Software Design Document – Final Draft
Online Ride Services

Team Members:
Mohammed Safian Syed - 50136688
Aditya Prasanna Handadi - 50135734
Santosh Kunwar – 50096459

CSCI 595: Research Literature & Techniques
Instructor: Dr. Hamid Azzo
Date: August 13, 2015

Department of Computer Science and Information Systems
Texas A & M University - Commerce

TABLE OF CONTENTS

1. Introduction.....3
   1.1 Motive.....3
   1.2 Objective.....3
   1.3 Facts & Literature Survey.....4
   1.4 Proposed Solution.....5
   1.5 Existing System.....6
   1.6 Proposed System.....6
   1.7 Modules.....6
2. Overall Architecture.....7
   2.1 Diagram.....7
3. Activity Diagram/Use Case Diagram.....8
   3.1 Activity Diagrams.....8
      3.1.1 Login Activity Diagram.....8
         3.1.1.1 Registration.....8
         3.1.1.2 Ride Login.....11
         3.1.1.3 Customer Login.....12
         3.1.1.4 Admin Login.....13
      3.1.2 Rider Activity Diagram.....14
      3.1.3 Customer Activity Diagram.....18
         3.1.3.1 Search A Ride.....20
         3.1.3.2 Advanced Booking.....22
         3.1.3.3 Emergency Notification.....23
      3.1.4 Admin Activity Diagram.....24
      3.1.5 Other Email Notifications.....25
4. Class Diagrams.....28
   4.1 Database ER Diagram.....28
   4.2 Class Diagram (Web Service).....29
      4.2.1 Registration.....29
      4.2.2 Customer Module.....30
      4.2.3 Ride Module.....31
      4.2.4 Class Diagram of Web Service Overall.....32
5. Requirements.....33
   5.1 Functional Requirements.....33
   5.2 System Requirements.....34
      5.2.1 Software Used.....34
      5.2.2 Software Requirements.....35
6. Conclusion.....35
7. Future Enhancements.....35
8. References.....36

Summary

Online Ride Services (ORS) is a Web Application, which provides users a centralized system to search and offer a ride from a particular source to destination. The intention of the project was to provide a user-friendly interface to search the routes, or book a desired ride for a specific route using the Google Maps API [1]. Users are also notified updates through emails. ORS provides users to choose the routes among the registered rides, and book rides that suit best for the users. In short, it is a carpooling [2] system made easy. This document features the motivation behind the project idea, the design plan, overall architecture, and a brief explanation of each feature ORS offers.

1. Introduction

1.1. Motive

Being students of Texas A&M University-Commerce, TX, finding rides is a drudgery unless one owns a vehicle. Apparently most part America faces the same problem of limited public transportation. A few transport facilities even if available are expensive. We have researched the International Student Association community page on Facebook™ and found to have more than 80 requests for rides since January 2015. Hence we thought of designing a system that can provide users to book and search rides easily, using the Google Maps™.

1.2 Objective

The objective of this project is to build a web-based ride sharing service which enables any registered user to do various activities like share a ride or search a ride based on desired route, date, and time. Our project is based on the concept of carpooling and provides benefit

to driver as well as passenger (commuter) by allowing them to travel together sharing driving times and expenses. The system will benefit commuters who can't drive, or can't afford to own a car personally and reduces traffic congestion and air pollution [3]. A user can share his ride and travel plans with others. Similarly, a user who is in need of a ride can see available options that will suit his/her travel plans. The Online Ride Service (ORS) will be a web application in Java integrated with Google Maps API™ , JavaScript, HTML and other frameworks.

1.3. FACTS & LITERATURE SURVEY

Studies shows that 45% of people living in this country doesn't have access to public transportation at all [4]. As a result, there has been a lot of increase in the number of cars people own. Increase in vehicles leads to traffic congestion and contributes to critical environmental issues like air pollution and noise pollution. Traffic congestion has cost Americans $124.00 billion a year and studies also show that 31% of Carbon-dioxide was produced by the automobile itself [5].

Carpooling is cost friendly and environment friendly and the concept of carpooling has been around for a while and been implemented in different ways. Uber™ [6] is one example of famous ride sharing (car pooling) company today. Using Uber app, consumer can submit a trip request, which is then transferred to ride sharing drivers. Although systems are being existing, they are not prevalent around most part of our country that are away from major cities. And most systems are not freeware and open source. These systems are very essential for general public to make transportation more efficient. With the help of Google Maps and

Java open source systems, a very user-friendly interface can be created for a Car Pooling System.

1.4. PROPOSED SOLUTION

When we look for a solution for the problems mentioned above, one solution would be to wait till public transportation reaches all part of the country, which is irrational. The other solution is to carpool. While being in a place away from the city, we thought that carpooling is a convenient and timesaving system for the problems we all are facing. With the help of carpooling, we can share ride with other people. It could provide solution to these traffic and environmental problems we are facing today. “Passengers pay less, drivers make more, we have fewer rides and its better for the environment” as said by San Francisco Assemblyman Phil Tang [7]. This is actually what motivated us to develop Online Ride Services.

1.5. EXISTING SYSTEM

- Existing systems are not prevalent around most part of our country that are away from major cities.
- Most systems are not freeware and open source.
- Uber™ and Lyft™ are example of existing ideas that is based on ride sharing. These systems are primarily cab sharing services. They are limited to major cities like New York City, Washington D.C., and Los Angeles.
- Uber™ is expensive and not accessible without a smartphone and credit card and most of the people can use the service. [8]

1.6. PROPOSED SYSTEM

- The proposed system is very essential for general public to make transportation more efficient in rural areas.
- Our system is not only helpful for finding rides in remote towns for people who cannot afford to buy a vehicle, but also helps in bigger cities by reducing traffic, resulting in less travel time, and better for environment.
- Making a simple user-friendly system for this idea of sharing rides would make a very convenient way of finding rides.
- Cheaper than alternatives.
- No need of a smartphone and a credit card.
- With the help of Google Maps and Java open source systems, a very user-friendly interface can be created for a Car Pooling System.
-

1.7. MODULES

Our proposed consist of several modules, which offer the following features:

- Registration – *for new users.*
- Login – *existing users authenticate.*
- Share A Ride – *if you want to share your trip.*
- Search A Ride – *if you are looking for a ride.*
- Advance Booking – *if you were searching a ride but didn't find an appropriate one.*
- Activity – *if you want to view your upcoming ride information*

- Safety and Security Alert – *if you want to notify an emergency to the admin.*
- Rate Quote – *how much your trip cost if you are booking a ride.*
- Admin – *for system administration activities.*
- Notifications – *email notification to keep rider and customer in sync.*

2. Overall Architecture

An ideal mobile system majorly has a three-tiered architecture, a front-end, a web service, and a database. Front end is the user interface build basically using HTML, with functionalities mainly included in JavaScript, and embedding Google Maps API into these JavaScript files.

2.1 Diagram

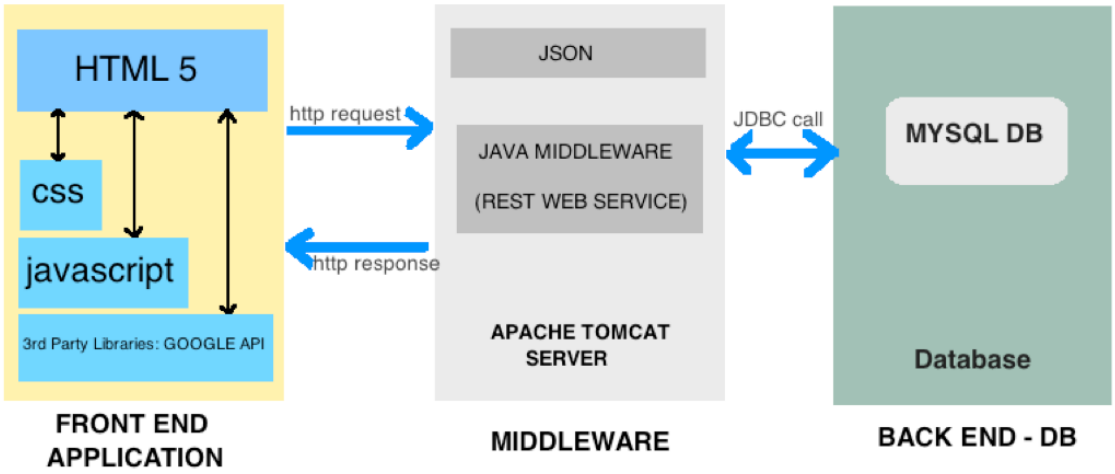

Figure: Architecture

Google Map's GeoLocation API locates the user's current location. Direction Service API is used to draw the route from source to destination on the map. Distance Matrix API can calculate distance between 2 co-ordinates of Google Map APIs.

Combining these 3 services from Google Maps, a map based user interface is provided to the ease of user.

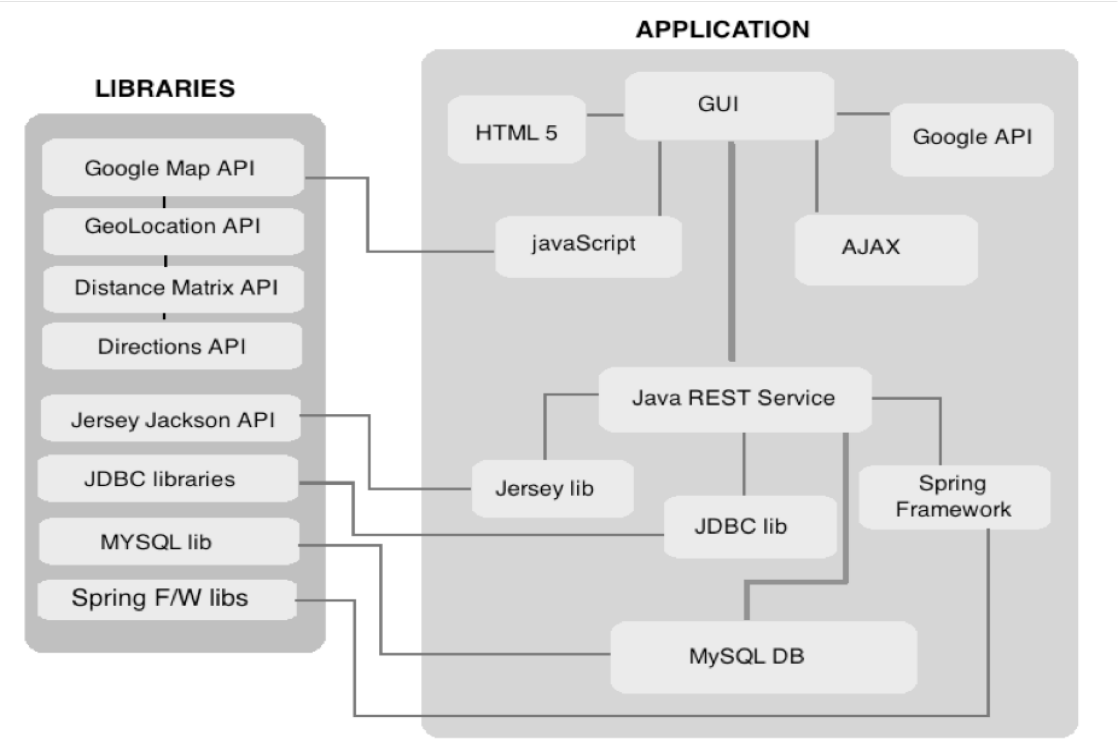

Figure: Overall System Architecture

3. Activity Diagram / Use Case Diagram

3.1 Activities Diagrams

3.1.1 Login Activity Diagram

The following diagram describes the login activity:

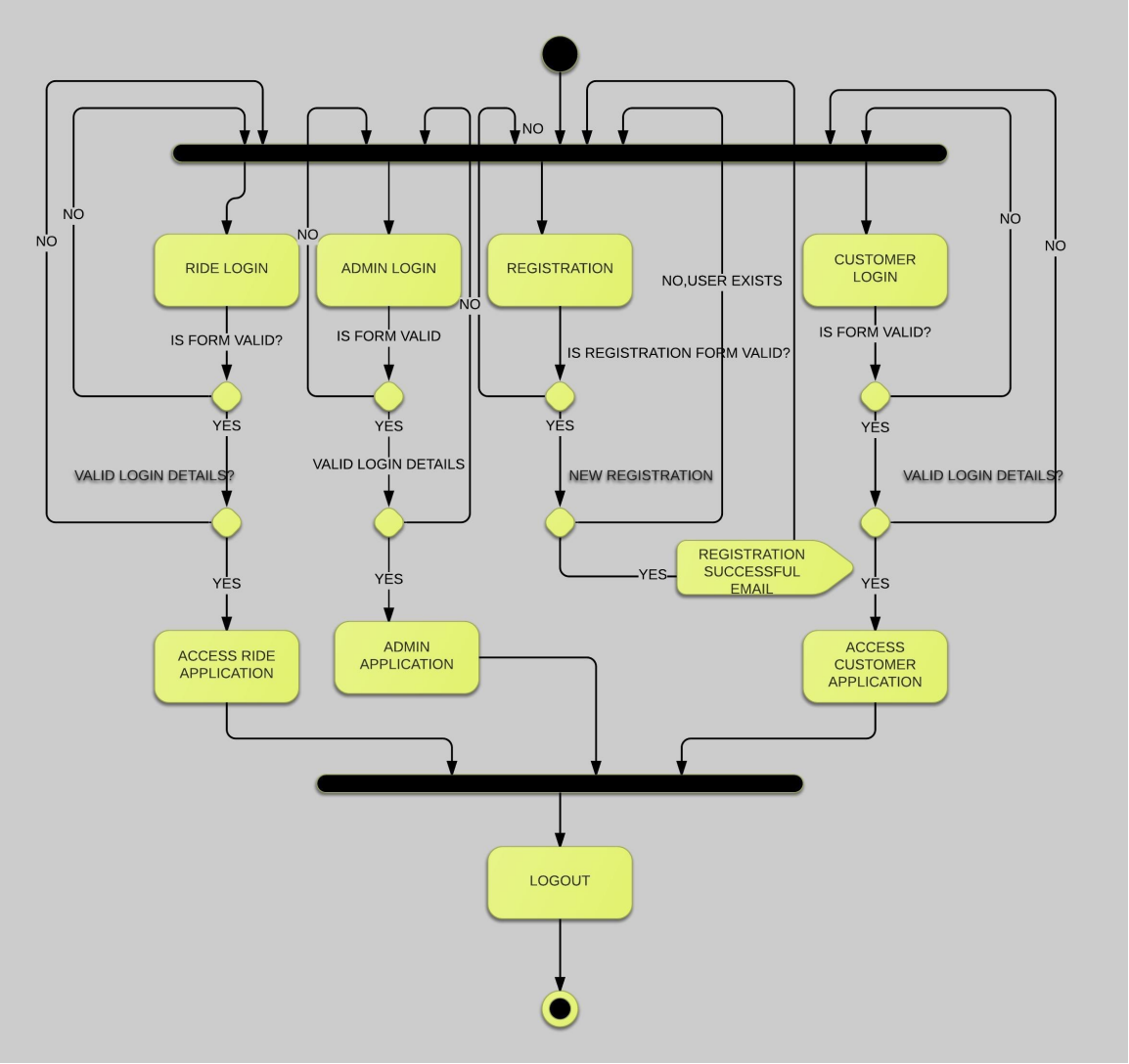

Figure: Login Activity Diagram

Login activity can again be described in four different sub-activities:

- **Registration:** Any new user who wishes to become a member and use the services offered must register first. The information entered by the user will be validated and only if the user doesn't already exist, the user is registered and confirmation email is sent.

ScreenShots:

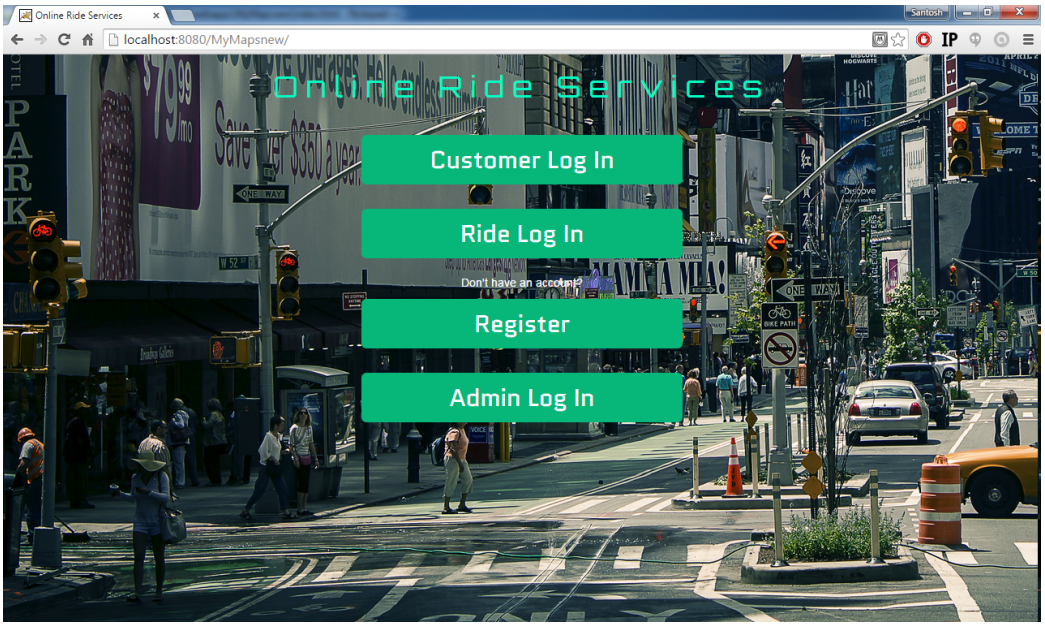

Figure: Home Page

The main home page will be displayed when a user enters the website. The user will have an option to either login or register as a new user. Registration is an important functionality in ORS system. Any user who wishes to become a member and use the services offered must register first. The purpose of the registration is to create a user account so that they can sign-in to the system using registered username and password. Email ID will be the username.

This page is implemented with simple HTML forms, and Bootstrap, CSS for better user interface.

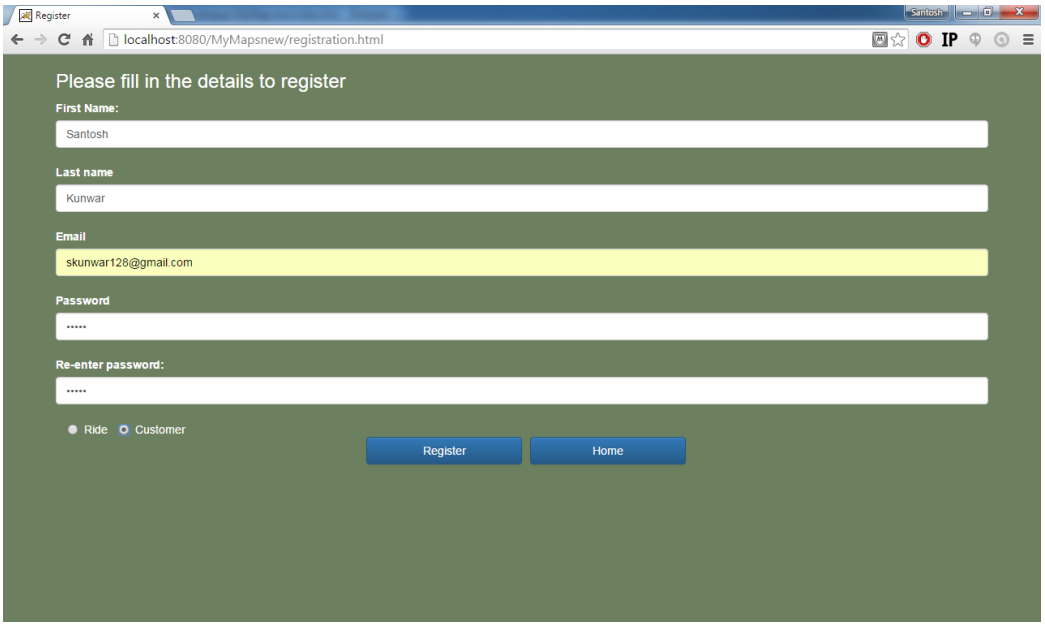

Figure: Registration Page

Registration will again have 2 selections; one for users who can share rides, which we would refer, as "Ride" users and another for users who need the ride called "Customer". As of now, the system is open as a common registration for using both modules. Once successfully registered, an email notification is sent:

Email:

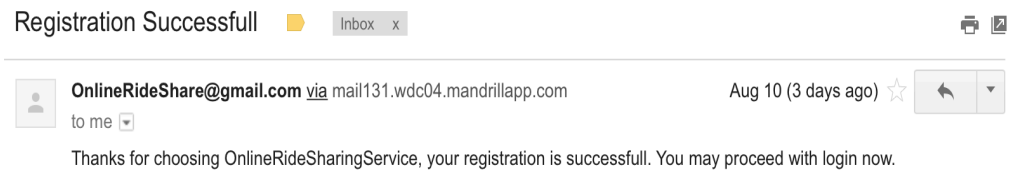

Figure: Registration Confirmation

- **Ride Login:** After registration, a user who wishes to share a ride can proceed to ride login using his/her login credentials. If the credentials don't match, user will

be denied to access the application. If the user provides correct login credentials and after validation, he/she will be able to access ride application where the user can share a ride based on a desired route, date and timing and perform various other activities as well.

