iAgent Trip Planner

A PROJECT REPORT

Submitted in partial fulfilment for the award of the degree of

MS

in

Software Engineering

By

Ambati Sai Praneeth, 10MSE0066

Under the Guidance of

Prof. M Nirmala

Assistant Professor (Senior)

SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING

VIT UNIVERSITY

DECLARATION BY THE CANDIDATE

I hereby declare that the project report entitled "**iAgent Trip Planner"** submitted by me to Vellore Institute of Technology University; Vellore in partial fulfilment of the requirement for the award of the degree of **MS** in **SOFTWARE ENGINEERING** is a record of bonafide project work carried out by me under the guidance of **Prof.NIRMALA M.** I further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Vellore Signature of the Candidates

Date: **Ambati Sai Praneeth**

VIT
UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)
Vellore - 632 014, Tamil Nadu, India

SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING [SITE]

SOFTWARE ENGINEERING DIVISION

BONAFIDE CERTIFICATE

This is to certify that the project report entitled "**iAgent Trip Planner"** submitted by **A.SAI PRANEETH (10MSE0066)** to Vellore Institute of Technology University, Vellore, in partial fulfilment of the requirement for the award of the degree of **MS** in **SOFTWARE ENGINEERING** is a record of bonafide work carried out by him/her under my guidance. The project fulfils the requirements as per the regulations of this Institute and in my opinion meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma and the same is certified.

Prof.Nirmala M
INTERNAL PROJECT SUPERVISOR
Assistant Professor (Senior)
SITE

Internal Examiner External Examiner

Acknowledgements

It is with great sense of satisfaction that we present our project work and wish to express my heartfelt thanks to all those who helped us out in completing the project. I would like to thank the chancellor of Vellore Institute of Technology, **Sri G. Viswanathan**, our program manager **Prof. Srinivasa Perumal R** and the project coordinator of M.S Software Engineering, **Prof. Krishna Chandramouli** for providing constant support throughout the project.

It is my privilege and pleasure to express my profound sense of respect, gratitude and indebtedness to our guide **Prof.Nirmala M**, Assistant Professor [Senior], School of information technology, for her indefatigable inspiration, guidance, cogent discussion, constructive criticisms and encouragement throughout the dissertation work.

Place: Vellore

Date: **Ambati Sai Praneeth**

Executive Summary

Under the current economic environment, customers become more prudent while more eager to invest in insurance. It always is, but becomes more critical to build as well as to maintain a strategic customer relationship with an innovative engagement model. Insurance agents will have many clients whom he/she needs to meet on a particular day for various business activities. She/he makes multiple appointments in various locations and manually records them.

Facing this new challenge, our client decided to provide a unified, supervised access of their resources and to evolve their application products.

This project involves two parts:

- a RESTful style web service and
- *a cross-plat-form mobile application.*

The “iAgent Trip Planner” Application provides the agents to manage their appointments by allowing them to Add/Modify/Delete/View. For a selected appointment, it displays the Google maps with the appointment location highlighted. It also locates the nearby clients on the Map, if there is a change in schedule. It performs many other activities like Displaying Routes from user current location to last appointment, Show nearby places (ATM's, Hospitals, Restaurants, Parking Garages etc) at the selected appointment locations, Display clients in the agent selected area, search customers in a particular ZIP code, Traffic & Weather Information.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

1.1 PROBLEM DEFINITION

In Insurance Business, Agent and Customer's relationships play a crucial role in developing the business. Insurance agents will have many clients whom he/she needs to meet on a particular day for various business activities. She/he makes multiple appointments in various locations and manually records them.

Facing this new challenge, our client decided to provide a unified, supervised access of their resources and to evolve their application products.

The "iAgent Trip Planner" Application provides the agents to manage their appointments by allowing them to Add, Modify, Delete, and View. For a selected appointment, it displays the Google maps with the appointment location highlighted. It also locates the nearby clients on the Map, if there is a change in schedule. It performs many other activities like Displaying Routes from user current location to last appointment, Show nearby places (ATM's, Hospitals, Restaurants, Parking Garages etc) at the selected appointment locations, Display clients in the agent selected area, search customers in a particular ZIP code, Traffic & Weather Information.

1.2 EXISTING SYSTEM

In the existing system, the insurance agents will manually record their appointment in notes or in remainder applications and it's hard to store all the details of appointments like address, phone number, timing, date etc.

1.3 DRAWBACKS OF EXISTING SYSTEM

- The agent might forget about the appointment.
- Agent might not know the address of client.
- Might not know the route to the address.
- Timings of two appointments might overlap.
    - Agent cannot find out nearby places like ATM, Hospitals, and Restaurants etc in a new area when required.
- Agent cannot know the nearby clients within his location.

1.4PROPOSED SYSTEM

Considering the existing system and the drawbacks of the existing system we have proposed a Cross platform application that overcomes the drawbacks.

- Cross Platform Application(Android/ iOS)
- Integrates Google Calendar with Google Maps
- Agents can Add/Edit/Delete/View Appointments at one place.
- Displays All Appointments on Google Maps.
- Provide Route from Agent Current Location to Last Appointment.
- Provide Traffic and Weather Information.
- Provide Customers in Agent selected Area.
- Agent can know the nearby places like ATM, Restaurants, Hospitals, and Parking Garages in appointments locations.
- Search clients in a specified ZIP Code.

1.5 ADVANTAGES OVER EXISTING SYSTEM

- Time Saving for agents.
- All in one application to maintain customer details and appointments.
- Easy to use.
- Calculates the total distance agents needs to cover in a day and also provide route.
- Differentiate Markers to easily identify today's appointments and future appointments.

1.6MOTIVATION AND OBJECTIVE

- To develop a cross Platform Mobile Application (iAgent Trip Planner)

- To Integrate RESTful style web service with the application that is being developed.

- To help the Insurance agents in the insurance company to build strategic customer relationships with innovative engagement models.

- To integrate Google Calendar API, Google Maps API within this application and help the insurance agents manage their Appointments.
- The main objective of this project is to communicate with RESTful style web services to allow the insurance agents access their respective client information, manage (Add, Delete, Modify, View) their appointments and locate the respective client information on Google Maps.

1.7 ORGANAISATION OF THE REPORT

This section gives a brief representation of each chapter in this report
Chapter1-INTRODUCTION

This chapter gives a brief introduction of this project and background of android, web services and it contains problem definition, aim & objectives of project, existing system, proposed system, advantages of proposed system, and disadvantages of existing system.

Chapter 2–OVERVIEW OF THE PROPOSED SYSTEM /LITERATUREREVIEW

This chapter gives a brief description of the whole proposed software system developed, system preliminary design, system planning and details of hardware and software requirements. A thorough review of literature with respect to the chosen field is projected.

Chapter 3- PROPOSED FRAMEWORK/ SYSTEM ARCHITECTURE

This chapter gives the base architecture of the system and it gives the architecture of different modules in the system.

Chapter 4- IMPLEMENTATION OF SYSTEM/METHODLOGY

This chapter should reflect development of the project such as: implementation, experimentation, optimization, evaluation etc and unit integration testing, Sample code, Screen shots of each modules etc.

Chapter 5- TESTING

This chapter consists of the brief explanation about testing & different types of testing. It also consists of the testing report of the application developed

Chapter 6- CONCLUSION AND FUTURE ENHANCEMENTS

This chapter summarizes the key aspects of the project (failures as well as successes) and states the conclusions. It also outlines about future work.

Chapter 7- REFRENCES

2. Overview/Literature Review of Proposed System

2.1 OVERALL DESCRIPTION

A Cross Plat form application, iAgent Trip Planner has been developed for Insurance Agent to simplify their Appointments, Client Management. This application helps the agents to Add/Edit/Delete/View the appointment with their clients in their daily basis. The appointment will be synced with the agents Google Calendar and all the appointments are displayed on Google Maps. Agent can perform different functions like find Route from his/her current location to last appointment, Search Nearby places, Traffic & Weather Information, Find Customers in a selected area, Search Customers in a specific ZIP Code, Update current Location, Search Customers at Map Centre, View Google Calendar etc.

2.2 SOFTWARE AND HARDWARE REQUIREMENTS

2.2.1 SOFTWARE REUIREMENTS

- Java JDK 7.0 or above
- Sencha touch
- Sencha CMD
- Node.js
- Phonegap /Cordova
- Android SDK
- Android Plug-in for Eclipse

2.2.2 HARDWARE REQUIREMENTS(For Development)

- Windows OS / iOS
- 4GB RAM
- 40GB Memory

2.2.3 HARDWARE REQUIREMENTS(For Application)

- Android Tablet / iPad
- 1GB RAM
- USB Debugging
- 10MB space (Minimum)

2.3 **MODULES**

1. Authorization
2. Appointment List
    - Display Appointments
    - Create New Appointment
    - Edit or Delete an existing appointment
    - Search Existing Customer
3. Map View
    - Mark Scheduled Appointments
    - View Appointment Details
    - Update Current Location
    - Show Route
    - Show Near By
    - Show All Markers
    - Reset Zoom
    - Search Customer by ZIP code
    - Show Customers Near Map Center
    - Display Customers in a selected area
4. Calendar View

2.4 **SYSTEM PLANNING**

iAgent Trip Planner

	Task Name	Start Date	End Date	Duration
1	**Authorization**	**02/16/15**	**02/23/15**	**6**
2	Authorization Module	02/16/15	02/23/15	6
3	**Calendar View**	**02/24/15**	**02/27/15**	**4**
4	Calendar View	02/24/15	02/27/15	4
5	**Appointment List**	**03/02/15**	**03/27/15**	**20**
6	Display Appointments	03/02/15	03/06/15	5
7	Create New Appointment	03/09/15	03/13/15	5
8	Edit/ Delete existing appointment	03/16/15	03/20/15	5
9	Search Existing Customer	03/23/15	03/27/15	5
10	**Map View**	**03/30/15**	**05/05/15**	**27**
11	Mark Scheduled Appointments	03/30/15	03/31/15	2
12	View Appointment details	04/01/15	04/03/15	3
13	Update Current Location	04/06/15	04/07/15	2
14	Show All markers	04/08/15	04/13/15	4
15	Show Route	04/14/15	04/17/15	4
16	Show Near By	04/20/15	04/23/15	4
17	Reset Zoom	04/24/15	04/24/15	1
18	Search Customer by ZIP code	04/24/15	04/27/15	2
19	Show Customers Near Map Center	04/28/15	04/30/15	3
20	Display Customers in a selected area	04/30/15	05/05/15	4

Fig 2.4.1 Project Plan

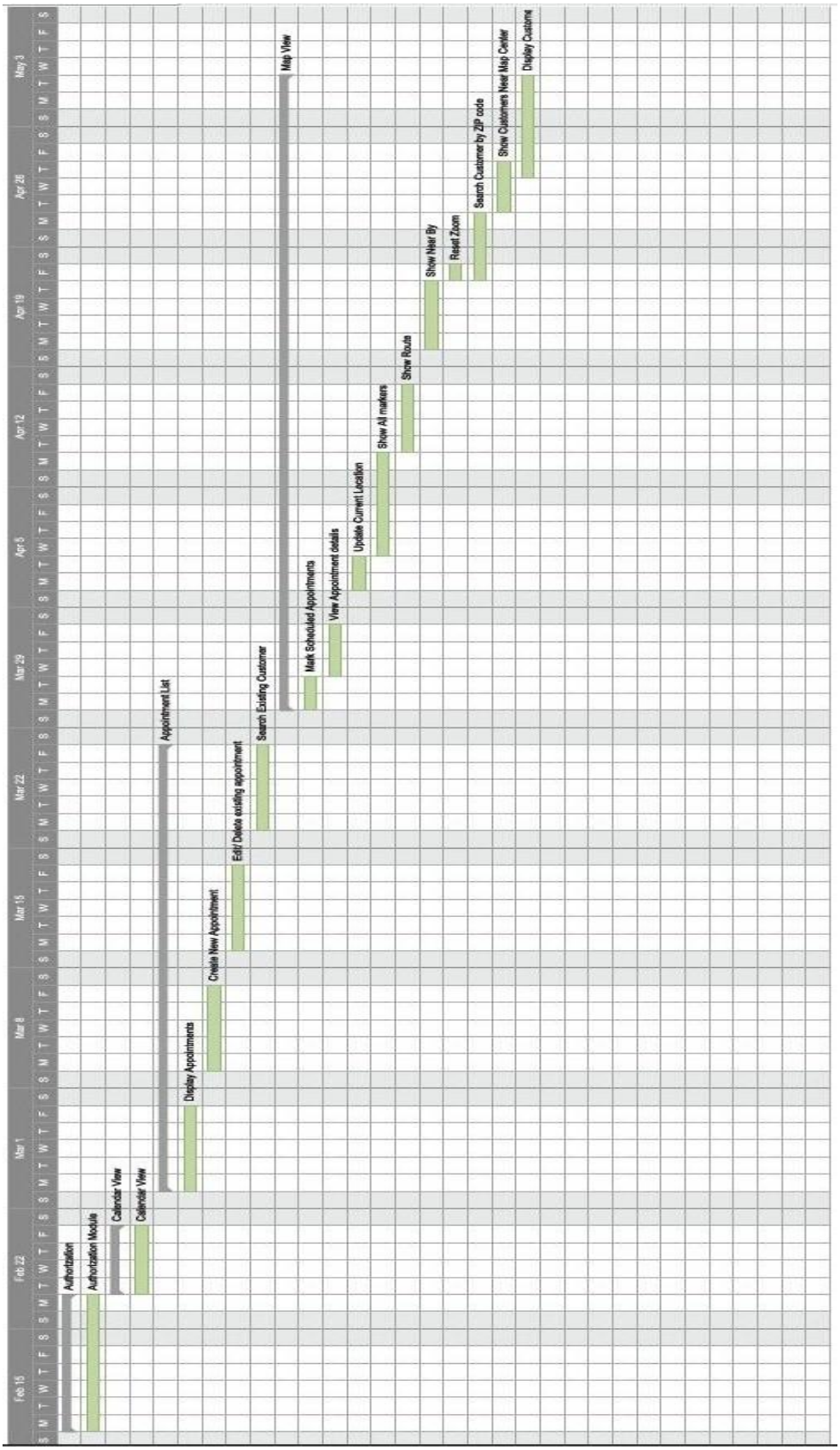

Fig 2.4.2 Gantt chart for Project Plan

2.5 LITERATURE REVIEW

2.5.1 **Title:** Implementation of Location based Services in Android using GPS and Web Services.[1]

In this paper, it's about the implementation of Location based services through Google Web Services and Walk Score Transit APIs on Android Phones to give multiple services to the user based on their Location. They have implemented LBS in two ways:

-To process location data in a server and to forward the generated response to the clients.

-To find location data for a mobile device-based application that can use it directly.

-Identified various constraints to implement Location Based Services which includes Technology Constraints, Infrastructure Constraints, and Market Failure.

In our project, the application being developed is supported with different platforms like Android, iOS, Windows Phone, Blackberry and it uses RESTful Web services which improves the performance of the application.

2.5.2 Title: Introduction to iPad Application Development with PhoneGap.[2]

In this paper, we propose a method to build applications for iPad using the PhoneGap framework, which uses building apps in HTML and javaScript. It is also good for web developer dives in mobile development and a simple way of making applications for cross-platform within a short time.

In our application that is being developed we also use the PhoneGap/Apache Cordova Framework for targeting different platforms.

2.5.3 **Title:** Performance Evaluation of RESTful Web Services for Mobile Devices.[3]

In this paper, it evaluates the RESTful web service for mobile devices against conventional SOAP web services. The experimental results show that RESTful web services outperform conventional SOAP web services. A recommendation to use RESTful web services on mobile devices has been concluded from experimental result.

In our application we will use RESTful style web services since the performance of RESTful style web services is much efficient than conventional SOAP services.A simple experiment found that the message size of RESTful web service is smaller than messages of Conventional SOAP web service.

3. PROPOSED FRAMEWORK/ SYSTEM ARCHITECTURE

3.1 ARCHITECTURE

iAgent Trip Planner has been implemented using MVC Framework in Sencha Touch. It integrates the Google Calendar API in the controller level and controller is the main level for the business logic of the application.

