2017

Mobile Application For Shipping Goods For Individuals And Truckers In India

Sendurr Selvaraj
University of South Carolina

MOBILE APPLICATION FOR SHIPPING GOODS FOR INDIVIDUALS AND TRUCKERS IN INDIA.

by

Sendurr Selvaraj

Bachelor of Engineering
Anna University, 2010

Submitted in Partial Fulfillment of the Requirements

For the Degree of Master of Science in

Computer Science and Engineering

College of Engineering and Computing

University of South Carolina

2017

DEDICATION

I would like to dedicate this thesis to my parents, brother and friends. Thank you very much for all of your support and encouragement throughout my thesis work.

ACKNOWLEDGEMENTS

I wish to place a record of my profound gratitude and appreciation to my thesis advisor Dr. Jianjun Hu, who first introduced me to Mobile Application Programming and nurtured the instinct of innovation in me. He guided me to pick ideas which can be converted to business models and always encouraged me to think big.

I take it a privilege to express my sincere thanks to Dr. Yan Tong who readily accepted to be a part of my thesis committee member and provided constant suggestions and encouragement during my thesis work.

I am thankful to Dr. Song Wang who too readily accepted to be a part of my thesis committee member and extended guidance and timely support for me to carry out my thesis work successfully.

I express my sincere thanks to Dr. Srihari Nelakuditi, Graduate Director, Department of Computer Science and Engineering, for his moral support in carrying out my thesis work.

I express my sincere gratitude to the staff of Department of Computer Science and Engineering for their continuous support rendered with administrative work of my thesis work.

ABSTRACT

India is a vast country with majority of its cities and towns connected through roads. Road transportation contributes to 86% share of the freight transport of the country with trucking companies dominating the entire space. With growing economy and demands raising, the quality of service of the trucking company remains poor. The major reasons are unorganized practice and lack of transparency. Moreover, limited access for customers to reach out to truckers to transport their goods.

This thesis aims to create a platform for customers and truckers to realize their needs with a help of a mobile application. Customers can search for truckers nearest to their location based on their needs. In addition, customers can also post their transport requirements which can be viewed by truckers. Truckers have options to update their travel plan well in advance making sure they run on full capacity. The application captures customers' ratings for truckers thus building truckers' creditability and in-turn improving quality of service. The platform provides a transparent mode of communication between customers and truckers on finalizing prices and eliminating middlemen, who in reality would draw commissions. The scope of the application can be extended to advertisement feeds, deals and truck sales as a revenue generation model to bare its operational cost.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ANGULARJS	Angular Java Script
API	Application Program Interface
APNS	Apple Push Notification Services
DOM	Document Object Model
GCM	Google Cloud Messaging
GPS	Global Positioning System
HTML	Hypertext Markup Language
HTTPS	Hypertext Transfer Protocol Secure
JSON	JavaScript Object Notation
MVC	Model View Controller
RESTFUL	Representation State Transfer
SDK	Software Development Kit
SMS	Short Message Service

CHAPTER 1

INTRODUCTION

The structure of Indian trucking industry is a complex network. Customers move their goods entirely on third party players. The core actors who serve the customers are trucking company, brokers/agents and truck owners.

Trucking companies are the primary players handling customers' orders. They are responsible for loading the goods at source and unloading the goods at the destination. They regulate market prices and allocate goods on tender based bids to truckers.

Brokers/Agents are intermediates for customer and trucking company or trucking company and truck owners. They play an important role in business continuity of the trucking industry.

Truck owners are ground workforce members who move goods from source to destination. They either own a fleet of trucks or a single truck, primarily acting as truck drivers. They target demand specific destinations and agree upon to a common shipping price with the trucking companies.

The industry is a contingent network where all stakeholders are reliant on each other irrespective of the percentage of contribution to the logistic workflow. The customers have very little option to contact truck drivers directly, portraying the trucking industry as an inefficient service oriented and unorganized industry.

Since the industry is dominated by trucking companies, the scope for technology advancements do not exercise down to trucker owners. Trucking companies use technology for self benefit and mask truck owners from customers.

The objective if this thesis is to offers a cognitive approach for more transparent logistic network beneficial to customers and truck owners. The application will be user friendly such that customers can contact truck drivers and vice versa in a couple of steps.

1.1 BACKGROUND

The Indian trucking industry is an infancy in the use of technology. The prime factors are incorporation of technology increases shipping cost and poor skill levels. In-spite of such challenges, the industry is focused in incorporating low rate technology adoption systems, which is a promising sign.

Recently, the Indian trucking industry has seen many startup ideas such as Porter, Blowhorn, Shippr, LOTrucks and Cargoji, which bridge the gap between customer and the trucking industry. Their main focus is customer centric, thereby formulated their ideas to mobile applications which can be readily available to customers. Having said that, mobile applications are user friendly and helped improve the quality of service.

Despite gaining traction among customers, these startups do not address the concerns of truck owners. They operate as a small scale trucking company operating with their own fleet of drivers and mask truck owners from customers. Tying up with truck owners to operate full time, requires to have the the trucks occupied always incurring huge operational costs. As a result, customers indirectly pay a higher shipping prices when availing such services.

The main goal of the proposed thesis is to connect truck owners with customers. The application is more of a forum to post and search for shipping information. The truck owners are given freedom of choice to partially or completely operate with the customer base of the application.

1.2 MOTIVATION

The motivation behind developing such an application are the following:

1. Trucking Industrial Reforms.
    a. Organize the structure of the trucking industry by eliminating middlemen.
    b. Provide a transparent logistic forum for customers to interact with truck owners.
2. Economic Benefits.
    a. Elimination of middlemen reduces shipping cost benefiting customers as well as truck owners to agree upon a reasonable shipping cost.
    b. Lessens unloaded spaces of trucks. Truckers can find goods to meet small to large load spaces of their truck helping them run on full load.
3. Quality of Service.
    a. With rating systems available for customers to rate truck owners, truckers would take serious efforts to ensure customer satisfaction.
    b. Now that truck owners are the sole benefiter of the shipping cost, truckers can guarantee better well maintained trucks.
4. Technology Benefits.

    a. Better logistic management. Truck owners ranging from a few fleet of trucks to large can effectively manage their trucks with no extra cost.
    b. Real time shipment tracking using mobile's GPS location.

5. Social Benefit.
    a. Better quality of service reduces shipping delays helping quality of service of other industries who are dependent on the shipped goods.
    b. Uplift the life style of truck owners and driver.

1.3 PROBLEM STATEMENT

The highly unorganized Indian trucking industry has made shipping goods for customers a tedious task. Problems in finding a transparent logistic network has caused goods movement through multiple stakeholders of the trucking industry. This has resulted in delay of loading, transporting and unloading of goods. Which further increases transportation cost to meet the cost demands of all the stakeholders in the network. Customers are forced to accept unfair pricing and hence the trucking community portrays a picture of bad quality of service and low customer satisfaction.

On the other hand, truck owners / drivers are suppressed by the industries upper dominant layer. Over 74% of truck owners own 1-5 truck, making it extremely difficult to operate their trucks independently without relying on large trucking companies. The major share of the amount paid by customers goes to trucking companies and brokers/agent, leaving a meager share to truck owners. Thereby being underpaid, truckers are forced to have unmaintained trucks causing frequent breakdowns and providing a poor quality of service.

CHAPTER 2

TRUCKDEAL MOBILE APPLICATION

2.1 PROPOSED SOLUTION

As mentioned in our problem statement, a holistic solution is needed to address problems faced by customers and truck owners in the trucking industry. The application should be very easy to use and reduce time and effort.

The proposed mobile application will be platform for customers and truck owners to post their shipping requirements. Customers have options to search for available truck owners near their current location. In addition, customers can post their shipping information which can be viewed by truck owners. Truck owners can also update their travel plans which can be viewed by an interested customer.

2.2 FEATURES

The features of the mobile application are as following:

- Search driver – This feature provides flexibility to customers to search for drivers to ship their goods. The application will use the customers current GPS location to search for nearby drivers. Customers can also specify their source location and destination location with pick date and time and the application will automatically look for drivers meeting the customer's

- inputs. The application also lists the results in a map view which helps to locate drivers with ease.
- Post a shipment – This feature allows customers to post their shipping information source location and destination location with pick date and time, information on the goods and expected cost. In addition, pictures of goods can also be attached to the shipping information.
- Search Customer – This feature enables drivers to search for customers for their source and destination. Details shared by the customer on the goods and the expected cost is also available to the driver.
- Post a trip – This features allows drivers to post their future trip plans in advance with available load capacity and goods preference. By doing so, drivers do not run short of goods on their travel dates.
- Order history – This features allows customers to view their list of shipping orders with status.
- Trip history – This feature allows drivers to view their list of trips. It acts as a small shipping management tool for truck owners.
- Fleet Management – Truck owners are also given addition option to feed in their list of trucks. This helps them to operate efficiently.
- Push Notification: The application will also feature a push notification notifying customers or truck owners on the current status of their shipments.
- Rating System – The application features a rating system to ensure quality of service rendered by truck drivers. Customers can rate and comment on

the truck driver's service, this helps building drivers' creditability for their future orders.

2.3 DESIGN AND DEVELOPMENT

Prior to building any application from scratch, it is important to plan the design flow of the application. The user interface design is pretty essential to know how the application is intended to work. Planning an application definitely involves some specific strategies. Many strategic questions come to the mind while designing and planning an application like there might be apps in the app store similar to your idea, what the need of such an application in the market is, what features make your app unique, what will make the people use your app and not the ones designed by others, how much time the application will take to develop, what user requirements either small or big should be considered, what category should the app fall into, etc.

After planning, next comes the process selecting the appropriate mobile application framework. This thesis work decided to be implemented using IONIC framework. IONIC framework is a hybrid mobile application development kit consisting simple HTML pages and JavaScript components using a Model View Controller (MVC) architecture. The key benefit in using IONC framework is that it helps us deploy our mobile application to any native mobile application platforms such as Android, IOS and Windows. There is no extra coding effort and the deployment is very simple and quick, enabling us to reach to wider network of mobile users.

2.4 APPLICATION WORKFLOWS

The application work flow is categorized basis the users who access the application. We have three users, customer, driver and an admin user. The customers are end users who use the application to search for drivers in order to transport their goods. Drivers are basic truck owners or truck drivers who use the application to search for customers and transport their goods. The admin user uses the application to verify the authentication of a driver user.

2.4.1 CUSTOMER PROFILE WORKFLOW

The customer registers with the application using the signup option during the first login and then on uses the login page. The application validates the login credential of the user and turns on the customer profile mode. In a customer profile mode, the application displays only the options which are designed specifically for a customer profile. After the login page, the user lands on the home page where the trip source and destination are entered. The next page capture the customers trip details such as sender, receiver information and the goods details.

At this stage all necessary information needed from the customer is available to the application. The applications algorithm looks for drivers matching the customer's trip information. The list of matched drivers is displayed to the customer. The customer can look into details of the drivers' trip information and based on his or her interest can select the driver to transport the goods. If any driver is selected by the customer, the driver is notified on the customer's interest and has an option to decline or confirm the customer's request. On the other side, when no matched drivers are found, the customer has an option to post his or her trip details which can be viewed by other interested drivers.

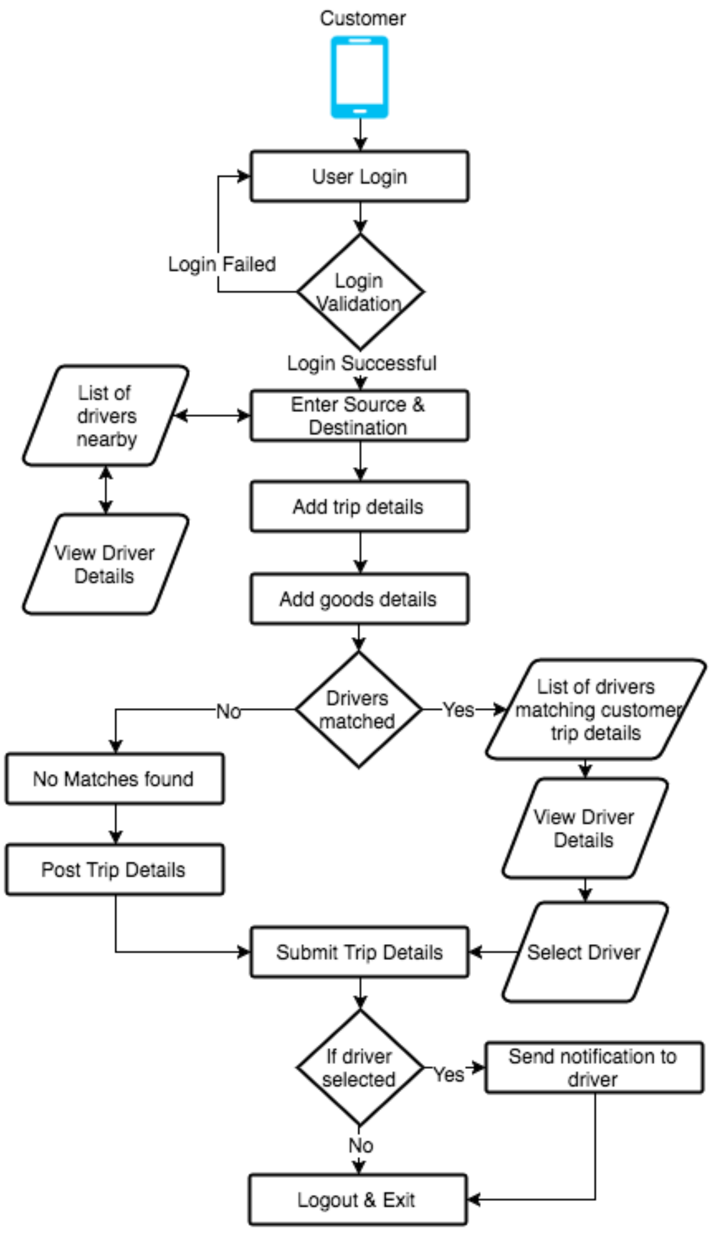


Figure 2.1: Customer Profile Workflow

2.4.2 DRIVER PROFILE WORKFLOW

The driver workflow is similar to customer workflow except with a few more verification checks. The driver registers with the application giving his or her information with a mandatory Driving License Number. The driver's account would be on hold until approved by an admin user. Meanwhile, the driver can add trucks, which will also be put on hold until approved by an admin user. Once the driver's profile is approved and has at least one approved truck, the driver can look out for customers using the home page. The home page lists the available customers nearby to the driver's location. The driver feeds in the trip source and destination information, attaches a truck from the list of approved trucks and the available load space with expected cost. The application's algorithm displays the list of customers which matches the driver's requirement. The driver can view the details of each customers and select a customer to transport their goods. A notification request is sent to the selected customer who can confirm or decline the request. Driver's also have an option to post their trip details if no customers are matched or if they are uninterested in the list of customers.

2.4.3 ADMIN PROFILE WORKFLOW

The admin workflow is a simple driver and truck management tool. Unlike driver profile and customer profile, the admin profile has no registration via the application. Admin profile registration a backend process by adding a new admin profile user. The home page has two tabs, drivers tab and trucks tab. The drivers tab list all driver profiles registered with application.

