PROJECT REPORT

ON

SOCIO TRAVEL

Submitted in partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER ENGINEERING DEPARTMENT
THE UNIVERSITY OF MUMBAI

SUBMITTED BY

SATVIK SHETTY

HARSHAVARDHAN POOJARY

AMEYA DESHMUKH

UNDER THE GUIDANCE OF

Prof. Nileema Pathak

COMPUTER ENGINEERING DEPARTMENT
ATHARVA COLLEGE OF ENGINEERING
MALAD (W), MUMBAI 400095.
YEAR 2013-2014

PROJECT REPORT

ON

SOCIO TRAVEL

SUBMITTED BY:

SATVIK SHETTY

HARSHAVARDHAN POOJARY

AMEYA DESHMUKH

UNDER THE GUIDENCE OF

PROF. NILEEMA PATHAK

COMPUTER ENGINEERING DEPARTMENT

ATHARVA COLLEGE OF ENGINEERING

MALAD (W), MUMBAI 400 095

YEAR – 2013-2014

CERTIFICATE

ATHARVA COLLEGE OF ENGINEERING

MALAD (W), MUMBAI 400095

YEAR – 2013-2014

This is to certify that the following students of Computer Engineering Department

SATVIK SHETTY

HARSHAVARDHAN POOJARY

AMEYA DESHMUKH

have submitted the project report titled SOCIO TRAVEL in partial fulfillment of the requirements for the degree of bachelor of engineering in Computer Engineering satisfactorily.

PROJECT GUIDE **H.O.D.** **PRINCIPAL**

INTERNAL EXAMINER **COLLEGE SEAL** **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

Before presenting out our project entitled "**SOCIO TRAVEL**", we would like to convey our sincere thanks to many people who guided us throughout the course of this project work.

First, we would like to express our sincere thanks to our Principal **Dr. SHRIKANT P KALLURKAR** for giving us the wonderful opportunity to carry out this project.

We wish to express our gratitude to our project guide **Prof. NILEEMA PATHAK** for selecting the project and providing us with all the details for proper presentation of the project. We also thank her for providing valuable input about the project and for her continuous feedback and support that has guided us through.

Finally, we would like to thank our **H.O.D. Prof. MAHENDRA PATIL** and all teaching, non-teaching staff of the college and friends for their moral support rendered during the course of the project work and for their direct and indirect involvement in the partial completion of our project work, which made our endeavor fruitful.

We would also like to thank the college staff members and our colleagues for their co-operation.

ABSTRACT

All over the world there are more than 10 billion cars and various other public transports such as taxis and auto rickshaws. Of these, according to a research conducted, there are approximately 78% vehicles that travel with only one passenger in them. Considering the amount of carbon emitted into the atmosphere, it would be wiser to utilize this space much more efficiently, thus reducing the number of vehicles on road and in turn reducing the carbon footprint as well as saving money. Our project named "**Socio Travel**" (android application named "Commutator") intends to put forward a solution to this.

Socio Travel is based on the concept of real-time ridesharing. Real-time ridesharing (also known as instant ridesharing, dynamic ridesharing, ad-hoc ridesharing, or dynamic carpooling) is a service that arranges one-time shared rides on a very short notice.

The users of the application search for rides by entering a source, destination and time of travel. It displays results based on fulfillment of certain criteria which the commuter can then join. It creates a way out for your daily commute while cutting down on your expenses. Find or create ride pools in and around your city, just enter the source and destination and collaborate with fellow travelers going to the same destination from in and around a place.

Commutator is an application through which people can overcome to a certain extent the already overcrowded roads and public transports and play their part towards a greener future.

List of Contents

List of Figures

List of Tables

INTRODUCTION AND MOTIVATION

1. Introduction and Motivation:

1.1 Introduction to Project

Over 60% of the population lives in urban areas. Air and noise pollution is getting worse with every passing year. Urban traffic is responsible for 40% of CO2 emissions and 70% of emissions of other pollutants arising from road transport. Also, according to a research conducted, there are approximately 78% vehicles that travel with only one passenger in them. Increasing traffic in town and city centers is responsible for chronic congestion, with the many adverse consequences that this entails in terms of delays and pollution. Every year millions are spent by the Government to deal with this phenomenon. Several solutions have been proposed to these problems, such as a diversity of intelligent transportation systems and solutions. But none of these solutions have ad-hoc trip arrangements.

Real-time ridesharing (also known as instant ridesharing, dynamic ridesharing, ad-hoc ridesharing, or dynamic carpooling) is a service that arranges one-time shared rides on a very short notice.

More than 65% mobile users worldwide have a smartphone. There are approximately 1.5 billion activated mobile devices running on Android as of today which makes Android the most used mobile operating system. Considering this huge user base, it would be wise to harness the potential and use this platform to create an application that could help in addressing the problems stated earlier.

With this project, Socio Travel, we intend to create an Android based application which implements the real-time ridesharing mechanism to create a promising approach for reducing energy consumption. We also intend to assuage traffic congestion while satisfying people's needs in commute and to create a modern and mobile approach that could be popular among students as well as adults alike. It would help utilize the empty seats in most passenger cars or public transports like taxis, auto rickshaws more efficiently and also prove to be of significant social and environmental benefit.

1.2 Problem Definition

1.2.1 Existing Systems

There are certain applications that have tried to use the concept of dynamic ridesharing. These are:

- **Meter Share**

This application was designed by Positive3 developers, which is developed to make the travelling experience easier. The application lets you create a ride, and share it with fellow travelers. It asks you to enlist details like the time when you'd like a ride, the number of free seats available and the end to end locations of your travel. The details (Name, Age, and Gender) of your fellow travelers are listed once they join your ride.

- **PoolMyRide**

This application was developed by Abhiskek Talwar. This app can help you find a daily carpool. PoolMyRide creates a way out for your daily commute while cutting down your expenses. Find ride pools in and around your city, just enter source and destination and wait for your Facebook friends and app users to share the ride.

1.2.2 Limitations

- **Poor User Interface**:
  The applications already designed and published in the market have a shoddy user interface which discourages the users.

- **App not well marketed**:
  Android users were unaware of presence of such applications in the market and thus usage has been to a bare minimum.

- **Requires compulsory Facebook login**:
  Very few users like to share their Facebook id with strangers or some of the users might not use Facebook itself.

- **Limited Features**:
  Features of the existing applications are unappealing and some of the applications do not even get the basic features right.

- **Frequent crashes and bugs**:
  The application crashes and stops responding too frequently for comfort. They are ridiculed with bugs and are not updated in time which proves to be pretty irritating for the user.

- **Not enough security for commuters**:
  No arrangements are done to protect commuters' especially female passengers from possible threats during the ride.

- **No proximity locations**:
  No checkpoints or proximity locations are designed in applications in which a user can join the ride in between or along the travel path.

1.3 Requirement Analysis

1.3.1 User Requirements

- As the application is developed for Android OS, the user must have a mobile device running on the Android platform.
- The device must have a mobile running on Android 3.0, HoneyComb as it is the minimum API that supports fragments used in the application.
- There should be a working data usage plan for the application to work.
- The user should have an SMS plan or sufficient balance in case of emergencies.

1.3.2 Functional Requirements

- The user should be able to login into his account or create an account if he's not already registered.
- The application should be able to validate the credentials entered.
- The user should be able to create, search or view past ride history.
- The user should be able to join a ride if available.
- The application should be able to alert the user with appropriate notifications on changes in the ride he has joined or created.
- Once the user has joined a ride, he should be able to communicate with other fellow commuters via chat and be able to view their details.
- The user should be able to set a SOS number of his choice.
- He/She should be able to view/edit his or her profile.
- The user should finally be able to logout of his application.

1.3.3 Non-Functional Requirements

- The application shall be light-weight and should not put an overhead on the performance of the mobile device.
- The application shall be efficient in terms of time required and shall be able to respond to the requests made to the server in a timely manner.
- The interface of the application shall be user friendly and convenient.
- The system shall be designed such that it shall minimize battery drain.

1.4 Scope

The basic functionality of the application, Commutator, is the same as the already existing systems that were explained earlier. But what sets this application apart from the existing ones are the following functionalities that we look to bring in and thus in turn hope to revolutionize the way people have been travelling especially in a busy city like Mumbai.

• User friendly
Many people (including us) found it difficult to use some of the existing applications, some because they were a plain eye-sore or difficult to go through in some cases. We intend to create a clutter free, simple, yet attractive user interface that could be easy to understand even by a layman.

• Creating a new user account without Facebook
Almost all existing travel sharing apps require the user to login through Facebook compulsorily. Some users find this inconvenient since they prefer not to use this as an option to login or might just not have an account. We will allow the other users to create a new account for themselves through Parse.

• Proximity location points
In the existing systems, one has to enter the exact same source and destination to find and share a ride successfully. But sharing the same source and destination is not possible in most cases. So what we intend to bring to the table is that people who travel can join the ride when they are in the vicinity of the source and want to travel to places nearby the destination.

• Time tolerance
Time tolerance is a user set time while creating a ride in which he can set the time difference. Suppose the user sets the time tolerance as 15 minutes, this means that the user doesn't mind travelling 15 minutes before or after the scheduled time. This way more number of users can find the ride.

• In-built ride chat
An in-built ride chat can help the travelling users to communicate with each other before the scheduled trip if they intend to discuss about time of meeting or destination they intend to travel to.

• **Safety For Women**

Safety for women has become a necessity in today's world, especially in a city like Mumbai. For women commuters creating/searching a ride, we will give them an option if they would prefer other women only or wouldn't mind travelling with male commuters too. Thus rides will be displayed according to this preference only.

• **SOS feature**

Commuters can set an emergency number to which a message would be sent instantly in case they sense they are in danger. The message can be customized to their liking. Additionally, Safety tips have been provided which all users must go through.

LITERATURE SURVEY AND RELATED THEORY

2. Literature Survey and Related Theory

2.1 Android (Operating System)

Android is a Linux based operating system. Initially developed by Android, Inc., which Google backed financially and later bought in 2005, Android was unveiled in 2007 along with the founding of the Open Handset Alliance: a consortium of hardware, software, and telecommunication companies devoted to advancing open standards for mobile devices. The first Android-powered phone was sold in October 2008. Android is open source and Google releases the code under the Apache License. This open-source code and permissive licensing allows the software to be freely modified and distributed by device manufacturers, wireless carriers and enthusiast developers. Additionally, Android has a large community of developers writing applications ("apps") that extend the functionality of devices. Anyone can upload a new application on the App Store as either free or payable by just paying a one-time nominal fee. These applications uploaded by the developers can be easily downloaded by the users and can enjoy additional features like games, interactive media and business plans.

Android uses the Java programming language. Getting started with the Android API is easy, the API is open, i.e. developers can access almost every low-level function and are not sandboxed. Android's user interface is based on direct manipulation, using touch inputs that loosely correspond to real-world actions, like swiping, tapping, pinching and reverse pinching to manipulate on-screen objects. In addition, the Android API allows easy access to the hardware components. Android devices are powerful mobile computers with permanent internet connectivity and a rich variety of built-in sensors. Interesting are the numerous communication interfaces like Wi-Fi, Bluetooth and GSM/UMTS, USB, and the integrated sensors, that is: accelerometer, gyroscope, compass and GPS. The Android Mobile Phone Platform by Google becomes more and more popular among software developers, because of its powerful capabilities and open architecture. Android is a technology that is making huge leaps in the mobile computing world .Currently Android powers millions of phones, tablets, and other devices and brings the power of Google and the web into your hands. As of May 2013, 48 billion apps have been installed from the Google Play store, and as of September 3, 2013, 1 billion Android devices have been activated.

2.2 Dynamic Ridesharing

Real-time ridesharing (also known as instant ridesharing, dynamic ridesharing, ad-hoc ridesharing, dynamic carpooling) is a service that arranges one-time shared rides or a trip on very short notice. A "trip" is defined as "a single instance of travel from one geographic area to another." This type of carpooling generally makes use of three recent technological advances:

- GPS navigation devices to determine a driver's route and arrange the shared ride
- Smartphones for a traveler to request a ride from wherever they happen to be
- Social networks to establish trust and accountability between drivers and passengers.

In an organized dynamic ridesharing program, people either offer or request rides from a central database or ridesharing agency. When a request is made, a database of potential drivers and riders (those who registered for the program) is searched for matches corresponding to the approximate time and destination of the request. The requestor receives the names and contact information of any matches made. Usually, the person requesting the ride contacts potential drivers to arrange it. In some cases, the ridesharing agencies may make initial contact with potential drivers and put the participants together only if a match is found.

Initial Ridesharing Projects:

Early real-time ridesharing projects began in the 1990s, but they faced obstacles such as the need to develop a user network and a convenient means of communication. Gradually the means of arranging the ride shifted from telephone to internet, email, and smartphone. Now with the advancement of technology, it has become much easier to implement dynamic ride-sharing and reap the benefits of it.

Difference between Dynamic Ridesharing and Traditional Ridesharing:

Dynamic ridesharing differs from traditional ridesharing in two ways. First, a traditional ridesharing system assumes that users have a fixed schedule and fixed origin and destination points. Dynamic ridesharing systems consider each trip individually and are designed to accommodate trips to random points at random times by matching user trips without regard to trip purpose. Second, dynamic ridesharing systems have to provide match information close to the time when

users need travel. Traditional ridesharing systems normally provide a match list through the mail, a process that can take several days.

Similar to traditional ridesharing, dynamic ridesharing can either be an organized program run by an agency or an informal system run by users (casual carpooling).

Why Dynamic ridesharing is needed:

Automobile travel causes serious adverse environmental effects and damage to human health. Air pollution emitted from vehicles, in the form of carbon monoxide, ground level ozone, nitrogen oxide, particulate matter, and sulfur dioxide, can cause respiratory problems and headaches in humans.

A 2010 survey at the University of California, Berkeley found 20% of respondents willing to use real-time ridesharing at least once a week; and real-time ridesharing was more popular among current drive-alone commuters (30%) than transit or non-motorized commuters.

Real-time ridesharing is promoted as a way to better utilize the empty seats in most passenger cars, thus lowering fuel usage and transport costs. It can serve areas not covered by a public transit system and act as a transit feeder service. It is also capable of serving one-time trips, not only recurrent commute trips or scheduled trips. Dynamic ridesharing, like car sharing, allows households to limit their car ownership by providing opportunities to use an alternative form of transportation that does not sacrifice convenience.

Furthermore, it can serve to limit the volume of car traffic, thereby reducing congestion and mitigating traffic's environmental impact.

The research paper by Andrew Amey from MIT on Real-time Ridesharing begins with a definition of “real-time” ridesharing and follows with a comprehensive categorization of the challenges hindering greater rideshare participation. The information gathered suggests that rather than being a single challenge to be overcome, the ‘rideshare challenge’ is a series of economic, behavioral, institutional and technological obstacles to be addressed. The potential opportunities, and obstacles, created by “real-time” innovations are then highlighted. The paper concludes with several recommended ‘next steps’ to further understand how rideshare participants use “real-time” services, focusing specifically on the need for multiple, comprehensive “real-time” rideshare trials. This study provides an important foundation upon which further “real-time” ridesharing research can take place.

2.3 Backend as a Service (BaaS)

Backend as a service (BaaS), also known as "mobile backend as a service" (MBaaS), is a model for providing web and mobile app developers with a way to link their applications to backend cloud storage while also providing features such as user management, push notifications, and integration with social networking services. These services are provided via the use of custom software development kits (SDKs) and application programming interfaces (APIs). BaaS is a relatively recent development in cloud computing, with most BaaS startups dating from 2011 or later.

