at Sensys Networks Inc., uses a combination of underground and overhead sensors in estimating parking occupancy. This system consists of several underground sensor nodes (covering an area of 3'x 3'), which use Earth's magnetic field to estimate a vehicle passing over them. For a group of such sensors, an overhead access point is also included in this vehicle detection system. Such an access point will analyse the signals collected at its group of nodes and applies several algorithms to detect the possibility of vehicle passing over the sensor. Essentially, each parking spot would have such a sensor under it. Data from these access points is further accumulated over larger network and used in determining characteristics of the system. These systems typically end up using a whole lot of sensors which makes the vehicle detection for parking a costly affair. Research by Caicedo on systems such as PARC (Parking Access and Revenue Control) takes an alternate approach at vehicle detection [13]. Rather than using the sensors to detect each parking space, this system uses them at key locations so as to divide the overall parking area into smaller zones and calculates the parking occupancy over such zones. This does result in loss of information, as the occupancy for each parking spot is no longer available. However, in many cases such level of accuracy is not particularly required. Such alternatives do provide a decent alternative which provides useful information at lesser costs.

2.2.2 Cellphones based

With onset of smart phones, it is becoming easier to share the information. As many smart phones are fitted with a GPS, it has become easier to understand their movements and with their contribution deduce information about traffic, congestion or even parking. As these technologies do not need any specific physical infrastructure in order to function, they turn out to be the cheapest alternatives for parking. With increasing acceptance of crowd sourcing technologies, such alternatives might end up being the thing of the future. Research shown in [7] uses GPS, accelerometer and gyroscope to detect the pattern of user's movement in order to deduce where they are walking or driving. Additionally, it utilizes techniques such as map matching, which basically checks if their movement is indoor or outdoor. By modelling the length of user's steps with help of height and speed, their model determines the mode and path of user's travel. A research done in [8] employs a similar approach in estimating the availability of parking space. They mainly use trained model to estimate the mode of travel and with help of patterns of mode changes (e.g. Car to Stationary to walk), the parking spots are guessed. Again by reversing such patterns, it is car leaving the spot is also modelled. In this research the researchers take this even further by aggregating this estimated parking data over the block group or a particular area and record this data over time. Such data is further utilized in coming up with statistical models estimating possibility of the parking space being occupied. However, these systems do have their own disadvantages, such as reliability of data and even more importantly, the privacy concerns of making the GPS data for personal phones available to a third party. There are certain non-intrusive cell phone based parking detection solutions such as Roadify, as these are more focused on the crowdsourcing part of it that is people have to specifically pick the option that they have parked a car at a place in order for the system to detect. However, depending on large number of people makes the reliability part of it more questionable as not everyone might choose to report their parking status.

2.2.3 RFID based

Essentially, these systems use Radio-frequency identification (RFID) technologies to detect the parking spaces available in the area. This is accomplished by saving information about the id for the vehicle on a microchip with an antenna. RFID tag, as these are commonly known as help identify the entering or leaving vehicles. Overhead scanners detect the information on these tags to record the parking status. Generally three types of RFIDs – (1) Active, (2) Passive and (3) semi- passive, are used. Active tags have their own power source and are considered to be better functioning. Although barcode and RFID tags have certain similarities, RFIDs end up being better choice in practice as unlike barcodes multiple RFIDs can be read at the same time, reducing the time of operation. However, passive tags are more common in parking management systems due to their low price. In [9], after utilizing RFID to collect information about availability of parking, it is updated on the web servers periodically for the help of users. This research discusses an automated parking spot allocation system by dividing this task into four components, namely, (1) Serial Port Communicator, (2) Free slot checker, (3) Parking charge calculator and (4) Free slot viewer. It provides a functionality where users can communicate with the system to get results in form of a text message and they will also be able to reserve a spot with help of a specific text message.

2.3 Parking Forecast and Assignment

After looking at several ways of collecting the parking availability information through the previous sections, it is also important to understand the existing research in utilizing this data to make optimal parking assignment. Research in several approaches in making optimal parking assignments has been covered in this section. One such interesting research is [11], which uses the travelling direction while allocating the parking spot, along with the parking request by the vehicle, in order to reduce congestion in the surroundings by avoiding unnecessary traffic maneuvers. Cellular automata and cognitive radio module has been used to simulate parking conditions in order to come up with an optimal parking location which will ensure the better performance for the overall system.

Generally, while making such allocations, data about currently available parking spaces are utilized. However, in reality, until the user reaches the parking area from his starting point, the space might have been occupied by some other vehicle. Such situation might end up defeating the whole purpose of this solution. Hence, research into forecasting the demand for parking spaces is required to get the possibility of parking spaces at the intended location being filled up before arrival. Examples of such research have also been covered in this section. On the other hand, research depicted in [12] focuses on building a parking management system capable of making smart decisions. They suggest that as leaving the decision of where to park after showing the alternatives to user might lead to imbalance in parking utilization. As higher number of drivers might end up choosing going for the lot with most number of parking spaces in the vicinity, making this decision for the drivers based on parameters such as parking availability and parking cost in order to make sure the effective utilization of available parking framework, in a way similar to the Nash equilibrium. Hence to achieve these objectives, researchers have proposed a mixed integer linear program (MILP) problem to determine optimal parking location and reserve that space for the user.

For predicting the parking space availability in real time, research such as the one shown in [13] uses discrete choice models to simulate the demand and based on its probability of availability, allocates the demand across the parking infrastructure. It mainly uses two algorithms in this process – (1) Real-time Availability Forecast (RAF) and (2) Parking Request Allocation (PRA). After forecasting the demand based on real time and historical data with help of discrete choice models, RAF uses this demand to generate probabilities for the space becoming empty in the future based on Gamma distribution. PRA uses it to allocate the parking requests. Essentially this model follows the iterative process of allocation, future departure estimation and availability forecasts. After running the model on two scales – aggregate as well as individual, authors did not find significant differences in the computing time. However, the computing time for these processes was indeed found to be really high. If the demand forecasting and allocation process takes a long time it defeats the purpose of doing real time allocation. Alternatively, if higher computing power is used, resources might get stuck in this process.

Research depicted in [13] evaluates another alternative method for forecasting demand for parking shown in [10].This method estimates possible availability of parking spaces with help Markov chain models and queuing theory by considering the historical data and expected time to reach destination. In analysis shown in [14], authors mention that the model works well with controlled access parking lots by producing decent computing times. In this evaluation, authors have employed theoretical methods to prove some assumptions held in the model presented in [13] to compare the effectiveness of the allocation models and computing time for the algorithms used.

2.4 Smart Phone based Applications

This section attempts to give an introduction to the existing research in the smartphone based applications and also to some existing parking management solutions. Research shown in [15] discusses the complete framework of having a smart phone based user interface for parking management systems and the advantages resulting from it. These researchers made an attempt to implement a complete system consisting of web applications for parking management staff and web or smart phone based applications for users. It also provides a way for parking controllers to manage the parked vehicles and help them in revenue collection and enforcement. Functions such as enabling users to monitor their parking activity by keeping them informed about the parking duration, utilization of algorithms such as Ray casting algorithm along with GPS precision, integration with apps such as Google maps & Foursquare and also enabling users to participate in crowdsourcing activities are some of its strong suits.

Examples of existing solutions:

1. ParkMobile: This mobile based app displays the data from parking meters and enables the user to enter the parking id or parking meter number and add time on the meter.
2. SFPark: This application was developed as a prototype for San Francisco in order to improve the parking utilization in the city. Other than displaying the available spaces and being able to pay online, this app enables to have a dynamic pricing module to improve parking utilization.

3. Report on Present Investigation

Solution to the parking problem has been suggested in further subsections. This solution has been designed reduce human interaction and hardware requirement to build a low cost and solutions for urban Indian scenario. The solution sets the following deliverables.

