DEPARTMENT OF COMPUTER SCIENCE ENGINEERING,

SCHOOL OF ENGINEERING AND TECHNOLOGY,

SHARDA UNIVERSITY, GREATER NOIDA

SOPA: A Parking Solution

A project submitted

In partial fulfillment of the requirements for the degree of Bachelor of Technology in Computer Science and Engineering

BY

Sonal Shriti (2018004874)

Akshat Gupta(2018011084)

Rahul Singh(2018009434)

Kumar Anurag(2018006817)

Mohd. Shahnawaz(2018012249)

Nandita Pokhriyal(2018013001)

SUPERVISED BY

Dr. Vishal Jain

May, 2022

CERTIFICATE

This is to certify that the report entitled "**SOPA: A Parking Solution**" submitted by "SONAL SHRITI(2018004874), AKSHAT GUPTA(2018011084), RAHUL SINGH(2018009434), KUMAR ANURAG(2018006817), MOHD. SHAHNAWAZ(2018012249) & NANDITA POKHRIYAL(2018013001)" to Sharda University, towards the fulfillment of requirements of the degree of "**Bachelor of Technology**" is record of bonafide final year Project work carried out by him in the "Department of Computer Science and Engineering, School of Engineering and Technology, Sharda University". The results/findings contained in this Project have not been submitted in part or full to any other University/Institute for award of any other Degree/Diploma.

Signature of Supervisor
Name: Dr. Vishal Jain
Designation: Associate Professor

Signature of Head of Department
Name: Prof. (Dr.) Nitin Rakesh

Place:
Date:

Signature of External Examiner
Date:

ABSTRACT

The vehicle density has increased at an alarming rate and continues to do so. This is especially apparent during the peak hours of the day. This has made it very hard for vehicle owners to find a place to park their automobiles. This study aims to develop a mobile application by employing state-of-the-art technologies to tackle this widespread problem. The application uses data supplied by the users to give spot seekers a preferred location where they can park their vehicles. It also encourages users to supply more data by introducing in-app currency. This can help to combat problems like fuel wastage and save time as well.

Index Terms- Parking solution, mobile, spots, application

ACKNOWLEDGEMENT

A major project is a golden opportunity for learning and self-development. We consider ourselves very lucky and honored to have so many wonderful people lead us through the completion of this project.

First and foremost we would like to thank Dr. Nitin Rakesh, HOD, CSE who gave us an opportunity to undertake this project. Along with our grateful thanks to Dr. Vishal Jain for his guidance in our project work.

He had, in spite of being extraordinarily busy with academics, took time out to hear, guide and keep us on the correct path. We do not know where we would have been without his help.

The CSE department monitored our progress and arranged all facilities to make the sailing so much easier. We would like to take this moment to acknowledge their contribution gratefully.

Name and signature of Students :

SONAL SHRITI(2018004874)

AKSHAT GUPTA(2018011084)

RAHUL SINGH(2018009434)

KUMAR ANURAG(2018006817)

MOHD. SHAHNAWAZ(2018012249)

NANDITA POKRIYAL(2018013001)

Table of Contents

List of Tables

List of Figures

Chapter1: INTRODUCTION

Parking is a day to day activity that characterizes life in cities. It is therefore the most common and necessary requirement of drivers to not only quickly but also efficiently search for parking spots. Visiting new locations more often than not comes with an inherent parking problem. Something that inspires humor are jokes about holiday parking lot wars. But the amount of time spent looking for someplace to put your car at, on crowded city streets or mall lots has increasingly become a matter of grave concern. A 2011 IBM study claimed that almost 30 percent of a metropolis city's traffic can be attributed to people looking for parking slots, with a third of New York City drivers reporting that they spend on an average about 20 minutes searching on the road every day. Technological advancements in various domains have been a great help in increasing the Not Operating Income (NOI) for investors and have therefore aided in producing higher profitability. An example of which in the problem domain being discussed would be, sensors being used to detect the number of parking spaces that go unused. These sensors are being utilized for a variety of purposes like determining the number of spaces that need to be maintained and thereby assisting in making marketing decisions. An important point that goes mostly unnoticed is the fact that people are usually unaware of existing common/local parking spaces whenever they are visiting a new place, be it in the same city or a new city. Existing parking solutions are mostly targeted on solving the issues relating to managing parking space, smart parking, or parking management. The issue of locating a parking spot is highly important in the context of the parking management and has not been worked on enough or in the right way. Existing parking spot location solutions mostly suffer with the lack of parking spot data or lack of data quality and no incentive to add to the parking data.

Parking is a day to day activity that characterizes life in cities. It is therefore the most common and necessary requirement of drivers to not only quickly but also efficiently search for parking spots .Visiting new locations more often than not comes with an inherent parking problem. Something that inspires humor are jokes about holiday parking lot wars. But the amount of time spent looking for someplace to put your car at, on crowded city streets or mall lots has increasingly become a matter of grave concern. A 2011 IBM study claimed that almost 30 percent of a metropolis city's traffic can be attributed to people looking for parking slots, with a third of New York City drivers reporting that they spend on an average about 20 minutes searching on the road every day. Technological advancements in various domains have been a great help in increasing the Not Operating Income (NOI) for investors

and have therefore aided in producing higher profitability. An example of which in the problem domain being discussed would be, sensors being used to detect the number of parking spaces that go unused. These sensors are being utilized for a variety of purposes like determining the number of spaces that need to be maintained and thereby assisting in making marketing decisions. An important point that goes mostly unnoticed is the fact that people are usually unaware of existing common/local parking spaces whenever they are visiting a new place, be it in the same city or a new city. Existing parking solutions are mostly targeted on solving the issues relating to managing parking space smart parking, or parking management. The issue of locating a parking spot is highly important in the context of the parking management and has not been worked on enough or in the right way. Existing parking spot location solutions mostly suffer with the lack of parking spot data or lack of data quality and no incentive to add to the parking data. Our solution attempts to solve this problem by using crowdsourcing to collect parking data from users and a reliable rating system to maintain the quality of the data obtained from crowdsourcing. We also figured out that many users might have personal space which they would be willing to monetize by leasing the space for parking at certain times of the day. We have developed a solution to solve this problem by providing a way for users to lease their personal space for parking which any other user can see , book and use for the consent period by paying a fee.

Users are provided with incentive to add parking data to the system through a robust in-app token system that is used to reward users for contributing to the system and can be spent on searching for parking spots or other in-app services and features.

The remainder of this paper is organized as per the following scheme: Section 2 provides an analysis of some related as well as unrelated work that has been done in this field. Section 3 sheds light on the methodology used to carry out the implementation. The section also discusses, at length, the different modules that make up the application. Section 4 consolidates all the results from the implementation. Lastly, section 5 mentions the future scope of the implementation.

Project Description:

Our solution attempts to solve this problem by using crowdsourcing to collect parking data from users and a reliable rating system to maintain the quality of the data obtained from crowdsourcing. We also figured out that many users might have personal space which they would be willing to monetize by leasing the space for parking at certain times of the day. We have developed a solution to solve this problem by providing a way for users to lease their personal space for parking which any other user can see , book and use for the consent period by paying a fee.

Users are provided with incentive to add parking data to the system through a robust in-app token system that is used to reward users for contributing to the system and can be spent on searching for parking spots or other in-app services and features.

1.1 Motivation

One of the most common issues nowadays is the absolute lack of parking spaces. Vehicles proceed to dwarf existing parking spaces, hence clogging streets. Episodes of savagery over-occupancy, distorted cars due to a space crunch, and cheating for stopping are a few issues that are often witnessed as an aftermath of ever increasing demand. An unregulated tax structure leads to a shortage of parking spaces. In Indian metros, parking is either free or negligibly priced, the expenses being unregulated for a long time now. For instance, Mumbai charges the same parking expense because it did 20 years ago and has one of the least taxes in the world. Since parking costs halt expanding after a certain period of time, the longer one remains in a parking space, the less one must pay. In Sarojini Nagar, Delhi, the parking cost may be a scanty Rs. 20 per hour with a standard charge of Rs. 100 for 24 hours, making parking indeed cheaper. Why is this an issue? Parking could be a rare commodity nowadays and ought to come with a cost. A low parking cost directly affects the number of vehicles on the street, contributing to air and noise pollution. The most perfect way to oversee the parking is by charging the proper cost for it. In the New York midtown range, the street range per individual is 33.3 sq. m ; in Mumbai's Null Bazaar however, it is simply 1.7 sq. m. This implies that a vehicle that stands in Mumbai's Null Bazar should ideally incur about 20 times as much cost as it would've within the New York midtown zone. This pretty much sums up the parking situation right now predominant within the nation since if Mumbai being India’s monetary capital doesn’t have it great then no other place does either. Another issue in Indian cities is the skewed request for on-street parking since it’s cheaper than off-street stopping. On-street parking issues frequently cause delays, particularly on streets with overwhelming traffic.

1.2 Overview

- Commuters face a problem where they struggle and spend their precious time looking for parking spots.
- Also a number of people have unused private parking space which can be utilized.
- There is a major lack of parking space data for app (need refinement ideas).

The solution to increasing demand and less space seems to be pretty straightforward—charging people for the resources they use however temporary the usage might be. One might think that the government. building parking spots at public spaces might as well solve the problem altogether(with relevant charges involved on part of the vehicle owners), but paying for what might be obtained for free isn't exactly something we Indians are particularly fond of, which leads to arrays of cars parked by the roadside making it congested for the people actually using that road for commuting.

Even if the authorities do manage to somehow extort the money from the vehicle owners, one can't possibly buy parking space at every place they go to, for instance, visiting a friend's place for a birthday party.

1.3 Expected Outcome

We propose an application that provides the right incentive to vehicle owners to share empty or available parking information with the rest of the users through a system of reward points (in-app currency).

- The users won't have to spend a single penny in order to be able to use the app and avail the associated benefits, all they would need to do is reach a certain target in terms of rewards points to be able to redeem reliable parking information related to their respective location, as and when required.
- The information itself would be put on the application by these same users in a bid to win more reward points. They will have to share the location using GPS along with a photograph of the place (which upon verification will be accepted and only then will the reward points be credited).
- A user would be allowed to view parking information only after he/she has reached a certain number of reward points and would be required to pay in those very same reward points, every time they view a certain parking location.

1.4 Possible risks

- Some users could falsely upload a private spot that does not belong to them.
- property validation can be a slow process and create a bottleneck
- False public spots can be added to the database and removing that based on reviews can also be challenging and a time taking process. by the time we remove a false spot, it might have already have degraded many user experiences

- when scaling the application through a city, there will come a point when freemium users will have no true public parking spots to add and therefore no way to earn karma coins forcing them to abandon or switch to premium

Chapter2: METHODOLOGY

2.1 Data and Relational Views

Table 2.1. A Comparative Analysis

S No.	Features	Parking Rhino	Best Parking (by Arrive)	Park Plus	SOPA (our solution)
1	Review/ Report	✓	✓	✓	✓
2	Location Search	✓	✓	✓	✓
3	Location wide slot display	✓	✓	✓	✓
4	Free Parking options	✓			✓
5	Pay and park options	✓	✓	✓	✓
6	Premium				✓
7	Easy Payments	✓	✓	✓	✓

S No.	Features	Parking Rhino	Best Parking (by Arrive)	Park Plus	SOPA (our solution)
8	Refund				✓
9	In-app Currency				✓
10	Users can upload potential Parking locations		✓	✓	✓
11	Spot Reservation		✓		✓(private spots)
12	Distinct Markers	✓			✓
13	Real-Time Navigation to Parking Location	✓	✓	✓	✓
14	Detailed location information	✓	✓	✓	✓
15	Lease out a spot				✓
16	Technology Used	IOT			React Native

2.1.1 Case Study: BestParking

BestParking is an innovative parking app that doubles as a search engine catering to parking requests. It guides its customers towards the most affordable and convenient spots in public as well as private spaces in North America. Company statistics claim that vehicle owners use their solutions, website and app to make an educated discussion based on cost comparison between all the parking facilities encumbered on their platform, before going ahead with reservations.

