DEVELOPMENT OF CARPOOLING WEBSITE WITH SMS TECHNOLOGY & SHORTEST PATH ALGORITHM

Submitted in partial fulfillment of the requirements

Of the degree of

Bachelor of Engineering

by

SHAIL SHAH	BE-4-68
SMIT SHAH	BE-4-69
AMAY YADAV	BE-4-75

Supervisors:

PROJECT GUIDE
PROF.MRS.JYOTI JOGLEKAR

Department Of Computer Engineering

SHAH & ANCHOR KUTCHHI ENGINEERING COLLEGE

CHEMBUR, MUMBAI-400 088, INDIA

2013-2014

Project Report Approval for B. E

This project report **Carpooling website with Group Matching algorithm & Communication methods** by **Shail Shah, Smit Shah and Amay Yadav** is approved for the degree of **Computer Engineering.**

Project Guide – Mrs. Jyoti Joglekar

External Guide

HOD (Comps) – Mr. Uday Bhave

Principal – Mrs. Uma Rao

Date: 05/05/2014

Place: Chembur

Declaration

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

----------	----------	----------
Shail Shah	Smit Shah	Amay Yadav
BE-4-68	BE-4-69	BE-4-75

Place: Chembur

Date: 05/05/2014

Attendance Record

Fortnight	**Duration**	Shail Shah (BE-4-68)	Smit Shah (BE-4-69)	Amay Yadav (BE-4-75)	**Guide's Sign**
Fortnight 1	01-10-2014 To 15-10-2014	P	P	P	
Fortnight 2	16-10-2014 To 31-10-2014	P	P	P	
Fortnight 3	01-01-2014 To 15-01-2014	P	P	P	
Fortnight 4	16-01-2014 To 31-01-2014	P	P	P	
Fortnight 5	01-02-2014 To 15-02-2014	P	P	P	
Fortnight 6	16-02-2014 To 28-02-2014	P	P	P	
Fortnight 7	01-03-2014 To 15-03-2014	P	P	P	
Fortnight 8	16-03-2014 To 31-03-2014	P	P	P	
Fortnight 9	15-04-2014 To 01-04-2014	P	P	P	

Plan Time & Task Management

Milestone	Task Description	Duration(weeks)	Performed By
Problem statement definition	• Identify needs and benefits • Literature Survey • Establish problem statement	3	• Amay Yadav • Smit Shah • Shail Shah
Designing	• Decide development tools • Design front end • Divide project in three modules	2	• Amay Yadav • Smit Shah • Shail Shah
Developing & testing Module 1	• Development of Algorithms for route • Adding Presentations • Test module	2	• Amay Yadav • Smit Shah • Shail Shah
Developing & Testing Module2: Sub_module1	• Algorithm testing • Auto complete Google maps • Comparison of algorithm on complexity	2	• Amay Yadav • Smit Shah • Shail Shah
Developing & Testing Module3	• Creation of database • Group assignment algorithm • Testing of algorithm with test cases	2	• Amay Yadav • Smit Shah • Shail Shah

Developing & Testing Module3: Sub_module1	• Designing of forms • Integration of maps on forms • Direction panel for maps • Testing of forms	2	• Amay Yadav • Smit Shah • Shail Shah
Developing & Testing Module3: Sub_module2	• Creation and testing of interfaces(i.e. member of group, view all rides, posted rides, join ride, remove)	1	• Amay Yadav • Smit Shah • Shail Shah
Developing & Testing Module3: Sub_module3	• Internal messaging system • SMS technology • Email delivery for verification of details	1	• Amay Yadav • Smit Shah • Shail Shah
Testing	• Generate Test Cases • Test each module • Test entire Project • Removal of bugs and errors	1	• Amay Yadav • Smit Shah • Shail Shah

Timeline and Gantt Chart

	Task Name	Duration	Start	Finish	Predecessors
1	Problem statement definition	14 days	Mon 07/10/13	Thu 24/10/13	
2	Identify needs and benefits	5 days	Mon 07/10/13	Fri 11/10/13	
3	Literature Survey	5 days?	Mon 14/10/13	Fri 18/10/13	2
4	Establish problem statement	5 days	Fri 18/10/13	Thu 24/10/13	
5	Designing	12 days	Fri 25/10/13	Fri 08/11/13	4
6	Decide development tools	4 days	Fri 25/10/13	Wed 30/10/13	1
7	Design front end	4 days	Tue 29/10/13	Fri 01/11/13	
8	Divide project in three modules	3 days	Sat 02/11/13	Tue 05/11/13	7
9	Developing & testing Module1	12 days	Tue 05/11/13	Sun 17/11/13	
10	Development of Algorithms for route	4 days	Wed 06/11/13	Sat 09/11/13	8
11	Adding Presentations	4 days	Sun 10/11/13	Wed 13/11/13	10
12	Test module	3 days	Thu 14/11/13	Sun 17/11/13	11
13	Developing & Testing Module2: Sub_module 1	12 days	Fri 15/11/13	Fri 29/11/13	
14	Algorithm testing	4 days	Mon 18/11/13	Thu 21/11/13	12
15	Auto complete Google maps	4 days	Fri 22/11/13	Wed 27/11/13	
16	Comparison of algorithm on complexity	4 days	Tue 26/11/13	Fri 29/11/13	
17	Developing & Testing Module3	12 days	Mon 23/12/13	Tue 07/01/14	
18	Creation of database	4 days	Mon 23/12/13	Thu 26/12/13	
19	Group assignment algorithm	4 days	Fri 27/12/13	Wed 01/01/14	18
20	Testing of algorithm with test cases	4 days	Tue 31/12/13	Fri 03/01/14	
21	Developing & Testing Module 3: Sub_module 1	12 days	Mon 06/01/14	Tue 21/01/14	
22	Designing of forms	6 days	Tue 07/01/14	Tue 14/01/14	
23	Integration of maps on forms	2 days	Tue 07/01/14	Wed 08/01/14	
24	Direction panel for maps	3 days	Thu 09/01/14	Mon 13/01/14	23
25	Testing of forms	3 days	Mon 13/01/14	Wed 15/01/14	
26	Developing & Testing Module 3: Sub_module 2	7 days	Mon 27/01/14	Tue 04/02/14	
27	Creation and testing of interfaces	7 days	Mon 27/01/14	Tue 04/02/14	
28	Developing & Testing Module 3: Sub_module 3	7 days	Thu 06/02/14	Fri 14/02/14	
29	Internal messaging system	3 days	Wed 05/02/14	Sat 08/02/14	
30	SMS technology	2 days	Mon 10/02/14	Tue 11/02/14	
31	Email delivery for verification of details	2 days	Tue 11/02/14	Wed 12/02/14	
32	Testing	7 days	Fri 14/02/14	Mon 24/02/14	
33	Generate Test Cases	2 days	Mon 17/02/14	Tue 18/02/14	
34	Test each module	15 days	Mon 24/02/14	Fri 14/03/14	
35	Test entire Project	17 days	Thu 27/02/14	Fri 21/03/14	
36	Removal of bugs and errors	3 days	Fri 21/03/14	Tue 25/03/14	

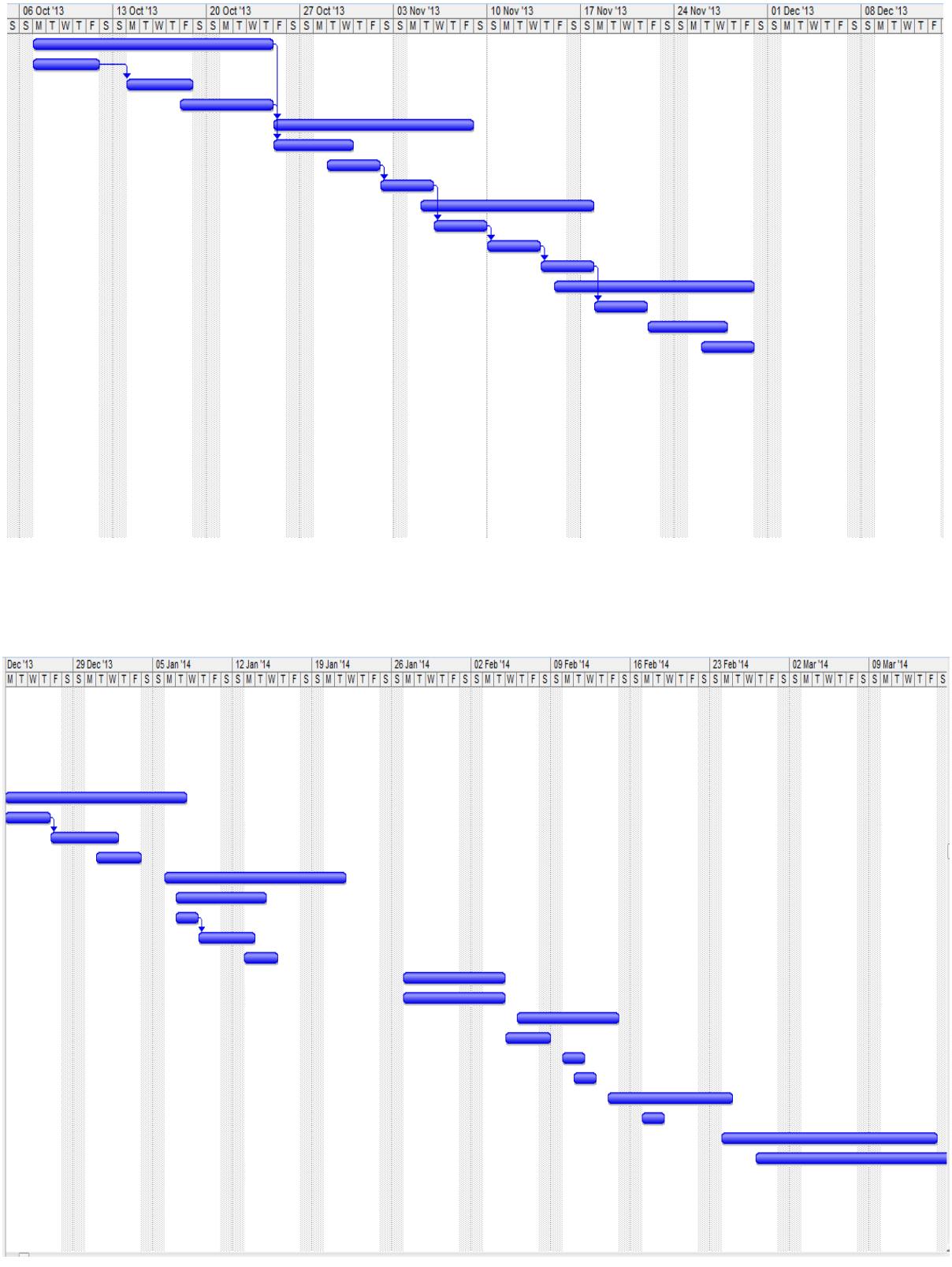


Fig Gantt chart of all modules

Abstract

Carpooling (also known as car-sharing or ride-sharing), is the sharing of car journey so that more than one person can travel in a car. By having more than one person in one vehicle, reduces their travel costs such as fuel & toll charges and it also reduces the stress of driving. Carpooling is also seen as environment friendly and sustainable mode of travel as sharing journey reduces carbon emissions, traffic congestion on the roads and also reduces the need for parking spaces. Authorities often encourage carpooling, especially during high pollution periods and shortage of fuel.

The most unique part of this topic is inclusion of two algorithms i.e. Dijkstra's and A* through which the routes are compared and the best efficient and shortest route is computed. Efficiency of the routes may vary by these algorithms but our purpose to find the shortest route is well served by these algorithms.

Our application provides a facility to see the available carpool groups as per user's outlook and schedule of journey. This helps the new user who intend to travel on a particular route and get assigned to a particular group. Such informed decisions will not only lessen the burden of daily commute alone but also reduces the cost as it gets shared among the members.

Carpooling is needed for cities like Mumbai and also other densely populated areas in the world. We intend to design a website to make the entire process of carpooling easier for people. Nowadays most people prefer private vehicle to travel due to delay caused in public transport system and luxuries provided by private vehicles. Pre-registration ensures that only identified people can get into the vehicle so that trust can be established.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1. Introduction

Carpooling is an activity that enables its pursuers to abandon their individual means of transport, to share a single vehicle for traveling. This affects the pollution and fuel consumption levels by ensuring a reduction in them and in the process relieves of the road from vehicular congestion. To resolve such issues Carpooling or Car sharing solves the problem by targeting the empty seats in the private cars. Workers or clients functioning in the same area can make use of such carpooling facility. On long journeys, it is common for passengers to only join for parts of the journey, and give a contribution based on the distance that they travel. This gives carpooling extra flexibility, and enables more people to share journeys and save money. As the users are unacquainted about other users who might aim to travel the same journey in particular time intervals we have developed an application that provides a facility to outlook the schedules and journey details about the carpool members which helps the other users who intends to travel on the same route can get assigned to one group and travel accordingly. [2]

Best efficient and shortest route directions will be provided which will lessen the time perspective, irrespective of traffic constraints. Our application is an attempt to make a system that is not only user-friendly but also has a security constraint that we have enforced so the users can be verified as authentic law abiding citizens by uploading legal proof for safety precautions. Assignment of members in a particular group is not only done by matching their start node and end node of the route but also through Artificial Intelligence (AI) and mathematical formulations which also allows the users who reside in close proximity of the route taken between these start node and end node, are privileged to be the part of same

carpool group. Use of such AI to create carpool groups and extrapolate pre-existing routes to fit in the new passengers in our application aims to offer a new approach towards pre-existing carpooling systems. [3]

Furthermore, carpooling has documented social and environmental benefits in reducing traffic congestion as number of vehicles on the road can be reduced significantly. As the system aims at the empty seats it increases vehicle occupancy. Pre-registration ensures that only identified people can get into the vehicle so that trust can be established.

1.1 Objective

Our main objective is to build a website and to implement different algorithms for route selection (Google Maps).Major algorithms include dijkstra's algorithm and A* algorithm to provide multiple paths for the same route and select the best shortest route by comparing these algorithms depending upon all these factors time, cost, distance and space complexity will be varied and approximately calculated.

For security purpose we will recommend the users to attach some identification and address proof during their registration process so as to avoid any violation.

Our website can allow a user to login and plug-in the details of their journey like their source, destination and time. The user can even search for a ride or he can post a ride. All this data is fed into a database and the algorithm calculates the suitable carpool group for that person and also the alternatives that he/she could take. This part of the program requires a domain and a database server run by apache tomcat. The group allocation is done on the basis on artificial intelligence and mathematical formulations. Using artificial intelligence the users in the closest proximity can be assigned to particular carpool group. The algorithm that runs will also calculate cost and compare the different alternatives, the algorithm will not simply find the best possible path but it will also try the random heuristic approach to alter the pre-existent routes to find a better one.

The website may not be available to all those who are devoid of internet connection (e.g. The people outside); So, in addition to that we also intend to build a carpool system in which people can easily send messages to communicate with each other so that the user remain updated about the particular ride or journey.

1.2 Methodology Used

1.2.1 Conceptual Idea

We started off with a simple idea to create a simple web based application to provide carpooling services worldwide. It was simple in the start but as we kept on extrapolating it we realized that it could never be something different. So we decided to make an AI that would create carpool groups and extrapolate pre-existing routes to fit in the new passengers. The web-based application would be as ordinary as any other application .The passenger would sign up and after logging in he/she would plug in his/her details. The mandatory details for carpooling would be their name, contact info, email id and most importantly the source, destination and time of travel. [3]

All this data would then be stored in a database. Now begins the important part; this is what the application is all about. The system follows the concept that many people would travel to many different places and to connect them they would first need to find each other. This is what the program actually does find these people and connect them. These people are then collectively called a carpool group. The number of people in a group would be less than or equal to the number of seats in the car. The constraints are the source, destination and stops to pick up or drop passengers along the way and the time.

1.2.2 Implementation Strategy

The constraints followed are that initial there would be no carpool group so when the first user logs in and plugs in his details then he is assigned a new carpool group. This group is represented in the database in a very explicit manner; it would have the info of all the members of that carpool group and the specific roads that are in that route. Every second a new member joins and he/she has his own unique route which needs to coincide with some other carpool group route to place him/her in that carpool group. This is the unique part of this AI based web based application the route maybe sometime need to be modified or even extrapolated to fit the multiple route descriptions. This is what is so unique about this program. Even if sometime a passenger's source or destination is approximately 200 meters from any road on the path then that route needs to be modified to this new passenger's route.

The problem is not that this process is time consuming but it would require loads of data to sift through. Hence, what we plan to create an entire database that would be designed in a tree structure and store the most frequent coordinates and based on approximate calculation the algorithm would figure out which carpool group will be best suited for that passenger. All these groups need to be pre-planned by the users themselves. The possibly suitable carpool groups are shown to the passenger after the algorithm crunches through the database. The confirmation is then sent via email and SMS. [4] The algorithm also knows the distance of the route and based on current fuel prices and average mileage of cars it can even compute the cost of travel and make it easier for the users to figure out of the rough estimate. This can also be further developed into a social platform.

1.3 Database of Maps for Route Calculation

The database used for roadmaps is not ours and it has to be taken from the biggest database of maps i.e. Google Maps. The data base of Google Maps returns all the required information that could be required to calculate the best optimal path in the program. The returned values are highly specific and very accurate.

The exact data that the Google maps actually provide are all the roads in that specific region of travel and to be more particular it gives us:

- The name of all the roads in that region.
- The starting and ending coordinates.
- The distance of each road individually.
- The time that would be taken to travel on each road individually.
- Intersection of those roads are also provided implicitly.

The database of Google maps is freely available on the internet and it is provided in an XML (extended Markup Language) format or JSON (JavaScript Object Notation) format. We have chosen the JSON format because of its simplicity of parenthesis matching. This file is returned to us in a very specific format and it needs to be parsed and stored for route computation. [2]

An extraordinary format like this requires some different type of parsing to only get the relevant data and the data type use is a hashmap that keeps track of every road via index key. Surprisingly this hash map data type is quite like the map data type and the data stored in it is in such a format that one road is linked to all the other roads that are intersecting it. Thus, it becomes easier during traversal. We have made a separate user-defined data type to store the road information in which includes is built in a hierarchical level beginning from a single coordinate to a road and from that to an intersection and then a complete block.[2]

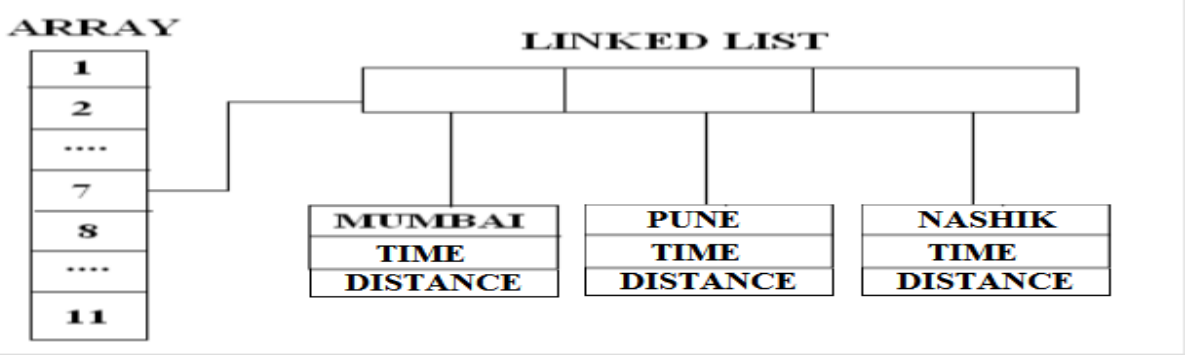


Fig 1.1 Pictorial representation of distance and time (hashmap) A sample JSON file:

"distance" : {
"text" : "0.3 km",
"value" : 278
},
"duration" : {
"text" : "1 min",
"value" : 28
},
"end_location" : {
"lat" : 19.1161638,
"lng" : 72.87042959999999
},
"html_instructions" : "Head \u003cb\u003enorth\u003c/b\u003e on Krishna Mandir Rd\u003c/b\u003e toward \u003cb\u003e5th Rd\u003c/b\u003e",
"polyline" : { "points" :
"start_location" : {
"lat" : 19.1138065,
"lng" : 72.87115059999999},
"travel_mode" : "DRIVING"},

Chapter 2

Literature Survey

2 .1 Literature Survey

There are several definitions of carpooling. Carpooling is collaborative use of car by at least two people for a certain ride determined by an aforementioned condition, with the primary purpose of gaining profit. A motor vehicle has a variable capacity ranging from 5 to 9 seats including the driver and the travel cost may be shared among the carpoolers. The most rudimentary concept of carpooling was introduced by co-workers who lived in the same locality and needed to travel every day between their home and office. So, they all decide to travel in the same car and split the cost of transportation equally amongst themselves. [2]

2.1.1 Overview

We have picked this topic because we believe that lesser traffic in our city and most likely all the other cities in the world is the need of the hour. It has a general appeal to it because it would combine software to people's daily lives. This topic is what we thought of as a real-world application and not only its ornate concept is so illuminating but also the basic notion to help society itself. This project may only look like a final year project but to us it is much more unique and original than that. A carpooling website only brings the possible matches that a user would like but our website has its own unique AI to modify routes so that it can compensate more passengers in the short path and still be efficient. The most basic reason why we wanted this project is because it will run in real-time without anyone monitoring it and it will connect people to other people who travel the same route and it will use database of people and the routes from Google maps itself to do carry out that task.[4]

2.1.2 Working

Passengers login to the web based application and enter their details; out of which most prevalent are their name, contact no, email id, source, destination and time. This information is stored in the database and then the algorithm runs. The algorithm is strictly uses artificial intelligence and does not require any human input. The algorithm finds all the carpool groups whose route coincides with the source and destination with some approximation and extrapolation of the route and has an empty seat as well.