Screenshots:

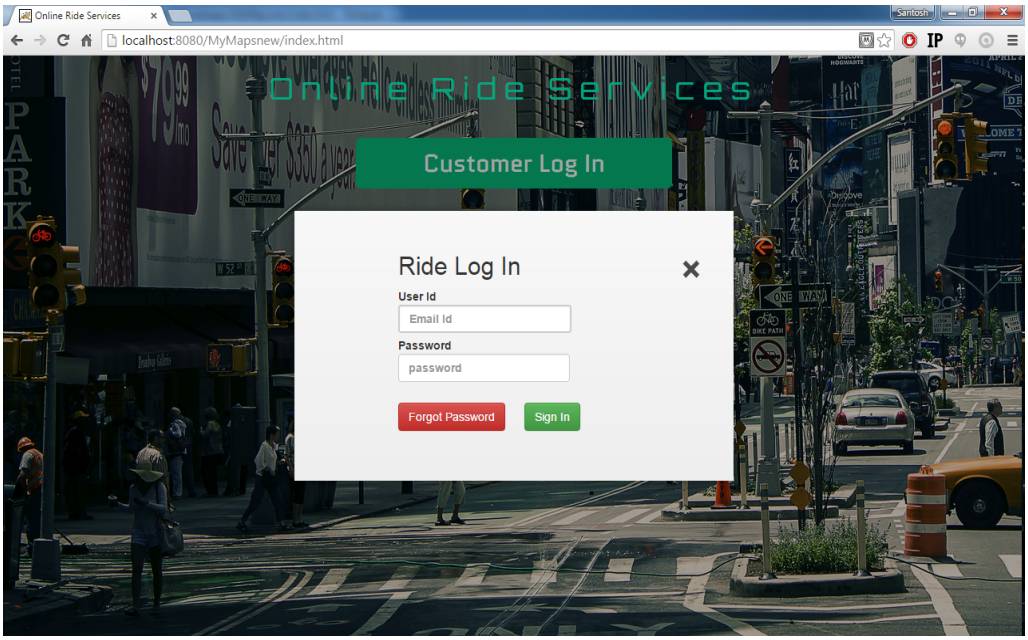

Figure: Ride Log In

This page is also enhanced using Bootstrap, CSS for better user interface.

- **Customer Login:** After successful registration, a user who wishes to search a ride can proceed to customer login using his/her login credentials. Once login is successful, he/she will be able to access customer application where the user can search a ride based on a desired route, date and timing and perform various other activities like booking ride and advance booking.

Screenshots:

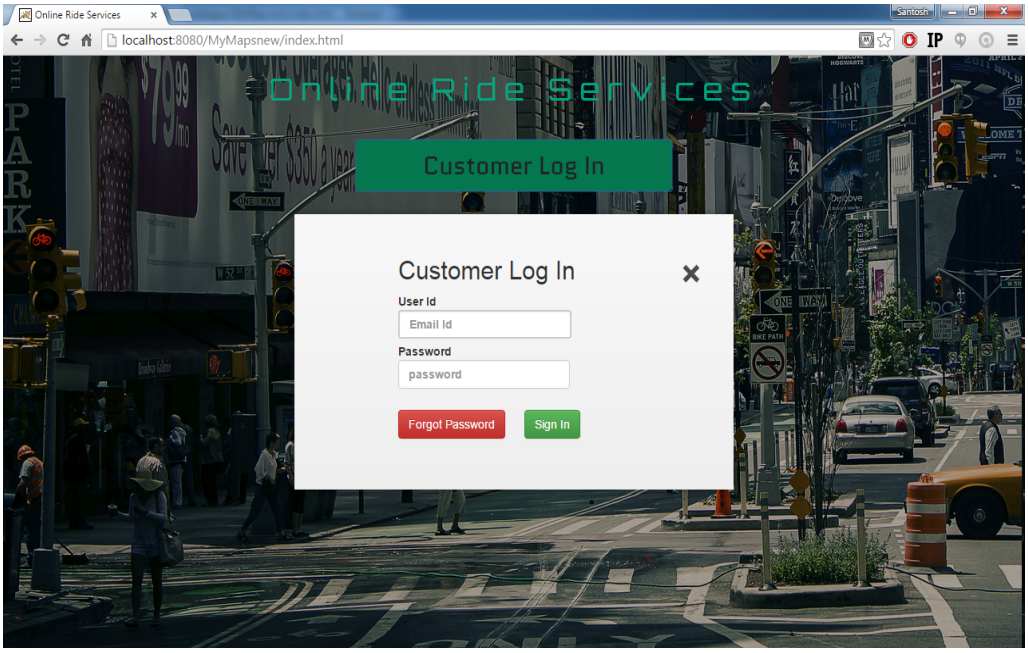

Figure: Customer Login

- **Admin Login:** Similarly Admin Login is also created.

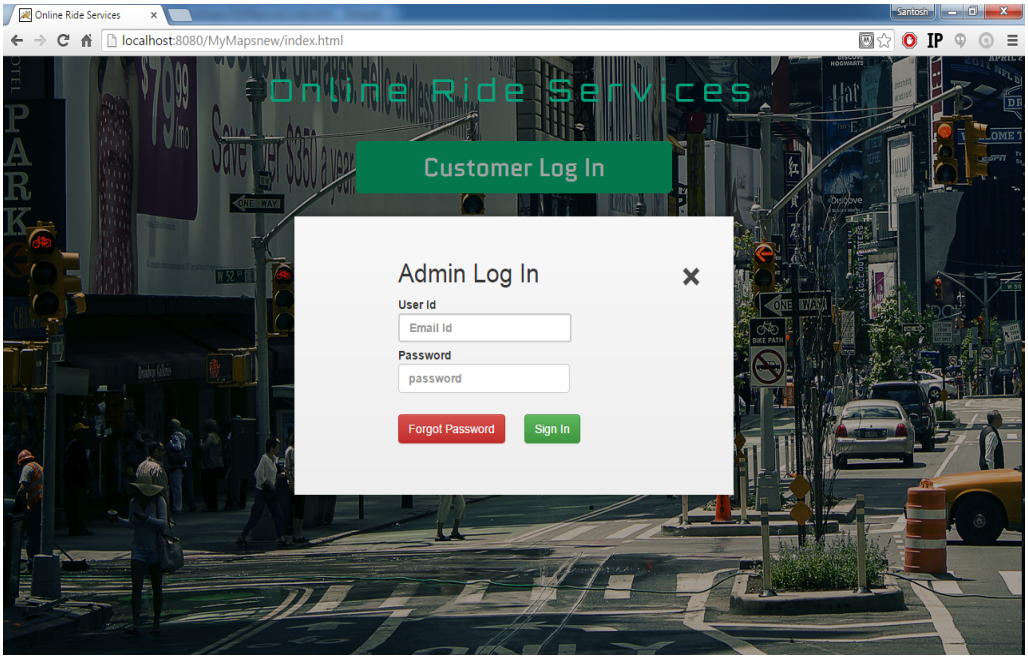

Figure: Admin Login

3.1.2: Rider Activity Diagram

The following diagram describes the rider activity for the Rider Module:

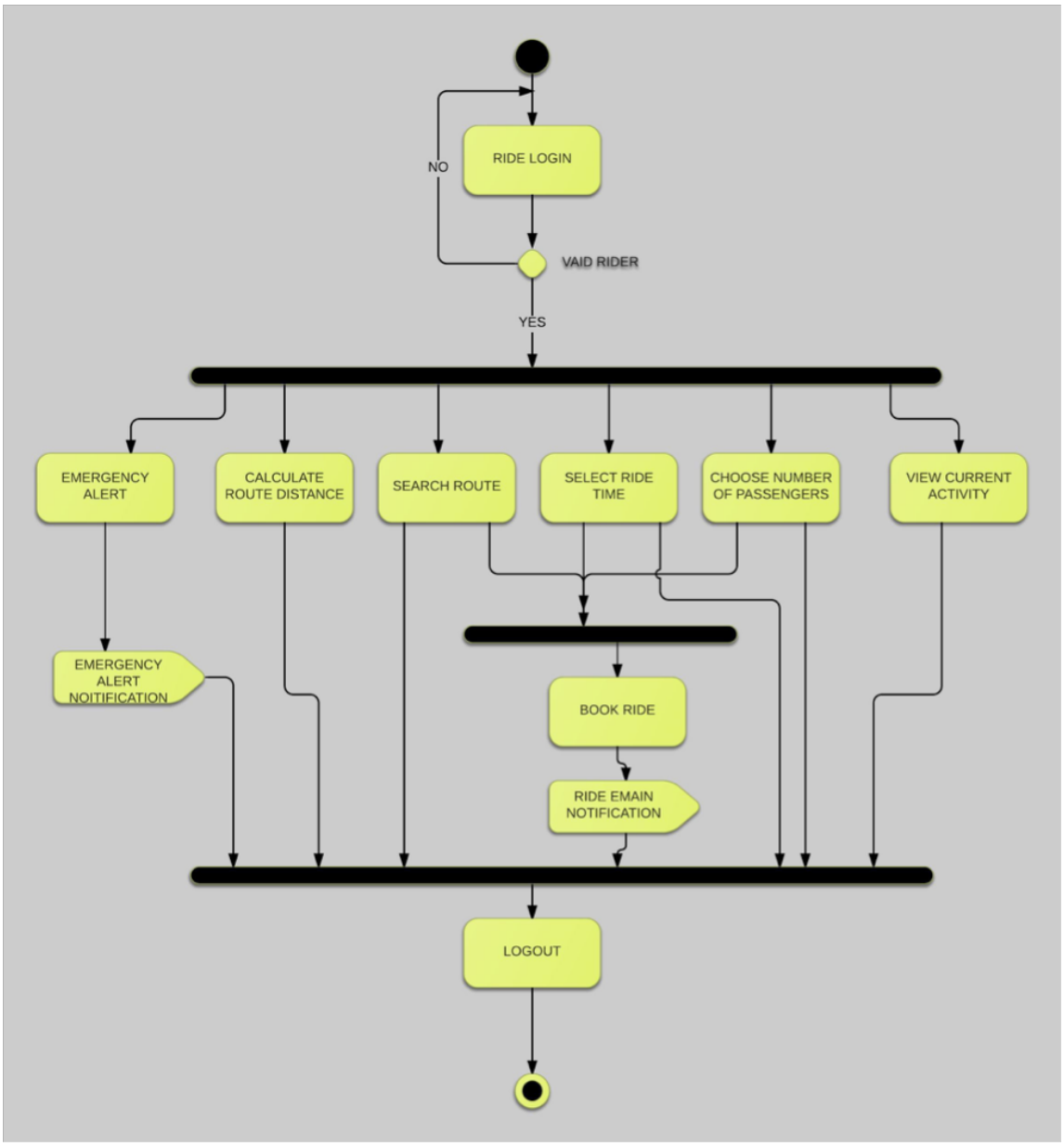

Figure 9: Rider Activity Diagram

Rider activity diagram shows how an existing user can login into the application to share a ride using Ride Login. If the provided credentials are valid, the user will be allowed to access the application and will be allowed to perform the following activities:

- **Custom Route:** The user will be allowed to map a route based on desired source and destination of travel.
- **Add Waypoint:** The user can add up to 8 drop-off points between their source and destination, and create his custom route to his destination. If not, by default a default closest route will be drawn.
- **Select Ride Time:** This allows the user to select the date and time of travel.
- **Choose Number of Passengers:** User can specify how many seats are available.
- **Book Ride (Save):** After performing above-mentioned activities, user will be able to save the ride he wants to share and a confirmation email will be sent.
- **Calculate Route Distance:** This activity allows the user to see the distance he/she will be travelling.
- **View Current Activity:** User can view all his/her upcoming information regarding the rides he/she has share.
- **Emergency Alert:** User can send emergency notification to the admin if needed.