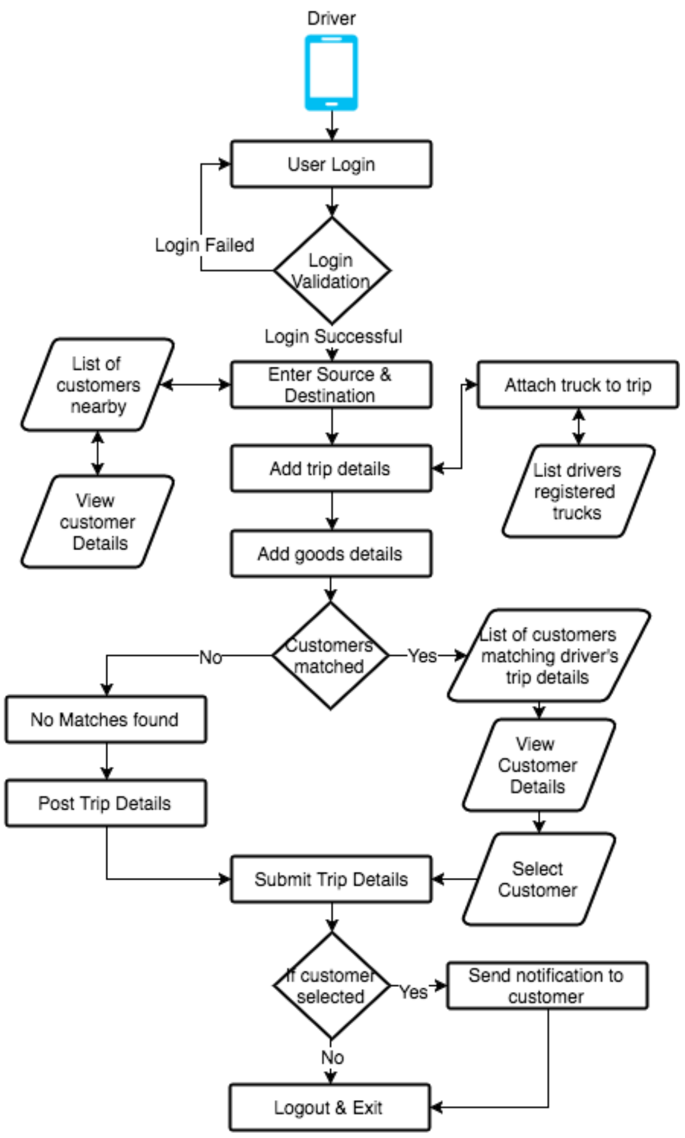


Figure 2.2: Driver Profile Workflow

The admin user can view details of each driver. On verifying the details of a driver, the admin user can approve the registration of the driver and a notification is end to the driver. The trucks tab is similar to driver's tab where the list trucks registered with the application are displayed. The admin user views the details of each truck and approves its registration. Once a truck is registered a notification is sent to the truck driver.

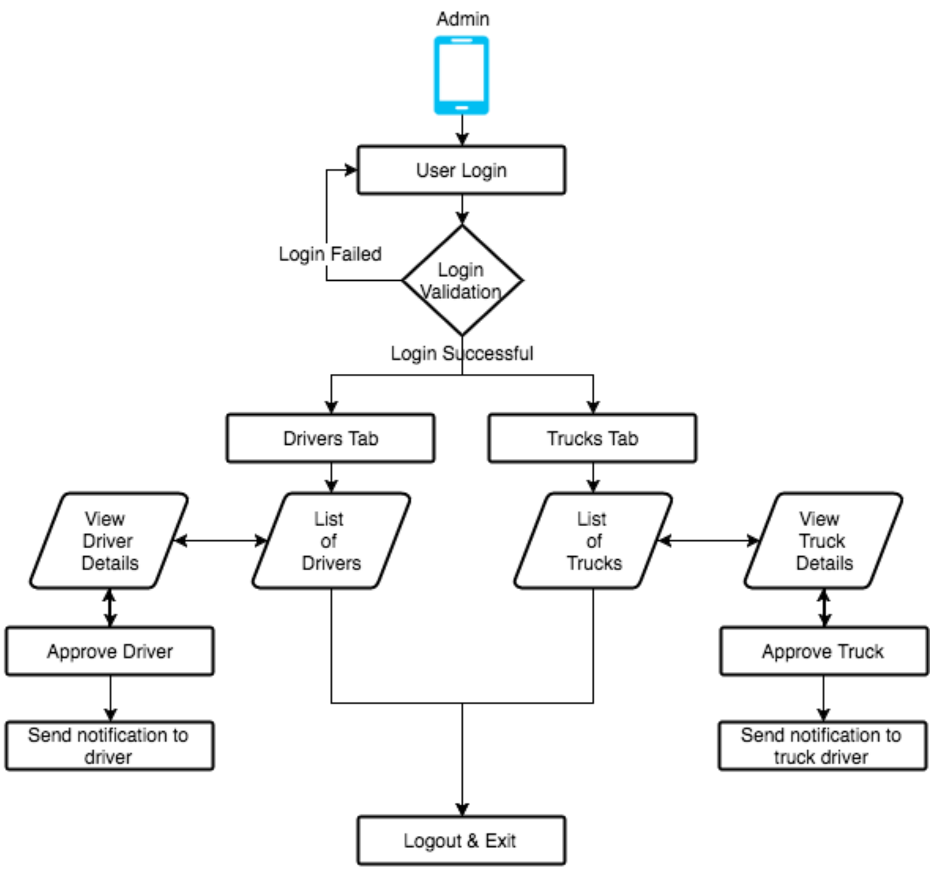


Figure 2.3: Admin Profile Workflow

2.5 LOGIN PAGE

The login page is a single entry point to the application. Every user needs to enter their mobile number and password as credentials to log into the application. The mobile number is treated as a username in the database.

The page has a signup link which calls the signup window in a pop up animation mode. The signup window supports driver and customer profile registration. The window is set to customer profile registration as default. Customer's feed in their first name, mobile number and password as mandatory fields.

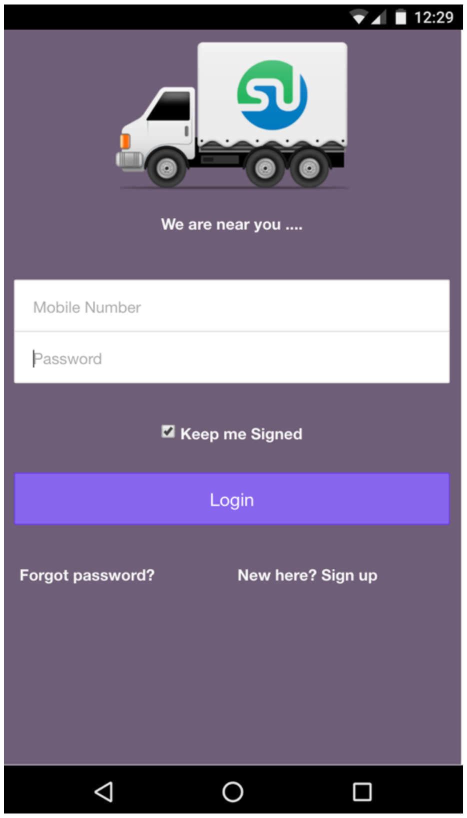


Figure 2.4: Login Page

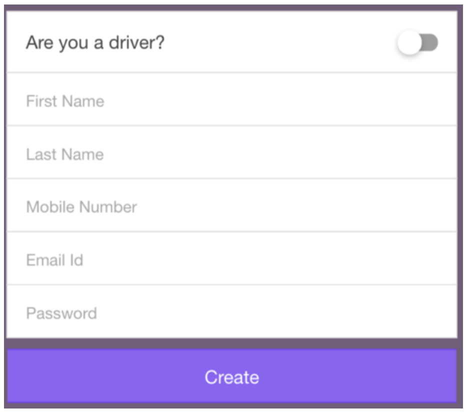


Figure 2.5: Customer Signup Window

A driver profile switch turns on driver profile creation mode with driver's date of birth, driving license number and picture of driving license as added mandatory inputs. The application uses the phone camera to capture and upload driver's driving license. Before processing profile creation, the input fields are validated for data integrity. A toast feature which is common in all mobile application platforms is incorporated to alert the user on a missing entry for a mandatory field.

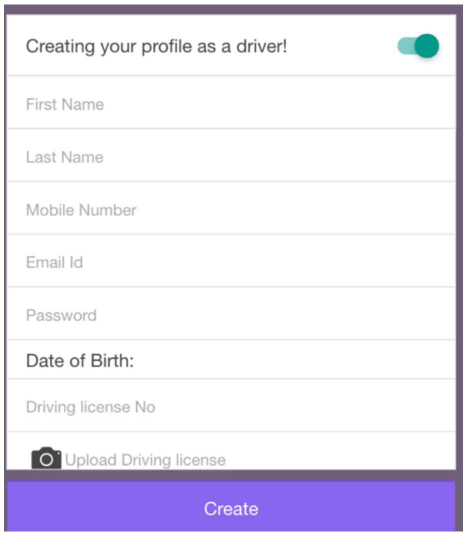


Figure 2.6: Driver Signup Window

The application is equipped with self credential storage using phones memory to hold user's credentials when a "keep me signed" option is selected. When the "keep me signed" option is selected and the users logs in the second time after signup, the credential are stored. Alternatively, when the "keep me signed" option is deselected, the stored credentials are cleared. This features enable faster and easy access to application.

2.6 MENU LAYOUT

The application is designed using menu layout. The menu contains links to other pages of the application and is hidden thereby not intervening with the applications home page as result it provides a convenient user interface. The user can click the menu button at any stage in the application to view the menus. The menu also displays a summary information of the logged in user.

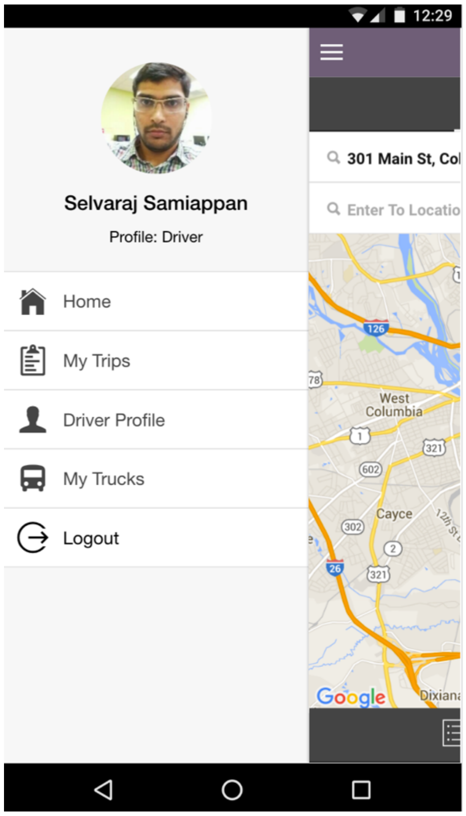


Figure 2.7: Application's Menu Layout

2.7 HOME PAGE

The home page is a common page both to customer profile and driver profile. The main objective of the home page is to collect the source and destination of the trip to be created by the user. The page has a map view and a list view. The application uses the mobile's GPS location to identify the current location of the user. The user's location is marked on to an interactive map. For a driver profile the map highlights nearby customers. Similarly, for a customer profile the map highlights nearby drivers. The list view is hidden by default and pops up when clicked by the user. The list view contains details of customers or drivers who are highlighted on the map. The source input field is auto populated with the address of the user's current location.

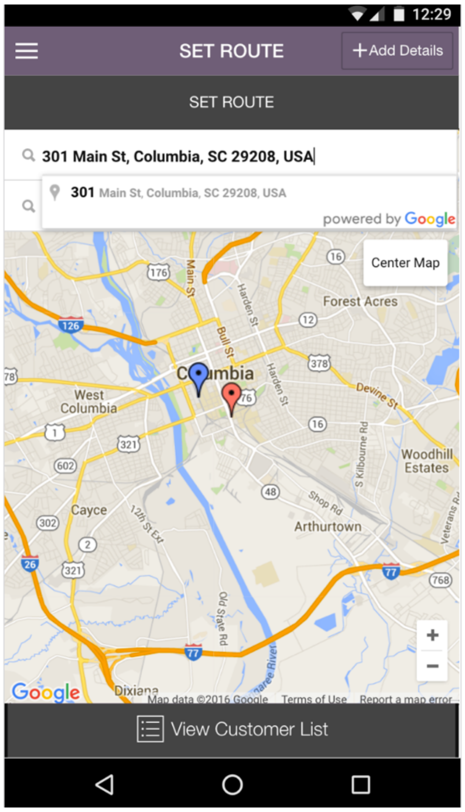
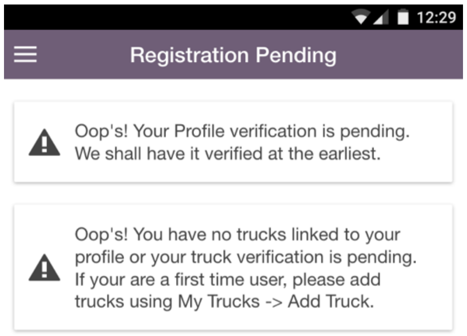


Figure 2.8: Home Page

Any changes to the map orientation like zoom in/out or map drag, the source field is populated with address of the center position of the map. The source and destination fields are enabled with Google Place services, which autocompletes the most suitable match basis the user's feed.



Oop's! Your Profile verification is pending. We shall have it verified at the earliest.

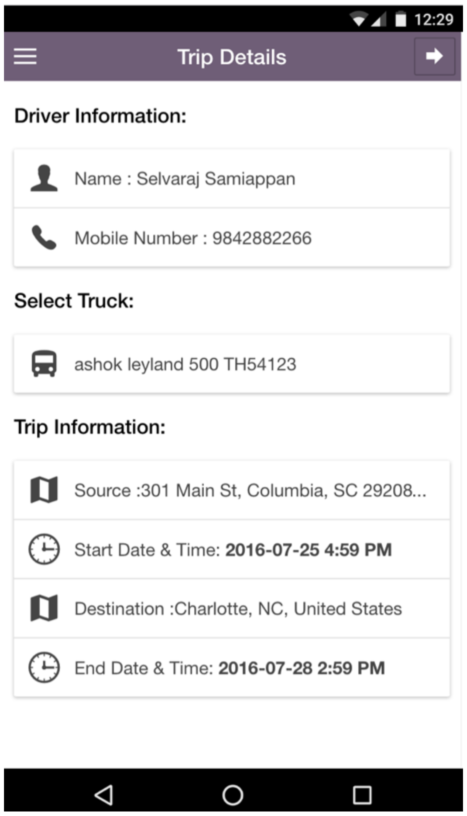
Oop's! You have no trucks linked to your profile or your truck verification is pending. If your are a first time user, please add trucks using My Trucks -> Add Truck.

Figure 2.9: Approval Pending Home Page

Alternatively, the map centers to display the place entered in the source field. The home page is disable for driver's profile which have their profile registration pending or have no approved trucks.

2.8 DRIVER TRIP DETAILS PAGE

The driver's trip details page consists of three pages. The first page display the source and destination entered by the driver in the home page. In addition, the driver is asked to select the start and end date & time of the trip. Validations are placed to make sure the start date & time is not past and the end date & time is not before the start date & time. The diver's list of approved trucks is listed to attach one truck to the trip being created.



Driver Information:

Name : Selvaraj Samiappan

Mobile Number : 9842882266

Select Truck:

ashok leyland 500 TH54123

Trip Information:

Source :301 Main St, Columbia, SC 29208...