In case the algorithm is unable to find a carpool group that coincides with the passenger's route or has an empty seat then the algorithm creates a new carpool group for future passengers who may travel on that route. Passengers then meet at the predefined locations and time; as the routes are already set, and the passengers expect to be picked up in a relatively short time. Carpooling service is provided to the customers or the passengers and it is very cost efficient and time efficient.

2.1.3 Security

The website and the algorithm may not be completely as efficient as people but that is negligible when the automation of the entire process is considered. The user has to upload his documents for verification or identification so that security can be maintained within the system and the people also need to notify about their changing schedules. The algorithm cannot determine traffic and signals so, even the time constraint may not be satisfied.

2.1.4 Algorithms for Route Calculation

Algorithms used in this project need to be modified because the constraints differ according to the different cases provided to them. For instance the two algorithms used are Dijkstra's Algorithm and A* algorithm used are for finding shortest paths. [2]

2.1.4.1 Dijkstra's Algorithm

The program picks up the Google map database of all the possible routes between two coordinates or locations and parses it to find the precise data that will be required. The program needs to have a specific input which is the names or coordinates of the source and destination and all the pickup and drop points along the way where different passengers get in and get out. The data on the roads consists of start and end coordinates, the distance, the time taken to cover that distance and the name of the road. The program then stores this data into a hash map data structure. The Dijkstra's algorithm works on the logic that after each new road is added it sifts through all the left out possible choices to find the best replacement to this newly added road and if no better alternative is then this one is the shortest path. [2]

Hence the pseudo code Algorithm is as given for the A* below [7]

```
Function Dijkstra(Graph, source):
        for each vertex v in Graph:                 // Initializations
        dist[v]  := infinity ;              // Unknown distance function from  source to v
        previous[v]  := undefined ;     // Previous node in optimal path
        end for                                     // from source
        dist[source]  := 0 ;                    // Distance from source to source
        Q := the set of all nodes in Graph ;
while Q is not empty:                           // The main loop
        u := vertex in Q with smallest distance in dist[]
        if dist[u] = infinity:
                break ;                             // all remaining vertices are
        end if                                  // inaccessible from source
        for each neighbor v of u:         // where v has not yet been removed from Q.
                alt := dist[u] + dist_between(u, v) ;
                if alt < dist[v]:                       // Relax (u,v,a)
                        dist[v]  := alt ;
                        previous[v]  := u ;
                        decrease-key v in Q;      // Reorder v in the Queue
        end if
end for
end while
return dist;
end function
```

2.1.4.2 A* Algorithm

The alternate algorithm used is the A* algorithm. It works in the same manner as dijkstra's algorithm but it doesn't require the distances all it really needs are the start and end coordinates. This algorithm then follows the greedy heuristic pattern on the coordinates to find the shortest route. The heuristic cost is calculated using the Pythagorean Theorem (also called Euclidian distance). This heuristic value is not the accurate distance but it is good enough. Based on the heuristic value the shortest path is computed which includes the start, the drop and pick up points and the end. [2]

If the heuristic function *h* is admissible, meaning that it never overestimates the actual minimal cost of reaching the goal, then A* is itself admissible if we do not use a closed set. If a closed set is used, then *h* must also be monotonic for A* to be optimal. This means that for any pair of adjacent nodes *x* and *y*, where *d(x, y)* denotes the length of the edge between them, we must have:

$$h(x) <= d(x,y) + h(y)$$

The time complexity of A* depends on the heuristic. In the worst case, the number of nodes expanded is exponential in the length of the solution (the shortest path), but it is polynomial when the search space is a tree, there is a single goal state, and the heuristic function *h* meets the following condition:

$$|h(x) - h^*(x)| = O(\log h^*(x))$$

Hence the pseudo code Algorithm is as given for the A* below [7]

```
function A*(start, goal)
closedset := the empty set    // The set of nodes already evaluated.
openset := {start}    // The set of tentative nodes to be evaluated, initially containing the start node
came_from := the empty map    // The map of navigated nodes.

g_score[start] := 0    // Cost from start along best known path.
   // Estimated total cost from start to goal through y.
f_score[start] := g_score[start] + heuristic_cost_estimate(start, goal)

while openset is not empty
        current := the node in openset having the lowest f_score[] value
        if current = goal
        return reconstruct_path(came_from, goal)
        remove current from openset
        add current to closedset
        for each neighbor in neighbor_nodes(current)
                tentative_g_score := g_score[current] + dist_between(current,neighbor)
        tentative_f_score := tentative_g_score + heuristic_cost_estimate(neighbor, goal)
        if neighbor in closedset and tentative_f_score>= f_score[neighbor]
                continue

                if neighbor not in openset or tentative_f_score<f_score[neighbor]
        came_from[neighbor] := current
                g_score[neighbor] := tentative_g_score
                f_score[neighbor] := tentative_f_score
                if neighbor not in openset
                add neighbor to openset
        return failure
function reconstruct_path(came_from, current_node)
        if current_node in came_from
                p := reconstruct_path(came_from, came_from[current_node])
                return (p + current_node)
        else
                return current_node
```

2.2 Comparison of Algorithms

	Dijkstra's Algorithm	A* Algorithm
Method used	Using unsorted array as priority queue.	Using a Min-heap as priority queue.
Data Structure	Graph with |V| vertices and |E| edges.	Graph with |V| vertices and |E| edges.
Time Complexity	O ($\|V^2\|$)	O(|V|+|E|) log |V|)
Space Complexity	O (|V|)	O (|V|)
Accuracy	More accurate (i.e. best accurate route with shortest path).	Less accurate (i.e. moderate and varies as per threshold value).
Speed	Computing time is slow.	Computing time is fast.
Distance	Similar to A* algorithm.	Similar to Dijkstra's algorithm.
Conclusion	Thus it provides more accuracy in terms of routes but computing time of algorithm is slow as it aims to get the shortest and efficient route.	Thus it provides less accuracy as threshold value (heuristics) is dependent but computing time of algorithm is faster than dijkstra's and provides more alternatives.

Table 2.1 Comparison of A* and Dijkstra algorithm

2.3 Google Maps

The program runs predominantly on the data provided by Google maps. Now these maps have a lot of roads and there is absolutely no telling which way to take for anyone. Now, these maps are extensive and in its true definition the epitome of GIS (Geographical Information Systems). The Google maps are extensively shaped to hold data which is specific to every point on the earth. The extensive detailing provided by these maps can only be judged by the simple fact that it takes up to 15 digits after decimal point in the latitude and longitude coordinate system. [4]

Our program only requires the roadmaps from the Google maps database. Now Google provides an efficient way to calculate routes. It is so dynamic that if the start is kept on any point on the earth and the stop is kept a thousand miles away even then Google algorithm will provide us a route along with the distance and stretched outline from start to stop on the map from the start to the end.

2.3.1 Our Use of Google Maps

Even though Google maps provides extraneous amounts of data which when coupled with its algorithms can give us all that we require for our project; we have not used its API for our program because of various reasons. Google maps give the best possible route in terms of time, distance and even traffic but we have implemented our own algorithms for route calculation. The algorithms implemented by us give us more control in group matching and re-routing of the overall paths taken by various groups.

As it has been explained we take all the roads that come in between the start and stop. The manner of taking these roads is via a JSON file provided by Google maps. This JSON file contains the name of all the roads, distances even time, start and stop coordinates. These JSON files are used predominantly for route calculation and rerouting. Google Maps also provides us with the auto complete feature which is very easy to implement in a textbox used by the client to find any location by name on the earth. The user can either post a ride (i.e. start a new ride) or join a ride (i.e. become a member of an existing one). In both these case the user is presented with two maps to give their start and stop on the map. Since it needs to be user friendly the user is shown the name of the place in the textbox with the auto complete

feature and also a marker on the map. Whereas we get the coordinates of those start and stop. With these two coordinates we are able to formulate a path between the start and the stop. All this route calculation is done via our algorithms. All the other details of the ride are straightforward store and retrieval from the database.

The most rudimentary and essential parameter are the coordinates. The present day idea and also implementation of carpooling is that there is either a common source for various stops for different members or there is a common destination for various pick up points; but our project implements a style in which there are multiple stops and multiple pick up points. This is a little difficult to implement since all the points are required and even after that the route needs to be calculated so that it covers every coordinate that a user needs to be picked up or dropped.

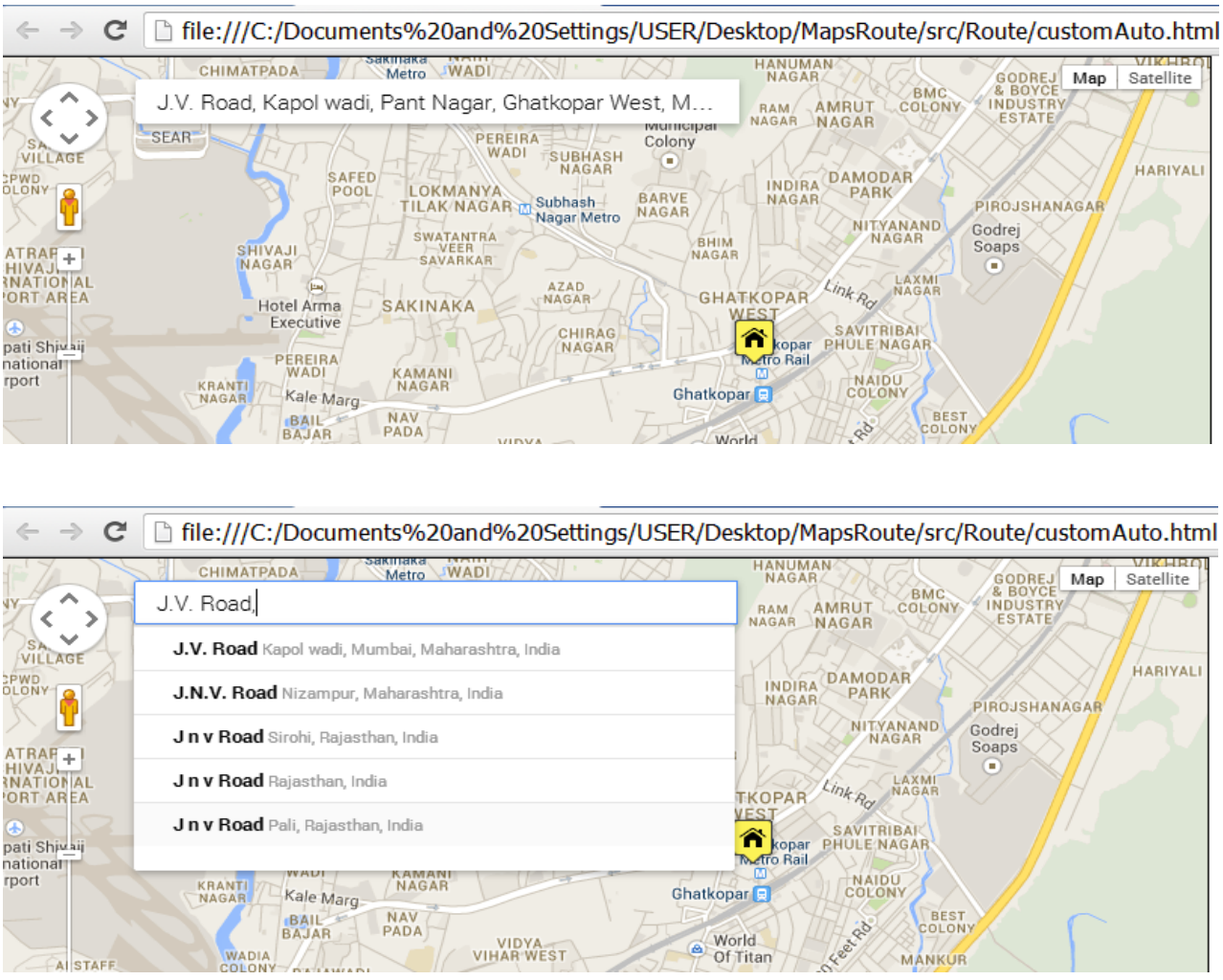


Fig 2.1 Google Maps with Auto Complete field

2.4 Method of Communication

Since carpooling is an everyday event the communication between the members of the group i.e. the driver and the passengers needs to absolutely impeccable. Simple communication like "I am not coming tomorrow because I am sick" messages need to be passed from the message sender to other people in that particular group. Also messages need to pass from the system to the user in a very dynamic manner. The users need to be notified instantly of any change in the ride timings and pick up and drop points. [5]

The users the drivers need to receive these messages immediately. They need to receive these messages even when they are not logged in. Any user of the website receives a mail at any instance there is a change in their status. At registration a mail is dispatched to them and at various other occasions. Also SMS is sent to users when it is entirely necessary to notify them that very instant. In addition to all this the website also has an internal messaging system in which a user can send a message to all the current members of a group that they are also a part of.

2.4.1 E-Mail

E-Mail is the oldest and most reliable way of communication over the internet. E-mail existed even before the internet itself. Also if the user can make an account on our carpooling website then he/she already has an E-Mail ID and it can also be assumed that they have a steady internet connection. E-Mail is the most basic and very ubiquitous way of communication in our website. An E-Mail is dispatched at every instance of change in a user's profile to the ID at which the user is registered. Also, we have used the Email-ID of any user as their primary login key.

Now when we say change in user's profile that would include circumstances like new user is sent a registration E-Mail or when a member posts a new ride then an E-Mail would be sent to that user about the ride details. If a member joins a carpool group even then an E-Mail is sent to that member and in addition to him an E-Mail is also sent to the driver of that group. An E-Mail is also dispatched with the password when a user forgets his/her password. Anyone who is not a member of our website can also contact us using our contact us page.

We have used the Java Mail API coupled up with SQL to send these Mails and usually an E-Mail is sent at every instance that the database is updated or edited to the relevant users. The Mail service can send email messages to one or more recipients. The message contains a subject, a plaintext body, and an optional HTML body. It can also contain file attachments, as well as a limited set of headers. To send an e-mail using your Java Application is simple enough but to start with you should have Java Mail API and Java Activation Framework (JAF) installed on your machine. After the installation of this API, it is a simple enough task to initiate a SMTP session on any of the E-Mail service providers. As soon as a session is initiated a mail can composed in various multi-body parts and can be sent via the mail API through the session created to any Email-ID in the world. The Mail API has been used extensively on our website so that we can communicate with the customer like any normal human-being would communicate with someone else.

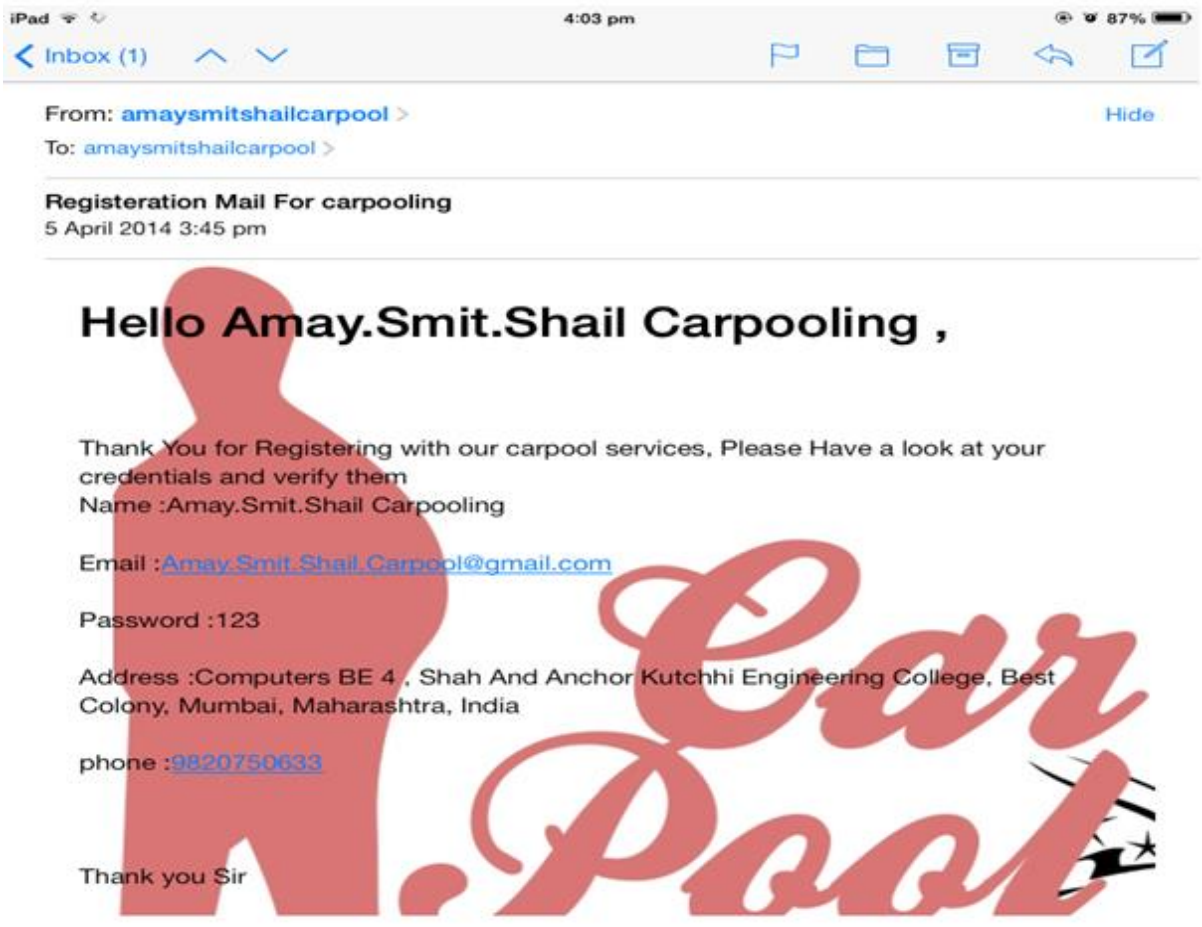


Fig 2.2 Email Verification for Carpool Registration

2.4.2 SMS

Some notifications are very important and need to be seen immediately by the concerned party. So we cannot use E-Mail in this case because no one sits with their E-Mail open all day; that's why we have used the SMS feature for the most absolutely important notifications. Sending an SMS proved to be difficult and all the messaging services are expensive. So we had to find a way to make it free and also legitimate. So, we had to try and make something new. I decided to do what I do best i.e. hack into a pre-existing system like Way2Sms.com and have a program run as a bot (scripting code that makes a 3rd party app feel like an actual human being is running it and not the code). [5]

We had to use this SMS feature sparingly as it is not exactly legitimate (i.e. strictly speaking not legal). A SMS is sent when it is absolutely necessary for a user to be notified immediately. For example when a user is registered and their details like phone number and E-Mail need to be verified that very instant. When a member joins or gets out of a pre-existing carpool group even then the driver needs to be exigently notified of this change so, that no confusion is caused the next day when someone shows up or doesn't show up at their specified location respectively.

When a user forgets their password even then they need to be sent their password on the phone that they have registered during sign up. The messaging feature uses the Selenium Web driver and the default browser taken for that implementation is Firefox which sometimes shows up and the entire process can be seen.

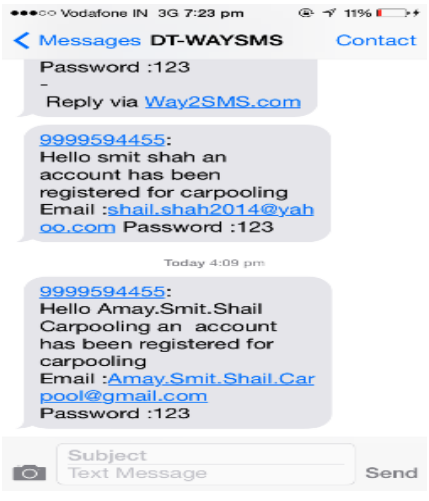


Fig 2.3 SMS notification for Registration

2.4.3 Messaging

The entire Messaging feature is an in-built component of our website. We realized E-Mail and SMS wouldn't suffice so we added this extra feature to our website. The messaging app in our website runs as a separate module and is completely loaded with unread notification and it even shows the user what is left to be read by them. The messaging app is dynamic and asynchronous in nature; hence it needs to be active in all the pages after login. The messaging app can only send messages to the people who are in the same group i.e. the message sender should be a member of the carpool group that he/she is sending the message to.

