

Travel Buddy- An Online Partner Sharing Software

Project Report

Submitted By:

Eshwar S – 16BCE0596

Rishab Gupta – 16BCE0757

Sharon Shibu – 16BCE2157

Sarthak Dahiya – 16BCE0782

Amreeta Joseph – 16BCE2206

Bharath R – 16BCE2252

Course Code: CSE4028

Object Oriented Software Developement

Professor Kumar K

School of Computer Science and Engineering



APRIL 2019

CERTIFICATE

This is to certify that the project work entitled “Travel Buddy” that is being submitted by Eshwar S (16BCE0596), Rishab Gupta (16BCE0757), Sharon Shibu (16BCE2157), Sarthak Dahiya (16BCE0782), Amreeta Joseph (16BCE2206) and Bharath R (16BCE2252) for the course CSE4028 - “Object Oriented Software Development” is a record of bonafide work done under my supervision. The contents of this project work in full or in parts, have neither been taken from any other source nor have been submitted for any other course.

Signature of Faculty

Prof. Kumar K

ACKNOWLEDGEMENT

We would like to thank our professor Kumar K for his precious guidance and constant support and the Dean of School of Computer Science and Engineering for their pleaded information and opportunity given to us for completion of our project.

INDEX

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	5
1	INTRODUCTION	6
1.1	PURPOSE	6
1.2	PROBLEM STATEMENT	6
1.3	MOTIVATION	6
1.4	PROJECT SCOPE	6
1.5	CHALLENGES	7
2	LITERATURE SURVEY	7
3	DETAILED DESIGN OF THE PRODUCT	9
3.1	LOGICAL ARCHITECTURE	9
3.2	FLOWCHARTS	11
3.3	USE CASE DIAGRAM	12
3.4	DETAILED DESIGN	13
3.5	MODULE DESCRIPTION	16
3.6	DOMAIN AND ANALYSIS	18
4	IMPLEMENTATION / PSEUDO CODE	19
4.1	GRAPHS / UI	19
4.2	CASE STUDY	21
5	COMPARATIVE ANALYSIS	23
5.1	COMPARISON GRAPH	23
5.2	SPAM ALGORITHM ANALYSIS	30
6	CONCLUSION AND FUTURE SCOPE	30
7	REFERENCES	31
8	APPENDICES	32
8.1	APPENDIX A: PROJECT TIMELINE	32
8.2	APPENDIX B: FINAL CODE	32
8.3	APPENDIX C: FINAL SCREENSHOT	42

ABSTRACT

Travel Buddy is an interactive carpooling service which provides the students of the college with an easier, cheaper and a safer option while traveling. Our project aim is to build the database for such a system and also provide the web interface for it. For solving this problem we will be creating a database of details, and then using exceptional handling to manage it. We provide traveler with travel companions, with whom the cost is shared. We provide companions on the basis of date of travel as well as the destination of travel. We will be taking the data from users like date, time, destination, no, of members, preferences, gender, and their username which will act as the unique ID. Then using the database management tools a user can either create a new cab for a journey thus becoming the admin for the cab or choose the cabs already created by users by sending requests to the admin of the cab. By this process a student using this website can share the cab with other students registered on this website and hence can minimize the cost of traveling.

The company's primary task is building information platform for fixed time and routes carpooling, which is the basis for connecting both sides of carpooling. Also, the company is responsible for checking the effectiveness and legitimacy of the information of drivers and passengers. Besides, it is necessary to focus on those participants with illegal violation and bad credit history.

Drivers register or login in the information platform and release the carpooling information of running time, location, and routes. Passengers register, login in, and submit the carpooling application with trip time, location, and routes information. Then, the information platform summarizes, checks, and matches all the information from drivers and passengers. Once successful, the system will send matched results to participants by computer and mobile client automatically. If participants are satisfied with matched results, the carpooling will be implemented on time. Otherwise, system must match the information circularly. Drivers or passengers can also directly select each other according to the information displayed without automatic matching by system. Participants will feedback their selections to the system. After completing, both sides of carpooling will make mutual evaluation on the platform. In order to improve the convenience of carpooling and mutual evaluation, the WEBSITE/mobile APP software can be developed later so as to improve dynamic real-time

1. INTRODUCTION

1.1. PURPOSE

Given the travelling details of a person (student), the aim is to find cabs having passengers travelling to the same destination during the similar time frame. The current method which students use for finding people to travel with together is asking around in the friend circle, using WhatsApp groups and trying to get at least one person to travel with. All this takes a huge effort and isn't efficient as there are thousands of students and you can reach hardly a hundred students via this way. If a person does not find anyone, he has to travel alone leading to situations like expensive cost, safety concerns and loneliness.

The purpose of this software is to make it easier for all the travelers to find people to travel with all in one comprehensive setup.

1.2. PROBLEM STATEMENT

Our project aim is to build the database for an interactive car pooling service which provides the students of the college with an easier, cheaper and a safer option while traveling system and also provide the web interface for it. For solving this problem we will be creating a database of details, and then using exceptional handling to manage it. We provide traveler with travel companions, with whom the cost is shared.

1.3. MOTIVATION

This product provides with a platform to make communication and bookings between the users of the application more accessible which provides an ease for the users as the provides them with the functionalities provided which are the cab sharing feature of the application and provides them with different functions for the same and also the contact details of other passengers travelling with them. We will provide the interface for different users of the software/product as per their functionality.

1.4 PROJECT SCOPE

Our solution of an online cab sharing system is aimed at finding a buddy to travel with with minimal effort and the additional benefits of travelling together like cost splitting, safety and social exposure. The trips can be pre-scheduled as the dates for travelling between cities during vacations are close by to each other. Also, the destinations are specific and limited to a few like airports, railway or bus stations, nearby metropolitan cities or vacation destinations. This allows the system to be feasible and find buddies. The concept which we have used is that a user can view the current trips and find one matching his travel details on one single page of the view cabs. The user can search through the cabs manually, they are by default arranged in an increasing order of departure time and date. He can also use filters like students of the same year, branch, etc.

This gives the option to find preferred buddies and have people with common interests together while travelling. After finding a cab, the user can send a request to the cab admin to join the cab, the cab admin can then have a look at the user's profile and accept or decline the request. If the request is accepted, the details of the passenger are added to the cab and details like phone no, etc are shared with the group of the cab. Although if the user doesn't find any cab matching his/her requirements, he/she can create a new trip and enter the details of the trip. After filling this form, he is made the admin of the cab and can accept or decline request from other users who wish to join the cab. Once the limit of the cab has been reached, the trip is removed from the list and more users cannot request to join the cab. Thus, excess number of people in a single cab can be avoided. The finalized group can then go ahead and book just a single cab. Thus, the portal provides a way for students to find cab buddies from all over the college and thus the probability of not finding a buddy to travel with is reduced drastically due to the exposure to all the other students on a common platform.

1.5 CHALLENGES

Implementation of the GSM module is a challenging task. The **module** uses the SIMComm function. It is able to communicate easily with the **module** through AT commands. **GSM** library contains several methods of communication with the **module**. The **GSM modem** can work with any **GSM** network operator SIM card like a mobile phone with its own unique telephone number. But implementation of this module is challenging due to our lack of knowledge in this field.

Our chatbot uses seq2seq model based on neural networks, and training the network is going to be a hard and time consuming task.

2. LITERATURE SURVEY

The adverse effect of mechanized private versatility on the earth can be diminished effectively by urging more individuals to carpool. From a mental point of view, just little is thought about the determinants of carpooling. Hence, this examination researched carpooling conduct dependent on a hypothetical foundation that coordinates the hypothesis of arranged conduct, the standard enactment model, and dispositional trust. Also, we considered carpooling from two separate points of view: Passengers sharing rides, and the drivers offering rides. We directed a review with an agent test of 342 members in Switzerland. The outcomes demonstrated that for both, travelers and drivers, regularizing angles, for example, distinct and individual standards, in a blend with apparent social control anticipated carpooling goal. The disposition toward carpooling conduct, be that as it may, did not have any prescient power with respect to carpooling expectation, neither for travelers nor drivers. Dispositional trust showed a roundabout impact on aim to carpool as a traveler or driver through saw social control. In light of these outcomes, we talk about useful ramifications for planning measures to advance carpooling effectively later on.

The paper presents a multiple criteria (MC) formulation of the carpooling optimization (CO) problem and a solution procedure that allows to solve it. The mathematical(1) model of the MCCO problem includes two major sub-problems, such as planning of the routes and matching carpoolers (drivers and passengers). Different aspects, including: economic, social, technical and market oriented are considered. The MCCO problem is solved with the application of an original

computational procedure based on the multiple criteria genetic algorithm, called NSGA II and the solutions' analysis and review technique, called Light Beam Search (LBS) method. The former method allows to generate a set of Pareto optimal solutions,(2) while the latter assists the carpoolers in finding the most desired compromise solution (common route and match between carpoolers). The results of computational experiments are presented. We find that solving the formulating carpooling problem in a heuristic manner is possible in reasonable time. One approach to guarantee supportable and ecological amicable versatility is the utilization of less vehicles for conveying more travelers, and carpooling is a way to accomplish this[3] objective. One significant worry in carpooling administrations is identified with trust, as carpooling clients need to either share their vehicles, in the event that they go about as drivers, or travel with outsiders on the off chance that they go about as travelers.