The biggest names in the lot business have partnered with BestParking to offer deals, specials and reservations to their online patronage. These businesses include names like Quik Park, Parkway Corp, Edison ParkFest.

BestParking was developed by Arrive, a leading producer of all encompassing mobile solutions. Their technology is used by companies like Honda, GasBuddy, etc. The two parking apps ParkWhiz and BestParking are Arrive's award winning solutions. Over 40 million people have been estimated to use their solutions web, app, API based and voice.

It can be used to find and reserve parking in cities across North America, including:
San Francisco, Chicago, Boston, Seattle, New Orleans, Philadelphia, Baltimore, Sacramento, Washington DC, Atlanta, San Diego, Los Angeles, Dallas, Toronto and it can be used to book airport parking at major airports including:
Los Angeles (LAX), Dallas (DFW), New York (JFK), Toronto (YYZ), Chicago (ORD), Newark (EWR)

The most prominent features of the software include:
1. Customers can find parking at any location, close or distant, for instant use or reservation. It also allows them to compare fares and pick a place that's not only desirable but affordable.
2. Upon reservation a pass is generated that facilitates the reserve's entry into and exit from the parking location.
3. The interface of the website as well as the application makes searching for parking locations, and the overall user experience seamless.
4. Parking locations can be searched from as vague information as the name of cross street or a nearby attraction. All the available parking locations upon searching would be made available to the user in a sorted list.

5.Details such as availability or unavailability of facilities like valet parking, locked or self park , indoor or outdoor are mentioned.

It works by providing provision for users to search for a parking location as needed. Destination as mentioned earlier can be a specific address or something vague like a landmark. Upon hitting the search button a list of parking spots will be generated, the users acn compare the pisces of the enlisted locations. Thereupon the user can reserve a spot at a location of their choice, pay online, get navigation aid via their favorite navigation app. Get entry into the location uing the pass.

Some of the advantages include:

1.Simple and reliable interface

2. Parking locations are sorted by price making it easier for users to select the best fit for themselves.

3. The pass provides for a smooth ride with no stops.

Some of the disadvantages include:

1.The app is limited in its reach. Only covers North America.

2.Locations that might or might not be available show up on the app. Could lead to a probable waste of the user's time.

2.1.2 Case Study: Park+

Park+ has managed to build its name as a leading player providing parking systems of paramount density with a line of most completed projects and an abundant product range in the US. Their patented solutions for parking systems which are automated are designed, installed, maintained and manufactured by them, along with other parking related products like, multi-level car stacking structures, automatic car towers.

Founded in 1969, half a century ago in Queens, New York, the company boasts a varied experience based on a multitude of software/ hardware solutions. Park+ has its headquarters in New York spread over 66000 sq. ft. and a manufacturing unit in New Jersey.

Park+ boasts of the following achievements:

The Florida office of Park+ employs in it 15000 sq. ft. of facility on Dania Beach, employs a complete service team and has installed if not hundreds then dozens of projects in the area, including names like Paramount AGV and Racks & Rails, both being global projects aimed at facilitating the global parking scenario.

California, Los Angeles, based Park+ office boasts dense parking solutions installed on the West Coast with tens of projects installed in the area. Park+ has positioned itself in such a way that it has developed

a capability of serving multiple arenas and regions in the U.S., including satellite offices and projects in Boston, Seattle, San Francisco, Philadelphia, Texas and San Antonio. Park+ has managed to establish a global presence with its reaching network of facilities, projects and suppliers.

Most of the automated parking solutions throughout the U.S. have been installed by Park+. These include but are not limited to systems like robotic valet facility in California, largest robotic valet service - one of its kind in Manhattan, and the first AGV robotic parking solution in London, England.

AUTOMATED GUIDED VEHICLE, OR AGV:

The manufacturing facility located at the Park+ head office manufactures the AGV (Automated Guided Vehicle) System . The AGV is a self sufficient, self charging, mobile, self directional robot that uses markers, sensors, computer vision, and lasers for guidance of self and management of the vehicles stored on trays. The AGV designed in house by a team of professionals in their respective fields, boast varying global patterns of operation. This unique design has started a new era for high density vehicle storage and maintenance and management, transforming parking experience of the masses and providing efficient, omnipotent and effective solutions.
The AGV components come in house manufactured from their main manufacturing facility in New Jersey. The facility equipped with water jet cutters, beam drillers, CNC machines, brake presses and robotic welders has manufactured the AGV to the tallest precision and quality standards in the U.S..

Park+ provides an integrated application for smart phone users providing a convenient way to raise and deal with vehicle parking along with a ton of added features like managing of electric automobile charging, inclusive of a control unit and managing a single input power to provide feed to four Electronic Vehicle charging stations.

With AGV technicians available 24x7, Park+ commands remote monitoring of its solutions. Live streaming as well as video recording ensures availability of surveillance in order to provide assistance as and when required. Their sales after service and support workforce operates incessantly in all regions of operation. Park Plus is proud of its over 50-year legacy as the industry leader in automated and mechanical parking solutions.

2.1.3 Case Study: Parking Rhino

With ParkingRhino, you can park your car or motorcycle smarter, faster, and easier. ParkingRhino is

a map-based geo navigation tool that helps you find parking near your present location or at a given destination. The app allows you to search for and view both free and paid parking places, as well as information on available amenities like valet services, car washes, and auto charging. ParkingRhino offers both secured and on-the-street parking at a variety of places, including bus stops, train stations, public parking lots, and airports.

Key Features:

- Separate markers indicating free parking, paid parking, concierge service, and event parking - Narrow or broaden parking search results by altering the search radius around the user's current mobile location.
- Enter a specific location in the Lookup text box to find parking in that region. - Tap on a specific spot to see what amenities are available, such as Car Wash, Concierge Service, covered parking, and Car Charging.
- Directions to the parking site in real time - Event information

ParkingRhino provides its users the following information about a parking location:

- Full address
- Distance from current location
- Photos of the location
- Hourly parking rates for weekdays and weekends
- Hours of operation for weekdays and weekends
- Total vehicle (car/park) parking capacity
- Availability of facilities for each location such as Car Wash, Valet Parking, covered parking, Car charging etc.
- Eventful places and event information in a particular city

They are adding more verified locations every day.

Advantages:

- ParkingRhino has started its operations in Bangalore, Hyderabad, Kochi, Kolkata, Mumbai,Chandigarh, Chennai, Delhi NCR, Guwahati, Pune & Shimla

Disadvantages:

- Sometimes OTP takes time to generate

- App buffers
- UI is not user friendly

2.2 Other Contemporary Approaches

Urban cities becoming incredibly populated and hence congested teemed with the increase in vehicle ownership trends has led to the traffic crisis not only on roads but also in the public - private space management [1]. This of late has become a key issue with high economical impact. Today, cities are responsible for more than 75% of waste production, 80% of emission, and 75% of energy-utilization [2]. Taking the example of Europe, road transportation produces about 20% of the total CO2 emissions, out of which 40% of the emissions are generated by urban mobility [3]. Estimates state that the vehicles running and searching for unoccupied parking spaces cause about 30% of the daily traffic congestion in urban areas [4]. An effectively managed smart parking system would allow drivers to do away with the time they waste looking for ideal parking spots, an ideal situation which would not only help save time but would also consequently help in the decongestion of the roads thereby, improving the whole traffic situation, this would also ensure the optimum usage of parking spaces all the while playing a vital role in the domain revenue generation [5]. Even though the solution of infrastructure construction and public transport enhancement (e.g. buses and the legalization of the Uber service) might seem like the most obvious choice to deal with this problem it's not the most practical one, the use of technology on the other hand is another solution - a partial one to this bottleneck problem in the economy and the built environment[6]. Parking space demand is more of a dynamic entity as it changes from time to time, so if seen logically, providing real-time parking information (location and occupancy related) is not only difficult but also futile [7]. The origin of the smart apps meant to assist the parking situation can be traced back to around two decades back with systems like, “Guided Parking System” and “Dynamic Parking Guidance System” [8]. With the advent of Parking Guidance and Information (PGI) systems, improving parking efficiency has improved to a certain extent [9]. The effects of which are evident in the reduced noise levels, fuel consumption and exhaust emissions (due to a decrease in the vehicle count at certain segments of the traffic and also due to shorter travel times). Farooqi et. al. [10] in his work on UParking provides a complete solution, with ANPR cameras capturing vehicle plate numbers at the parking location and matching the obtained information against their database before providing the vehicle entry into the lot. Kurek et. al. [11] goes on to describe various typologies found in contemporary solutions, contemporary to ours is the typology based on Parking Guidance and Information (PGI) systems’ smart applications section where it has been described to have made the use of app sourced data along with GPS as a possible answer to providing navigational parking aid. The PGI systems typically consist of four constituents, monitoring parking lots, disseminating the

information thus collected, communication technology, and the control systems [12]. These systems use

either cameras or sensors to monitor the lot for possible occupancy. In Japan, the updating of the PGI information is done via a team of what they've termed as "walking surveyors" for entering data into GPS navigational systems (mounted on vehicles), that would show car park vacancies [13]. An alternative solution to Japan's is using a bunch of vehicles equipped with a variety of sensors to capture the street scenes and situation for transforming into (Geographic Information System) GIS data [14]. The sensor based techniques can be further classified into the off-road and on-road, depending on the kind of sensors used. The knowledge body in this arena is ever expanding with enormous deployments since 1996[15].

Anand et. al. [16] has included IOT to carry out the implementation. The application requires the user to register themselves in the app first in order to use the services. A registration card is then given to the drivers which is issued on a one-time basis, and the data of the driver is stored in the database in Raspberry Pi. Teong Ang et. al. [17] has proposed in his work a somewhat upgraded solution to the one proposed by Anand et. al [16], in that his proposed system iSCAPS uses Near Field Communication (NFC) function of smart phones as a stand-in for the aforementioned registration cards. An arduino microcontroller acts as the brain of this system which also provides an additional functionality of searching for one's vehicle in case one forgets the spot they had parked their vehicle in.

There exist several ways in which these smart parking systems can be designed employing the array of technologies that are on the rise. Some of which are sensor based solutions such as magnetic and acoustic sensors (on-road), ultrasounds, RFIDs, smart cameras, etc. A more advanced solution for the same is crowdsourcing, which is based on using ultrasonic sensors fitted into vehicles, for detecting empty slots near a parked vehicle followed by information dissemination to drivers searching for these slots. "There are also apps that give information on on-street parking, recommending the best zones to search for unoccupied parking spots, or giving the opportunity to announce when a user is about to move a vehicle, leaving free space, and in some cases, allowing reservations for these new free spots. Each of the above mentioned functionalities can use the data that has been obtained automatically via cell phones (crowdsensing) or provided manually by the app users (crowdsourcing). While off the street parking information has the potential to help users reduce the price to pay or to select the closest available space at their destination, on the street parking information usually has as its main goal the reduction of multiple cars chasing a single space"[18].