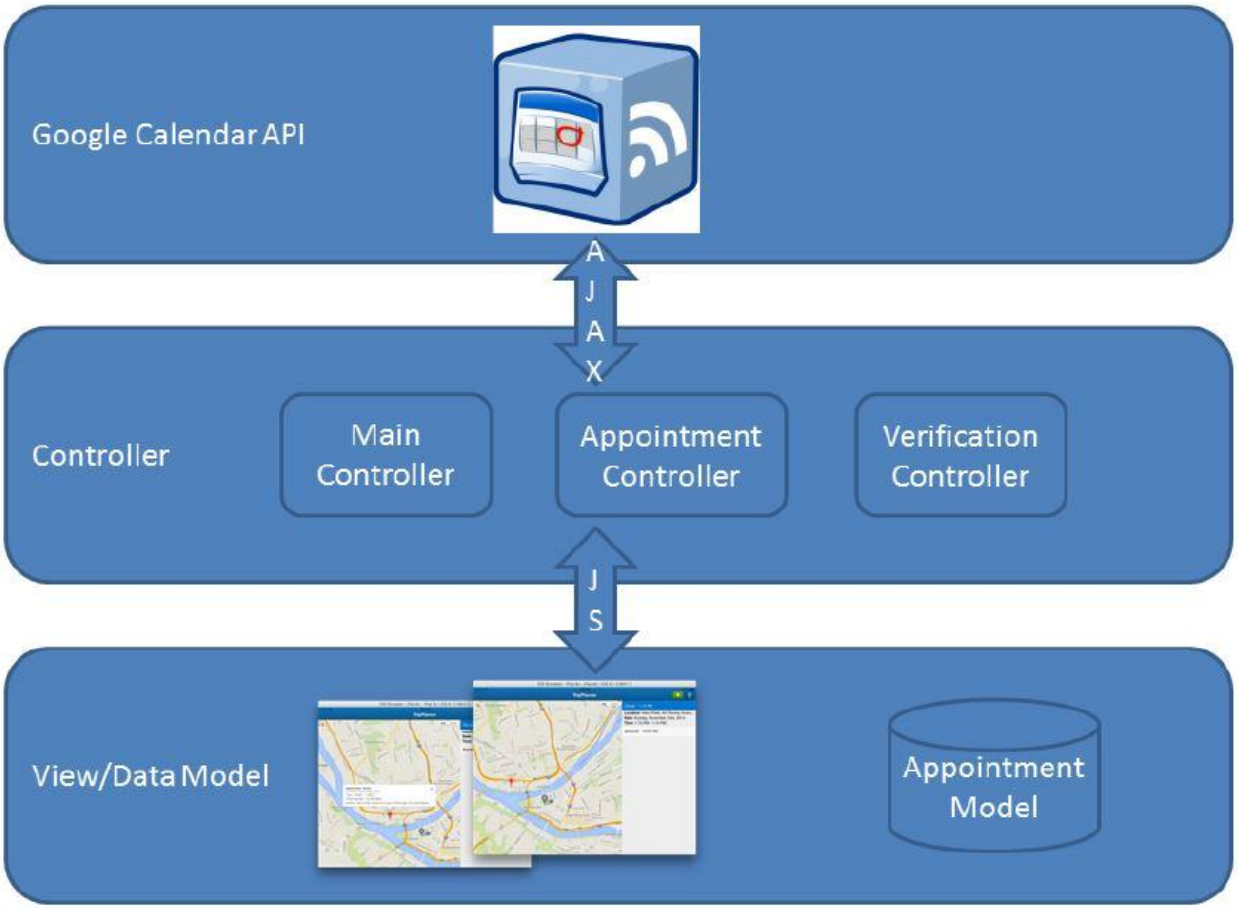

Fig 3.1.1 System Architecture

3.1.1 CONTROLLERS:

Controller is the classes who are responsible for the main interactive between the data model and the view presenting to users. For iAgent Trip Planner, there are three controllers:

- Main Controller
- Appointment Controller
- Verification Controller

3.1.1.1. Main Controller:

Main Controller is responsible for handling the user action in MainController.js and passing these actions to corresponding handler class.

Table 3.1.1.1. Main Controller functionalities

User Action	Passing Function	Handler Class
Authorization	handleAuthClick()	/controller/verification
New Appointment	newAppt()	/controller/Appointment
Show Calendar	showCalendar()	/controller/Appointment
Edit Appointment	editAppt()	/controller/Appointment
Expand Appointment List	expandAppt()	/controller/Main
Collapse Appointment List	collapseAppt()	/controller/Main
Show Route	showRoute()	/controller/Main
Show NearBy	showNearBy()	/contoller/Main
Draw Rectangle	DrawRect()	/controller/Main
Display Traffic	TrafficLayer()	/view/Map
Display Weather	WeatherLayer()	/view/Map

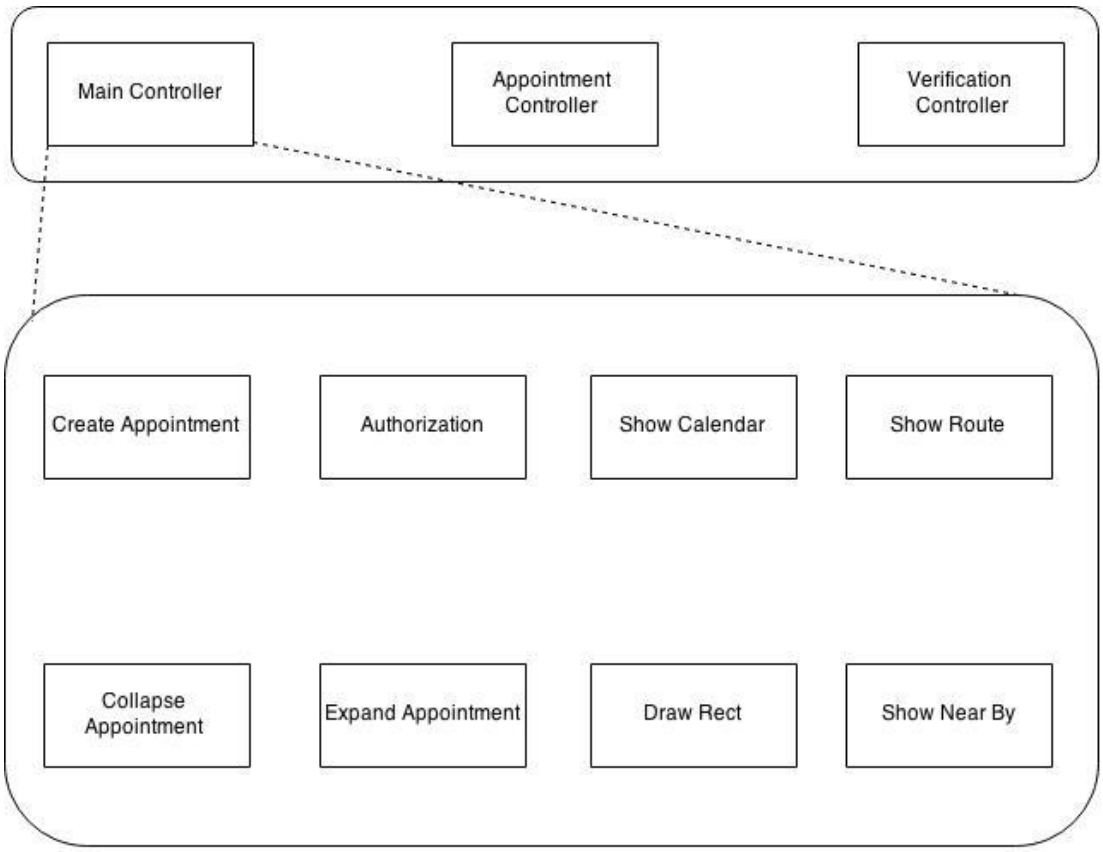

Figure 3.1.1.1. Main Controller in detail

3.1.1.2 Appointment Controller:

Appointment Controller is responsible for handling all logics that could be executed on the appointment view like create, view, delete and edit Appointment

Table 3.1.1.2. Appointment Controller functionalities

Caller	Callee Function	Output
/view/ApptFormContainer	submitApptForm()	Success/Failure Pop Up Message
/view/ApptFormContainer	deleteAppt()	success: void cancel: false
/controller/Main	getAppointments()	/store/Appointments
/controller/Main	createCalendarView() Google	Google Calendar Webpage
/controller/Main	createApptForm()	return application form view
/view/ApptFormContainer	SearchCustomer()	Displays Customers in List
/view/ApptFormContainer	PopulateResults()	Populates Customer name and Customers Address to Appointment form

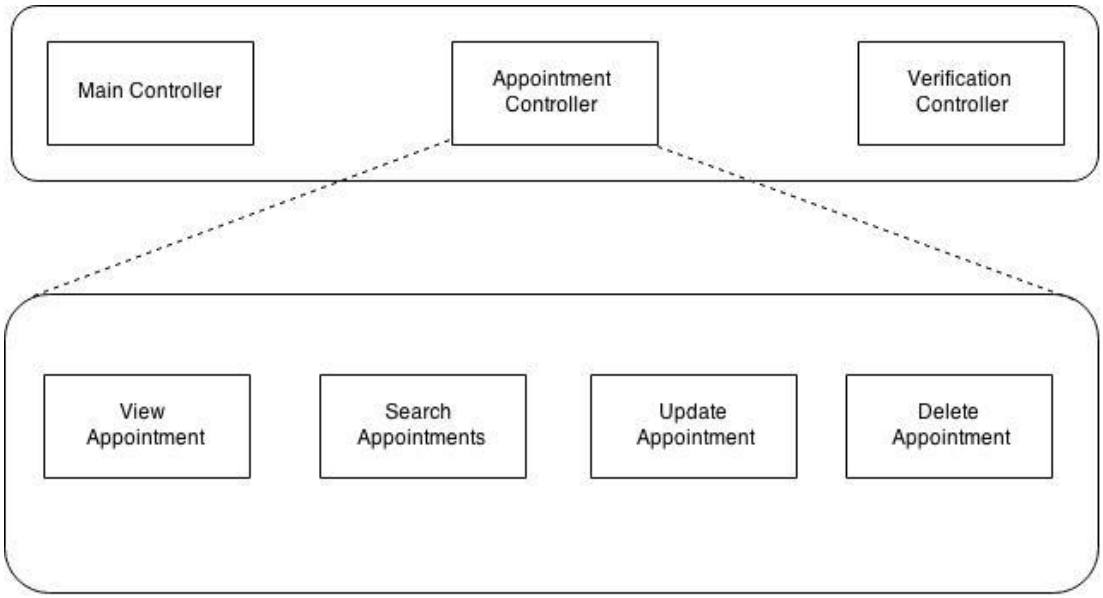

Figure 3.1.1.2 Appointment Controller in detail

3.1.1.3 Verification Controller:

Verification Controller is responsible for handling the authorization process of Google Calendar.

Table 3.1.1.3. Verification Controller functionalities

Caller	Callee Function	Output
/conroller/Main	makeVerification()	Google Authorization Webpage

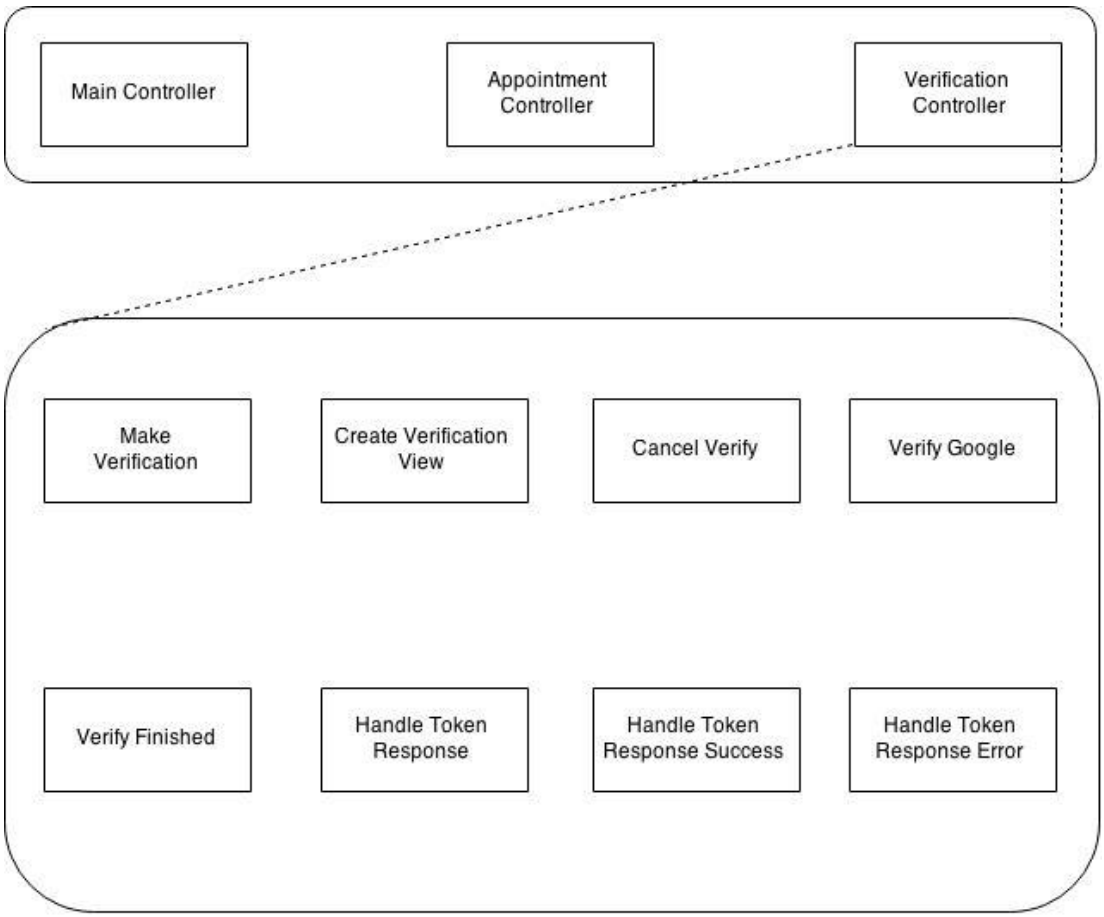

Figure 3.1.1.3 Verification Controller in detail

3.1.2 VIEW:

For the view part, we applied the composite strategy which means one view for user could be composed by several components.

Here is the mapping between the user view and the view components.

Table 3.1.2 Main View & View Components

User View (UI)	Components	Description
Main	/view/MainContainer	Main Container for holding components
	/view/Map	Component for rendering Appointment on the map. The map contains an array called "markers", where each member of the array is an object with two pieces of information: the information to show when the marker is clicked, and the actual Google marker object.
	/view/ApptList	Showing the appointment list
Appointment Detail	/view/ApptFormContainer	Main container for an appointment to be created, edited or deleted
	/view/TimePickerField	Time Picker component for an appointment detail
Authorization View	/view/VerificationContainer	Main container for managing the Google Calendar Authorization process

3.1.3 Data Model

The application currently has a single model - Appointment. The model distributes IDs with a UUID strategy provided by Sencha.

Table 3.1.3 Data Model

Field	Type	Req	Purpose	Example
title	string	Yes	Represents a human readable identifier for the instance.	**Meeting with** Team
location	string	Yes	The address / name of the location of the appointment (options procvided by Google Places Geolocation services).	123 Fake Street, Pittsburgh, PA, 15213
date	date	Yes	Date object representing when the appointment will occur (created by DatePicker field). The date will be stored with the start time as the time portion of the date value.	September 26th, 2013
start	string	Yes	Represents the start time of the appointment (created by Sencha TimePicker plugin)	13:00:00
end	string	Yes	Represents the start time of the appointment (created by Sencha TimePicker plugin)	14:00:00
customer	string	No	Some identifier of the customer.	John Smith
note	string	No	Any notes the agent may want to assign to the customer.	Discuss the Steelers

3.1.4 Validations

The model currently contains a total of 6 validations. The first 5 are validations for requiring the presence of each the 5 fields. The final validation is a custom validation to make sure the start time of the appointment is before the end time. To add this validation, the validate() method of the model was overwritten. These validations will run before the model instance is created or updated.

3.1.5 Data Store

There is currently a single store in our application - Appointments. The store is linked to the Appointment model, sorts the appointments by date and uses local storage (in-browser storage).
The *future* appointment model instances within the store are re-created each time the user opens up Trip Planner by pulling information from the Google Calendar API. When these instances are created, markers are placed onto the Map in the MainContainer view.

3.2 DETAILED DESIGN DESCRIPTION

3.2.1. Authorization:

The Trip planner application accesses the Google calendar of the user. Before the application access the calendar the user of the application has to authorize with their Google account. The application generates a unique API key that allows the application to access the Google Calendar.

3.2.2. Appointment List:

3.2.2.1 Display Appointments:

As soon as the user gets authorized the application will retrieve the user's appointments from his Google calendar and displayed in a tabular form. The application displays only current day's appointments on the list and is sorted according to time.

3.2.2.2 Create New Appointment:

The agent can schedule a new appointment by clicking on the new appointment button. The agent will specify the Appointment title, Customer Name, Appointment location, Date, Start & End Time, Notes.

3.2.2.3 Edit or Delete an existing appointment:

The Agent can edit and delete appointments by double clicking on the appointment item

3.2.2.4 Search Existing Customer:

In Appointment form, the agent can search the customer by specifying his/her name. If the search results are found, it will be displayed in a list. Agent will select the respective customer, so that the name of the customer and location of the customer will be automatically populated to the appointment form.