Although a fairly nascent industry, trends indicate that these services are gaining mainstream traction with enterprise consumers. The global BaaS market had an estimated value of $216.5 million in 2012 and projected to grow to $7.7 billion by 2017. Examples of consumer backend providers include Parse (acquired by Facebook), StackMob, Apigee, QuickBlox, etc.

Evolution of Mobile Ecosystem:

Earlier the world of cell phones consisted of only two main players, the service providers and the handset OEMs. With the advent of smart phones and cloud technology mobile app developers and cloud service providers, API and SDK layers, have also become an integral part of the mobile ecosystem.

Need for BaaS:

Web and mobile apps require a set of features on the backend, including push notifications, integration with social networks, and cloud storage. Each of these services has their own API that must be individually incorporated into an app, a process that can be time-consuming and complicated for app developers. BaaS providers form a bridge between the frontend of an application and various cloud-based backends via a unified API and SDK.

Providing a consistent way to manage backend data means that developers do not need to redevelop their own backend for each of the services that their apps need to access, potentially saving both time and money.

Although similar to other cloud-computing developer tools, such as software as a service (SaaS), infrastructure as a service (IaaS), and platform as a service (PaaS), BaaS is distinct from these other services in that it specifically addresses the cloud-computing needs of web and mobile app developers by providing a unified means of connecting their apps to cloud services.

ANALYSIS AND DESIGN

3. Analysis and Design

3.1 Design Model

Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product.

Agile Methods break the product into small incremental builds. These builds are provided in iterations. Each iteration typically lasts from about one to three weeks. Every iteration involves cross functional teams working simultaneously on various areas like planning, requirements analysis, design, coding, unit testing, and acceptance testing. Each release is thoroughly tested to ensure software quality is maintained.

At the end of the iteration a working product is displayed to the customer and important stakeholders.

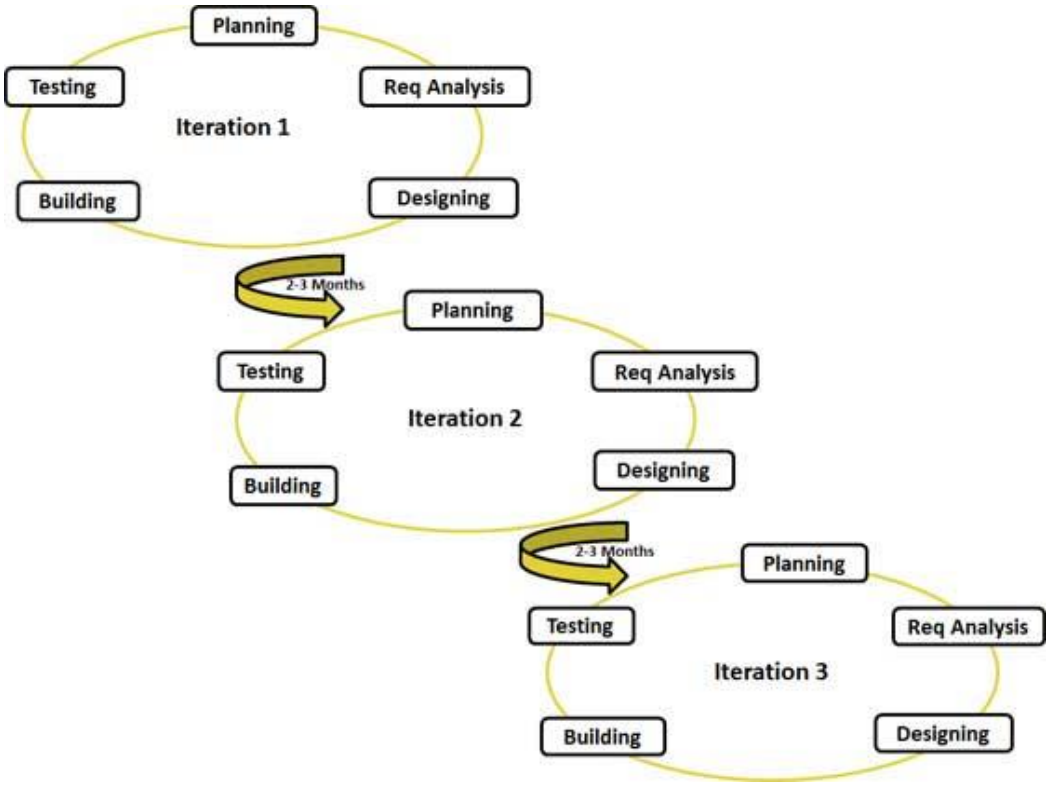


FIGURE 3.1: AGILE MODEL

In agile the tasks are divided into time boxes (small time frames) to deliver specific features for a release.

Iterative approach is taken and working software build is delivered after each iteration. Each build is incremental in terms of features; the final build holds all the features required by the customer.

Reasons for selecting Agile Development Model:

- The freedom agile gives to change is very important. New changes can be implemented at very little cost because of the frequency of new increments that are produced.
- To implement a new feature the developers need to lose only the work of a few days, or even only hours, to roll back and implement it.
- In agile model very limited planning is required to get started with the project.
- Changes can be discussed and features can be newly effected or removed based on feedback. This effectively gives the customer the finished system they want or need.
- Promotes teamwork and cross training.
- Functionality can be developed rapidly and demonstrated.

Phases of Agile Model:

- **Planning:**
  - This phase will involve establishing plan for software engineering work; addresses technical tasks, resources, work products, and work schedule. Timeline chart is used for planning and scheduling. All that needs to be done and all that is needed to get started is well documented.
  - With every increment, the plan might change although the ultimate goal remains the same.

- **Requirement Analysis:**
  - This phase deals with the basic requirements of the system and must be understood by the software engineer, who is called the analyst. In this the information domain, functional, behavioral requirement of the system is understood. The entire requirements will be then well documented and discussed by the team.
  - Steps are taken to carry out analysis, whether new requirements effects the current flow of the system.

- **Designing:**
  - The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts.
  - Design elements describe the desired system features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, pseudo-code, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the system in sufficient detail, such that skilled developers and engineers may develop and deliver the system with minimal additional input design.

- **Building:**
  - In this step the design is translated into machine readable form. Coding will be done using Eclipse which makes use of the various Android APIs, packages and libraries.
  - Development of the application and integration of various modules is done in this stage.

- **Testing:**
  - This phase begins once the coding is done. This phase includes testing and fixing bugs. This is done at various levels.
  - While performing testing, major focus is on connectivity of the application. The testing ensures execution at all the paths, functional behaviors.
  - The purpose of testing is to uncover errors, fix the bugs and meet the requirements.
  - In Agile model, every iterative development is tested before proceeding to the next iteration.