Parking more often than not is considered an arduous task, one that requires the driver to be in

possession of a certain level of expertise, in order to be able to safely and properly maneuver the vehicle into position. Perpendicular parking, among the many other methods, is considered the most complex, especially taking drivers with disabilities into perspective. Gamal et. al.[19] has, in his work, tackled this problem of perpendicular parking with the help of Deep Convolutional Neural Networks (CNNs). The neural network has been trained to imitate the Automatic Parking Assist Systems (APAS). Another work in this domain by F. Alshehri et al. [20], deals with the problem of "wrong parking"- vehicles being parked out of angles or at odd angles, by using a simple method that monitors the cars which are parked in a parking lot via two arduinos, one fixed and one mobile, along with a ultrasonic sensor, PIR motion sensors and a Nexion display attached to the fixed arduino to display the results. While the solution is effectively area/ park specific it does provide a working idea for a human proctor to act on, by color coded information on the Nexion screen of the issues the system has been made to deal with. In [21] the authors have suggested something called pocketsourcing for the automatic detection of available parking slots by making use of mobile phone sensors. The experimental results have been claimed to be 94% correctly predicted. A smart parking application called ParkTAG

uses an implementation where an automatic detection algorithm is used, this type of strategy is currently being used by various smart apps in European countries. The main barrier faced with an approach of this sort is to persuade the users to let the application keep continuing its operation in the background [22]. Another issue faced at the beginning stages of deployment would be insufficient data for the algorithm to work on due to lack of users, this is called a cold start problem which was solved in the aforementioned Japanese approach of using surveyors. Other works in the area contemporary to ours, focus on the overall enhancement of the working task by taking something similar to a cost function whose value relies on the distance from the destination and the money charged for the parking space.

Chapter 3: DESIGN CRITERIA

3.1 Requirements

Our requirements consist of a set of features that we would like to implement in our system.

We are aiming for a mobile app since that is the simpler platform commuters use. We decided to go with React Native framework for the mobile app since it is cross platform, i.e, we can use it to create

mobile applications for both iOS and Android systems which combined accounts for the majority of the users. Next, we require a database and cloud storage services to store out growing parking location data , any related metadata or the user information which should be accessible to all users. We decided to go with mongoDB since it is a scalable database and is easy to use. For the storage service we opted for AWS S3 since we were already aiming for Amazon Web Services for the cloud since it has a lot of affordable features and a strong community with documentations for every service. Next, we finalized our backend framework that is supposed to be running the application on cloud. We decided to go with FastAPI since it is a lightweight framework that is easy to use allows easier scalability and documentation tools. We took the microservices approach for the backend instead of the traditional Monolith approach since it is easier to scale and allows for easier testing and can be smartly used to increase the cost effectiveness and uptime of the application. Next comes the payment services that are supposed to be implemented for private spot lease and parking management. We decided to go with Razorpay since it is a popular payment service that is easy to use and has a lot of features that are useful for our application.