- An Android App from which the customer is able to search for parking lots, get information about them and also book them. App will also provide a feature to manage individual booking.
- Low cost QR code based parking lot design
- Prediction algorithm for future availability information
- Software architecture for the backend of the system, which includes database design, events handling and learning mechanism

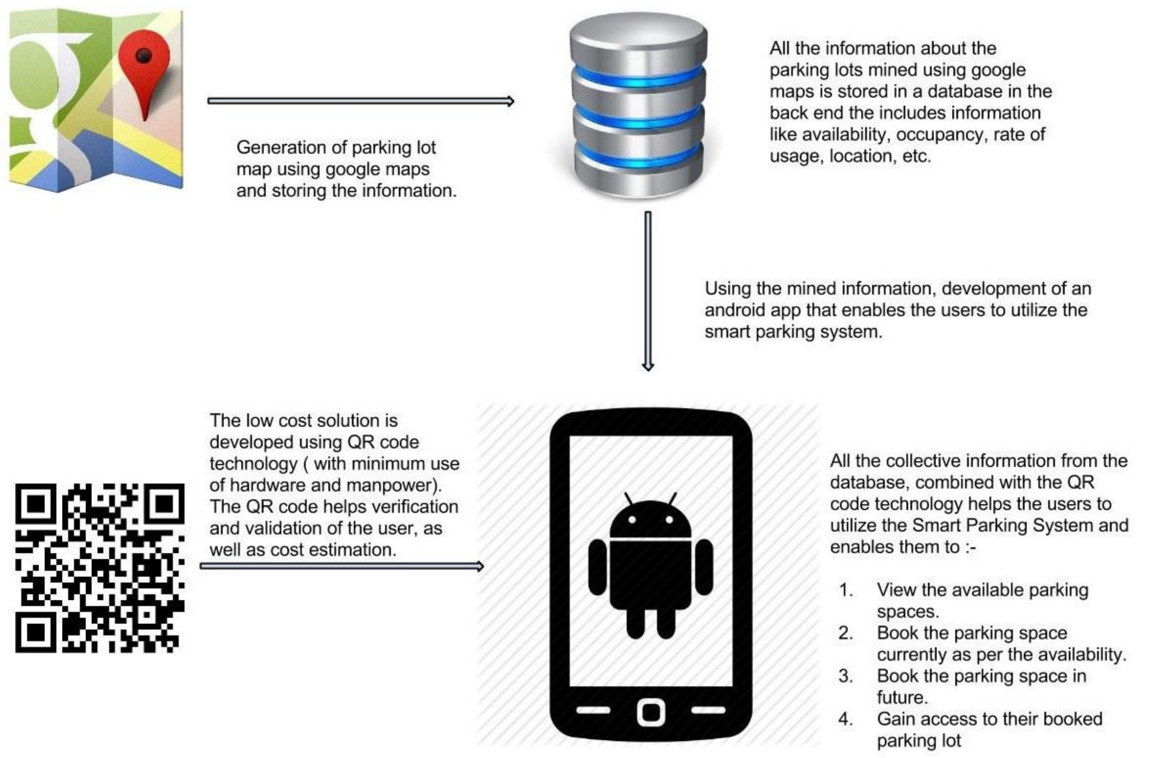

Generation of parking lot map using google maps and storing the information.
All the information about the parking lots mined using google maps is stored in a database in the back end the includes information like availability, occupancy, rate of usage, location, etc.
Using the mined information, development of an android app that enables the users to utilize the smart parking system.
The low cost solution is developed using QR code technology (with minimum use of hardware and manpower). The QR code helps verification and validation of the user, as well as cost estimation.
All the collective information from the database, combined with the QR code technology helps the users to utilize the Smart Parking System and enables them to :-
1. View the available parking spaces.
2. Book the parking space currently as per the availability.
3. Book the parking space in future.
4. Gain access to their booked parking lot

Fig: Flowchart representing the proposed solution

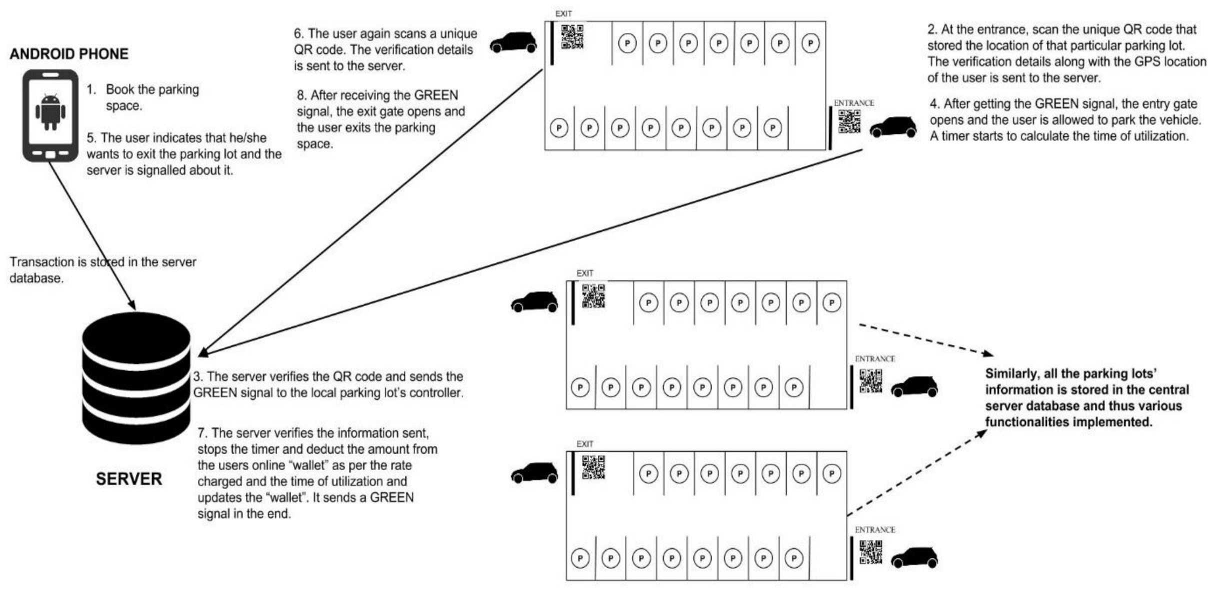

ANDROID PHONE
1. Book the parking space.
5. The user indicates that he/she wants to exit the parking lot and the server is signalled about it.
6. The user again scans a unique QR code. The verification details is sent to the server.
8. After receiving the GREEN signal, the exit gate opens and the user exits the parking space.
EXIT
ENTRANCE
2. At the entrance, scan the unique QR code that stored the location of that particular parking lot. The verification details along with the GPS location of the user is sent to the server.
4. After getting the GREEN signal, the entry gate opens and the user is allowed to park the vehicle. A timer starts to calculate the time of utilization.
Transaction is stored in the server database.
3. The server verifies the QR code and sends the GREEN signal to the local parking lot's controller.
7. The server verifies the information sent, stops the timer and deduct the amount from the users online "wallet" as per the rate charged and the time of utilization and updates the "wallet". It sends a GREEN signal in the end.
SERVER
Similarly, all the parking lots' information is stored in the central server database and thus various functionalities implemented.

Fig: Flow diagram showing use-case of a normal transaction

3.1 Objectives of the proposed solution

3.1.1 User side Android app development

The aim of the project is to enable the public with an application that solves the parking issues of a city and is easily accessible and mobile. Thus, the application is to be developed on Android platform. The app allows the users to search for parking space, provide them with space availability information and also lets them book a space in future from their mobile phone.

3.1.2 Parking space availability maintenance using QR codes

A smart parking car management system requires a parking lot with minimal or no human interaction. As the human involvement is limited, the utilization of automation is increased. This involves a parking lot which automatically verifies a customer, allows or denies entrance to the lot and successfully acknowledges the system about a leaving customer at minimal cost.

The parking lot has an entrance which consists of a mechanical parking gate arm which allows a customer to enter the parking lot as soon as s/he is verified to be the actual person who has booked a space in the parking lot. The mechanical parking gate arm has a QR code which is unique to that parking lot. The uniqueness of the QR code for a parking space is maintained as the QR code stores the Google map location of that particular parking space.

The verification is a two-step process. First, the customer (who has already pre-booked a space) is required to scan the QR code using his smartphone, via the Android App. This QR code is used to signal the system that the customer has arrived at the parking lot. Second, the GPS location of the user needs to be sent to the system, so to confirm that the QR code scanned has actually been done on the particular parking lot. This also prevents malicious users trying to trick the system.

During the exit, similar process follows. The entire process is timed and used for cost estimation.

3.1.3 Parking space management and cost estimation

Each user needs to be informed about the availability of the parking space. An unplanned parking system leads to wastage of time, money and fuel. It is also a major factor in traffic congestion and increase in air pollution. The precise knowledge of the availability of parking space and cost helps the users to choose the nearest and most economical option efficiently.

The management of the system is done in the backend using three tables, i.e., Registered Users, Parking Spaces and Transaction table. The availability and occupancy of each parking lot is stored in the database and modified accordingly. All the reservations are stored in the transaction table. A timer is set to measure the time of utilization of the parking space which starts and stops after the QR code verification during entry and exit. This enables the cost estimation based on the time of utilization.