One approach to handle trust concerns is the usage of client notoriety evaluation components, whose goal is to give positioning of clients regard to their conduct, based on criticism given by different clients. This paper displays a recently presented notoriety evaluation instrument for carpooling applications,[4] which, notwithstanding input given by different clients, considers client travel inclinations. Primer test results have demonstrated that the proposed

instrument is hearty against assaults by malevolent clients, on their endeavor to imperil the dependability of the framework, as it jelly the genuine notoriety scores of the clients, when

the entrance rate of noxious clients increments. Carpooling is viewed as an option, reasonable, socially satisfactory and earth inviting mean of transport, which relates to the advanced financial difficulties and the movement of social changes. It is giving to the drivers and the travelers the chance to travel together utilizing a similar vehicle. Since it involves a promising versatility arrangement, its advancement to the more youthful ages that are building up their movement propensities would be increasingly proficient. Thessaloniki is a Greek medium measured city, which has numerous youngsters originating from territories everywhere throughout the nation to learn at the scholarly or specialized/proficient dimension.

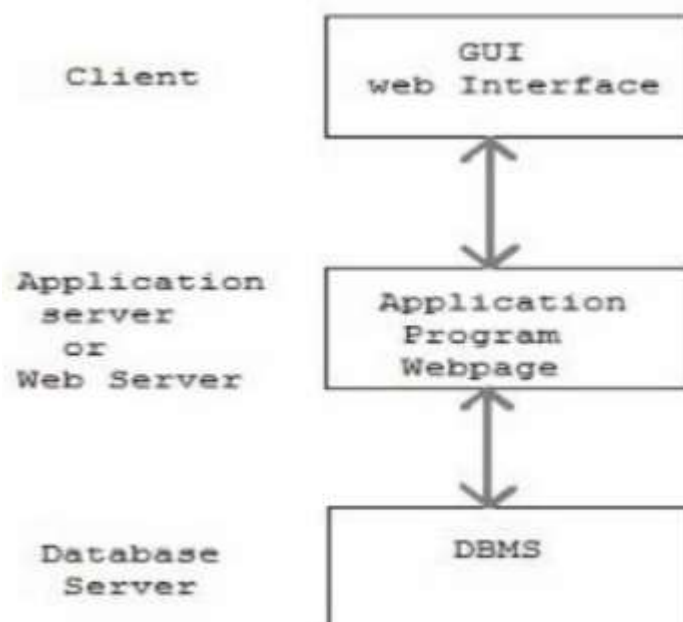
This paper researches the presentation of a carpooling framework serving the interurban excursions of the understudies of Thessaloniki to their close-by home urban areas and the other way around. At present, the nonappearance of numerous elective portability alternatives, the expanded expense of vehicle driving furthermore, support and the low dimension of administration given by the current interurban transport arrangement of Thessaloniki oblige the versatility. In light of the discoveries got from the audit of University carpooling plans, an on-line survey was planned and an the broad poll study was led, being routed to the college understudies of the city of Thessaloniki (in particular, understudies of Aristotle University of Thessaloniki (AUTh), University of Macedonia and Technical Institution of Thessaloniki) in request to investigate their inclinations with respect to the production of a carpooling framework, being overseen by and given only to the individuals from the scholarly network. The examination of the review featured the positive position of the respondents. In view of the overview's discoveries the carpooling's plan of action and the stage's structure were proposed to serve the understudies and the scholastic also, the regulatory staff of the previously mentioned Universities Roughly 40 percent of fuel utilization in vast urban areas is identified with transportation. A perceptible measure of fuel is squandered because of traffic clog in pinnacle hours. Transportation organizers search for arrangements to lessen blockage to spare fuel and increment vitality productivity. One of the arrangements is carpooling that stresses on a mutual

utilization of private vehicles. In this paper, the factors which induce voyagers to pick carpooling are examined for Tehran city, capital of Iran. An expressed inclinations (SP) review has been utilized to watch voyagers' propensity of carpooling. SP is a review system which numerically demonstrates the inclinations, in light of individuals' expressed decisions and their reactions to speculative circumstances. The overview surveys rounded out by 470 voyagers utilized their very own cars. Thinking about the information, carpooling impacts are dissected in various circumstances. In this methodology an interest work is aligned and used to anticipate percent of explorers pick carpooling. At the point when every single intrigued explorer, autonomy[5] of knowing suitable rideshare or not, pick carpooling at that point vehicle trips every day would diminish around 780000 vehicle trips for every day and decrease yearly fuel utilization by 336.53 million liters. The outcomes demonstrate that if proper procedures like carpooling sites are intended to help voyagers for recognizing suitable rideshares, carpooling would increment by 30 percent and this expansion will diminish yearly fuel utilization around 240 million liters. Results additionally demonstrate that high inhabitation vehicle paths (HOV) that diminish travel time for ridesharing may not very impact on carpooling inclination of explorers.

3. DETAILED DESIGN OF THE PROJECT

3.1 LOGICAL ARCHITECTURE

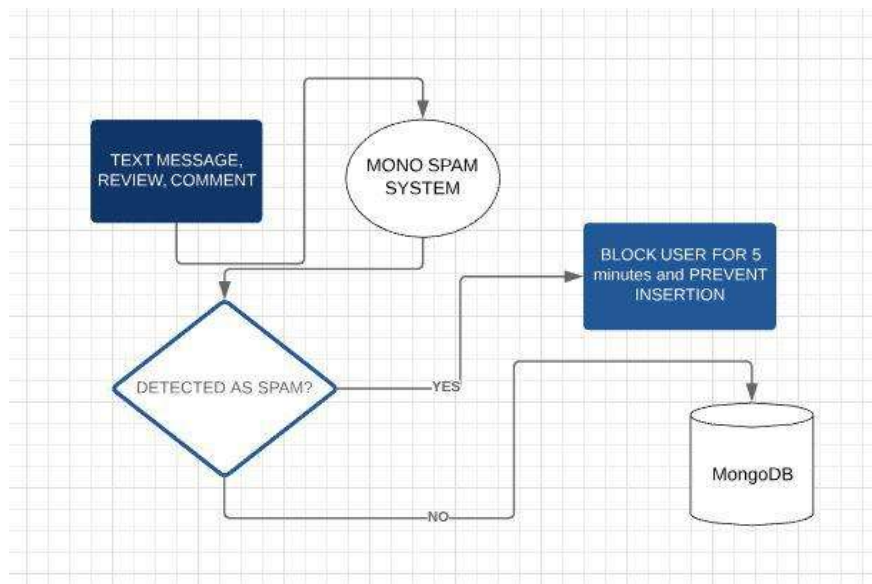
The architecture provides the top level design view of a system and provides a basis for more detailed design work. The architecture adopted is client server architecture, an intensive 3 layered system. Figure showing the 3 level client server application



Client/Server Architecture is a model that helps simplifies the development of the software by dividing the process into client and server. This model is our main architectural design, because this architecture aims at allowing access for one or more clients to several resources or functionalities in a central server; it separates our system requirements into two easily programmable systems. First, the client, which acts as the User Interface, requests data from the server, and waits for the server's response. Secondly, the server, which authorises what privileges each user is granted based on their

account type. In the Client/Server model, there are many types of tiers available to describe the system architecture. These tiers are meant to physically separate applications into different concentrations that can be distributed between client and server. Typically, the tiers are categorized by application, presentation, and data processing. Usually the tiers are separated as presentation, middle, and data. Presentation tier is where users interact with the applications. Middle tier is the communication medium for both presentation and data tier. Data tier is where the data is located, saved, and modified. Travel Buddy will have a 3-tier Client-Server architectural style as we have a presentation tier, logic tier, and data tier. Starting from the lowest tier, Travel Buddy has a data tier to store and retrieve user information, emails, chat information, forums, etc. The middle tier, which is the logic tier, coordinates and moves information back and forth between the data and the presentation tier. The logic tier also evaluates, decides, and processes commands. The top most tier is the the presentation tier. The presentation tier translates tasks and results into something a user can understand. It will translate it to a webpage in which the user can easily navigate. For example, the user will enter information and hit “login” on the presentation tier, the presentation tier will send it to the logic tier and the logic tier will decide what to look for. After it decides what it wants, it sends it to the data tier and will retrieve the user information from the data tier. The logic tier will then decide if the information sent from the presentation page matches with the data, and return a result of successful or unsuccessful login.