3.2 Design Diagrams

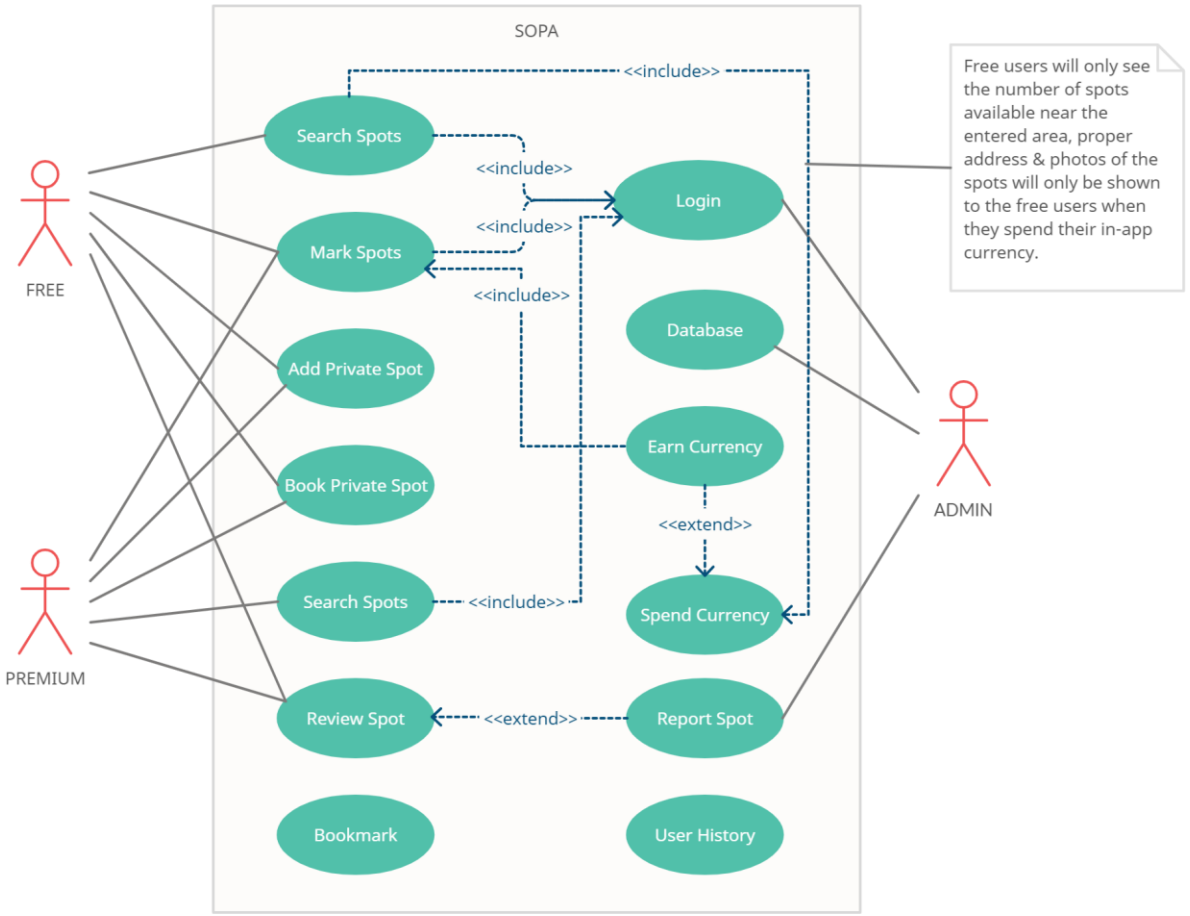

Figure 3.1. Use Case Diagram

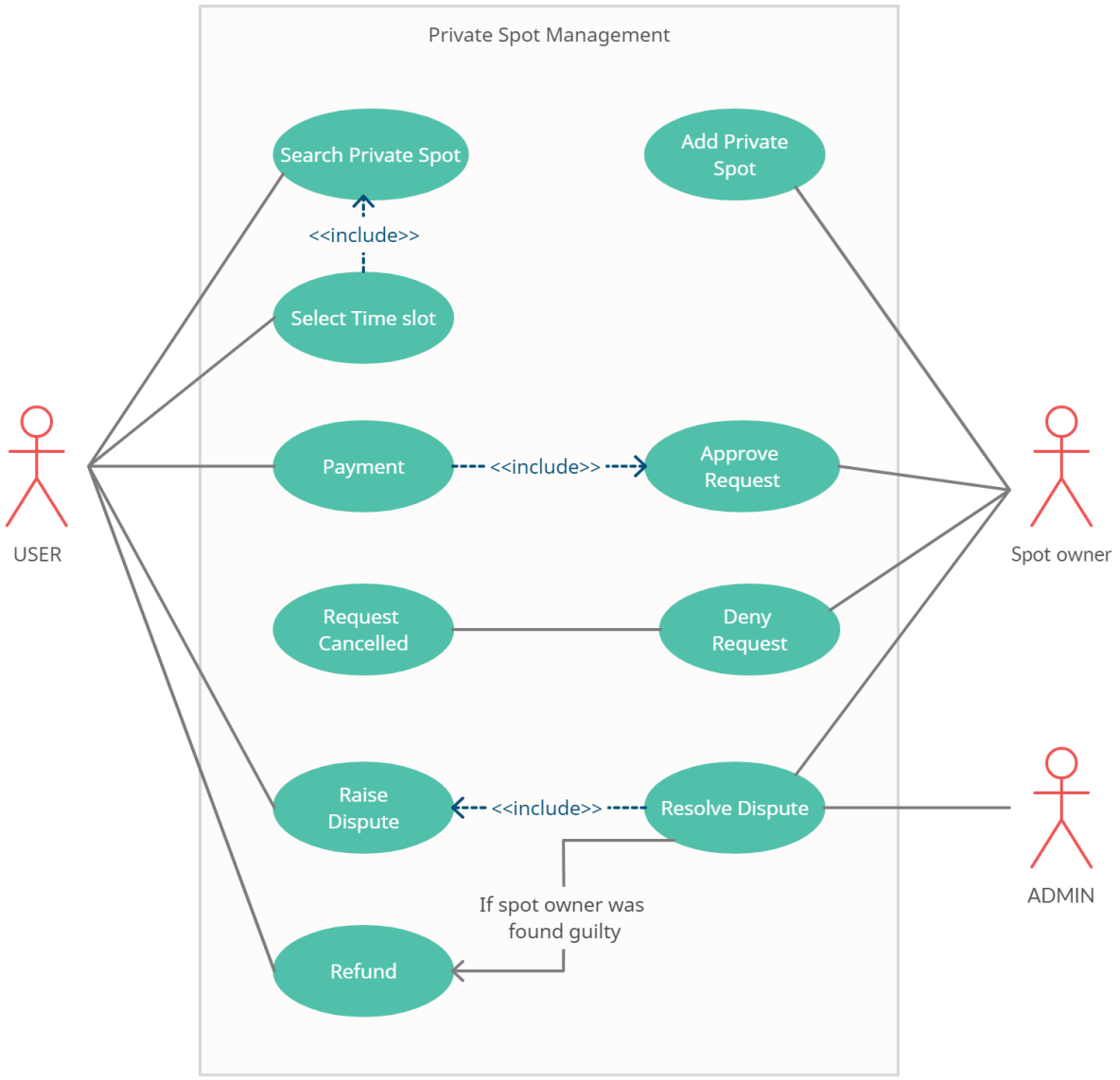

Figure 3.2. Use case diagram for private spots

Figure 3.3. DFD Level 0

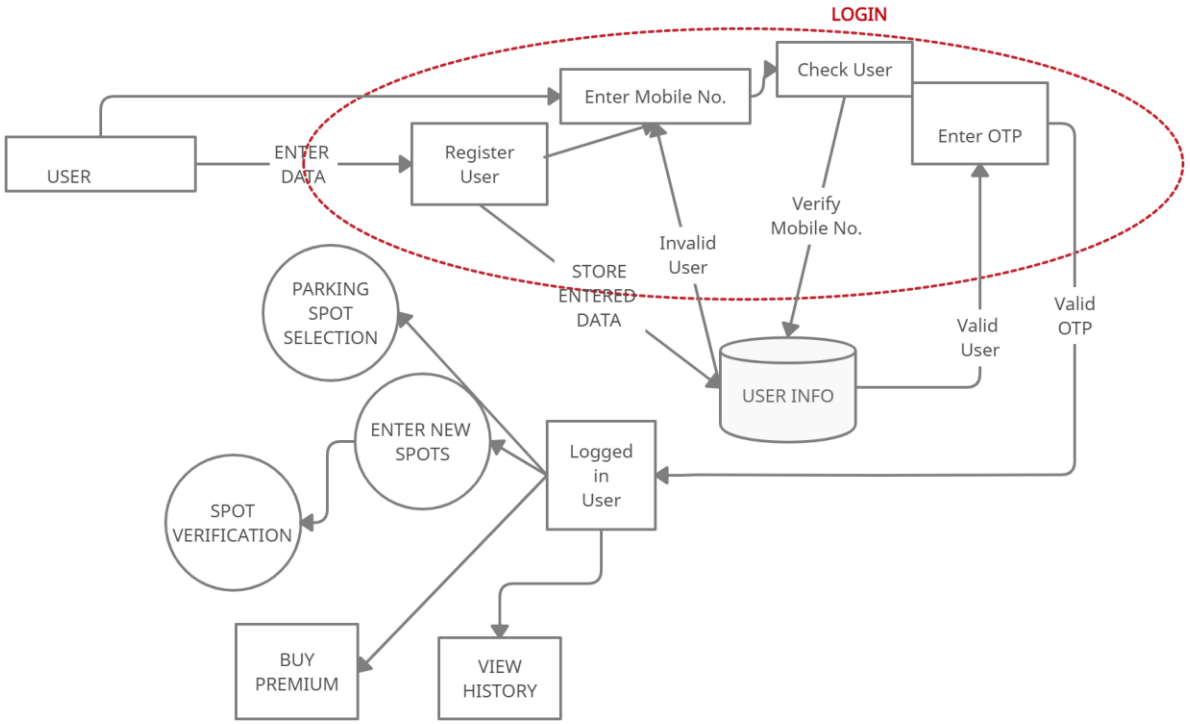

Figure 3.4 DFD Level 1

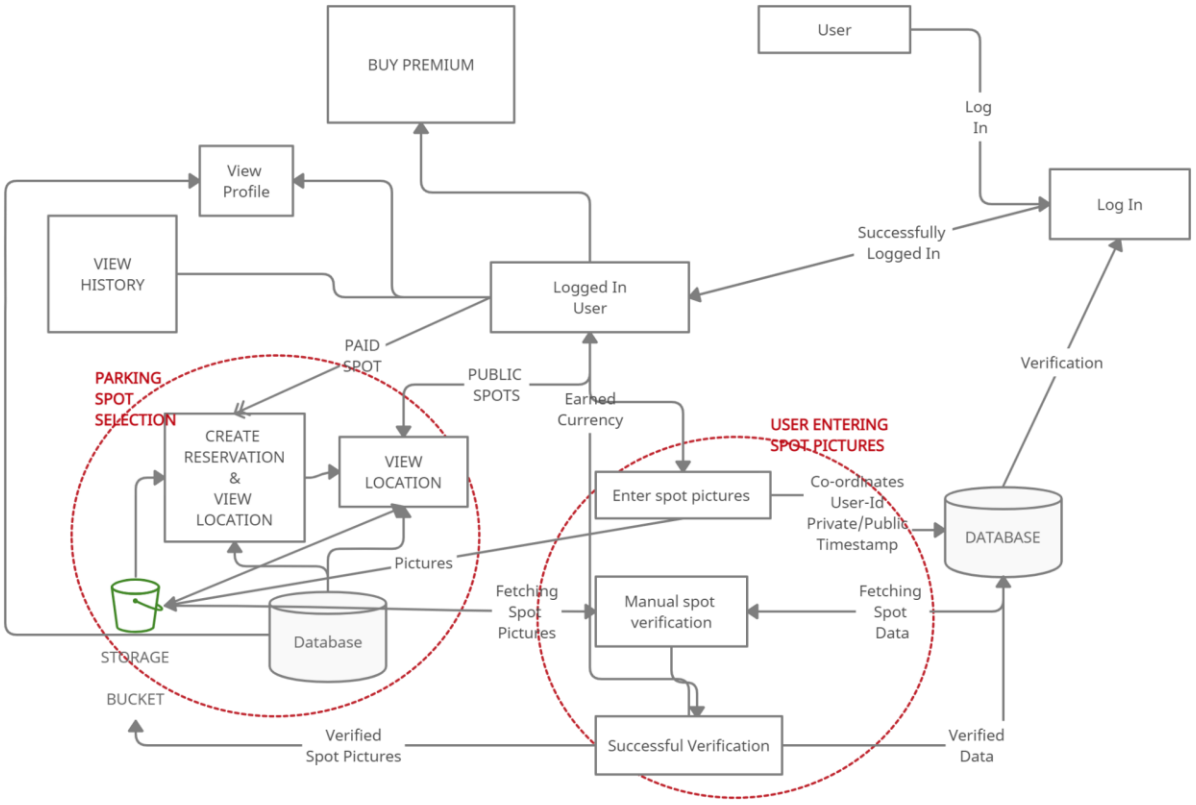

Figure 3.5. DFD Level 2

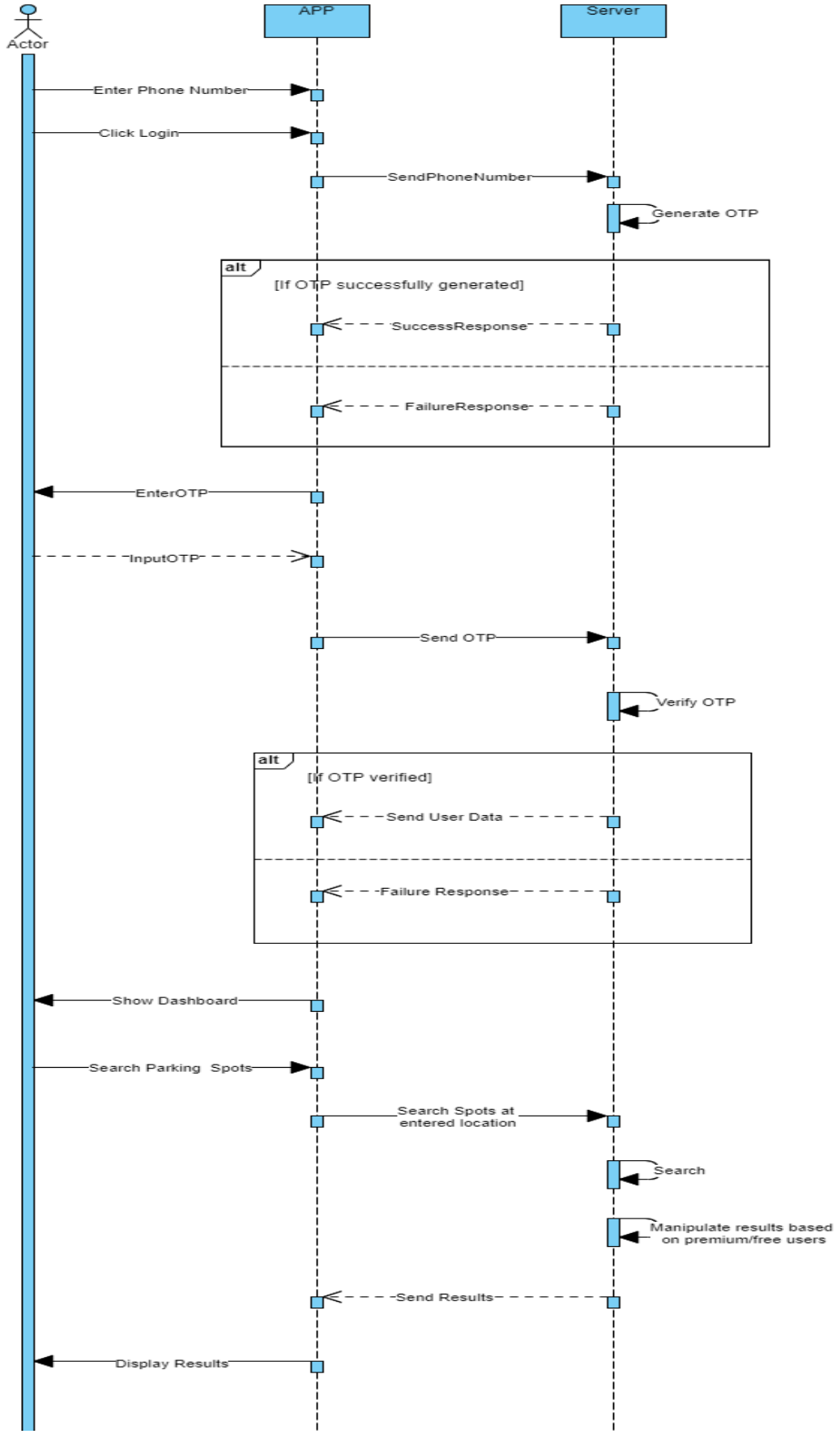

Figure 3.6. Sequence Diagram

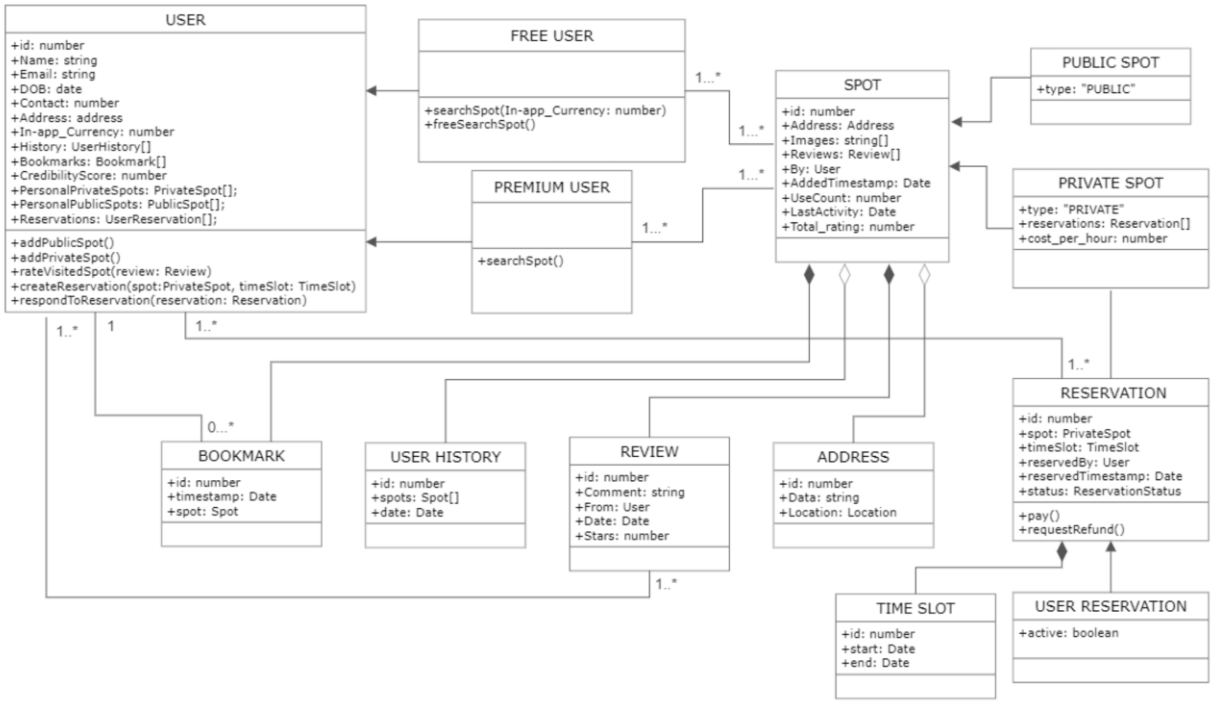

Figure 3.7. Class Diaagram

3.3 System Design

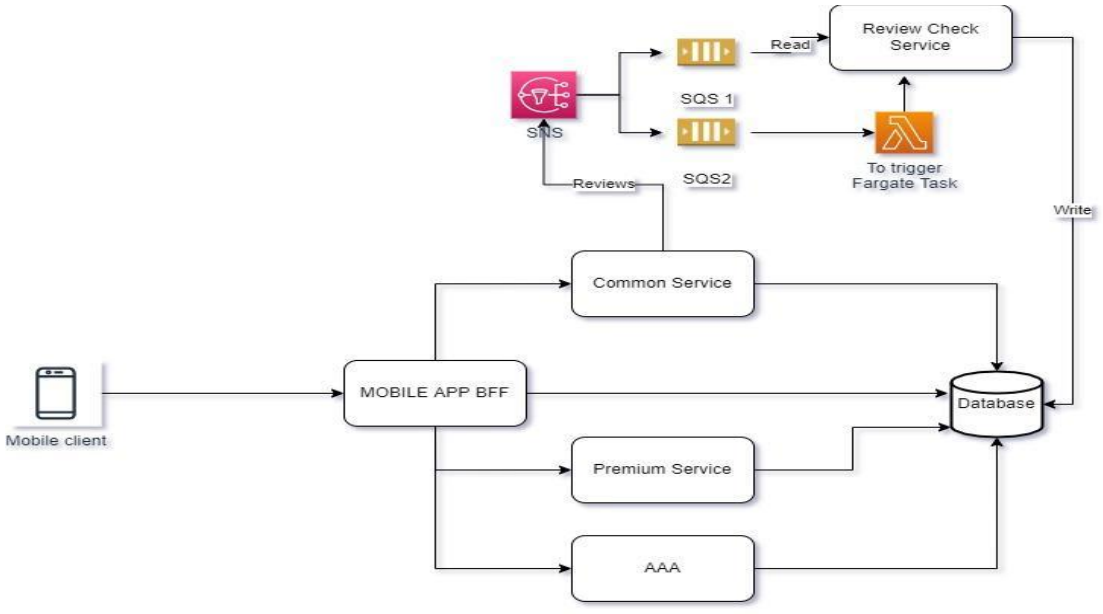

Figure 3.8 Review Spot Architecture

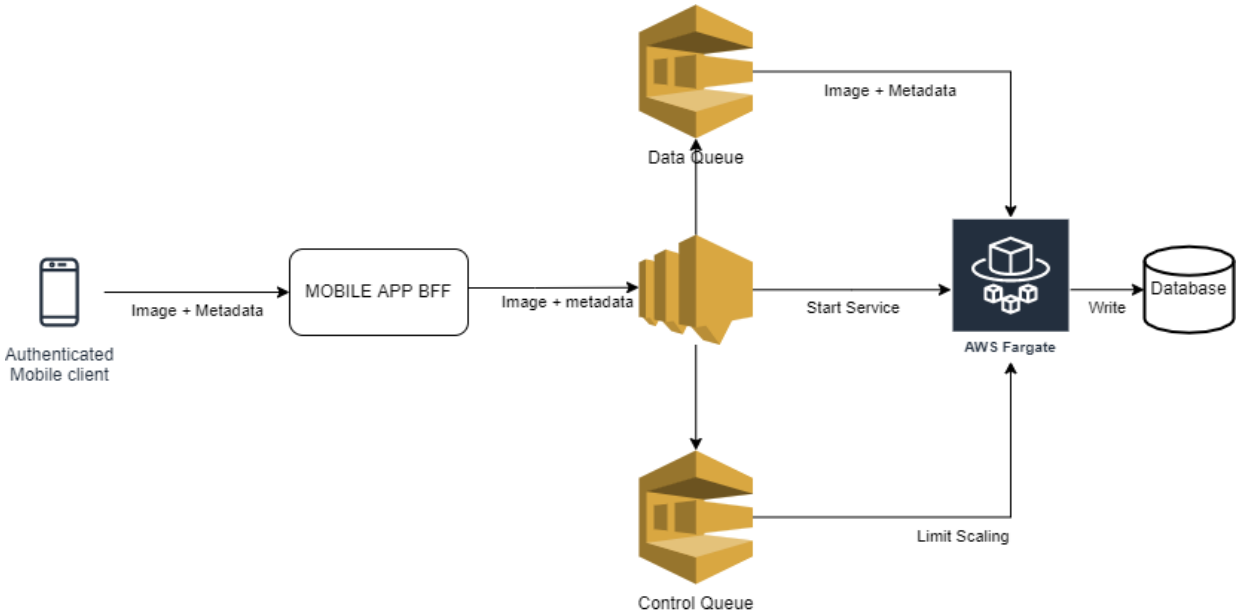

Figure 3.9. Add spot (along with image review) architecture

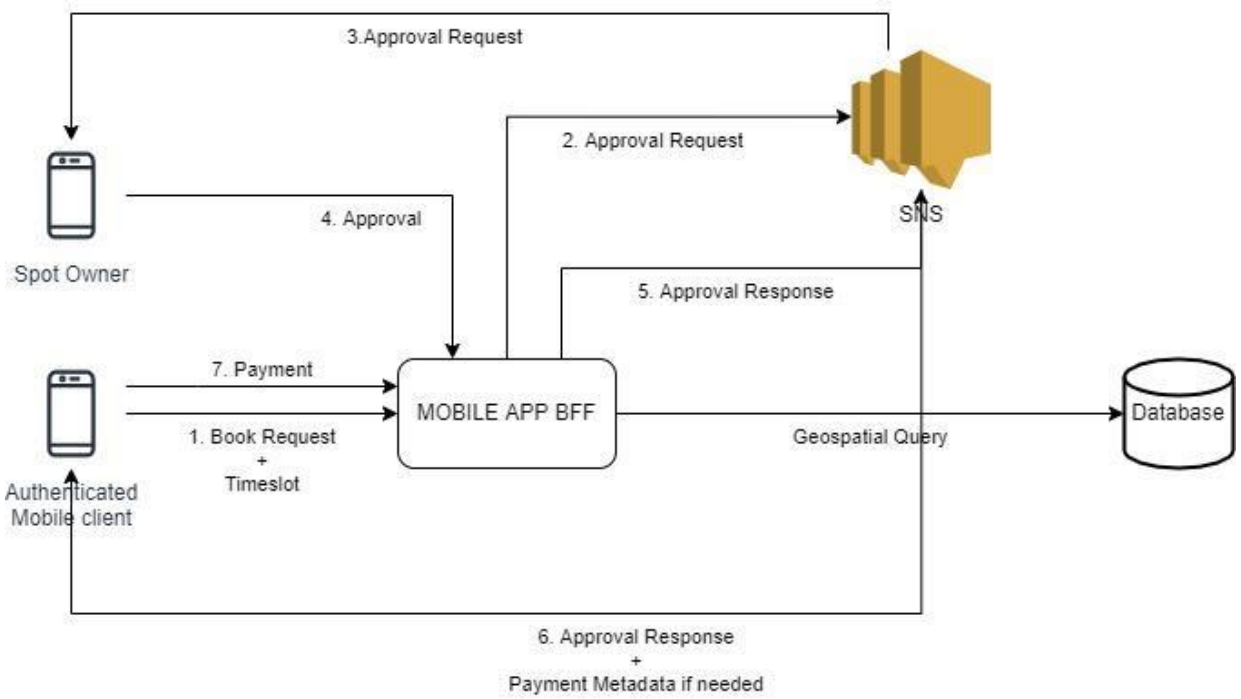

Figure 3.10. Rent spot architecture

Chapter 4: DEVELOPMENT & IMPLEMENTATION

4.1 Developmental Feasibility

The overall developmental feasibility of the application is described below.

4.1.1. Economic Feasibility

The application surely lies in the economically feasible criteria. The cost is low and user will find it to be very cost effective

4.1.2. Technical Feasibility

The architecture of the application ensures that users get the most relevant recommendations according to their preferences and the whole process remains hassle-free.

4.1.3. Operational Feasibility

The application aims to be used by users who want to ensure that they do not have any trouble finding parking spots when traveling to unfamiliar places. Another aim is to ensure that users regularly upload spaces as well to help increase options.

4.1.4. Schedule Feasibility

Given the application's design and phases of implementation, the time frame in which it was carried out is feasible. The application required mostly software implementation. The system was implemented and tested completely during this duration.

4.2 Mobile BFF Endpoints

4.2.1. POST /user/login/otp/validate

Description: This endpoint is responsible for validating OTPs generated through the Login/Sign-In APIs.

Parameters

Name	Description	M/O
OtpValidationRequestBody	*OtpValidationRequestBody*	M

Example

```
{
  "otpId": "string",
  "otp": "string",
  "mobile": "string
}
```

Return Type

ExpiringToken

Content Type

application/json

Responses

As per common return types

4.2.2. POST /user/login/{type}