To accomplish user abstraction and group communication i.e. one too many, we have made a simple enough page that displays all the rides that that member is a part of which would include both posted and group member rides. The user is given option to send his/her message to select groups only. If a user sends a message to a particular group then that message is not sent to every group member individually instead it is store in the group database using the Text data type in SQL along with the timestamp and sender's name and a notification variable is also set for every individual member to indicate whether they have read that message or not. The notification of unread messages gets displayed to every member in that particular group.

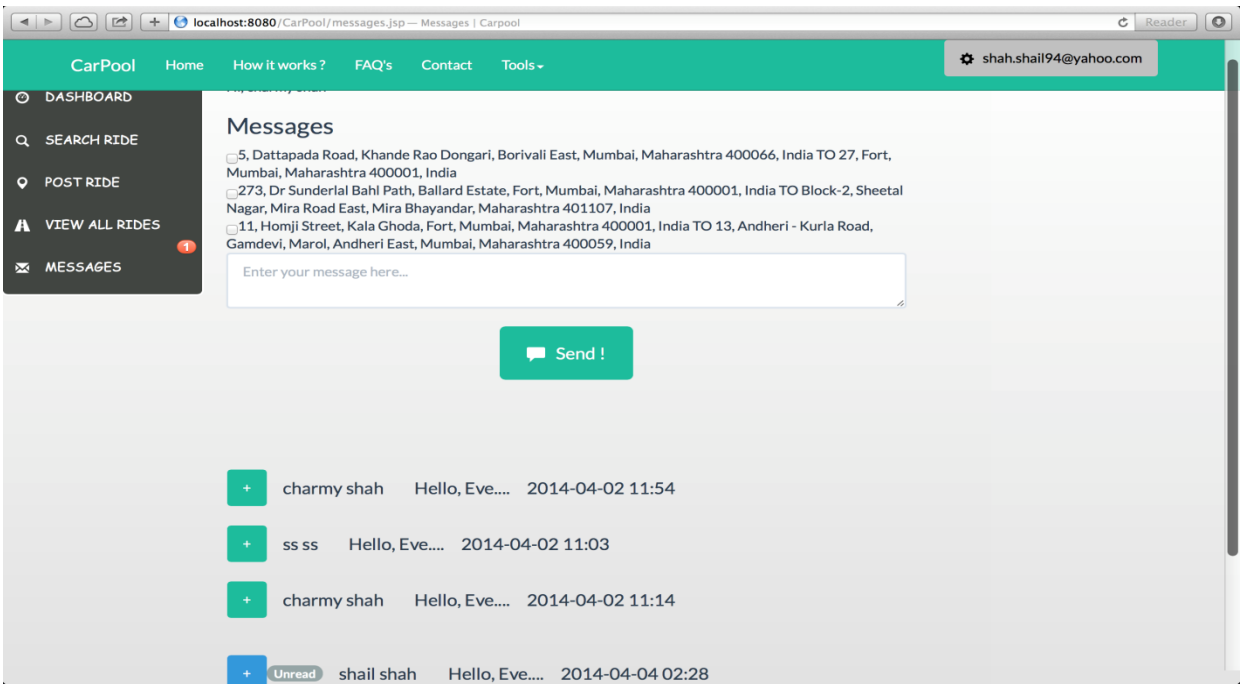


Fig 2.4 Internal Messaging System

2.5 Working Method

All the basic data is taken from the client at registration time. After being registered anybody can *Post Ride* or *Search Ride*. The initiation of carpool groups is done by the *Post Ride* option. The client is asked to feed in all the information about the Carpool Group he/she is forming. Also it is mandatory for the person posting the ride to be the driver. The details are as follows:-

- Name Car.
- Number of seats in car (including driver seat).
- Type of fuel used to run the car.
- Days or Dates & Time of travel.
- The Source and Destination of the Route.

All these details are fed into the database along with secondary information like city/state/country and center of the route that is all computed by the algorithm. These are all the Carpool groups in there adolescence.

Now if someone wants to search a carpool group that they are compatible with then they can use the *Search Ride*. This is where the plot gets more contorted. We need to take in the following details from the client.

- Days or Dates & Time of travel.
- The Source and Destination of the Route.

After taking in these details the predefined routes that are in close proximity to the route entered are all picked up from the database. This is done by using the country, state and city column in the database. Once all these routes are taken into account. We use an algorithm that finds the best possible match for the client. This is done by first using the centers of all these routes and seeing if the clients source and destination lie within the routes radius (i.e. is a circle). Then once it's been established that the clients does lie within the extent of the predefined carpool group; we use the same center to draw an ellipse using the previous diameter of the circle as the major axis and the major axis start point is the carpool groups route's source and the end point is groups route's destination.

2.5.1 Ellipse Algorithm (Group Matching Algorithm)

After going through various papers and ideas we realized that the best way to do anything is to come up with something that's original and is ours in its truest form. Hence we decided to approach the group matching algorithm in a geometrical sense so as to re-route the existent route and formulate a new one for every new member in the group. First we decided to use a rectangular approach such that the line joining the start and end would be one of the diagonal of the rectangle and all the points lying in that rectangle would be accepted in the route. But this approach was soon discarded because then route could end being too widespread and might become an inconvenience for the members along the path as opposed to a member who would be within the boundaries of the rectangle but way too far from the diagonal.

In the end we came up with the idea of using an ellipse. The ellipse is a wonderful shape because it can squeeze in what you like and discard the rest not like a circle and it can be mathematically defined as well. The idea behind the ellipse was simple that we take the major axis of the ellipse as the line that joins the start and stop coordinate of the ride. The minor axis is taken as constant or variable based on the deviation required in the route. Now if any user wants to join a specific route they will be asked for their start and stop and that is when the ellipse comes into play. A hypothetical ellipse is made by the program using the start and stop coordinates of the ride as the line for the major axis and a minor axis is also formulated. Now if the person joining the ride has his/her start and stop inside the ellipse then they are accepted in the route otherwise they are not.

Now the logic behind this is simple enough but the earth is so big and there could be a million rides in the database the algorithm won't make an ellipse every time a user searched for a ride. Hence the table consisting of all the relevant information also contains a country column. First the country column is matched with the users search ride in a specific country. After which a radial search is made based upon the user's request every ride is taken which overlaps or touches tangentially or is around some distance to the user's start and stop. Now that we have taken a finite amount of rides into consideration we can now use our ellipse program to figure out if a user will fit into a group or not. Before the ellipse part is taken simple parameters like time of travel, specific days of travel in a week and also the number of seats available in the car are also taken into account. Once all these parameters are considered only then a user is allowed to enter into the group.

The formulas used are as follows:

Predefined Carpool Group

Source	Point A (x_1, y_1)
Destination	Point B (x_2, y_2)
Centre	Point O ($\lvert (x_1 - x_2)/2 \rvert$, $\lvert (y_1 - y_2)/2 \rvert$) = ($x_0$, y_0)
Euclidean Distance	$r = ((x_1 - x_2)^2 + (y_1 - y_2)^2)^{1/2}$
Circle	$(x - x_1)^2 + (y - y_2)^2 = r^2$
Ellipse	$(x - x_0)^2 / r^2 + (y - y_0)^2 / (r/2)^2 = 1$

Table 2.2 Formulations for carpool group

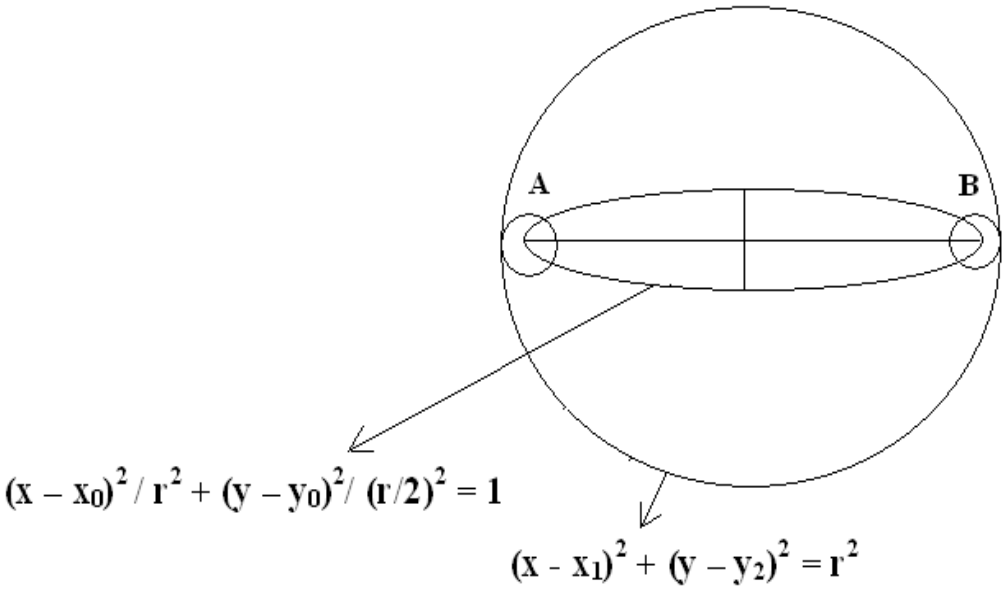


Fig 2.5 Elliptical representation of route

2.5.2 Flowchart

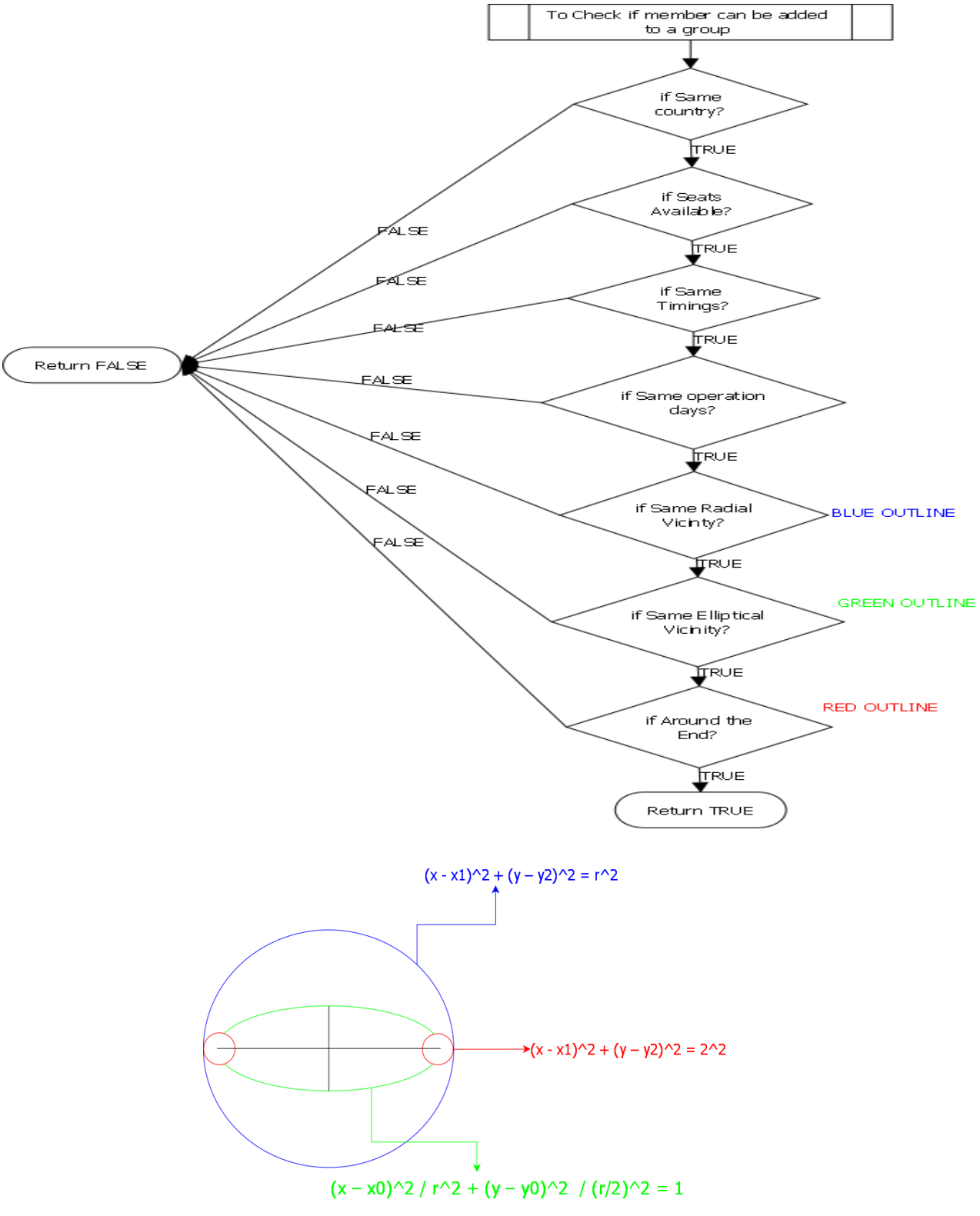


Fig 2.6 Flowchart for ellipse

Chapter 3

Problem Statement

Problem Statement

There is acute problem of traffic on roads these days and the increasing fuel prices add to the misery of daily users of personal vehicles. Also use of vehicles causes pollution which has its adverse effects. Car sharing is a solution but issues like security and trust come into picture.

Can this problem be solved? Solution to this problem is Carpool system. The Carpool system would enable its user a safe and secure way to share cars. This could include both short daily journeys such as going to workplace within the city and also long inter-city trips. The customer enters personal details and route details as the input according to the availability of the service customer will be assigned to the appropriate group which will also include the members share the same or nearby route and also along the same travel route.

In some case if the members are less (i.e. 1 or 2) then the services will not be provided as the cost will be comparatively larger and service will not be beneficial for the customer. Only trusted users will be allowed to use this so misuse of the service is avoided. In few cases if the user is in the close proximity of start and end node and wants to travel along the same path then that user is also fortunate to get added to the same group through various mathematical formulation and AI so that user is allowed to get assigned into preexisting group rather than a new group.

Overall Process

- Registration process for a new user.
- Sign in process after registration.
- Dashboard contains the layout of all posted rides and to which carpool group the user belong.
- Post ride form wherein the user can post his ride for the first time.
- Search ride form where in user can search for preexisting ride.
- View all rides pages contains list of all the rides posted by other users and not by it.
- Internal messaging system for sending messages to only the group members about any ride delay.
- After registration, post ride, join ride or remove ride a notification will be delivered via email and SMS service

Chapter 4

Requirement Analysis

4.1 SRS Document

4.1.1 Introduction

4.1.1.1 Purpose

This document is a Requirements Specification providing the details for the Carpool application. The document will be used to elaborate the functionality of the Carpool application. The document addresses the concerns of two domains within the system. One domain that is intended is a web-based application that is accessible from an Internet browser.

4.1.1.2 Documents Conventions

The SRS document has been drafted according to University of Mumbai guidelines for preparation of thesis/dissertations/reports.

4.1.1.3 Intended Audience

The intended audiences are the daily commuters who travel the same route daily.

4.1.1.4 Additional Information

Commuters that live near each other and share a common destination form the simplest and most common "carpool" arrangement. Carpooling is an ideal cost saving arrangement, particularly for those individuals who commute long distances to and from work each day have limited access to public transit and few transportation options available to them

4.1.1.5 References

http://www.carpool.in/etiqiette.htm

http://www.carcabpool.com/carpool-beneifits.html

4.1.2 Overall Description

4.1.2.1 Product Perspective

The carpool application will allow the users to register and update their profile. Once registered, users will be able to see locations of other users near his or her area. Users can then select groups to carpool to a pre-determined location or a location for special events.

4.1.2.2 Product Functions

The carpool application will support the following functionality:

- Login – logging onto the web server.
- Search Ride – search other members participating in the carpool.
- View trip's detail – view detail of the one-time event schedules.
- Search the route for a particular ride.

4.1.2.3 User classes and characteristics

The primary users of the carpool application are as follows:

Stakeholder Name	Priority
Full-time Employee	High
Software developer	Medium
Software tester	Medium
Admin maintaining the web server	High

Table 4.1 Primary Users of Carpool Application

4.1.2.4 Operating environment

Minimum Requirements:

- Any OS will do as it is web application.
- Web Browser preferably ones that support javascript.
- Internet connection.

4.1.2.5 User environment

The first step is reminding commuters of how much money and time they can save by carpooling or taking public transit. When determining how much money you can save, be sure to count not only your savings on gas but also reduced wear and tear on your vehicle, value depreciation and even possible insurance savings. After entering information such as the number of days per week you work, your distance to work, and how many miles per gallon your car gets, you might be surprised by how much you can save by carpooling.

4.1.2.6 Implementation Constraints

Software constraints:

- Net Framework 1.0 or higher.

4.1.2.7 Assumptions and Dependencies

Stakeholder should be proficient enough to browse the World Wide Web. Each stakeholder has appropriate information to register for carpool. Administrator will have the web server running at all times to avoid downtime of the web server.

4.1.3 Interface Requirements

4.1.3.1 User Interfaces

On the main page, the user will be given an option to login or register as a new user. Once logged in, the user will be a given the option to edit their profile, perform a carpool search, create a one-time event, create a recursive carpool, view one-time events and recursive carpools, and delete carpool.

When creating carpools, users will have the ability to establish their own preferred start and end points. Then, when establishing carpools, they will be able to select from others' start and end locations or use one of the pre-defined locations.

4.1.3.2 Hardware Interfaces

The hardware interfaces (such as network connectivity) will be managed by a third-party Internet Service Provider.

4.1.3.3 Software Interfaces

- Java language
- SQL server
- Apache Tomcat Server for the database connectivity that uses JSP.

4.1.3.5 Communication protocols and interfaces

As mentioned earlier internet connection is necessary for searching of routes and exchanging of messages so that the customer remains updated about the particular ride.