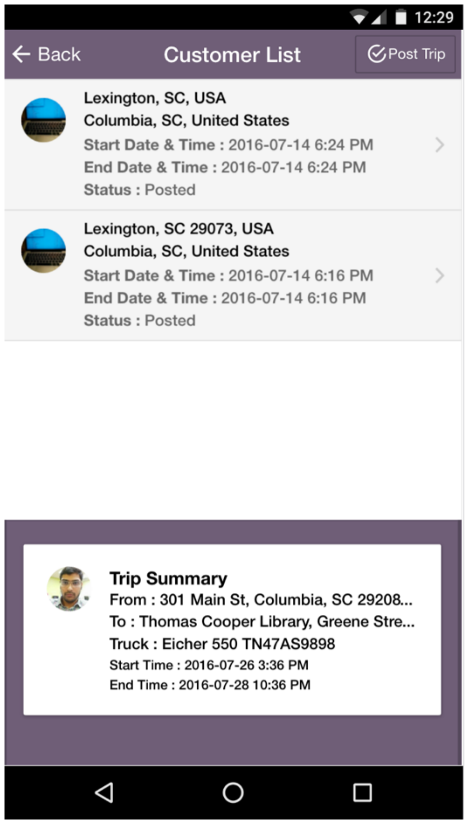
Start Date & Time: **2016-07-25 4:59 PM**

Destination :Charlotte, NC, United States

End Date & Time: **2016-07-28 2:59 PM**

Figure 2.10: Driver Trip Detail Page -1

The next page collects the goods information for the trip such as available load area in the truck, expected cost, any specific conditions can be highlighted in the comments and a picture of the truck load area can also be captured and uploaded. On the click of search for customer button, the mandatory input fields are validated and the application's algorithm displays the best match for the driver's search criteria. The driver has an option to view at the trip details of the matched customer and request the customer to transport their goods. In such instance the driver's trip status is set to requested and the selected customer's trip changes to 'please respond'. Once the customer accepts the request both customer's and driver's trip status is changed to confirmed.



Lexington, SC, USA
Columbia, SC, United States
Start Date & Time : 2016-07-14 6:24 PM
End Date & Time : 2016-07-14 6:24 PM
Status : Posted

Lexington, SC 29073, USA
Columbia, SC, United States
Start Date & Time : 2016-07-14 6:16 PM
End Date & Time : 2016-07-14 6:16 PM
Status : Posted

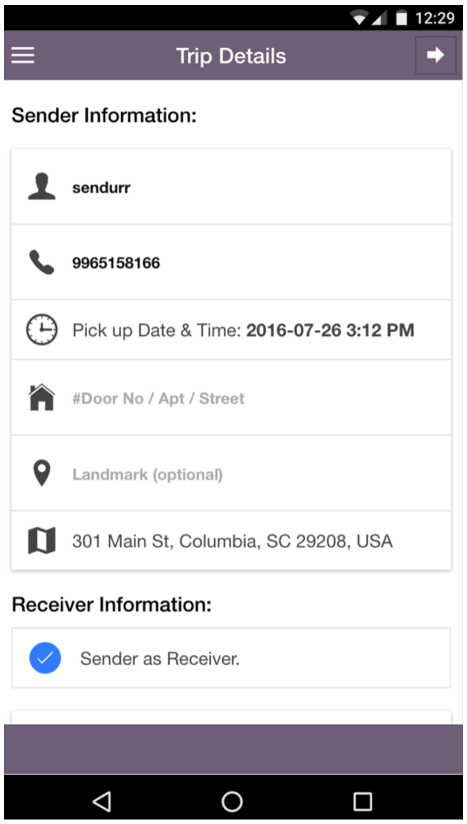
Trip Summary
From : 301 Main St, Columbia, SC 29208...
To : Thomas Cooper Library, Greene Stre...
Truck : Eicher 550 TN47AS9898
Start Time : 2016-07-26 3:36 PM
End Time : 2016-07-28 10:36 PM

Figure 2.11: Matched Customer's Trip for Drivers

If the driver is uninterested in the matched customer trips or there are no matched trips available, the driver has an option to post their trip. The posted driver's trip is available to customers searching for drivers to transport their goods. This option provides flexibility to drivers to plan ahead of trip helping them to run with a lesser unoccupied load space.

2.9 CUSTOMER TRIP DETAILS PAGE

The customer trip details page collects the sender's information and receiver information such as name, address and date& time. For a more user friendly interface, the sender's name is auto populated with first and last name of the logged in user from the profile information. Moreover, the page features a provision to auto populate the sender information in the receiver information. The source and destination information rom the home page is also displayed in a non-editable format.



Sender Information:

sendurr

9965158166

Pick up Date & Time: 2016-07-26 3:12 PM

#Door No / Apt / Street

Landmark (optional)

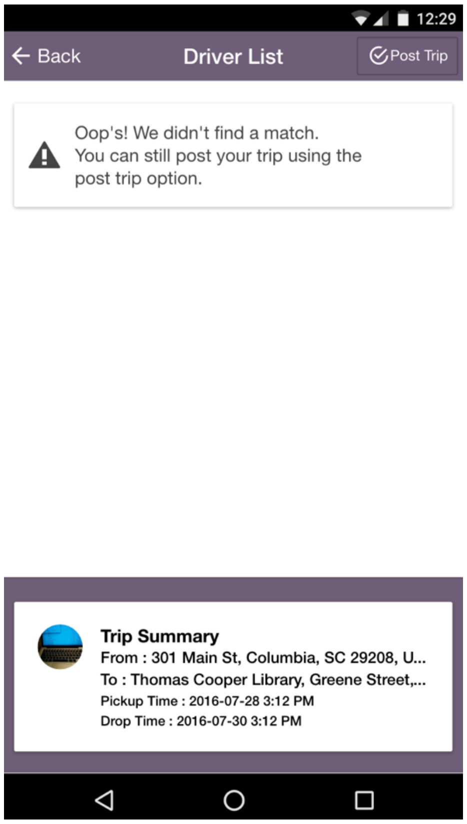
301 Main St, Columbia, SC 29208, USA

Receiver Information:

✓ Sender as Receiver.

Figure 2.12: Customer Trip Detail page – 1

Similar to the driver trip detail page, the next page captures the goods information of the customer. The type and weight of the goods are mandatory fields. The customer has an option to enter the expected transport cost, specific comments and upload the picture of the goods to be transported. When all mandatory fields are entered the customer can search for drivers. The application lists the matched drivers for the customer's request.



Oop's! We didn't find a match.
You can still post your trip using the post trip option.

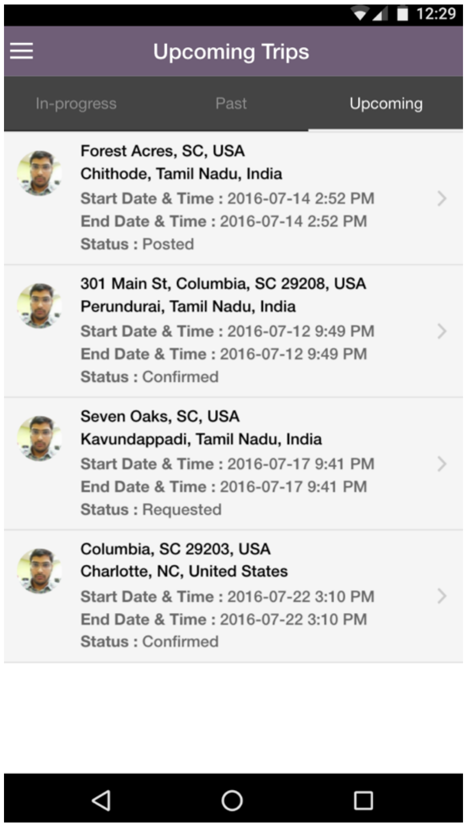
Trip Summary
From : 301 Main St, Columbia, SC 29208, U...
To : Thomas Cooper Library, Greene Street,...
Pickup Time : 2016-07-28 3:12 PM
Drop Time : 2016-07-30 3:12 PM

Figure 2.13: Matched Driver's List for Customers

The customer can view the details of the driver and if interested can send a request to the driver. At this stage the customer's trip status is set to requested and the driver's trip status is changed to "please respond". Once the driver accepts the request, both the driver's and customer's trip status is changed to confirmed. If the customer is uninterested in the listed driver's trip or if no matched driver's trip is found, the customer can post their trip. The posted trip is available for drivers to view and send out a request to transport the customer's goods.

2.10 MY TRIP PAGE

The "my trip page" displays all the trips created by the user. This page is common to both driver and customer profile users. The page has tab layout, with three tabs, In-progress, Past and Upcoming tab. The In-progress tab holds all the trips which are live such a trip started by the driver. The Past tab holds all the history trips such as cancelled and completed trips. The Upcoming tab holds all the future trips such as posted, confirmed and requested trips.



Forest Acres, SC, USA
Chithode, Tamil Nadu, India
Start Date & Time : 2016-07-14 2:52 PM
End Date & Time : 2016-07-14 2:52 PM
Status : Posted

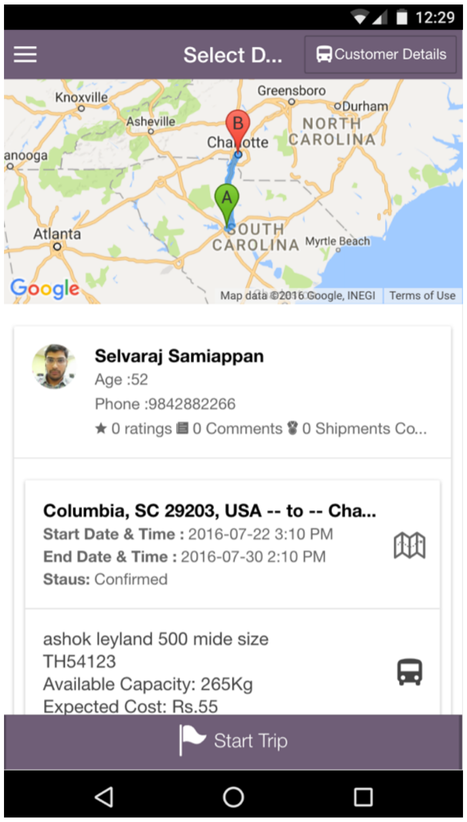
301 Main St, Columbia, SC 29208, USA
Perundurai, Tamil Nadu, India
Start Date & Time : 2016-07-12 9:49 PM
End Date & Time : 2016-07-12 9:49 PM
Status : Confirmed

Seven Oaks, SC, USA
Kavundappadi, Tamil Nadu, India
Start Date & Time : 2016-07-17 9:41 PM
End Date & Time : 2016-07-17 9:41 PM
Status : Requested

Columbia, SC 29203, USA
Charlotte, NC, United States
Start Date & Time : 2016-07-22 3:10 PM
End Date & Time : 2016-07-22 3:10 PM
Status : Confirmed

Figure 2.14: My Trip - Tab Page

The details of each trip can be viewed by clicking the trip. The detailed customer trip view displays the source, destination, pick up/drop date & time, sender information, receiver information and goods details. The detailed driver trip view displays the source, destination, start time & date, end date & time, age, driver rating/comments, attached truck and load area information. On clicking the rating/comment section, a detailed list of all ratings and comments given by customers for the driver are displayed. In addition, the source and destination are marked on a map with route on both the driver and customer detail trip views.



Selvaraj Samiappan
Age :52
Phone :9842882266
★ 0 ratings 0 Comments 0 Shipments Co...

Columbia, SC 29203, USA -- to -- Cha...
Start Date & Time : 2016-07-22 3:10 PM
End Date & Time : 2016-07-30 2:10 PM
Staus: Confirmed

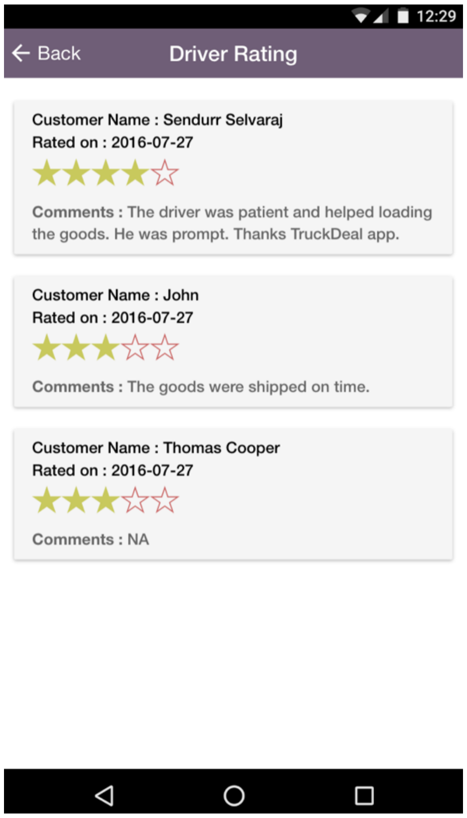
ashok leyland 500 mide size
TH54123
Available Capacity: 265Kg
Expected Cost: Rs.55

Figure 2.15: Driver Trip Detail Page

The driver's and customer's sensitive information such as sender, receiver information, contact details and truck registration number are hidden to either of the parties until a trip is confirmed by both the parties. The confirmed, in-progress, requested status driver trips contains a link to view the customer details and vice versa. Drivers are provided with option to start confirmed trip or complete an in-progress trip.

2.11 RATING PAGE

The rating page displays driver's rating and comments by customers. Customers can rate drivers only at the completion of a trip. Driver's can view their rating's through profile page and driver trip detail page. Whereas customers can view driver's rating during pre-selection and post-selection of the driver for their trip.



Customer Name : Sendurr Selvaraj
Rated on : 2016-07-27
★★★★☆
Comments : The driver was patient and helped loading the goods. He was prompt. Thanks TruckDeal app.

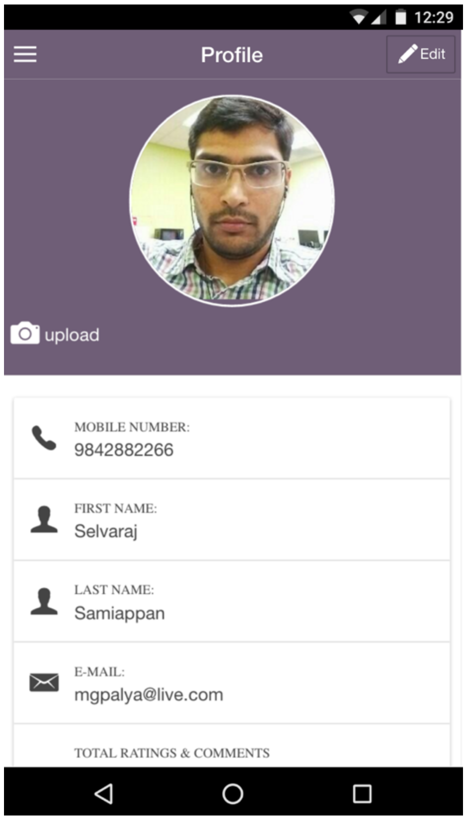
Customer Name : John
Rated on : 2016-07-27
★★★☆☆
Comments : The goods were shipped on time.

Customer Name : Thomas Cooper
Rated on : 2016-07-27
★★★☆☆
Comments : NA

Figure 2.16: Driver Rating Page

2.12 PROFILE PAGE

The profile page displays the logged in user's profile information. The page displays all the information entered by the user during sign up process. In addition to the sign up information, user's profile picture and address are also displayed. The profile picture can be uploaded by the user using the mobile camera. An edit option is available to edit the profile information. However, certain mandatory fields such as first name and mobile number are non-editable fields.