3.2 ER Diagram

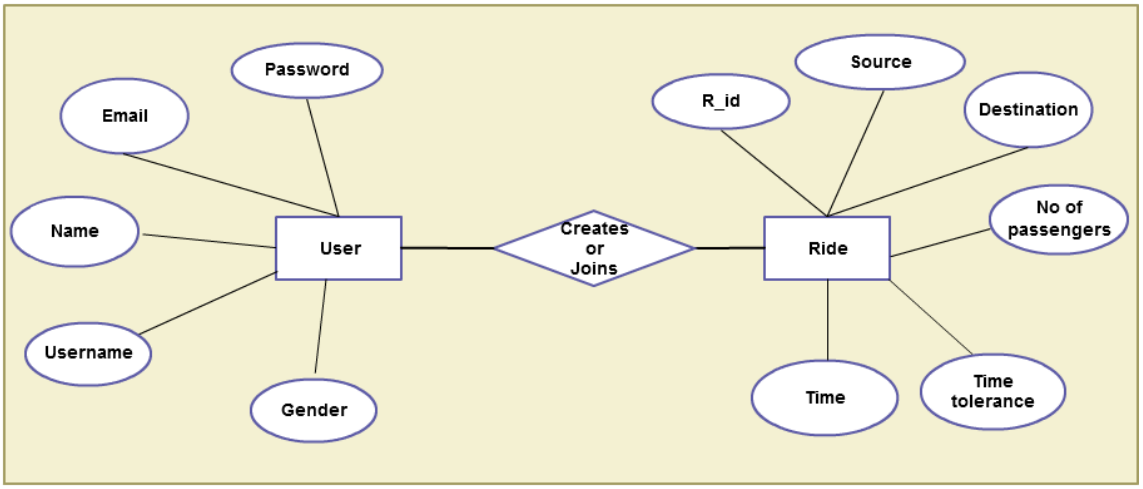


FIGURE 3.2: ER DIAGRAM

3.3 Data Flow Diagram:

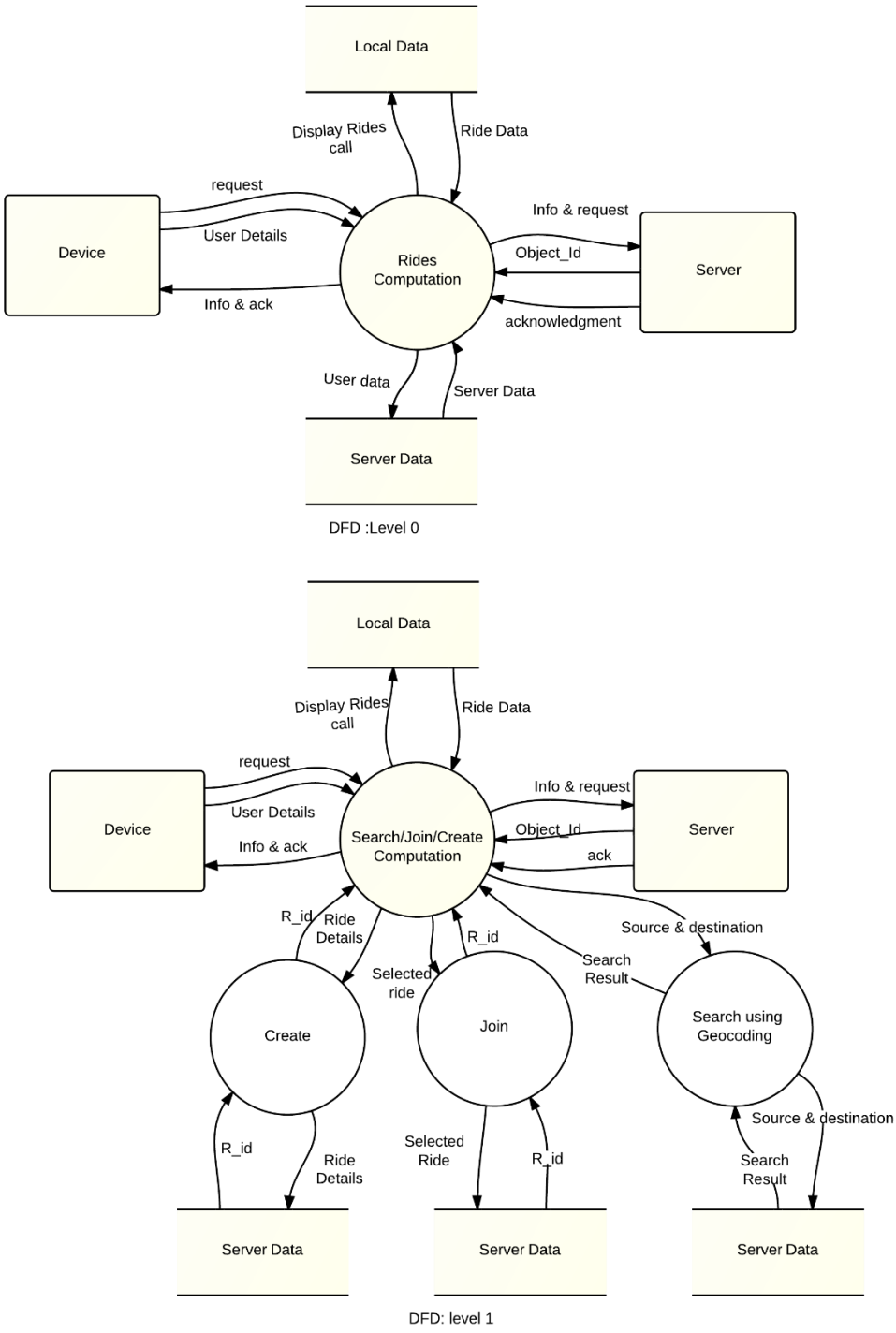


FIGURE 3.3: DATA FLOW DIAGRAM

3.4 UML Diagram:

3.4.1 Class Diagram:

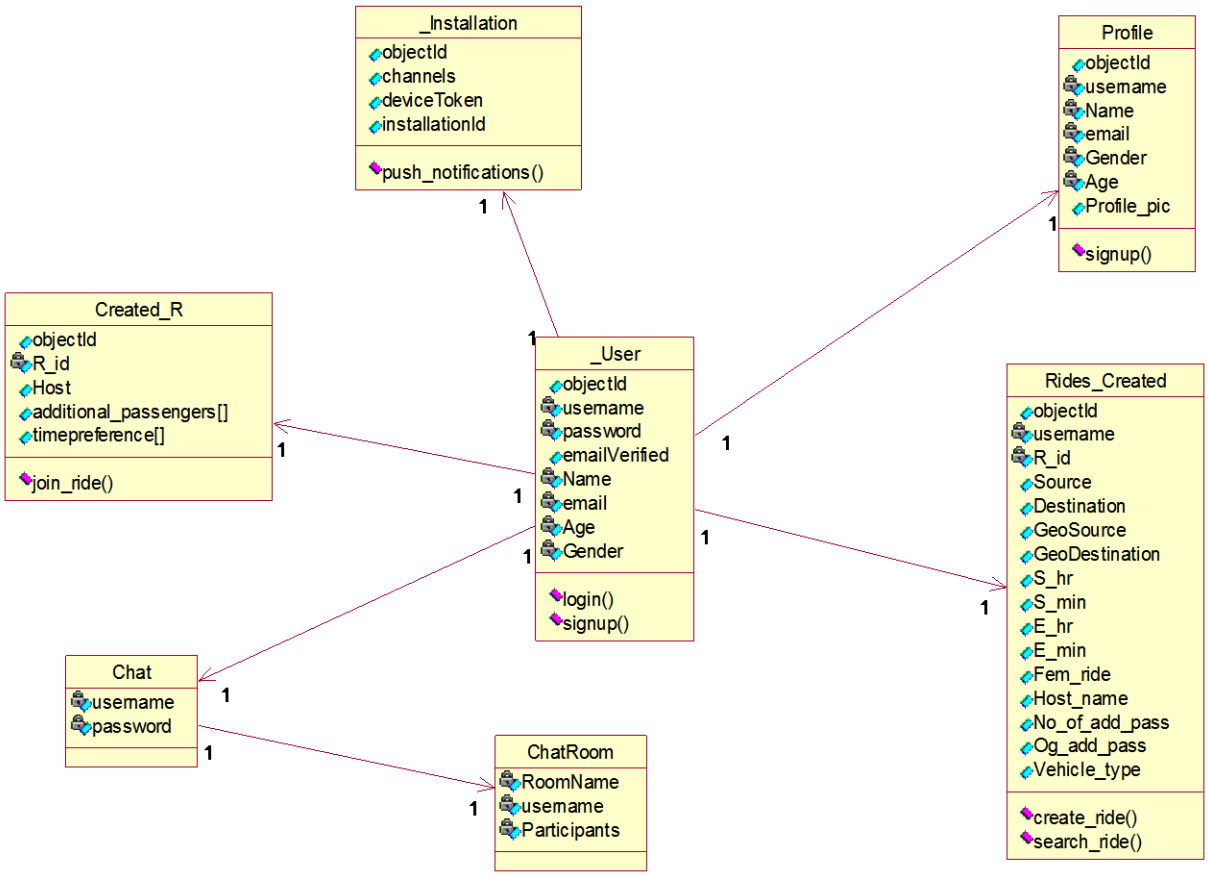


FIGURE 3.4: CLASS DIAGRAM

3.4.2 Use Case Diagram

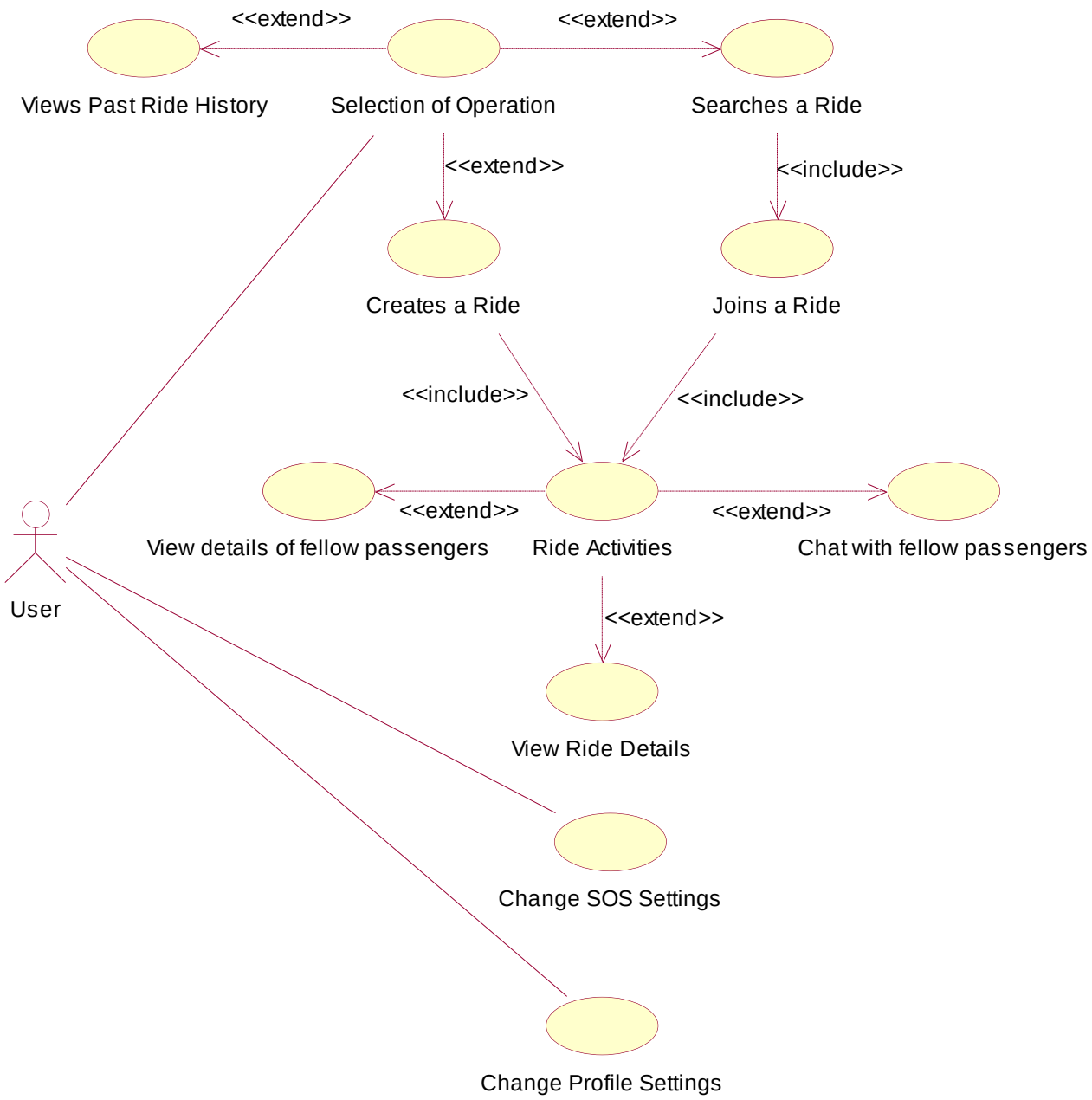


FIGURE 3.5: USE CASE DIAGRAM

3.4.3 Sequence Diagram:

3.4.3.1 Sign Up and Login Sequence Diagram

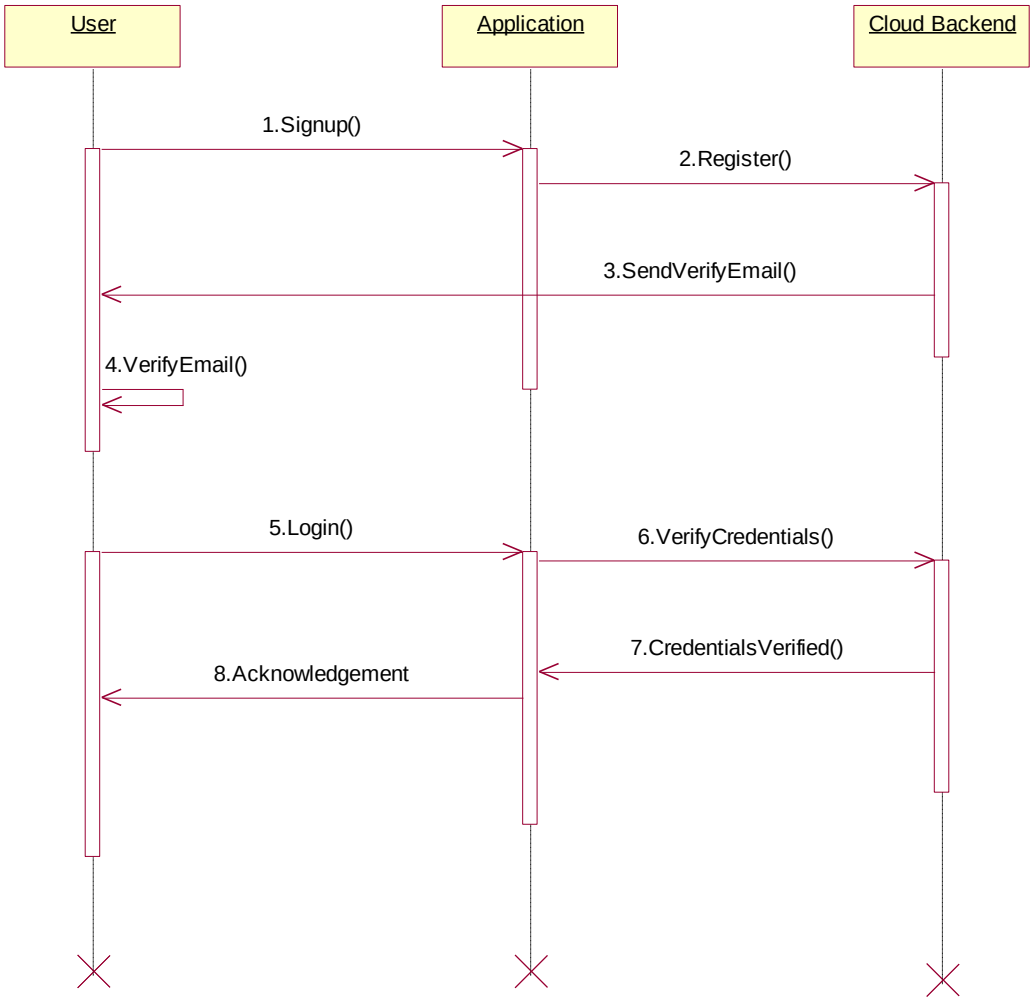


FIGURE 3.6: SEQUENCE DIAGRAM 1

3.4.3.2 Create Ride Sequence Diagram

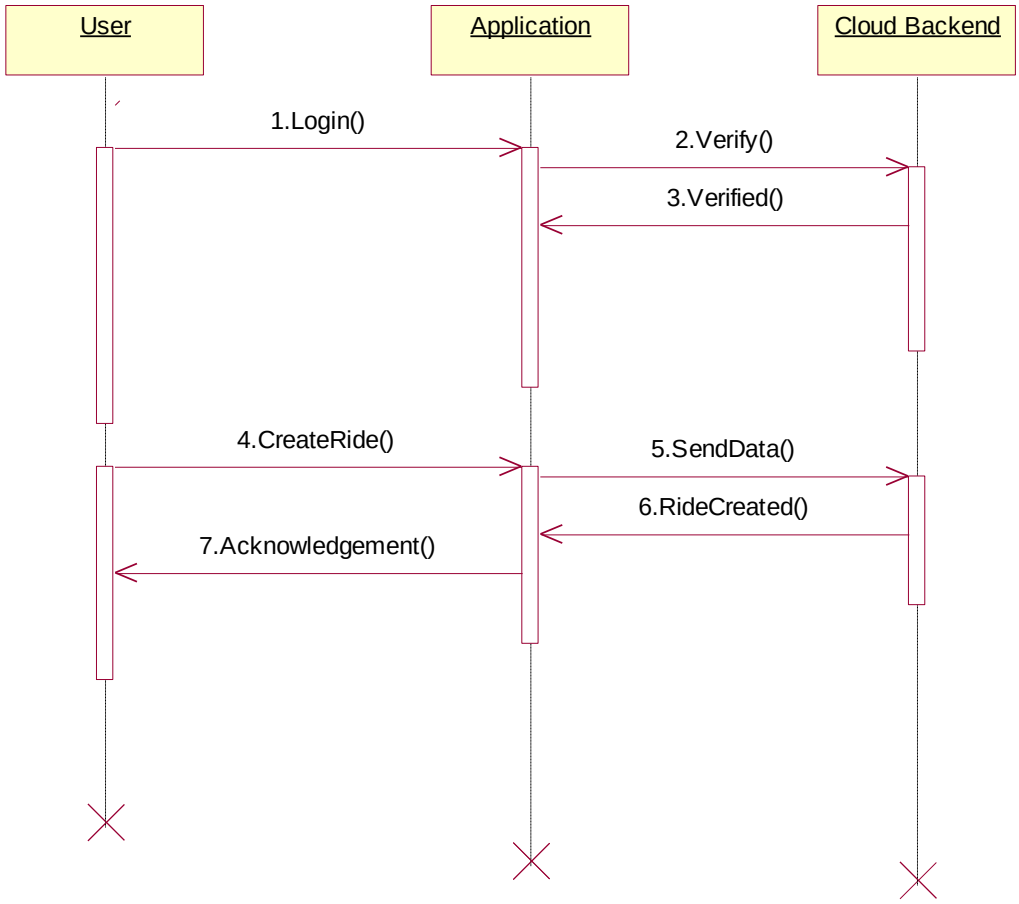


FIGURE 3.7: SEQUENCE DIAGRAM 2

3.4.3.3 Search Ride Sequence Diagram

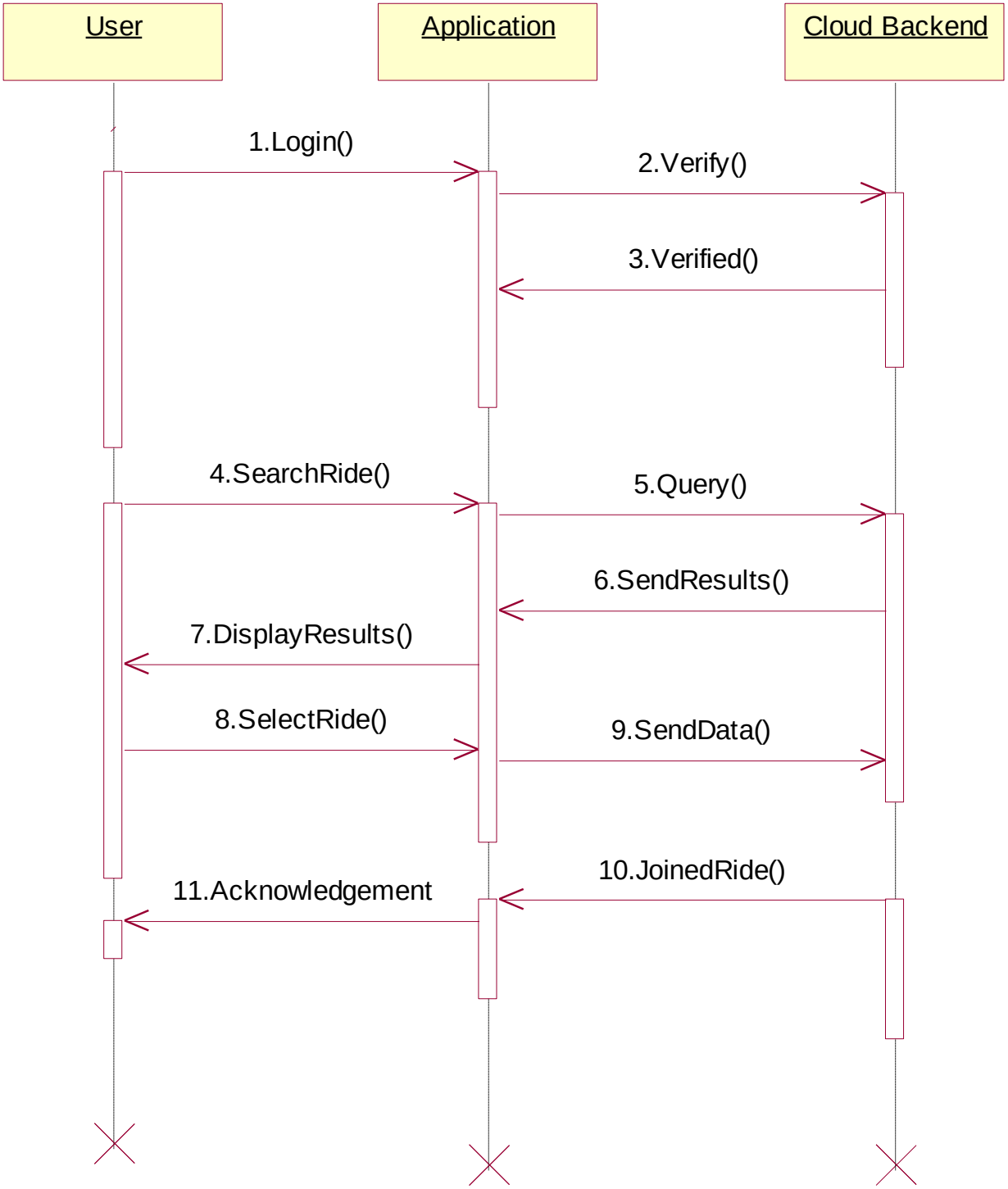


FIGURE 3.8: SEQUENCE DIAGRAM 3

3.4.4 Activity Diagram:

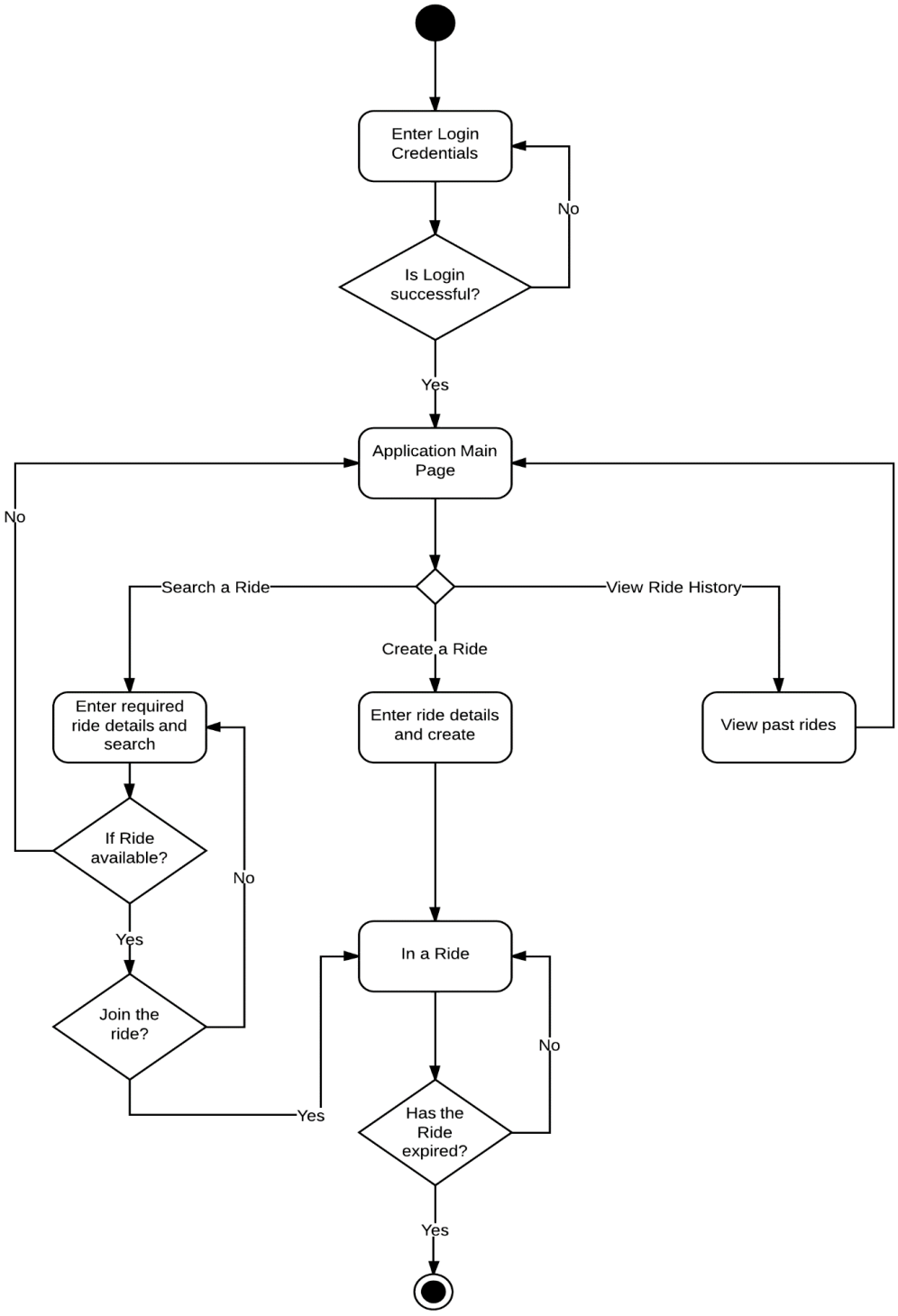
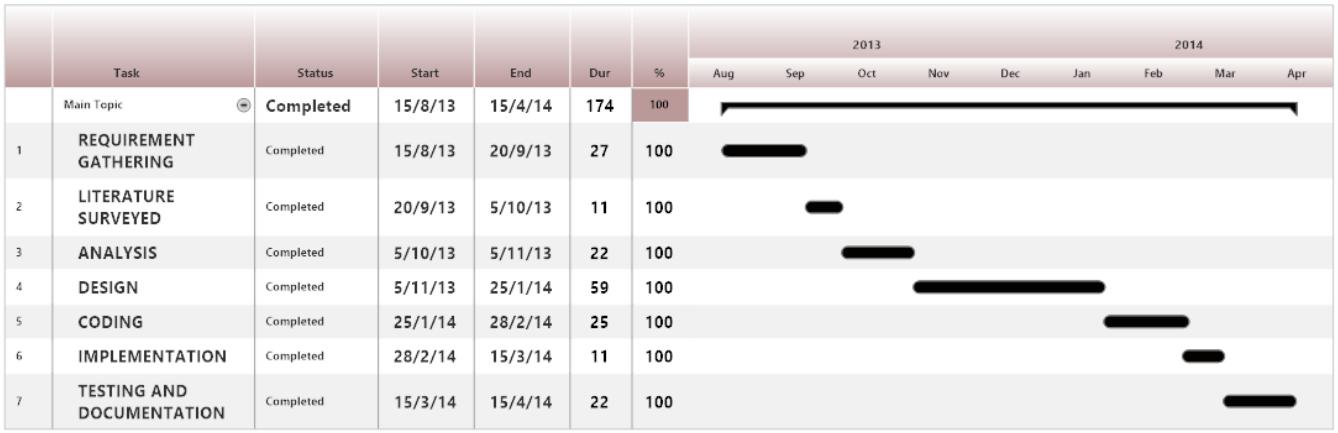


FIGURE 3.9: ACTIVITY DIAGRAM

PROJECT TIME AND TASK DISTRIBUTION

4. Project Time and Task Distribution:



	Task	Status	Start	End	Dur	%
	Main Topic	Completed	15/8/13	15/4/14	174	100
1	REQUIREMENT GATHERING	Completed	15/8/13	20/9/13	27	100
2	LITERATURE SURVEYED	Completed	20/9/13	5/10/13	11	100
3	ANALYSIS	Completed	5/10/13	5/11/13	22	100
4	DESIGN	Completed	5/11/13	25/1/14	59	100
5	CODING	Completed	25/1/14	28/2/14	25	100
6	IMPLEMENTATION	Completed	28/2/14	15/3/14	11	100
7	TESTING AND DOCUMENTATION	Completed	15/3/14	15/4/14	22	100