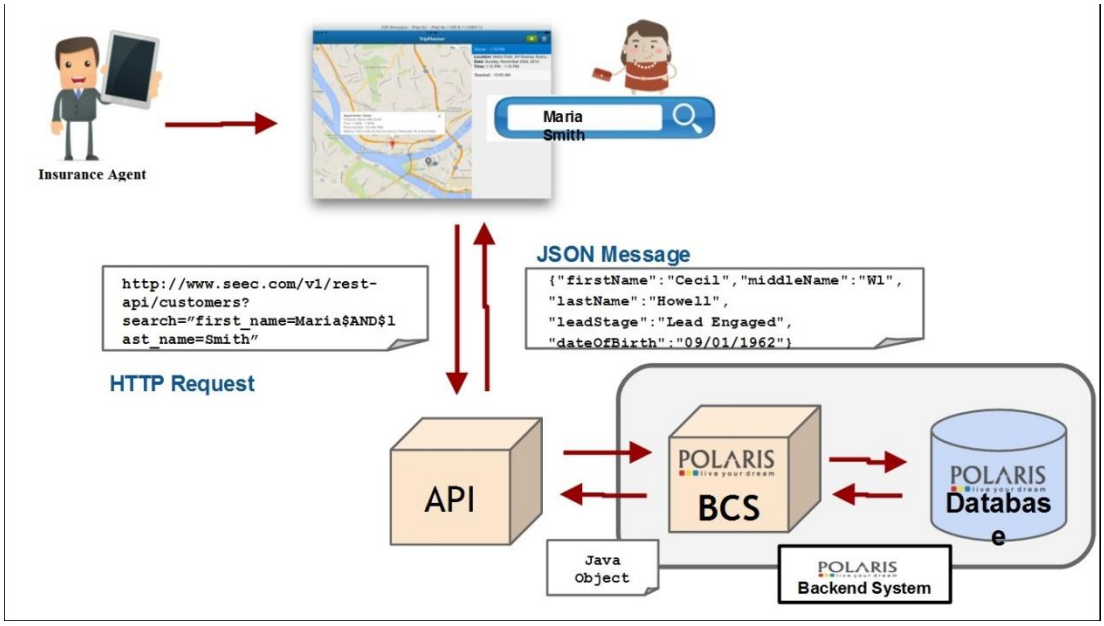

Fig 3.2.2.4 Search Customer Detailed Design

3.2.3. Map View:

3.2.3.1 Mark Scheduled Appointments:

The appointments that are displayed in a tabular form are marked on the Google maps. On Maps the current days appointments are displayed with Red Marker and the other markers are displayed with Grey Markers.

3.2.3.2 View Appointment Details:

When the agent hovers over the marker on the map, the application will display the details of the appointment in a info-window.

3.2.3.3 Update Current Location:

On clicking the ‘Update Current Location’ button on the map, the location on the map should be set to the agent’s current location.

3.2.3.4 Show Route:

On clicking the 'Show Route' button, the maps will display the route of current day’s appointments from the first appointment location to last appointment location including the other appointments as waypoints.

3.2.3.5 Show Near By:

On selecting the Show Near By button, a pop up window with current days appointments will be displayed. The agents selects the required appointments and once it was selected the application display one more pop up asking to select 'Search nearby Restaurants', 'Search nearby Parking Garages', 'Search nearby ATM's', 'Search nearby Banks' ,'Search nearby Hospitals'.

The agents select any on option that is required and the respective results will be displayed on Google map with markers.

3.2.3.6 Show All Markers:

On clicking the 'Show All Markers' button on the map, the size of the map will adjust to show all the markers on the map. It will collect all the boundary details of the appointments and sets the map zoom so, that all the markers on the map will be visible.

3.2.3.7 Reset Zoom:

On clicking the 'Reset Zoom' button on the map, the map resets to the original view. It will display the location of the first appointment.

3.2.3.8 Search Customer by ZIP code:

On clicking the 'Search Customers by Zip' code button a pop up opens which captures the zip code and on submit will display the customers in that zip code.

3.2.3.9 Show Customers Near Map Centre:

On clicking the 'Show Customers Near Map Centre' button on the map, the customers at that zip code will be displayed.

3.2.3.10 Display Customers in a selected area:

The agent should be able to select an area (like a rectangle or square) on the map and he should be shown all the customers in that area.

3.2.4. Calendar View

On selecting the Calendar button, the Google calendar view will be displayed with the appointment details.

3.3 OVERALL SYSTEM ARCHITECTURE

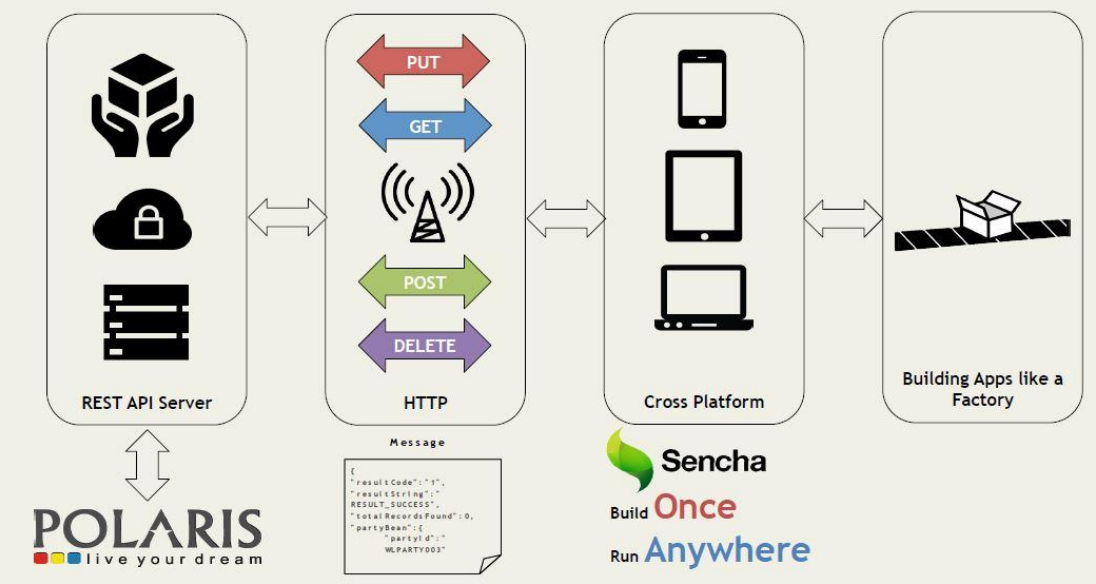

Fig 3.3 Overall System Architecture

3.4 UML DIAGRAMS

The Unified Modeling Language (UML) is a standard language for writing software blue prints. The UML is a language for

- Visualizing
- Specifying
- Constructing
- Documenting the artifacts of a software intensive system.

1. It is mainly used in the analysis of application.

1. It supports the entire Software Development life Cycle.
2. It supports diverse application areas.
3. Several CASE tools support it e.g. Rational, Together/j.
4. It is used to express the requirements of the software system we are developing.

3.4.1 USE CASE DIAGRAM

A use case diagram is a graph of actors set of use cases enclosed by a system boundary, communication associations between the actors and users and generalization among use cases. The use case model defines the outside (actors) and inside (use case) of the system's behavior.

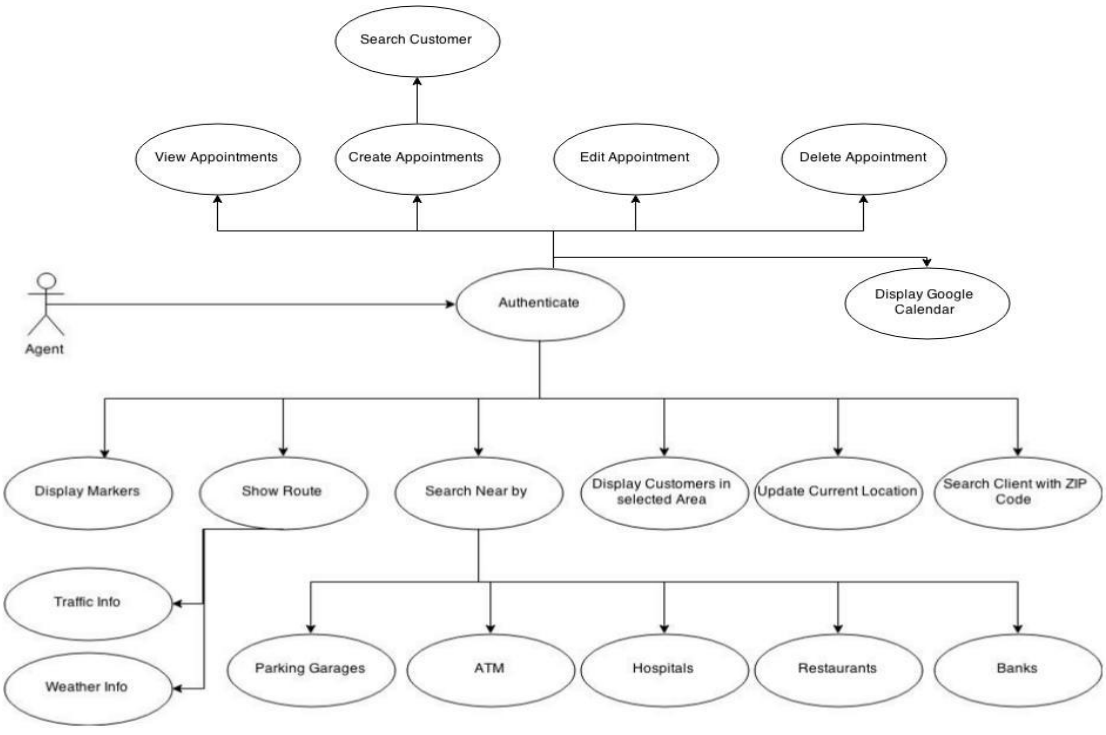

Fig 3.3.1 Use Case Diagram

3.4.2 CLASS DIAGRAM

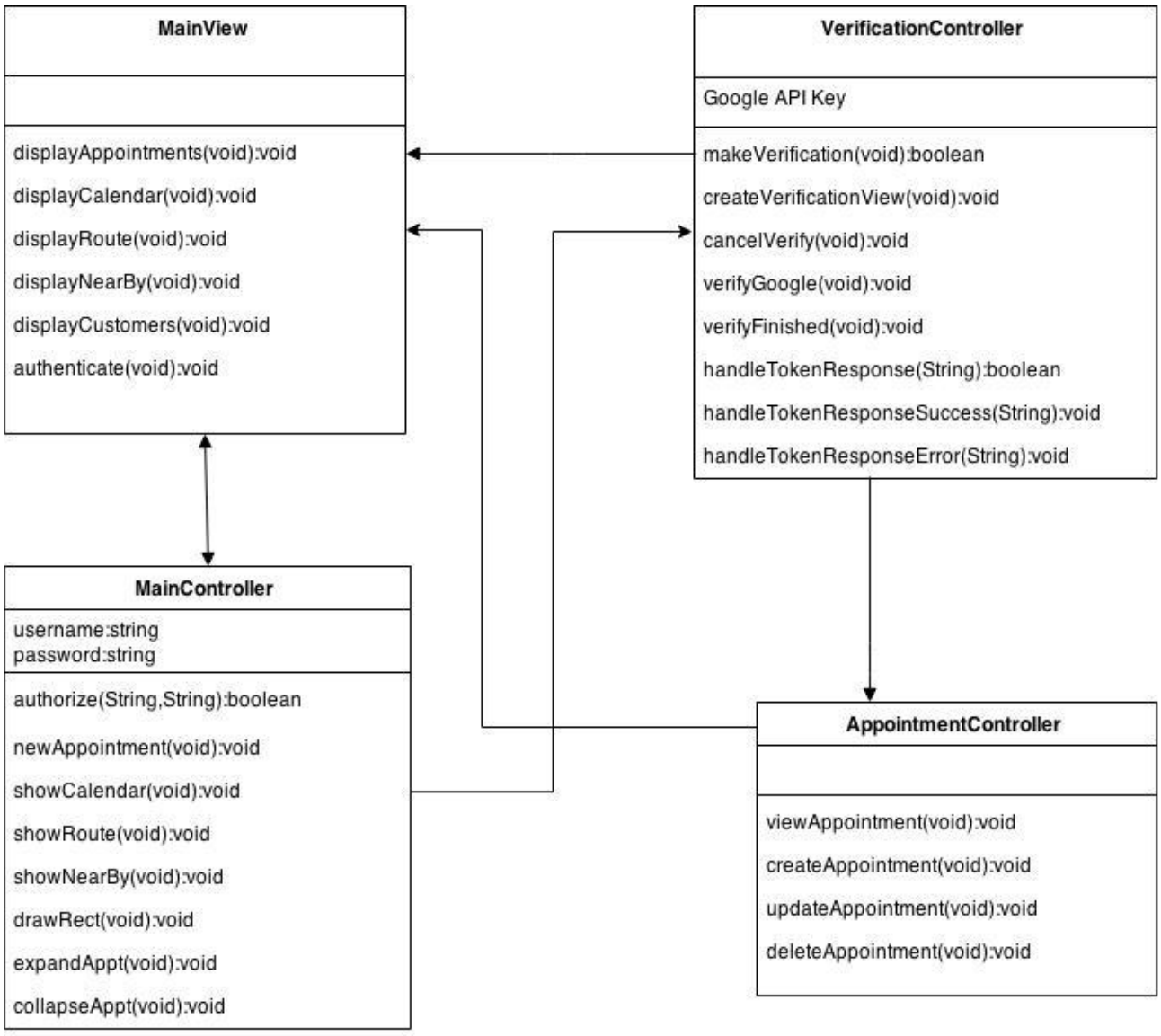

Fig 3.3.2 Class Diagram

3.4.3 ACTIVITY DIAGRAM

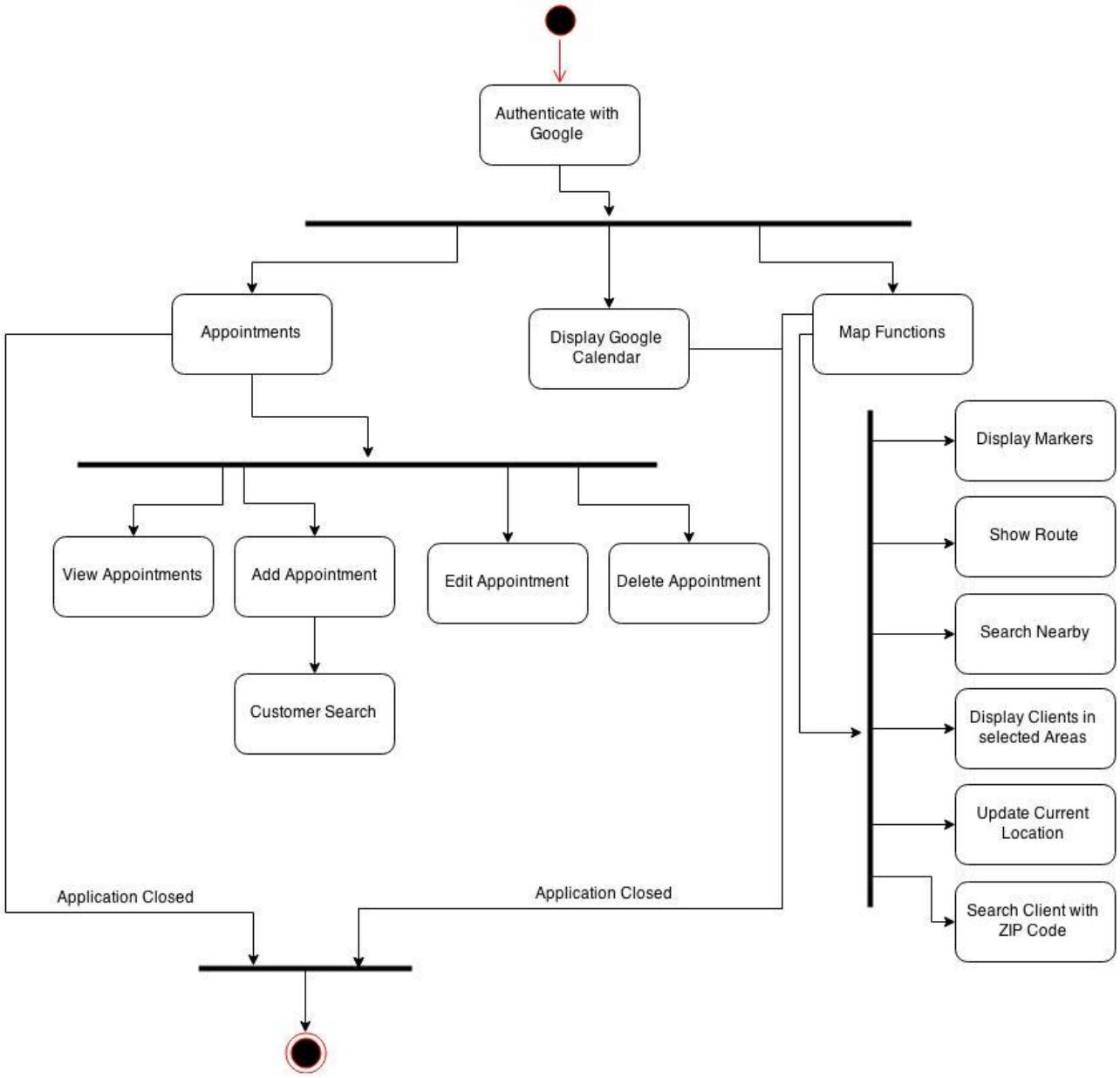

Fig 3.3.3 Activity Diagram

4. IMPLEMENTATION OF SYSTEM/METHODLOGY

4.1 Tools used in implementation

The tools that we have used to implement the project are as follows

- Sencha Touch/Apache Cordova
- Eclipse IDE with Android Development Tools plug-in.

4.1.1 Sencha Touch/ Apache Cordova

Sencha Touch:-

Sencha Touch is the leading MVC-based JavaScript framework for building cross-platform mobile web applications. Sencha Touch leverages hardware acceleration techniques to provide high-performance UI components for mobile devices.