SPAM DETECTION OF CAR POOLING SYSTEM ARCHITECTURE



Subsystem, Component, or Module 1 ...N

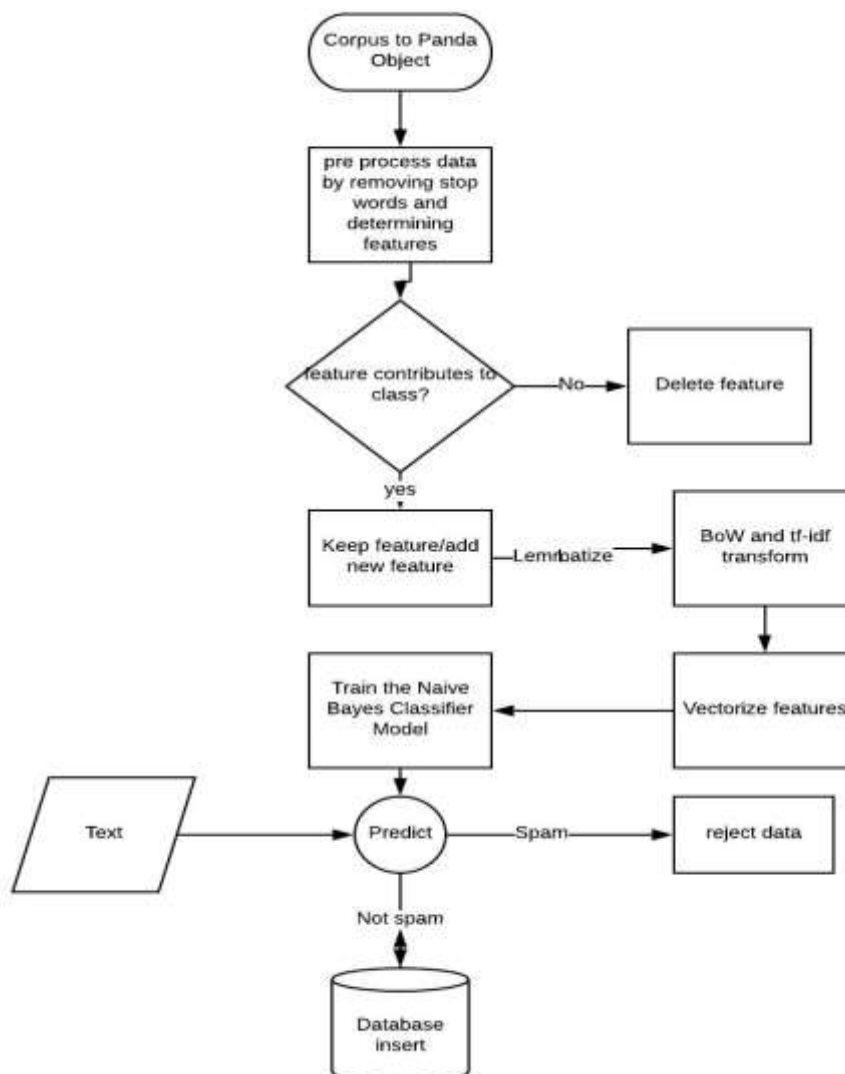
Three subsystems are created to help achieve the architectural style.

- **Website (Front End):** This subsystem deals with user interactions, where the clients may interact with the server through the Travel Buddy website. It maintains the display and connection between the database and user commands to retrieve and send data.
- **Application:** This subsystem is the connection between the users and the data with the technical operations. It consists of the functions that is needed to connect the front end and back end.
- **Database (Back End):** The database subsystem handles data management and new data. It maintains the flow of data processes.

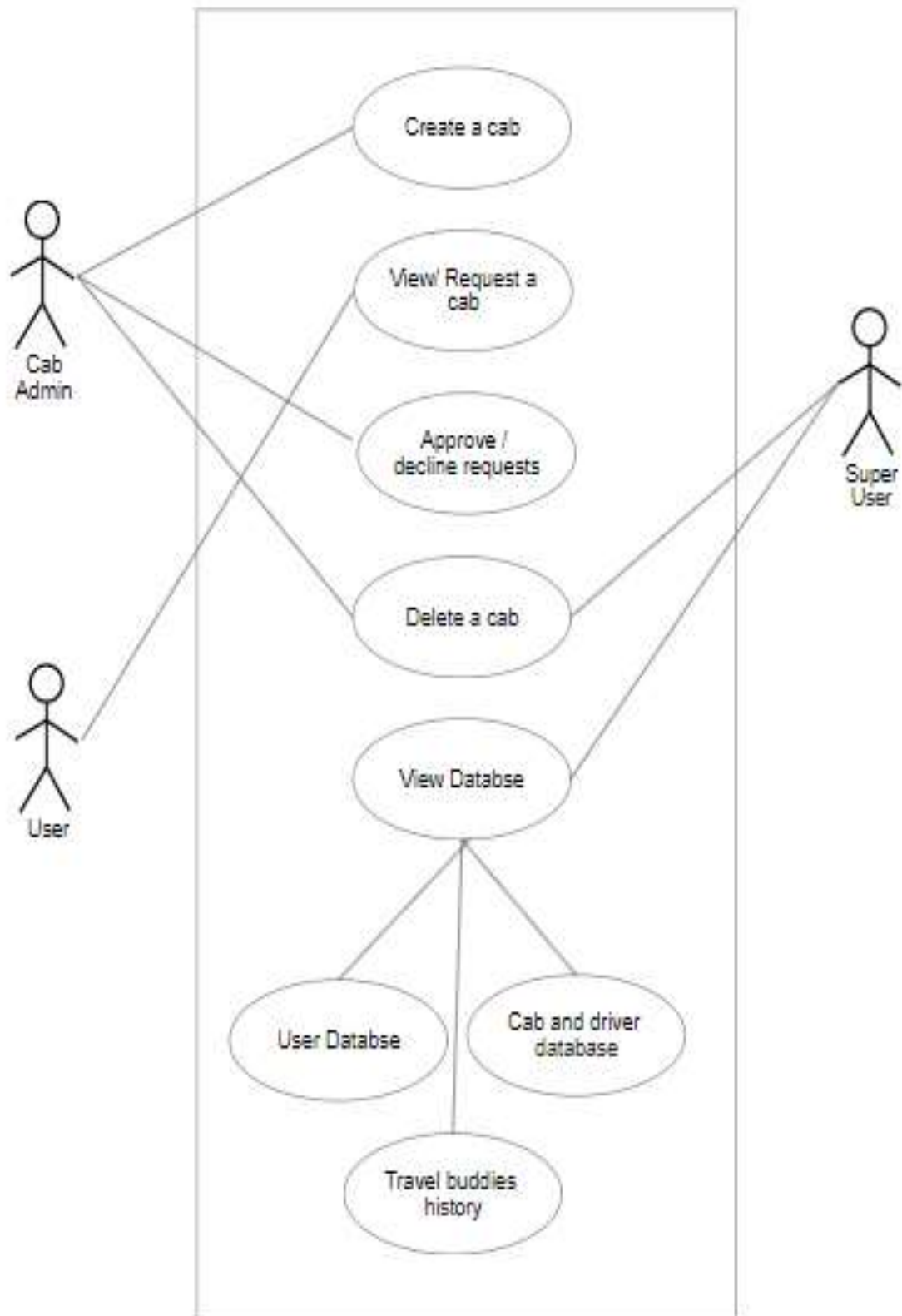
Strategy 1...N

The Travel Buddy uses an object oriented design paradigm approach for better implementation. The advantage of this has been described in the later course of this document.

3.2 FLOWCHART



3.3 USE CASE DIAGRAM



3.4 DETAILED DESIGN

Introduction

3.4.1 Purpose

This software design specification is made with the purpose of outlining the software architecture and design of the Travel Buddy Application. The document will provide developers an insight in meeting the needs and requirements efficiently and effectively. Moreover, the document facilitates communication and understanding of the system by providing several views of the system design. This design will detail the implementation of the requirements as defined in the Software Requirements Specification for designing the Travel Buddy Application.

3.4.2 System Overview

This product provides with a platform to make communication and bookings between the users of the application more accessible which provides an ease for the users as the provides them with the functionalities provided which are the cab sharing feature of the application and provides them with different functions for the same and also the contact details of other passengers travelling with them. We will provide the interface for different users of the software/product as per their functionality.

3.4.3 Design Map

The following document contains the design specifications for the Travel Buddy Application. The first section of the document formulates the design considerations and entities to be kept in mind before indulging into implementing design paradigms.

3.4.4 Definitions and Acronyms

User Profile: An identity users use to display themselves as in the software. There is a page dedicated for every user to customize their personal data with personalized editable content (i.e. email, phone etc).

Forum: An interactive portal where users can post about their upcoming trips and other people travelling to and from the same locations can request to join.

Administrators (Admins): Developers and creators of the software application responsible in maintaining and managing the system architecture.

3.4.5 Design Considerations

This section contains the design considerations kept in mind for designing the Travel Buddy Application.

3.4.5.1 Assumptions

The database is assumed to be active every time for accessing the system. The user is supposed to have net connectivity at all times when accessing the system.