3.1.4 Data Collection Module

The development of the entire app depends on the availability of the data. Here, data refers to information regarding all parking places available in the city. This information includes name, address, geolocation, capacity and rates of the parking places. However, these data are currently not available with us. So the development of a data collection module is required.

The data collection module is proposed to use Google Places API Web Service to mine all the public places of a particular city. Google Places API Web Service requires the acquirement of a server key from where all the API calls are to be expected. This is easily acquired. Then an arbitrary center is selected within the city and then a radius covering the entire city is provided to get information of potential parking places within that radius. We can achieve parking place name, address, geolocation and its Google place identifier (place id). These retrieved information is considered as unverified data. Further verification needs to be done which involves initially checking if the place actually has a public parking place and secondly, getting the capacity and rates information.

Two APIs within Google Places API Web Service are used.

1. Radar Search Requests. This API call is used to get the place id of all public places of a particular geo-location and a radius around it. The following types of public places are retrieved. These are also the parameters given to the search request.
   - parking
   - airport
   - hospital
   - shopping_mall
   - amusement_park
   - zoo
   - train_station
   - stadium
   - movie_theater
2. Place Details Requests. This API call is used to get the details of the results retrieved in radar search request using the place ids.

After the data is retrieved in place details request, the information is inserted into the database.

Hurdles on the proposed solution:

Google Places Api Web Service allows a maximum of 200 search results per Radar Search.

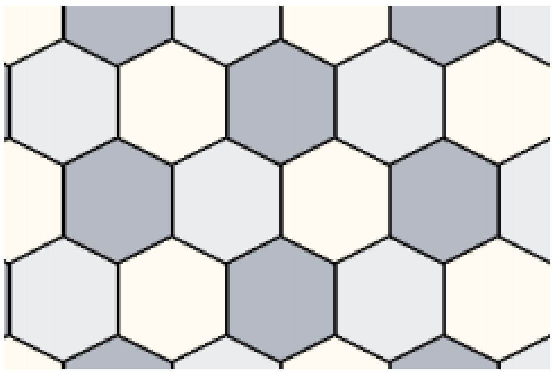
Google Places Api Web Service allows a maximum of 1000 api calls in 24 hour timespan, but by verifying identity by activating billing, maximum api calls allowed can be increased to 150000 api calls. No additional cost is charged to use the api. So this hurdle can be easily solved.

Improvements on the proposed solution:

Among the above mentioned hurdles, the hurdle of bigger issue is the first one. As a city will have thousands of potential parking places, a single radar search yielding 200 search results isn’t feasible.

To overcome this problem, several smaller radar searches of the city can be done with smaller radius per search, which practically will not have more than 200 potential parking places. Now the main problem comes down to divide the city (a flat surface) into equal sized circular zones (for each radar search), so that they have minimal intersection and thus yield less number of collisions (i.e. a same place coming up twice in two separate radar searches with adjacent, intersecting regions).

The closest approximation of this can be achieved by dividing the city (flat surface) by regular hexagonal tessellation rather than circles. This tessellation is best achieved when the original flat surface is a hexagon itself. Thus, the city itself is considered as a giant hexagon and further tessellation is done on top of it. Then, each radar search can be done assuming the circumcenter of each of the smaller hexagons as the latitude and longitude of the radar search center and the radius of circle circumscribing the same as the radius for the radar search.



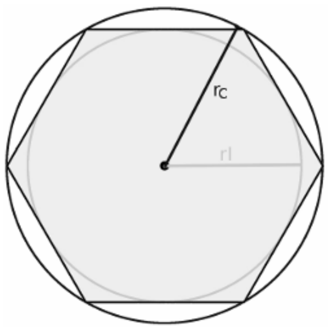


Fig: Hexagonal Tessellation of a flat surface and circumcircle of a hexagon

Hexagon tessellation involves getting the circumcenter and radius of the circumcircle of the hexagons. All of this can be achieved by simply getting the next hexagon with reference to first hexagon’s circumcenter.

Let the first hexagon’s circumradius be r_c and its inner circle’s radius be r_1.

By simple geometry,

$$r1 = rc\,\frac{\sqrt{3}}{2}$$

Then, following the first diagram,

The hexagon on the right of any hexagon would have its circumcenter at $2*r_1$ right of its own circumcenter (horizontal displacement) and no vertical displacement.

The hexagon on the next row would have its circumcenter at r_1 horizontal displacement and $1.5*r_c$ vertical displacement.

Next step of the solution would involve converting the distances and coordinates in terms of latitude and longitude. This is extremely vital as this helps us to directly solve the hexagon tessellation problem in accordance with geolocations and real world distance. It is achieved using the following two formulae.

Distance between two geolocation is calculated using Haversine formula.

$a = \sin^2(\Delta\varphi/2) + \cos\varphi 1 \cdot \cos\varphi 2 \cdot \sin^2(\Delta\lambda/2)$

$c = 2 \cdot \mathrm{atan2}(\sqrt{a}, \sqrt{(1-a)})$

$d = R \cdot c$

where, φ is latitude, λ is longitude, R is earth's radius (mean radius = 6,378.14km)

Note that angles need to be in radians to pass to trig functions.

Given distance, bearing and starting point, the final geolocation is given by the following formula:

$\varphi 2 = \mathrm{asin}(\sin\varphi 1 \cdot \cos\delta + \cos\varphi 1 \cdot \sin\delta \cdot \cos\theta)$

$\lambda 2 = \lambda 1 + \mathrm{atan2}(\sin\theta \cdot \sin\delta \cdot \cos\varphi 1, \cos\delta - \sin\varphi 1 \cdot \sin\varphi 2)$

where, φ is latitude, λ is longitude, θ is the bearing (clockwise from north), δ is the angular distance d/R; d being the distance travelled, R the earth's radius (mean radius = 6378.14 km)

3.1.5 Development of Prediction Algorithm

The proposed “intelligent” parking space inventory control system is based on a linear function of five parameters and gradient descent search through the space of possible hypothesis. In the first step of the proposed model, we determine six major factors that effects the occupancy and availability of a parking space.

Every day a significant percentage of drivers in single-occupancy vehicles search for a parking space. Additionally, less experienced drivers or out-of-towners further contribute to the increase of traffic congestion. Every parking search strategy is composed of a set of vague rules. It is usually difficult to describe these rules explicitly. The type of the planned activity, time of a day, day of the week, current congestion, and potentially available parking places have significant influence on a chosen parking search strategy.

Why do we need Parking Guidance System?

Parking guidance systems usually do not change the occupancy rate or average parking duration. Drivers easily become familiar with the parking guidance systems, and majority of them use, thrust and appreciate the help of the systems. Guidance systems significantly increase the probability of finding vacant parking space, mitigate frustration of the drivers–visitors unfamiliar with the city center, decrease the queues in front of parking garages, decrease the total amount of vehicle-miles traveled (particularly in the city centers), decrease the average trip time, energy consumption, and air pollution. Parking guidance system is a part of comprehensive parking policy and traffic management system, whose other elements are street parking control (including sanctions for the illegally parked vehicles), parking fare structure, and parking revenue management system.

Designing a Learning System for prediction of space availability of Parking Space

In this approach, we keep on collecting the real time data from the database constructed in the back-end.

- **Choosing the Training Experience**

  The first design choice we face is to choose the type of training experience from which our system will learn. The type of training experience available can have a significant impact on success or failure of the learner. One key attribute is whether the training experience provides direct or indirect feedback regarding the choices made by the performance system.

In the proposed system, the training data is basically the real time data collected from the parking transaction table and worked upon against the predicted values.

A second important attribute of the training experience is the degree to which the learner controls the sequence of training examples. For example, the learner might rely on the teacher to provide the real time data or itself will make assumptions and proceed.

A parking prediction learning problem:

- **Task *T***: Predicting the availability of a parking space.
- **Performance measure *P***: Difference between the actual value and prediction value of availability of a parking space.
- **Training Experience *E***: The real time data collection, predicting values for availability and comparing with the actual values to obtain accuracy.

- **Choosing the Target Function**

  The next design choice is to determine exactly what type of knowledge will belearned and how this will be used by the performance program. In the proposed system, we use the "**Predicted Availability**" (PA) target function in order to predict the available space in a parking lot.

- **Choosing a Representation for the Target Function**

  Now that we have specified the ideal target function **PA**, we must choose a representation that the learning program will use to describe the function **PA'** that it will learn. As with earlier design choices, we again have many options. We could, for example, allow the program to represent using a large table with a distinct entry specifying the value for each distinct board state. Or we could allow it to represent using a collection of rules that match against features of the board state, or a quadratic polynomial function of predefined features that effect the parking management system, or an artificial neural network.