4.2 Use Case Diagram

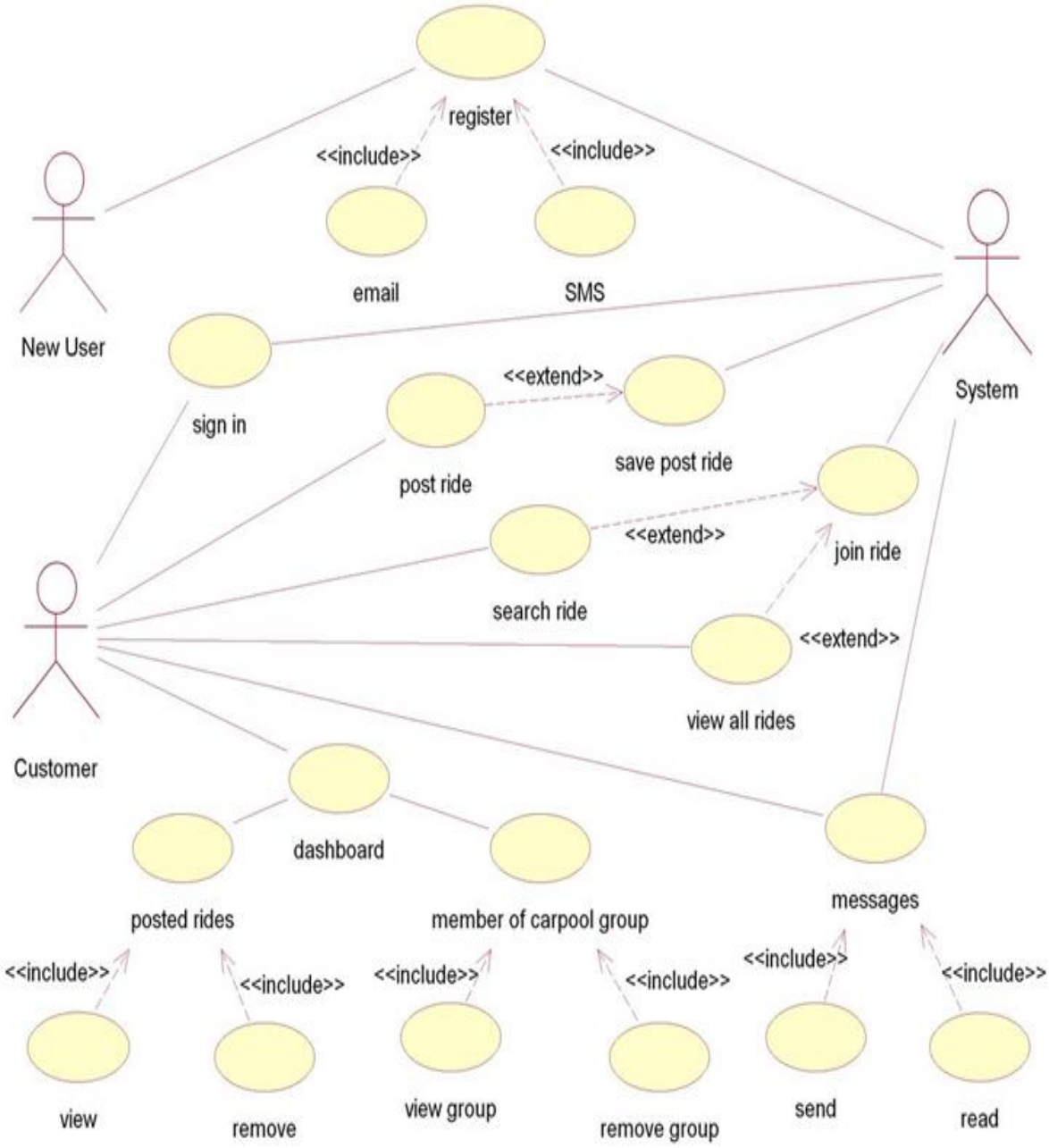


Fig 4.1 Use Case diagram

4.3 Activity Diagram

4.3.1 Post ride

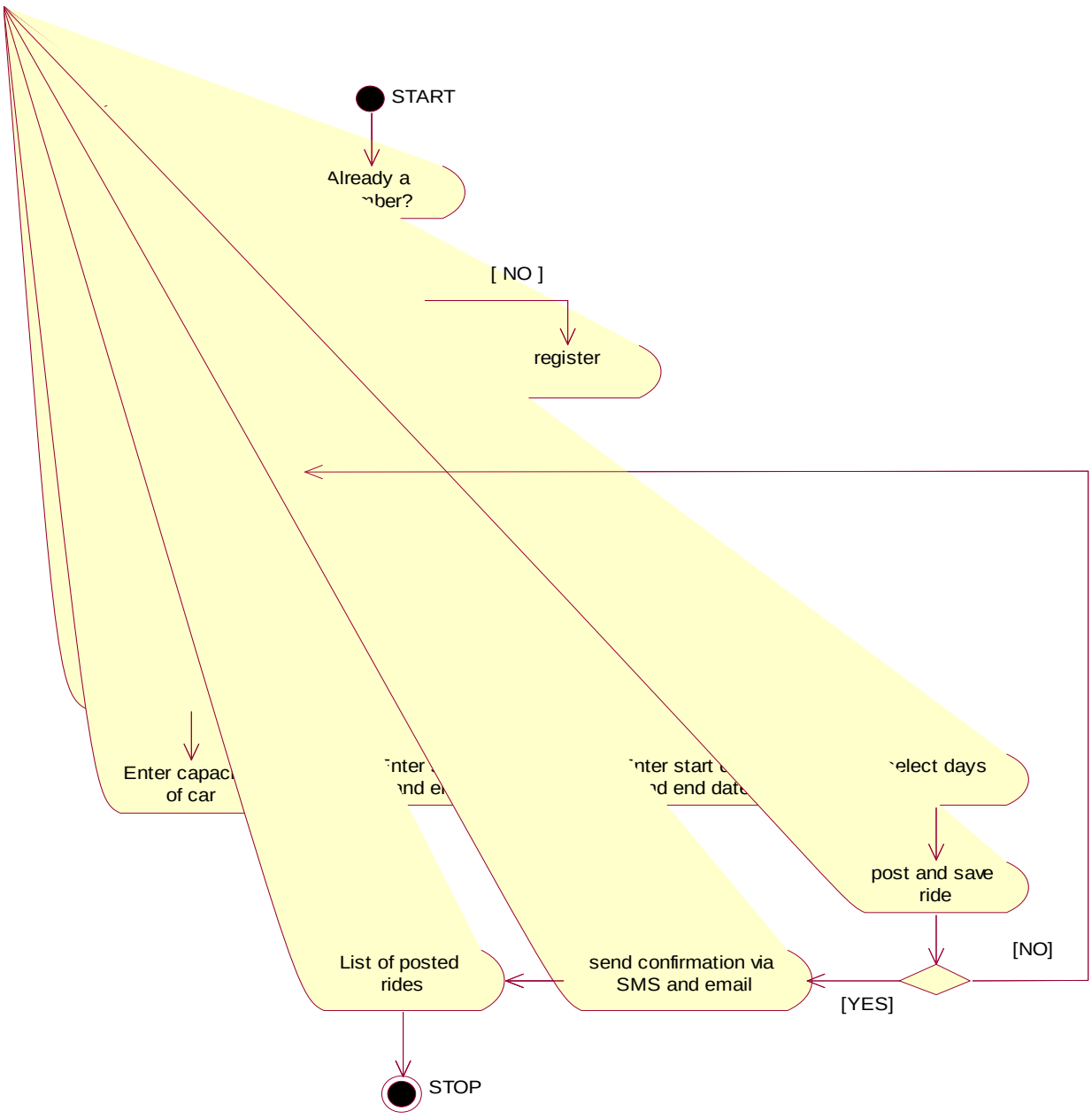


Fig 4.2 Activity diagram for Post ride

4.3.2 Search ride

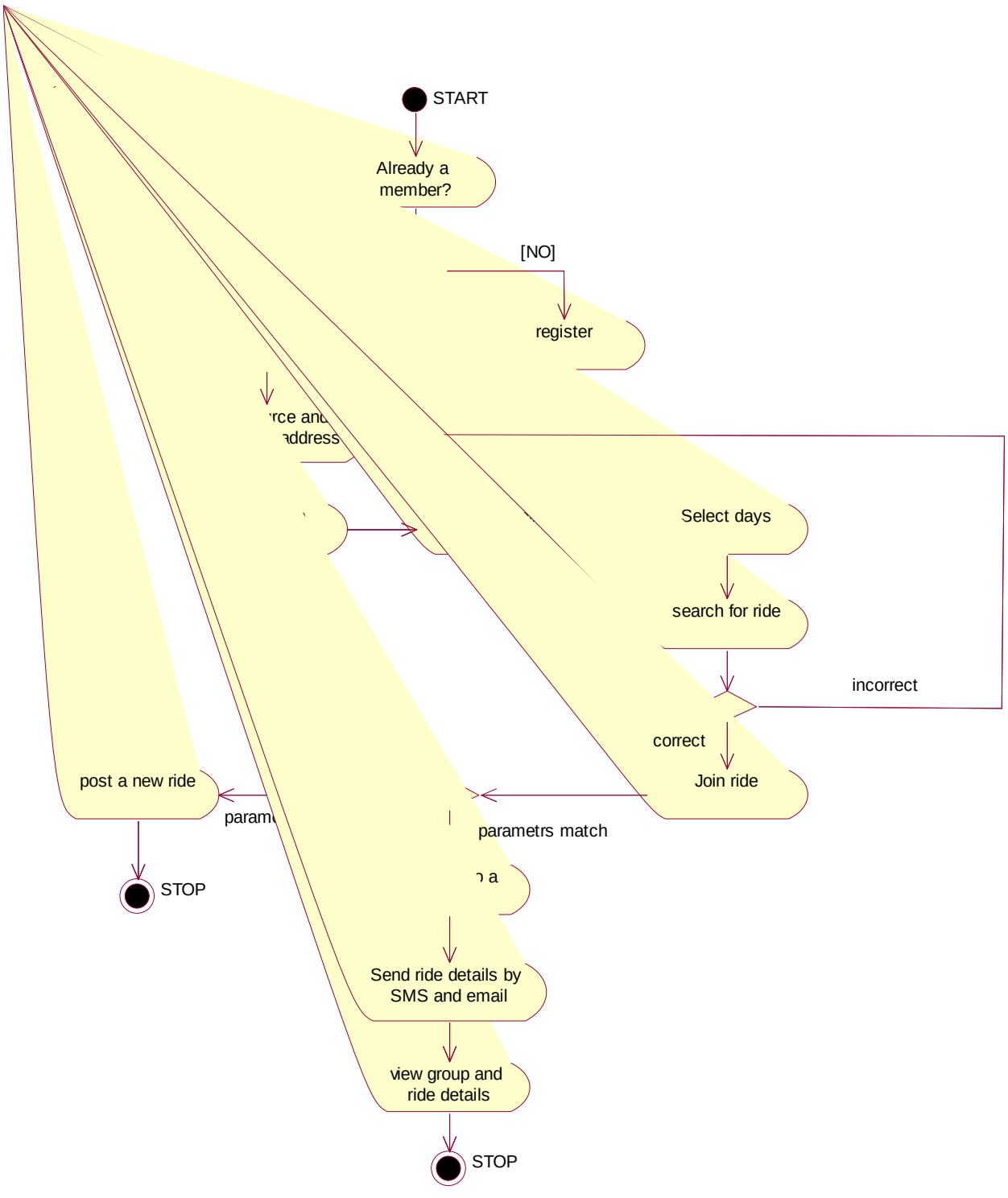


Fig 4.3 Activity diagram for Search ride

4.4 System Features

4.4.1 System Features A

- Sending registration or confirmation reports through emails.
- Calculation of distance or time required for the vehicle to reach the customers points.
- Group assignment on the basis on the nearest location.
- Professional look and feel.
- Browser testing and support for IE, Mozilla and chrome.

4.4.1.1 Description and priority

The process of registration, search ride, post ride are all necessary to act in a flow through which the verification details will be recorded and also deliver to the intended user. This avails each member about any modification of his group and ride which will maintain the process to work in a flow and always keeps the user updated.

4.4.1.2 Action

In case if the user does not receive any confirmation about his ride prior to the travel date then he is expected to inform or contact the service provider to that an alternative action of measure is provided without any delay or loss.

4.4.1.3 Functional Requirements

- A system who can enter the customer details like name, contact number, address, email id, vehicle details, route details, time etc.
- Users can register the details through website.
- The user can even search for a ride and post a ride.
- The facility to see the available services like distance of the route by different algorithms and their efficiency.
- Facility to check whether the driver or the passenger is a registered user or not for this purpose a verification of the customer identity or address proof is must.

4.5 Other nonfunctional requirements

4.5.1 Performance Requirements

- Secure access to confidential data (carpooling details).
- 24*7 availability.
- Flexible service based architecture will be highly desirable for future extension

4.5.2 Safety Requirements

- Customers are required to carry up-to-date identification proof to verify their identity.
- Maintenance of all cars should be done using regular intervals.
- Users are expected to drive with accurate measures and regulations.

4.5.3 Security Requirements

- Secure access to confidential data (carpooling details).
- Email ids and password should be kept private.
- For verification purpose user identification proof should be attach during registration.

4.5.4 Software quality attributes

- Functionality: The system provides interoperability, accuracy and reliability.
- Reliability: The system will be available to user till the time the user is connected to the internet.
- Usability: The system is user friendly i.e. the user can understand it easily, learn the system easily and can operate the system easily.

Chapter 5

Project Design

5.1 Software Model

The System is to be developed using incremental model. The incremental model combines the elements of waterfall model applied in an iterative fashion. The incremental model applies linear sequences in a staggered fashion as the calendar time progress.

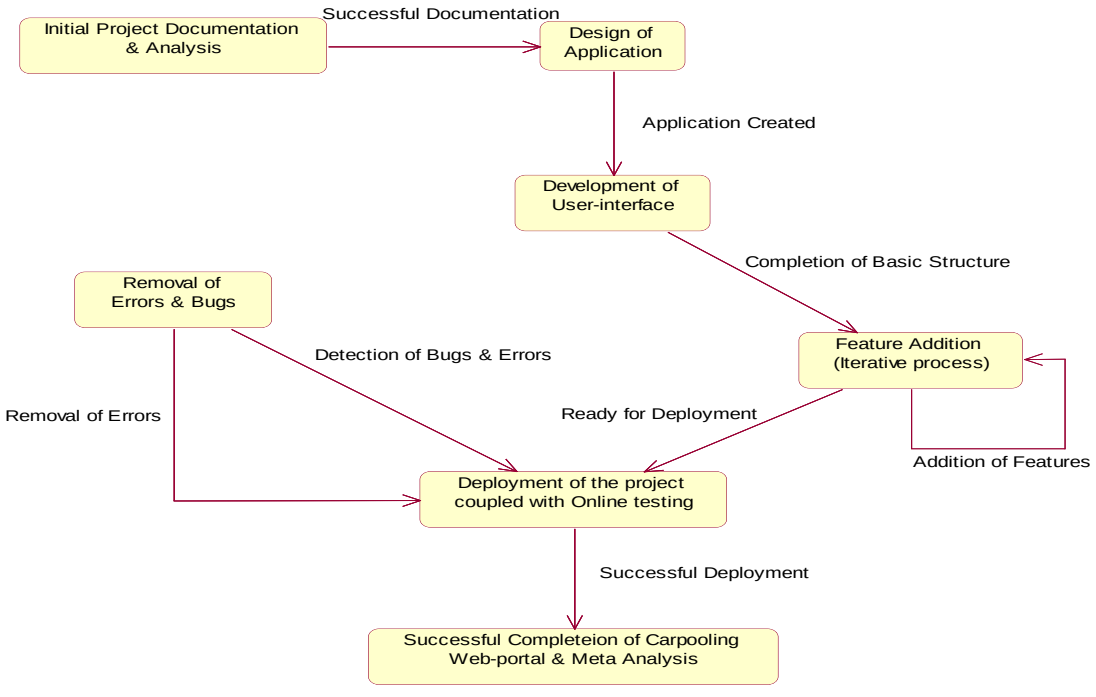


Fig 5.1 Software Model

5.2 Database Schema

A database schema of a database system is its structure described in a formal language supported by the database management system (DBMS) and refers to the organization of data to create a blueprint of how a database will be constructed (divided into database tables).

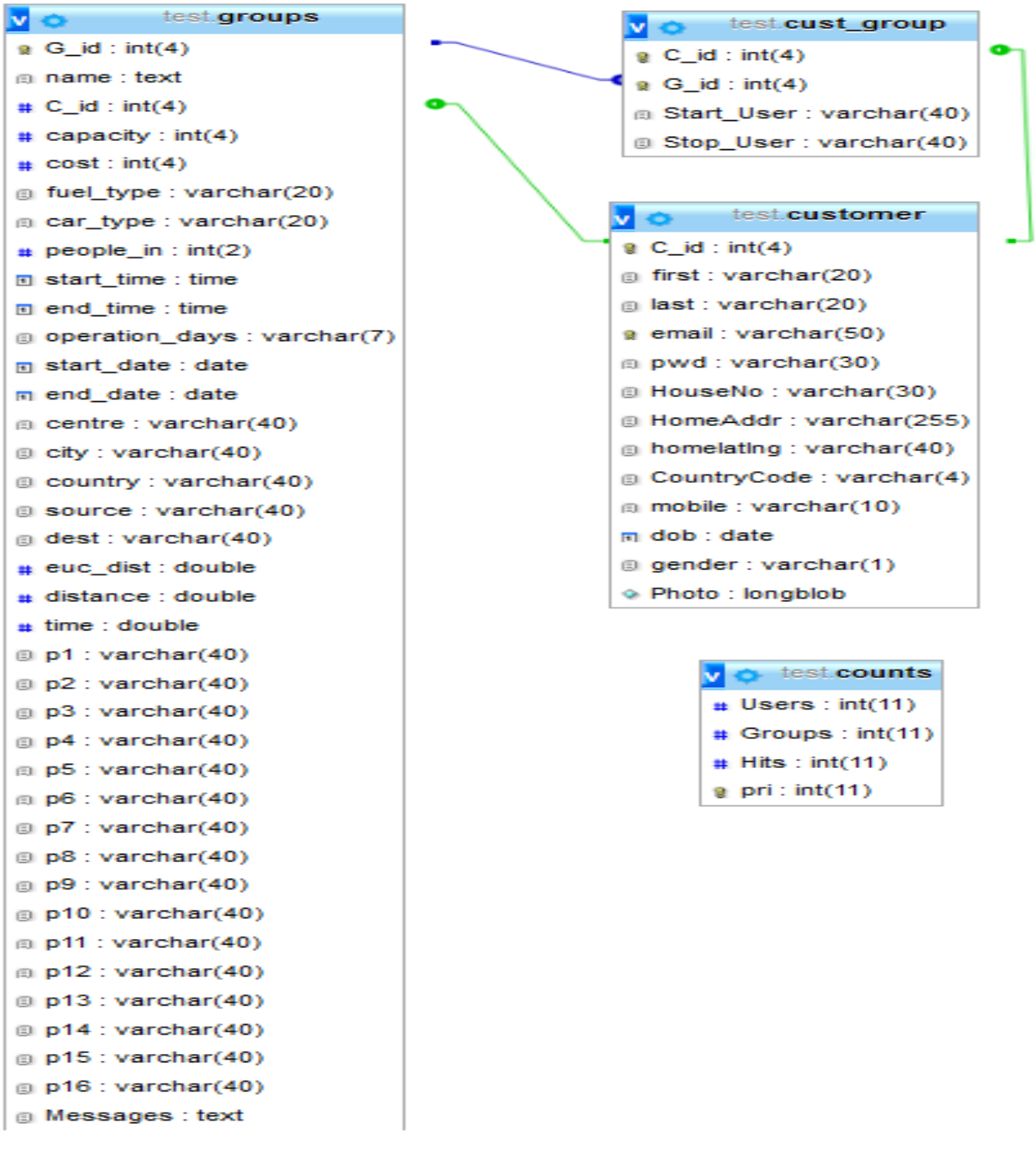
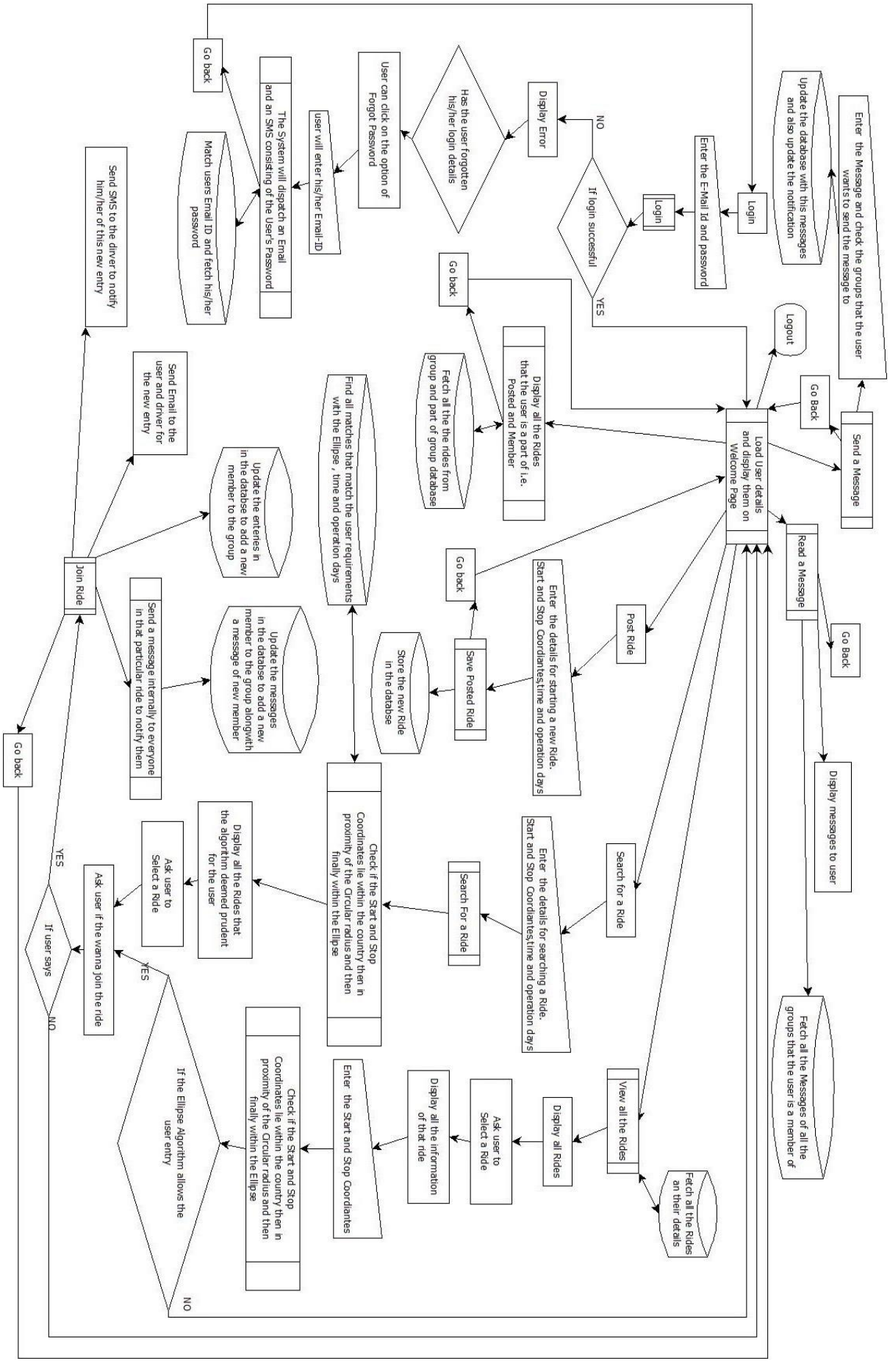