MOBILE NUMBER:
9842882266

FIRST NAME:
Selvaraj

LAST NAME:
Samiappan

E-MAIL:
mgpalya@live.com

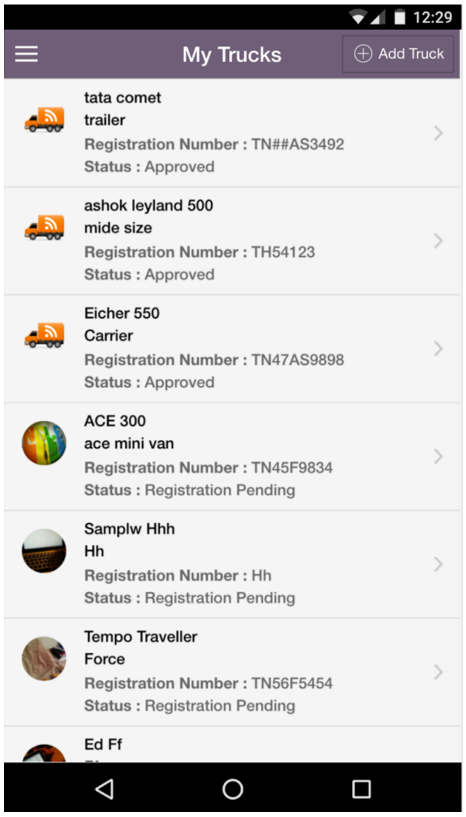
TOTAL RATINGS & COMMENTS

Figure 2.17: Driver's Profile Page

The driver profile has date of birth and driving license number as non-editable fields in addition. The driver's number of trucks, number of completed shipments, ratings and comments are available for the driver view in the profile page.

2.13 TRUCKS PAGE

The truck's page is a truck inventory page for truck drivers and available only to driver profile. The driver's can see all their trucks in a single list view and a detailed view. The page also hosts a truck create option to add new trucks. The add truck page collects truck's make, model, registration number, load area width/height/capacity, registration document picture, truck front view and truck load area picture. A new added truck is sent for admin's approval before it can be attached to the driver's trip. The application restricts editing of truck details to ones which have been approved.



tata comet
trailer
Registration Number : TN##AS3492
Status : Approved

ashok leyland 500
mide size
Registration Number : TH54123
Status : Approved

Eicher 550
Carrier
Registration Number : TN47AS9898
Status : Approved

ACE 300
ace mini van
Registration Number : TN45F9834
Status : Registration Pending

Samplw Hhh
Hh
Registration Number : Hh
Status : Registration Pending

Tempo Traveller
Force
Registration Number : TN56F5454
Status : Registration Pending

Ed Ff

Figure 18: Driver's Truck Page

CHAPTER 3

TECHNICAL SPECIFICATION

This chapter describes the technical specification of the mobile application. The application is build on IONIC mobile application framework with a Parse Backend database hosted on Parse Cloud Services. The following section details on technical composition of the application.

3.1 IONIC MOBILE FRAMEWORK

IONIC is an open source, front-end SDK for developing hybrid mobile apps with HTML5. Ionic provides mobile-optimized HTML, CSS, and JS components, as well as gestures and tools for building highly interactive apps. Ionic is performance efficient with its minimal DOM manipulation and hardware-accelerated transitions as compared to other frameworks in this league. Ionic uses ANGULARJS as its JavaScript framework. With the power of ANGULARJS inside a framework like Ionic, the possibilities are unlimited (you can use any ANGULARJS component inside Ionic as long as it makes sense in a mobile app). Ionic has a very good integration with Cordova's device API. This means that you can access device APIs using a library like ngCordova and integrate it with the beautiful user interface components of Ionic.

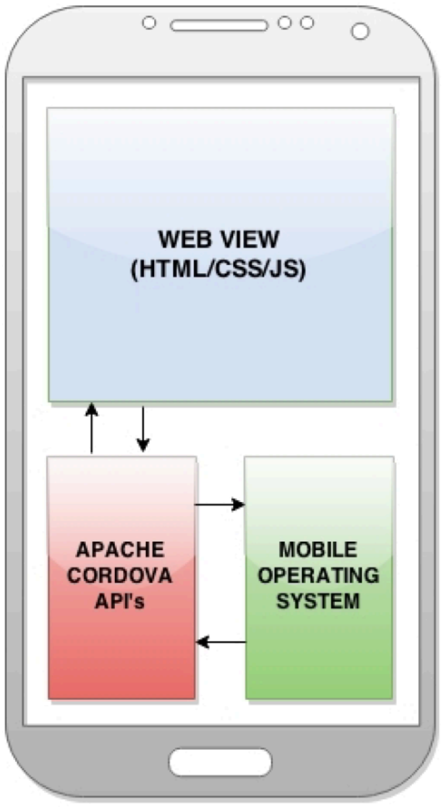


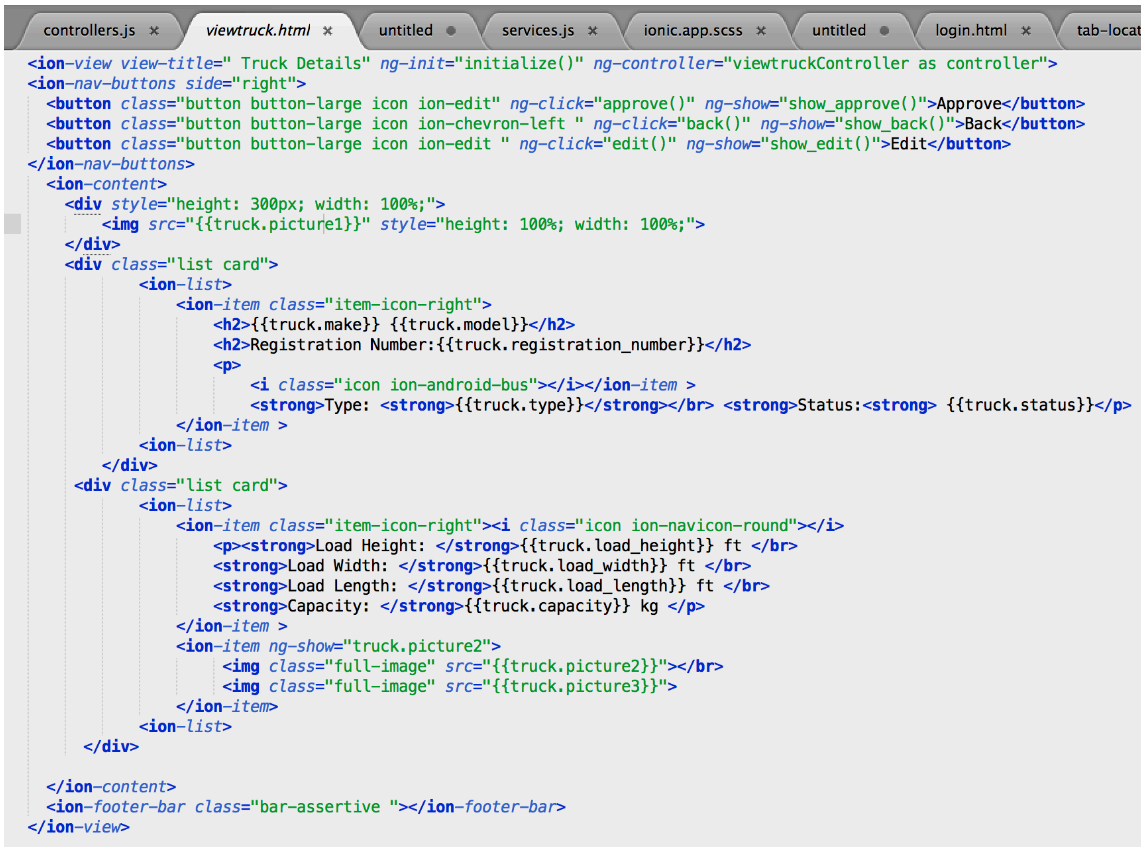
Figure 3.1: Hybrid Mobile Application Architecture.

3.1.1 ANGULAR JS

ANGULARJS is a structural framework for dynamic web apps. It lets you use HTML as your template language and lets you extend HTML's syntax to express your application's components clearly and succinctly. It uses client side Model View Controller (MVC) architecture. An IONIC application is broken down into five major pieces: Views, Controllers, Data (Services and Factories), App Configuration, and Directives. The Views, Controllers, and Data are the most recognizable pieces from a MVC perspective.

- VIEWS:

  A view is referred to a webpage or app page viewable on mobile. The view consists of templates which holds DOM elements. The DOM elements are bound to controller components and can be dynamically populated with values during application run time.

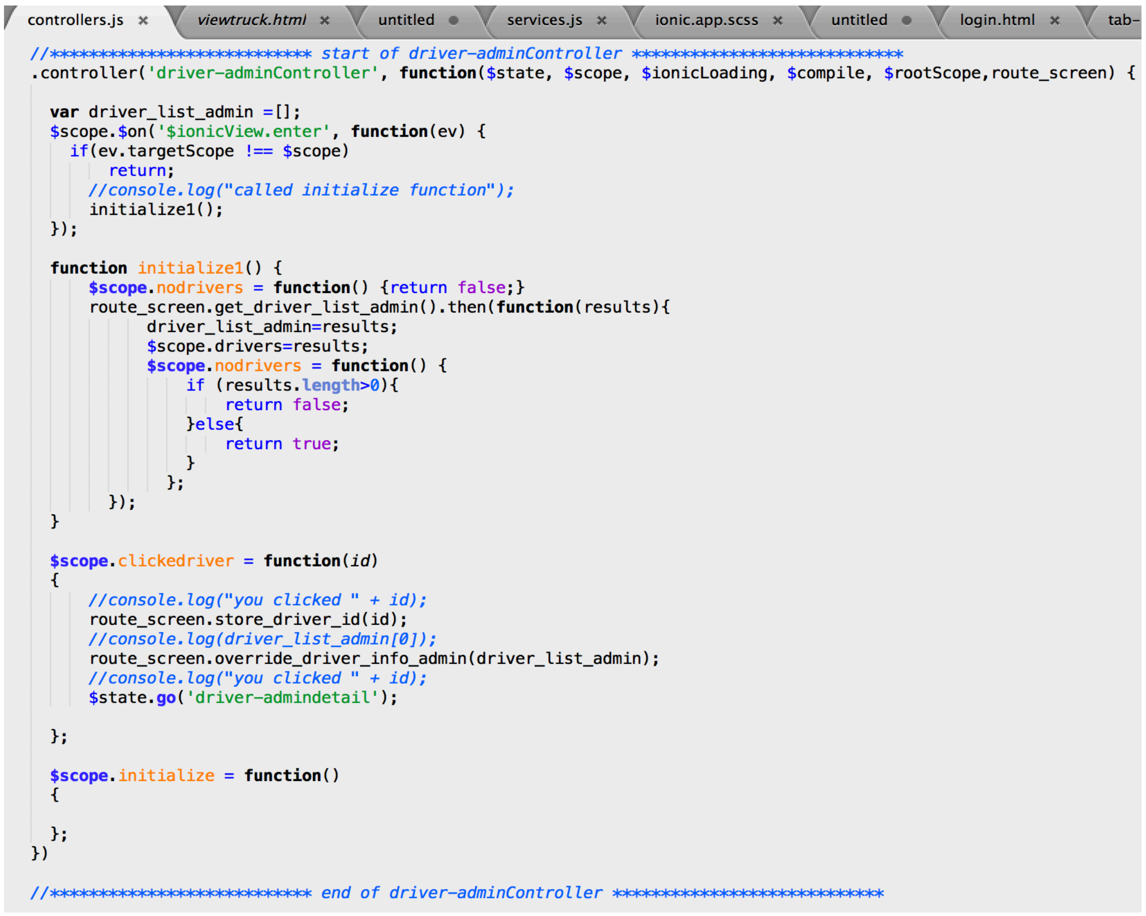


```html
<ion-view view-title=" Truck Details" ng-init="initialize()" ng-controller="viewtruckController as controller">
<ion-nav-buttons side="right">
  <button class="button button-large icon ion-edit" ng-click="approve()" ng-show="show_approve()">Approve</button>
  <button class="button button-large icon ion-chevron-left " ng-click="back()" ng-show="show_back()">Back</button>
  <button class="button button-large icon ion-edit " ng-click="edit()" ng-show="show_edit()">Edit</button>
</ion-nav-buttons>
  <ion-content>
    <div style="height: 300px; width: 100%;">
        <img src="{{truck.picture1}}" style="height: 100%; width: 100%;">
    </div>
    <div class="list card">
            <ion-list>
                <ion-item class="item-icon-right">
                    <h2>{{truck.make}} {{truck.model}}</h2>
                    <h2>Registration Number:{{truck.registration_number}}</h2>
                    <p>
                        <i class="icon ion-android-bus"></i></ion-item >
                        <strong>Type: <strong>{{truck.type}}</strong></br> <strong>Status:<strong> {{truck.status}}</p>
                </ion-item >
            <ion-list>
        </div>
     <div class="list card">
            <ion-list>
                <ion-item class="item-icon-right"><i class="icon ion-navicon-round"></i>
                    <p><strong>Load Height: </strong>{{truck.load_height}} ft </br>
                    <strong>Load Width: </strong>{{truck.load_width}} ft </br>
                    <strong>Load Length: </strong>{{truck.load_length}} ft </br>
                    <strong>Capacity: </strong>{{truck.capacity}} kg </p>
                </ion-item >
                <ion-item ng-show="truck.picture2">
                     <img class="full-image" src="{{truck.picture2}}"></br>
                     <img class="full-image" src="{{truck.picture3}}">
                </ion-item>
            <ion-list>
      </div>

  </ion-content>
  <ion-footer-bar class="bar-assertive "></ion-footer-bar>
</ion-view>
```

Figure 3.2: Screenshot of IONIC View/Template.

- CONTROLLERS:

Controllers are the main heart of the application, where flow of data and application logic is controlled. Each time a page/view is invoked, a controller is called at the backend. The controller uses a view as a template to display a page to the user and make calls to the data layer classes (factories/services) to get the actual data to bind to the template. The controller assigns this data to a $scope variable, which is then bound to the view. The $scope is an object that contains data defined by the controller used to build the view.