  To keep the discussion brief, let us choose a simple representation: for any given parking lot at a particular time, the function will be calculated as a linear combination of the following features:

  - x_1 : Capacity of parking lot
  - x_2: Day of week when the parking space is required

- x_3: Time of day when parking space is required
- x_4: Current availability of parking lot
- x_5: Time difference between the booking time and booked time
- x_6: Number of booking requests made at present till the booked time

Thus, our learning program will represent **PA'** as a linear function of the form:

$$\mathbf{PA' = w_0 + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5 + w_6x_6}$$

where, w_0 through w_6 are numerical coefficients, or weights, to be chosen by the learning algorithm. Learned values for the weights w_0 through w_6 will determine the relative importance of the various board features in determining the value of the board, whereas the weight w_0 will provide an additive constant to the board value.

Partial design of a prediction learning program for parking space:

- **Task *T***: Predicting the availability of a parking space.
- **Performance measure *P***: Difference between the actual value and prediction value of availability of a parking space.
- **Training Experience *E***: The real time data collection, predicting values for availability and comparing with the actual values to obtain accuracy.
- **Target Function:** PA: Parking Space -> R
- **Target Function Representation:**

$$\mathbf{V' = w_0 + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5 + w_6x_6}$$

The first three items above correspond to the specification of the learning task, whereas the final two items constitute design choices for the implementation of the learning program. Notice the net effect of this set of design choices is to reduce the problem of learning a checkers strategy to the problem of learning values for the coefficients w0 through w6 in the target function representation.

- **Choosing a Function Approximation Algorithm**

  In order to learn the target function **f** we require a set of training examples, each describing the availability of space in a parking lot and the training value **V'**. The training example is the real time data obtained from the transaction table of the parking lot.

  Adjusting the weights:

  Now, the task is to specify the learning algorithm for choosing the weights w_i to best fit the set of training examples.)}.

  As a first step we must define what we mean by the best fit to the training data.

  One common approach is to define the best hypothesis, or set of weights, as that which minimizes the squared error E between the training values and the values predicted by the hypothesis V.

$$E \equiv \sum_{(b,PA_{train}(b))\epsilon\ training\ examples}(PA_{train}(b) - PA'(b))^2$$

  Thus, we require an algorithm that will incrementally refine the weights as new training examples become available and that will be robust to errors in these estimated training values. One such algorithm is called **the least mean squares, or LMS** training rule.

  The LMS algorithm is defined as follows:

  LMS weight update rule.

  For each training example (b, PA_{train}(b)) ;where 'b' is input parameters

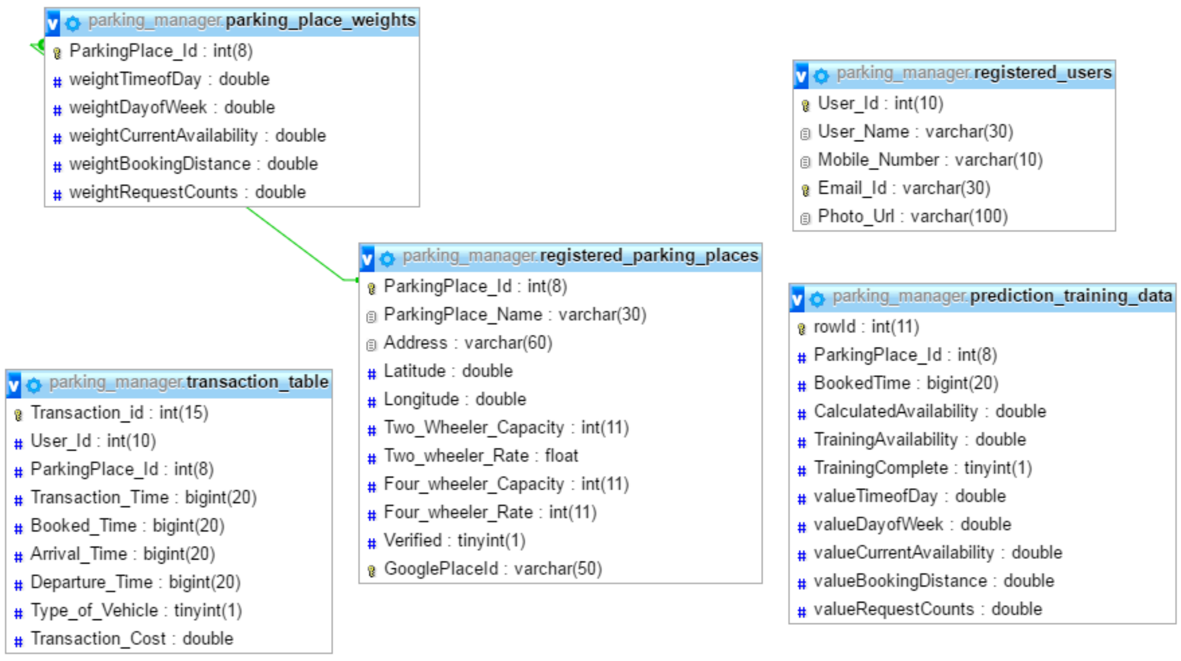
  - Use the current weights to calculate PA'(b)
  - For each weight w_i, update it as

    $w_i \leftarrow w_i + n(PA_{train}(b) - PA'(b))\ x_i$

4. Implementation

The implementation of the proposed solution can be aptly divided into three sections viz. Android App Development, Backend database design and accesses and Prediction Algorithm Design.

4.1 Backend Database Design



parking_manager.parking_place_weights
ParkingPlace_Id : int(8)
weightTimeofDay : double
weightDayofWeek : double
weightCurrentAvailability : double
weightBookingDistance : double
weightRequestCounts : double
parking_manager.registered_users
User_Id : int(10)
User_Name : varchar(30)
Mobile_Number : varchar(10)
Email_Id : varchar(30)
Photo_Url : varchar(100)
parking_manager.registered_parking_places
ParkingPlace_Id : int(8)
ParkingPlace_Name : varchar(30)
Address : varchar(60)
Latitude : double
Longitude : double
Two_Wheeler_Capacity : int(11)
Two_wheeler_Rate : float
Four_wheeler_Capacity : int(11)
Four_wheeler_Rate : int(11)
Verified : tinyint(1)
GooglePlaceId : varchar(50)
parking_manager.prediction_training_data
rowId : int(11)
ParkingPlace_Id : int(8)
BookedTime : bigint(20)
CalculatedAvailability : double
TrainingAvailability : double
TrainingComplete : tinyint(1)
valueTimeofDay : double
valueDayofWeek : double
valueCurrentAvailability : double
valueBookingDistance : double
valueRequestCounts : double
parking_manager.transaction_table
Transaction_id : int(15)
User_Id : int(10)
ParkingPlace_Id : int(8)
Transaction_Time : bigint(20)
Booked_Time : bigint(20)
Arrival_Time : bigint(20)
Departure_Time : bigint(20)
Type_of_Vehicle : tinyint(1)
Transaction_Cost : double

Fig: Database Schema

The backend for the app consists of a database design in order to keep track of the registered users and parking places. It is a central server, which means a single database is used to handle all parking lots, users and transactions. MySQL is the DBMS of choice for this project. It manages the reservation, booking and cancellation transactions. The database includes tables like "RegisteredUsers", "RegisteredParkingPlaces" and "TransactionTable". It has been developed on localhost under XAMPP environment.

1. **"Registered Users"** table: This table stores the details of all the users who have registered and installed the app on their mobile phones. Each user has a unique 10-digit key as "User_Id" which is the primary key. Other information like user_name, Mobile number, e-mail_id and photo url are also stored.