Users can logout whenever he wishes to do so.

Screenshots:

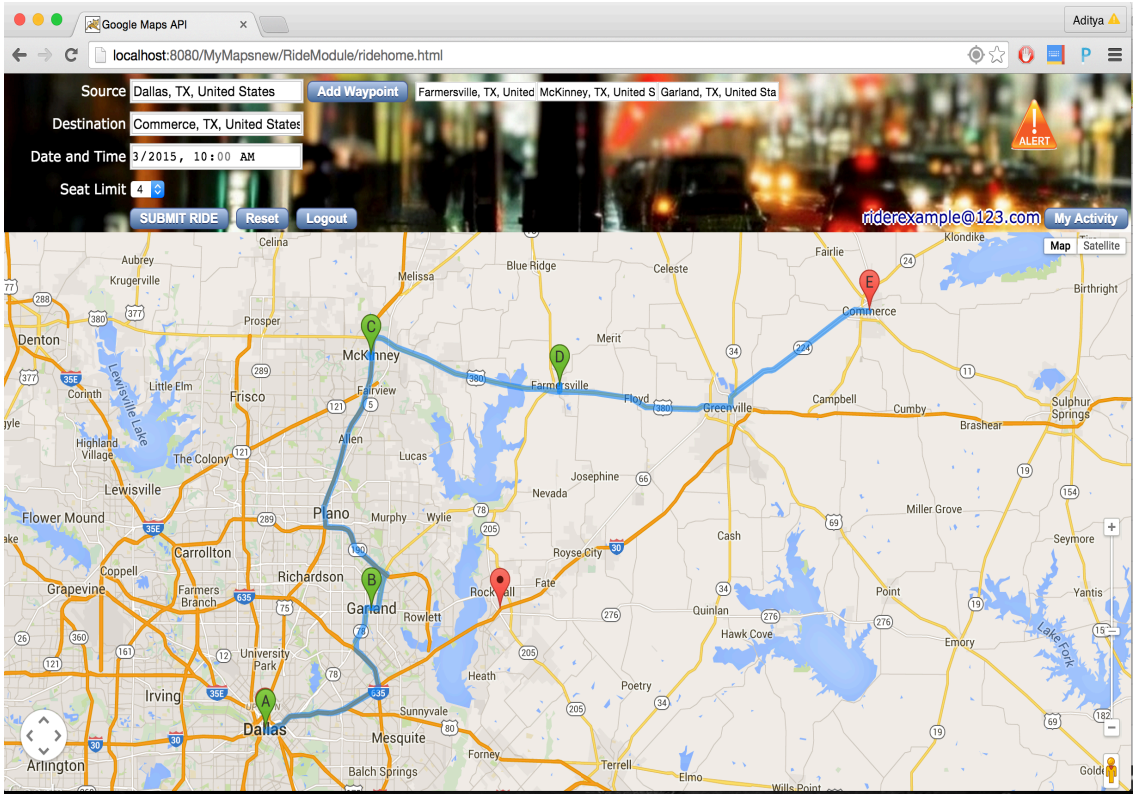

Figure: Rider Page

ShareARide allows a registered user having a car to share a ride. For that, he/she has to provide information regarding ride origin and destination with the date, timing, and available seats. Once all the information is shared and confirmed, a view of the route on Google map is shown along with the origin, waypoints, and destination.

This page is implemented using JavaScript, GoogleMap API's Geolocation API, Direction Service API, and Ajax™ for user interface showing map and routes, all at same time. The page interacts with the server using *Ajax* call for submitting the data to the server in the form of JSON. The Java web service uses Jackson- Jersey API to convert this JSON to POJO instances containing corresponding to JSON data. And this in turn is saved to the database using JDBC.

The Geolocation API first locates the user first. When he adds source and waypoints and destination, each of these fields are treated as a new Google Map marker objects, and a custom route is drawn on the map using the Direction Service API from Google, according to the waypoints provided by the user. The required data is sent to the server in the form of JSON object, and the server parses the JSON data and stores in the DB. This Asynchronous client server communication helps the overall front-end system perform faster than the Java's built-in Web App Structures.

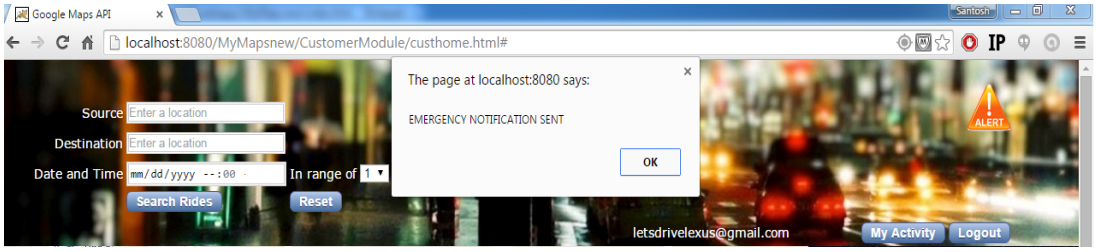

Figure: Alert

In case of emergency, the rider can notify the admin of the situation by pressing the Alert button. Current location will be included in the email.

3.1.3: Customer Activity Diagram

The following diagram describes the customer activity:

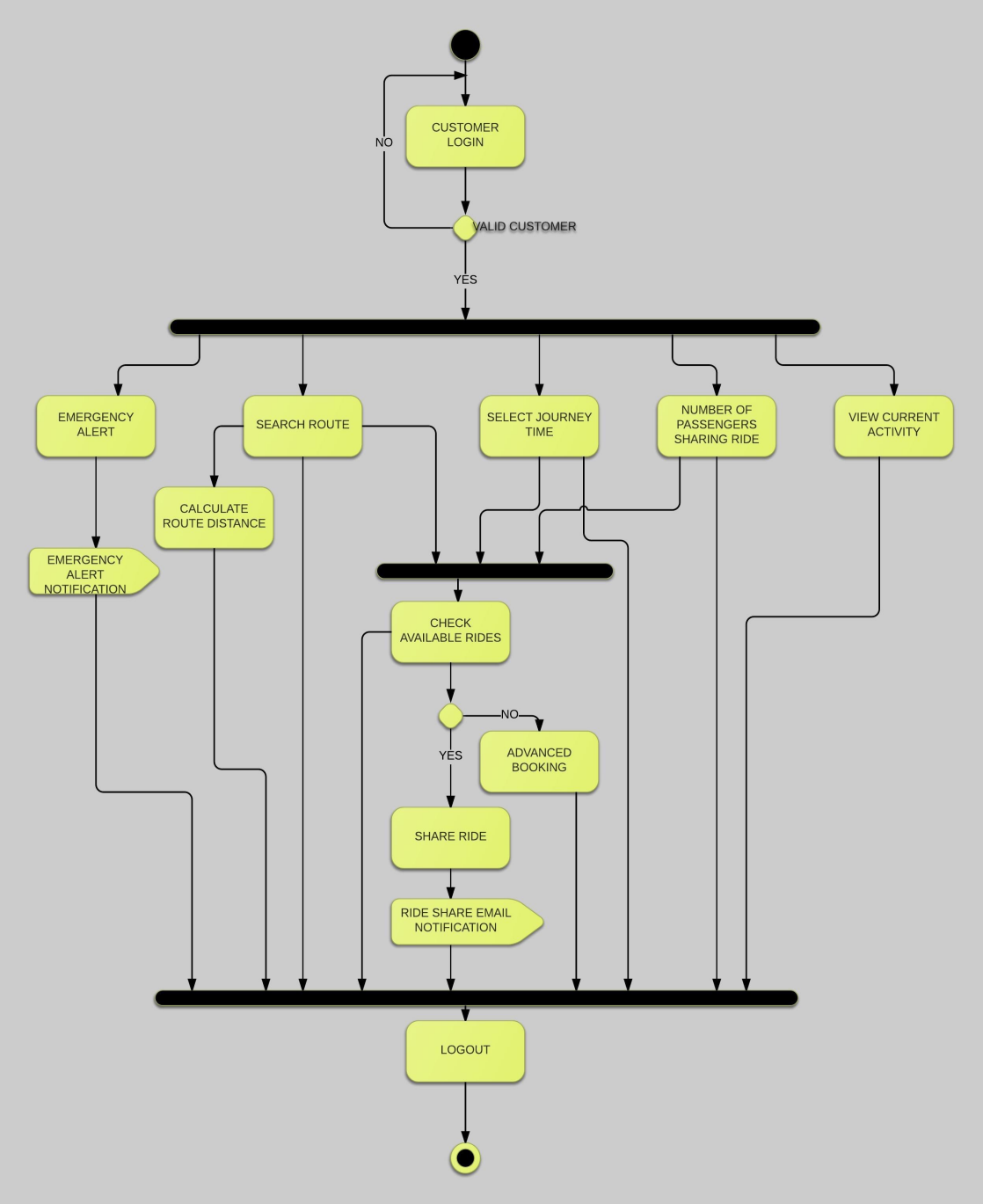

Figure 11: Customer Activity Diagram

Customer activity diagram shows how an existing user can login into the application to search for a ride using Customer Login. If the provided credentials are not valid, the user will be denied access to the application. If the provided credentials are valid, the user will be allowed to access the application and will be allowed to perform the following customer activities:

- **Search Route:** The user will be allowed to search a route based on desired source and destination of travel.
- **Select Journey Time:** User needs to select the date and time while searching for available rides.
- **Check Available Rides:** After doing above-mentioned activities, user will be able to search for available rides matching his/her criteria.
- **Book Ride:** If any rides are available matching the users desired route, date and time, user can book the ride of his/her choice. After successful booking, a confirmation email will be sent.
- **Advanced Booking:** User can use this feature to post his ride needs in advance if the user cannot find any rides at the moment. Later if any ride is shared that would match the advance request, then that particular user will be alerted so that he could login and book the ride.
- **Calculate Route Distance:** This activity allows the user to calculate the distance he/she will be travelling.
- **View Current Activity:** User can view all his/her upcoming information regarding the rides he/she has booked.
- **Emergency Alert:** User can send emergency notification to the admin if needed.

Once the user logs out, customer activity is completed.