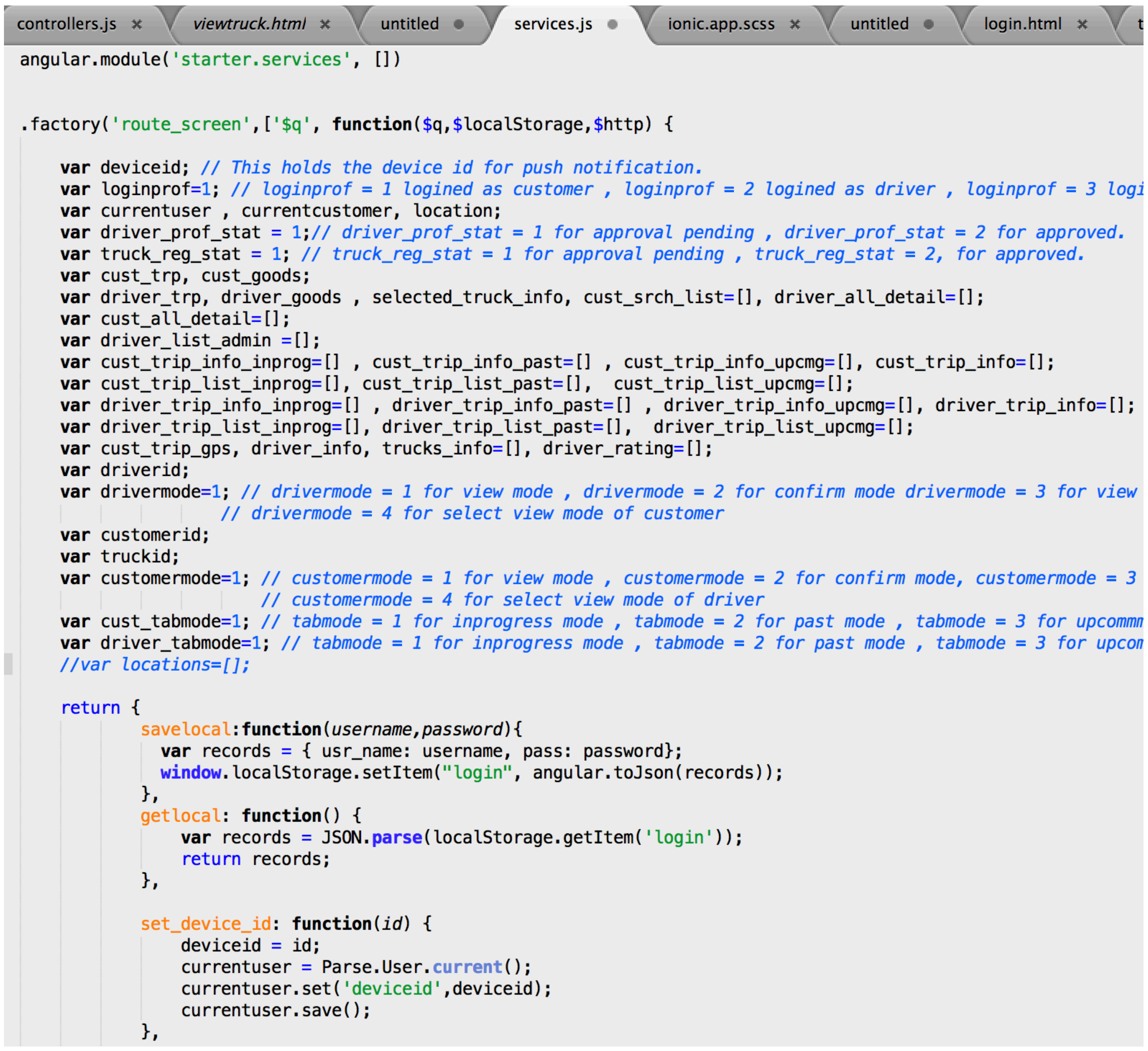


```
//**************************** start of driver-adminController ****************************
.controller('driver-adminController', function($state, $scope, $ionicLoading, $compile, $rootScope,route_screen) {

  var driver_list_admin =[];
  $scope.$on('$ionicView.enter', function(ev) {
    if(ev.targetScope !== $scope)
        return;
      //console.log("called initialize function");
      initialize1();
  });

  function initialize1() {
      $scope.nodrivers = function() {return false;}
      route_screen.get_driver_list_admin().then(function(results){
            driver_list_admin=results;
            $scope.drivers=results;
            $scope.nodrivers = function() {
                if (results.length>0){
                    return false;
                }else{
                    return true;
                }
              };
        });
  }

  $scope.clickedriver = function(id)
  {
      //console.log("you clicked " + id);
      route_screen.store_driver_id(id);
      //console.log(driver_list_admin[0]);
      route_screen.override_driver_info_admin(driver_list_admin);
      //console.log("you clicked " + id);
      $state.go('driver-admindetail');

  };

  $scope.initialize = function()
  {

  };
})

//**************************** end of driver-adminController ****************************
```

Figure 3.3: Screenshot of IONC Controller.

- DATA (FACTORIES / SERVICES):

The Data layer of an Ionic app is the provider of data, usually from a external backend or web service. The controller requests the data from the Data layer to use in binding to the view. These Data layer classes are known as Services or Factories.

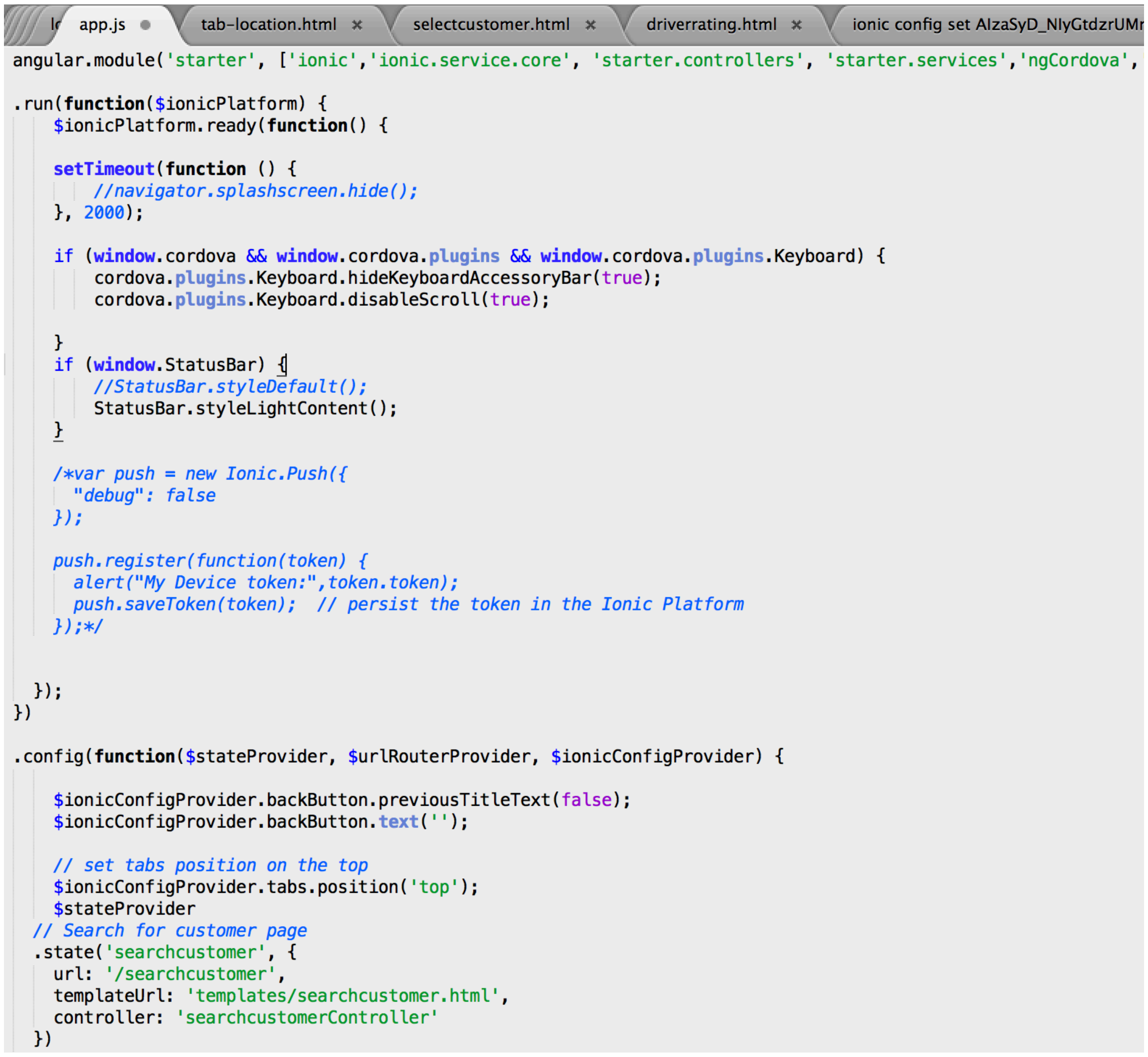


```
angular.module('starter.services', [])


.factory('route_screen',['$q', function($q,$localStorage,$http) {

    var deviceid; // This holds the device id for push notification.
    var loginprof=1; // loginprof = 1 logined as customer , loginprof = 2 logined as driver , loginprof = 3 logi
    var currentuser , currentcustomer, location;
    var driver_prof_stat = 1;// driver_prof_stat = 1 for approval pending , driver_prof_stat = 2 for approved.
    var truck_reg_stat = 1; // truck_reg_stat = 1 for approval pending , truck_reg_stat = 2, for approved.
    var cust_trp, cust_goods;
    var driver_trp, driver_goods , selected_truck_info, cust_srch_list=[], driver_all_detail=[];
    var cust_all_detail=[];
    var driver_list_admin =[];
    var cust_trip_info_inprog=[] , cust_trip_info_past=[] , cust_trip_info_upcmg=[], cust_trip_info=[];
    var cust_trip_list_inprog=[], cust_trip_list_past=[],  cust_trip_list_upcmg=[];
    var driver_trip_info_inprog=[] , driver_trip_info_past=[] , driver_trip_info_upcmg=[], driver_trip_info=[];
    var driver_trip_list_inprog=[], driver_trip_list_past=[],  driver_trip_list_upcmg=[];
    var cust_trip_gps, driver_info, trucks_info=[], driver_rating=[];
    var driverid;
    var drivermode=1; // drivermode = 1 for view mode , drivermode = 2 for confirm mode drivermode = 3 for view
                    // drivermode = 4 for select view mode of customer
    var customerid;
    var truckid;
    var customermode=1; // customermode = 1 for view mode , customermode = 2 for confirm mode, customermode = 3
                        // customermode = 4 for select view mode of driver
    var cust_tabmode=1; // tabmode = 1 for inprogress mode , tabmode = 2 for past mode , tabmode = 3 for upcommn
    var driver_tabmode=1; // tabmode = 1 for inprogress mode , tabmode = 2 for past mode , tabmode = 3 for upcon
    //var locations=[];

    return {
            savelocal:function(username,password){
              var records = { usr_name: username, pass: password};
              window.localStorage.setItem("login", angular.toJson(records));
            },
            getlocal: function() {
                var records = JSON.parse(localStorage.getItem('login'));
                return records;
            },

            set_device_id: function(id) {
                deviceid = id;
                currentuser = Parse.User.current();
                currentuser.set('deviceid',deviceid);
                currentuser.save();
            },
```

Figure 3.4: Screenshot of IONIC Services.

- APP CONFIGURATION:

The App Configuration is similar to a configuration file in an application. It holds the states of the application and their corresponding views/templates and controllers. The default lading view of the application is also configured here.

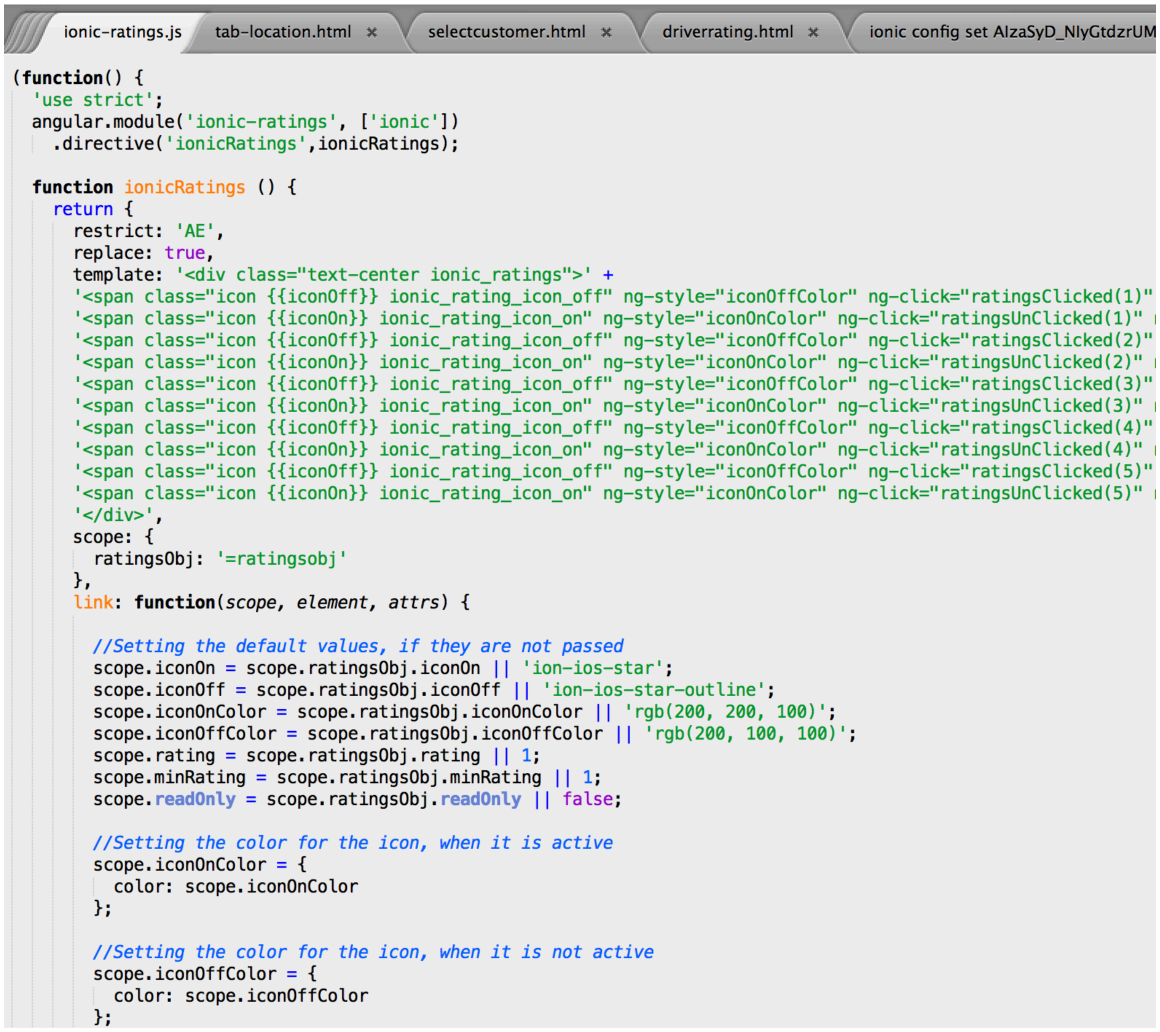


```
angular.module('starter', ['ionic','ionic.service.core', 'starter.controllers', 'starter.services','ngCordova',

.run(function($ionicPlatform) {
    $ionicPlatform.ready(function() {

    setTimeout(function () {
        //navigator.splashscreen.hide();
    }, 2000);

    if (window.cordova && window.cordova.plugins && window.cordova.plugins.Keyboard) {
        cordova.plugins.Keyboard.hideKeyboardAccessoryBar(true);
        cordova.plugins.Keyboard.disableScroll(true);

    }
    if (window.StatusBar) {
        //StatusBar.styleDefault();
        StatusBar.styleLightContent();
    }

    /*var push = new Ionic.Push({
      "debug": false
    });

    push.register(function(token) {
      alert("My Device token:",token.token);
      push.saveToken(token);  // persist the token in the Ionic Platform
    });*/


  });
})

.config(function($stateProvider, $urlRouterProvider, $ionicConfigProvider) {

    $ionicConfigProvider.backButton.previousTitleText(false);
    $ionicConfigProvider.backButton.text('');

    // set tabs position on the top
    $ionicConfigProvider.tabs.position('top');
    $stateProvider
  // Search for customer page
  .state('searchcustomer', {
    url: '/searchcustomer',
    templateUrl: 'templates/searchcustomer.html',
    controller: 'searchcustomerController'
  })
```

Figure 3.5: Screenshot of IONIC App Configuration File.