It has over 50 built-in UI components and native looking themes for all major mobile platforms, Sencha Touch provides everything you need to create impressive apps that work on iOS, Android, BlackBerry, Windows Phone, and more. A novel and adaptive layout engine, fluid animations, and smooth scrolling features allow developers to build applications that respond to user actions nearly instantaneously, much like native technologies.

Advantages in using Sencha Touch for Development:-

- High-Performance, Native-Looking UI Widgets
- Adaptive layouts, Animations, and smooth Scrollings
- Backend Agnostic Data package
- Advanced mobile Charting package
- Device features and native packaging

Apache Cordova:-

Apache Cordova is a platform for building native mobile applications using HTML, CSS And JavaScript.When using the Cordova APIs, an app can be built without any native code (Java, Objective-C, etc) from the app developer. Instead, web technologies are used, and they are hosted in the app itself locally (generally not on a remote http server). And because these JavaScript APIs are consistent across multiple device platforms and built on web standards, the app should be portable to other device platforms with minimal to no changes. Apps using

Cordova are still packaged as apps using the platform SDKs, and can be made available for installation from each device's app store.

Cordova provides a set of uniform JavaScript libraries that can be invoked, with device-specific native backing code for those JavaScript libraries. Cordova is available for the following platforms: iOS, Android, Blackberry, Windows Phone, Palm WebOS, Bada, and Symbian.

4.1.2 Eclipse with ANDROID DEVELOPMENT TOOL Plug-In

Android Development Tools (ADT) is a plug-in for the Eclipse IDE that is designed to give us a powerful, integrated environment in which to build Android applications. ADT extends the capabilities of Eclipse to let us quickly set up new Android projects, create an application UI, add packages based on the Android Framework API, debug our applications using the Android SDK tools, and even export signed (or unsigned) .apk files in order to distribute our application.

Developing in Eclipse with ADT is highly recommended and is the fastest way to get started. With the guided project setup it provides, as well as tools integration, custom XML editors, and debug output pane, ADT gives you an incredible boost in developing Android applications.

In our project, we used Eclipse IDE for creating the user interface for each module and also coding each module which each module request another program written in php.

Creating an Android project

The ADT plug-in provides a New Project Wizard that we can use to quickly create a new Android project (or a project from existing code). To create a new project:

- Select File > New > Project.
- Select Android > Android Project, and click Next.
- Select the contents for the project:

Enter a *Project Name*. This will be the name of the folder where our project is created.

Under Contents, select Create new project in workspace. Select your project workspace location.

Enter an *Application name*. This is the human-readable title for your application — the name that will appear on the Android device.

Enter a *Package name*. This is the package namespace (following the same rules as for packages in the Java programming language) where all your source code will reside.

Select *Create Activity* (optional, of course, but common) and enter a name for your main Activity class.

Enter a *Min SDK Version*. This is an integer that indicates the minimum API Level required to properly run your application. Entering this here automatically sets the min Sdk Version attribute in the <uses-sdk> of your Android Manifest file. If you're unsure of the appropriate API Level to use, copy the API Level listed for the Build Target you selected in the Target tab.

After all the coding is done we need to specify the permissions that our application needs to access from the phone like Internet access,wi-fi access, accessing phone contacts etc.

Running your application:

Before you can run your application on the Android Emulator, you must create an Android Virtual Device (AVD). An AVD is a configuration that specifies the Android platform to be used on the emulator.

Creating an AVD:

With ADT 0.9.3 and above, the Android SDK and AVD Manager provides a simple graphical interface for creating and managing AVDs.

To create an AVD with the AVD Manager:

Select Window > Android SDK and AVD Manager, or click the Android SDK and AVD Manager icon (a black device) in the Eclipse toolbar.

In the Virtual Devices panel, you'll see a list of existing AVDs. Click New to create a new AVD.

Fill in the details for the AVD.

Give it a name, a platform target, an SD card image (optional), and a skin (HVGA is default).

Click Create AVD.

Your AVD is now ready and you can close the AVD Manager. In the next section, you'll see how the AVD is used when launching your application on an emulator.

Running your application:

To run (or debug) your application, select Run > Run (or Run > Debug) from the Eclipse main menu. The ADT plug-in will automatically create a default launch configuration for the project.