Fig 5.2 Database Schema of the system

5.3 Overall system processing



5.4 Sequence Diagram

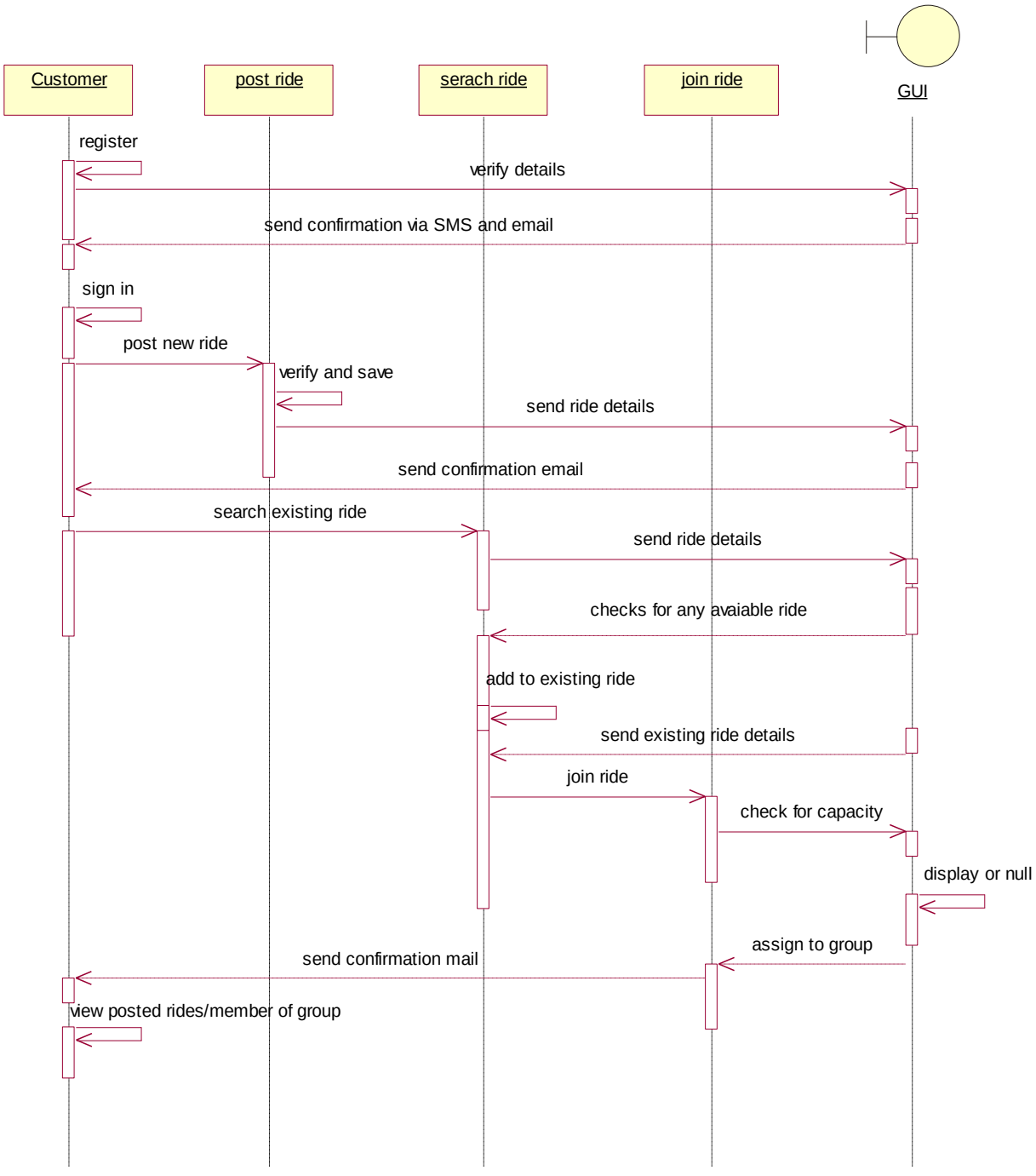


Fig 5.4 Sequence diagram

5.5 Flowchart for trip route optimization

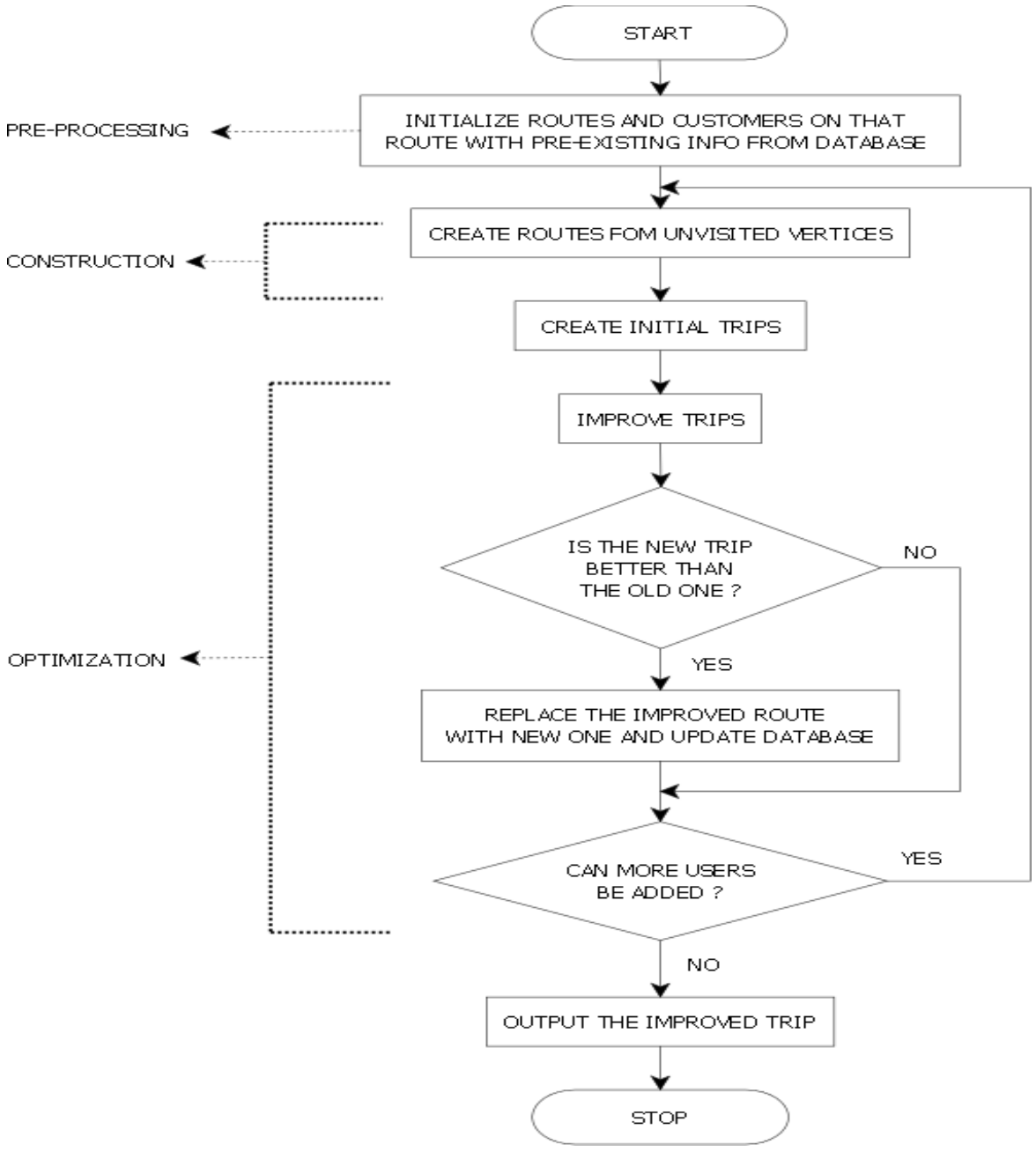


Fig 5.5 flowchart for trip route optimization

Chapter 6

Implementation Details

6.1 Overall processing

STEPS:

1. **Registration form:** The user needs to register where he is asked for his personal details. After he submits his registration details a notification message is delivered via SMS and email.

2. **Sign in field:** As user sign in various captions will be provided like Dashboard, Search ride, Post ride, View all rides & Messages.

3. **Dashboard page:** It contains 2 fields posted rides and member of carpool group. List of all rides posted is displayed on posted rides. User can join and remove his ride through this field.

4. **Post ride form:** If there is no exiting or matching ride for user he needs to post his ride which includes major factors like source and destination place his start and end time and also capacity of the car.

5. **Search ride form:** It basically allows the user to get into already exiting ride if the parameters of the ride matches and becomes the member of same group. If a ride according to user already exits then he can join that ride directly jus by inputting his source and destination details as he can view other details below.

6. **View all rides:** The user who posted his ride cannot be seen here but it can be viewed by all the other users so that they can join the ride if their details match.

7. **Messages send to group members:** User can send messages only to those who are assigned to his group and not to all users due to which any modification about ride will be delivered to that group member at same time.

8. **Forget password field:** Password can be recovered by just inputting the email address so that his password will be recovered and notification will be forwarded via SMS and email.

6.1.1 Registration Process

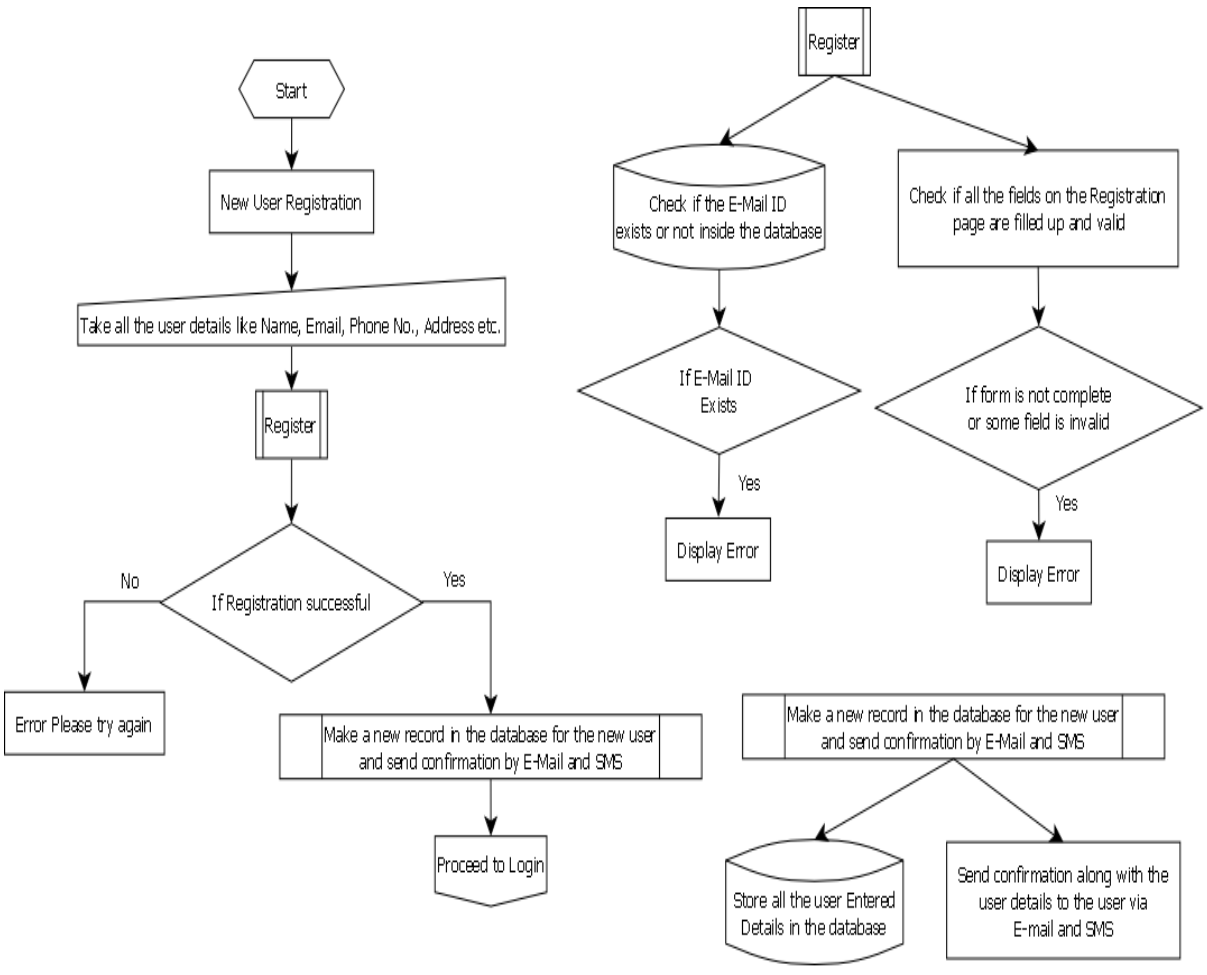


Fig 6.1 Registration process of system

6.2 Languages used

1. **JAVA -** We have used it to write all our algorithms which include the group matching algorithm and the route calculation algorithms. It's modularity due to its object formation property is very advantageous. We have used JDK 8.

2. **JSP (Java Server Pages)& Servlets-** Java Server Pages (JSP) is a technology that helps software developers create dynamically generated web pages based on HTML, XML, or other document types.

3. **HTML (Hypertext Markup Language) –**HTML is what any website is about and it is our frontend with the entire fancy GUI. It is brilliant based on the fact that by using JSP and JavaScript and CSS and AJAX and JQuery and the Google map API's it is all consumed by HTML and it is the form of our final output.

4. **JavaScript -** JavaScript (JS) is a dynamic computer programming language. It is most commonly used as part of web browsers, whose implementations allow client-side scripts to interact with the user, control the browser, communicate asynchronously, and alter the document content that is displayed.

5. **AJAX (Asynchronous JavaScript and XML) -** It is a group of interrelated Web development techniques used on the client-side to create asynchronous Web applications. With Ajax,

6. **Cascading Style Sheets (CSS) -** It is a style sheet language used for describing the look and formatting of a document written in a markup language. It was from the bootstrap website and the design is called flatly

7. **Structured Query Language (SQL) -** The scope of SQL includes data insert, query, update and delete, schema creation and modification, and data access control. Since it is a client based website with features that require a log of saved files for every user it made sense to use a database query language like SQL.

6.3 APIS & Packages used

1. **Google Maps API –** Google Maps is a web mapping service application and technology provided by Google, powering many map-based services, including the Google Maps website, Google Ride Finder, Google Transit, and maps embedded on third-party websites via the Google Maps API. It offers street maps and a route planner for traveling by foot, car, bike (beta), or with public transportation.

2. **Java Mail API -** Java Mail is a Java API used to send and receive email via SMTP, POP3 and IMAP. Java Mail is built into the Java EE platform, but also provides an optional package for use in Java SE. We have used the E-Mail as astute way of communication. An E-Mail is sent for any kind of activity by the user.

3. **Selenium Testing API -** Selenium is a portable software testing framework for web applications. Selenium provides a record/playback tool for authoring tests without learning a test scripting language (Selenium IDE). Selenium has not been actually used for any testing in the website all that we have done using it is to send an SMS. All that selenium does is to use an already existing SMS sending website to send an SMS programmatically without any user input .We require the SMS feature to alert users of any change that needs to be seen immediately by the user.

6.4 Development tools used

1. **NetBeans -** NetBeans is an integrated development environment (IDE) for developing primarily with Java, but also with other languages, in particular PHP, C/C++, and HTML5. We have made our entire project in NetBeans as it comes coupled with all of Java and JSP features and it also has Apache Tomcat embedded inside it. As our entire project is based on Java it was the most sensible choice to use NetBeans.

2. **Apache Tomcat -** Apache Tomcat (or simply Tomcat, formerly also Jakarta Tomcat) is an open source web server and servlet container developed by the Apache Software Foundation (ASF). Jasper is Tomcat's JSP Engine. Jasper parses JSP files to compile them into Java code as servlets (that can be handled by Catalina). At runtime, Jasper detects changes to JSP files and recompiles them. We needed apache tomcat not only to

run as a server but also to deploy all our JSP's and Apache is used to run the phpmyadmin of the MySQL feature.

3. **XAMPP for MySQL -**It is a free and open source cross-platform web server solution stack package, consisting mainly of the Apache HTTP Server, MySQL database, and interpreters for scripts written in the PHP and Perl programming languages but we have used Java and that too JSP to connect to the MySQL database created by Xampp. All the database of MySQL has been created by the phpmyadmin of Xampp.