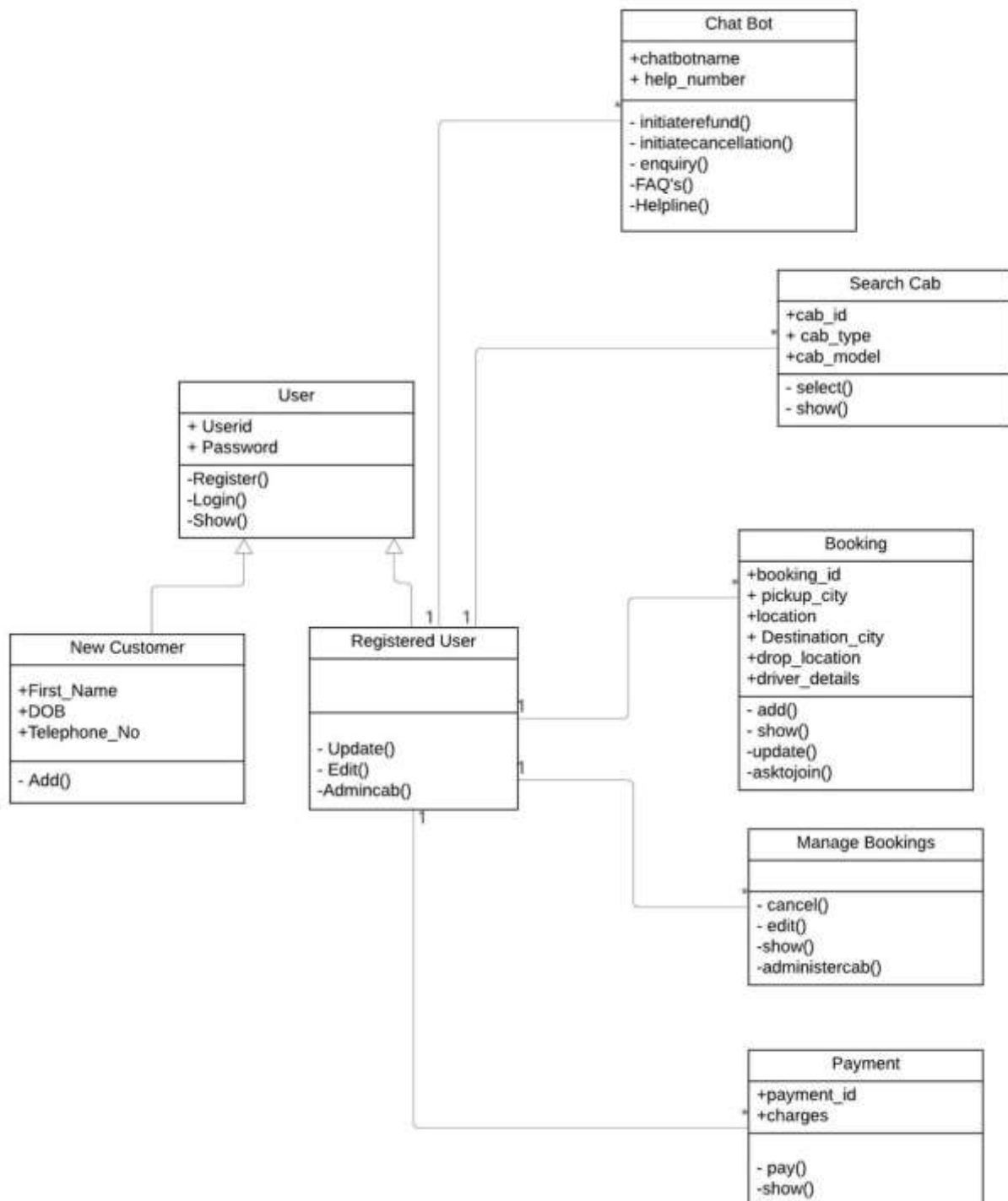
3.4.5.2 System Environment

This product will be an online application which will reside in the client's machine which will be a Windows operating system. It will be available to all the users connected with the platform. The database used to store the data will be SQL server.

3.4.5.3 Design Methodology

The class diagram is best suited to cater to the design methodology of the tool.

CLASS DIAGRAM:



3.5 MODULE DESCRIPTION

- Class: User – The customer who wants to book a cab through our platform.

The following operations are performed:

Register() – The new users are required to register first to create a profile of the user.

Login() – The user is required to enter the userid and password which is verified.

Show() – The user_id and password is displayed.

- Class: New Customer – First time customers are required to first register by filling the form.

The following operations are performed:

Add() – The customer needs to provide his/her name, date of birth, telephone no and create login details.

- Class: Registered User - Once a registered user is logged in, they can perform several tasks like becoming an admin, update, etc.

The following operations are performed:

Update - The users can update their trip/information.

Edit() – The users can edit/modify their trip/information.

Admincab() – The user can become an admin of a cab and he/she can choose the passengers they want to travel along with.

- Class: Chatbot – The chatbot uses AI based algorithms which can cater to the needs of the user.

The following operations are performed:

Initiatefund() – The chatbot can initiate the transfer of funds.

Initiatecancellation() – If the user wants to cancel a cab then this can be done easily through the chatbot.

Enquiry() – If the user has any problems, they can ask the chatbot of answers.

FAQ's() – The frequently asked questions are displayed to the user along with the answer to the user through the chatbot.

Helpline() – If the problem is not resolved through the chatbot then the chatbot provides the user with an 24x7 helpline.

- Class: Search cab – The user can search the cab and it is displayed along with the cab details.

The following operations are performed:

Select() – The user can select a desired cab.

Show() – The cab details such as cab_id, cab_type, cab_model are displayed to the customer.

- Class: Booking – The user can book a cab.

The following operations are performed:

Add() – The user can create a booking.

Show() – The details of the trip such as booking_id, pickup_location, destination_location and driver details are displayed to the user.

Update() – If there are any changes the user can update/modify the trip.

Asktojoin() – If the user is not the admin of the cab then he/she can request the admin of a cab if they can join the ride.

- Class: Manage bookings – Any changes can be made to the bookings through this module.

The following operations are performed:

Cancel() – The cab can be cancelled.

Edit() – Changes to the trip can be made according to the user.

Show() – All the bookings done by the user can be viewed.

Administercab() – The admin of the cab can accept/decline another user's request to travel together.

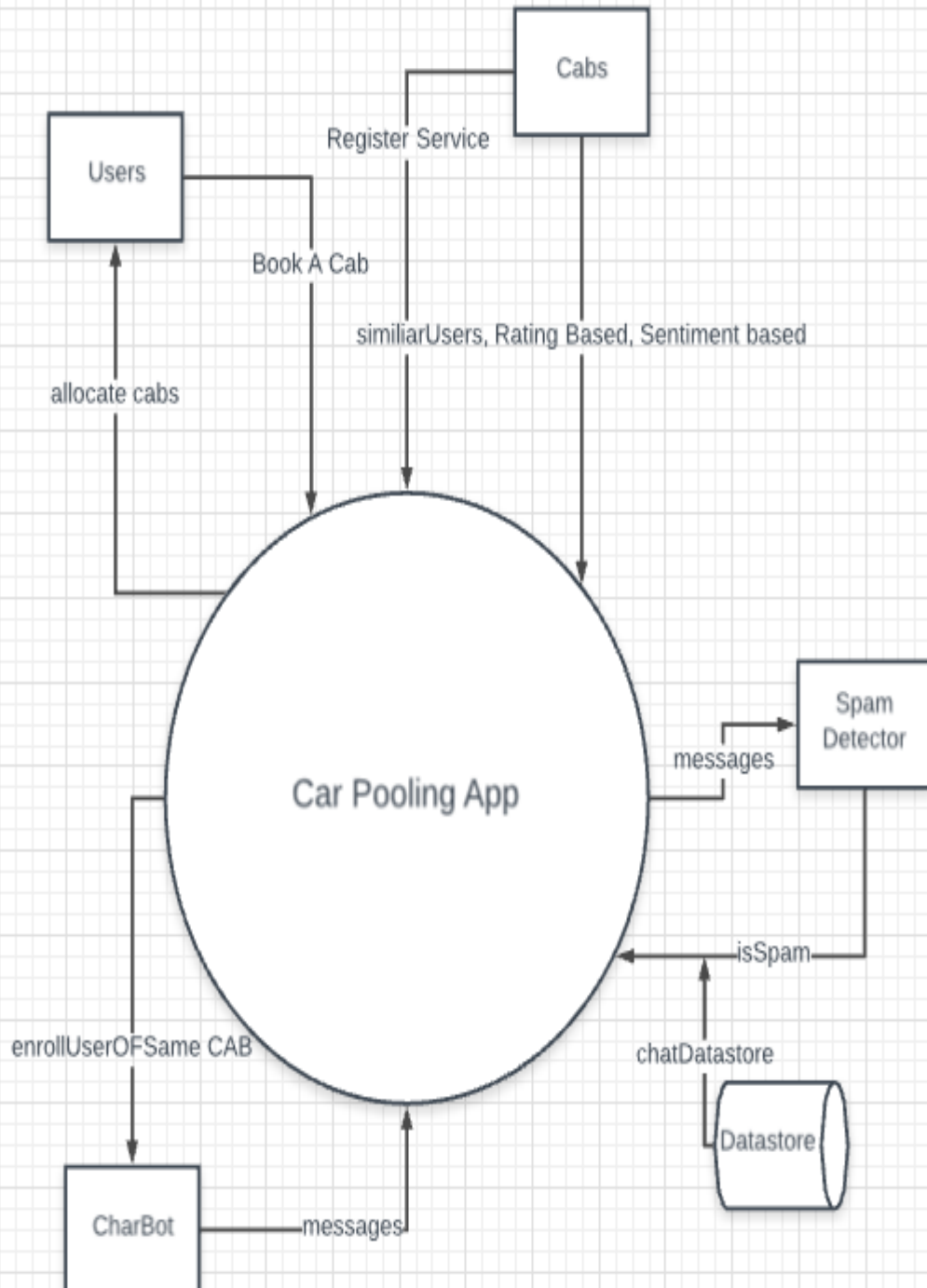
- Class: Payment – The users can pay through various means and their invoice is displayed.

The following operations are performed:

Pay() – The user pays the displayed amount through one of the provided means.

Show() – The invoice and acknowledgement slip is displayed to the user.

3.6 DOMAIN MODEL



4. IMPLEMENTATION / PSEUDO CODE

4.1 GRAPHS / UI

4.1.1 The Login Page



The user will be directed to this page, where he/she will have to login. The user will be provided with a username and password entering space. They can either login or, if they've forgotten their password then go through that option.

4.1.2 Home Page

The user here, will have the option to either book a cab, create a new one. They can also get familiar with the criteria and working of TravelBuddy or logout. All of these are present in the top navigation bar. There is a small description, on the main portion of the space with the same options also present on the right bottom.

BOOK A CAB

Find a cab, with an availability and a time and destination that suits you.