- DIRECTIVES:

Directives are markers on a DOM element (such as an attribute, element name, comment or CSS class) that tell ANGULARJS HTML compiler ($compile) to attach a specified behavior to that DOM element or even transform the DOM element and its children.



```
(function() {
  'use strict';
  angular.module('ionic-ratings', ['ionic'])
    .directive('ionicRatings',ionicRatings);

  function ionicRatings () {
    return {
      restrict: 'AE',
      replace: true,
      template: '<div class="text-center ionic_ratings">' +
      '<span class="icon {{iconOff}} ionic_rating_icon_off" ng-style="iconOffColor" ng-click="ratingsClicked(1)"
      '<span class="icon {{iconOn}} ionic_rating_icon_on" ng-style="iconOnColor" ng-click="ratingsUnClicked(1)"
      '<span class="icon {{iconOff}} ionic_rating_icon_off" ng-style="iconOffColor" ng-click="ratingsClicked(2)"
      '<span class="icon {{iconOn}} ionic_rating_icon_on" ng-style="iconOnColor" ng-click="ratingsUnClicked(2)"
      '<span class="icon {{iconOff}} ionic_rating_icon_off" ng-style="iconOffColor" ng-click="ratingsClicked(3)"
      '<span class="icon {{iconOn}} ionic_rating_icon_on" ng-style="iconOnColor" ng-click="ratingsUnClicked(3)"
      '<span class="icon {{iconOff}} ionic_rating_icon_off" ng-style="iconOffColor" ng-click="ratingsClicked(4)"
      '<span class="icon {{iconOn}} ionic_rating_icon_on" ng-style="iconOnColor" ng-click="ratingsUnClicked(4)"
      '<span class="icon {{iconOff}} ionic_rating_icon_off" ng-style="iconOffColor" ng-click="ratingsClicked(5)"
      '<span class="icon {{iconOn}} ionic_rating_icon_on" ng-style="iconOnColor" ng-click="ratingsUnClicked(5)"
      '</div>',
      scope: {
        ratingsObj: '=ratingsobj'
      },
      link: function(scope, element, attrs) {

        //Setting the default values, if they are not passed
        scope.iconOn = scope.ratingsObj.iconOn || 'ion-ios-star';
        scope.iconOff = scope.ratingsObj.iconOff || 'ion-ios-star-outline';
        scope.iconOnColor = scope.ratingsObj.iconOnColor || 'rgb(200, 200, 100)';
        scope.iconOffColor = scope.ratingsObj.iconOffColor || 'rgb(200, 100, 100)';
        scope.rating = scope.ratingsObj.rating || 1;
        scope.minRating = scope.ratingsObj.minRating || 1;
        scope.readOnly = scope.ratingsObj.readOnly || false;

        //Setting the color for the icon, when it is active
        scope.iconOnColor = {
          color: scope.iconOnColor
        };

        //Setting the color for the icon, when it is not active
        scope.iconOffColor = {
          color: scope.iconOffColor
        };
```

Figure 3.6: Screenshot of IONC Directive.

3.2 APACHE CORDOVA

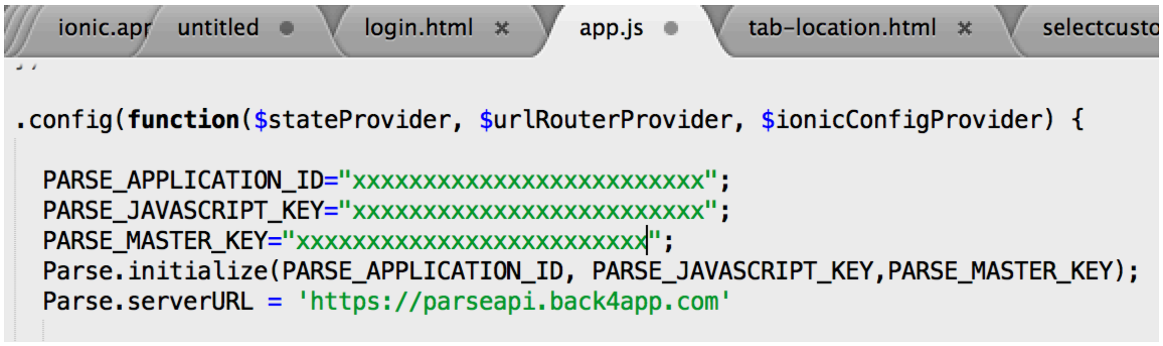
Cordova is the piece of software that stitches the web application and the native application together. Apache Cordova does not just stitch the web app with the native app, but it also provides a set of APIs written in JavaScript to interact with the native features of the device. The APIs JavaScript can talk to the device in its native language enabling application developers to use the phone’s camera, Bluetooth, contacts, storage, etc.

3.3 PARSE SERVER DATABASE

Parse server is a backend cloud service provider which hosts the application's database. It is built on a Mongo DB NoSQL database. Parse server provides SDK for various platforms such as iOS, Android, JavaScript, PHP, .NET etc. Our mobile application interacts with Parse server through JavaScript SDK. Tables in parse server database are referred to as classes and every table row is an object of the table class. The applications controller module interacts with the database by creating parse object which is an instance of a specific subclass with a class name of the table. The retrieved table rows are objects which contain key-value pairs of JSON compatible data.

3.3.1 PARSE SERVER AUTHENTICATION

At the start of the mobile application an active connection is to be established between the application and Parse server database. Parse.initialize() method is called to authenticate the mobile application's access. The method takes in the Parse application id, Parse JavaScript key and master key as input arguments. These authentication keys are auto created by Parse server during the database set up at Parse website. On occasion of application launch, the stored keys are passed to Parse server for authentication.

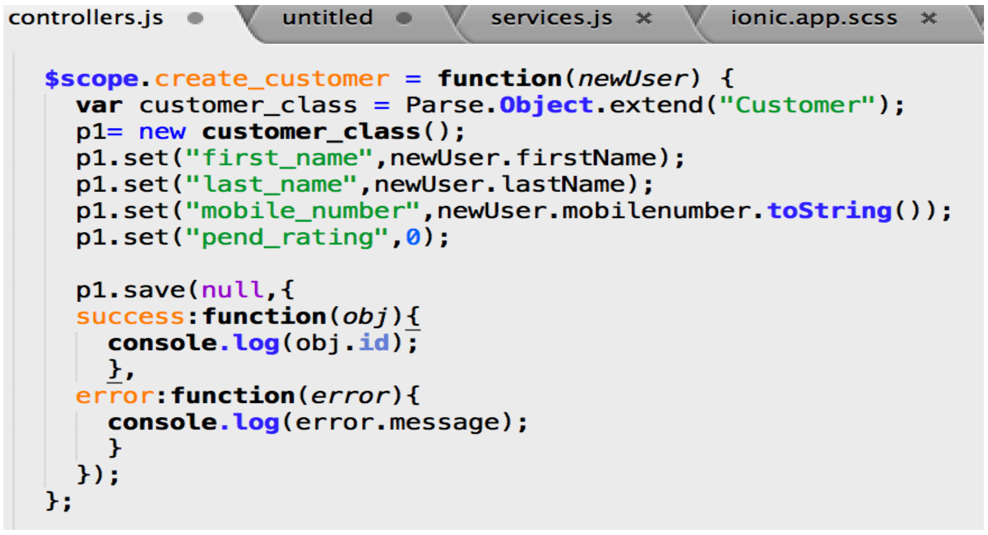


```
.config(function($stateProvider, $urlRouterProvider, $ionicConfigProvider) {

  PARSE_APPLICATION_ID="xxxxxxxxxxxxxxxxxxxxxxxxx";
  PARSE_JAVASCRIPT_KEY="xxxxxxxxxxxxxxxxxxxxxxxxx";
  PARSE_MASTER_KEY="xxxxxxxxxxxxxxxxxxxxxxxxx";
  Parse.initialize(PARSE_APPLICATION_ID, PARSE_JAVASCRIPT_KEY,PARSE_MASTER_KEY);
  Parse.serverURL = 'https://parseapi.back4app.com'
```

Figure 3.7: Screenshot of Parse Server Authentication Code.

3.3.2 ADDING A RECORD TO PARSE SERVER TABLE

Adding a record to a Parse serve table is simple. A new parse subclass is created which extends the properties the table class. Objects are created for the subclass which represent the new record. The row field data is updated using set() method and save() method is invoked to save on the Parse server database. The save() method is embedded with a promise feature which returns either a success or an error callback functionality. On success, the new record is returned as a Parse object which contains an unique id to represent the object among other associated objects of the class. On error, a error object is returned which contains an error code and a error message.

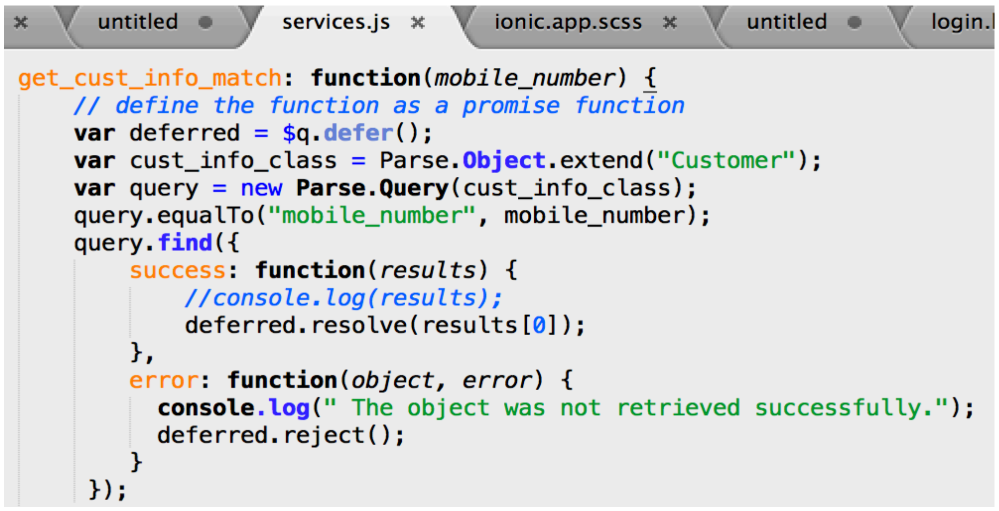


```
$scope.create_customer = function(newUser) {
  var customer_class = Parse.Object.extend("Customer");
  p1= new customer_class();
  p1.set("first_name",newUser.firstName);
  p1.set("last_name",newUser.lastName);
  p1.set("mobile_number",newUser.mobilenumber.toString());
  p1.set("pend_rating",0);

  p1.save(null,{
  success:function(obj){
    console.log(obj.id);
    },
  error:function(error){
    console.log(error.message);
    }
  });
};
```

Figure 3.8: Screenshot of adding a new record to Parse Server Table.

3.3.3 QUERYING A PARSE SERVER TABLE

Querying a Parse server table is similar to adding a record. A subclass of table class is created and a parse query object is created for the subclass. Query constrains such as equal to, less than, not equal etc. are added to the query object. A find() or get() method is used to retrieve the rows from the table.

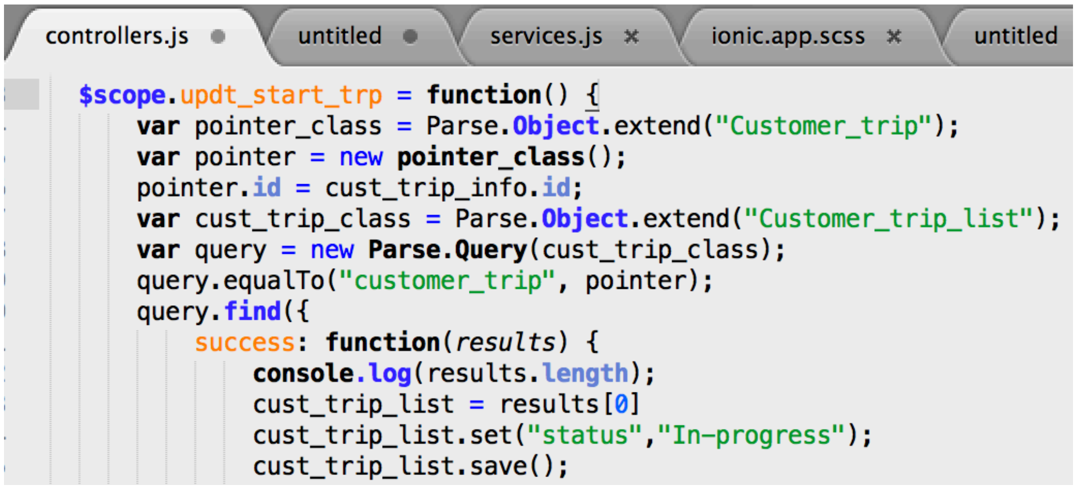


```
get_cust_info_match: function(mobile_number) {
    // define the function as a promise function
    var deferred = $q.defer();
    var cust_info_class = Parse.Object.extend("Customer");
    var query = new Parse.Query(cust_info_class);
    query.equalTo("mobile_number", mobile_number);
    query.find({
        success: function(results) {
            //console.log(results);
            deferred.resolve(results[0]);
        },
        error: function(object, error) {
          console.log(" The object was not retrieved successfully.");
          deferred.reject();
        }
     });
```

Figure 3.9: Screenshot of Querying a Parse Server Table.

3.3.4 UPDATING A PARSE SERVER TABLE

To update a Parse server table, the row to be updated is first queried as described in the previous section. The row fields of the object returned are updated and save() method is invoke to save the updated information on the Parse server.

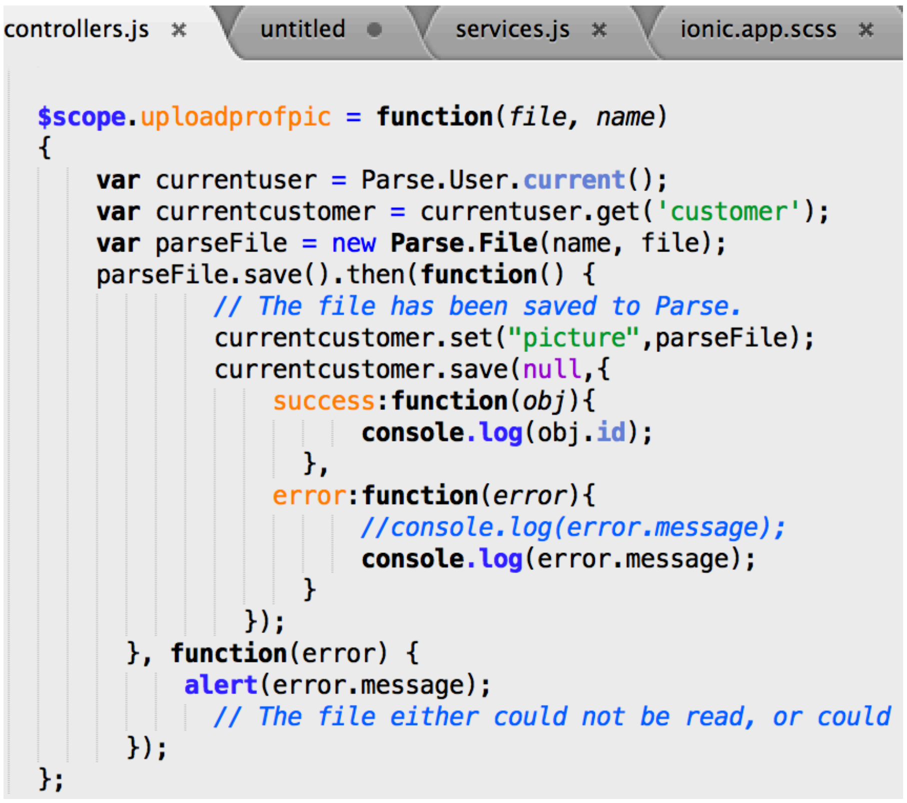


```
$scope.updt_start_trp = function() {
    var pointer_class = Parse.Object.extend("Customer_trip");
    var pointer = new pointer_class();
    pointer.id = cust_trip_info.id;
    var cust_trip_class = Parse.Object.extend("Customer_trip_list");
    var query = new Parse.Query(cust_trip_class);
    query.equalTo("customer_trip", pointer);
    query.find({
        success: function(results) {
            console.log(results.length);
            cust_trip_list = results[0]
            cust_trip_list.set("status","In-progress");
            cust_trip_list.save();
```

Figure 3.10: Screenshot of updating a Parse Server Table.