FIGURE 4.1: GANTT CHART

TASKS CARRIED OUT:

A. REQUIREMENT GATHERING: - START DATE: 15/8/2013

- In this phase all the requirements of the project are studied and understood.
- The software required for building the application is acquired.

B. LITERATURE SURVEYED: - START DATE: 20/9/2013

- All the theory and literature related to the project are studied here.
- Similar projects were surveyed.

C. ANALYSIS: - START DATE: 5/10/2013

- All the analysis regarding the project is done in this phase.
- The analysis involves Risk analysis, market analysis and feasibility.

D. DESIGN: -START DATE: 5/11/2013

- Here the project design and basic user interface is visualized and created on paper in terms of graphs, diagrams and charts.
- The basic GUI that is to be coded is created here.

E. CODING:-START DATE: 25/1/2014

- This is the most important phase where the coding of various modules is done.
- We backup the XML layout with java code.

F. IMPLEMENTATION: - START DATE: 28/2/2014

- Here all the different modules and parts created are assembled together and integrated into one to create the application

  .

G. TESTING AND DOCUMENTATION: - START DATE: 15/3/2014

- In the testing phase the various modules are tested independently as well as in an integrated phase.
- Tests like unit testing, black box testing and white box testing are performed.
- Finally the required documentation is done.