NAME	DATE	PLACE	PEOPLE	
Salonee	2.3.18	Delhi, India	3/5	Book
Salonee	2.3.18	Delhi, India	3/5	Book
Salonee	2.3.18	Delhi, India	3/5	Book
				Create a Cab >

4.1.3 BOOK A CAB

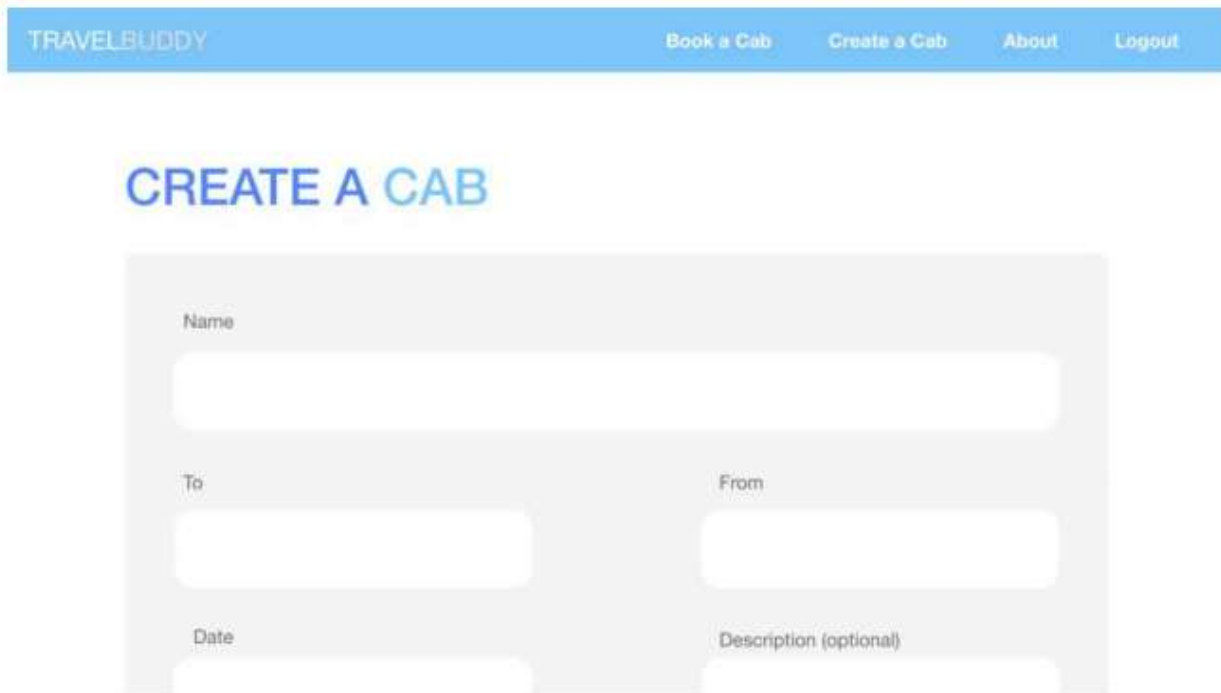
BOOK A CAB

June

This page will have a calendar, in which the user can choose the date he/she would like to travel. This would be all scrollable to create an ease of access. After, selecting a date, the user can choose from the options available for the cabs and confirm his ride.

4.1.4 Create a Cab

If the user wants to create his own cab, as none of the current options suit their need they can through this page. The form requires your name, and travel details as well as a contact number. The form, will send a notification to the creator, whenever someone wishes to join the cab.



The screenshot shows the 'CREATE A CAB' form in the TRAVELBUDDY application. The header bar is blue with the text 'TRAVELBUDDY' on the left and navigation links 'Book a Cab', 'Create a Cab', 'About', and 'Logout' on the right. The main heading 'CREATE A CAB' is in large blue letters. Below it is a light gray form box containing the following fields: 'Name' (a single-line text input), 'To' (a single-line text input), 'From' (a single-line text input), 'Date' (a single-line text input), and 'Description (optional)' (a multi-line text area).

4.2 CASE STUDY

In the event that there is an approach to spare cost, at that point a larger part of the Indian purchasers will attempt to look at it. That has been one of the more vital manners by which ride sharing applications have had the capacity to develop in a nation like India. The money saving advantages to going with outsiders far exceeds any potential issues emerging from trust or burglary. The course is regularly the equivalent, in case you're driving to and from work, and the journey consumes a great deal of petroleum. Presently with the cost of petroleum nearly achieving Rs. 82 for each liter, drivers must be competing for ride-sharing applications in huge numbers. Despite the fact that there are appropriated players in the business, there isn't any one contender that is radiating through the cycle. Not at all like Uber and Ola, ride-sharing applications don't have a similar prevalence or advertising spending plans to contact mass crowds. At this moment, their most solid option is verbal exchange and cost-impetuses.

That is the place BlaBlaCar comes in. With it's ride-sharing network, it has the ability to change the manner in which that Indians consider happy with voyaging. Take for instance an average journey from Mumbai to Pune. The ride will cost you around Rs 350, regardless of whether you travel from BlaBlaCar or by means of a between city transport.

With the BlaBlaCar, you can demand booked stops, have charming organization and be dropped precisely where you have to. There is additionally a GPS-empowered ride following application for expanded security and following. You have full oversight despite the fact that you're riding with a stranger. More extensively, on the off chance that we take a gander at the ride-sharing idea from a taxi-aggregator perspective, the money saving advantages are gigantic. With Uber and Ola offering pool-share rides for a couple of years now, clients are very pleasing inside the horde of a force vehicle. With the additional impetus of cooling and affirmed rides, there is minimal motivator for them to co-opt different strategies. "We have seen a high take-up for the class on both the client just as the driver accomplice side. Urban communities, for example, Delhi, Bengaluru, Mumbai, Hyderabad and Kolkata are the greatest clients of Ola Share," said Vishal Kaul, COO Ola.

An ongoing Uber and BCG report demonstrated that congestion and traffic is going to cost the main four urban communities in India upwards of \$22 billion per year. With more autos being acquainted with level 2 urban communities also, that issue is going to mount after some time. With the accessibility of ride-sharing, the penchant to buy a low to mid-layered vehicle additionally goes down. Similar to the case with space usage, there is likewise a critical pattern of decrease in devoted parking spots as more individuals like to ride-share. For these application organization, there is benefit in development. The more the interest for the applications, the more these vehicles are continually moving near. A dead-stop vehicle or one that hasn't been actuated for a significant lot is a loss to the framework. For Uber and Ola, it's about 30% everything being equal, however the pattern keeps on going upwards demonstrating that more Indians need to attempt these administrations at a moderate shared-rate. "With carpooling, Ola and Uber are getting clients who might not generally utilize their administrations regularly in light of the costs. In this way, the organization are gaining admittance to a lot of clients who are cost-cognizant, don't utilize a taxi regularly and rather utilize a transport or an auto rickshaw. The vast majority of these shoppers are youthful experts. After some time, they will move on from utilizing a mutual vehicle to booking a whole taxi, which makes it a splendid system to locally available new clients, said Sreedhar Prasad, accomplice, web based business and new companies, KPMG India. With everything taken into account, the interest for these ride-sharing applications is expanding and the nation is running on it day by day. With regards to the littler players, there's Ride which is putting forth a social carpooling knowledge. They even work with driving organization to deal with their transportation needs too. For suburbanites, ideas like these bode well as they navigate a similar course in excess of five times each week. "The instalment angle is a coordinated arrangement between the driver and the individual profiting the ride. Be that as it may, at Rs 3/km, it is as shabby as utilizing an open transport, yet offering the comfort of a vehicle," said Nitin Chaddha, Business Development Officer of sRide. There's additionally ToGo, Ridely and a large group of littler players that offer a social riding background. They haven't yet taken scale, the manner in which that conventional aggregators have are as yet endeavoring to locate that next huge jump. This could be a sign with regards to the trust factor for some Indians, who would prefer to become friends with a couple of associates and ride-share independently. Be that as it may, for voyagers going between state and for the individuals who aren't ready to go with others, it's an incredible method to spare expense and ensure the earth.

5. COMPARITIVE ANALYSIS

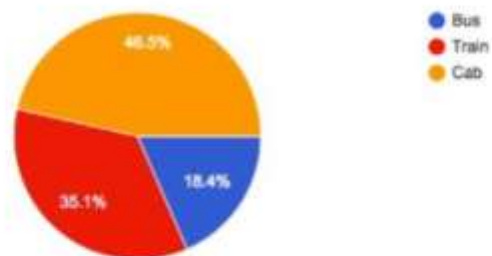
5.1 COMPARISON GRAPH

We got a total of 934 responses from the students of VIT. Below are the detailed results and analysis of it.

- Majority of the people ~ **64%**, had already tried CarPooling, while only ~ 34% had not.

Which mode of transport do you prefer generally to reach your home or to the airport?

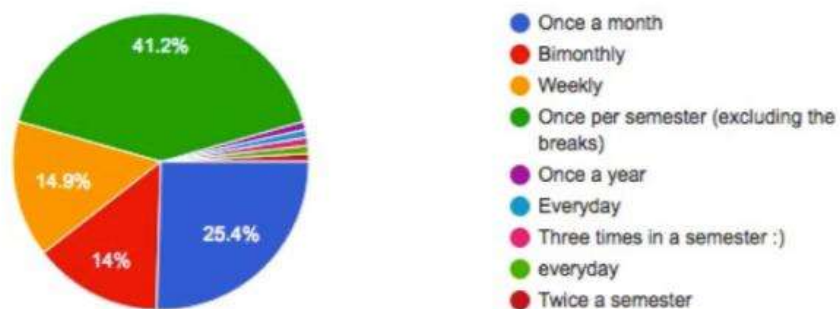
934



- Most of the people prefer Cabs ~ **46.5%** over train ~ **35.1%** and only **18.4%** of the total responses chose Bus as their preferred mode of transport.