3.3.5 STORING AN IMAGE ON PARSE SERVER TABLE

Storing an image on to Parse server table is a two step process. First the image to be stored is uploaded as a file to Parse server as a parse file object. The object contains meta data such the file name and file data. The images captured in our application is encoded to base64 string format and attached to the file object and save() method is invoked to upload the image file. After the file is successfully uploaded, as a second step the returned object of file upload process is attached to the row object using set() method and save() is invoked to save the row object on to the Parse server table.



```
$scope.uploadprofpic = function(file, name)
{
    var currentuser = Parse.User.current();
    var currentcustomer = currentuser.get('customer');
    var parseFile = new Parse.File(name, file);
    parseFile.save().then(function() {
            // The file has been saved to Parse.
            currentcustomer.set("picture",parseFile);
            currentcustomer.save(null,{
                success:function(obj){
                        console.log(obj.id);
                    },
                error:function(error){
                        //console.log(error.message);
                        console.log(error.message);
                    }
                });
        }, function(error) {
            alert(error.message);
            // The file either could not be read, or could
        });
};
```

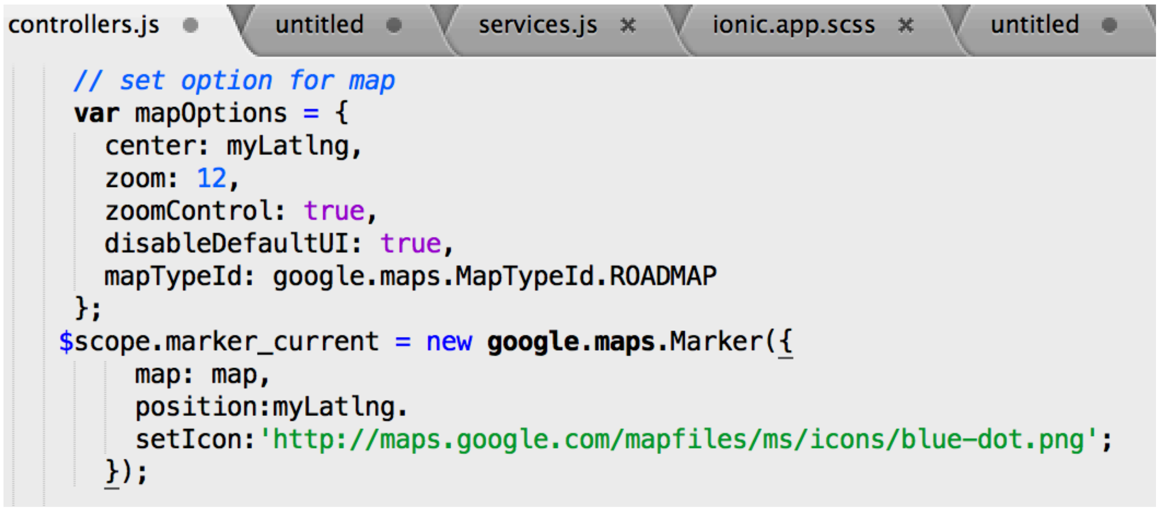
Figure 3.11: Screenshot of Uploading Image to Parse Server Table.

3.4 GOOGLE MAP INTERGRATION

The mobile application uses Google maps for a real time location services. The maps enable users to locate customer/truckers with precise locations on a map. In addition, the map provides route info and distance between source and destination. The application uses Google Maps JavaScript API to avail Google map services. To start of a new project is created in google developer console and Google Maps JavaScript API is activated. Once activated appropriate keys are created and fed into the applications index HTML page. The index page being the landing page of the application, Google Map Services are invoked with the key at the launch of the application.

3.4.1 MARKING CURRENT LOCATION ON MAP

It is vital to display the the current location of the user on a map for better user-friendly approach. To know the current location, the application uses Cordova plugin which in turn communicates with phone's GPS to determine the current location in terms of latitude and longitude. The map defined in the applications view is a web page frame with a dynamic picture and add-ons such as search and zoom that are embedded over the image. The default map properties are controlled creating a new map object and defining the map options. When the phone GPS co-ordinates are retrieved, a marker is dropped on the map specifying the GPS co-ordinates. Options to the change the icon of the marker is available by creating a map marker object and updating its icon option with the URL of the new icon.

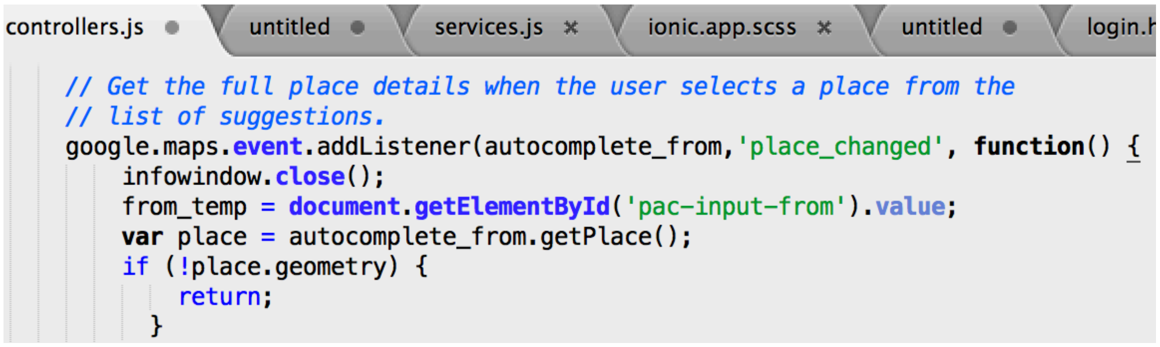


```
// set option for map
var mapOptions = {
  center: myLatlng,
  zoom: 12,
  zoomControl: true,
  disableDefaultUI: true,
  mapTypeId: google.maps.MapTypeId.ROADMAP
};
$scope.marker_current = new google.maps.Marker({
    map: map,
    position:myLatlng.
    setIcon:'http://maps.google.com/mapfiles/ms/icons/blue-dot.png';
  });
```

Figure 3.12: Screenshot of Google Map Options.

3.4.2 AUTOCOMPLETE FOR ADDRESS SEARCH

Autocomplete is an efficient feature for users to search for the accurate source/destination. It also eliminates users from entering incorrect places thereby maintaining the applications data integrity. The auto complete is an bound to the input source/destination search boxes. A listener is added to the search boxes such that each time a character is entered; Google auto complete services is invoked to list all relevant places matching the entered character.

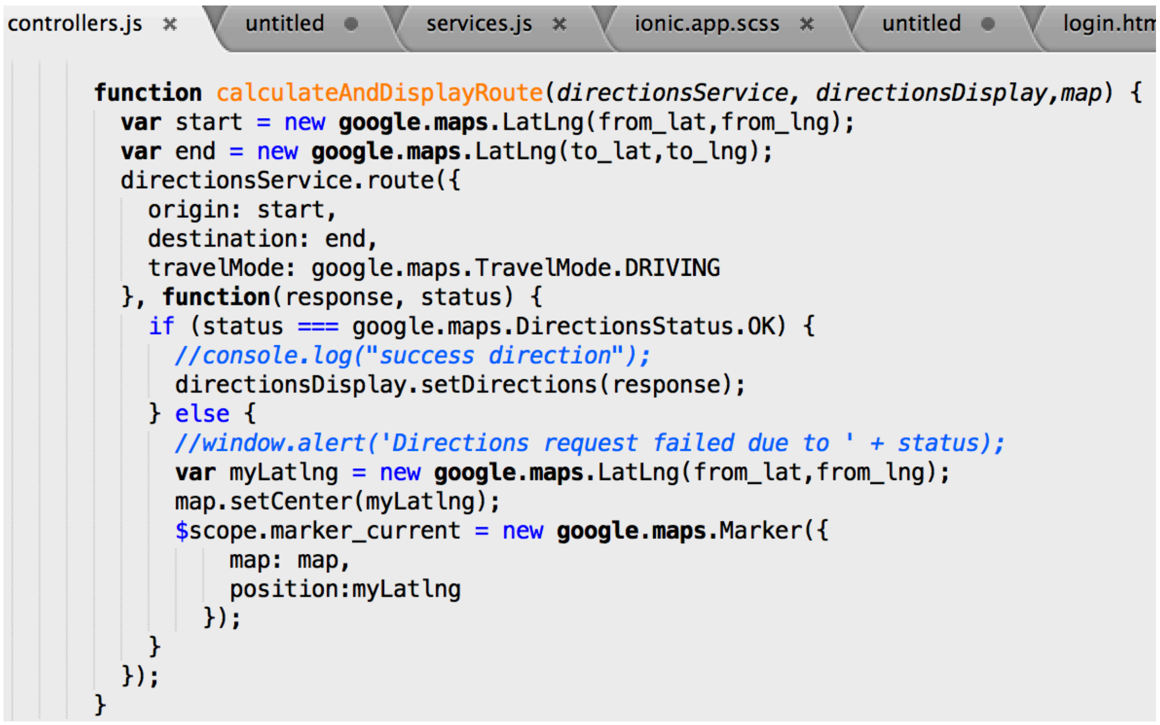


```
// Get the full place details when the user selects a place from the
// list of suggestions.
google.maps.event.addListener(autocomplete_from,'place_changed', function() {
    infowindow.close();
    from_temp = document.getElementById('pac-input-from').value;
    var place = autocomplete_from.getPlace();
    if (!place.geometry) {
        return;
    }
```

Figure 3.13: Screenshot of Google Autocomplete Address Search.

3.4.3 DIRECTION SERVICE

The direction service helps the application’s users to have a fair idea of how far is their destination from the source. The application uses Google’s direction service to determine the route. The latitude and longitude of the source and destination are fed to the direction service object. On a successful call back, the route is plotted on the map with markers highlighting the source and destination. In case no route is available, a marker is dropped only on the source.



```
function calculateAndDisplayRoute(directionsService, directionsDisplay,map) {
  var start = new google.maps.LatLng(from_lat,from_lng);
  var end = new google.maps.LatLng(to_lat,to_lng);
  directionsService.route({
    origin: start,
    destination: end,
    travelMode: google.maps.TravelMode.DRIVING
  }, function(response, status) {
    if (status === google.maps.DirectionsStatus.OK) {
      //console.log("success direction");
      directionsDisplay.setDirections(response);
    } else {
      //window.alert('Directions request failed due to ' + status);
      var myLatlng = new google.maps.LatLng(from_lat,from_lng);
      map.setCenter(myLatlng);
      $scope.marker_current = new google.maps.Marker({
        map: map,
        position:myLatlng
      });
    }
  });
}
```

Figure 3.14: Screenshot of Google Direction Services.

3.5 PUSH NOTIFICATION

Push notification is an essential feature for all modern mobile applications. Notifications keep users engaged and encouraged in using the mobile application. Its a direct channel of communication to the user for sending announcements, reminders and updates. In our mobile application, push notification plays an important role in updating customers and truckers. In addition, truckers receive notification alerts when their profile or truck is approved. The mobile application uses Ionic push service to send notification to mobile users. The deployed mobile applications are registered with native push notification services such as GCM – Google Cloud Messaging for Android platform and APN – Apple Push Notification Service for iOS platform. These native push notification services identify each device using a unique device id which is a key to communicate with the mobile device.

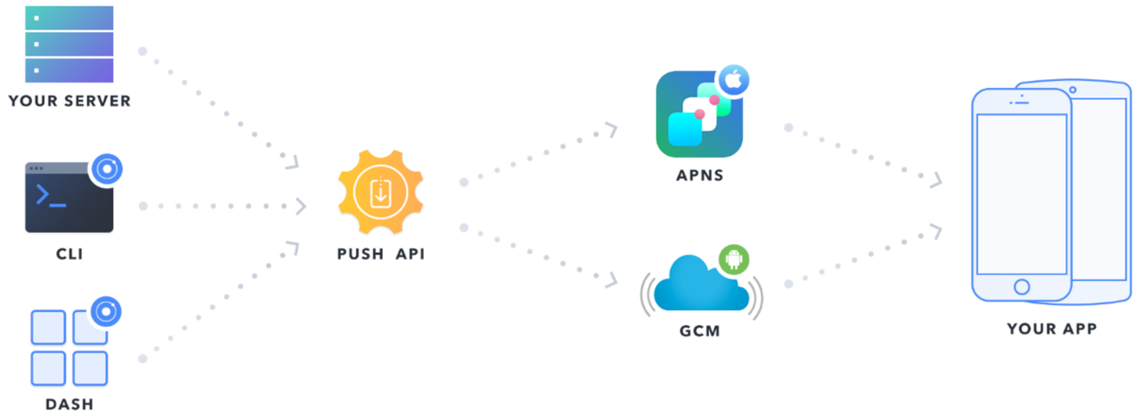
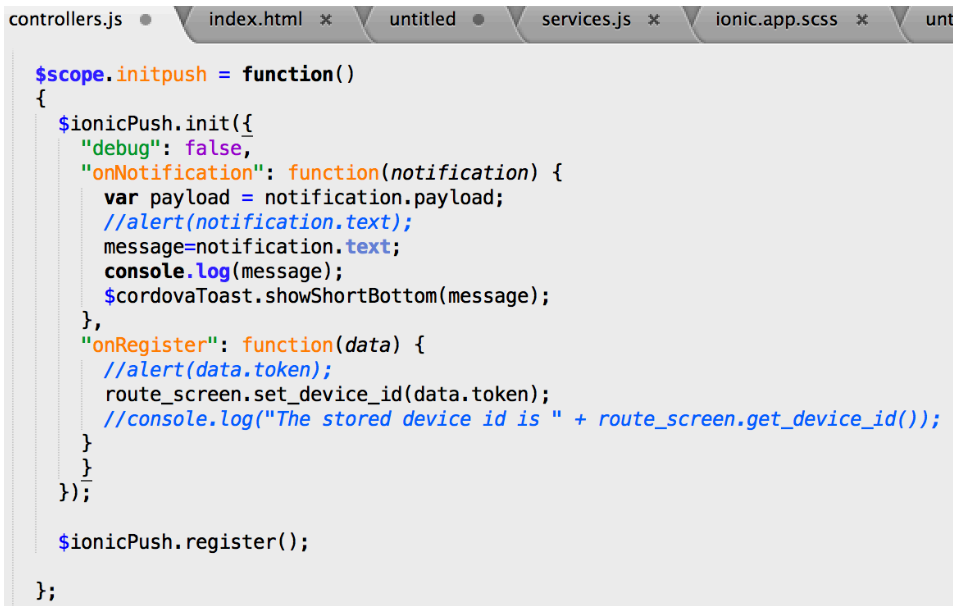


Figure 3.15: IONIC Push Notification Architecture.

In turn Ionic push notification server is registered with native push notification services by means of a project id created in native push notification services. The Ionic push notification server acts as a middleman between native push notification services and our mobile application.