6.5 Snapshots

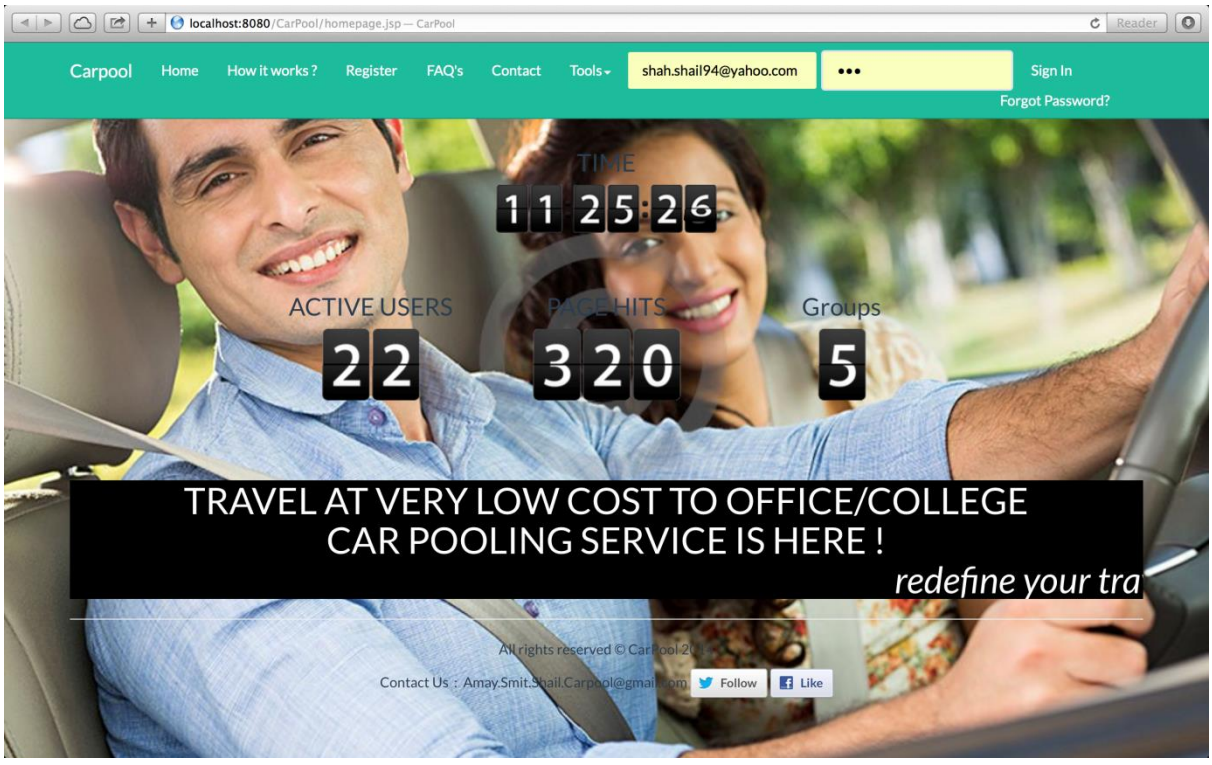


Fig 6.2 Homepage

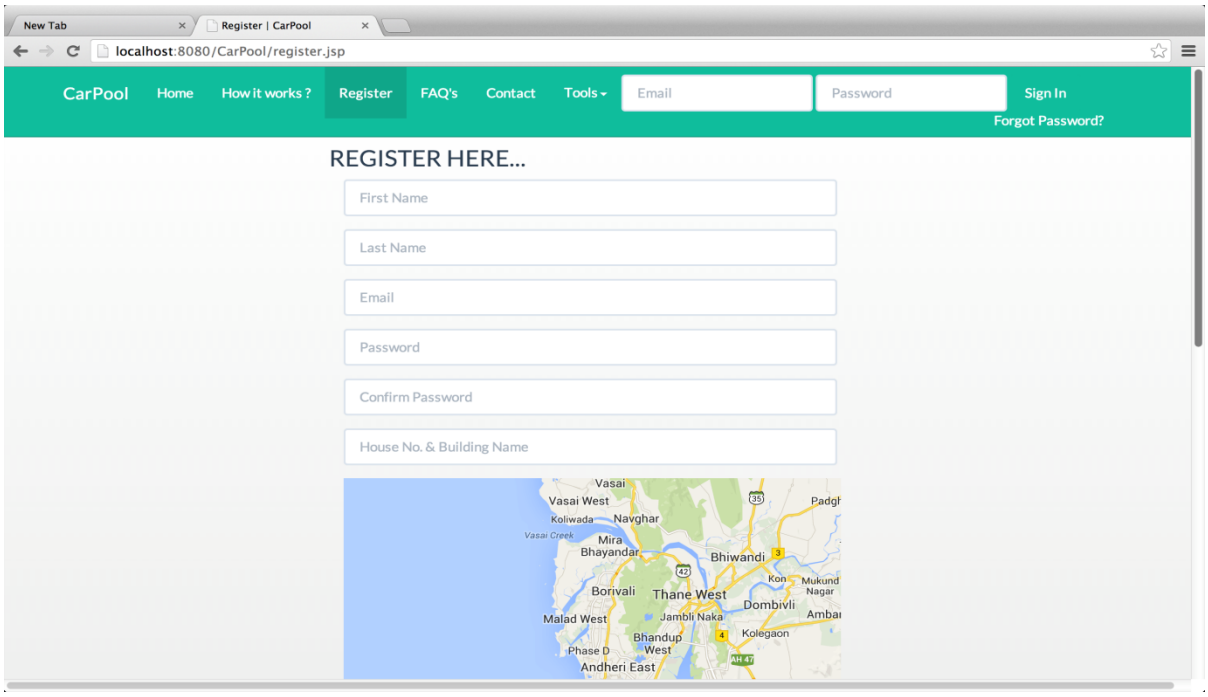


Fig 6.3 Registration Page part 1

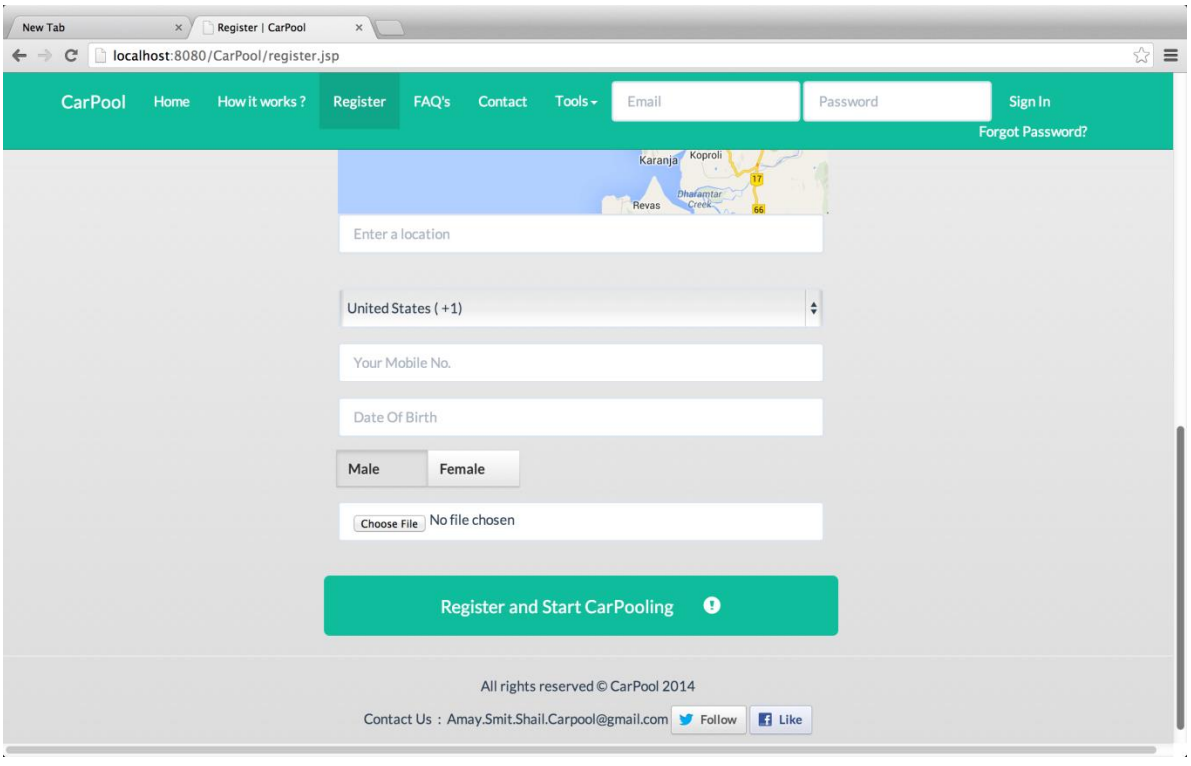


Fig 6.4 Registration Page part 2

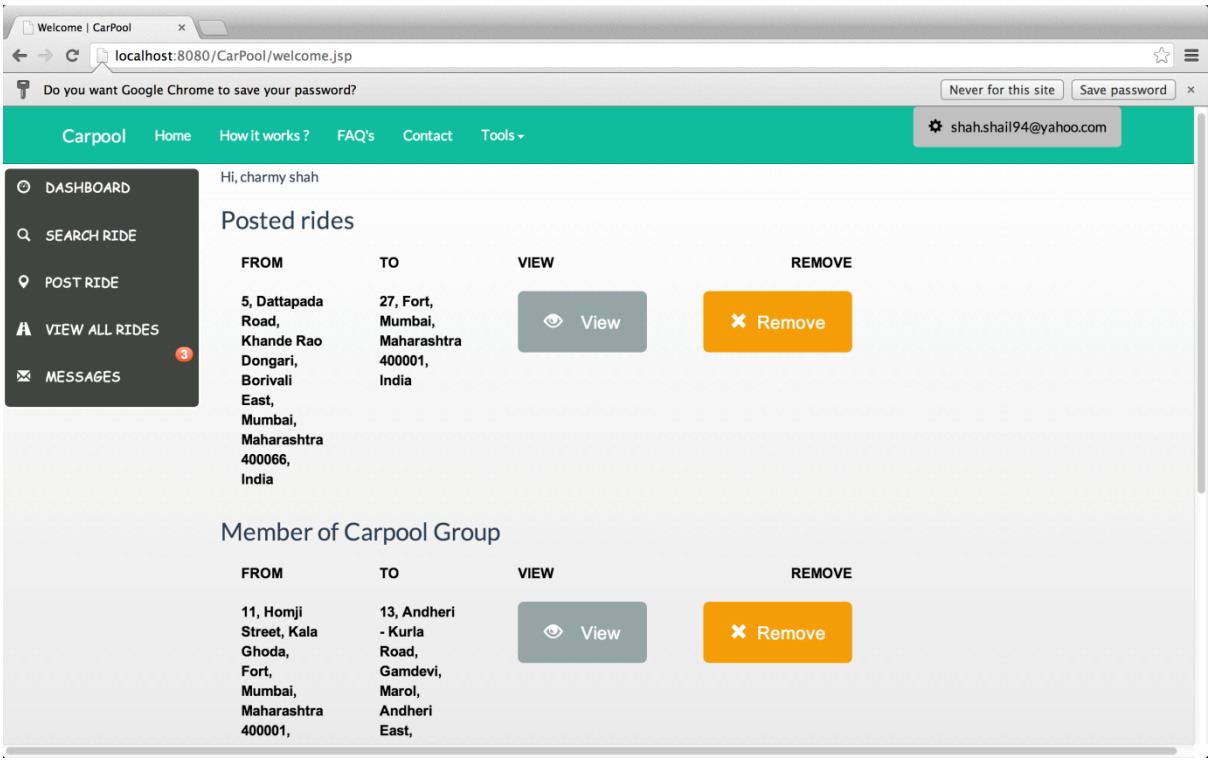


Fig 6.5 list of Posted Rides and Member of Carpool Group

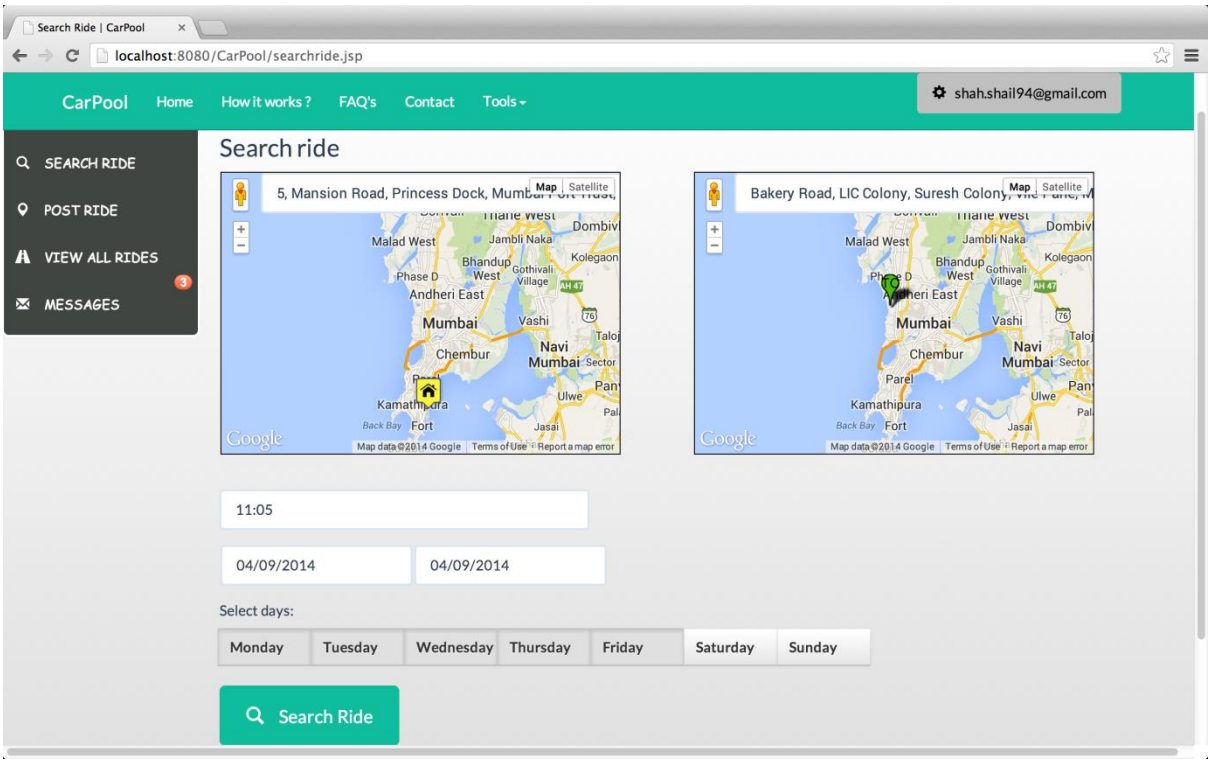


Fig 6.6 Search Ride Form

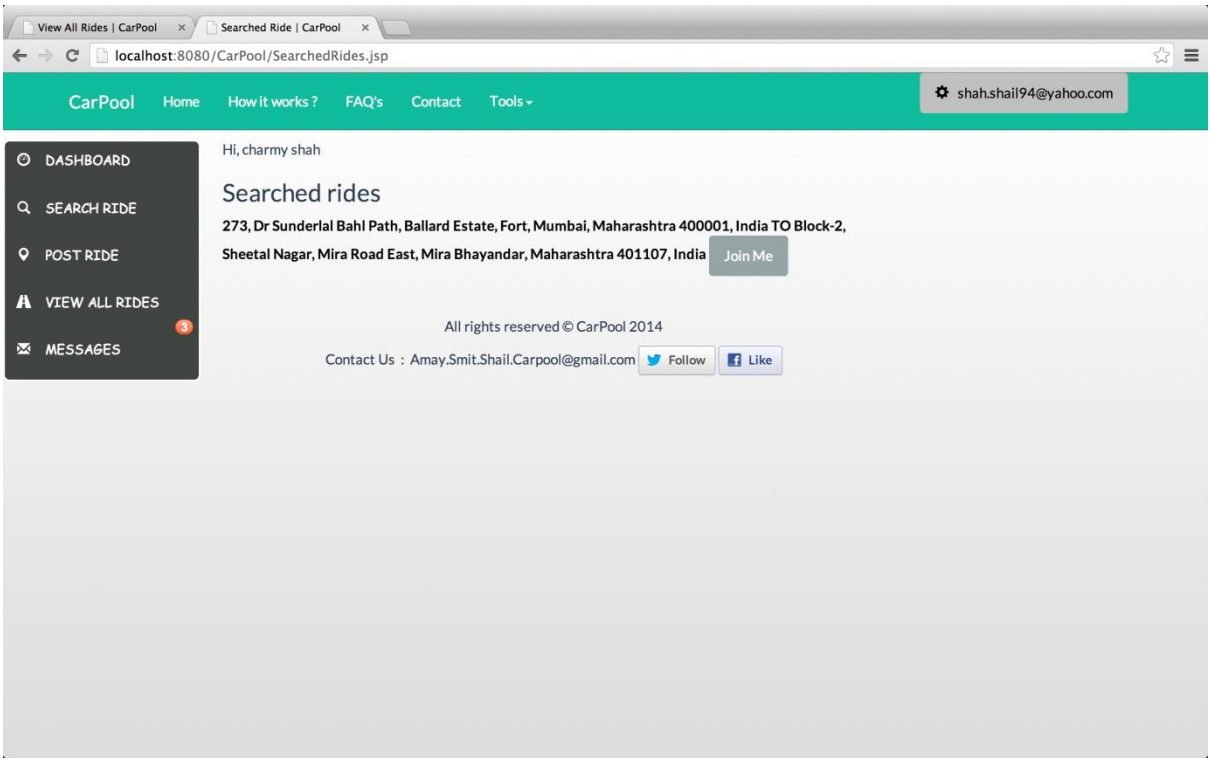


Fig 6.7 Result after Search Ride

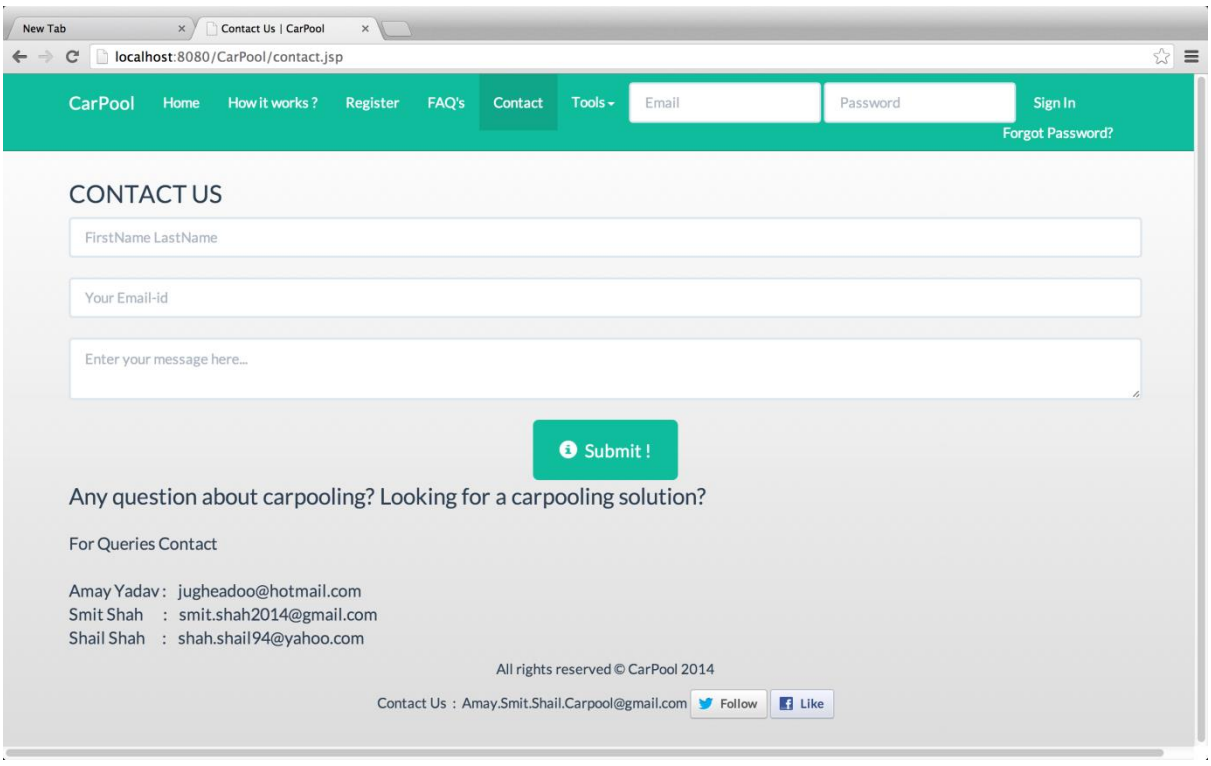


Fig 6.8 Contact us

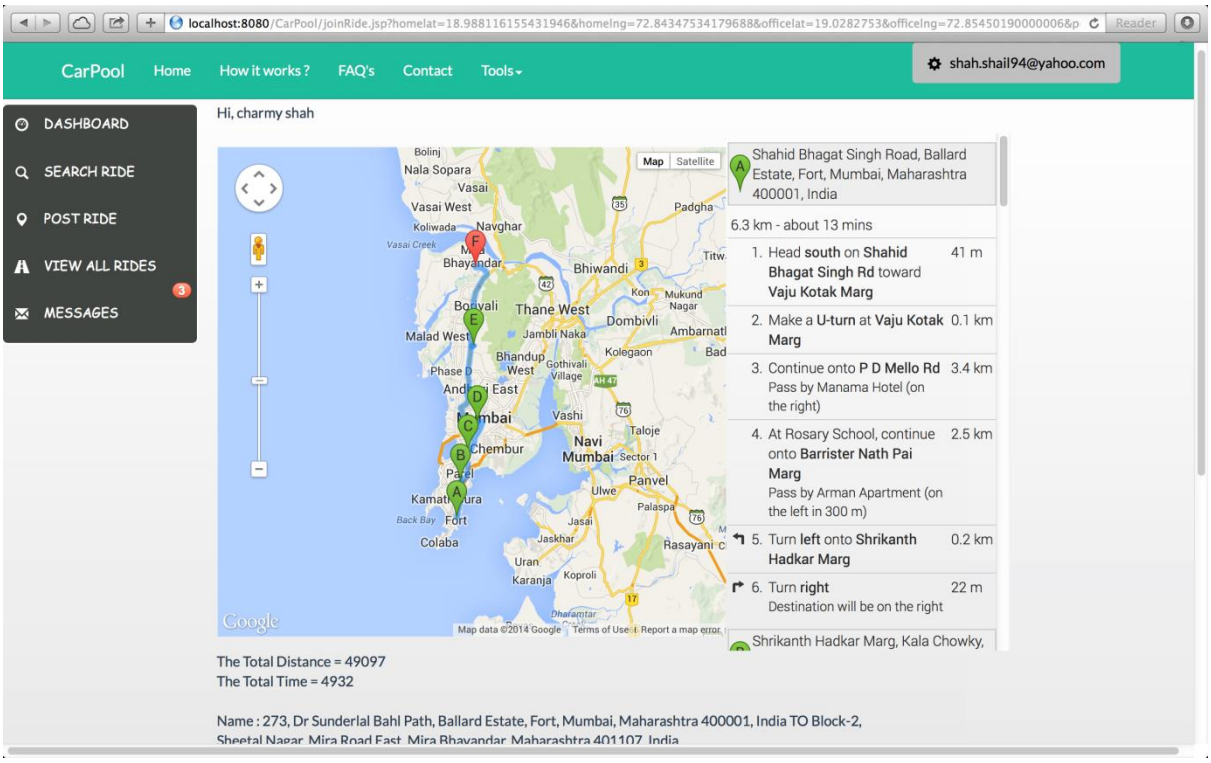


Fig 6.9 Joins the Driver Ride part 1

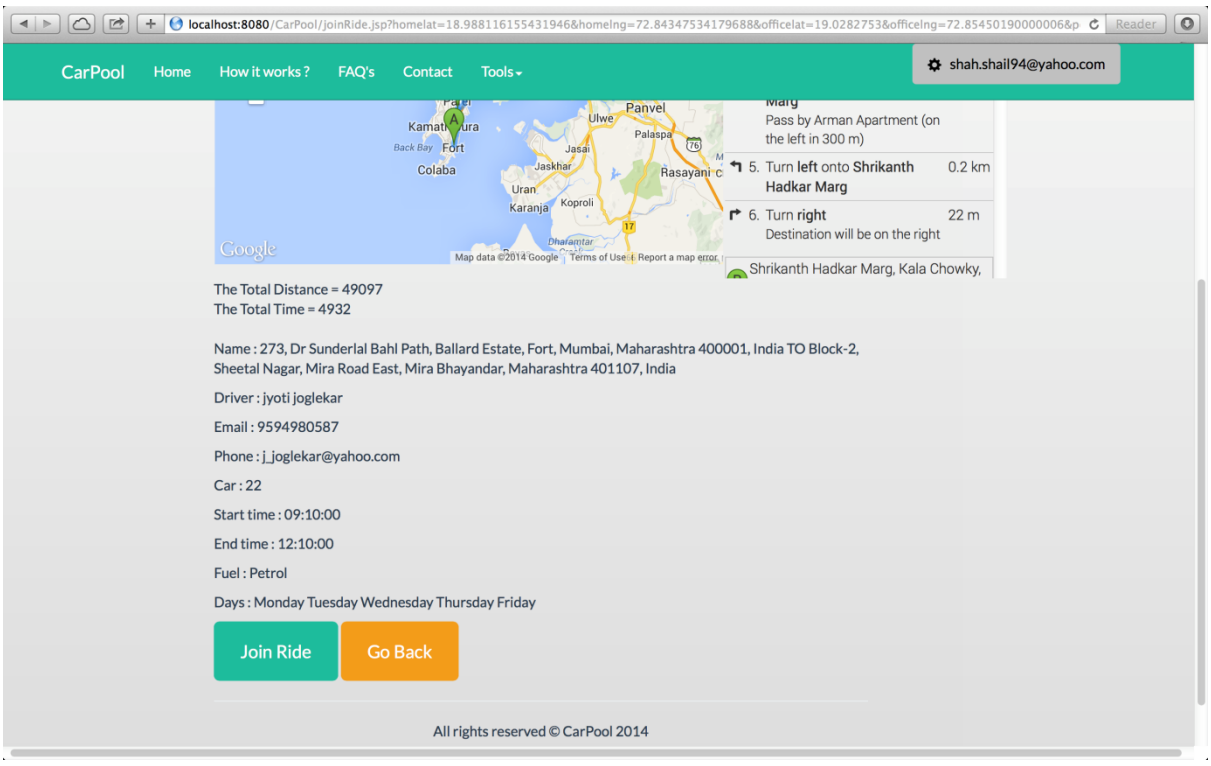


Fig 6.10 Joins the Driver Ride part 2

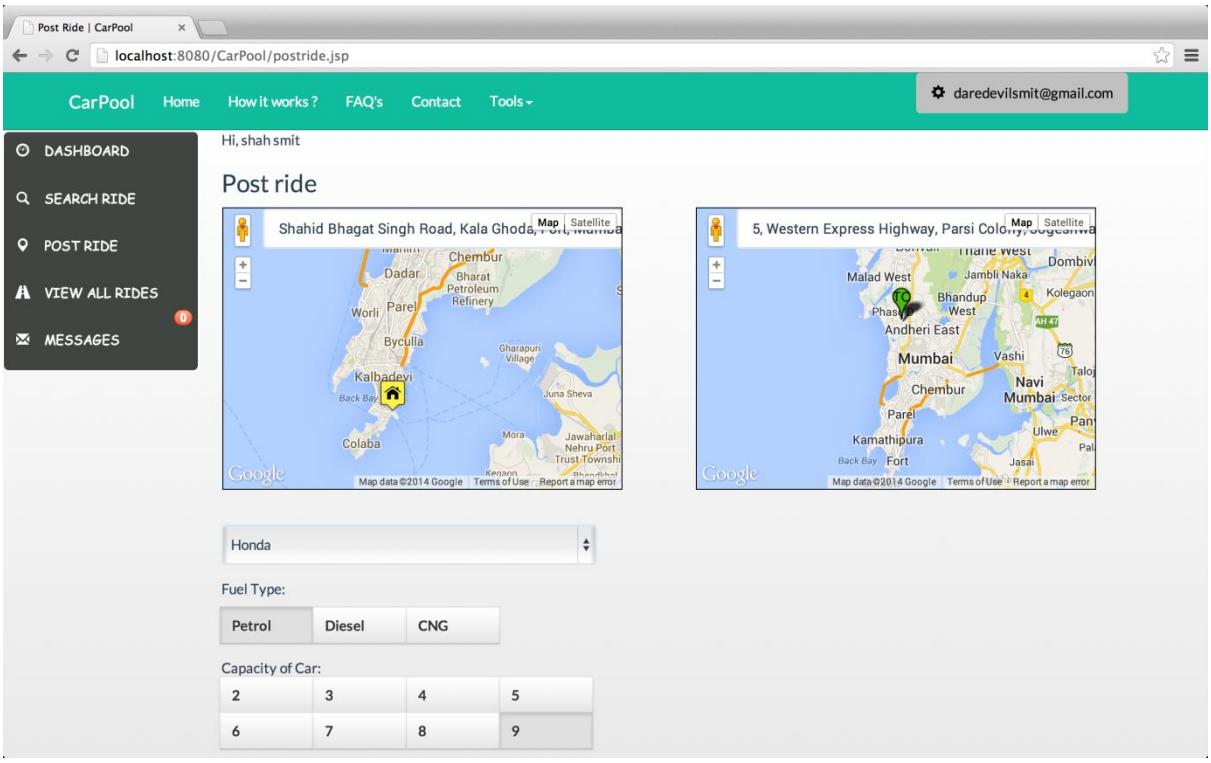


Fig 6.11 Post Ride Form Part 1

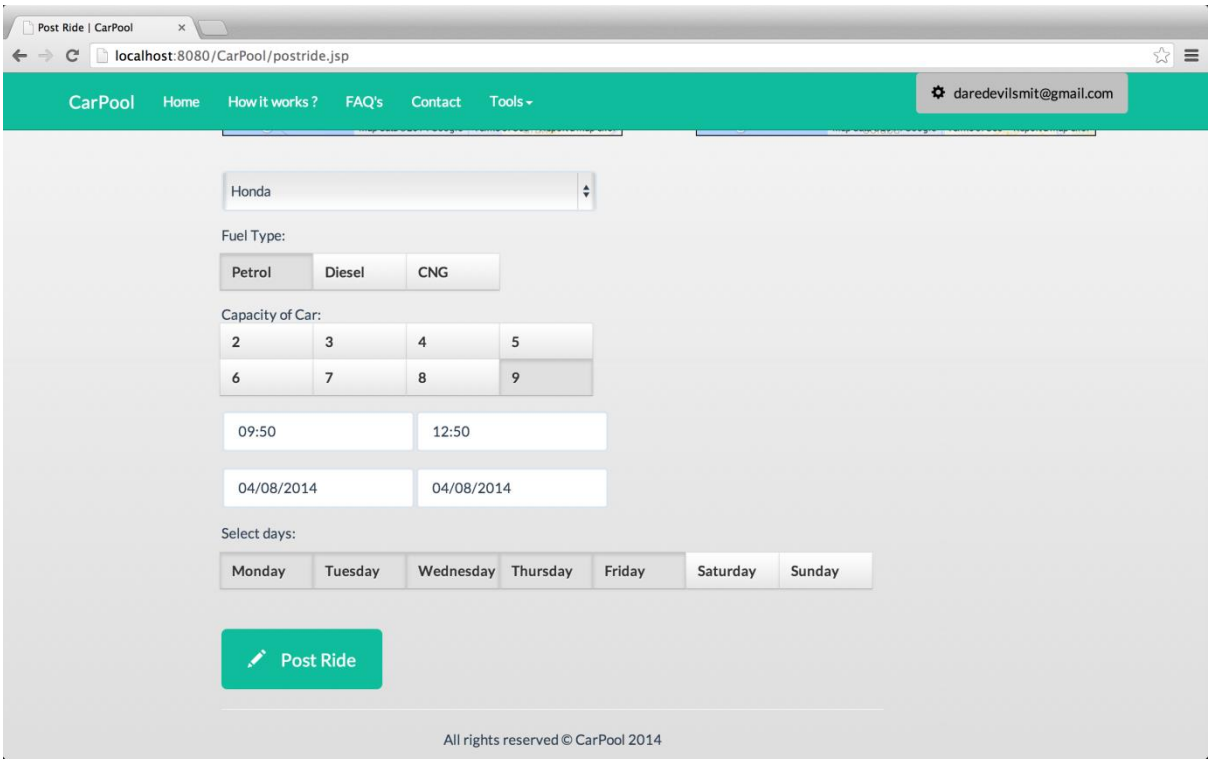


Fig 6.12 post ride form part 2

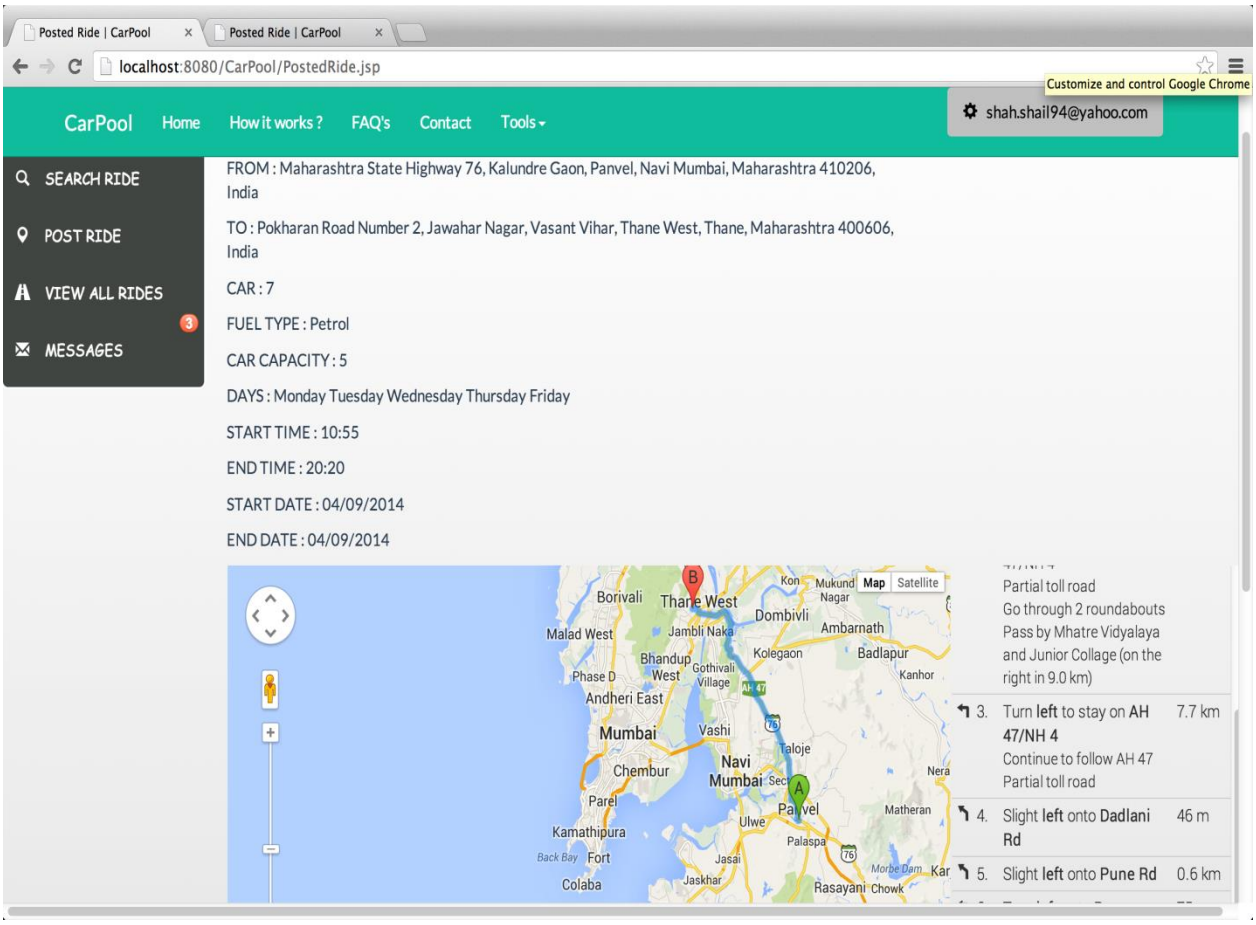


Fig 6.13 Display of Post ride

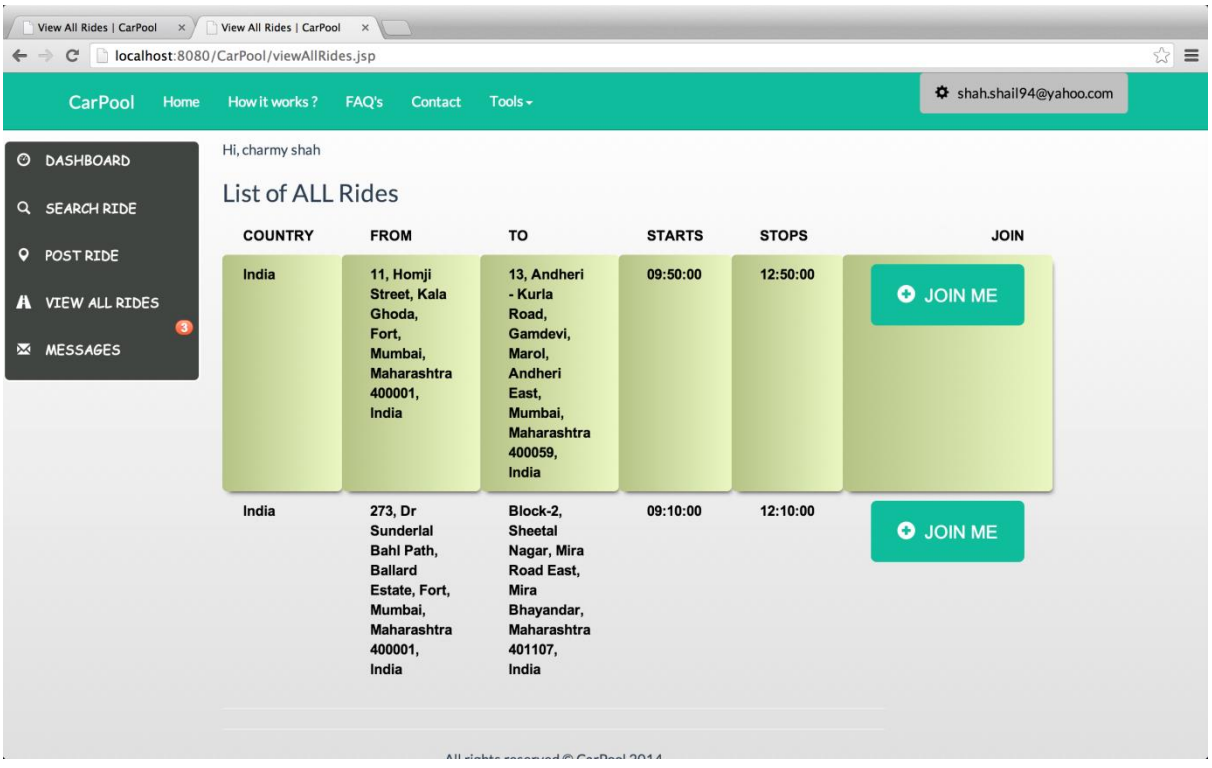


Fig 6.14 View all Rides pages

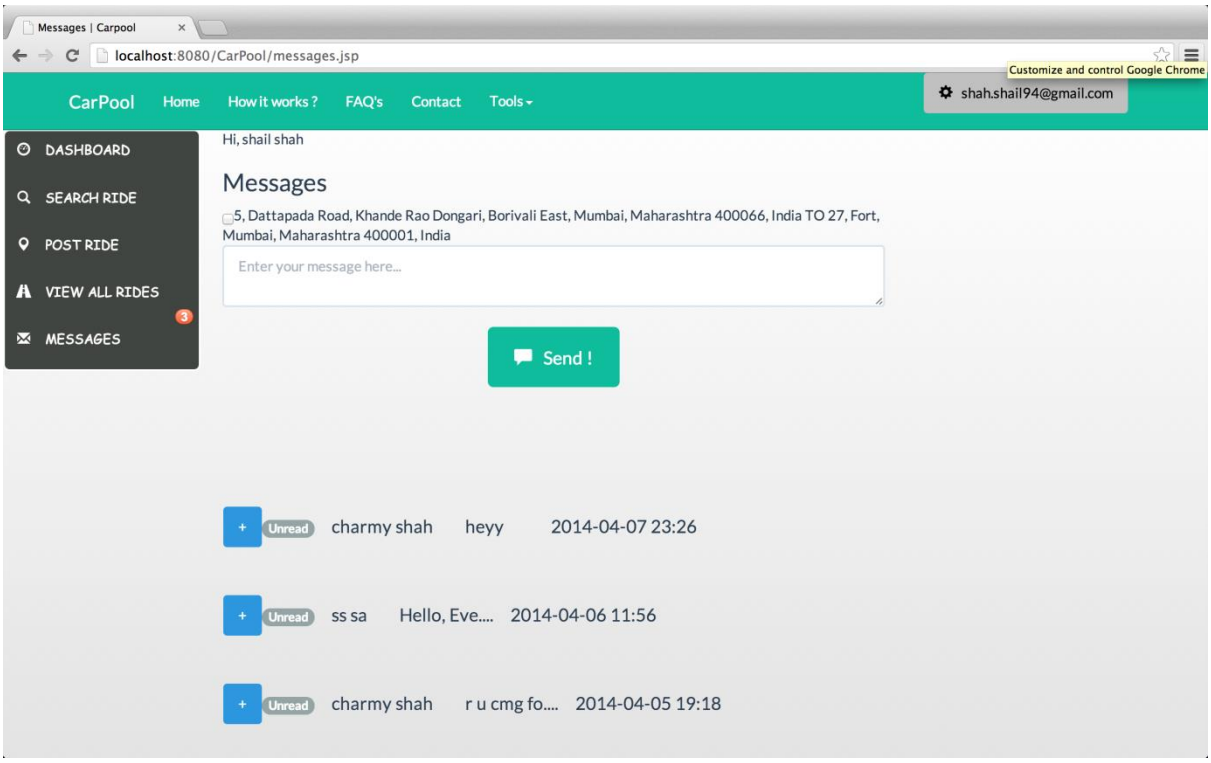


Fig 6.15 Messages send to Group Members

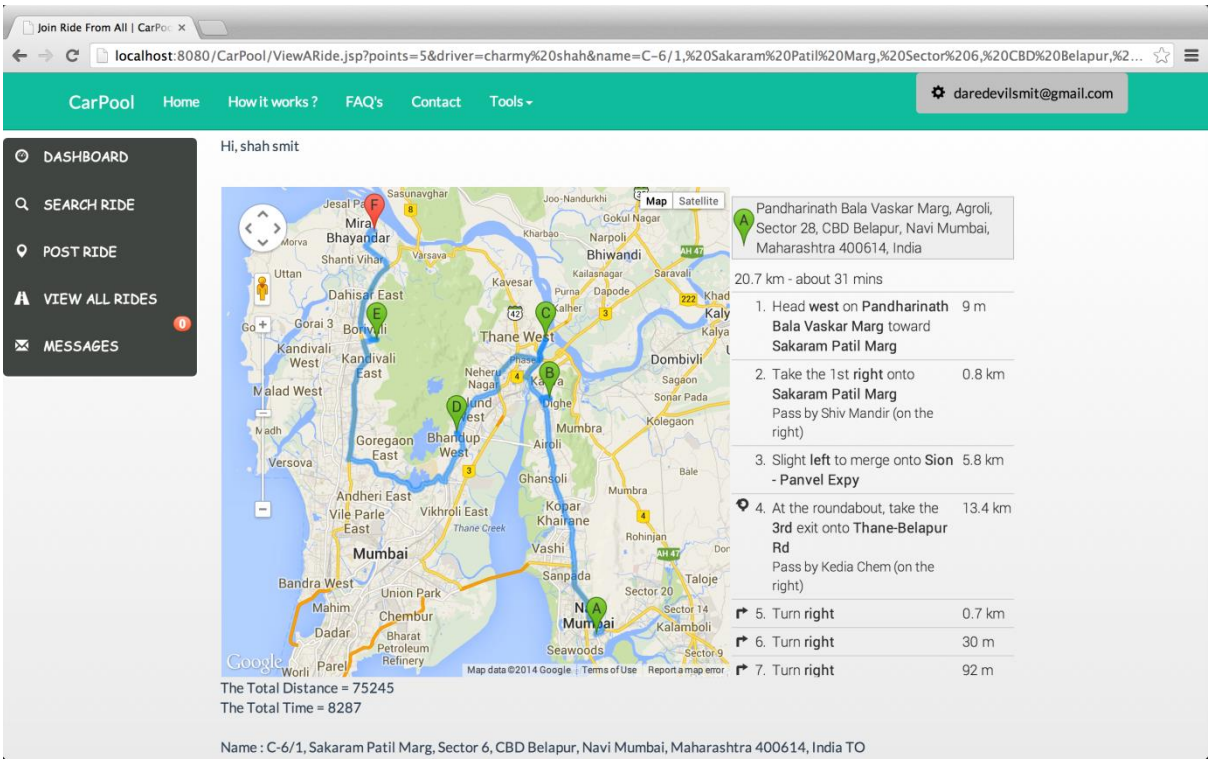


Fig 6.16 Multiple users added to Group

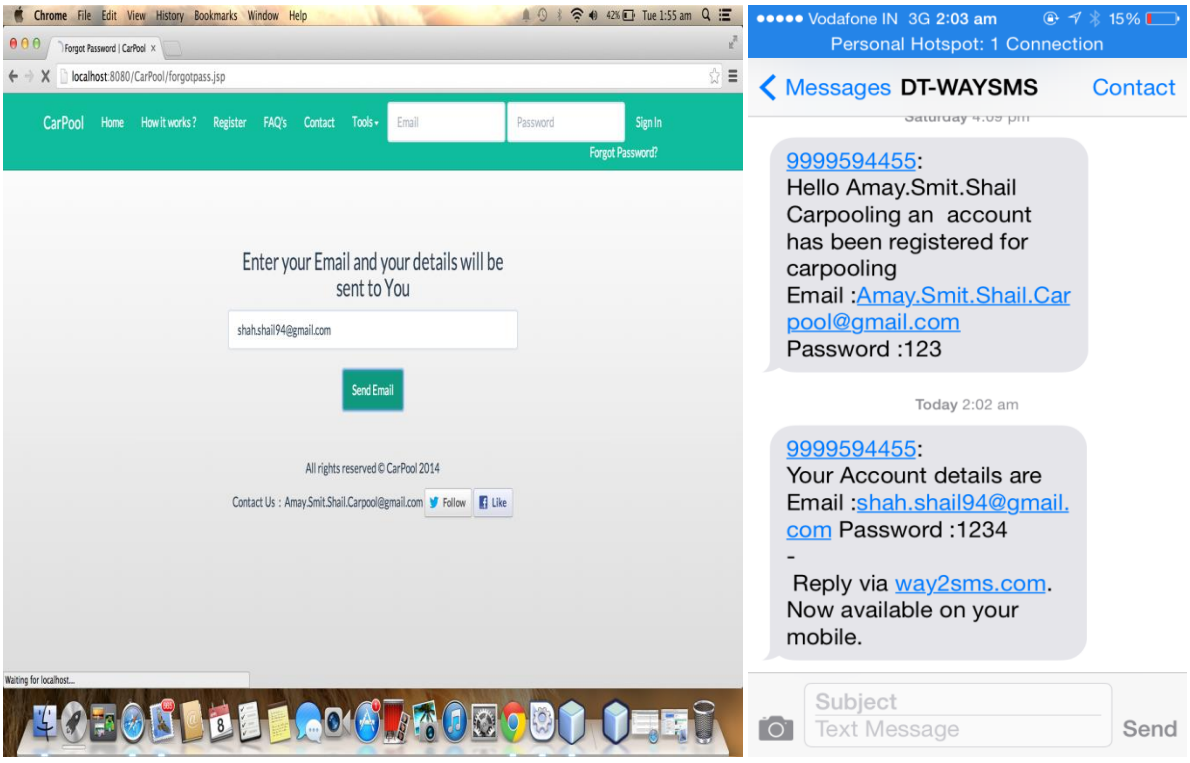


Fig 6.17 Forgot Password

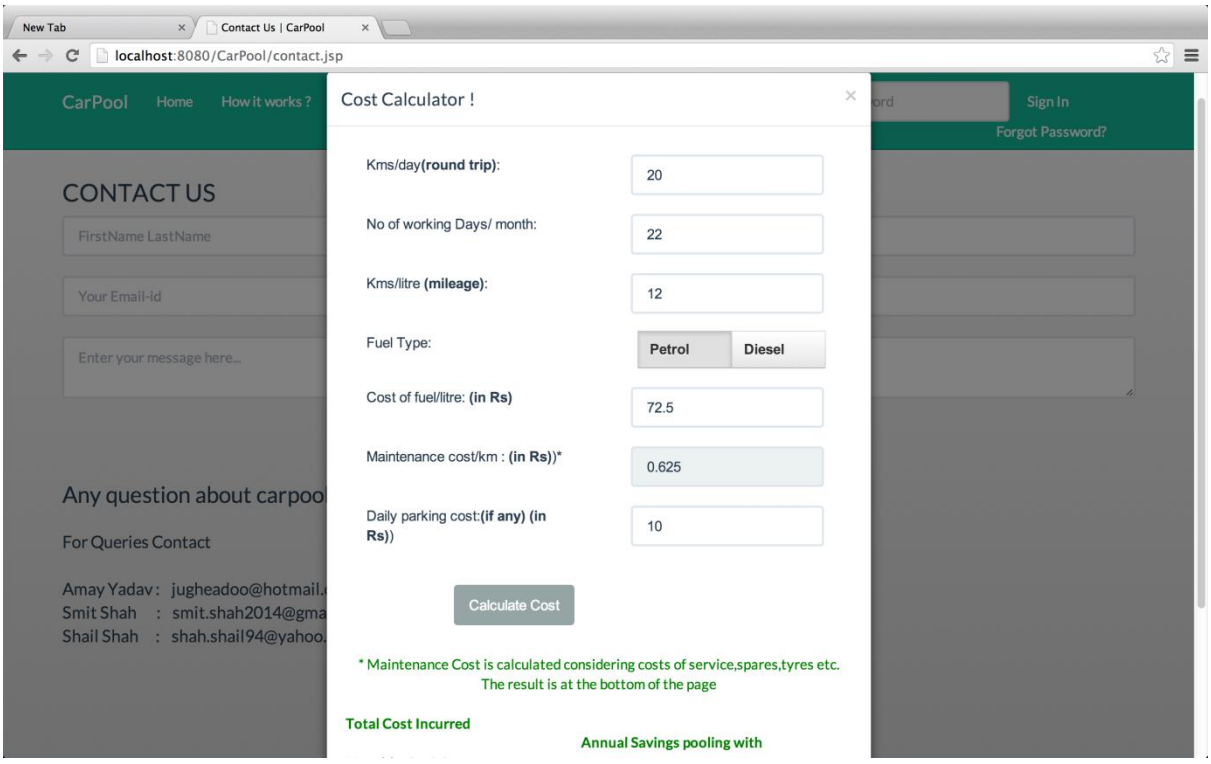


Fig 6.18 Cost Calculator

Chapter 7

Testing

7.1 Testing

TEST CASE	
Name of the test case: Group/route matching	
Pre-Condition: The user should be a new user and he/she is posting a new ride.	
Description:	contains source and destination fields
Input:	Place the markers each on source and destination field on the map.
Expected Output:	Display of his route on map with one marker as start node and other as end node.
Actual Output:	Appropriate route based on group matching algorithm
Result:	Test passed.

TEST CASE	
Name of the test case: Group/route matching	
Pre-Condition: Another user adds itself to a preexisting route.	
Description:	contains source and destination fields
Input:	Place the markers each on source and destination field on the map. Other details matching with already existing ride.
Expected Output:	Display of his route on map with one marker as start node and other as end node within the actual start and end node of the journey.
Actual Output:	Appropriate route based on group matching algorithm as it falls inside the elliptical part of preexisting route posted by the driver.
Result:	Test passed.

TEST CASE	
Name of the test case: Group/route matching	
Pre-Condition: Another user adds itself to a preexisting route.	
Description:	contains source and destination fields
Input:	Marker for source and destination placed far away.
Expected Output:	Display of his route on map with one marker as start node and other as end node but outside the elliptical boundary.
Actual Output:	Route does not lie within the elliptical boundary as the parameters fail to match with preexisting ride checked by group matching algorithm
Result:	Test passed.

TEST CASE	
Name of the test case: Search ride	
Pre-Condition: User search for a ride.	
Description:	Unaware whether their already exist a ride similar to his details.
Input:	Source and destination of his ride including time, date and days.
Expected Output:	If there exists a ride in which he can accommodate, then after search caption it displays the ride of other users matching his details.
Actual Output:	Join the ride as the parameters matches by the algorithm and becomes the member of that group.
Result:	Test passed.

TEST CASE	
Name of the test case: Message delivery	