IMPLEMENTATION

5. Implementation:

5.1 Workflow Diagram:

5.1.1 Sign Up Workflow Diagram:

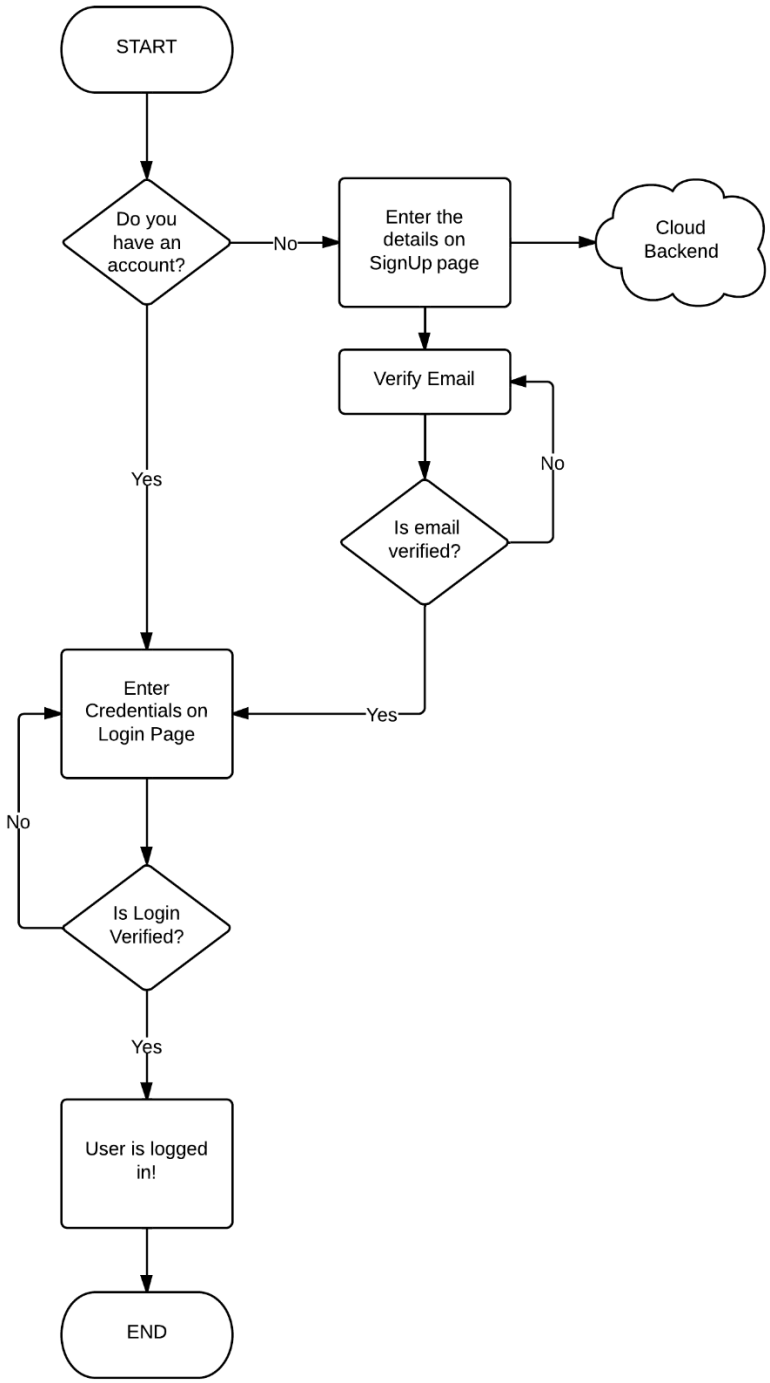


FIGURE 5.1: WORKFLOW DIAGRAM 1

5.1.2 Create Ride Workflow Diagram:

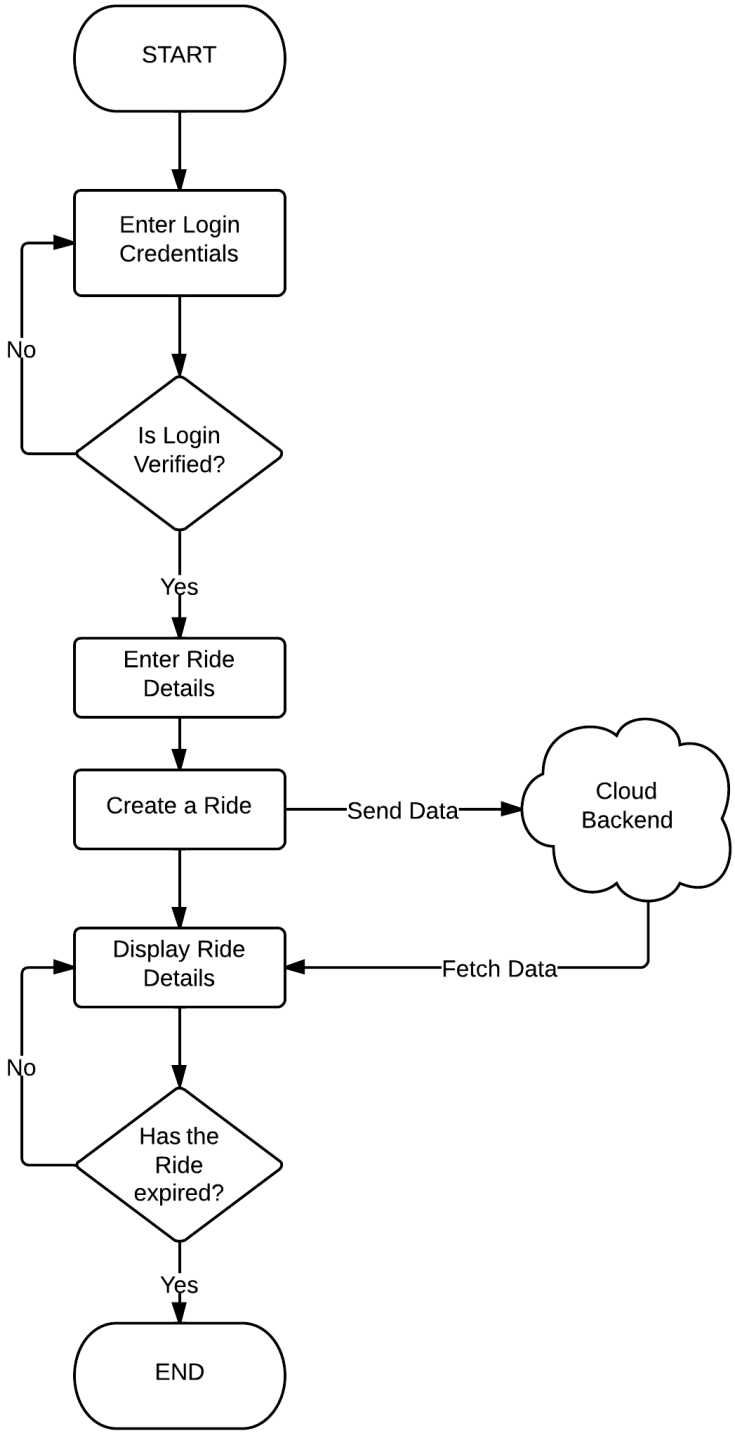


FIGURE 5.2: WORKFLOW DIAGRAM 2

5.1.3 Search Ride Workflow Diagram:

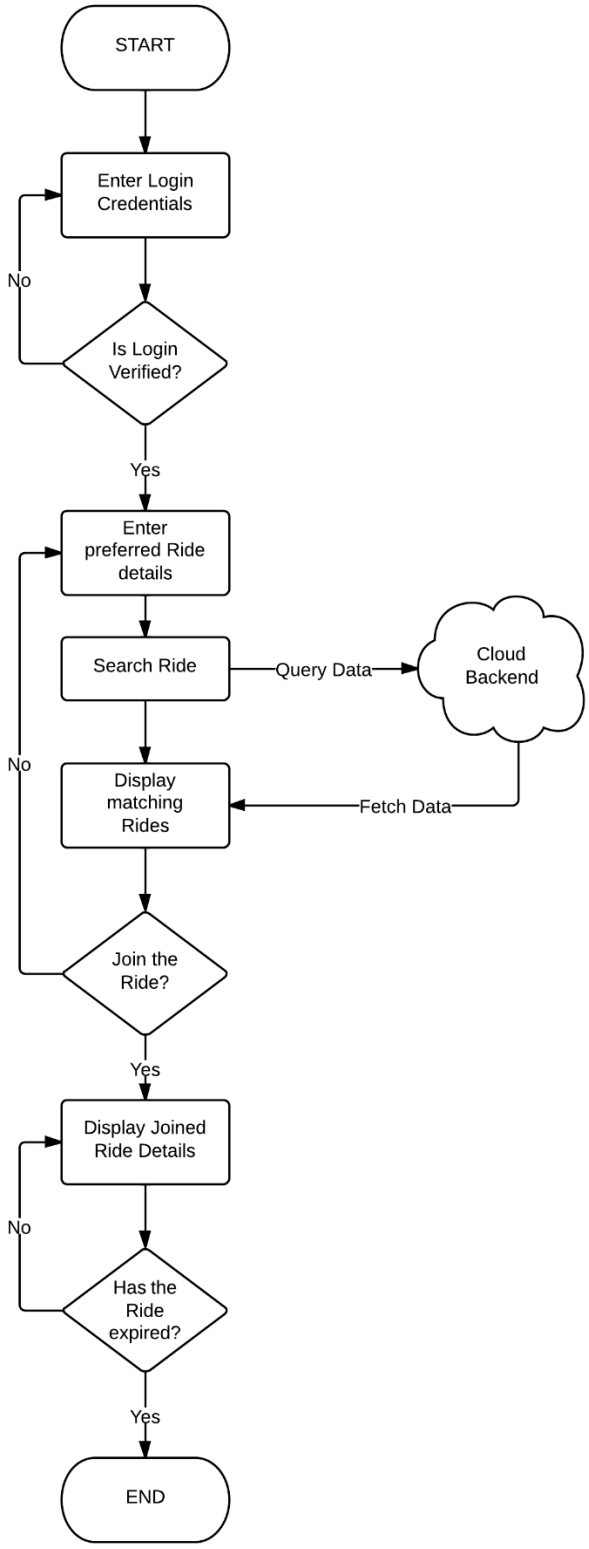


FIGURE 5.3: WORKFLOW DIAGRAM 3

5.2 Integration of Modules:

In this section we will discuss the different modules in our project and we will also show how various modules are integrated together. We will show how various activities such as login, sign up, ride creation and search are performed and how they interact with each other and how they interact with the server and database which constitutes the backend, thus presenting the project as a coherent whole.

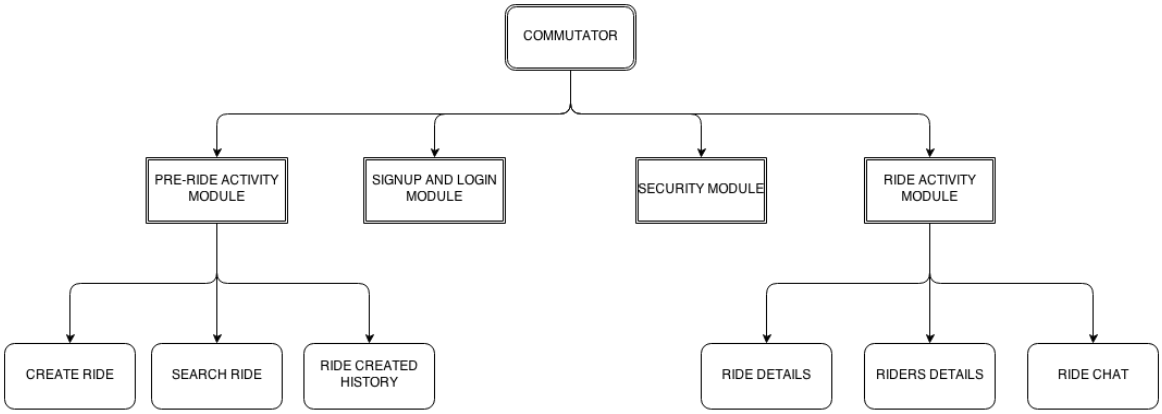


FIGURE 5.4: MODULE TREE

5.2.1 Sign Up and Login Module:

The user's first encounter with the application starts with the Login page. He is presented the options to login through a previously registered account or sign up if he is using the application for the first time. Once the user fills in the required form shown below, he will be asked to verify his email address. An email will be sent to the entered email address, the user as to visit the link sent to verify himself. The user's data is stored in two different classes namely _User and Profile. Data from the class _User is used to verify every time the user tries to login. Data from the class Profile is used for displaying purposes. When the user signs up he is also registered in QuickBlox, which is used as a chat backend.

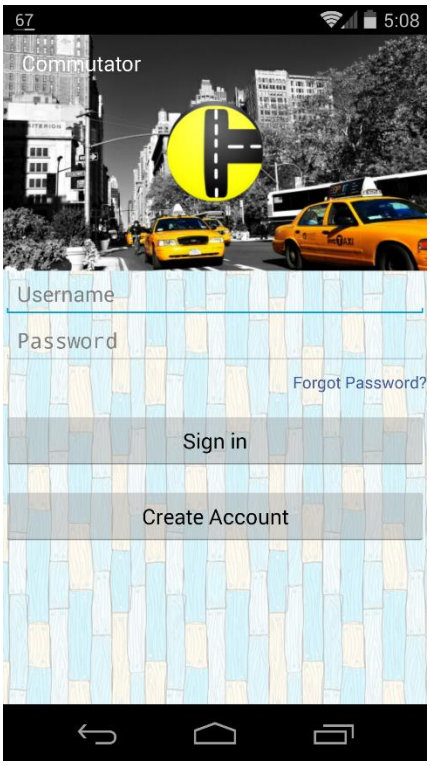


FIGURE 5.5: LOGIN

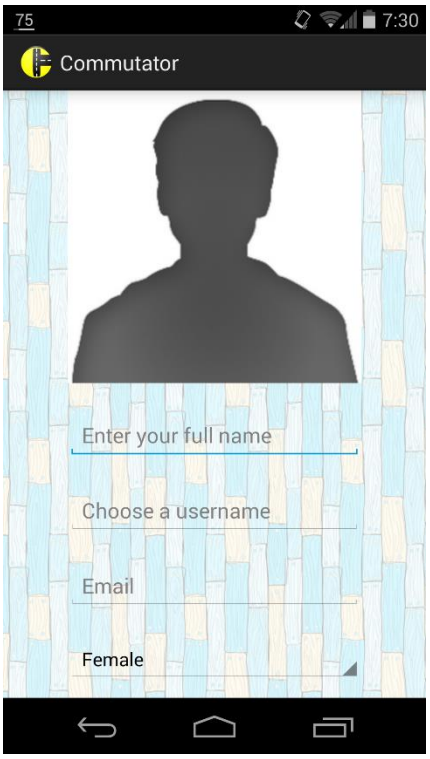


FIGURE 5.6: SIGN UP

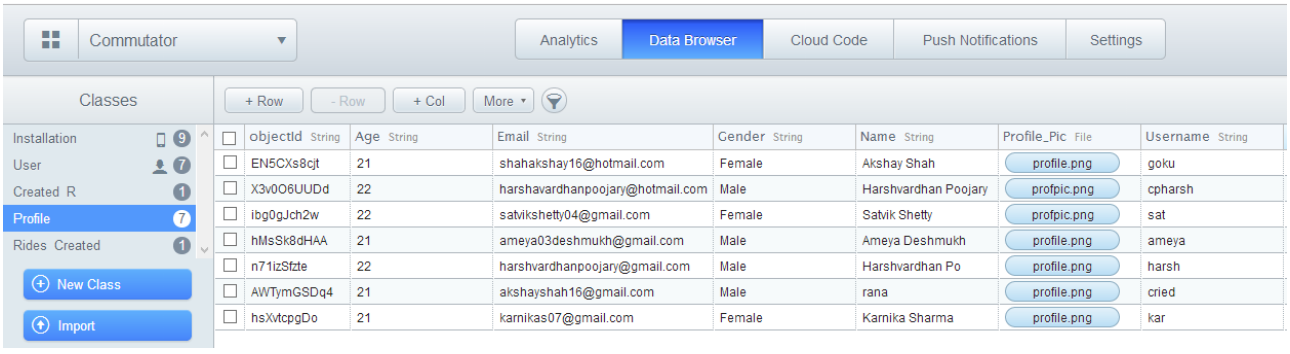


FIGURE 5.7: PROFILE CLASS

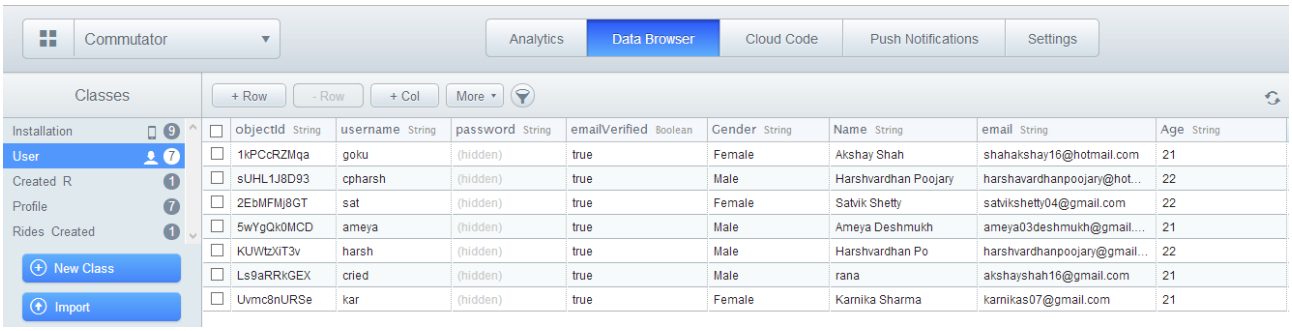


FIGURE 5.8: _USER CLASS

5.2.2 Pre-ride Activity Module:

5.2.2.1 Create a ride:

In this part a user is given the power to create a ride of his choice. The form he is required to fill in shown below. The data from this form is uploaded to the backend. A new row is created with in two classes namely,
Rides_Created and Created_R. Their structure can be seen in the screenshot provided. Each user has a unique ride id (R_Id). Once a ride is created it can be distinguished by its ride id. The source and destination entered are turned into geopoints and stored in the Rides_Created. The time tolerance as well is calculated efficiently by the application and stored in the backend. All the search queries are associated with Rides_Created. Created_R stores the names and the time preferences of the fellow passengers who have joined the ride.

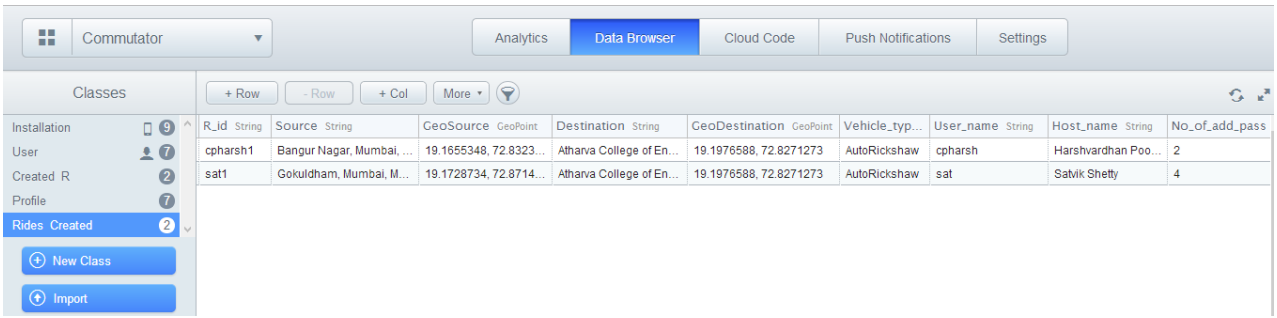


FIGURE 5.9: RIDES_CREATED CLASS

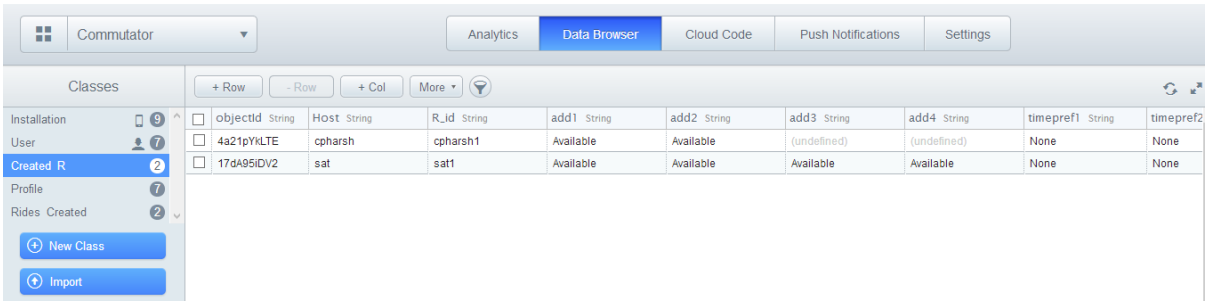


FIGURE 5.10: CREATED_R CLASS

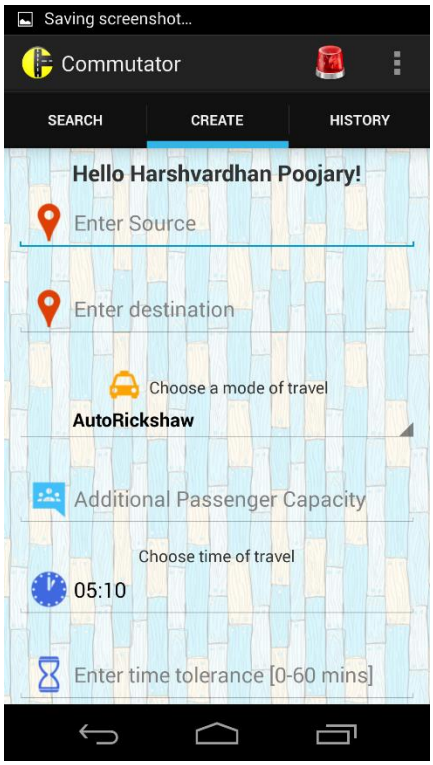


FIGURE 5.11: CREATE RIDE FRAGMENT

5.2.2.2 Search a ride:

A user can search a ride using this fragment. The source, destination and time entered by the user are the parameters for the search. The source and destination entered by the user is converted to geopoint and are used in a query on the class Rides_Created. A ride shows up in the search results if all of the following are true:

1. The source is present in one and a half kilometre radius of any the created rides' source.
2. The destination is present in one and a half kilometre radius of any the created rides' destination.
3. The time preference entered by the user lies in between the time preference set by the ride creator.

Once the search results are displayed to the user he can either join the ride or search for another one with different parameters. If the user joins the displayed ride he'll move on the Ride Activity module.