Screenshot:

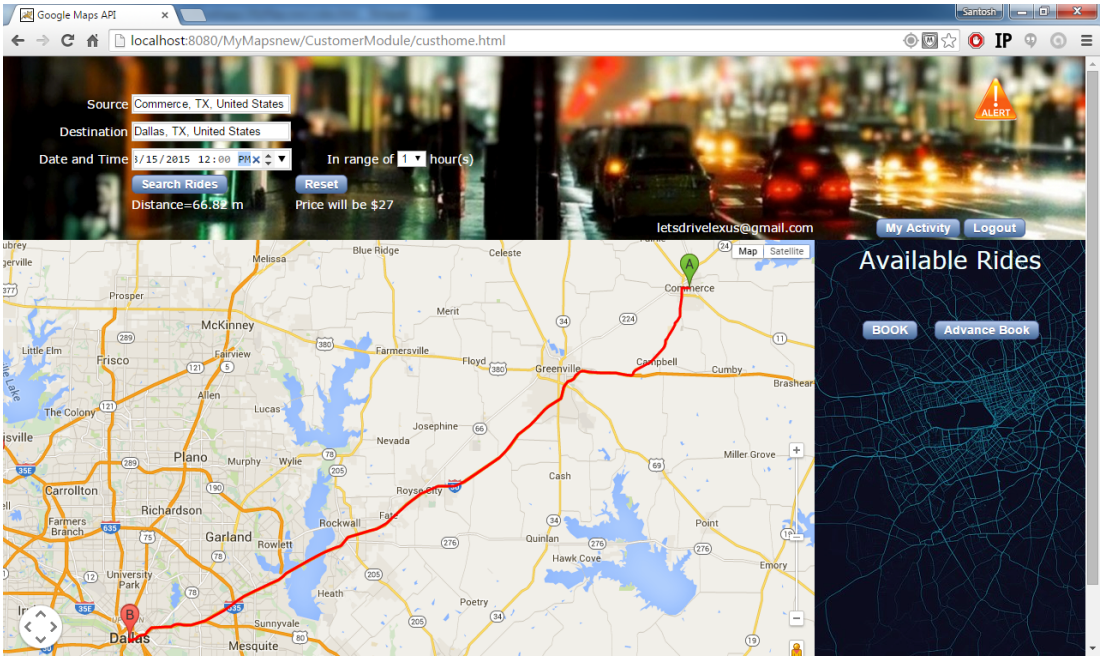

Figure: Customer Page

Search A Ride:

SearchARide functionality is dependent on the information stored in the database. A user who needs a ride can search the site for a suitable ride by providing the information regarding his origin and destination places and timeframe when he needs it.

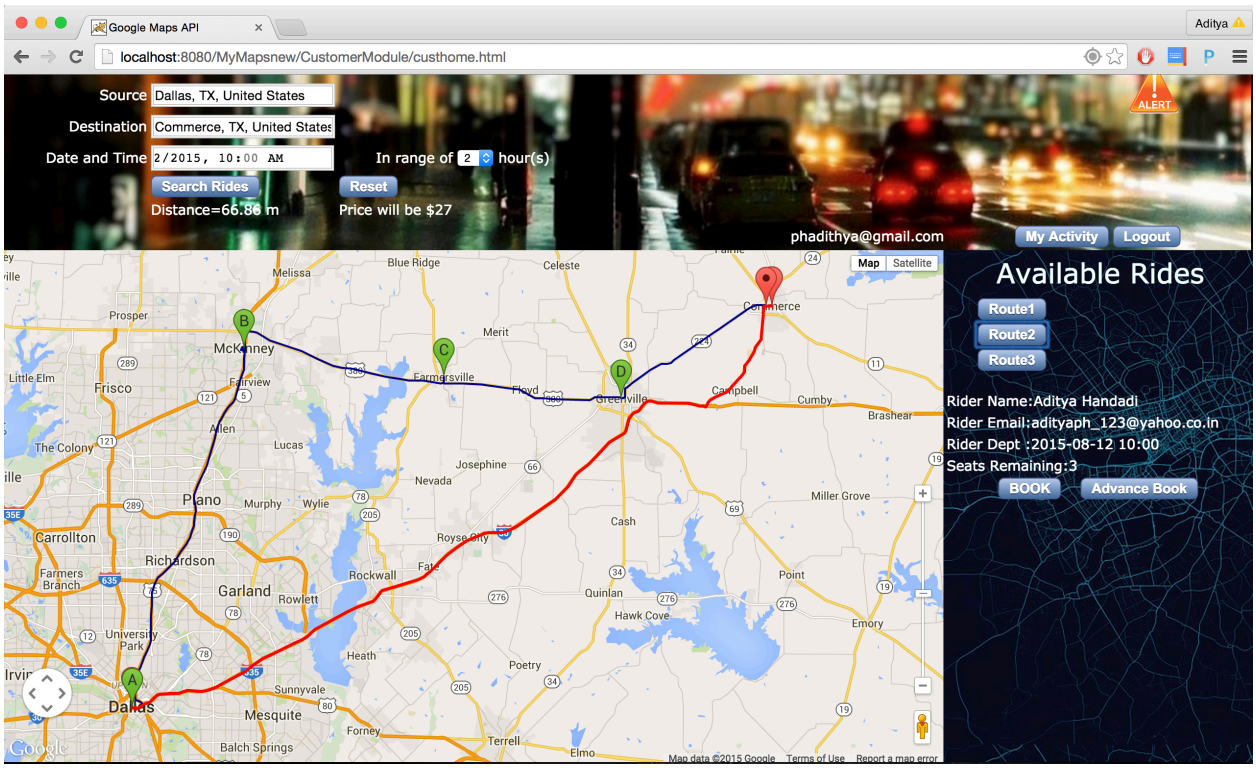

Figure: Available Rides

After the customer searches for ride, all the available rides matching the search criteria will be displayed. 'N' number of buttons means there are 'N' available rides. Customer can view the rider information and route view on Google map by clicking respective available route button. The user can choose any of the ride showed in the search result and book that particular ride.

The functionality works similar to Ride Module, but the client server communication works backwards. The server sends all the available data relevant to his ride as a JSON object, which is parsed by the front end JavaScript. The parsed JSON data contains necessary information of the rider's route. These set of routes are each compared with rider's route, if they are on the way or not, using the Google Map polyline feature.

Customer will be able to see all the riders travelling along his destination for his timeframe.

Advanced Booking:

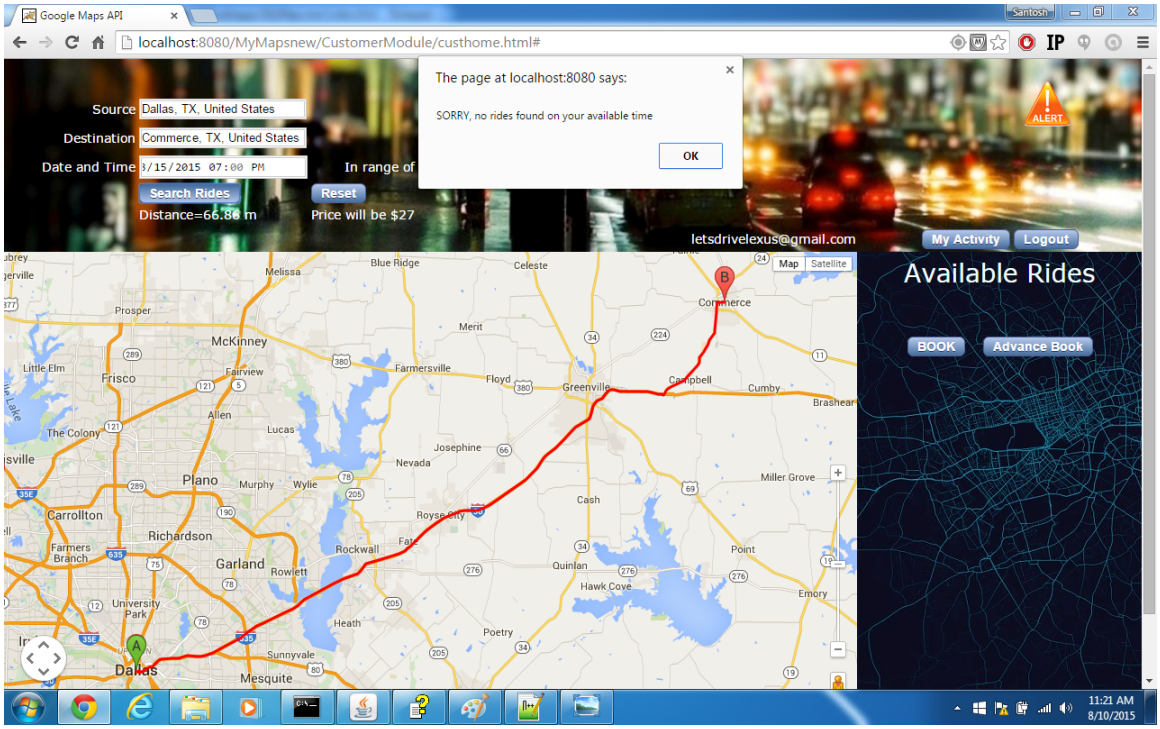

Figure: Advance Booking

User can use this feature to post his/her ride needs in advance if the user cannot find any rides instantly. Later if any ride is shared that would match the advance request, then that particular user will be alerted so that he could login and book the ride.

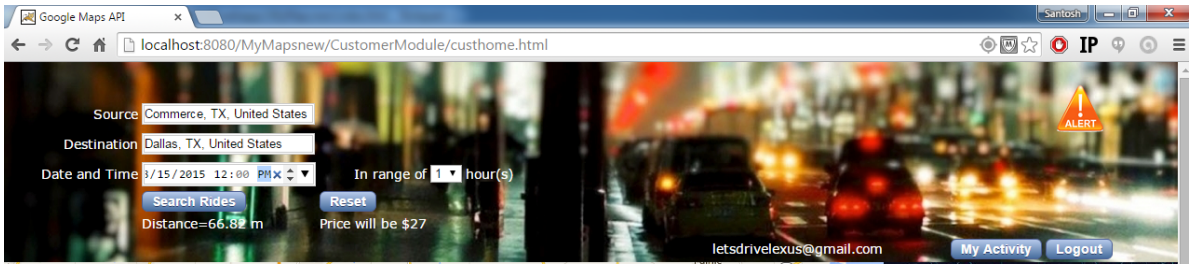

Figure: Rate Quote

This functionality displays trip cost to the user, which is automatically calculated by the system based on the admin's business rules and user's pick-up and drop-off points.

Emergency Notification:

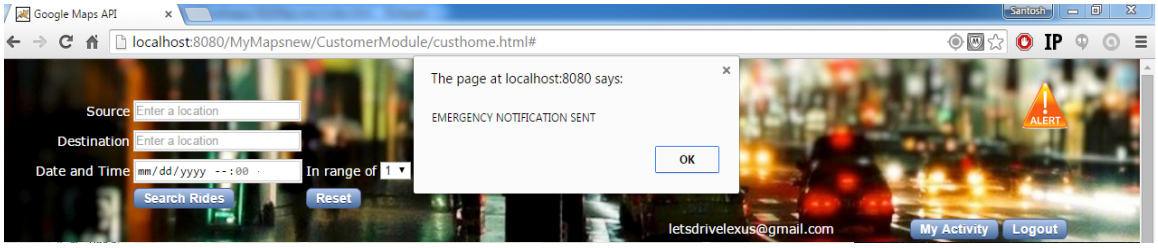

Figure: Alert

For safety and security, there is one Alert feature. When user wants to report an emergency, clicking the alert symbol will notify the admin over email immediately, giving the co-ordinates in the message body, from where the emergency was reported.