4.2 SCREENSHOTS

4.2.1. Authorization Module

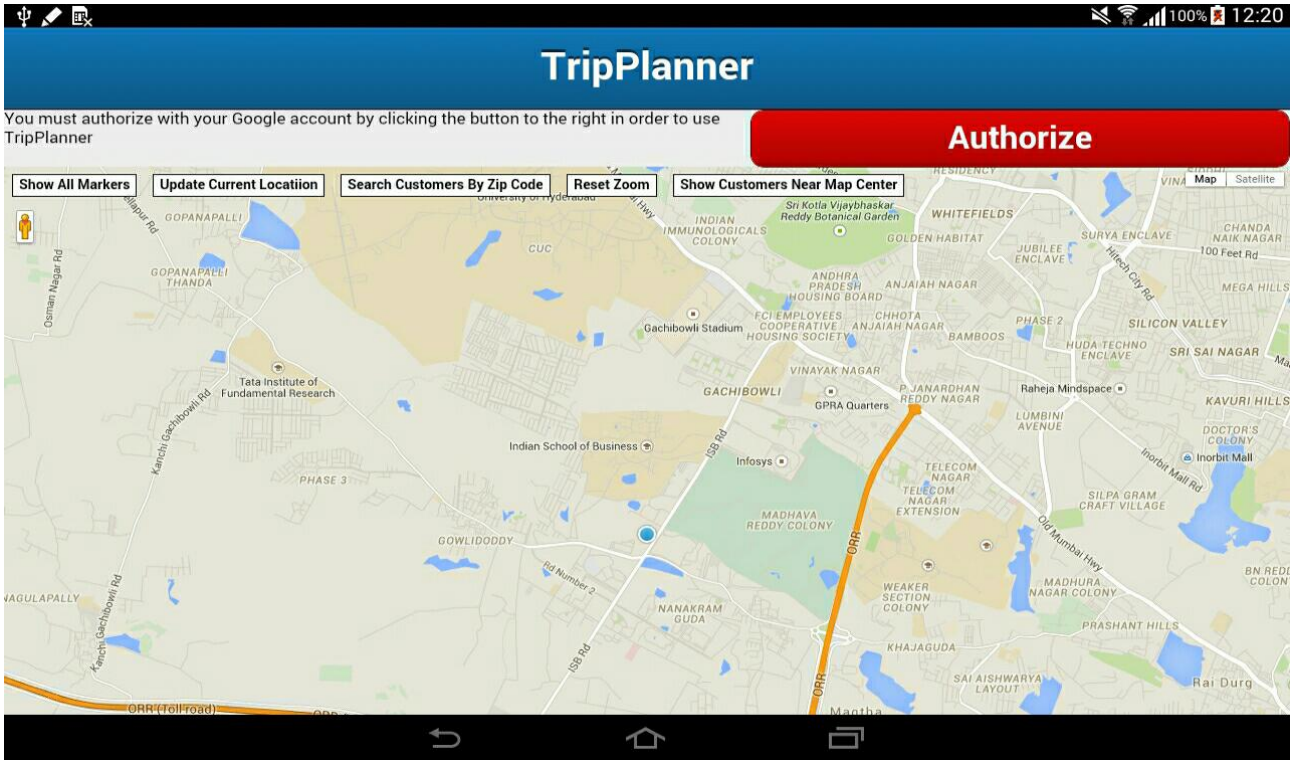

Fig 4.2.1.1 Application Home page before authorization

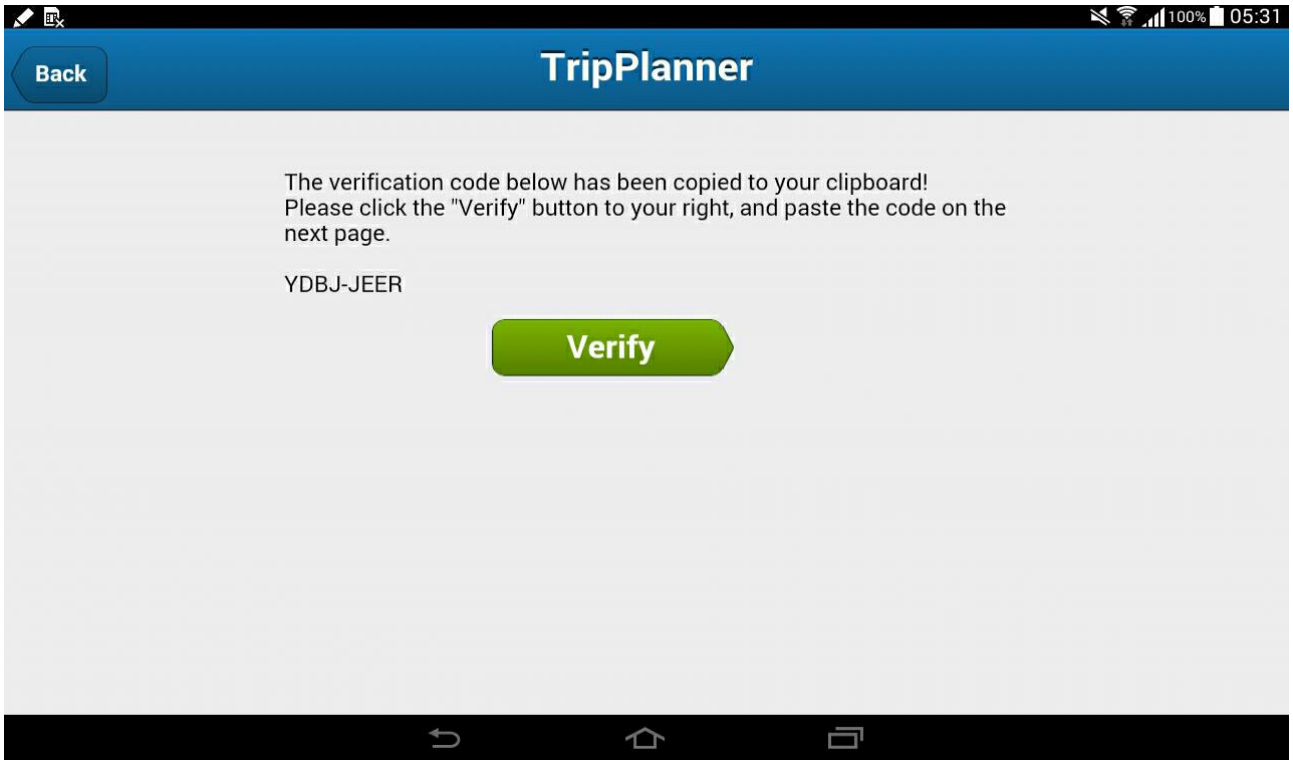

Fig 4.2.1.2 Authentication Token Generation

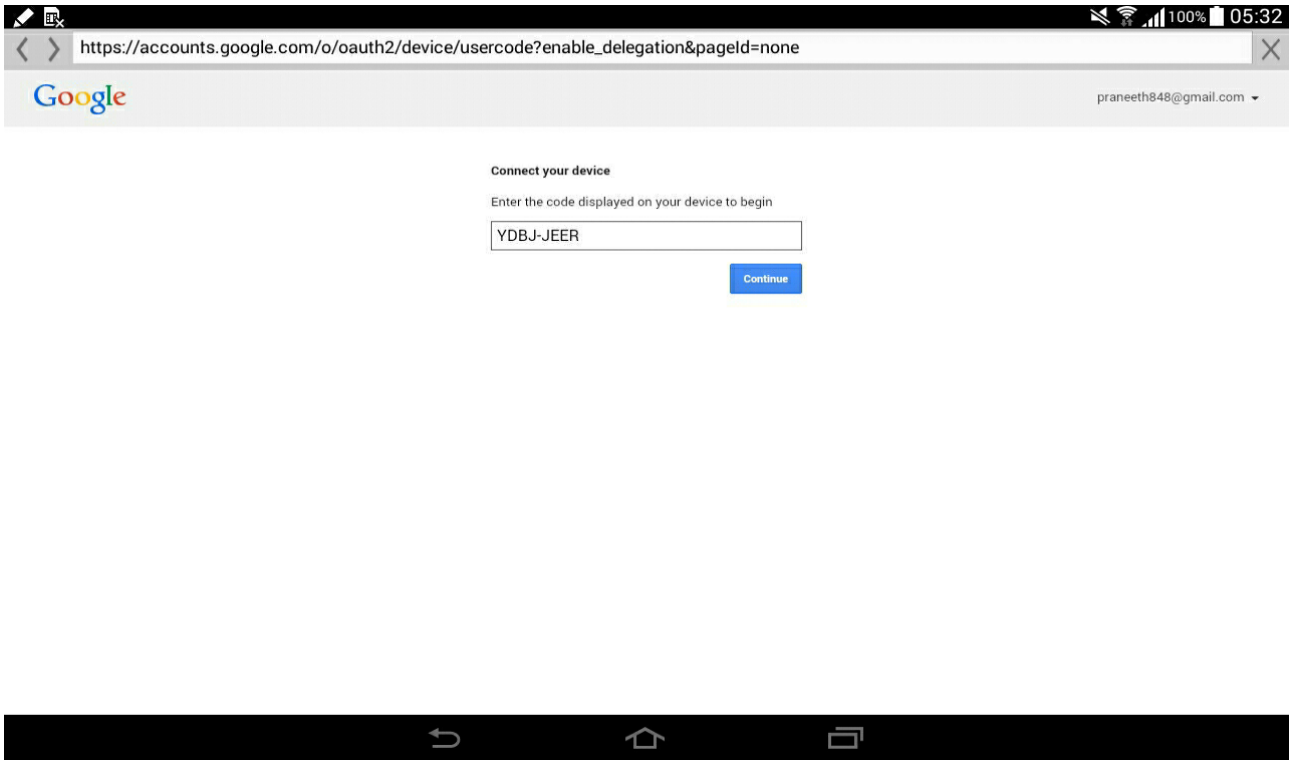

Fig 4.2.1.3 Token Verification by Google API

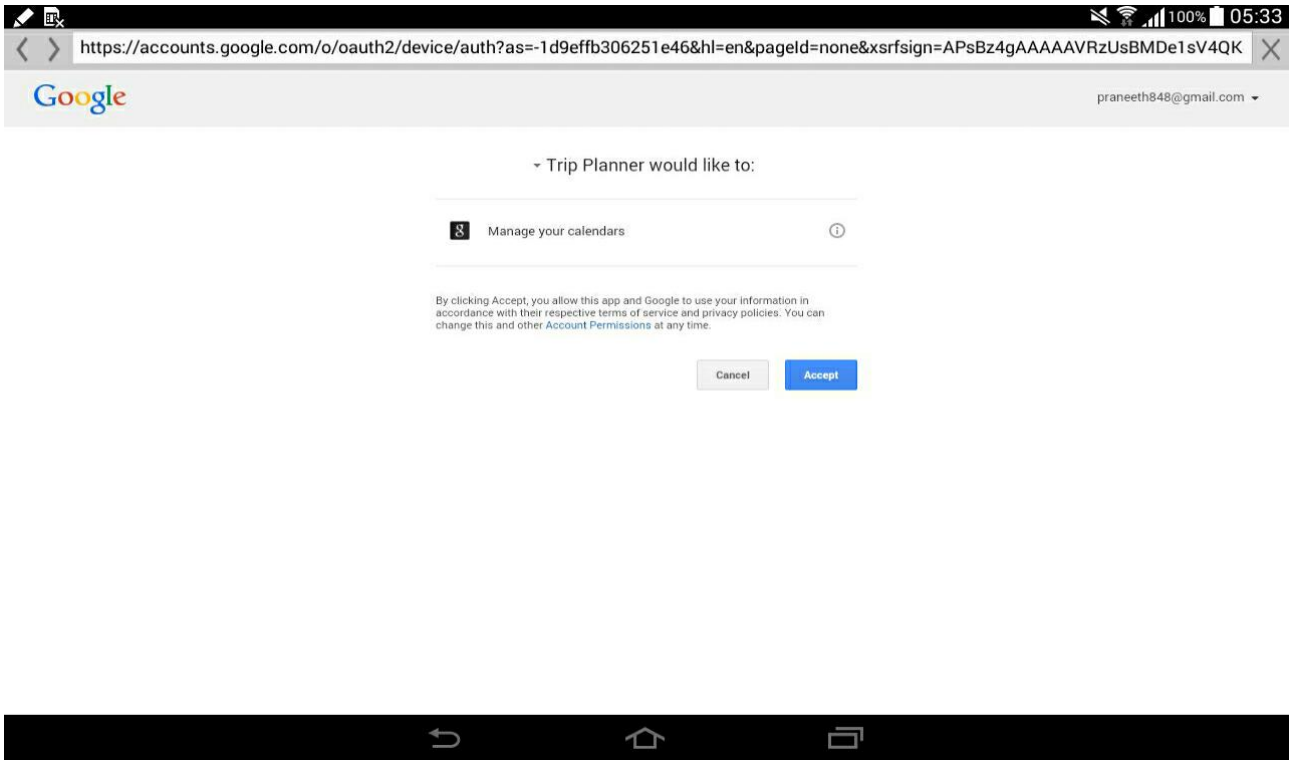

Fig 4.2.1.4 User Acceptance for Authorization

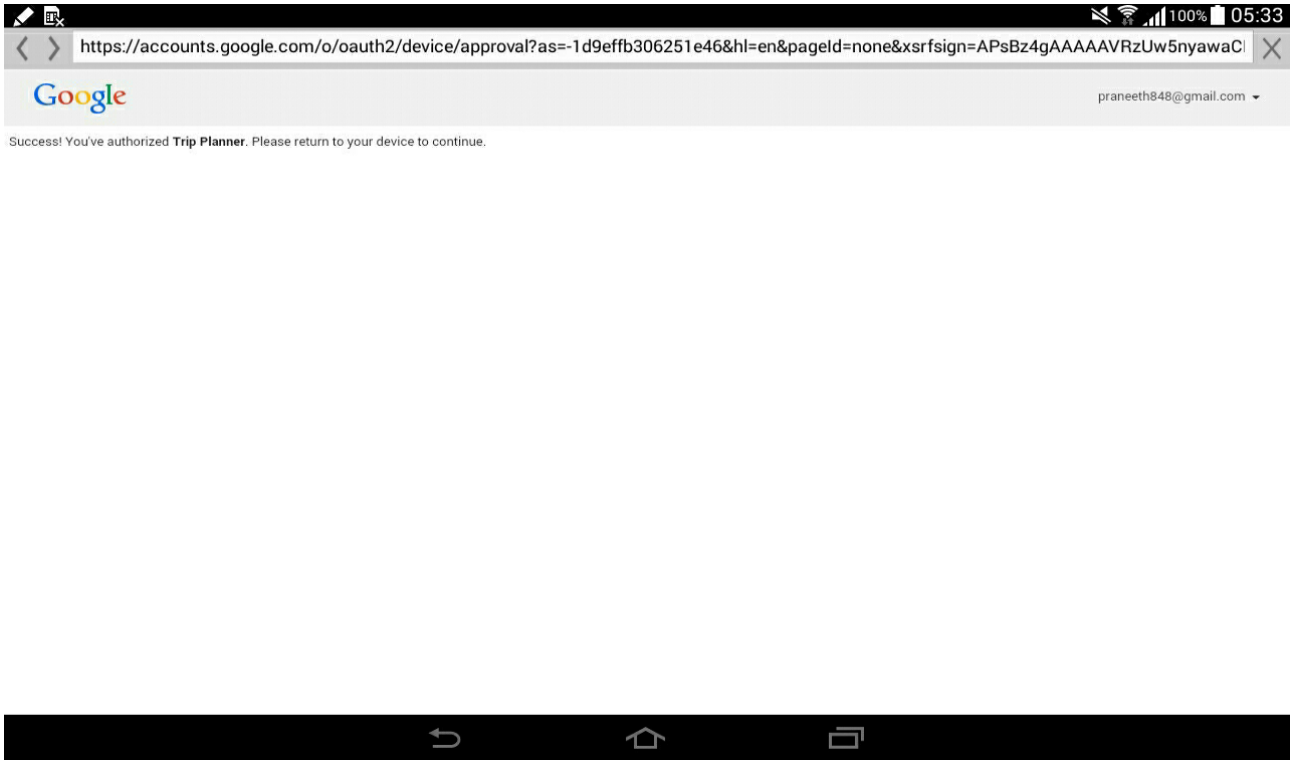

Fig 4.2.1.5 Authorization Successful

4.2.2. Appointment List Module

4.2.2.1 Display Appointments

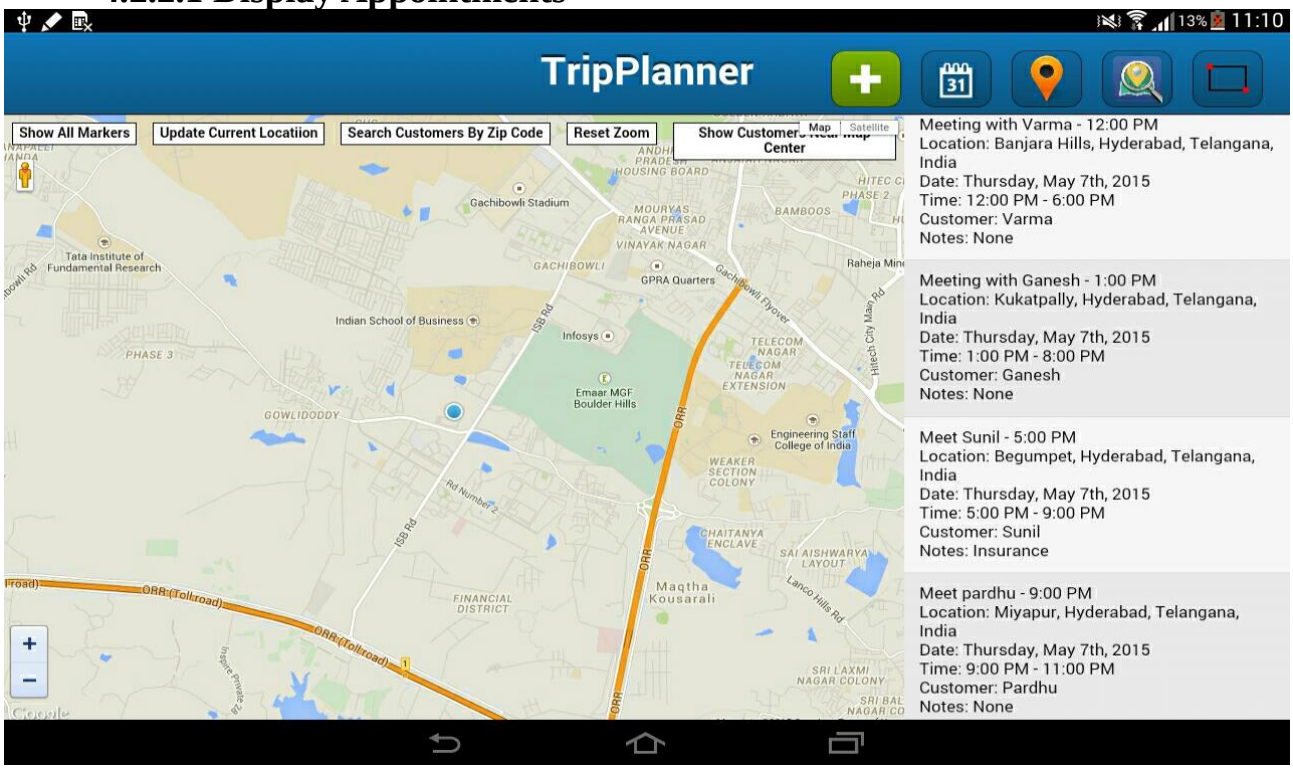

Fig 4.2.2.1 Displaying Appointments in list