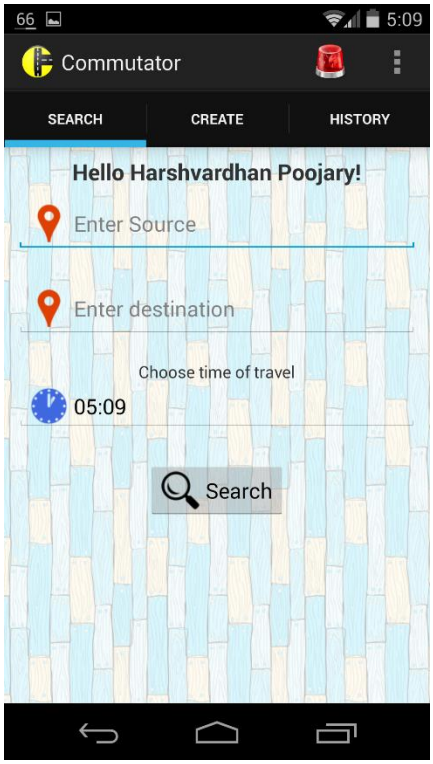


FIGURE 5.12: SEARCH RIDE FRAGMENT

5.2.2.3 View ride created history:

This feature enables the user to view the rides that he has created in the past. It also gives him an option to erase all the rides stored. This module is not much of functional use and will be developed in the future.

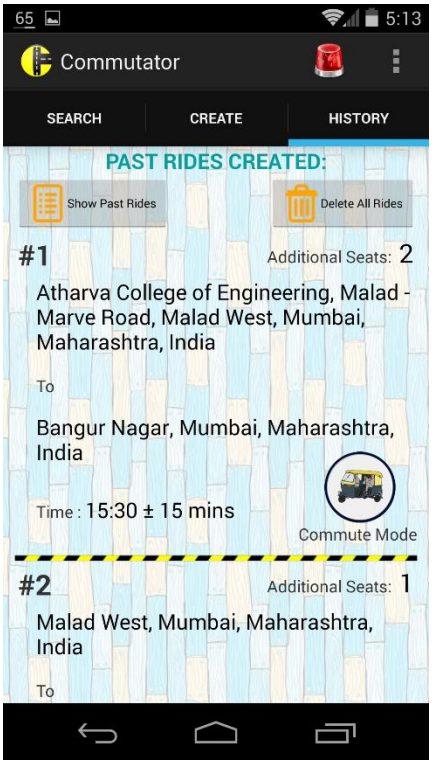


FIGURE 5.13: RIDE CREATED HISTORY FRAGMENT

5.2.3 Ride Activity Module:

5.2.3.1 Ride details:

This fragment displays the details of the ride the user is currently a part of. This fragment contains a functional button. The creator of the ride will be displayed a “Delete Ride” button. While the fellow passengers of the ride will be displayed a “Leave Ride” button. These buttons perform the task specified by their name.

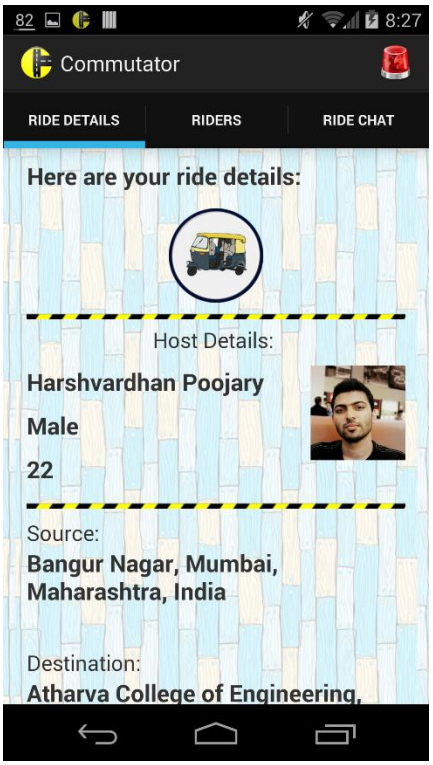


FIGURE 5.14: RIDE DETAILS FRAGMENT

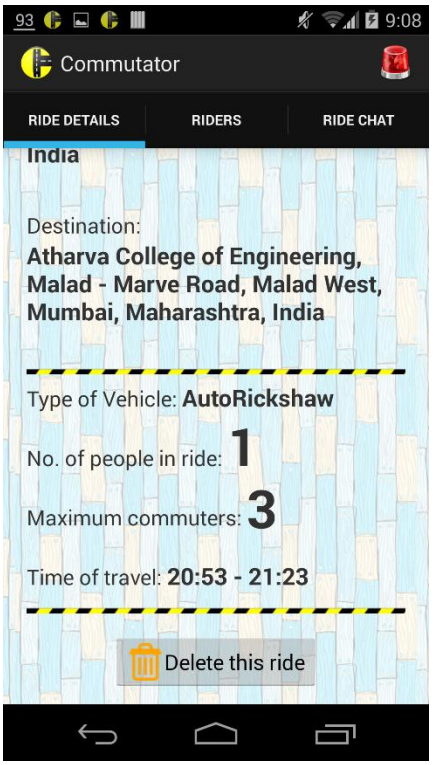


FIGURE 5.15: DELETE BUTTON

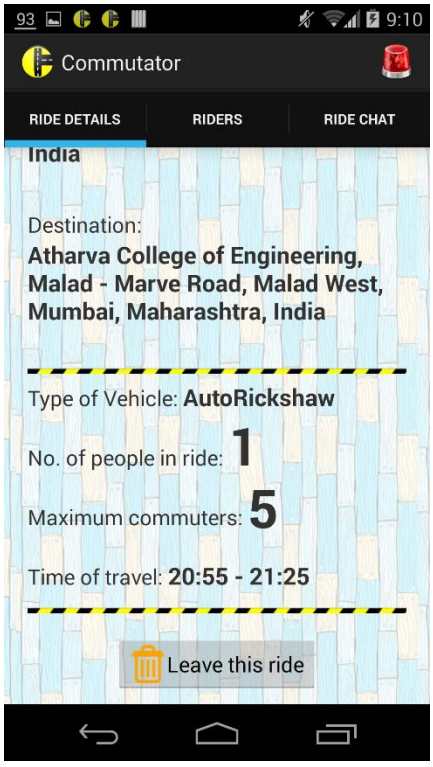


FIGURE 5.16: LEAVE BUTTON

5.2.3.2 Riders details:

This fragment displays the details of the fellow riders present in the ride. This fragment contains a functional button named “Refresh”, when pressed it performs a refresh on the list of riders i.e. fetches data of fellow commuters from the cloud backend. The list contains various details of the riders which are:

- Name
- Gender
- Age
- Time preference

FIGURE 5.17: RIDERS DETAILS FRAGMENT

5.2.3.3 Ride chat:

This fragment allows the users present in the same ride to communicate among themselves to sort out the execution of the ride. When the user signs up he is also a registered at QuickBlox, which is used as the chat backend. Every time the user logs in, he logs into QucikBlox too. The ride creator creates the chat room whose name is the same as the ride id. Commuters joining the ride are added

dynamically into the chat room. Once the ride expires or is deleted, the chat room's history is erased.

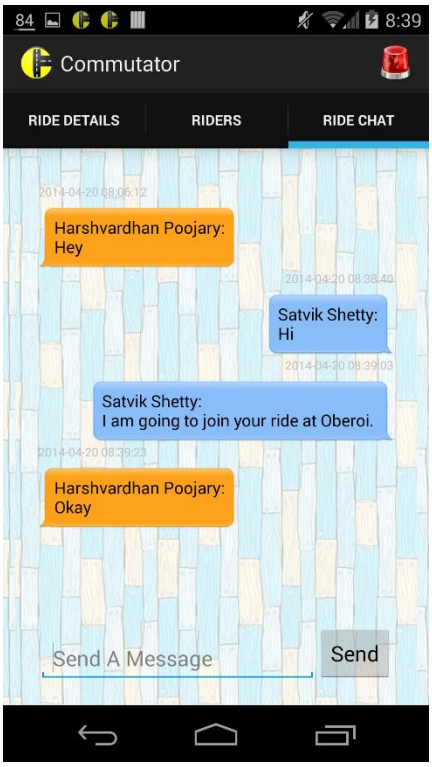


FIGURE 5.18: RIDE CHAT FRAGMENT

	ID	Login	Full Name	Email
	947282	**ameya (Account Owner)**	Ameya Deshmukh	ameya03deshmukh@gmai...
☐	1004672	harsh		
☐	1004713	sat		
☐	1004771	cpharsh		
☐	1004801	goku		

FIGURE 5.19: QUICKBLOX USER TABLE

Room JID	Room Name	Participants	History	Info
8372_asdf1@muc.chat.quickblox.com		0	view	view
8372_harsh1@muc.chat.quickblox.com		0	view	view
8372_mlyelccvkm@muc.chat.quickblox.com	8372_mlyelccvkm	1	view	view
8372_rides1@muc.chat.quickblox.com	rides1	0	view	view
8372_rides2@muc.chat.quickblox.com	rides2	0	view	view
8372_rides3@muc.chat.quickblox.com	rides3	0	view	view
8372_rides4@muc.chat.quickblox.com	rides4	0	view	view
8372_rides@muc.chat.quickblox.com	Rides	0	view	view
8372_sat1@muc.chat.quickblox.com		0	view	view
8372_trial1@muc.chat.quickblox.com		1	view	view

FIGURE 5.20: QUICKBLOX CHAT ROOM TABLE

5.2.4 Security Module:

The user is equipped with a security feature known as SOS. It allows the user to store a mobile number, to which a text message can be sent in case of danger. The text message is customizable. To send a text the user must press the beacon button present at right of the action bar. Safety tips has also been integrated into the application.

FIGURE 5.21: SET SOS MESSAGE

FIGURE 5.22: SET SOS NUMBER

FIGURE 5.23: SAFETY TIPS

TESTING AND TEST CASES

6. Testing and Test Cases:

6.1 Testing:

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation.

Test techniques include, but are not limited to the process of executing a program or application with the intent of finding software bugs (errors or other defects).

Software testing can be stated as the process of validating and verifying that a computer program/application/product:

- Meets the requirements that guided its design and development.
- Works as expected.
- Can be implemented with the same characteristics.
- Satisfies the needs of stakeholders.

Software testing, depending on the testing method employed, can be implemented at any time in the software development process. Traditionally most of the test effort occurs after the requirements have been defined and the coding process has been completed, but in the Agile approaches most of the test effort is on-going. As such, the methodology of the test is governed by the chosen software development methodology.

In our case, we have chosen the Agile model for the development of our application. Thus the testing is carried out after integration of every model, no matter how small they might be, to verify the proper functioning of the application.

Testing mobile applications is different and more complex than testing traditional desktop and web applications. A comprehensive mobile testing strategy includes device and network infrastructure, optimized selection of target devices, and an effective combination of manual and automated testing tools to cover both functional and non-functional testing.

6.1.1 White Box Testing

White-box testing is a method of testing the application at the level of the source code. White-box test design techniques include:

- Control flow testing
- Data flow testing
- Branch testing
- Path testing
- Statement coverage
- Decision coverage

White-box testing is the use of these techniques as guidelines to create an error free environment by examining any fragile code. These White-box testing techniques are the building blocks of white-box testing, whose essence is the careful testing of the application at the source code level to prevent any hidden errors later on.

The whole point of white-box testing is the ability to know which line of the code is being executed and being able to identify what the correct output should be.

In white box testing we make sure that the system that we built is being fully exercised thus ensuring that it is performing the functions for which it has been programmed.

In our system we have interfaced various methods and we have to make sure that this interface is visible to the user. Here our main and sole aim is to exercise all the technical capabilities of the system that we have built.

6.1.2 Black Box Testing

The technique of testing without having any knowledge of the interior workings of the application is Black Box testing. The tester is oblivious to the system architecture and does not have access to the source code.

Typically, when performing a black box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.

Black-box testing treats the software as a "black box", examining functionality without any knowledge of internal implementation. The testers are only aware of what the software is supposed to do, not how it does it. Black-box testing methods include: equivalence partitioning, boundary value analysis, all-pairs testing, state transition tables, decision table testing, fuzz testing, model-based testing, use case testing, exploratory testing and specification-based testing.

In black box testing we test the system at random for some random functionalities and depending on the output that we get we come to the conclusion that whether the system we have built is right or wrong. For testing we followed 3rd person testing system. We make a 3rd person use the application and his/her feedback is reported.

6.2 Test Cases:

6.2.1 Test Case 1 : GUI

This test case tests the usability and response of the GUI against various actions performed on it.

Test Case	Action	Expected Result	Actual Result	Pass/ Fail
1	Click on Login	The application should accept the username and password and grant access to authorized user.	User is logged into the application	Pass
2	Click on Create Account	The application should display activity asking user details for registration.	A screen pops up asking user details.	Pass
3	Click on Forgot Password	The application should display activity asking user email.	A screen pops up asking user email details.	Pass

4	Click on SOS button	The application should send a message to the number, if stored.	A message is sent to the number set.	Pass
5	Click on Search	A screen displaying appropriate search results should open.	A screen pop up with search result	Pass
6	Click on Create My Ride	A screen giving ride created should appear and a notification is displayed.	The required activity opens and a notification is displayed	Pass
7	Click on Show Past Rides	List view should populate with past ride details	List populates with details	Pass
8	Click on Delete All Rides	Deletes all the populated past ride details	Rides are getting deleted	Pass
9	Click on Join	A screen giving ride details should appear and a notification is displayed	The required screen opens and a notification is displayed	Pass
10	Click on Delete Ride	The application should display a confirmation box and on ‘Yes’, the previous activity and a notification about ride been deleted should be displayed	The required screen opens and a notification is displayed	Pass
11	Click on Leave Ride	The application should display a confirmation box and on ‘Yes’, the previous activity should be displayed.	The required screen opens and a notification is displayed	Pass

TABLE 6.1: TEST CASE - GUI

6.2.2 Test Case 2: Functionality

This test case tests the functionality and working of the application.