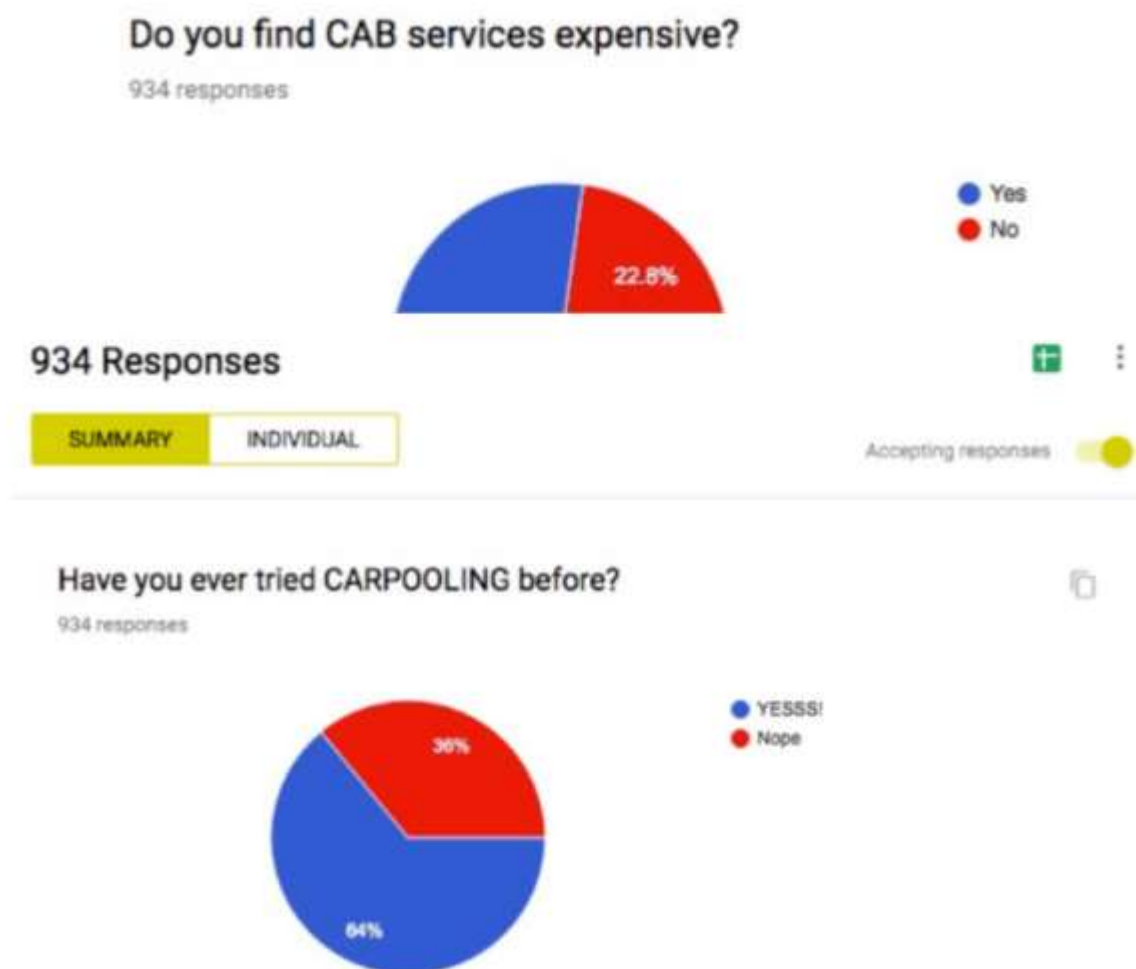
On an average, how often do you travel to/fro from VIT in a semester?

934



- Majority of the people ~ **41.2%** travel home once per semester, followed by ~ **25.4%** travelling home once a month and ~**14%** travelling bimonthly

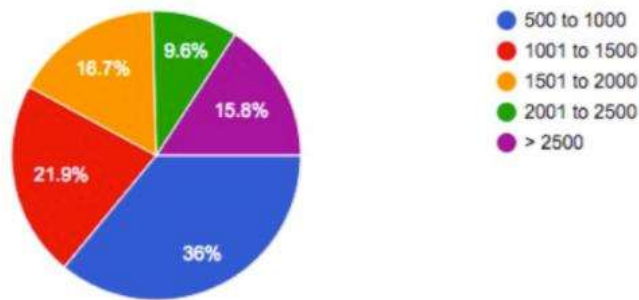
We also noted ~**14.9%** people travel home every week



- A staggering ~**77.2%** of the people find Cab services expensive.

Roughly, how much do you spend on traveling (Commuting to/from VIT)

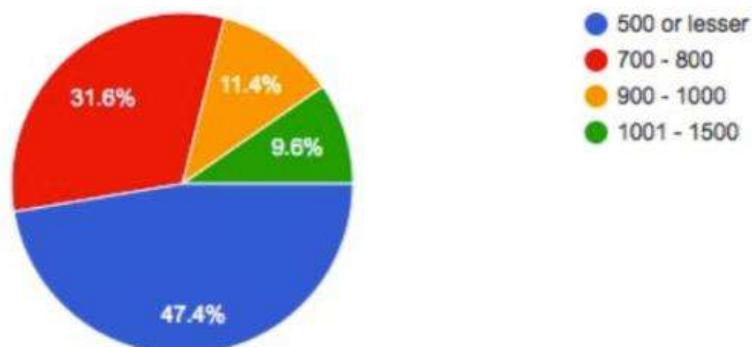
934 responses



Majority ~ **36%** spend between 500 - 1000 on travel.

How much extra are you willing to pay to take a cab instead if bus services or train services?

934 responses



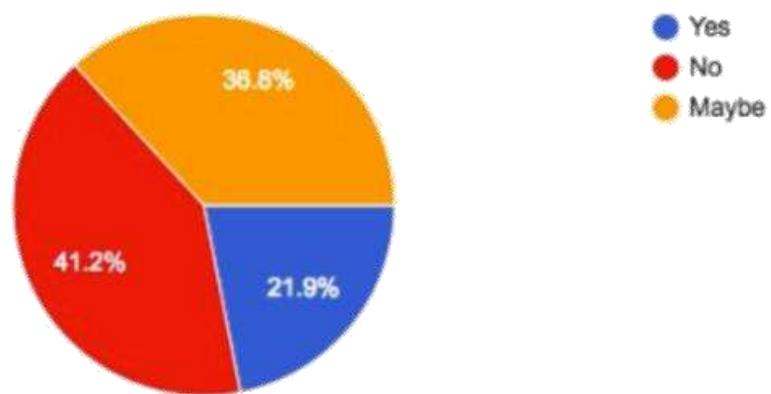
- The average cost spend on commuting to/from VIT is around **Rs 700-1500**.
- Majority of the people ~**47.4%** are willing to pay around **Rs 500** to take a cab instead of a bus/ train to increase the level of comfort, with around ~**9.4%** willing to spend even **Rs 1500**.

•**41.2%** of the responses we recorded had no problem with sharing a cab with a stranger, while **36.8%** were inclined towards it under certain conditions.

•Only ~**21.9%** had a problem sharing their cab with a stranger.

Would you mind sharing your cab with a stranger?

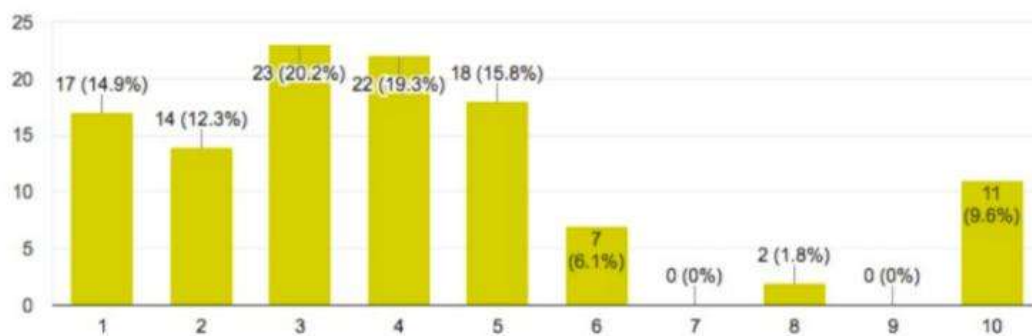
934 responses



•~52.6% of the people have no problem waiting + - 1 hour.

With how many people will you be comfortable in sharing the cab?

934 responses

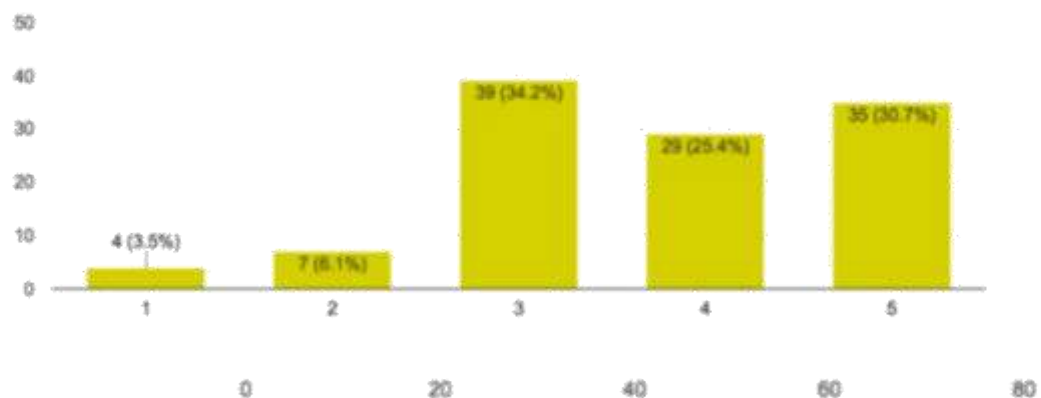


Most of the people are comfortable sharing their cab with 3-4 people.