Pre-Condition: messages delivered to group members only.	
Description:	Details of the group and all the users included in that group
Input:	Text message
Expected Output:	Messages delivered.
Actual Output:	Messages delivered to all the group members so as to be aware about any modifications made related to ride.
Result:	Test passed.

Chapter 8

Result and Analysis

8.1 Result and Analysis

The results from our program cannot be analyzed in a hypothetical manner because everything is definitive there are no variables to be considered and hence the result would be pretty much the same every time we calculate it.

The most fascinating and important part of our website is the group matching and route calculation algorithm. The route calculation algorithm runs in the background and cannot be seen by the user as it is a website we have followed the object oriented pattern of abstraction to the letter. This is something that we need to show to any person judging our work because they won't be able to contemplate it by merely running the program.

The group matching and re-routing can only be shown by adding people into a group and showing the routes in a chronological order. This is what we have made evident by placing these snapshots for the consideration of anyone who wants to see our work in actual motion without getting involved with the intrinsic details of the website.

Step 1: A new Ride is created or as we call it A Ride is posted. From North of Mumbai to the South of Mumbai

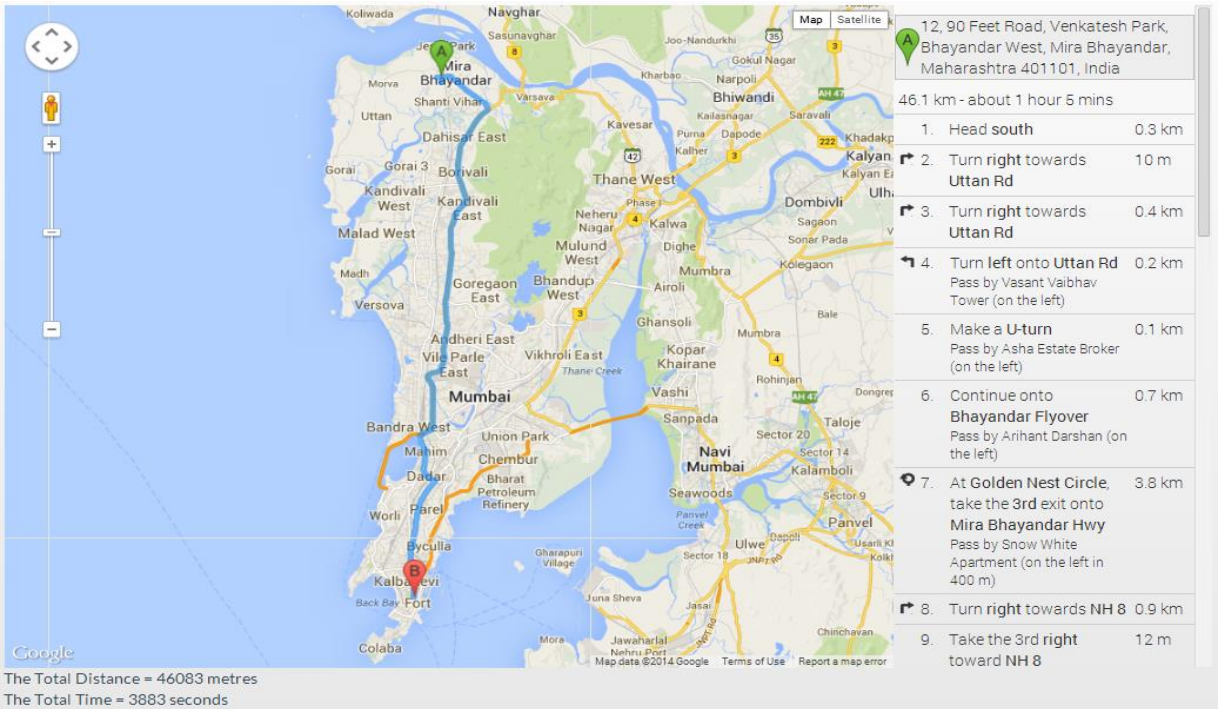


Fig 8.1 Maps with single route

Step 2: A new member joins the carpool ride in step1 with his/her start and stop in midway of that ride. It is evident from the ellipse that these members start and stop within the ellipse. Also the route is changed to fit in the markers B and C into the route. As the route fall under the elliptical part the programs add that route to the existing one and get assigned into the same group.

But in case if the parameters lie far away outside the elliptical region then it cannot be added into the same group as the parameters differ. Also if the route lies closet to start and end node within a small coverage of (e.g. 2km) still he will be assigned to the same group as it lies within the defined parameters

Even though the route alters in step 2 still it selects the best efficient and shortest path so that the cost constraint does not affect the users.

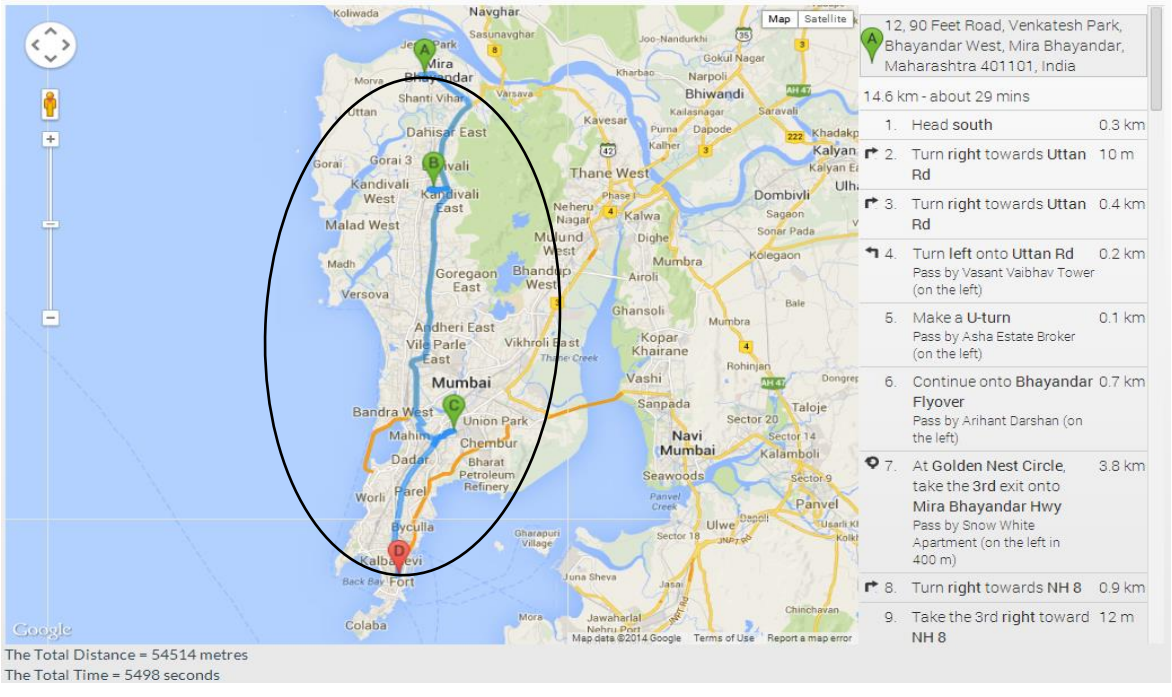


Fig 8.2 Maps with multiple routes (2 members)

Step 3: A new member joins the carpool ride in step 2 with their start and stop in midway of that ride which also lies within the ellipse. Also the route is changed to fit in the markers C and E into the route. As in the case of step 2 the points were B and C but now they have become B and D and the new coordinates for this member are C and E. Also the route has been re-routed to fit in the drop and pick-up points.

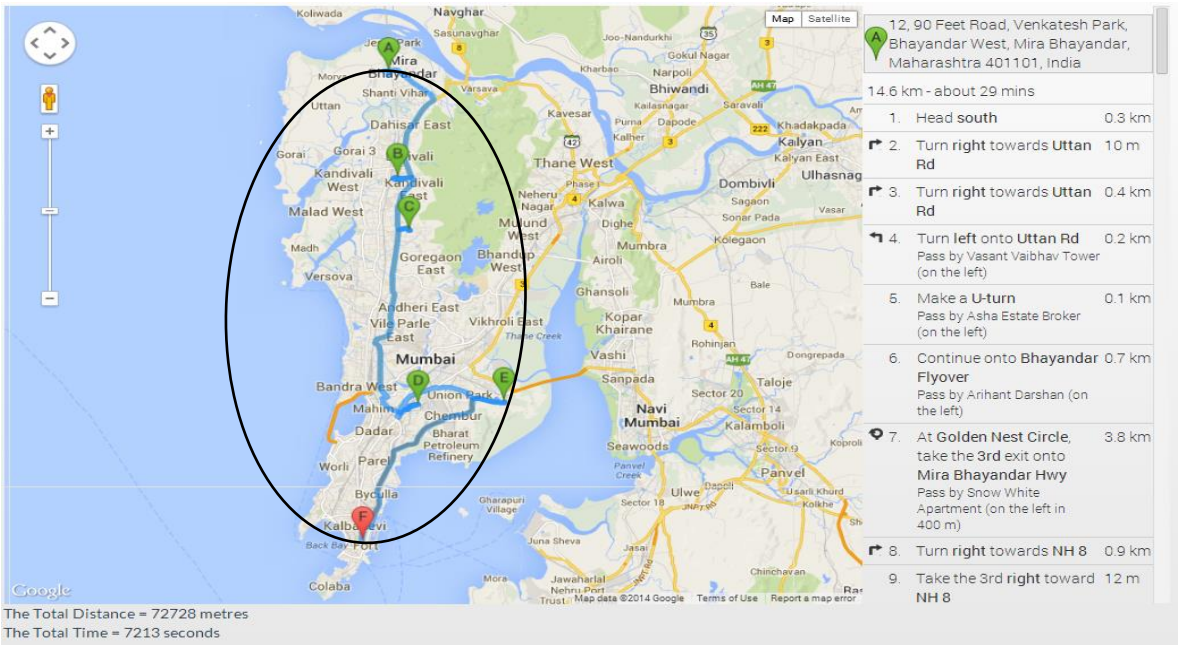


Fig 8.3 Maps with multiple routes (3 members)

Chapter 9

Conclusion and Future Scope

Conclusion and Future scope

We believe that we have made our project to a final stage where no more further development would be required. The only thing that would remain undone would be minor tweaks for instance someone could be made a member of two different rides with the same route but different operation says. The internal messaging could also be improved by using an actual Email like environment. The messaging app could also be improved by starting one to one communication between the users of the websites and also by maintaining a contact list.