2. **“Registered Parking Places”** table: This table stored the details of all the mapped, surveyed and field verified parking spaces in the city. Each parking place has a unique 8-digit key as “Place_id”, which is also the primary key for this table. Other attributes like Name, Google_Map_Location, Address, 2-wheeler capacity and 4-wheeler capacity of vehicles and their rates (dynamic or static) are also stored in the table.
3. **“Transaction”** table: This table is the table that manages every transaction made by the users of a particular city. A unique key is auto-generated that is the “Transaction_Id” which is the primary key for the table. There are two attributes, i.e., “User_id” and “Place_id”, which are foreign key from the table “Registered Users” and “Registered Parking Paces” respectively. These two key provide information regarding the user and place information. Other attributes such as transaction timestamp, arrival timestamp, departure timestamp, type of vehicle and cost are also stored in the table to manage the parking system. Type of vehicle can have values either 2 or 4. Transaction timestamp is the timestamp when the booking was made and thus has a non-negative long value. Arrival timestamp and departure timestamp values are the timestamps when the user has arrived and departed from the parking place. Their default value is -1 if the user hasn’t arrived or departed yet.
4. **“Parking Place Weights”** table: This table contains is used for the assistance for prediction algorithm. Prediction algorithm predicts parking availability for individual parking lot, without any relation with other parking lots in the database, using linear function and stochastic gradient-descent search. The linear function is characterized by particular parameters and their effectiveness (weights). These weights are stored in this table for individual parking lots.
5. **“Prediction training data”** table: This table is used for training the prediction machine learning algorithm. It consists of all the predicted availability of the parking lot and also the actual availability at the booked time. This data, in combination with the values of the individual parameters in parking lot, is used to train the algorithm by updating the weights in **“Parking Place Weights”** table.

4.2 User status

According to our system, user can be in one of the three predefined status.

1. “notbooked” – User does not have an active booking. This means that user can now select any registered parking place of their choice and then book a parking place.

This means that currently there are either no booking or only completed bookings in the “Transaction” table under the user in question. In the latter case, the user will only have bookings in transaction table which contains both arrival and departure timestamps as some non-negative long value.

2. “booked” – User has booked a space in some parking place but hasn’t arrived at the parking lot to park the car. This means that the user has to go to parking lot at his booked time and park his car.

   The “Transaction” table would contain exactly one transaction under the user with both arrival and departure timestamps as -1.
3. “parkingstarted” – User has reached the parking lot and parked his car, but they haven’t yet completed their parking and thus haven’t left the parking lot.

   The “Transaction” table would contain exactly one transaction under the user with arrival time as some non-negative timestamp value corresponding to the UNIX time when he arrived at the parking lot and the departure time would have the value -1.

When the user completes his parking and leaves the parking lot, his status is again reverted to “notbooked” as the departure time mentioned in the “parkingstarted” status is updated to the non-negative timestamp value corresponding to the UNIX time when he leaves the parking lot.

4.3 Two-step verification for User status changes

Whenever a user status changes from “booked” to “parkingstarted” (i.e. user parks the vehicle) or “parkingstarted” to “notbooked” (i.e. user has completed his parking), there should a verification process required in order to ensure that the QR code being scanned (to enter or exit the parking lot) is being taken in the actual parking lot or some malicious user is trying to attack the system.

This verification is a two-step process. First, the customer (who has already pre-booked a space) is required to scan the QR code using his smartphone, via the Android App. The scanned code, if valid, contains information about the parking lot (parking place identifier and location). This triggers the following series of events.

1. The current GPS location is of the user is acquired.

2. This location is cross-referenced with the location of parking lot to see if the user is in-fact at the parking lot with the allowed error of 40m. This is the first level verification which checks to see if the user is physically at the parking lot.
3. The scanned QR code data is sent to backend which is again cross-referenced with the data at the backend. This is second level verification which checks if the user is actually scanning the QR code at the parking lot.

After all these steps are completed, the user is allowed to pass through the gate of parking lot i.e. their status is changed.

4.4 Android App Development

An app named "Parking Manager" has been developed on Android platform. The IDE used for the develoment of the app is Android Studio. Various libraries and other open source projects have been used to develop the app.

The entire app has the following general flow diagram.

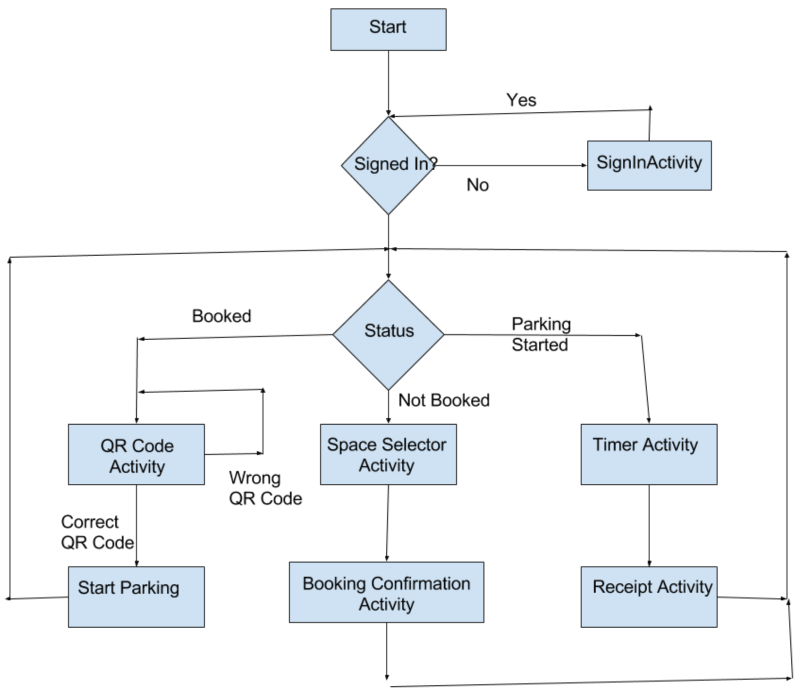


Fig: Flow diagram of Parking Manager App

1. **SignInActivity:** As the App opens, SignInAcitivity is initialized. This activity initially checks the *SharedPreferences* file, “prefs” for a key-value pair which indicates if a user has logged in from this device or not.

   If no user has logged in, then it opens the sign in page. The sign in goes forward with the help of *GoogleSignIn* API. This API helps us get all the user information required ex. name, email etc. This information is stored in both our local device *(SharedPreferences)* as well as our remote database using PHP scripts. If some user has already signed in from this device, then it goes forward to the next step.

   Then the app checks for the user’s current status according to our system.

   If the current status is “notbooked”, then app redirects the user to DateTimePickerActivity for the user to select a booking date and time.

   Else if the current status is “booked”, then app redirects to QRCodeScannerActivity for the user to scan the QR code (refer our proposed solution) when required, which would allow the user to park the vehicle in the parking lot.

   Else if the current status is “parkingstarted”, then app redirects to TimerActivity which informs the user about how long they have parked the vehicle for.

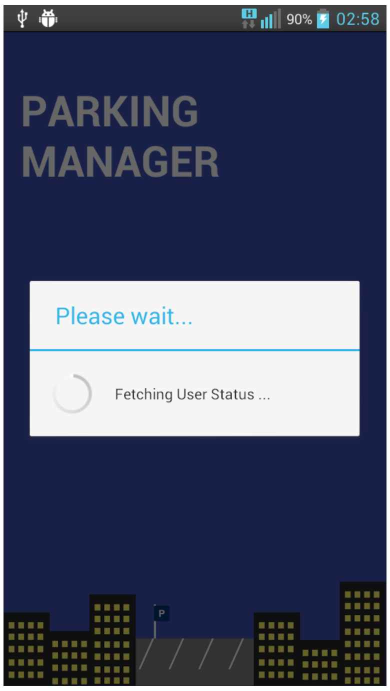


Fig: SignInActivity Screenshot

2. **DateTimePickerActivity:** This activity is used to get the user preference for at what date and time the user requires the parking lot place and for what kind of vehicle (2 wheeler or 4 wheeler). The page uses *material design* for rendering its views. This information is again stored in the *SharedPreferences* file. After the selection is made, the app directs to SpaceSelectorActivity**.**

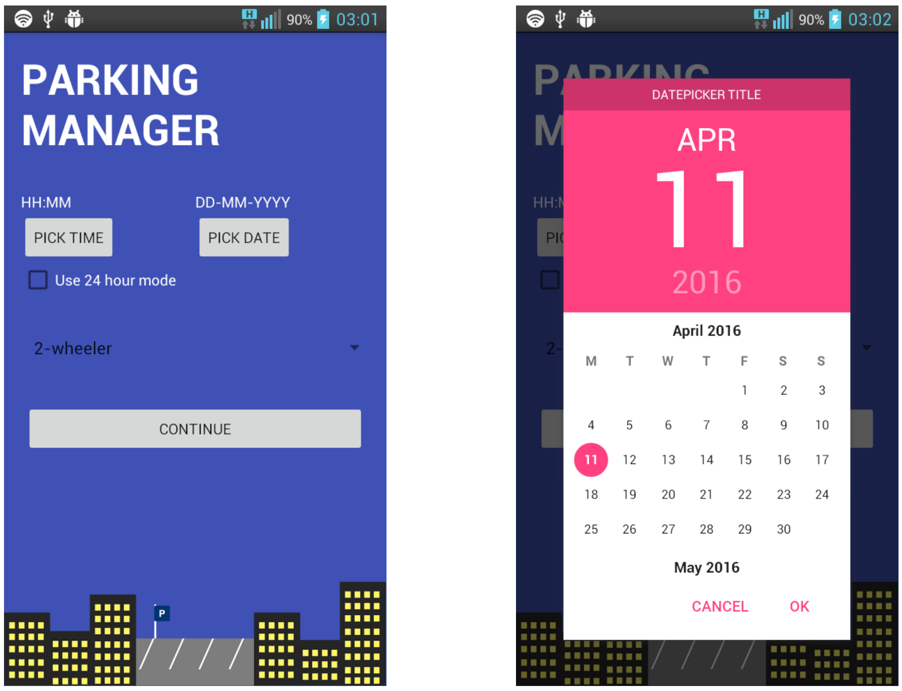


Fig: DateTimePickerActivity Screenshots

3. **SpaceSelectorActivity:** This activity loads the Google Maps. It has custom markers on the map which shows all the available parking spaces that are in our backend. It uses the PHP scripts to get the parking lot information from our remote database and then renders all the information in the map. Once a user selects a parking lot of his choice, then we redirect to BookingConfirmationActivity.