- Majority of people ~71% have no preference in this matter.
- Majority consider Travel Buddy to be safe.

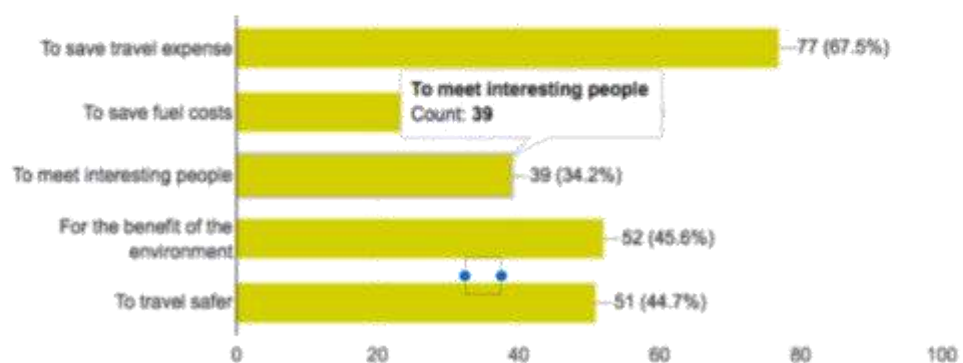
Considering safety, how safe will you consider the idea os 'Travel Buddy' to be?

934 responses



What would be your primary reason for opting for car sharing?

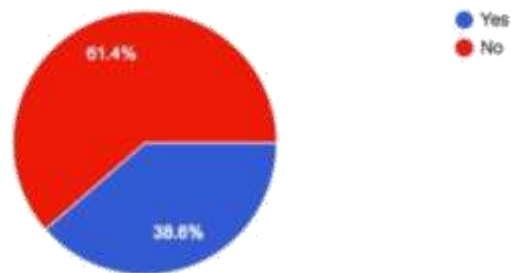
934



- Majority of people ~77% have “Save travel expense” as their reason for carpooling.

Would you mind stopping for a meal layover while traveling?

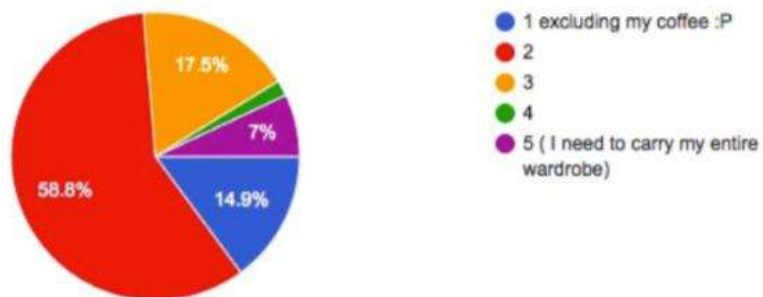
934 responses



- Majority of the people ~**58.8%** had around 2 luggage to carry with them.

In general, what's your luggage count?

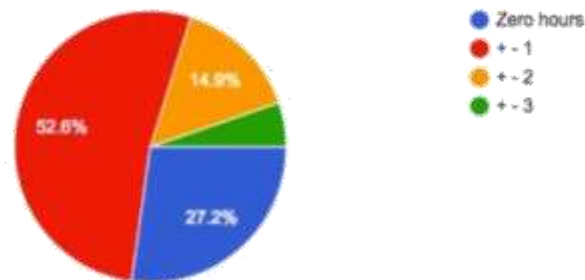
934 responses



- **61.4%** had no problem in stopping for a meal midway their traveling.

How flexible will you be in waiting for your buddy?

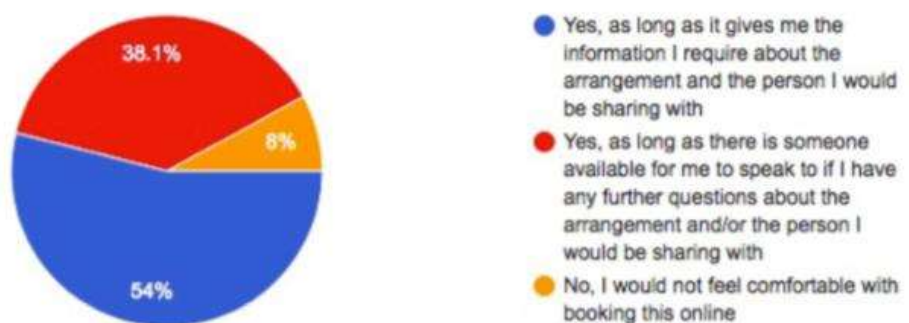
934 responses



- ~ **92.1%** of the people were open to the idea of travel buddy provided necessary information about the driver and their traveling companion be provided to them beforehand.

Would you be willing to set up a Travel Buddy (car sharing arrangement) through a website?

934 responses



5.2 SPAM ALGORITHM ANALYSIS

Algorithm	Accuracy	Comments
Naïve Bayes	96.5	The recall and confidence level is pretty good when it comes to spam detection with respect to chat. Car pooling spam can be done by chat and global location cords. Naïve Bayes seems a good option
Random forest	78	Works very well with location co ordinates but fails with respect to Natural Language Processing term frequency transforms. Good to consider for inclusion in location system
Support Vector Machines	93	Works good in both aspects but is time consuming and more probabilistic. So we choose to work using Naïve Bayes
Deep learning technique	NIL	Takes too much time to execute using keras library in python

6. CONCLUSION

- The software was successfully tested for various units.
- The tests proved that the interface is smooth and there is proper communication among different modules of the system
- The database transaction including updating and deletions work successfully.
- The career posting forum had shown errors which can be improved upon.
- The bugs found have been removed after inspection.
- The software is in conformance to the standards.

FUTURE WORK:

- Implement GSM module
- Fully incorporate the chatbot in our system.

7. REFERENCES

- [1] Aberer, K., Despotovic, Z., 2001. Managing trust in a peer-2-peer information system. In: Proceedings of the Tenth International Conference on Information and Knowledge Management (CIKM'01), pp. 310–317. Advogato, 2017. Advogato. URL [Online; accessed 27-March-2017].
- [2] Alswailim, M. A., Hassanein, H. S., Zulkernine, M., 2016. A reputation system to evaluate participants for participatory sensing. In: 2016 IEEE Global Communications Conference, GLOBECOM 2016 - Proceedings.
- [3] Abrahamse, W., Keall, M., 2012. Effectiveness of a web-based intervention to encourage carpooling to work: A case study of Wellington, New Zealand. *Transport policy*, 21, 45-51.
- [4] Batsouka, C., Tsikas, N., Genitsaris, E., Naniopoulos, A., 2013. Investigation of the travel characteristics of the students making interurban trips from and to their homeplace and the feasibility of implementing a carpooling system: The case of Thessaloniki, Greece. 6th International Congress on Transport Research. Thessaloniki, Greece. [in Greek]
- [5] Correia, G., & Viegas, J. (2011). Carpooling and carpool clubs: Clarifying concepts and assessing value enhancement possibilities through a stated reference web survey in Lisbon, Portugal. *Transportation Research Part A*, 45, 896–913. Field, A. (2003). *Discovering statistics using SPSS for windows*. London: SAGE publication, 105-106.

8. APPENDICES

8.1 Appendix A: Project Timeline

January	SRS Preparation
January	SDS Preparation
February	Design Implementation
March	Testing
Late March	Final Changes and Deployment

8.2 Appendix B: Final Code

(this isn't all the code)

Forms:

```
from django import forms

from .models import User_details, Cab, Passengers


from django import forms

from django.contrib.auth.forms import UserCreationForm
from django.contrib.auth.models import User


class SignUpForm(UserCreationForm):

    email = forms.CharField(max_length=254, required=True,
widget=forms.EmailInput())

    class Meta:

        model = User
```



```
fields = ('username', 'email', 'password1', 'password2')
```

Apps:

```
from django.apps import AppConfig

class CabshareConfig(AppConfig):

    name = 'cabshare'
```

Admin:

```
from django.contrib import admin

from .models import User_details, Cab, Passengers
from django.contrib.auth.models import User

# Register your models here.

class DisplayPassengers(admin.ModelAdmin):
    list_display = ['id', 'user', 'of_cab']

class DisplayCab(admin.ModelAdmin):

    list_display = ['id', 'source', 'destination', 'dep_date_time']

class DisplayUser_details(admin.ModelAdmin):

    list_display = ['id', 'user', 'first_name', 'contact_no']

admin.site.register(User_details, DisplayUser_details)

admin.site.register(Cab, DisplayCab)

admin.site.register(Passengers, DisplayPassengers)
```