Description : This API is responsible for generating an OTP for existing users.

Path Parameters

Name	Description	M/O	Default
type	device type	M	null

Body Parameter

Name	Description	M/O
ExistingUser	*ExistingUser*	M

Example

```
{
  "countryCode": "+91",
  "mobile": "stringstri"
}
```

Return Type

OtpVerificationResponseBody

Content Type

application/json

Responses

As per common return types

4.2.3. POST /user/register/otp/validate

Description : This API is responsible for validating OTPs generated through Sign-Up/Register API.

Body Parameter

Name	Description	M/O
OtpValidationRequestB ody	*OtpValidationRequestBody*	M

Example

```
{
  "otpId": "string",
  "otp": "string",
  "mobile": "string
}
```

Return Type

ExpiringToken

Content Type

application/json

Responses

As per common return types

4.2.4. POST /user/register/{type}

Description : This API is responsible for generating OTP for verification of new users.

Path Parameters

Name	Description	M/O	Default
type	device type	M	null

Body Parameter

Name	Description	M/O
UserModel		M

Example

```
{
  "countryCode": "+91",
  "mobile": "stringstri",
  "userName": "string",
  "googleId": "string",
  "email": "string",
  "imageUrl": "string"
}
```

Return Type

OtpVerificationResponseBody

Content Type

application/json

Responses

As per common **return types**

4.2.5. POST /user/logout

Description : This API is responsible for logging out users. It takes in the existing valid authentication key and sets it to expired so the same authentication token cannot be used again.

Header Parameters

Name	Description	M/O	Default
x-sopa-key	authentication key	M	null

Example

```
{
x-sopa-key: "string"
}
```

Return Type

LogoutModel

Content Type

application/json

Responses

As per common return types

4.2.6. GET /reservation/list/created

Description : This API lists the spot reservations created by the user.

Header Parameters

Name	Description	M/O	Default
x-sopa-key	authentication key	M	null

Example

```
curl -X 'GET' \
  'http://example.com/reservation/list/created' \
  -H 'accept: application/json' \
  -H 'x-sopa-key: string'
```

Return Type

array[*ReservationModel*]

Content Type

application/json

Responses

As per common return types

4.2.7. GET /reservation/list/raised

Description : This API lists the spot reservations created on the user's spot.