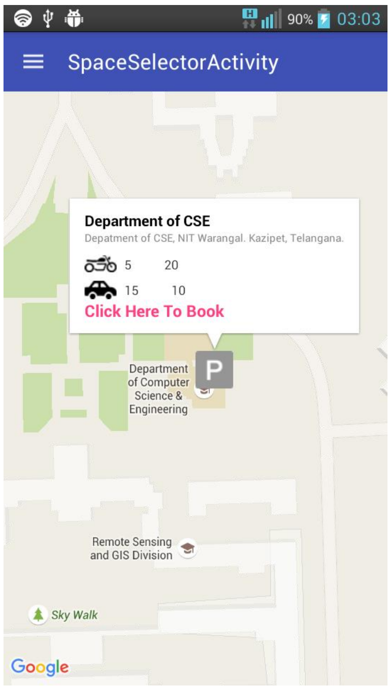
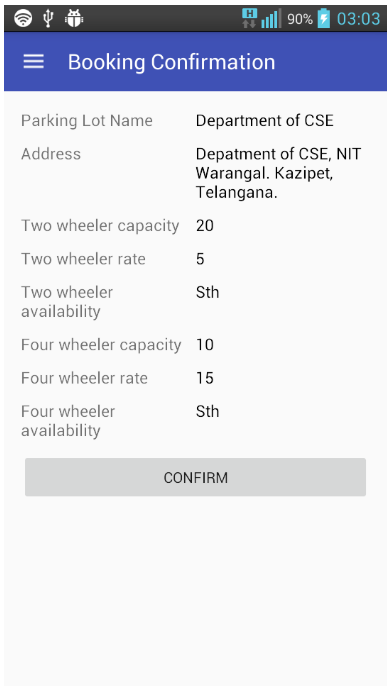

90% 03:03
SpaceSelectorActivity
Department of CSE
Depatment of CSE, NIT Warangal. Kazipet, Telangana.
5 20
15 10
Click Here To Book
Department of Computer Science & Engineering
Remote Sensing and GIS Division
Sky Walk
Google

Fig: SpaceSelectorActivity Screenshot

4. **BookingConfirmationActivity:** This activity contains the summary of all the information of the parking lot selected in SpaceSelectorActivity. A simple confirmation is needed from the user. After the user confirms, the space is booked for the user, the transaction is recorded in our database and the user has a "booked" status. The app then directs to QRCodeScannerActivity.



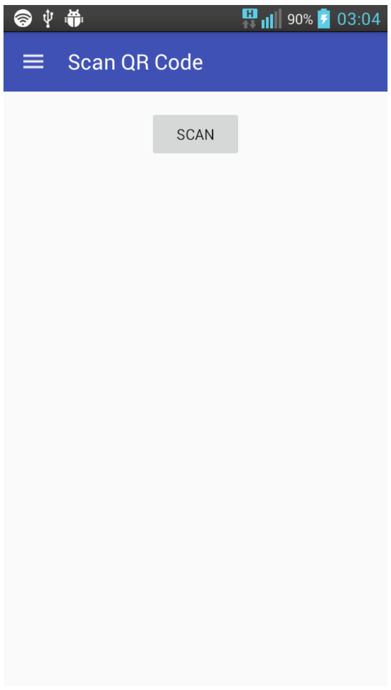
90% 03:03

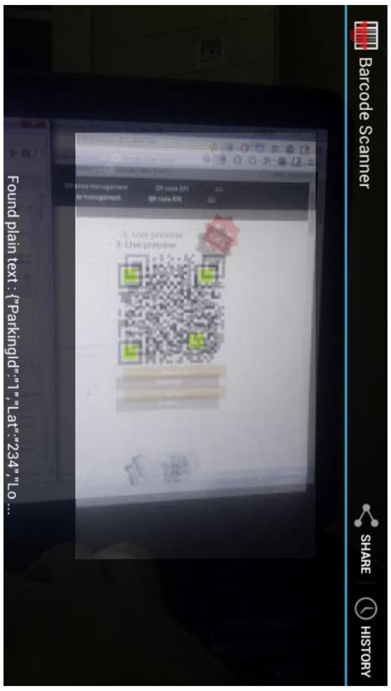
Booking Confirmation

Parking Lot Name	Department of CSE
Address	Depatment of CSE, NIT Warangal. Kazipet, Telangana.
Two wheeler capacity	20
Two wheeler rate	5
Two wheeler availability	Sth
Four wheeler capacity	10
Four wheeler rate	15
Four wheeler availability	Sth

Fig: BookingConfirmationActivity Screenshot

5. **QRCodeScannerActivity:** This activity has a Scan button, which is to be clicked by the user when he reaches the parking lot. This app has a dependency on a third party app called *Barcode Scanner by Zxing*. The first time the Scan button is pressed, it urges the user to install the Barcode Scanner app from the Play Store. On future scans, it opens the app and can now scan QR codes. When it detects a valid QR code, it scans it and returns the acquired information(String) back to our app. QR codes of the parking lots are designed to be in the minimized *JSON* format containing three fields, *ParkingLotId, Latitude and Longitude*. If the scanned QR code doesn't match the mentioned format, then app says the scanned QR code is erroneous. Else, it sends the information, along with userid, and the user's actual *GPS* location to the backend where a PHP script follows two step verification and if successful, then the user is bumped to "parked" status. This then directs to timer activity.





Found plain text : {"ParkingId":"1","Lat":"234","Lo…

Fig: QRCodeScannerActivity Screenshot and Zxing Barcode Scanner

6. **TimerActivity:** This activity simply shows how long it has been since the user has parked the vehicle in the spot. It also has a stop parking button, which again does the same tasks as QRCodeScannerActivity. All the same two step verifications are done. This is done when the user is leaving the Parking lot after completing his parking. This changes the user's status to "notbooked" again and redirects to receipt activity.

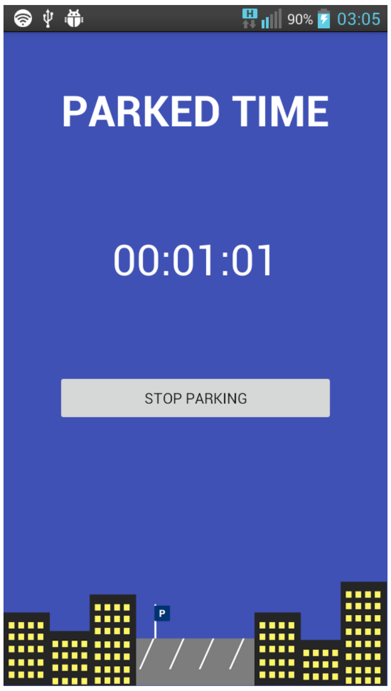
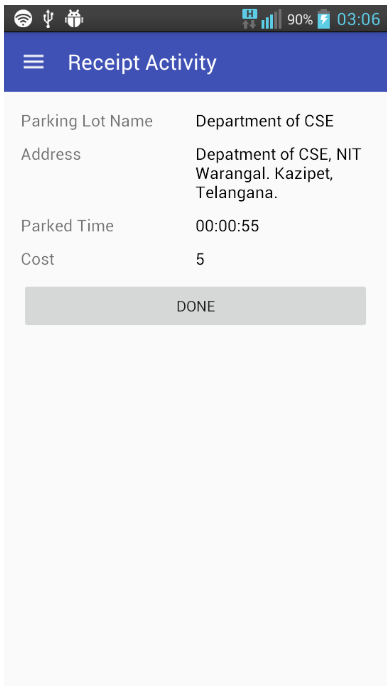


Fig: TimerActivity Screenshot

7. **ReceiptActivity:** This activity just shows the billing amount of the parked car so that the user can pay the amount by any of the method required.