Models:

```
from django.db import models

from django.contrib.auth.models import User

# Create your models here.

class User_details(models.Model):

    first_name = models.CharField(max_length=20)
```

```

last_name = models.CharField(max_length=20)

gender = models.CharField(max_length=1) # male = m, female = f, other = o
dob = models.DateField()

contact_no = models.IntegerField()

contact_sharing = models.BooleanField(default=False)

user = models.ForeignKey(User, related_name='user', on_delete = models.CASCADE)

class Cab(models.Model):

    source = models.CharField(max_length=20)

    destination = models.CharField(max_length=20)

    dep_date_time = models.DateTimeField()

    size = models.IntegerField()

    cab_admin = models.ForeignKey(User, related_name='cab_admin', on_delete =
models.CASCADE)

    created_at = models.DateTimeField(auto_now_add=True)

class Passengers(models.Model):

    user = models.ForeignKey(User, related_name='passangers', on_delete =
models.CASCADE)

    of_cab = models.ForeignKey(Cab, related_name='of_cab', on_delete =
models.CASCADE)

    is_cab_admin = models.BooleanField()

    approval_status = models.CharField(max_length=1) # requested = r, approved = a,
declined = d

```

Views:

```

from django.shortcuts import render, get_object_or_404, redirect
from django.contrib.auth import login as auth_login

from django.contrib.auth.decorators import login_required
from django.contrib.auth.models import User

from .models import User_details, Cab, Passengers
from .forms import SignUpForm

# Create your views here

```

```

def view_cabs(request):

    cabs = Cab.objects.all()

    return render(request, 'view_cabs.html', {'cabs': cabs})

@login_required(login_url='/login')

def cab_info(request, pk):

    cab = get_object_or_404(Cab, id=pk)

    passengers = list(Passengers.objects.filter(of_cab=pk))

    if request.method == 'POST':

        username = request.user.username

        user = get_object_or_404(User, username=username) # checked that user
exists

        if (Passengers.objects.filter(user=user, of_cab=cab).count()) > 0:

            return redirect('cab_info', pk=pk)

        else:

            passengers = Passengers.objects.create(

                user = user,

                of_cab = cab,

                is_cab_admin = 0,

                approval_status = 'r',

            )

            return redirect('cab_info', pk=pk)

    try:

        current_user = Passengers.objects.get(user = request.user, of_cab = cab)

    except Passengers.DoesNotExist:

        current_user = None

    if current_user != None:

        if current_user.approval_status == 'a':

```

```

        contact_access=True

    else:

        contact_access=False

    else:

        contact_access=False

    return render(request, 'cab_info.html', {'cab': cab, 'passengers': passengers,
'contact_access': contact_access})

@login_required(login_url='/login')

def contact_details(request, pk):

    current_user = Passengers.objects.get(user = request.user, of_cab = pk)

    if current_user.approval_status == 'a':

        cab = get_object_or_404(Cab, id=pk)

        passengers = list(Passengers.objects.filter(of_cab=pk,
approval_status='a'))

        for passenger in passengers:

            passenger_contact_details = list(User_details.objects.filter(user =
passenger.user))

            return render(request, 'contact_details.html', {'cab': cab, 'passengers':
passengers, 'passenger_contact_details': passenger_contact_details})

    return render(request, 'not_authorized.html')

@login_required(login_url='/login')

def approve_cab(request, pk_cab, pk_user):

    cab = get_object_or_404(Cab, id=pk_cab)

    user = get_object_or_404(User, username=pk_user)

    passenger = Passengers.objects.get(of_cab=pk_cab, user=user)

    #if request.method == 'POST':

    passenger.approval_status = 'a'

```

```

passenger.save()

passengers = list(Passengers.objects.filter(of_cab=pk_cab))

return render(request, 'manage_cab.html', {'cab': cab, 'passengers':
passengers})

@login_required(login_url='/login')

def decline_cab(request, pk_cab, pk_user):

    cab = get_object_or_404(Cab, id=pk_cab)

    user = get_object_or_404(User, username=pk_user)

    passenger = Passengers.objects.get(of_cab=pk_cab, user=user)

    #if request.method == 'POST':

    passenger.approval_status = 'd'

    passenger.save()

    passengers = list(Passengers.objects.filter(of_cab=pk_cab))

    return render(request, 'manage_cab.html', {'cab': cab, 'passengers':
passengers})

def user_info_for_request_cab(request, pk):

    cab = get_object_or_404(Cab, id=pk) # checked that cab exists
    if request.method == 'POST':

        username = request.POST['username']

        user = get_object_or_404(User, username=username) # checked that user
exists

        passengers = Passengers.objects.create(

            user = user,

            of_cab = cab,

            is_cab_admin = 0,

```

```

        approval_status = 'r',

    )

    return redirect('cab_info', pk=pk)

    return render(request, 'user_info_for_request_cab.html')

def user_info_for_user_details(request):

    if request.method == 'POST':

        username = request.POST['username']

        user = get_object_or_404(User, username=username)

        return redirect('user_details', username=user)

    return render(request, 'user_info_for_user_details.html')

def user_info_for_new_cab(request):

    if request.method == 'POST':

        username = request.POST['username']

        user = get_object_or_404(User, username=username)

        return redirect('new_cab', username=user)

    return render(request, 'user_info_for_new_cab.html')

@login_required(login_url='/login')

def user_details(request, username):

    user = get_object_or_404(User, username=username) # TODO: get the currently
logged in user

    if request.method == 'POST':

        first_name = request.POST['first_name']

        last_name = request.POST['last_name']

        gender = request.POST['gender']

        dob = request.POST['dob']

        contact_no = request.POST['contact_no']

```

```

        contact_sharing = request.POST['contact_sharing']

        current_user = User.objects.get(username=username)

        user_details = User_details.objects.create(
            first_name = first_name, last_name =
            last_name,

            gender = gender,

            dob = dob,

            contact_no = contact_no,

            contact_sharing = contact_sharing,

            user = current_user,

        )

        return redirect('view_cabs') # TODO: redirect to the created topic page

    return render(request, 'user_details.html',{'User': user})

@login_required(login_url='/login')

def new_cab(request, username):

    user = get_object_or_404(User, username=username) # TODO: get the currently
logged in user

    if request.method == 'POST':

        destination = request.POST['destination']

        source = request.POST['source']

        dep_date = request.POST['dep_date']

        dep_time = request.POST['dep_time']

        dep_date_time = dep_date + ' ' + dep_time

        size = request.POST['size']

        current_user = User.objects.get(username=username)

        cab = Cab.objects.create(

            source = source,

```

```

        destination = destination,

        dep_date_time = dep_date_time,

        size = size,

        cab_admin = current_user,

    )

    created_cab = Cab.objects.get(

        source = source,

        destination = destination,

        dep_date_time = dep_date_time,

        size = size,

        cab_admin = current_user,

    )

    passengers = Passengers.objects.create(

        user = current_user,

        of_cab = created_cab,

        is_cab_admin = 1,

        approval_status = 'a',

    )

    return redirect('view_cabs') # TODO: redirect to the created cab page

    return render(request, 'new_cab.html', {'User': user})

def user_info_for_admin_panel(request):

    if request.method == 'POST':

        username = request.POST['username']

        user = get_object_or_404(User, username=username)

        return redirect('cab_admin_panel', username=user)

    return render(request, 'user_info_for_admin_panel.html')

```



```

@login_required(login_url='/login')

def cab_admin_panel(request, username):

    user = get_object_or_404(User, username=username)

    cabs = list(Cab.objects.filter(cab_admin=user))

    return render(request, 'cab_admin_panel.html', {'User': user, 'cabs': cabs})


@login_required(login_url='/login')
def manage_cab(request, username, pk):

    cab = get_object_or_404(Cab, id=pk)

    user = get_object_or_404(User, username=username)

    passengers = list(Passengers.objects.filter(of_cab=pk))

    return render(request, 'manage_cab.html', {'cab': cab, 'passengers':
passengers})


def signup(request):

    # TODO: If user is already logged in, redirect him/her to the view_cabs page
    if request.method == 'POST':

        form = SignUpForm(request.POST)
        if form.is_valid():

            user = form.save()

            auth_login(request, user)
            return redirect('view_cabs')

    else:

        form = SignUpForm()

    return render(request, 'signup.html', {'form': form})

```

8.3 Appendix C: Final Screenshots

Travel Buddy

Log in

Sign up

Cabs

New Cab

User Details

Admin Panel

From	To	Date and Time	Cab Admin	Current Passengers
VIT Vellore	Bangalore	Oct. 24, 2018, 6 p.m.	Sakari	<div>View Info</div>
Chennai	VIT Vellore	Nov. 10, 2018, noon	akshay1	<div>View Info</div>
Vellore	Mumbai	Oct. 12, 2018, 4 a.m.	saloneegupta	<div>View Info</div>

Travel Buddy

Log in

Sign up

Log in

Username:

akshay1

Password:

Log in

New to Travel Buddy?

Sign up

Travel Buddy

Log in

Sign up

Sign up

Username:

Required, 100 characters or fewer. Letters, digits and @/./+/-/_ only.

Email:

Password:

Your password can't be too similar to your other personal information.

Your password must contain at least 8 characters.

Your password can't be a commonly used password.

Your password can't be entirely numeric.

Password confirmation:

Enter the same password as before, for verification.

Create an account

Already have an account?

Log in

Travel Buddy

Log out

Saket

Hello Saket !

Please fill the following details.

Source

Destination

Departure date

dd/mm/yyyy

Departure time

Cab Size

3

Post New Cab

Travel Buddy

Log out

Saket

Hello Saket !

Please complete your user profile by entering the following details.

First Name

Last Name

Gender

☐ Male ☐ Female ☐ Other

Date of birth

Contact Number

Share Contact

☐ Yes ☐ No

Update Details

Travel Buddy

Log out

Saket

This is your Admin Panel Saket !

From	To	Date and Time	Manage
VIT Vellore	Bangalore	Oct. 24, 2018, 6 p.m.	<div>Manage Cab</div>