Header Parameters

Name	Description	M/O	Default
x-sopa-key	authentication key	M	null

Example

```
curl -X 'GET' \
  'http://example.com/reservation/list/raised' \
  -H 'accept: application/json' \
  -H 'x-sopa-key: string'
```

Return Type

array[*RaisedReservations*]

Content Type

application/json

Responses

As per common return types

4.2.8. POST /reservation/create/{spot_id}

Description : This API creates a reservation for the specific spot id.

Path Parameters

Name	Description	M/O	Default
spot_id	spot id	M	null

Body Parameter

Name	Description	M/O
TimeSlot	*TimeSlot*	M

Header Parameters

Name	Description	M/O	Default
x-sopa-key	authentication key	M	null

Example

```
{
  "start": "string",
  "end": "string"
}
```

Return Type

ReservationModel

Content Type

application/json

Responses

As per common return types

4.2.9. POST /reservation/respond

Description : This API is used to respond to reservation requests created on the current user's spot.

Body

Parameter

Name	Description	M/O
RespondReservationModel	*RespondReservationModel*	M

Header Parameters

Name	Description	M/O	Default
x-sopa-key	authentication key	M	null

Example

```
{
  "reservation_id": "string",
  "response": "ACCEPTED",
  "requested_amount": 0
}
```

Return Type

ReservationModel

Content Type

application/json

Responses

As per common return types

4.2.10. GET /spot/search

Description : This API is used to search a parking spot using text by matching the spot titles.

Header Parameters

Name	Description	M/O	Default
x-sopa-key	authentication key	M	null

Query Parameters

Name	Description	M/O	Default
query	spot title to search for	M	null

Example

```
curl -X 'GET' \
  'http://example.com/spot/search?query=query' \
  -H 'accept: application/json' \
  -H 'x-sopa-key: str'
```

Return Type

Array [spots]

Content Type

application/json

Responses

As per common return types

4.2.11. GET /spot/get/{spot_id}

Description: This API is used to fetch a specific spot detail.

Path Parameters

Name	Description	M/O	Default
spot_id	spot id	M	null

Header Parameters

Name	Description	M/O	Default
x-sopa-key	authentication key	M	null

Example

```
curl -X 'GET' \
  'http://example.com/spot/get/id' \
  -H 'accept: application/json' \
```

Return Type

SpotModel

Content Type

application/json

Responses

As per common return types

4.2.12. POST /spot/review/{spot_id}

Description : This API is used to post parking spot reviews.

Path Parameters

Name	Description	M/O	Default
spot_id	spot id	M	null

Body Parameter

Name	Description	M/O
ReviewModel	*ReviewModel*	M

Header Parameters

Name	Description	M/O	Default
x-sopa-key	authentication key	M	null

Example

```
{
  "comment": "string",
  "stars": 5
}
```

Return Type

GenericResponseModel

Content Type

application/json

Responses

As per common return types

4.2.13. POST /spot/get

Description : This API is used to fetch all nearby spots in the provided radius.

Body Parameter

Name	Description	M/O
Location	*Location*	M

Header Parameters

Name	Description	M/O	Default
x-sopa-key	authentication key	M	null

Query Parameters

Name	Description	M/O	Default
Distance	radius for spot search	O	5000
spend_karma	if user wants to spend karma	C	true

Example

```
{
  "latitude": 90,
  "longitude": 180,
  "altitude": 0
}
```

Return Type

Array [spots]

Content Type

application/json

Responses

As per common return types

4.2.14. POST /user/spot/add

Description : This API allows user to add a new parking spot to the database.

Body Parameter

Name	Description	M/O
AddSpotLocationModel	*AddSpotLocationModel*	M

Header Parameters

Name	Description	M/O	Default
x-sopa-key	authentication key	M	null

Example


```
{
  "address": {
    "data": "string",
    "location": {
      "latitude": 90,
      "longitude": 180,
      "altitude": 0
    }
  },
  "images": [
    "string",
    "string"
  ],
  "type": "PUBLIC"
}
```


Return Type

GenericResponseModel

Content Type

application/json

Responses

As per common return types

4.2.15. POST /user/spot/image/add

Description : This API adds images to a temporary S3 bucket and returns image ids which can be used to attach with parking spot details when adding a new parking spot.

Body Parameter

Name	Description	M/O
string	[string]	M

Header Parameters

Name	Description	M/O	Default
x-sopa-key	authentication key	M	null

Example

```
[
String
]
```

Return Type

List [string]

Content Type

application/json

Responses

As per common return types

4.2.16. GET /user/details

Description : This API returns the user details.

Header Parameters

Name	Description	M/O	Default
x-sopa-key	authentication key	M	null

Example

```
curl -X 'GET' \
  'http://example.com/user/details' \
  -H 'accept: application/json' \
  -H 'x-sopa-key: str'
```

Return Type

UserDetailsModel

Content Type

application/json

Responses

As per common return types

4.2.17. PUT /user/details

Description : This API updates user details.

Body Parameter

Name	Description	M/O
UserDetails	*UserDetails*	M

Header Parameters

Name	Description	M/O	Default
x-sopa-key	authentication key	M	null

Example


```
{
  "name": "string",
  "email": "string",
  "dob": 0,
  "address": {
    "data": "string",
    "location": {
      "latitude": 90,
      "longitude": 180,
      "altitude": 0
    }
  },
  "imageUrl": "string"
}
```


Return Type

UserDetailsModel

Content Type

application/json

Responses

As per common return types

4.2.18. POST /user/premium/complete

Description : This API is supposed to be called after the Premium Purchase is complete.

Body Parameter

Name	Description	M/O
TransactionCompleteRequest	*TransactionCompleteRequest*	M

Header Parameters

Name	Description	M/O	Default
x-sopa-key	authentication key	M	null

Example

```
{
  "razorpay_order_id": "string",
  "razorpay_payment_id": "string",
  "razorpay_signature": "string",
  "receipt": "string"
}
```

Return Type

UserDetailsModel

Content Type

application/json

Responses

As per common return types

4.2.19. POST /user/premium/initiate

Description : This API helps initiate a Premium Payment.

Header Parameters

Name	Description	M/O	Default
x-sopa-key	authentication key	M	null

Example

```
{
x-sopa-key: "string"
}
```

Return Type

InitiatePremiumResponse

Content Type

application/json

Responses

As per common return types

4.3 Mobile BFF Models

4.3.1. AddSpotLocationModel

Field Name	M/O	Type	Description
address	M	Address	Address containing geolocation
images	M	List of [string]	List of image urls for spot
type	M	Enum	Spot Type

4.3.2. Address

Field Name	M/O	Type	Description
data	M	String	Address Specification
location	M	Location	Location Specification

4.3.3. ExistingUser

Field Name	M/O	Type	Description
countryCode	O	String	Country Code
mobile	M	String	Mobile Number

4.3.4. ExpiringToken

Field Name	M/O	Type	Description
auth	M	String	Authentication Token

4.3.5. GenericResponseModel

Field Name	M/O	Type	Description
result	M	Boolean	Result
message	M	String	Message

4.3.6. HTTPValidationError

Field Name	M/O	Type	Description
detail		List [*ValidationError*]	Validation Error Detail

4.3.7. InitiatePremiumResponse

Field Name	M/O	Type	Description
order_id	M	String	order id
name	O	String	name of customer
amount	M	BigDecimal	cost of premium
receipt	M	String	razorpay variable

currency	O	String	Currency for amount
prefill	M	PaymentPrefill	Details to prefill for user
key	M	String	transaction key

4.3.8. Location

Field Name	M/O	Type	Description
latitude	M	BigDecimal	Latitude
longitude	M	BigDecimal	Longitude
altitude	O	BigDecimal	Altitude

4.3.9. LogoutModel

Field Name	M/O	Type	Description
result	M	Boolean	Success

4.3.10. OtpValidationRequestBody

Field Name	M/O	Type	Description
otpId	M	String	otp id
otp	M	String	otp
mobile	M	String	mobile number

4.3.11. OtpVerificationResponseBody

Field Name	M/O	Type	Description
otpId	M	String	otp id

4.3.12. PaymentPrefill

Field Name	M/O	Type	Description
email	O	String	email id of user
contact	M	String	contact number of user
name	O	String	name of user

4.3.13. PublicReviewModel

Field Name	M/O	Type	Description
userName	M	String	User Name
imageUrl	O	String	User Image
comment	M	String	Comment
stars	M	Integer	Stars
date	M	Date	Date

4.3.14. PublicUserModel

Field Name	M/O	Type	Description
userName	M	String	User Name
imageUrl	O	String	User Image

4.3.15. RaisedReservations

Field Name	M/O	Type	Description
spot_id	M	String	spot id
name	M	String	name
reservations	M	List [*ReservationModel*]	list of reservations

4.3.16. ReservationModel

Field Name	M/O	Type	Description
reservation_id	M	String	reservation id
spot_id	M	String	spot id
name	M	String	name
timeslot	M	TimeSlotSchema	timeslot
created_at	X	Date	created at
status	M	Enum	reservation status
amount	O	BigDecimal	reservation amount

4.3.17. ResponseReservationModel

Field Name	M/O	Type	Description
reservation_id	M	String	reservation id
response	M	Enum	response
requested_amount	O	BigDecimal	requested amount

4.3.18. ReviewModel

Field Name	M/O	Type	Description
comment	M	String	Comment
stars	M	Integer	Stars

4.3.19. SpotModel

Field Name	M/O	Type	Description
address	M	Address	address with geolocation
images	M	List [string]	image urls
type	M	String	Spot Type
spotId	M	String	Spot Id

reviews	M	List [*PublicReviewModel*]	Reviews
by	M	PublicUserModel	User Details
addedTimestamp	M	Date	Added Timestamp
useCount	M	Integer	Use Count
lastActivity	M	Date	Last Activity
totalRating	M	BigDecimal	Total Rating

4.3.20. SpotModelObfuscated

Field Name	M/O	Type	Description
address	M	null	null for address
images	M	List [string]	image urls
type		String	Spot Type
spotId	M	String	Spot Id
reviews	M	List [*PublicReviewModel*]	Reviews
by	M	PublicUserModel	User Details
addedTimestamp	M	Date	Added Timestamp
useCount	M	Integer	Use Count
lastActivity	M	Date	Last Activity
totalRating	M	BigDecimal	Total Rating

4.3.21. TimeSlot

Field Name	M/O	Type	Description
start	M	String	start time
end	M	String	end time

4.3.22. TimeSlotSchema

Field Name	M/O	Type	Description
Start	M	Integer	start time
End	M	Integer	end time

4.3.23. TransactionCompleteRequest

Field Name	M/O	Type	Description
razorpay_order_id	M	String	razorpay variable
razorpay_payment_id	M	String	razorpay variable
razorpay_signature	X	String	razorpay variable
receipt	X	String	razorpay variable

4.3.24. UserDetails

Field Name	M/O	Type	Description
name	M	String	user name
email	O	String	user email
dob	O	Integer	user date of birth
address	O	Address	user address
imageUrl	O	String	Avatar Image URL

4.3.25. UserDetailsModel

Field Name	M/O	Type	Description
mobile	M	String	Mobile Number
countryCode	M	String	Country Code
email	O	String	Email Id
userName	M	String	User Name
imageUrl	O	String	Avatar Image URL
address	O	Address	user address
karma	M	Integer	user karma coins
credibilityScore	O	Integer	user’s credibility score
dob	O	Integer	user’s date of birth
type	M	Enum	user type
premiumTill	O	Date	premium valid till

4.3.26. UserModel

Field Name	M/O	Type	Description
countryCode	M	String	country code
mobile	M	String	mobile
userName	M	String	user name
googleId	O	String	google id
email	O	String	email id
imageUrl	O	String	image url

4.3.27. ValidationError

Field Name	M/O	Type	Description
loc	M	List [string]	Error at
msg	M	String	Error message
type	M	String	Error type

4.3.27. CommonResponses

Code	Message	Datatype
200	Successful Response	Respective response
422	Validation Error	*HTTPValidationError*

4.4 System modules and flow of implementation

The system can be broadly divided into four modules on the basis of their functionalities. The first module is the authentication module that is responsible for authenticating and authorizing the users. The second module is the spot module that is responsible for the spot functionality. The third module is the in-app currency module that is responsible for the in-app currency functionality. The fourth module is the rent module that is responsible for the rent functionality.