Parking Lot Name	Department of CSE
Address	Depatment of CSE, NIT Warangal. Kazipet, Telangana.
Parked Time	00:00:55
Cost	5

Fig: Receipt Activity Screenshot

4.5 Backend Scripts

The different use cases in the Smart Parking Management System is defined using PHP scripts that run the Back end and are tested against the local database using a Google chrome extension “Postman”.

The following are the use cases defined in the Parking Management System and the PHP scripts used for them.

1. **User Registration:** A user registers in the app and enter the basic details as required. The script verifies the user and also prevents redundant data using the primary field “EmailID”. The PHP file for this use case is “RegisterUser.php”.

2. **Booking the parking place:** The user books a parking place as per his requirement, i.e., location, time, type of vehicle (2-wheeler or 4-wheeler) and budget. The script takes input as “User_Id”, “ParkingPlace_Id”, “Booked_Time” and “Type_of-Vehicle” and enters the booking details in the “Transaction Table” in order to book the parking place for the user. In the “Transaction_Table”, the script adds more information like “Booking_Time”, which is set to current timestamp and other information such as “Arrival_Time” and “Departure_Time” that are set to “-1”.

   Note: Significance of values of “Arrival_Time” and “Departure_Time”

   When both the values are set to “-1” in the “Transaction_Table”, it means that only booking by the user is done and user hasn’t arrived in the parking place yet.

   When “Arrival_Time” is set to some value other than “-1” and “Departure_Time” is equal to “-1” in the “Transaction_Table”, it means that the user has arrived in the parking place but not departed from the place yet. The user is still utilizing the parking place and the background clock is still ticking and calculation the duration of utilization of parking.

   When both “Arrival_Time” and “Departure_Time” are set to some value other than “-1”, it means that the user has departed from the parking place, i.e., the utilization is over. Based on the difference between the “Arrival_Time” and “Departure_Time” and rate of the parking place, the cost of parking is calculated.

   There is no scenario possible in which “Arrival_Time” equals “-1” and “Departure_Time” is some value other than “-1”.

3. **Start Parking:** The user has arrived at the parking place and parked and vehicle. The QR code verification triggers the server and the clock start for the parking of the vehicle and continues unless “Stop Parking” is requested and QR code is verified again. The “StartParking.php” script sets the “Arrival_Time” in the “Transaction_Table” to the current timestamp as per the utility specified above.

4. **Stop Parking:** The user has finished the parking utilization and has requested to stop the parking. The QR code verification triggers the server and the timer stop. The “StopParking.php” script sets the “Departure_Time” to the then current timestamp value. Based on the difference between the “Arrival_Time” and “Departure_Time” the cost of parking utilization is deduced.

5. **Get Parking Place Details:** This PHP script is used in order to view all the available and verified parking places in an area on the Google map viewed via the app. The script simply queries the “Registered_Parking_Places” table and displays all the parking places with details.

6. **Get Time:** This PHP script is used to obtain the “Arrival_Time” from the “Transaction_Table” in the local device in case there occurred an error and network is lost during the parking of the vehicle when user has started the parking but not requested to stop.

7. **Parking Place Info:** This PHP script returns the details of a particular parking place based on the input “ParkingPlace_Id”. This is used in the app when a user selects a parking place point on the map and the corresponding details are displayed. In the backend, the PHP script runs the simple query on the “Registered_Parking_Place” table and view the details of a particular parking place.

8. **DistanceCalc:** This PHP script is used for the two step verification to check if the user’s current GPS location is within the periphery of 40m of the parking lot’s location. This script uses Google DistanceMatrix Api with a server key acquired to be used for the purpose of this application. This API gives us the estimated distance and between an origin and destination. We provide the origin as the user’s current GPS location and destination as the parking lot’s known location, we get the distance between the two locations. We simply check if the distance is within the aforementioned range of 40m.

9. **Get User Status:** This provides us with the user's current status i.e. notbooked or booked or parkingstarted. In case of booked and parkingstarted, it also provides location (latitude and longitude) of the parking lot currently associated with the user.

4.6 Data Collection Module

Data Collection Module was developed in PHP which would collect the data (potential parking lot information) of a given radius around a particular geolocation. A hexagon was created around the given circular location and then hexagonal tessellation was conducted on top of it. Then radar search was conducted over each smaller hexagon and for each place returned, a place details search was conducted to get the details.

A test run of the algorithm was conducted over 'Hyderabad, India' city, with the constraint of having maximum of 1000 queries on the Google Places API Web Service.

Input: Hyderabad's central geolocation: 17.433959, 78.457611

Hyderabad's approximate radius: 26 km

Output: 915 new potential parking places

Query Runtime: 11 minutes (approx.)

Collisions (Places repeated over adjacent hexagonal searches): 65

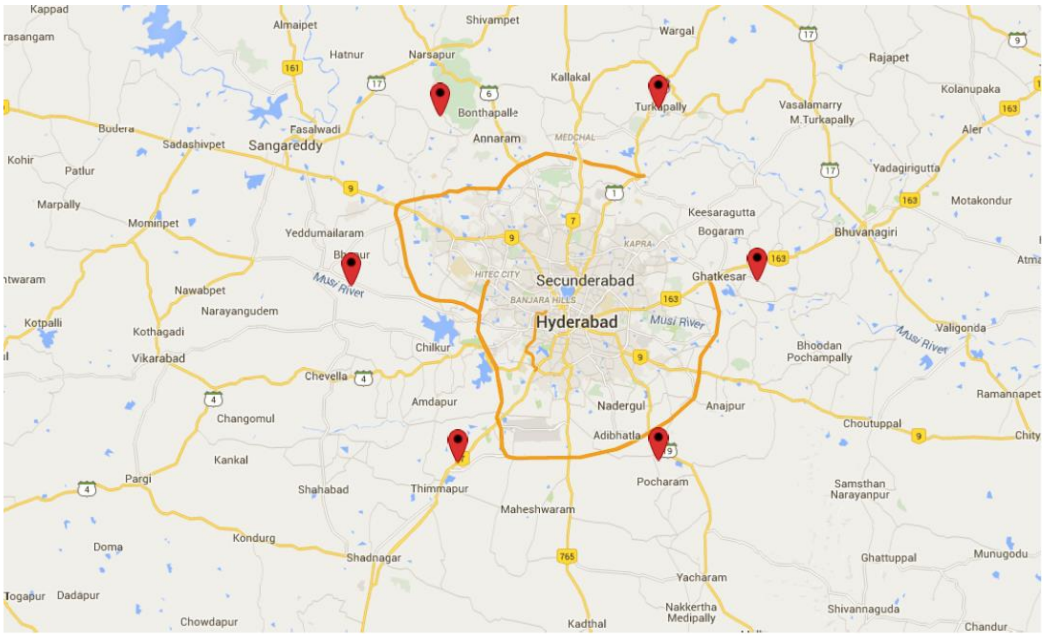


Fig: Outer Hexagon Creation of Hyderabad

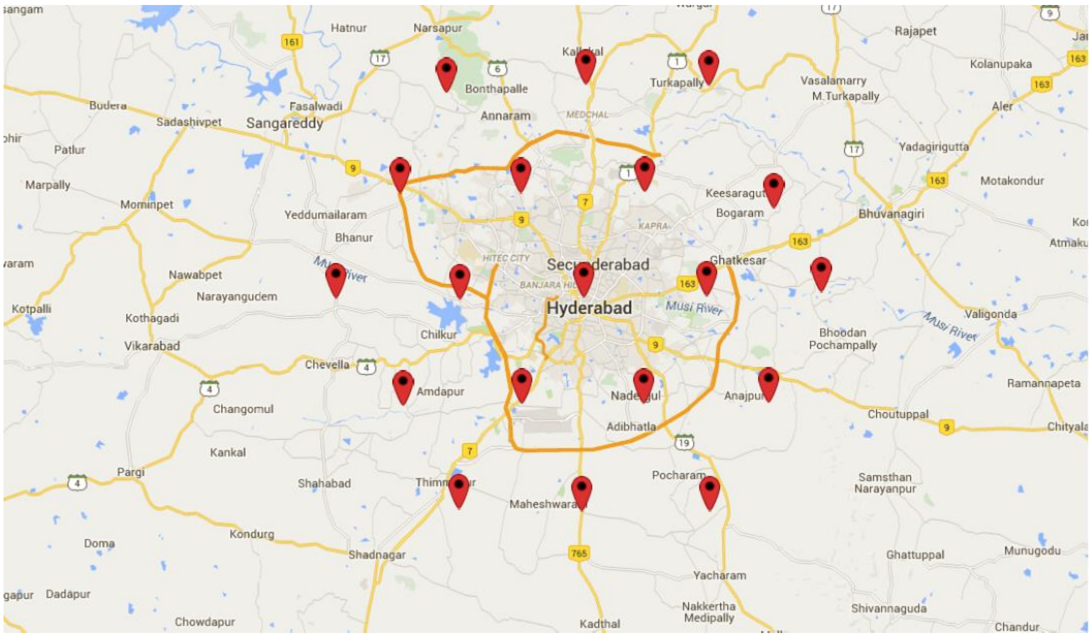


Fig: Circumcenters of all the hexagons created after the hexagonal tessellation of the city