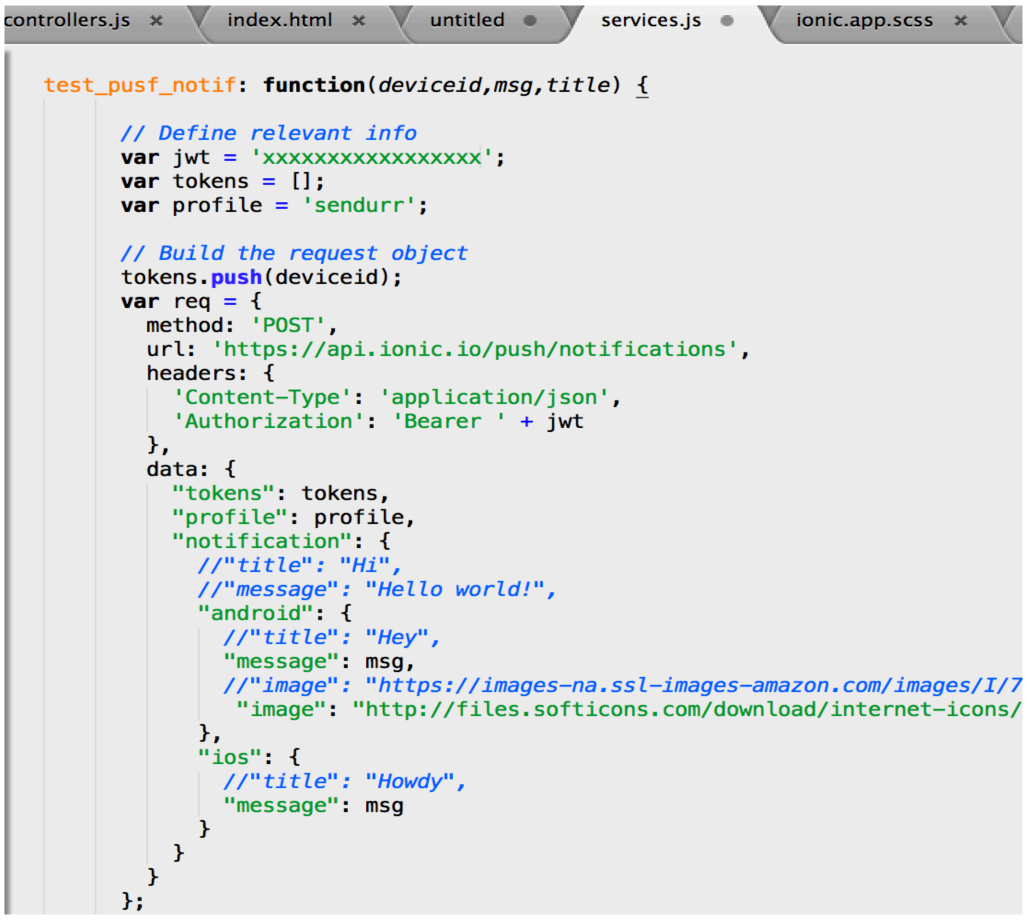
Each time a user logs on to the mobile application, the application registers itself with native push notification services. A successful registration is returned a device id which is stored along with the user’s login information in the Parse server database. Now every registered user can be notified using the stored device id. The ini() method is a promise function which accepts to return functions. First the register function returns the registered device id. Second a listener notification function awaiting notifications.



```
$scope.initpush = function()
{
  $ionicPush.init({
    "debug": false,
    "onNotification": function(notification) {
      var payload = notification.payload;
      //alert(notification.text);
      message=notification.text;
      console.log(message);
      $cordovaToast.showShortBottom(message);
    },
    "onRegister": function(data) {
      //alert(data.token);
      route_screen.set_device_id(data.token);
      //console.log("The stored device id is " + route_screen.get_device_id());
    }
    }
  });

  $ionicPush.register();

};
```

Figure 3.16: Screenshot of Native Push Notification Registration.

Sending a push notification is handled by making a HTTPS post request to the Ionic push notification server. The request contains an authentication key which verifies the authenticity of the request. The request data consists of device id, notification message, title, sub title and image for the notification icons. Platform specific options help customize the notification look and feel. The application uses Cordova push notification plugin to present the notification content to different mobile application platforms. $http ANGULARJS service method is used to trigger a push notification request with Ionic push notification server. The $http is a promise function which returns a success or an error. Once push notification is sent from Ionic push notification server, the listener function of init() method returns a notification alert. The notification message is then converted to native mobile application format using Cordova push notification plugin and notified to the mobile user.



```
test_pusf_notif: function(deviceid,msg,title) {

      // Define relevant info
      var jwt = 'xxxxxxxxxxxxxxxxx';
      var tokens = [];
      var profile = 'sendurr';

      // Build the request object
      tokens.push(deviceid);
      var req = {
        method: 'POST',
        url: 'https://api.ionic.io/push/notifications',
        headers: {
          'Content-Type': 'application/json',
          'Authorization': 'Bearer ' + jwt
        },
        data: {
          "tokens": tokens,
          "profile": profile,
          "notification": {
            //"title": "Hi",
            //"message": "Hello world!",
            "android": {
              //"title": "Hey",
              "message": msg,
              //"image": "https://images-na.ssl-images-amazon.com/images/I/7
               "image": "http://files.softicons.com/download/internet-icons/
            },
            "ios": {
              //"title": "Howdy",
              "message": msg
            }
          }
        }
      };
```

Figure 3.17: Screenshot of Sending Push Notification.

CHAPTER 4

COMPARATIVE STUDY OF OTHER LOGISTIC APPLICATIONS

In this chapter we do a comparative study of other logistic mobile applications available in the market. As discussed earlier the main focus of every logistic application is to provide a customer friendly service with the use of technology. Each application has its own unique feature targeting to give a structure to the unprofessional and unregulated Indian trucking industry. In the coming sections we shall discuss on top emerging logistic mobile applications.

4.1 PORTER - HIRE TRUCKS IN INDIA

Porter is a revolutionary approach that makes intra-city pickups and deliveries thoroughly efficient and reliable. They have a varied fleet of vehicles servicing customers across Mumbai, Delhi, Bengaluru, Hyderabad, and Chennai in India. The company concentrates on fleet management and truck utilization with majority of its customers ranging from small and medium enterprises and a few large institutions.

The application features real time truck tracking systems with periodic SMS alerts. It offers transparent pricing with rate cards available making hazel free seamless consumer experience. The application's fleet management tool includes live tracking with replay of historic tracks, Periodic SMS/email alerts, Geo-fencing alerts, Scheduled

Reports and Client specific customization. Payments are handled either through cash on delivery/pickup or Porter cash wallet which includes discount coupons.

Overall the application targets serving the needs of small to large enterprises but lacks addressing a common individual customer. Even though the application features on the booking for individual customers, the slight higher shipping prices evades individual customers from utilizing the application for a regular shipping. The rate card is definitely a better option to the industries unorganized pricing practices. However, customers do not have real time price estimators based on distance which might lead to unexpected final delivery costs. Customers intending to ship small goods which a lesser load space would ultimately have to pay a price for the entire load area.

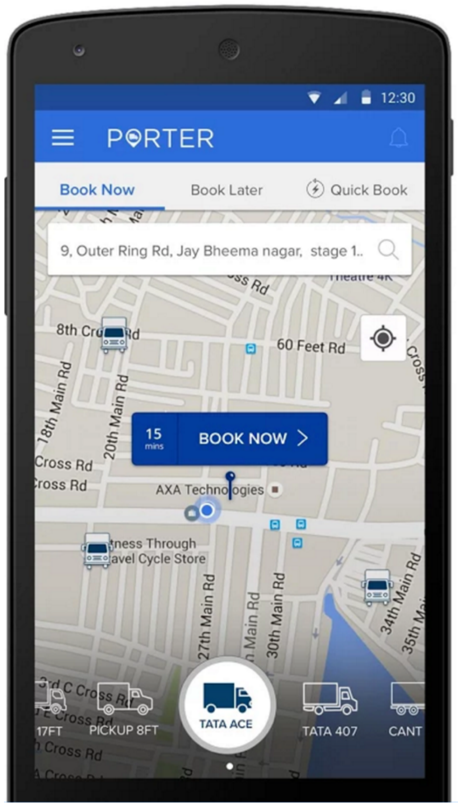


Figure 4.1: Porter Booking Screen.

On the trucker's side, truckers have no option to operate on themselves and have to dependent totally on the trips assigned by Porter. A share of the trip cost goes as commission to Porter which makes them no different to current truck operators. With a better fleet management and transparent pricing, Poster has attempted to organize the structure of the trucking industry but still lacks a fair customer and truck driver involvement.

4.2 BLOWHORN

Blowhorn offers a tech platform connecting customers who want to move goods quickly and effortlessly to owners of mini-trucks. Blowhorn is organizing a massive disorganized market: mini-truck drivers. Their technology is easily used by both consumers and by truck owner/drivers, making it simpler for consumers to move goods or household possessions around a city. It provides an intra-city logistic platform operating in Chennai, Bangalore and Hyderabad in India.

The striking feature of Blow horn is that is works with mini-truck owners. The market for mini-truck transportation in India is estimated to be between thirty and sixty thousand crore rupees ($4.8-9.6 billion USD) annually. There is a mismatch in the market where Blowhorn wants to come in. Mini-truck drivers suffer from inefficient utilization, sometimes not getting a booking for a load to carry for a day or more. At the same time, consumers have ever-increasing needs to move things around the crowded streets of India's major metros.

The application offers real time tracking and reliable service. The company has targeted individual customers who are rightly looking for a shipping platform to ship basic day to day goods such as washing machines, furniture etc. The online payment system is a great plus in the application with options to apply discount coupons. The application really inches closer in addressing common individuals, yet misses addressing concerns of truckers. Here too, truckers have fewer choice to operate independently and have no choice in selection of goods to be shipped. The application does not serve as a platform for truckers and customers to interact and meet their needs.

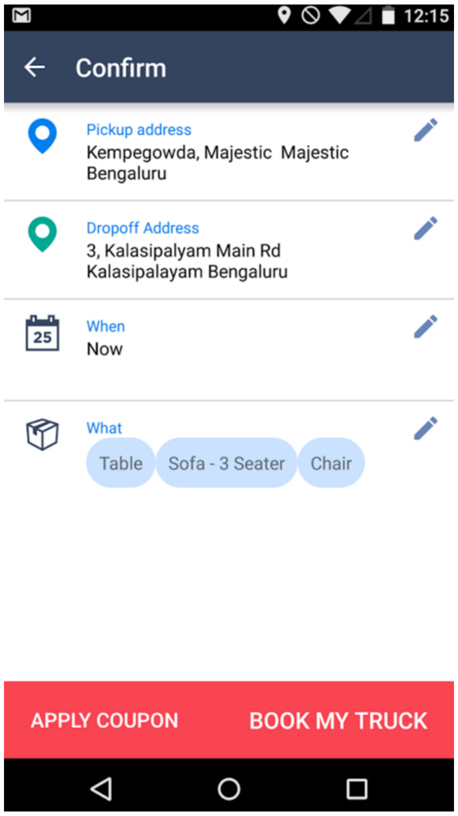


Figure 4.2: Blowhorn Booking Confirmation Page.

4.3 RETURN TRUCKS

Return Trucks is a truck and load matching platform. Customers can post the truck availability and load availability for free and load owners can book their truck. There are more options like exchanging information, tracking trucks, etc. The application tackles the industries inefficient practice on truck booking which is heavily dependent on intermediates in the form of Agents and Brokers. The main focus point is on returning trucks whose utilization is low when returning back to home location.

The application has tapped opportunities in the Indian trucking industry by providing affordable technological solutions to small and medium players that actually help them get loads and empower them with technology in building their reputation. Helping return truck drivers with relatively better truck utilization has reduced the lead time for the return journey while the load owners are assured of timely delivery at a comparatively lesser cost as well as reduce carbon footprint.

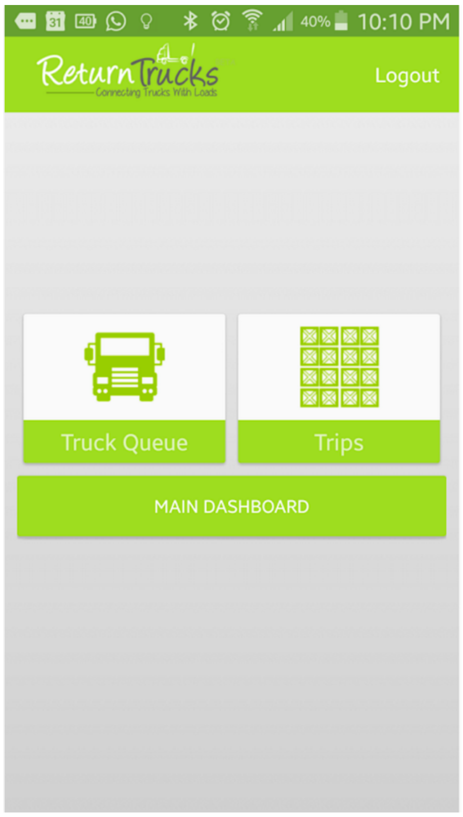


Figure 4.3: Return Truck - Homepage.

Return Trucks has certainly inched closer to connect truckers with load owners. Still it seems pricing isn't available before hand for the load owner to estimate the cost for shipment. The application targets mainly corporate customers who demand high quality of service which the application would meet to stay in business. The down side would be their pricing would be high which would not help individual customers. Overall the application is on same pages with our idea what has definitely worked its way to provide a platform for trucker and load owners to connect with each other.

CHAPTER 5

FUTURE WORK AND CONCLUSION

5.1 FUTURE WORKS

This section highlights on future work planned with regard to the proposed thesis idea.

I. Self-sustained revenue model:

   In the long run, with more number of truckers and customers utilizing the mobile application, will require to larger databases, efficient algorithms to satisfy the users' needs. All necessary infrastructure demands investment cost. The most innovative approach would be customizing the mobile application to produce a revenue generation model. In-app advertisements, truck classifieds can be thought of to generate revenue for the application.

II. Real Time Truck Tracking:

   The current application does not house a real time truck tracking feature due to inadequate mobile internet connectivity across India. The application depends on mobile internet to determine the user's location. Installing native GPS on trucks can help track shipment precisely which might also incur cost.

III. Enhanced User Interface:

   Personalized enhancements like storing user's previous source and destination searches, adding favorite routes, changing form based shipment creation to graphical based shipment creation, would enhance user experience. This will also increase the application’s performance.

IV. Data Analytics:

The application’s data can provide a perfect platform to study shipment habits of customers, understand and respond to geographical demands. The outcome of data analytics can be formulated to recommendations to truckers that helps them execute better business models.

V. IOS platform:

Keeping in mind the huge number of Android users in India compared to IOS, the mobile application is built for Android platform and launched in Google Play Store. As part of future needs, an IOS version will have to be developed.

5.2 CONCLUSION

The proposed mobile application offers a transparent logistic forum for customers and truckers to ship goods. The application also eliminates middlemen resulting in reduction in shipment costs. The current unstructured Indian trucking industry renders poor customer service, but the mobile application proves, with incorporation of technology the Indian trucking industry could evolve into a new era focusing on customer service. Yes, there are still challenges such as uneducated drivers, less infrastructure support for technology, universal shipment pricing, government polices. However, the proposed mobile application objective for customer satisfaction can dominate the challenges and bring in a need for a revolutionary change to Indian trucking industry.

REFERENCES

[1] G. Raghuram “An Overview of the Trucking Sector in India: Significance and Structure” *No WP2015-12-02, IIMA Working Papers from Indian Institute of Management Ahmedabad, Research and Publication Department, 2015*

[2] Siddharth Jain “Changing trends in the Indian Trucking Industry- Promising Outlook” https://www.linkedin.com/pulse/changing-trends-indian-trucking-industry-promising-outlook-jain

[3] World Bank. 2005. “India: Road Transport Service Efficiency Study”. https://openknowledge.worldbank.org/handle/10986/8356

[4] Andersson, Christian, and Nikhil Puranik. "Understanding the Indian on-road transport customer." (2012). http://publications.lib.chalmers.se/records/fulltext/159359.pdf

[5] "Porter - Hire trucks in India" - https://play.google.com/store/apps/details?id=com.theporter.android.customerapp&hl=en

[6] "Cargoji (β) Book a mini truck" - https://play.google.com/store/apps/details?id=com.cargoji.cargojiuserapp&hl=en

[7] "LOTrucks" - https://play.google.com/store/apps/details?id=com.lotrucks.lotrucks&hl=en

[8] "IONIC framework" – http://ionicframework.com/getting-started/

[9] "Parse Database" – https://parse.com/docs/js/guide

APPENDIX A: GOOGLE PLAY STORE

Link to TrcukDeal mobile application launched on Google Play Store for Android mobile users. https://play.google.com/store/apps/details?id=com.truckdeal.sendurr2208