As far as the social and graphical layout of the website is considered it cannot be made improved because we like to keep things simplified and our websites accomplishes that. The most enthused and useful future improvement that we could ponder over was a hopping scheme for fitting an individual into multiple rides for a single journey. The pulchritudinous of the group matching algorithm lies in the simple analogy that all that the algorithm requires are the coordinates of start, stop, and drop and pick up points.

This project can also be viewed as social networking site but its sole purpose is to formulate a regular travel plan for a lot of people driving onto the same route and to also deplete the adverse effects on environment that all this car pollution is causing. It is one of the most meaningful pursuits cause it’s speaks out one’s personal way to bring social work to the internet and in turn computer science.

References

[1] Jeffery Miller, (2007). "Algorithms and Data Structures for the Real-Time Processing of traffic data." *Dissertation Presented to the Faculty and the school of engineering University of southern California*

[2] ShangyaoYan, Chun-Ying Chen, (2011). "A model and a solution algorithm for the car pooling problem with pre-matching information. *"Department of Civil Engineering, National Central University*, Chungli 32001, Taiwan.

[3] Calvo R. W., Luigi, F. L., Haastrup, P., & Maniezzo, V. (2004). "A distributed geographic information system for the daily Car-Pooling Problem. *"Computers and Operations Research*, 31, 2263–2278.

[4] Coslovich, L., Pesenti, R., & Ukovich, W. (2006)."A two-phase insertion technique of unexpected customers for a dynamic dial-a-ride problem. *"European Journal of Operational Research*, 175, 1605–1615.

[5] Ferrari, E. & Manzini, R. & Pareschi, A. & Persona, A. & Regattieri, A. (2003). "The Car Pooling Problem: Heuristic Algorithms Based on Savings Functions". Journal of Advanced Transportation. 37(3): 243-272.

[6] R. Baldacci, V. Maniezzo and A. Mingozzi, "An Exact Method for the Car Pooling Problem Based on Lagrangian Column Generation," Operations Research, Vol. 53, No. 3, 2004, pp. 422-439. doi:10.1287/opre.1030.0106

[7] http://www.wikipedia.org/

Acknowledgement

We express our sincere gratitude to our project mentor Mrs. Jyoti Joglekar for the valuable guidance and advice. She inspired us greatly to work in this project. Her willingness to motivate us contributed tremendously to our project. Without her encouragement and guidance this project would not have materialized.

Besides, we would like to thank the authority of Mumbai University (MU) for providing us with a good environment and facilities to complete this project. Also, we would like to take this opportunity to thank our college Shah and Anchor Kutchhi Engineering College (SAKEC) for offering this subject.

Finally, an honorable mention goes to our families and friends for their continued support and understanding shown by them during the completion of this project. Without the help of all the people mentioned above and others who may not find mention herein above, we would have faced many difficulties while completing this project.

The guidance and support received from all the members who contributed to this project, was vital for the success of the project. We are grateful for their constant support and help.

Shail Shah	Smit Shah	Amay Yadav
BE-4-68	BE-4-69	BE-4-75

Place: Chembur

Date: 05/05/2014