4.2.2.2 Create New Appointment

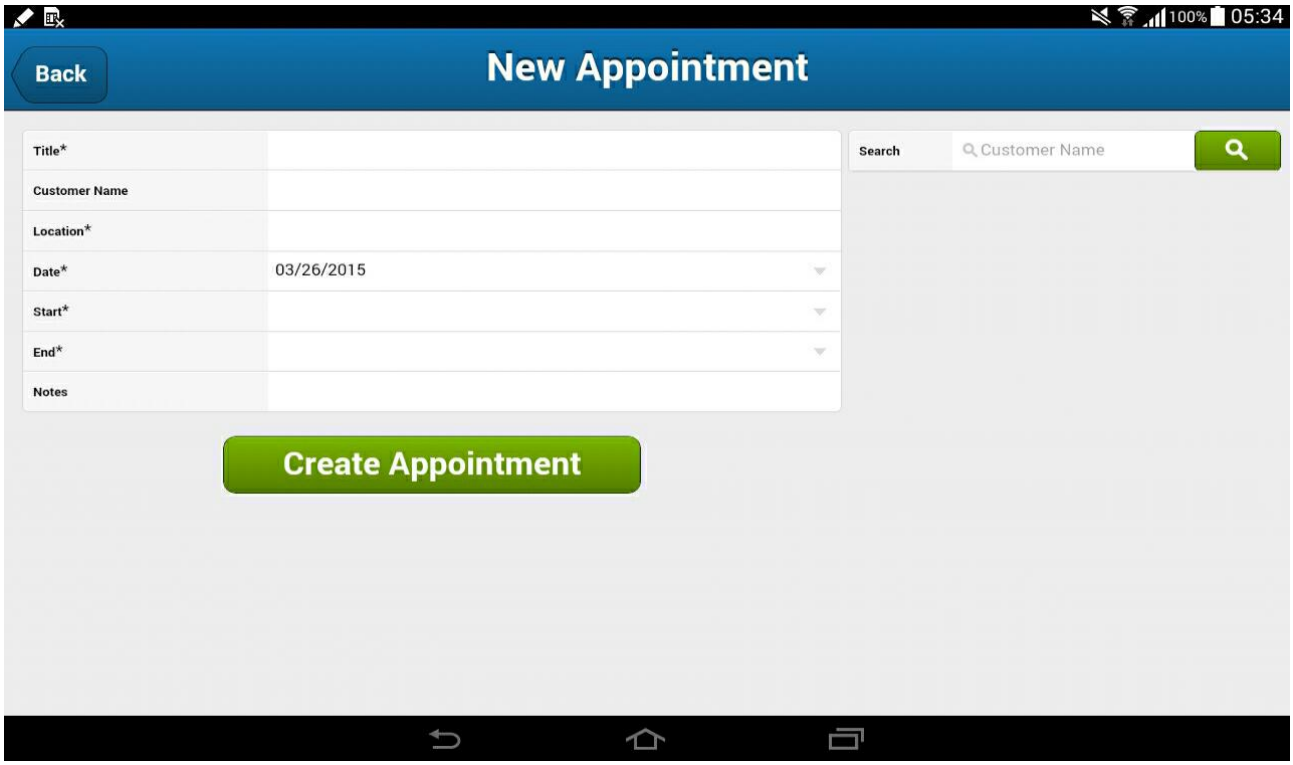

Fig 4.2.2.2.1 Create New Appointment Form

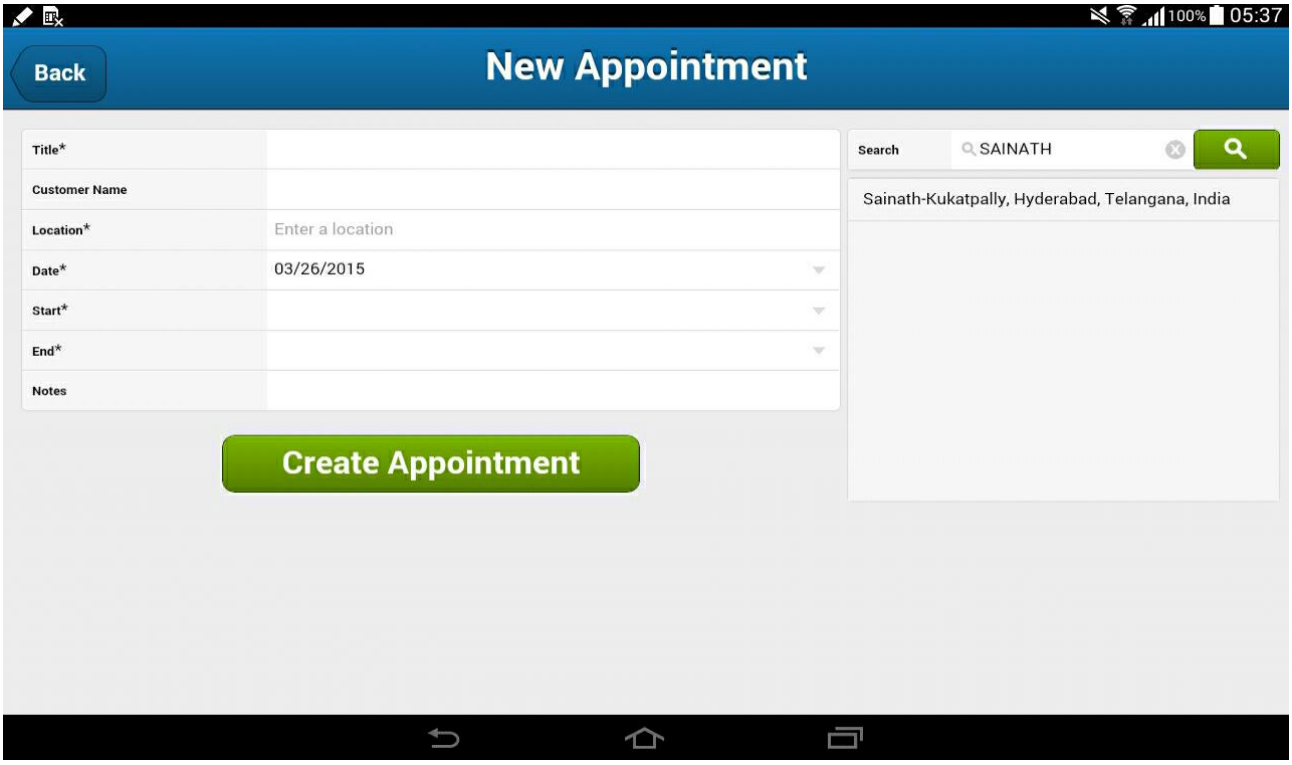

Fig 4.2.2.2.2 Search Customer

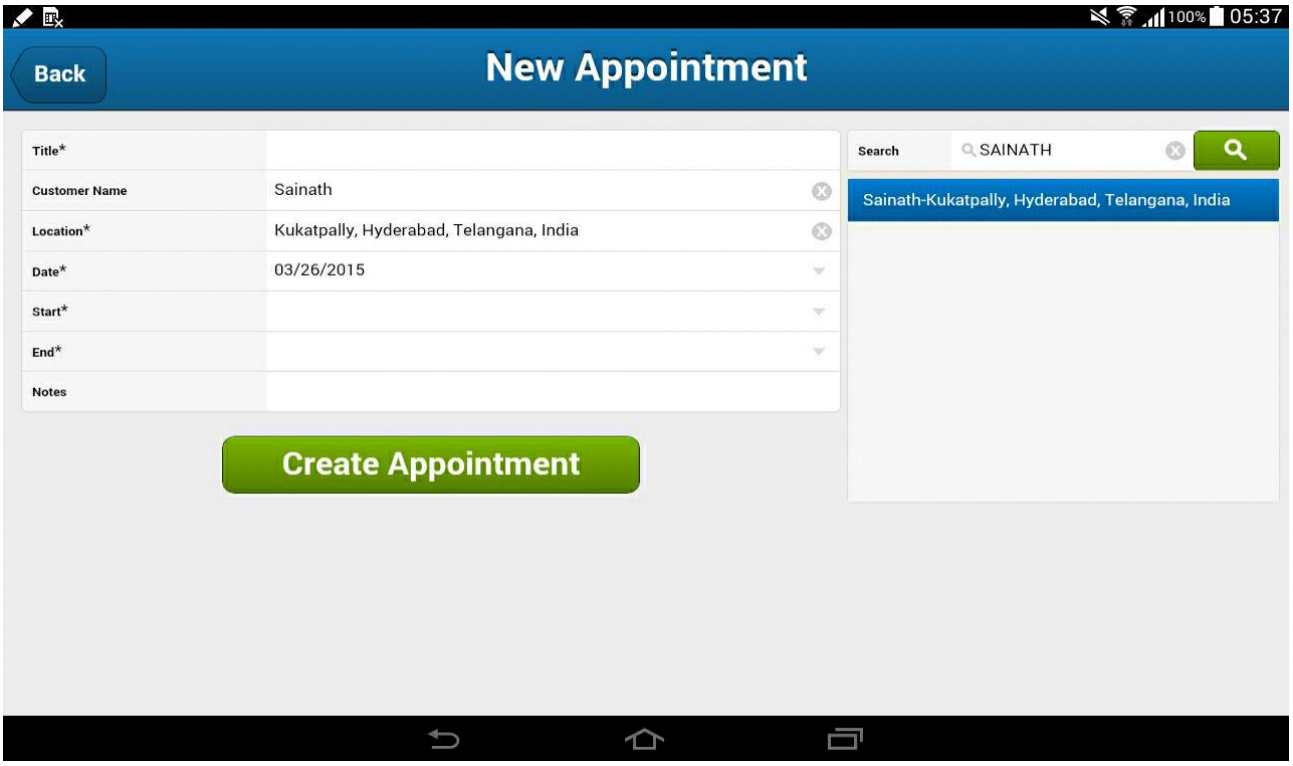

Fig 4.2.2.2.3 Search Results Populated to Appt Form

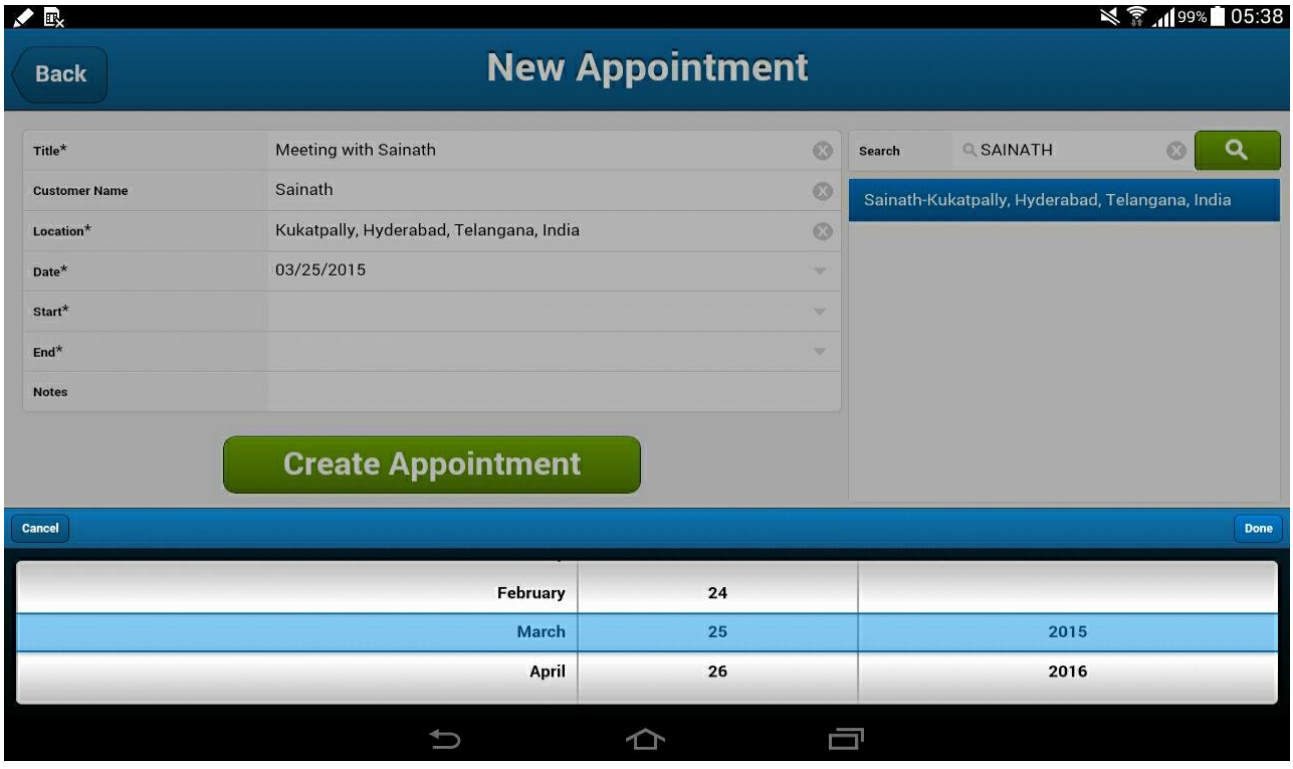

Fig 4.2.2.2.4 Date picker

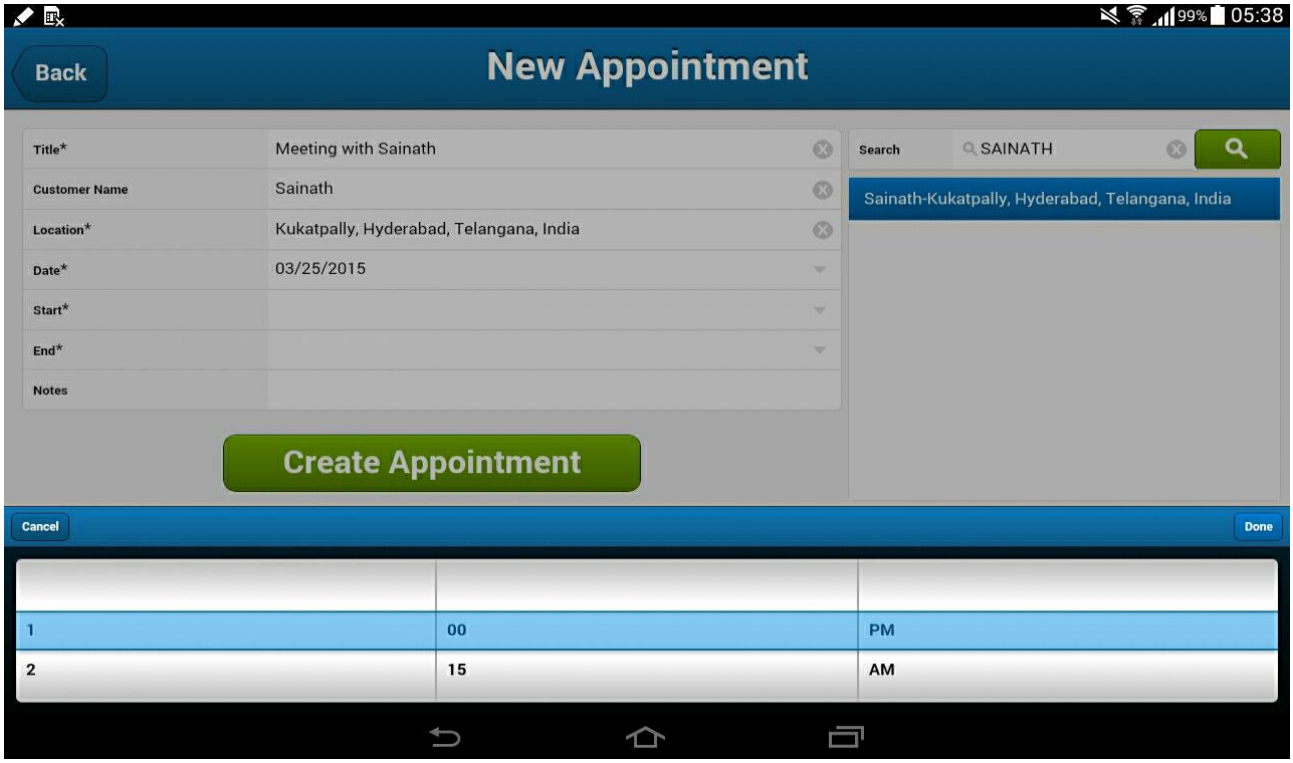

Fig 4.2.2.2.5 Time Picker

4.2.2.3 Edit or Delete an existing appointment

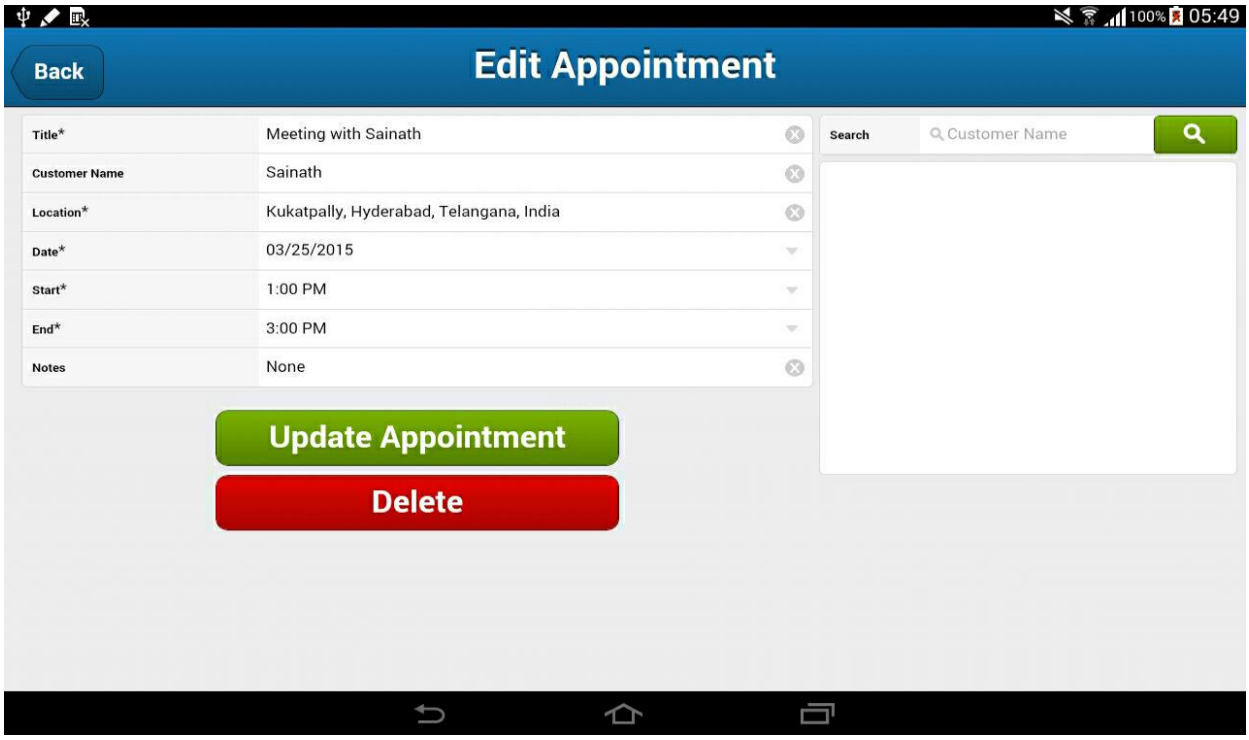

Fig 4.2.2.3 Edit/Delete Appt Form

4.2.2.4 Search Existing Customer

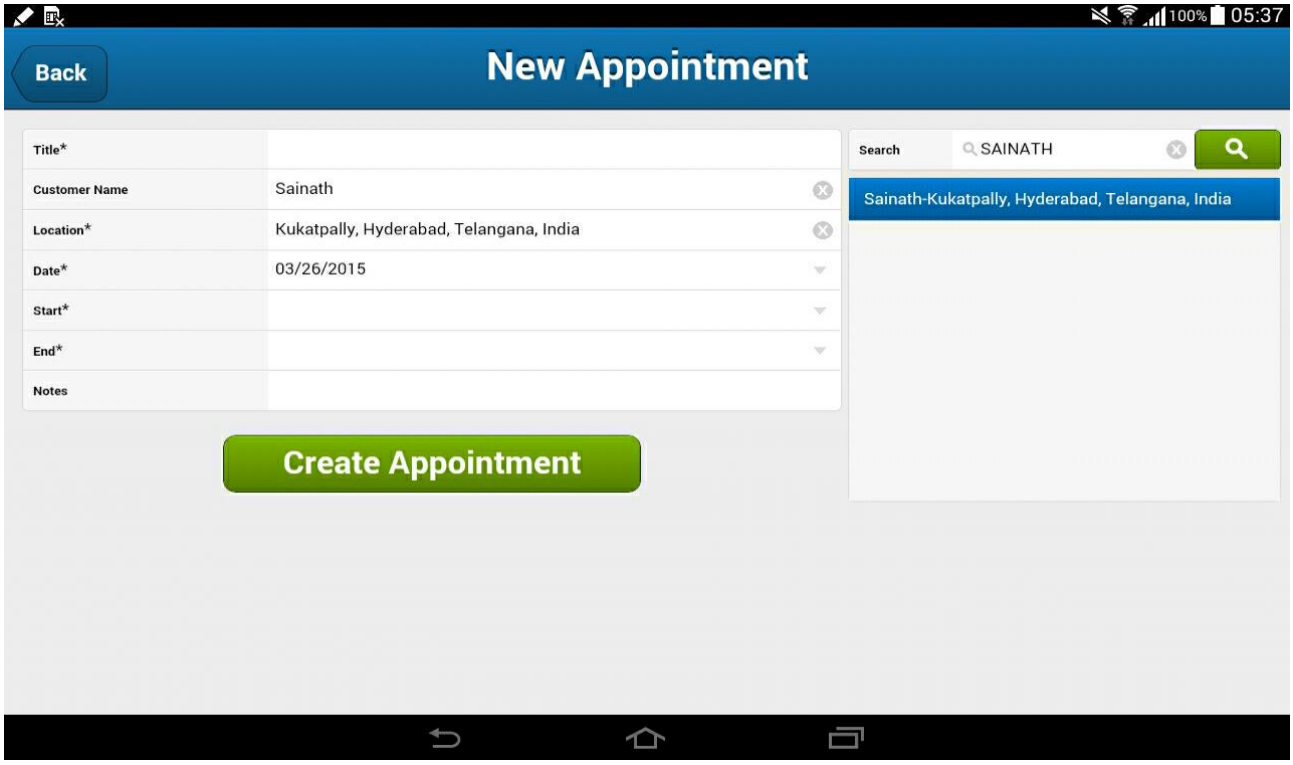