Email Notification:

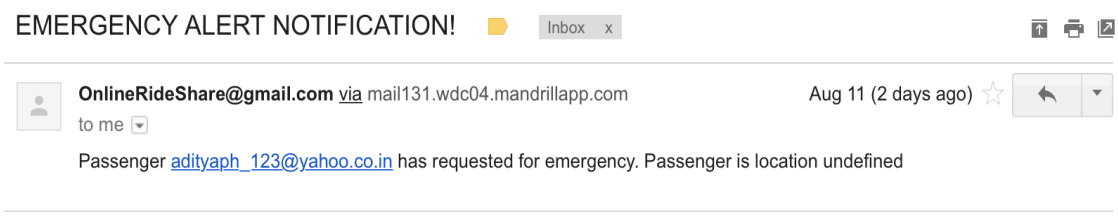

Figure: Email Notification

3.1.4: Admin Activity Diagram:

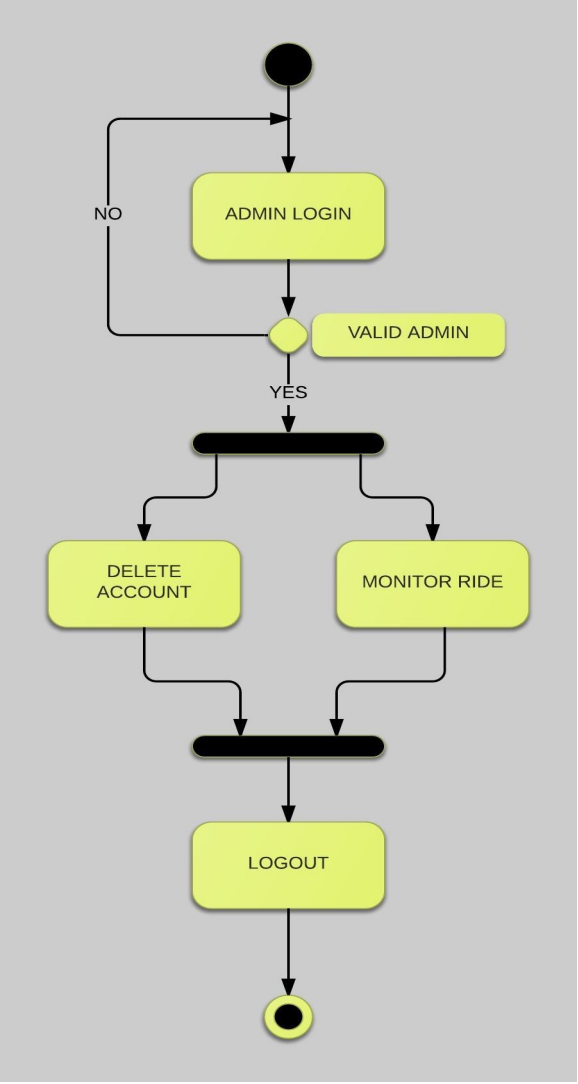

Figure 19: Admin Activity Diagram

The Admin Login redirects to another page to show the list of all rider activities so far as below. Only he has privileges to cancel the ride by entering the route id. A

similar functionality without delete ride feature is available for Riders and Customers to view their booking activity.

Screenshot

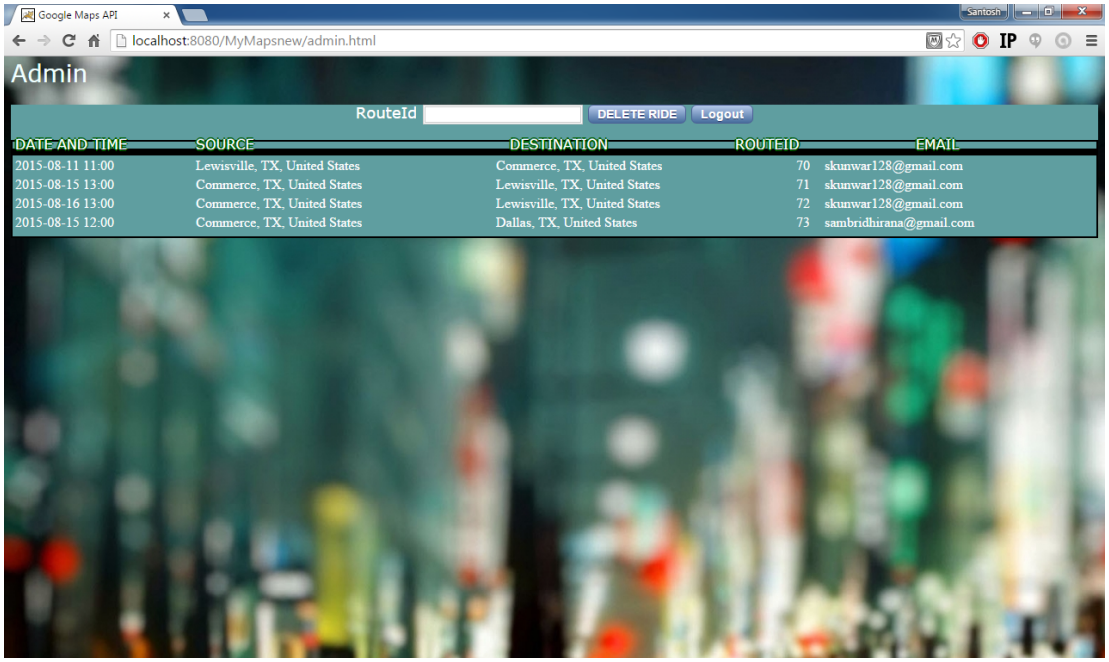

Figure: Admin Page

3.1.5: Other Email Notifications

When a rider registers a ride:

Figure: Email Notification

When a customer books a ride, the customer is notified.

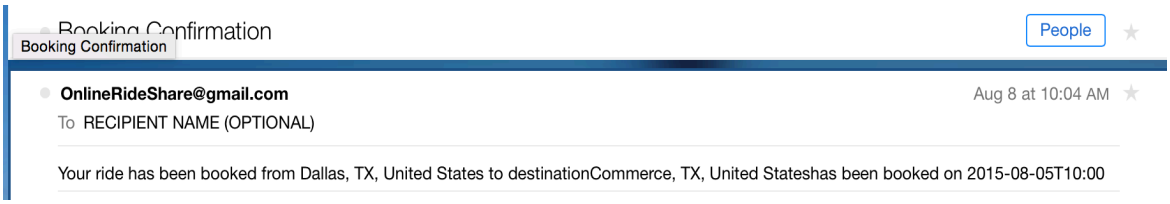

Figure: Email Notification

And the rider is also notified, when customer books a ride:

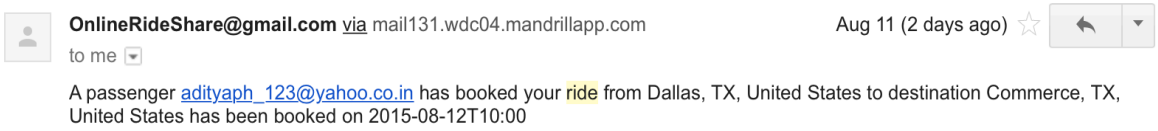

Figure: Email Notification

When a customer does advanced booking and if a ride is available after sometime:

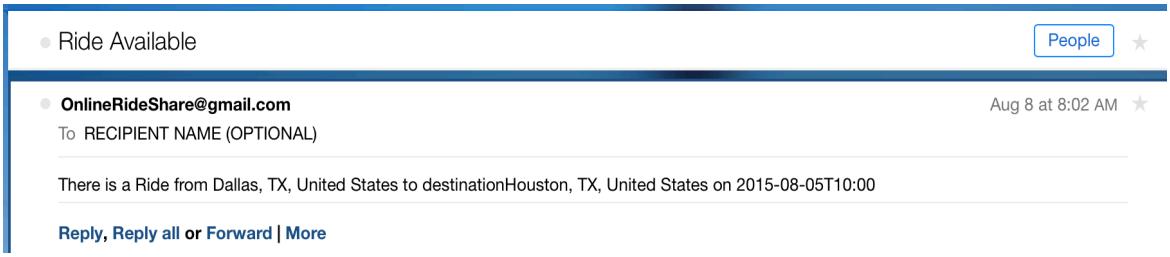

Figure: Email Notification

3.2 Use Case Diagram

There are various actions in our application that can be performed by three actors – rider, customer and admin. Only login action is common to rider, customer and admin. Below diagrams depicts the different actions by three users:

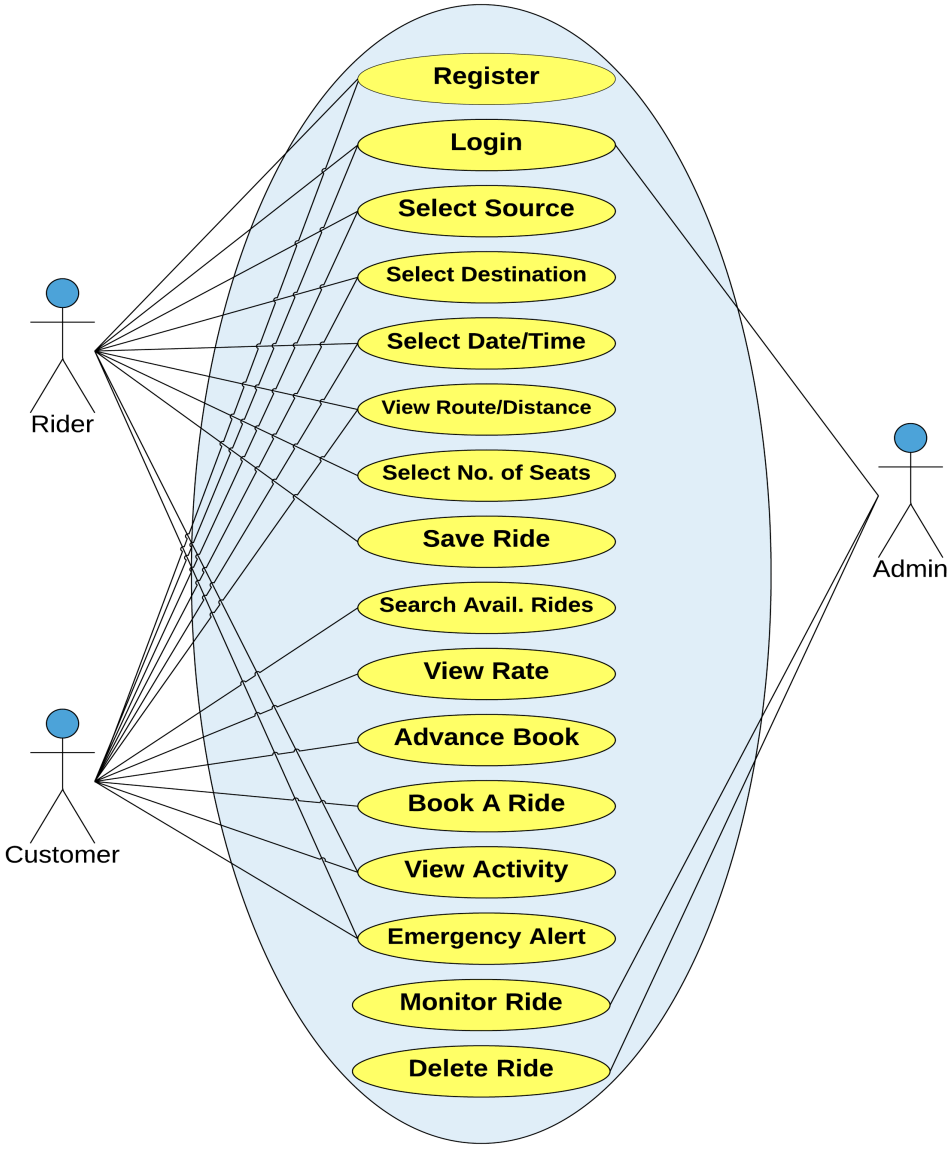

Figure: Use Case Diagram

4. Class Diagrams

4.1. Database ER Diagram:

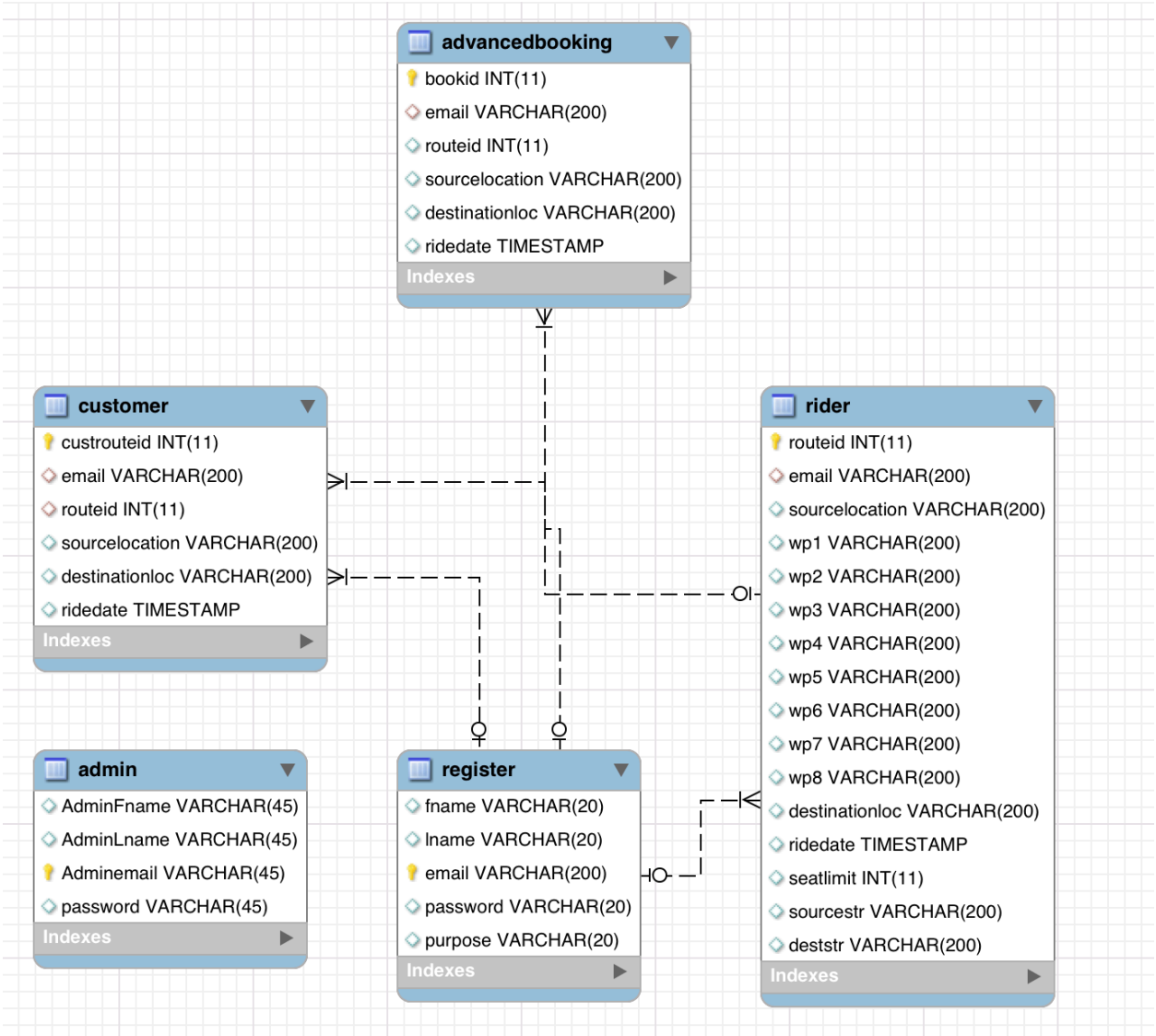

Figure: Database ER diagram

Register table contains all the registered user data. The “email” attribute is a NOT NULL primary key, which is a foreign key for other tables “rider” and “customer”. Rider table contains all the parameters to save a ride, and its route. Customer table is designed to contain information of all the booking information, if a customer after searching a ride,

books one. Advanced Booking contains information to book a ride in advance. For all three later tables, primary key is an auto-incremented DB value.

4.2. Class Diagram (Web Service)

Registration:

Below is the class diagram for Registration Module. It contains 3 classes, model – *RegisterVO.java*, corresponds to Register form of our register page, Controller – *RegisterService.java*, which interacts with Dao – RegisterDao to perform DB operations.

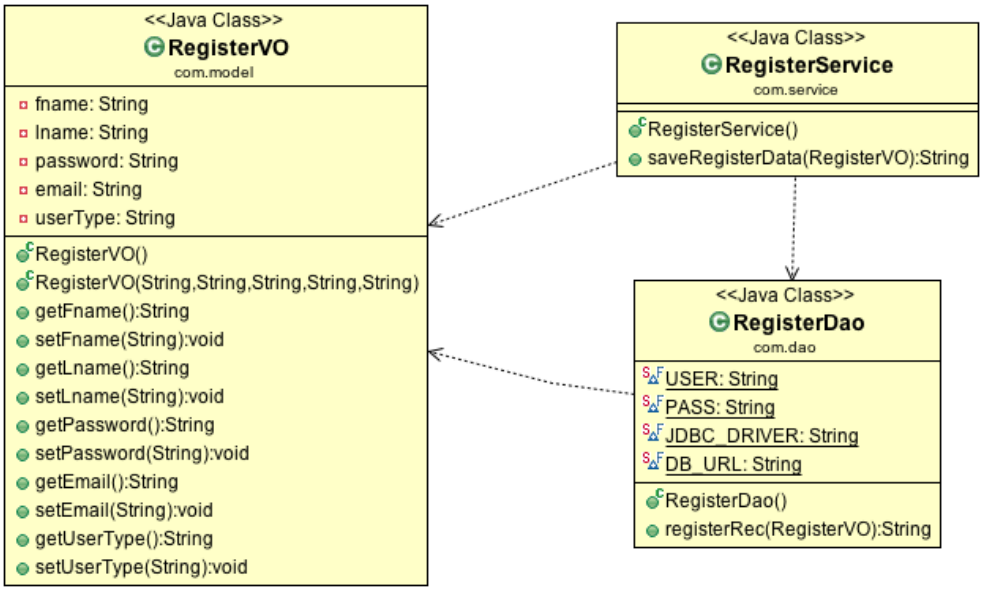

Figure: Class Diagram (register)

Customer Module:

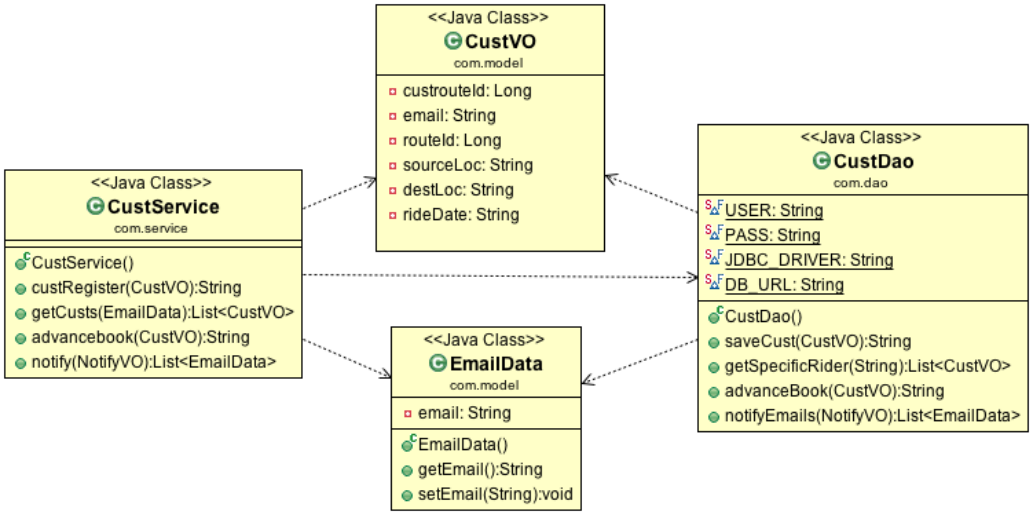

Figure: Class Diagram (customer)

Above is the class diagram for Customer Module. This contains, model - *CustVO.java*, service – *CustService.java*, and *CustDao.java* as its Dao layer class. *EmailData.java* is also another model used, used to instantiate a variable in *CustVO.java*. *CustVO.java* contains all the values when the JSON data is sent to the server. Service class uses this POJO to send to DAO layer, to perform DB operations using JDBC.