Test Case	Action	Expected Result	Actual Result	Pass / Fail
1	Validation of login page.	The application should open the next activity.	The necessary activity opens when data is verified.	Pass
2	Incorrect /Incomplete details during login	A toast message suggesting incorrect/incomplete details is expected.	A toast suggesting incorrect/incomplete details pops up.	Pass
3	Sign Up details filled correctly	The user should be registered in the server database and email verification activity should open up.	The user is registered and directed to email verification activity.	Pass
4	Incorrect details during sign up.	A toast message suggesting incorrect & incomplete details is expected.	A toast suggesting incorrect/incomplete details pops up.	Pass
5	Validation for Forgot Password.	An email should be sent with a reset password link to the registered email address.	Email is sent with a reset password link.	Pass
6	Creating a Ride	A ride details are uploaded on the server database. User should be directed to Ride activity and should receive a push notification from server.	Ride details are uploaded. User is directed to Ride Activity. Commuters present in the ride get a push notification from server.	Pass
7	Searching a Ride	If matching ride is present in the cloud backend, needs to be displayed in the results.	Parameters are perfectly matched and respective relevant results are displayed.	Pass
8	Deleting Ride created history	Data from local database is erased.	All rows from the database are cleared.	Pass
9	Displaying Ride created history	Retrieve the data from local database and display it	Data is retrieved and displayed.	Pass
10	Deleting a Ride	The ride should be deleted from the server database. The user should be directed to home screen and fellow riders should get push notification from server	The ride is deleted from the server. The user is directed to home screen and gets a push notification.	Pass

11	Leaving a ride	The user details should be deleted from the ride data. Others present in the ride should be notified with a push notification and user should be directed to Main Activity	The user details are deleted from the ride. Fellow riders are sent a push notification. The user is directed to the main activity.	Pass
12	Joining a ride	The user details should be added to the ride data. Others present in the ride should be notified with a push notification and user should be directed to Ride Activity	The user details are added to the ride data. Fellow riders are sent a push notification. The user is directed to the Ride Activity.	Pass

TABLE 6.2: TEST CASE - FUNCTIONALITY

6.2.3 Check for internet connection

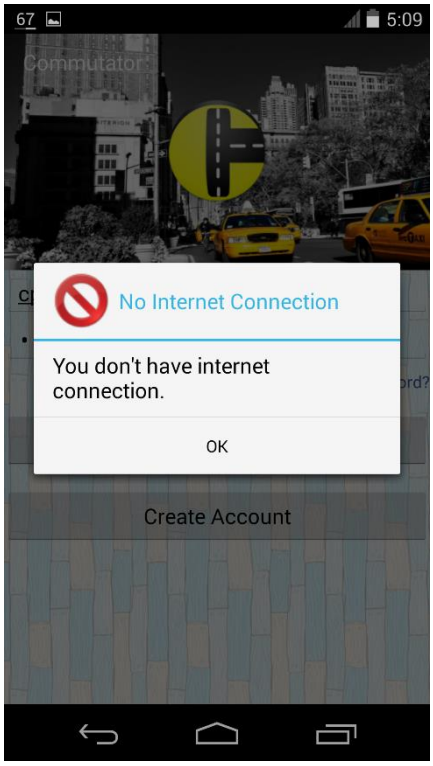


FIGURE 6.1 INTERNET CONNECTION CHECK

6.2.4 Check for empty fields

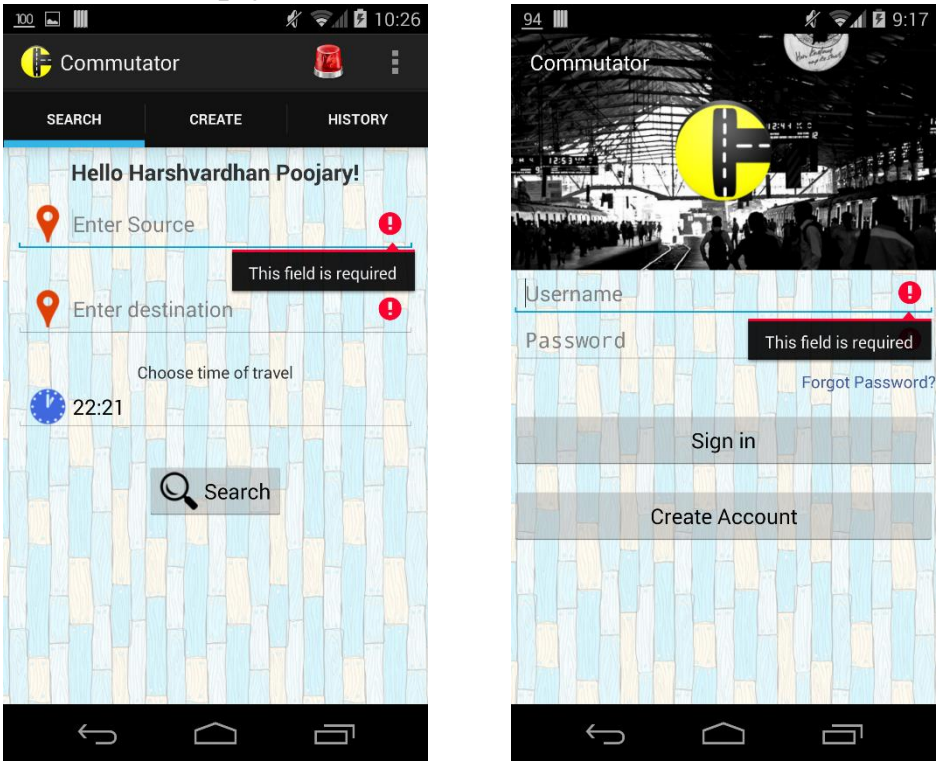


FIGURE 6.2: EMPTY FIELDS CHECK

6.2.5 Verification of valid username/password

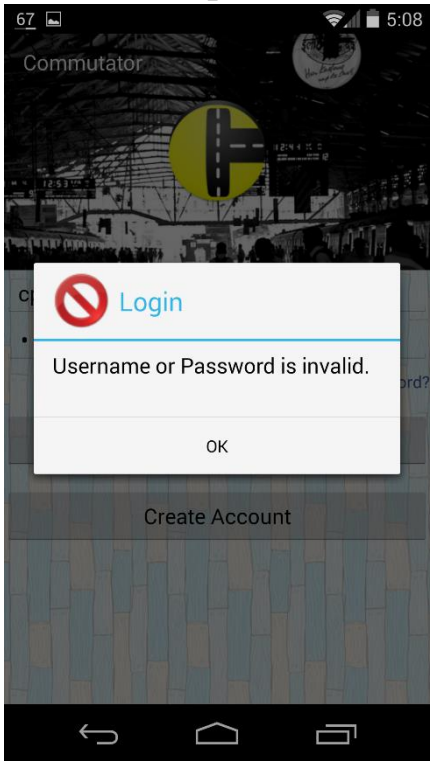


FIGURE 6.3: VALID USERNAME/PASSWORD CHECK

6.2.6 Email verification check

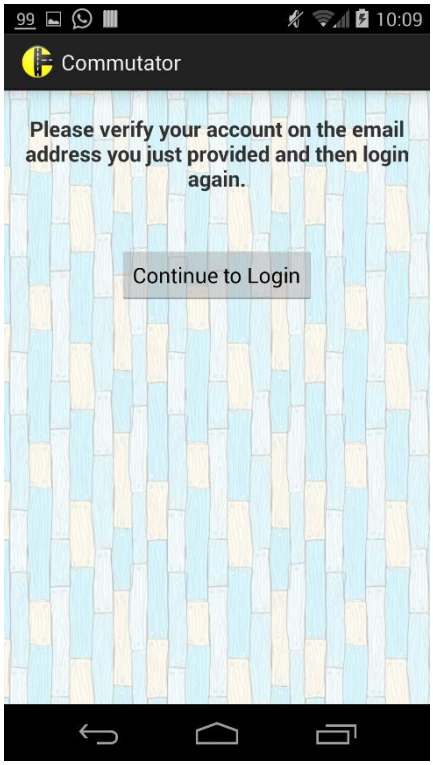


FIGURE 6.4: EMAIL VERIFICATION ON SIGNUP CHECK

6.2.7 Check for SOS number

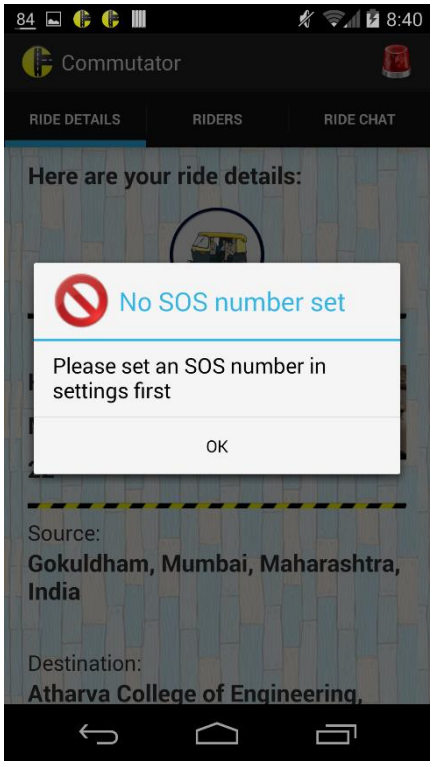


FIGURE 6.5: SOS NUMBER CHECK

6.2.8 Check for valid fields

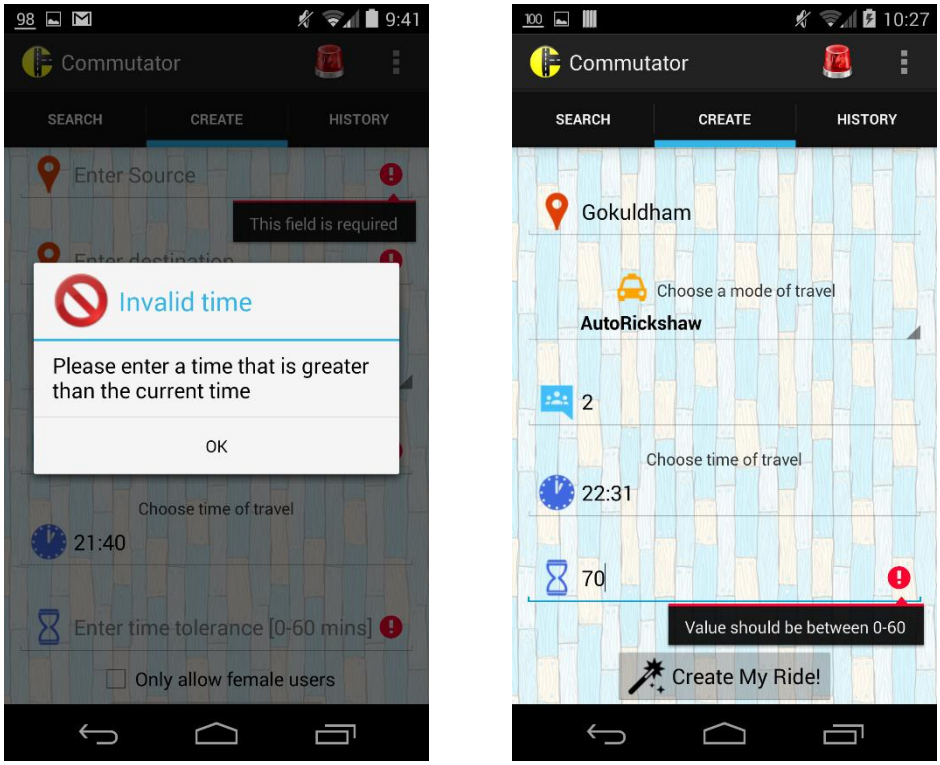


FIGURE 6.6: CHECK FOR VALID FIELDS

6.2.9 Check for female rides option

objectId String	username Strin...	password String	emailVerified Boolea...	Gender String	Name String
2EbMFMj8GT	sat	(hidden)	true	Female	Satvik Shetty

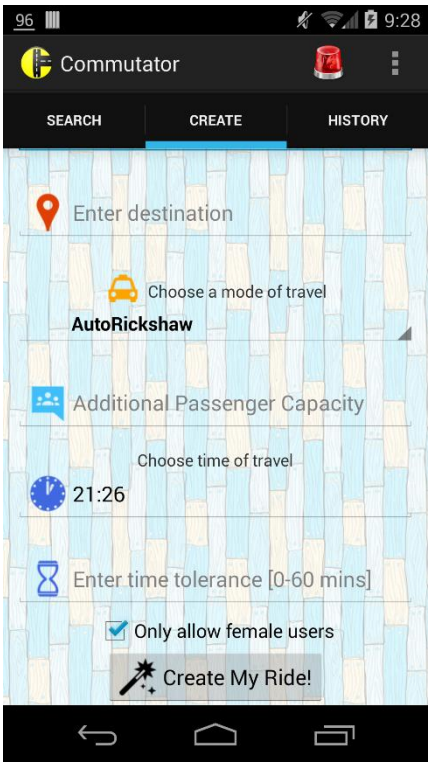


FIGURE 6.7: FEMALE RIDES OPTION CHECK

6.2.10 Check for no ride created history

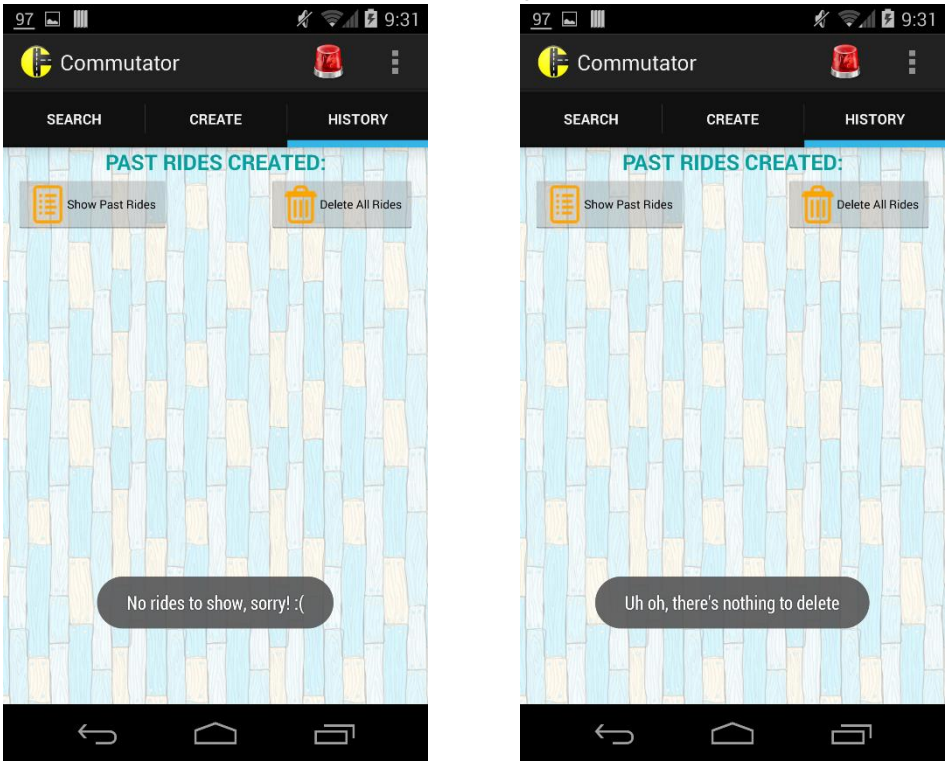


FIGURE 6.8: NO RIDE CREATED HISTORY CHECK

6.2.11 Exit confirmation check

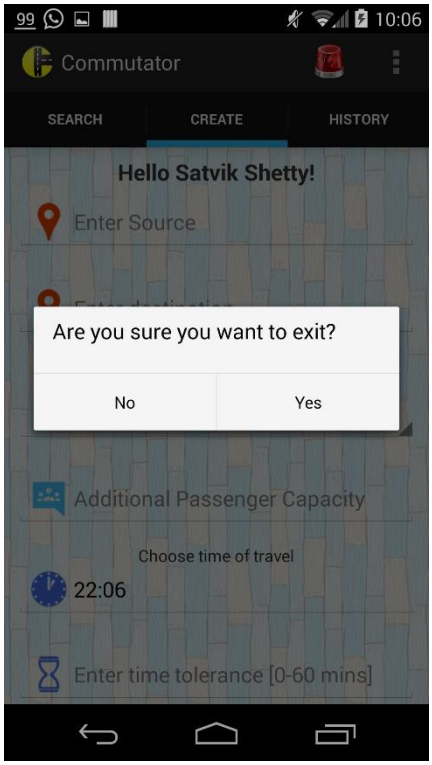


FIGURE 6.9: EXIT CONFIRMATION CHECK

TECHNOLOGY USED

7. Technology Used:

7.1 Description of Software/Server:

7.1.1 Eclipse

Eclipse is a community for individuals and organizations who wish to collaborate on commercially-friendly open source software. Its projects are focused on building an open development platform comprised of extensible frameworks, tools and runtimes for building, deploying and managing software across the lifecycle. The Eclipse Foundation is a not-for-profit, member supported corporation that hosts the Eclipse projects and helps cultivate both an open source community and an ecosystem of complementary products and services.

Released under the terms of the Eclipse Public License, Eclipse SDK is a free and open source software.

What does Eclipse provide?