Fig 4.2.2.4 Search Existing Customer

4.2.3. Map View

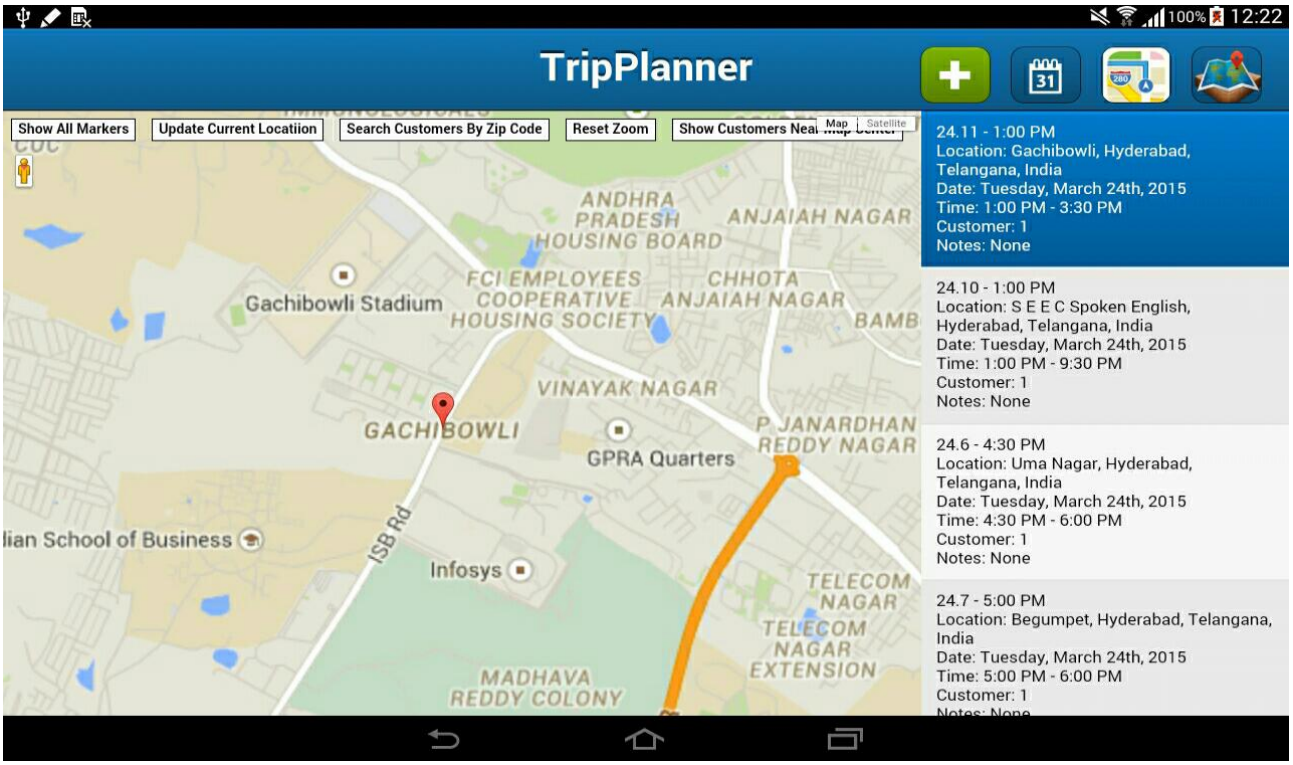

Fig 4.2.3.1 Mark Scheduled Appointments

4.2.3.2 View Appointment Details

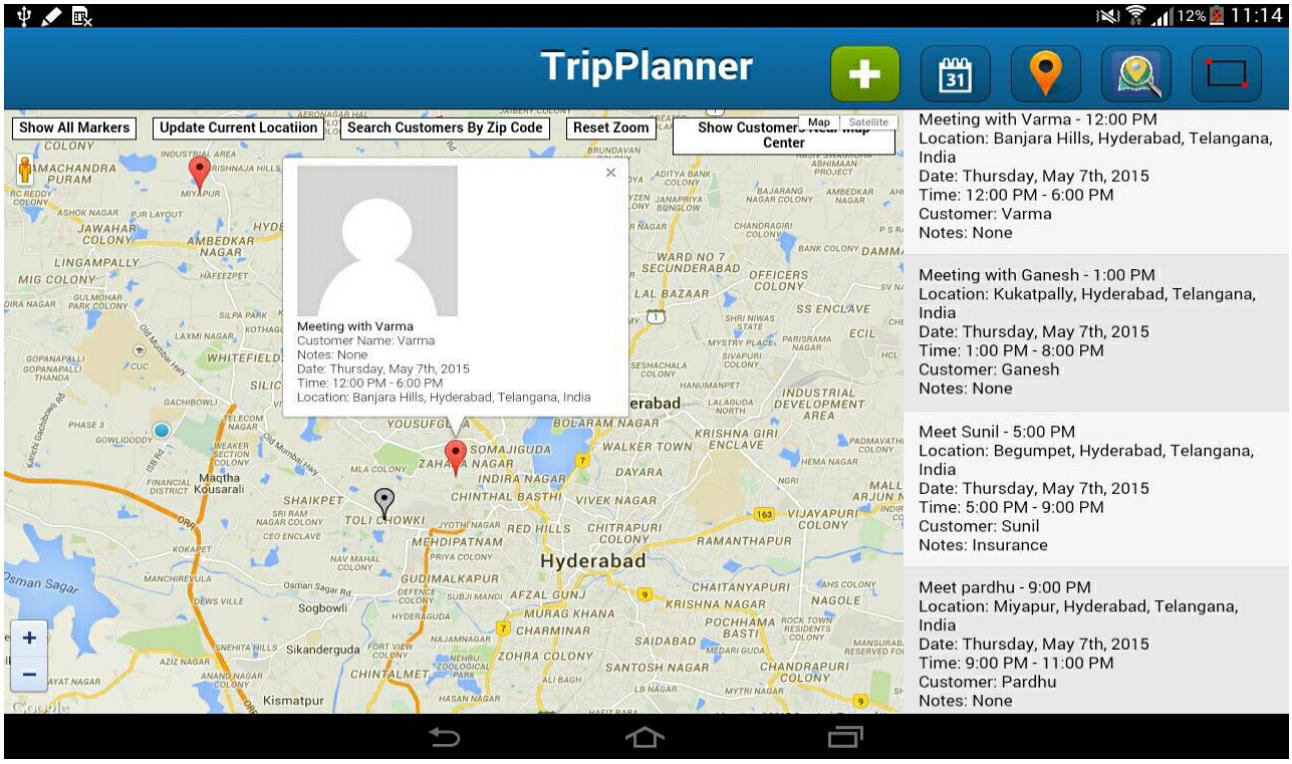

Fig 4.2.3.2 View Appointment Details

4.2.3.3Update Current Location

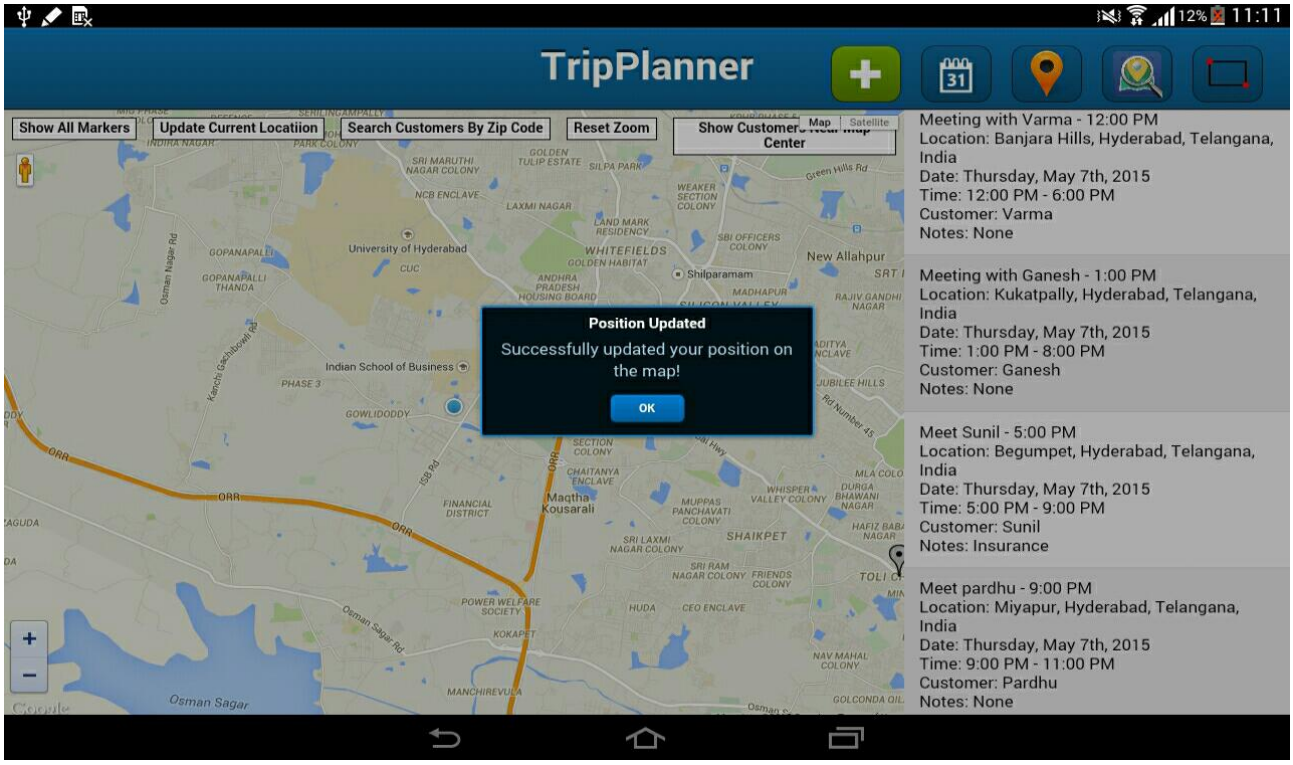

Fig 4.2.3.3Update Current Location

4.2.3.4 Show Route

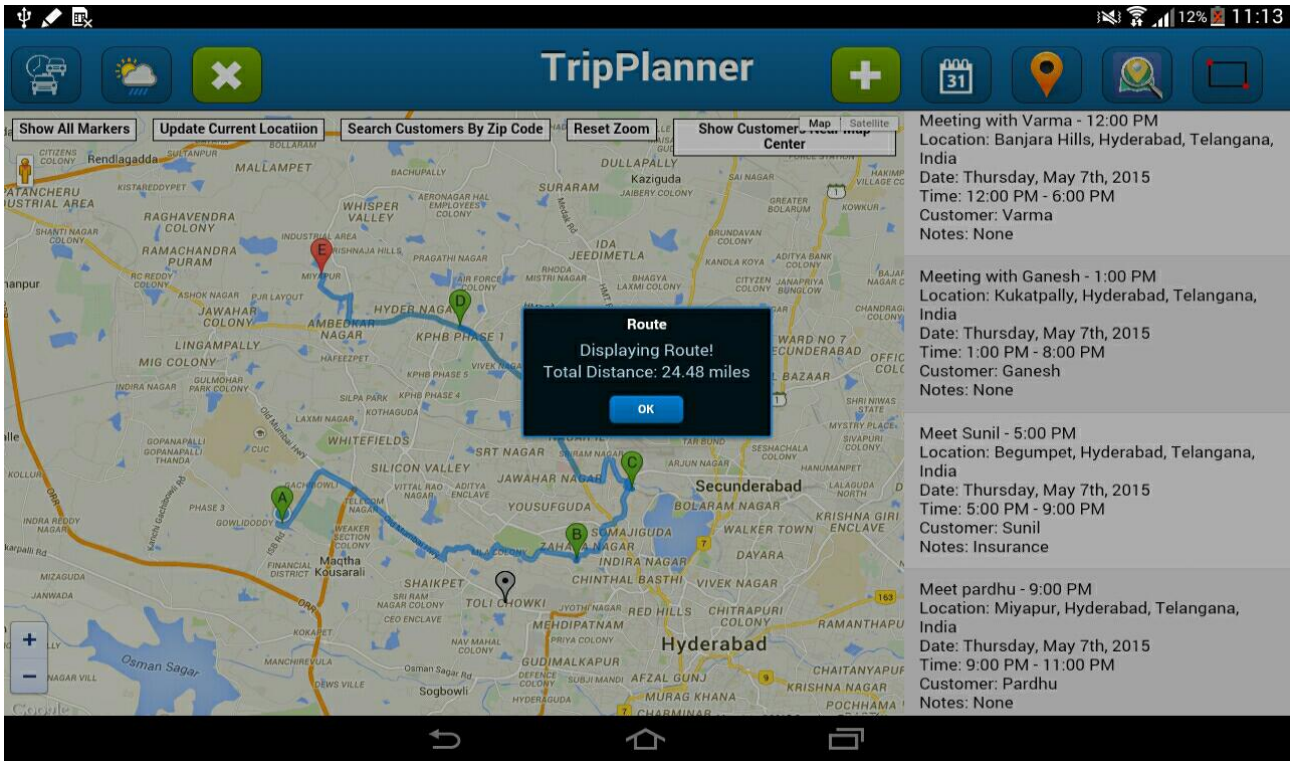

Fig 4.2.3.4.1 Show Route with Distance

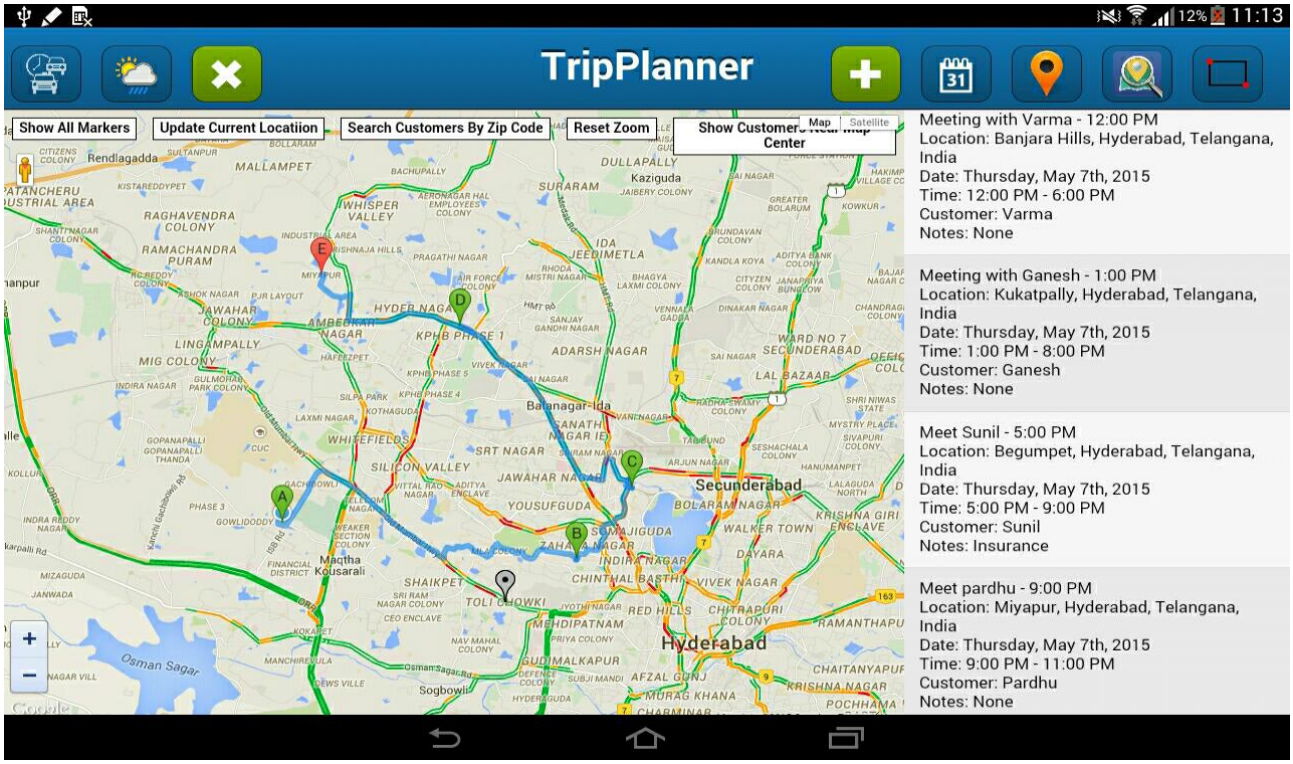

Fig 4.2.3.4.2 Show Route with Traffic Information

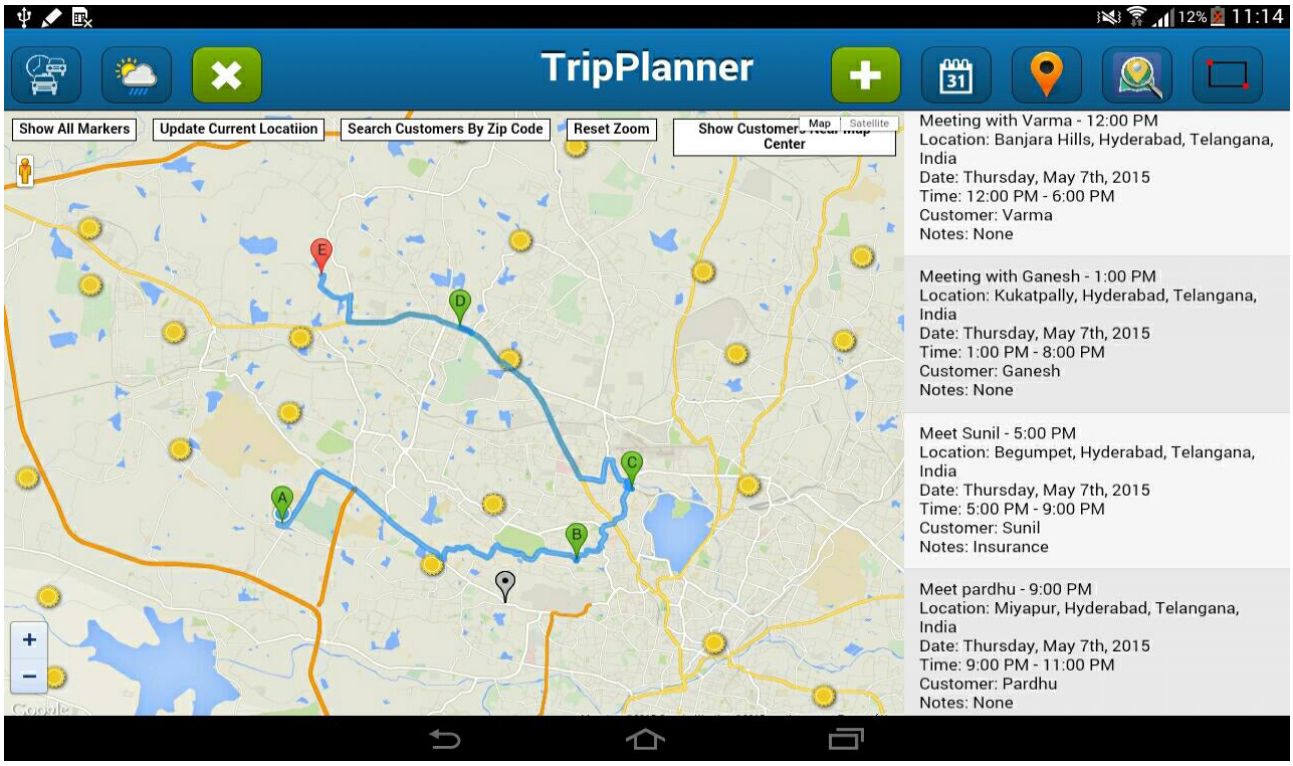

Fig 4.2.3.4.3 Show Route with Weather Information