Ride Module:

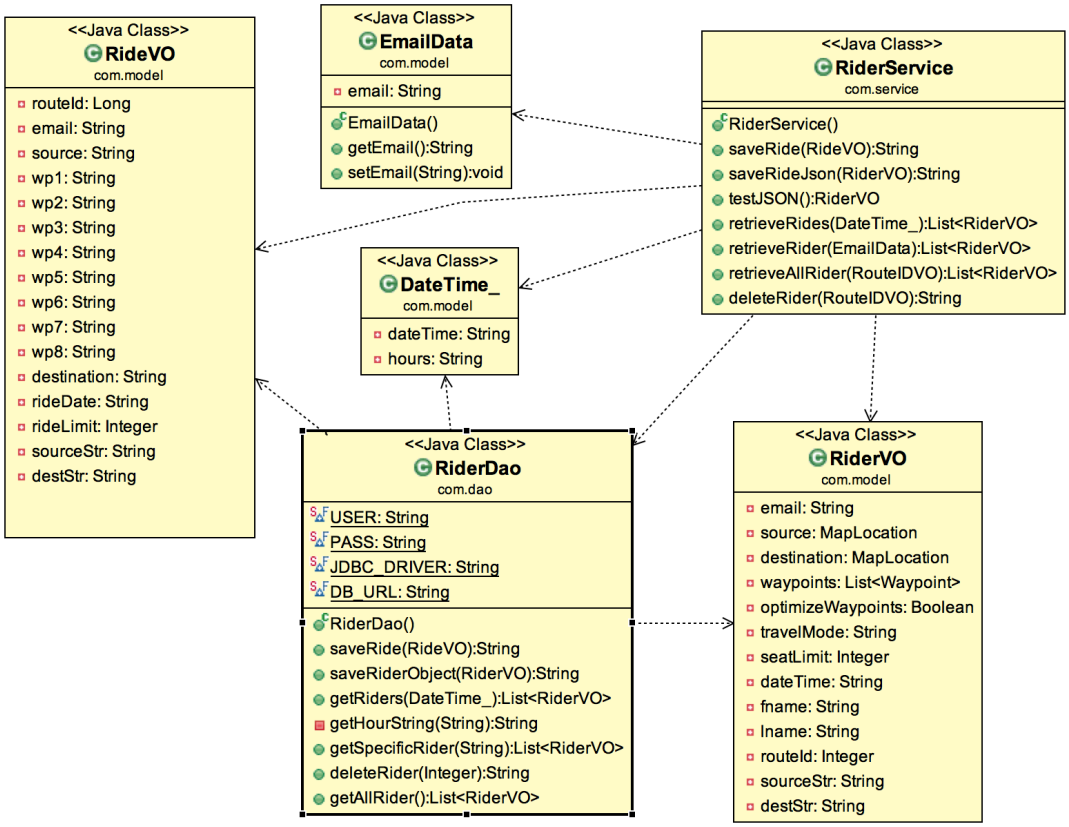

Figure: Class Diagram (Ride)

Class Diagram for Ride Module is shown below [Class Diagram (ride)]. Similar to Customer Module, this contains model like *RideVO.java*, *RiderVO.java*, *and EmailData.java* in its model layer. Service layer is *RiderService.java*, and Dao is *RiderDao.java*. The *RiderVO.java* is the model corresponding to the JSON request, approaching the server, where as the *RideVO.java* is the model that corresponds to the back-end database. These two models are used to format the data that are compatible to both database and front end JSON data, just like a typical functionality of a middle-ware.

Class Diagram of Web Service Overall:

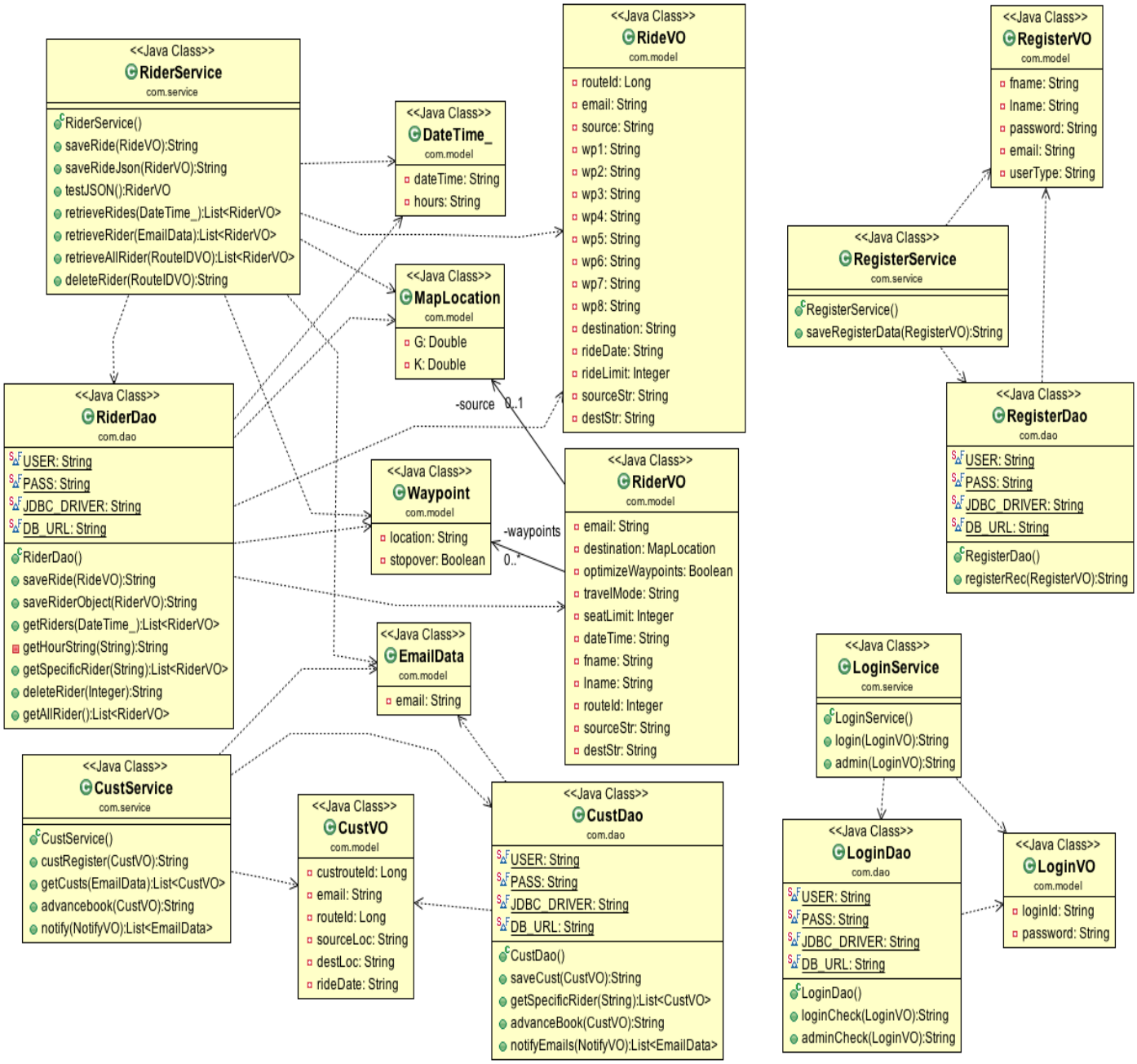

Figure: Class Diagram (Entire Web Service)

5. Requirements:

5.1. Functional Requirement

Requirement ID	Requirement Statement	Priority
ORS_R_01	The ORS should register users based on their types (i.e. ride = offer rides / search rides, customer = search rides)	High
ORS_R_02	The ORS should login users once registered.	Low
ORS_R_03	The ORS should display shared ride's originating location, and destination, and configure number of seats, time and date of travel on Google maps.	Medium
ORS_R_04	The ORS should display possible routes, appropriate to time and date of travel of customer on Google maps while searching for a ride.	Medium
ORS_R_05	The ORS should allow users to put their rides request in advance if none available at the moment.	High
ORS_R_06	The ORS get email notifications for advanced booking events, when the ride is available.	High
ORS_R_07	The ORS should report any security related issues initiated by the user along with the location to the admin for proper action.	Low
ORS_R_08	The ORS should display rate quotes based on the pickup and drop off location.	Low
ORS_R_09	The ORS should allow admins (type 3 users) to monitor rides shared and booked.	High
ORS_R_10	The ORS should allow user to customize his route adding waypoints, and be able to save the custom route created.	High
ORS_R_11	The ORS should be able to show the custom routes of the riders rather than default routes for users who are searching for the ride.	High

Table: Functional Requirement

5.2. System Requirements

5.2.1 Software Used

Operating System:	Mac OS X Yosemite (10.10.3), Window 7 64-bit, Windows 8.1 Pro 64-bit, Linux
User Interface:	HTML5, CSS, Bootstrap, Google Map API.
Programming Language:	Java
Database:	MySQL
For Validations:	JavaScript
IDE:	Eclipse
Web-server	Apache Tomcat 8.0.24

Table: System Requirements

5.2.2. Hardware Used

Hard Disk:	128GB PCIe-based flash storage
RAM:	8GB 1866MHz LPDDR3 SDRAM
Processor:	2.7GHz Dual-core Intel Core i5, Turbo Boost up to 3.1GHz

Table: System Requirements

5.2.3 Software Requirements:

Browser:	Safari 8.0.7 (OR) Internet Explorer 10 (OR) Google Chrome (OR) Mozilla Firefox. Google Chrome PREFERRED

6. Conclusion

This system is not only helpful for finding rides in remote towns for people who cannot afford to buy a vehicle, but also in bigger cities reducing traffic, less travel time, and better for environment. Making a simple user-friendly system for this idea of sharing rides would make a very convenient way of finding rides.

7. Future Enhancements

Since Online Ride Services is developed from open source systems, it provides a platform for more enhancements easily, also making the system scalable. It's user friendly and flexibility features can revolutionize the car-pooling culture in this country. A lot more features can be added to bridge the communication between riders and customer. This system is three-tiered system. All three tiers, front end, middleware and backend can be hosted individually, separately or together in different platforms and domains. It can be hosted in cloud systems like Amazon Web Server and can accommodate requests from thousands of users.

References

[1] *Google Maps API reference document* , by Google Map Developers [Online]. Available: https://developers.google.com/maps/documentation/javascript/reference?hl=en

[2] B.Srivastava. (2012, March 29). *Making Car Pooling Work – Myths and Where to Start* [Online]. Available: https://domino.research.ibm.com/library/cyberdig.nsf/papers/13126524EC12E1FA852579D0003FF105/$File/RI12014.pdf

[3] Levofsky, A. Greenberg (2001, January 7-11). *Organized Dynamic Ride Sharing: The Potential Environmental Benefits And The Opportunity For Advancing the Concept* [Online]. Available: http://ridesharechoices.scripts.mit.edu/home/wp-content/papers/GreenburgLevofsky-OrganizedDynamicRidesharing.pdf

[4] Transit [Online]. Available: http://www.infrastructurereportcard.org/a/#p/transit/overview

[5] F. Guerrini (2014, October 14). *Traffic Congestion Costs Americans $124 Billion A Year, Report Says* [Online]. Available: http://www.forbes.com/sites/federicoguerrini/2014/10/14/traffic-congestion-costs-americans-124-billion-a-year-report-says/.

[6] J. Funk. (2015, April 13). *Uber's Business Model* [Online]. Available: http://www.slideshare.net/funk97/ubers-business-model

[7] J. B. White. (2015, May 22). *Carpooling by Uber, Lyft OK'd by California Assembly* [Online]. Available: http://www.sacbee.com/news/politics-government/capitol-alert/article21693111.html

[9] S. Dent (2014, June 27). *What you need to know about Uber, Lyft and other app-based car services* [Online]. Available: http://www.engadget.com/2014/06/27/uber-lyft-explainer/

[10] J. V. Hall and A. B. Krueger (2015, January 22). *An Analysis of the Labor Market for Uber's Driver-Partners in the United States* [Online]. Available: https://s3.amazonaws.com/uber-static/comms/PDF/Uber_Driver-Partners_Hall_Kreuger_2015.pdf

[11] Sun Microsystems. *Java Location™ Services* [Online]. Available: http://www.pbinsight.com.br/files/resource-library/resource-files/java_location_svcs_whitepaper.pdf

[12] *Java Client Library For Google Maps API Web Services* [Online]. Available: https://github.com/googlemaps/google-maps-services-java