4.4.1 Authentication

Authentication module contains five REST APIs and a function to validate, identify and map incoming requests to their respective user accounts. The five REST APIs include :

- Sign-Up: This API is supposed to take in user details of a new user and create a temporary user entry for the new users and send an OTP to the input user mobile for the verification process.
- Verify Sign-Up: This API is supposed to take in the OTP input from user and validate the temporary account created by the Sign-Up API. After validation, this API creates a permanent user entry for the user and returns an authentication key for the user to validate their future requests.
- Sign-In: This API is supposed to take in login requests from existing users through ther mobile numbers. The send an OTP to the user mobile for verification.
- Verify Sign-In: This API is supposed to verify Sign-In requests generated through the Sign-In API by matching their OTP and otp ids. In case of a successful match, it returns an authentication key that can be attached by the frontend to future request headers to validate requests.

4.4.2 Spot Management

Spot Management Module is responsible for managing all Spot related operations by any user. That includes adding, searching, locating, reviewing, rating and bookmarking spots. A spot can be of type PUBLIC or PRIVATE.

Public Spots are, as the name suggests, public and anyone can freely use those if available. Private Spots on the other hand are owned by the user and requires the other potential users to rent the spot before use.

- Adding a spot requires the user to provide atleast two freshly clicked pictures of the spot along with their geolocation, a spot title, and spot type. Therefore adding a parking spot requires the user to be physically present in the parking spot they're adding. In the backend, we validate the parking spot by ensuring that no other existing spot lies in the proximity of the newly added parking spot to ensure unique spots for our users.
- Searching a spot takes in a string input from user and returns a list of parking spots with matching titles to the input.
- Locating spots takes in the geolocation coordinates from the user's device and returns a list of spots that lie in a 5km radius of the user's location.
- Rating and Reviewing process takes in user's comments and ratings out of 5 for a spot and adds it to the parking spot. The reviews are used to suggest the relevance of a parking spot for users and also helps identify any false spots.
- A spot can be bookmarked by any user for quick access. The adds the spot id to the user's bookmark. This allows the user to access the spot without having to search for it.

4.4.3 In-app Currency

Our solution relies on an in-app currency to provide users with all the app functionalities. The currency is internally termed as "Karma Coins". The user spends karma coins when searching for parking spots. Karma coins can be earned by an user by adding new parking spots. Therefore this module has bindings with Spot module events. Whenever a new spot is successfully added by a user, Karma module kicks in and increases the user's karma coins. This also acts as an incentive for our users to keep adding new parking spots which is eventually beneficial for the application since it depends on crowd sourcing of parking spot data. Although the requirement of in-app currency can be bypassed by any user with a premium subscription when using the app functionalities. They still get to increase the karma coins for future use incase they add more parking spots.

4.4.4 Rent Spot

Our application provides an interface to publish private space as parking spots for rent. Users needing to park in private spots raise a rent request to the spot owner with a date and a time slot. The owner can then provide the rent rates or deny the rent request. Incase the rent rates are acceptable by both parties, the user can park their vehicle in the alloted parking space of the spot owner. In case of any disputes, we block the payment until the dispute has been resolved and agreed upon by both parties. All communication during the complete Rent Spot flow happens through in-app notifications and in-app messaging.

Flow of Implementation

The implementation of these modules were carried in the order of Authentication Module, Spot Module, Rent Module and In-app Currency. We decided to complete the authentication module first because that was the most basic and necessary module which the other modules depend upon. Next we chose to go for Spot Module since that contains the core features of our application and Rent module depended on the completion of the Spot Module. After Spot Module we could parallely contine development on the Rent Module and the In-app Currency since these were now not blocked or interdependent at this point.

4.5 Implementation

The review verification system is currently a very simple one with room for enhancement in the future since our architecture allows for a lot of flexibility and capabilities for the system. Initially we will be implementing a basic word filtering system that will be used to filter out the reviews that use blocked words. Our payment systems had to be carefully planned so as to allow room for fairly solving any disputes between the two parties. We decided to implement a payment system that is based on the Razorpay payment service. Initially on receiving a payment we will be temporarily holding the amount within our system for an agreeable amount of time during which the paying party can raise any disputes which then will be to be resolved with the receiver party under our mediation and only upon the resolution of the dispute will the amount be released to the paying or receiving party. Our parking location recommendations will work on the basis of the nearest and highest rated locations based on the number and recency of reviews, recent reviews being given a higher weightage allowing for any edge cases where a previous popular parking spot might not exist anymore or have gone down in quality.

Chapter 5: RESULTS & TESTING

5.1 Result

The solution presents a smart, crowdsourced parking location search application based on modern cloud technologies. The SOPA system combines a mobile application that offers a variety of features and services, collecting and sharing parking data and incentifying users to do so that makes it a self-sustaining solution. The mobile application is supported by an interconnected microservice architecture on the cloud. This system proposes an all round solution to finding a relevant and safe parking spot. The system will also be extended
to cover additional new features. It will be tested practically in various situations.

5.1.1 App screenshots

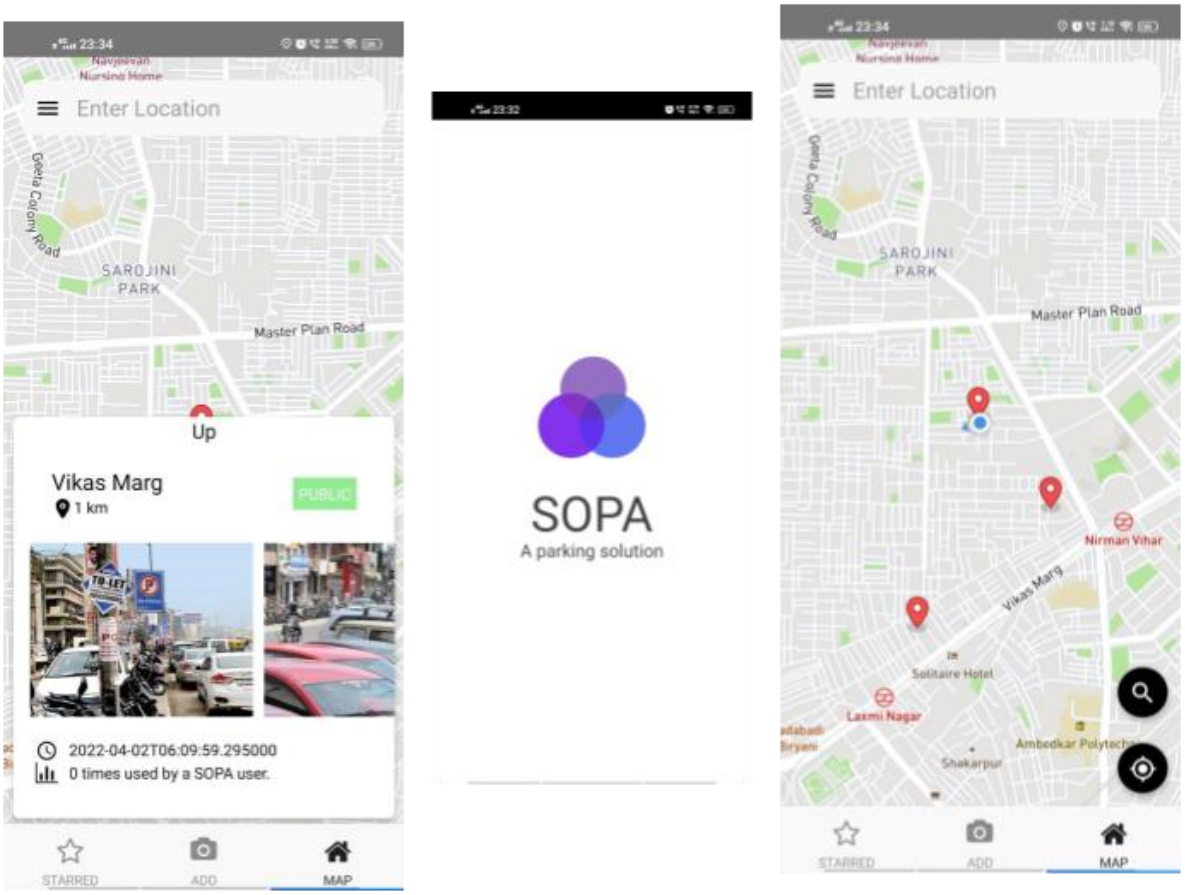

Figure 5.1. SOPA: Map Feature

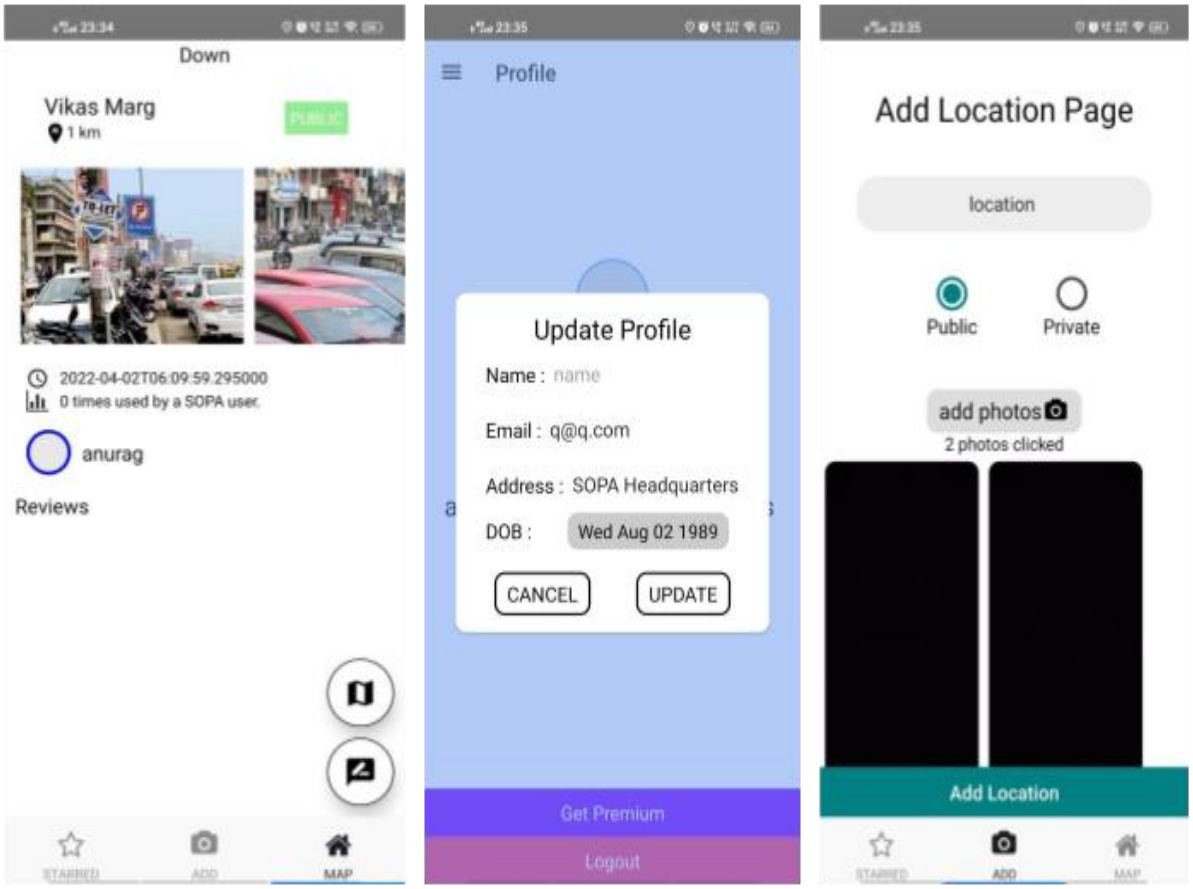

Figure 5.2. SOPA: Location adding and searching features

5.2 Testing

Our application went through extensive testing processes after every update and all new integrations wither with third party or our frontend. Testing on the backend was done by hitting each API with all possible successful case variations and their corresponding failure cases. We tested the frontend by running the app on our personal devices, using the application and reporting our findings.