In computer programming, Eclipse is an integrated development environment (IDE). It contains a base workspace and an extensible plug-in system for customizing the environment. Written mostly in Java, Eclipse can be used to develop applications. By means of various plug-ins, Eclipse may also be used to develop applications in other programming languages like Ada, ABAP, C, C++, COBOL, FORTRAN, Haskell, JavaScript, Lasso, Perl, PHP, Python, R,

Ruby (including Ruby on Rails framework), Scala, Clojure, Groovy, Scheme, and Erlang. It can also be used to develop packages for the software Mathematica. Development environments include the Eclipse Java development tools (JDT) for Java and Scala, Eclipse CDT for C/C++ and Eclipse PDT for PHP, among others. Users can extend its abilities by installing plug-ins written for the Eclipse Platform, such as development toolkits for other programming languages, and can write and contribute their own plug-in modules.

History of Eclipse:

The Eclipse Project was originally created by IBM in November 2001 and supported by a consortium of software vendors. The Eclipse Foundation was created in January 2004 as an independent not-for-profit corporation to act as the steward of the Eclipse community. The independent not-for-profit corporation was created to allow a vendor neutral and open, transparent community to be established around Eclipse.

Android Development Tools plugin for Eclipse:

We will be using ADT plugin for eclipse for developing the android application which includes interface as well as java programming. A vast majority of android apps owe their existence to eclipse. Google officially supports it, and has created the android development tools plugin for eclipse and integrated its AVD manager virtual device management into the tool as well.

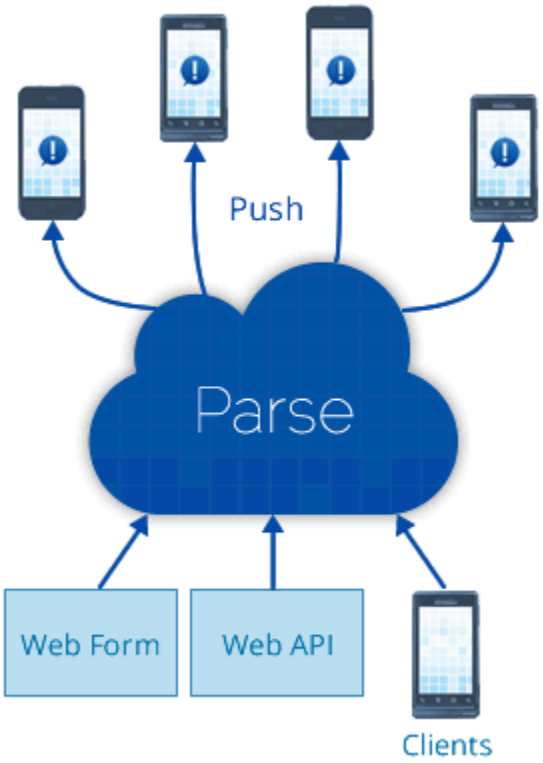
Android Development Tools (ADT) is a plugin for the eclipse IDE that is designed to give you a powerful, integrated environment in which to build android applications. ADT extends the capabilities of eclipse to let you quickly set up new android projects, create an application UI, add components based on the android framework API, debug your applications using the android SDK tools, and even export signed or unsigned apk files in order to distribute your application.

The AVD Manager:

The AVD Manager is an easy to use interface to manage your AVD (Android Virtual Device) configurations. An AVD is a device configuration for the android emulator that allows you to model different configurations of android powered devices. When you start the AVD manager on eclipse or run the android tool on the command line you will see the AVD manager.

7.1.2 Parse BaaS

Parse.com is a hosted service, providing backend services to end-user applications. Parse is the cloud app platform for Windows 8, Windows Phone 8, iOS, Android, JavaScript, and OS X. With Parse, you can add a scalable and powerful backend easily to support your applications. SDKs are available for a number of these platforms. In this case, we are using the Parse Android SDK in our application. An application can be made easily with the features available without worrying much about server management. It makes use of AWS, Ruby, Rails, MongoDB, Redis, MySQL, Chef, Nagios, Cacti, Capistrano, and HAProxy along with Java(Android SDK) and JavaScript.



FEATURES:

- **Parse Core**

  Save data in the cloud : Parse handles everything you need to store data securely and efficiently in the cloud. One can store basic data types, including locations and photos, and query across them with ease.

Make your app social : With Parse, we can connect our users via the traditional login or third party social networks such as Facebook or twitter with just a few lines of code. Linking accounts across networks, resetting passwords, and email verification can also be done.

Cloud Code : One can use the Parse JavaScript SDK to add custom validations and endpoints to make an app more powerful than ever. These will be run on the server to perform certain functions as requested by the developer.

Background Jobs : One can easily schedule recurring tasks like sending engagement emails, updating data, and long running computation at any time interval via the dashboard.

- **Parse Push**

  Push notifications are a direct channel to your app's users. One can keep them updated and engaged with app updates, promotions, and more sent directly to their device. Notifications can be to your whole user base, or a segment (through channels) based on the message being sent.

  The Push Console allows any user, technical or not, to send notifications directly from the Parse web interface. It allows you to preview your notifications in real time on the platforms you target.

  Pro and Enterprise users can take advantage of the push scheduling feature. One can schedule push notifications in advance without going through the hassle of updating the users when the task is done.

- **Parse Analytics**

  Measure App Usage : View your app open rates and API request data via the Parse dashboard. See real-time analytics on API requests based on REST verbs, device type, and Parse class being accessed.

  Powerful Dashboard : View all of your app usage, push campaigns, and custom analytics in the user-friendly Parse dashboard. Overlay graphs, filter by date, and more to gain insight into the effectiveness of your app.

7.1.3 QuickBlox MBaaS

QuickBlox is a MBaaS with multi dynamic cloud channel technology. Its structure is designed to support web applications, android applications, IOS7 applications, Linux and Blackberry platforms. The services are scalable and powerful. It makes use of AWS, Ruby, Rails, MongoDB, Redis, MySQL, Chef, Nagios, Cacti, Capistrano, and HAProxy along with Java (Android SDK) and JavaScript. In this app we are implementing their android SDK for chat services.

- Infinitely scalable, 99.5% uptime guarantee and powerful disaster recovery.
- All data is transmitted through 256 bit AES encryption. Hourly backups and built-in firewall.
- Daily stats indicating API usage for communication and data modules, breakdown by apps and user accounts.
- A monthly account review from your technical account manager including advice on infrastructure scale up/down.

Features:

- **1:1 chat** - Private user to user chat / IM
- **Group chat** - Unlimited chat rooms, either temporary or permanent
- **Incoming chat alerts** (push notifications) for offline users (via Messages)
- **File attachments** - cloud hosted via Content so both users don't have to be online for the transfer to take place. XMPP p2p file transfers are also supported. Allows users to send photos, videos and other files.
- **Location based chat** – integrated with Location, have your users chat over map (Google Map, Apple maps, Bing) and see distance to each other, hide or share their location, see internal or Facebook check-ins of other users.

- **Photos / Avatars** – Have users Facebook, Linkedin or custom profile photos next to their chat messages
- **Quoting** – display citations in chat room replies
- **AR chat** – chat in Augmented Reality view
- **Voice chat / call –** enable 1:1 voice calls in your app
- **Video chat recording** – record your video calling session
- **NAT traversal** – The TURN server and optimization system takes care of NAT traversal and traffic compression making sure your chat, voice and video traffic reaches all users across networks with different configurations

Chat services used is Chat are 2.0 grade. The chats and chat rooms are well secured with robust and complex encryption and decryption algorithms.

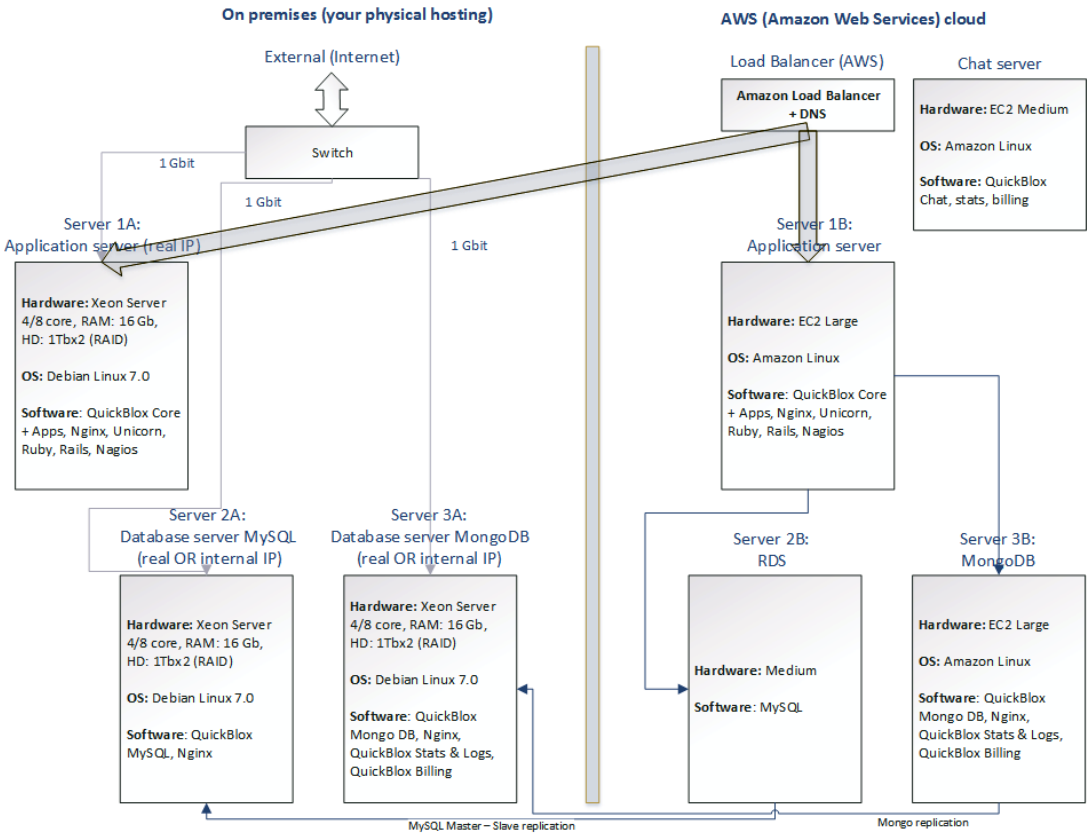


FIGURE 7.1: QUICKBLOX LOAD BALANCER

In this configuration QuickBlox has chosen to use AWS Load Balancer as a main load balancer. However, alternative configurations are possible including using Nginx / HAProxy for load balancing on premises or using schemes without conventional load balancers at all. Chat server is not duplicated. QuickBlox has chosen to go with Master -> Slave MySQL replication while MongoDB does a great job on replication itself.

7.2 Software Requirements:

- 32/64 bit Windows Vista or higher
- JDK 1.4 or higher
- Android SDK
- Eclipse IDE + ADT plugin

7.3 Hardware Requirements:

- **Mobile Requirements:**
  - Android 3.0 or higher
  - At least 10 Mb Storage space
  - 512 Mb RAM

- **Desktop Requirements:**
  - 1 GB RAM or higher
  - 1.8 GHz processor or higher
  - At least 1 GB Hard Disk space

FUTURE SCOPE

8. Future Scope:

• Trip Reminders
A push notification will be displayed to the user a fixed time (say 15 minutes) before the scheduled trip so as to notify him/her of the upcoming trip so as to lessen their chances of arriving late at the pickup point.

• Approximate cost estimation
One of the features that could set this application apart is the inclusion of an automatic cost estimation feature. For those sharing a private car one usually doesn't know how much he has to pay to the owner of the car. This can be calculated using the distance covered and the latest price of petrol/diesel.

• Commuter pickup points
In the near future, what we intend to bring to the table is that users can join the ride when they are in between the path along the source and destination. Different routes would have to be shown to the user who creates a ride. Based on the path he is taking, other people can join the ride, if they need to travel along the same route.

• Location Alarm
Often the user might not be from the city or might not know when his location will arrive. So the location alarm would use the GPS to inform the user when he is about to reach his set destination. This might also be useful in case the user falls asleep during his/her journey.

• Interstate journeys
This application could be evolved into one that could be used by travelers to travel between different cities so as to allow the people the freedom to create a shared ride without going to a travel agency and not waiting for a long period for people to share the ride with.

• Travel guide incorporation
The user can initially set his places of interest. Based on the location of the user, the nearby places of interest could be displayed. These could include cinemas, restaurants, museums, etc. Also people could find others with similar interests and go together in groups to sites around the city.

CONCLUSION

9. Conclusion:

Our main objective in creating this application is to help the user find a ride easily as well as cheaply. The endless wait for public transport vehicles is tiresome, and travelling by public transport has become tedious due to the increase in number of passengers, the endless traffic and the frequent transport strikes.

In 78% of all rides, the driver rides alone. So with the help of this application, the driver can utilize the empty seats by carpooling. This will help him share the cost of fuel. The users will be salvaged from the mirage about the public transport comfort level. It could also provide households with the opportunity to save money by reducing the number of cars they own, but without sacrificing convenience.

This application will help users combat the continuous high rise in travelling expenses by sharing the cost of travel with other users. Real-time ridesharing applications like ours will in turn help in assuaging traffic, help save fuel, money, time, and ultimately the environment and thus revolutionize the way we have been travelling in a population intense city like ours.

REFERENCES

10. References:

[1] Kirshner. D.:- Definition of Dynamic Ridesharing. http://dynamicridesharing.org/~dynami11/wiki/index.php?title=Definition_of_Dynamic_Ridesharing.

[2] Steger-Vonmetz .Improving modal choice and transport efficiency with the virtual ridesharing agency. D.C. Intelligent Transportation Systems, 2005. Proceedings. 2005 IEEE

[3] Transportation Demand Management Encyclopedia entry on Ride Sharing: http://www.vtpi.org/tdm/tdm34.htm.

[4] Real-time Ridesharing, Wikipedia, http://en.wikipedia.org/wiki/Real-time_ridesharing

[5] Rimantas Benetis, S. Jensen, Gytis Karciauskas, and Simonas Saltenis, "Nearest and reverse nearest neighbor queries for moving objects", The VLDB Journal 15, 3, pp. 229-249, 2006

[6] J. Yuan, Y. Zheng, L. Zhang, X. Xie, and G. Sun, "Where to find my next passenger", In Proc. of UbiComp 2011, ACM, 109-118.

[7] Amber Levofsky and Allen Greenberg, "Organized dynamic ride sharing: The potential environmental benefits and the opportunity for advancing the concept", Paper No. 01-0577, Transportation Research Board, 2001 Annual Meeting.

[8] Using the Parse SDK for Android, https://www.parse.com/docs/android_guide.

[9] Debugging and coding issues discussed and solved, https://www.stackoverflow.com.

[10] Understanding quickblox server and learn to send and retrieve chat messages, quickblox.com /docs/android_api

[11] Understanding the xmpp chat severs and services, https://xmpp.org/xmpp-software/servers/

[12] Understanding the Basics of Backend as a Service (BaaS), http://mobile.siliconindia.com/news/Understanding-the-Basics-of-Backend-as-a-Service-BaaS-nid-126045.html

[13] Backend as a service, Wikipedia, http://en.wikipedia.org/wiki/Backend_as_a_service

[14] Libraries and SDK borrowed from, https://www.github.com.

[15] Software testing, Wikipedia, http://en.wikipedia.org/wiki/Software_testing.

[16] Android Tutorials, https://www.Lynda.com, https://www.newBoston.com.

[17] Android Application Development and APIs, http://developer.android.com