4.7 Prediction Algorithm

Prediction Algorithm is implemented in PHP and backend MySQL database. It shares the same database which contains transaction and parking place information. The algorithm consists of two parts, one for predication of availability and another for training the algorithm itself.

Prediction: The algorithm requires just three parameters viz. parking lot identifier, UNIX timestamp of the booking and type of vehicle (2-wheeler or 4-wheeler). It fetches corresponding weights from "parking_place_weights" table and "registered_parking_places" table. Then, for a given booking time, it extracts all the required parameters (refer proposed solution). Then, it calculates the probable availability and inserts the information into "prediction_training_data" table for future training.

Training: The training isn't initiated automatically but has to be triggered by an outside person over a period of time or after a certain number of trainings for individual parking lot. On trigger, the algorithm fetches all the rows from "prediction_training_data" which hasn't been used for training in the past and whose booking time has passed. Actual availability of the parking lot at that time is calculated. Then all the acquired information is used to update the weights for the individual least mean square rule. The algorithm learns with each new booking request.

5. Results and Discussion

The outcomes of the project can be viewed in the following domains.

User-experience: The front-end Android App has been completed which along in collaboration with backend PHP scripts, database support, internet connectivity and various APIs (Google and Zxing) allows users to book a parking place of their choice for a time of their choice. The app allows the users to book parking spots with ease as the App is location based (map is shown) so that the users can easily book parking spots where they desire. A simple flow diagram. of the App reveals that it isn't too complex and user experience is given top priority.

Backend Design: The backend is designed with a central server, which consists of required database and data access layer. The DBMS used is MySQL. The data access layer of the project is developed in PHP. These technologies are the most widely used technology in the market. Currently, local server is used for the data storage and data access. This local server is created under XAMPP environment and accessed via Wi-Fi. Other Google APIs and any other dependencies are fulfilled by having a direct connection with the Internet.

Communication between Parking Manager App and Backend: The communication takes place via HTTPClient Message Handlers and Custom Delegates designed on Android (Java) from the App's perspective. These handlers and delegates call the required script and gets the response message. All communication takes place using JSON formatted strings.

Development of functional Data Collection Module: The developed collection module is a standalone module which is able to collect information from Google Maps APIs to collect various potential parking places of a city.

Development of Prediction Algorithm: A basic machine learning algorithm was developed which is used to both predict the future availability and train itself to minimize errors in prediction.

6. Future Works

- Testing of the given application for load management. If the load management isn’t handled, then a decentralized server could be developed as required.
- Development of payment portal for the end users to pay the system without any hassle of cash.
- Test-data collection of all the transactions to feed into the prediction algorithm. It might require some time for data collection. A few months of data might be useful to train the prediction module.
- Collected parking lot information needs to be verified and its accuracy is to be measured.

7. Literature Cited

1. Bell, M.C., Galatioto, F.: 'Simulation of illegal double parking: quantifying the traffic and pollutant impacts' Proc. Fourth Int. SIIV Congress, Palermo, Italy, 2007, pp. 12 – 14

2. Barone, R.E., Giuffrè, T., Siniscalchi, S.M., Morgano, M.A., Tesoriere, G.: 'Architecture for parking management in smart cities' Intelligent Transport Systems, IET, 2014, 8, pp. 445 – 452

3. Tang, V.W.S., Yuan Zheng, Jiannong Cao: 'An Intelligent Car Park Management System based on Wireless Sensor Networks' Pervasive Computing and Applications, 2006 1st International Symposium on, 2006, pp. 65 – 70

4. Ganti, R.K., Fan Ye, Hui Lei: 'Mobile crowdsensing: current state and future challenges' Communications Magazine, IEEE, 2011, 49, pp. 32 – 39

5. Prasad, A.S.G.; Sharath, U.; Amith, B.; Supritha, B.R.; Asokan, S.; Hegde, G.M.: 'Fiber Braggs Grating Sensor instrumentation for parking Space occupancy management' International Conference on Optical Engineering, 2013, pp. 1-4

6. Haoui A. et al: 'Wireless magnetic sensors for traffic surveillance' Elsevier journal, Transportation Research, 2008, part C 16, pp. 294 – 306

7. Lan et al.: 'An intelligent driver location system for smart parking' Elsevier Publications, Expert Systems with Applications, 2014, pp. 2443 – 2456

8. Bo Xu, Wolfson, O., Jie Yang, Stenneth, L., Yu, P.S., Nelson, P.C.2013: 'Real-time Street Parking Availability Estimation', IEEE 14th International Conference on Mobile Data Management, 2013

9. Ganesan, K., Vignesh, K.: 'Automated Parking Slot Allocation using RFID Technology'. Signal Processing and Its Applications, 2007. ISSPA 2007. 9th International Symposium on, 2007, pp. 1- 4

10. Caliskan, M., Barthels, A., Scheuermann, B., Mauve, M.: 'Predicting parking lot occupancy in vehicular ad-hoc networks' 65th IEEE Vehicular Technology Conference, VTC 2007, IEEE Press, Dublin, Ireland, 2007, pp. 277– 281

11. Gwo-Jiun Horng; Chi-Hsuan Wang; Sheng-Tzong Cheng: ‘Using cellular automata on recommendation mechanism for smart parking in vehicular environments’ Consumer Electronics, Communications and Networks (CECNet), 2012 2nd International Conference on, 2012, pp. 3683 – 3686

12. Geng et al (2012). ‘A new “Smart Parking” System infrastructure and Implementation’. 15th meeting of EURO Working Group on Transportation, Elsevier Publications, Procedia - Social and Behavioral Sciences 54. Pages – 1278 – 1287.

13. Caicedo et al (2012). ‘Prediction of parking space availability in real time’. Expert Systems with Applications 39 (2012) 7281–7290

14. Klappenecker et al. (2014). ‘Finding available parking spaces made easy’. Ad Hoc Networks 12 (2014) 243–249.

15. Grazioli et al. (2013). ‘Collaborative Mobile Application and Advanced Services for Smart Parking’. 2013 IEEE 14th International Conference on Mobile Data Management. 978-0-7695-4973-6/13

16. Dusˇan Teodorovic´, Panta Lucˇic´: ‘Intelligent parking systems’. European Journal of Operational Research 175 (2006) pp. 1666–1681

17. Cheng Tiexina , Tai Miaomiaoa , Ma Zeb: ‘The Model of Parking Demand Forecast for the Urban CCD’. 2012 International Conference on Future Energy, Environment, and Materials, Energy Procedia 16 (2012) pp. 1393 – 1400

18. Felix Caicedo: ‘The use of space availability information in ‘‘PARC” systems to reduce search times in parking facilities’. Transportation Research Part C 17 (2009) pp. 56–68

8. Acknowledgements

We were fortunate enough to do a project while pursuing Bachelor of Technology in Computer Science and Engineering at NIT Warangal. We have learned a lot and enjoyed doing our project under the support of wonderful people. We sincerely would like to thank every individual who helped in our project.

We would like to thank our project guide and mentor, Dr. Rashmi Ranjan Rout for his help, guidance and invaluable inputs during the course of this study.

We would like to take this opportunity once again to thank Dr. Ch. Sudhakar, Head of the Department, Computer Science and Engineering, NIT Warangal, for giving us an opportunity and resources to work on this project and supporting me throughout. We are also grateful to our dissertation committee members for giving us an opportunity to work on this project and giving valuable comments on my proposal and research.

Anjali Tiwari

Avala Ganesh

Bishal Kumar Shrestha