5.2.1 Success Cases

Figure 5.3. Login -> Send OTP

.

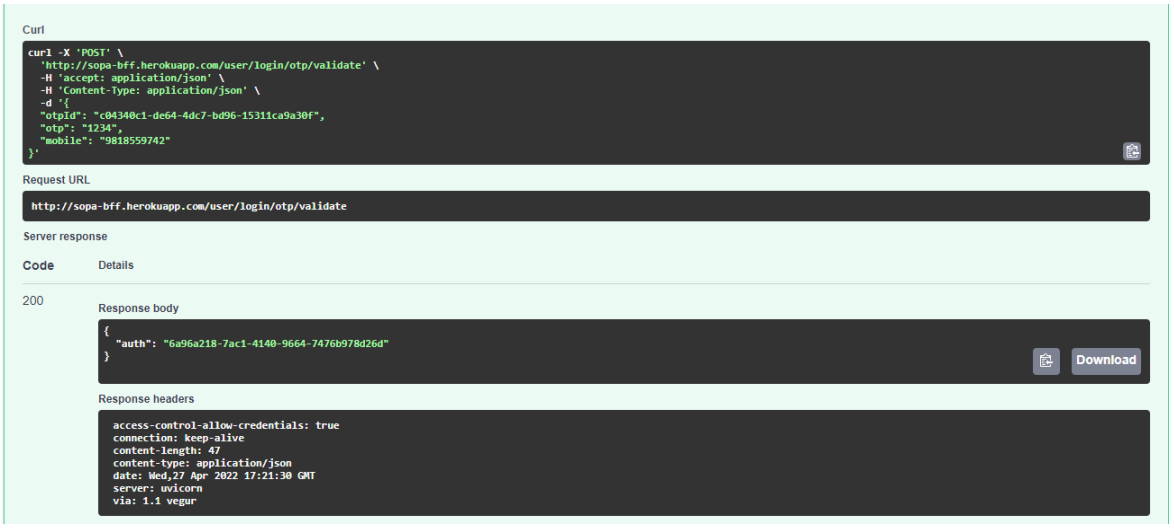

Figure 5.4. Login -> Verify OTP

Figure 5.5. Show user details

5.2.2 Failure Cases

- some private spot might not be owned by the user who posted it
- property validation can be a slow process and create a bottleneck
- false public spots can be added to the database and removing that based on reviews can also be challenging and a time taking process. by the time we remove a false spot, it might have already have degraded many user experiences
- when scaling the application through a city, there will come a point when freemium users will have no true public parking spots to add and therefore no way to earn karma coins forcing them to abandon or switch to premium

5.3 Type of testing adapted

- **Unit Testing :** Unit testing was done by testing the functionalities after implementing each unit or classes on the backend. Every class has a set of testcases which we test for before integrating with the frontend.
- **Integration Testing :** After every integration we test the complete application from the user's perspective by performing multiple tasks and covering all features completed.
- **Regression Testing :** After every new update on the backend, we make sure to test that the existing functionalities do not get affected. This step becomes even easier since we use a microservices architecture which enables us to have small independent services. Therefore the scope of our testing after microservice update lies within that specific updated microservice instead of the complete application since all the microservices function independently.

- **System Testing :** System testing is not required in the case of our backend since it uses Docker Images which can be run on any system with a docker engine and thus are not dependent on the system itself. Frontend of our application though requires System Testing and we tested the Frontend on 4 different devices since we have limited resources or system availability. Although in these four different systems, our application performs fairly well and similarly.

Chapter 6: CONCLUSION & FUTURE IMPROVEMENTS

6.1 Usability of Product/System

SOPA aims to solve the widespread problem of parking. The user can upload parking spot information like pictures, location etc. Users looking for parking spots can rate the locations, to indicate how the experience was. This helps to form a reliable rating system to maintain the quality of the data obtained from crowdsourcing. Users that own a personal space which is unused and would like to lease it out to earn a side income can upload these spaces as well. The application allows other users to see , book and use it for the consent period by paying a fee. Since the product heavily relies on user data for parking spots, it is imperative that users take the time to upload spots as well. To incentivize the process, users are given an in-app currency termed “Karma”. This currency can then be cashed in and user can make use of multiple features like hidden spots, reviews etc.,

6.2 Limitations

- The application depends on human verification which takes time and is very costly to scale thus creating issues when users increase.
- The process of property validation can be a slow process and create a bottleneck
- Since illegitimate parking spaces can also be uploaded , they need to be removed via reviews. This can be a tedious task and a user could already have booked it leading to an unpleasant experience.
- when scaling the application through a city, there will come a point when free users will have no true public parking spots to add and therefore no way to earn karma coins forcing them to abandon or switch to premium,

6.3 Scope of Improvement

The future scope of this system has many points. The first point is to improve spot image verification once the system gathers a bulk of public spot images in the database from crowdsourcing. This can help to build models to better predict if the spot image is relevant and help to add a scoring algorithm for the in-app currency. To prevent use of profanities and indecent remarks, better language checks will be implemented and profanity filters will be incorporated for spot reviews. The use of Machine Learning algorithms will be done to improve the user's experience and generate highly accurate spot recommendations to fit the user's requirements and make the experience smooth and hassle free.

References :

[1] Tian, Zhao, et al. "Determination of key nodes in urban road traffic network." *Proceeding of the 11th World Congress on Intelligent Control and Automation*. IEEE, 2014.

[2] Lam, Patrick TI, and Wenjing Yang. "Application of technology to car parking facilities in Asian smart cities." *Journal of Facilities Management* (2019).

[3] Hallik, Lea, et al. "Reflectance measurements at climate change experiment sites in Europe." *2013 IEEE International Geoscience and Remote Sensing Symposium-IGARSS*. IEEE, 2013..

[4] Ji, Zhanlin, et al. "A cloud-based intelligent car parking services for smart cities." *2014 XXXIth URSI General Assembly and Scientific Symposium (URSI GASS)*. IEEE, 2014.

[5] Assim, Marwa, and Alauddin Al-Omary. "A survey of IoT-based smart parking systems in smart cities." (2021): 35-38.

[6] Cheshire, David. *Building revolutions: Applying the circular economy to the built environment*. Routledge, 2019.

[7] Tingting, Tong, and Zhou Jin. "Space-Time Model of Urban Parking Charge." *2009 Second International Conference on Intelligent Computation Technology and Automation*. Vol. 2. IEEE, 2009.

[8] Milosavljevic, N., & Simicevic, J.. Parking guidance and information system. *Sustainable Parking Management*, 2019.

[9] Di Martino, Sergio, Vincenzo Norman Vitale, and Fabian Bock. "Comparing different on-street parking information for parking guidance and information systems." *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019.

[10] Farooqi, Norah, et al. "UParking: Developing a smart parking management system using the internet of things." *2019 Sixth HCT Information Technology Trends (ITT)*. IEEE, 2019.

[11] Kurek, Agata. "Methods of Parking Measurements—Research of Parking Characteristics in Paid Parking Zones with Dynamic Parking Information." *Research Methods in Modern Urban Transportation Systems and Networks*. Springer, Cham, 2021. 185-193.

[12] Nie, Yan, et al. "Crowd-parking: A new idea of parking guidance based on crowdsourcing of parking location information from automobiles." *IECON 2019-45th Annual Conference of the IEEE Industrial Electronics Society*. Vol. 1. IEEE, 2019.

[13] Caixia, Liu, and Li Daosheng. "Comparative Review of Logistics Park Planning among China, Japan and Germany: response to territorial development in China." *2021 40th Chinese Control Conference (CCC)*. IEEE, 2021.

[14] Zhang, Zusheng, et al. "Collaborative sensing-based parking tracking system with wireless magnetic sensor network." *IEEE Sensors Journal* 20.9 (2020): 4859-4867.

[15] Jung, Andrea, Paul Schwarzbach, and Oliver Michler. "Future Parking Applications: Wireless Sensor Network Positioning for Highly Automated in-House Parking." *ICINCO*. 2020

[16] Anand, Abhijeet, et al. "Smart Parking System (S-Park)–A Novel Application to Provide Real-Time Parking Solution." *2020 Third International Conference on Multimedia Processing, Communication & Information Technology (MPCIT)*. IEEE, 2020.

[17] Ang, Jin Teong, et al. "iSCAPS-Innovative Smart Car Park System integrated with NFC technology and e-Valet function." *2013 World Congress on Computer and Information Technology (WCCIT)*. IEEE, 2013.

[18] Ji, Zhanlin, et al. "A cloud-based car parking middleware for IoT-based smart cities: Design and implementation." *Sensors* 14.12 (2014): 22372-22393.

[19] Gamal, Omar, et al. "Assistive parking systems knowledge transfer to end-to-end deep learning for autonomous parking." *2020 6th International Conference on Mechatronics and Robotics Engineering (ICMRE)*. IEEE, 2020.

[20] Alshehri, Faris, et al. "Smart Parking System for Monitoring Cars and Wrong Parking." *2019 2nd International Conference on Computer Applications & Information Security (ICCAIS)*. IEEE, 2019..

[21] Enríquez, Fernando, et al. "Existing approaches to smart parking: An overview." *International Conference on Smart Cities*. Springer, Cham, 2017.

[22] Chen, Mingsong, Runze Fang, and Lei Peng. "Exploration of Parking Guidance based on Vehicle Crowdsourcing." *2020 IEEE 5th International Conference on Signal and Image Processing (ICSIP)*. IEEE, 2020.

Bibliography:

- https://www.proptiger.com/guide/post/in-the-right-direction-why-india-needs-a-better-parking-policy
- https://blog.getmyparking.com/2019/02/14/issues-with-parking-in-indian-metropolises/#:~:text=One%20of%20the%20most%20common,are%20some%20problems%20that%20result.
- https://play.google.com/store/apps/details?id=com.parkingrhino&hl=en_IN&gl=US
- https://www.researchgate.net/publication/338161212_Evaluation_of_Parking_Characteristics_A_case_study_of_Delhi
- https://play.google.com/store/apps/details?id=com.theparkingspot.tpscustomer&hl=en&gl=US