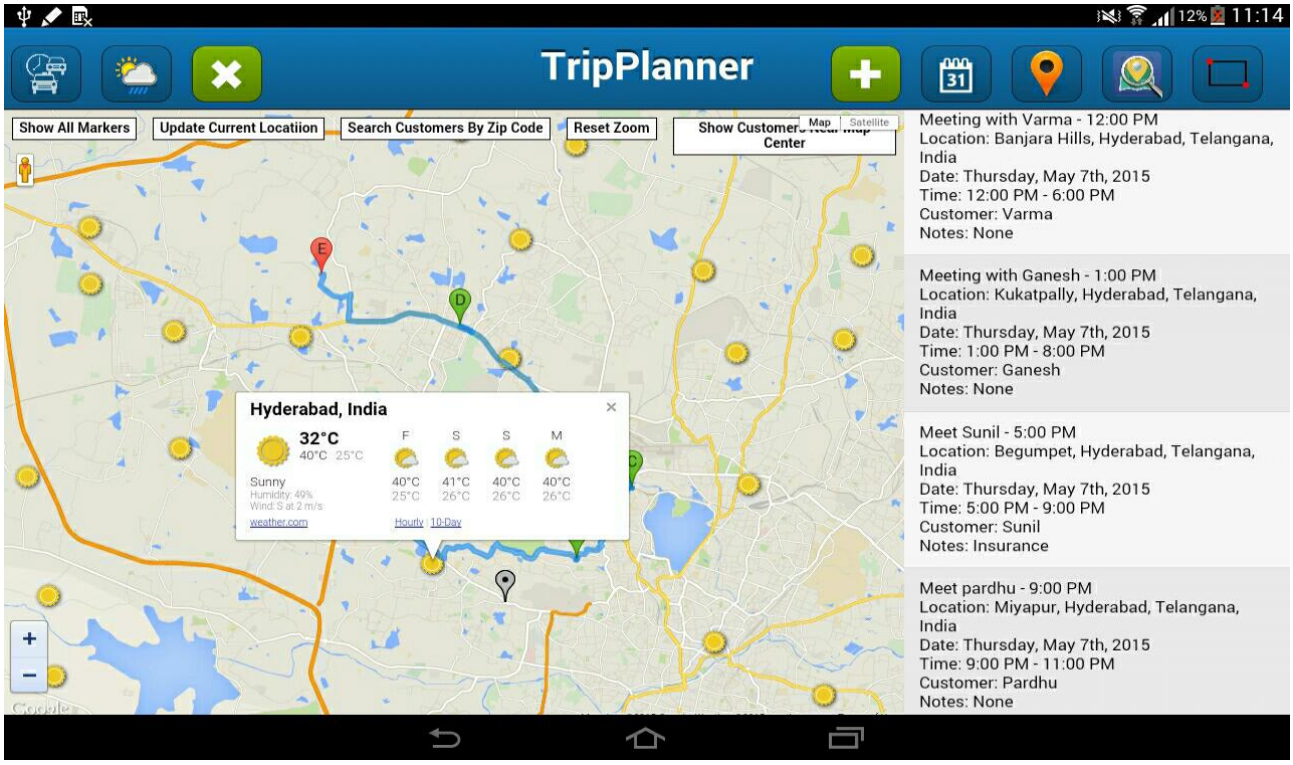

Fig 4.2.3.4.4 Show Route with Detailed weather Information

4.2.3.5 Show Near By

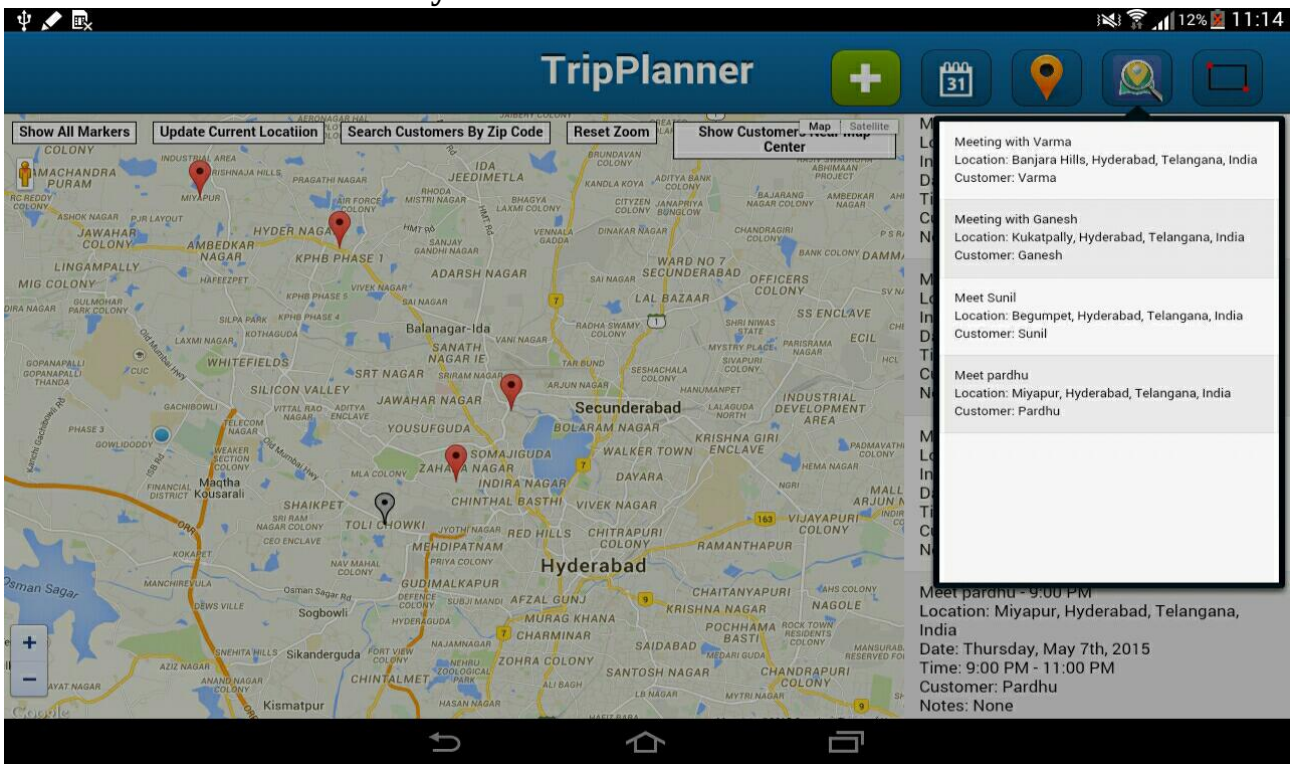

Fig 4.2.3.5.1 Show Near By select Appointment

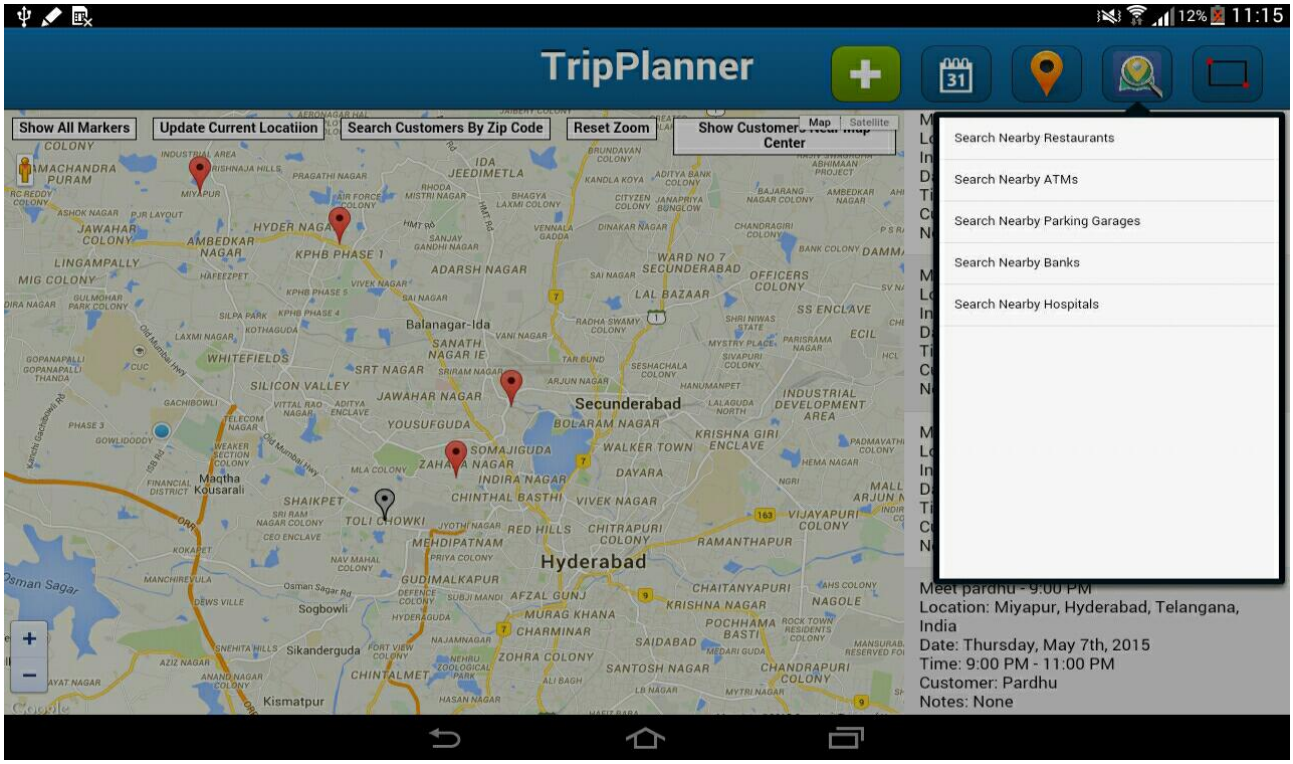

Fig 4.2.3.5.2 Show Near By, select search item

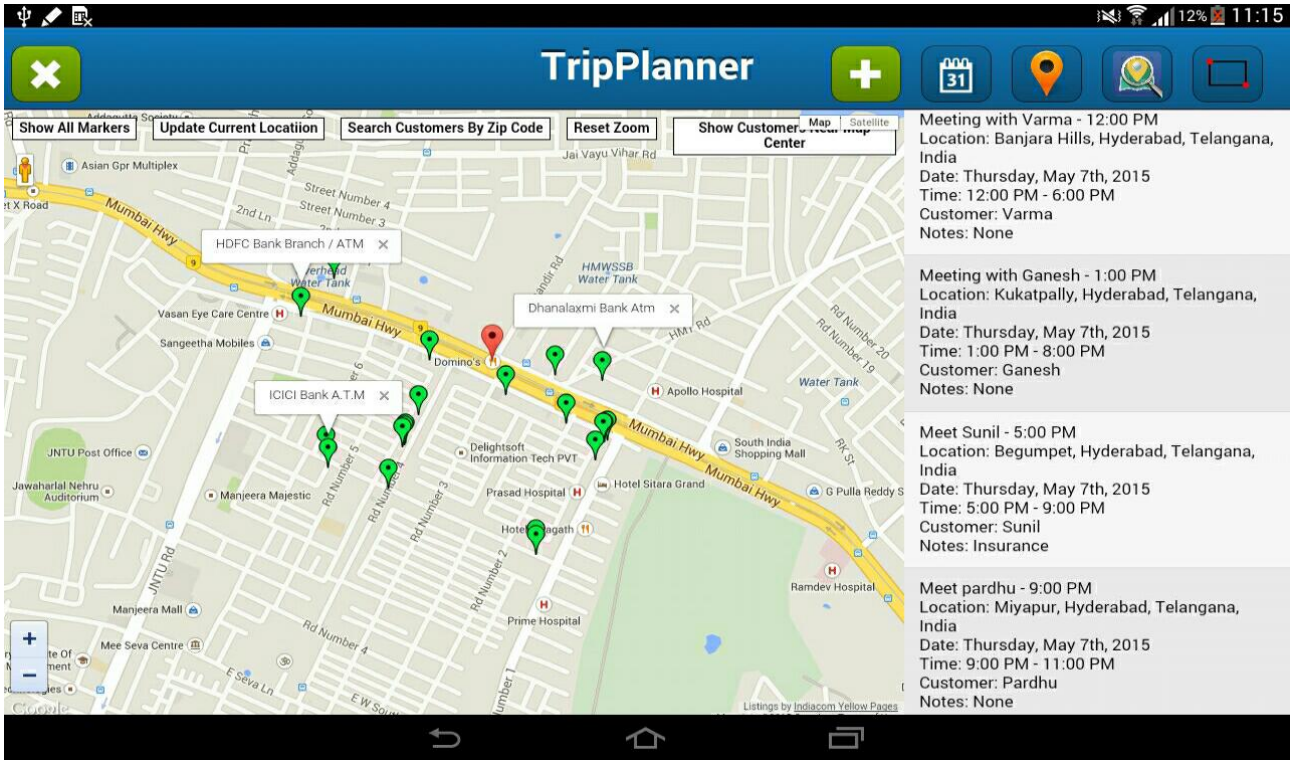

Fig 4.2.3.5.3 Displaying Show NearBy Results

4.2.3.6 Show All Markers

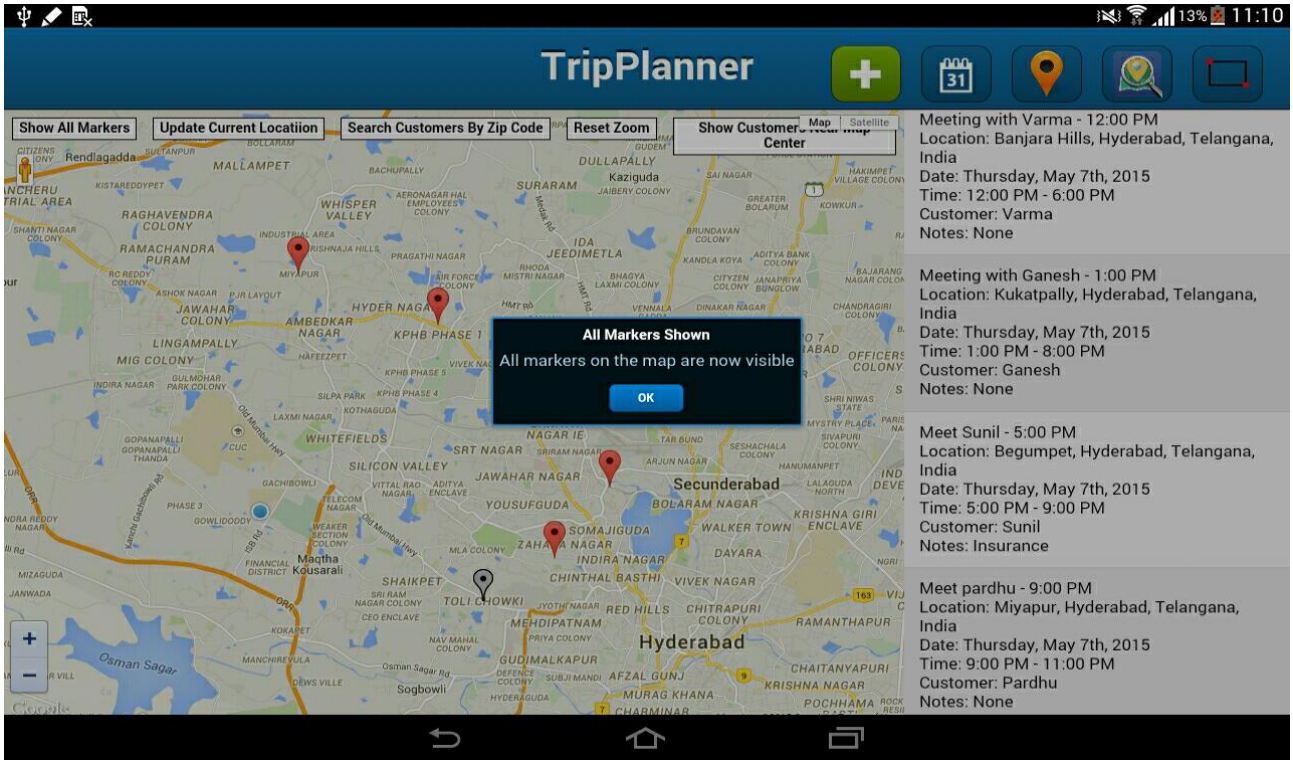

Fig 4.2.3.6 Show All Markers

4.2.3.7 Reset Zoom

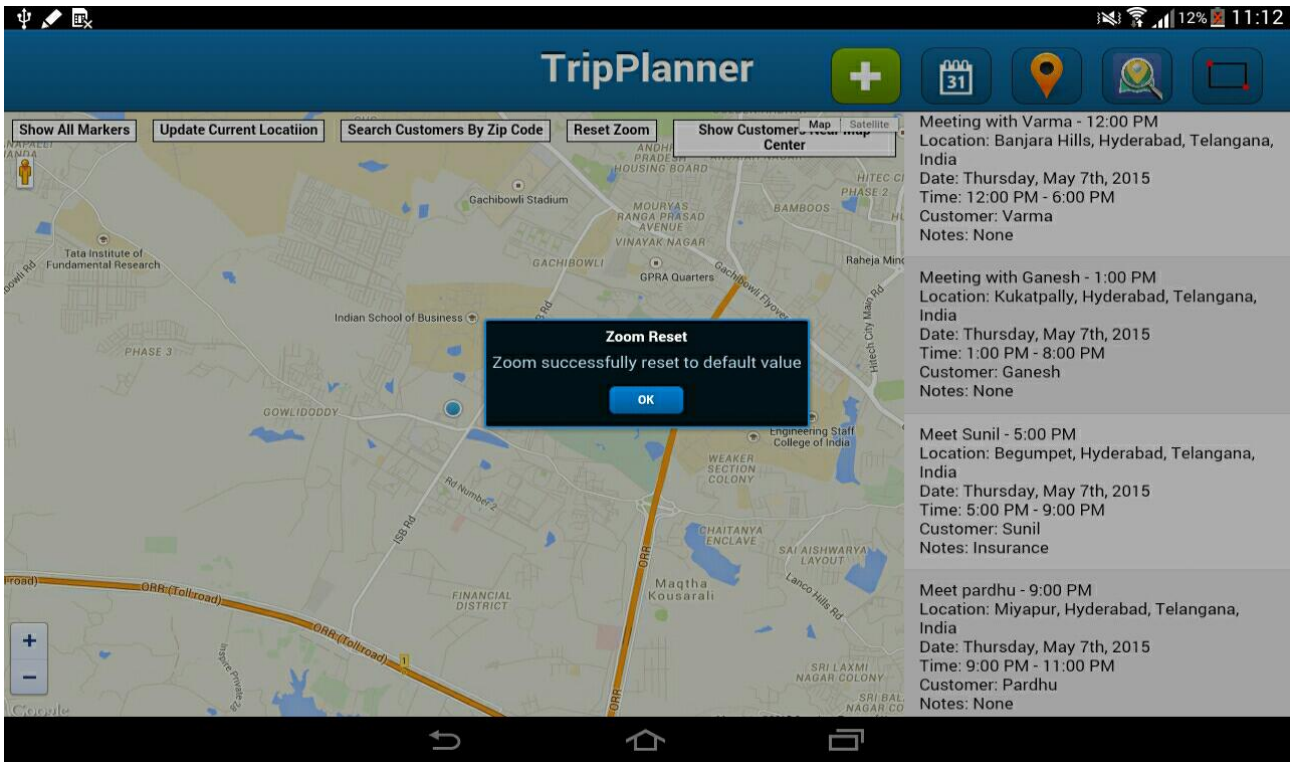

Fig 4.2.3.7 Reset Zoom

4.2.3.8 Search Customer by ZIP code

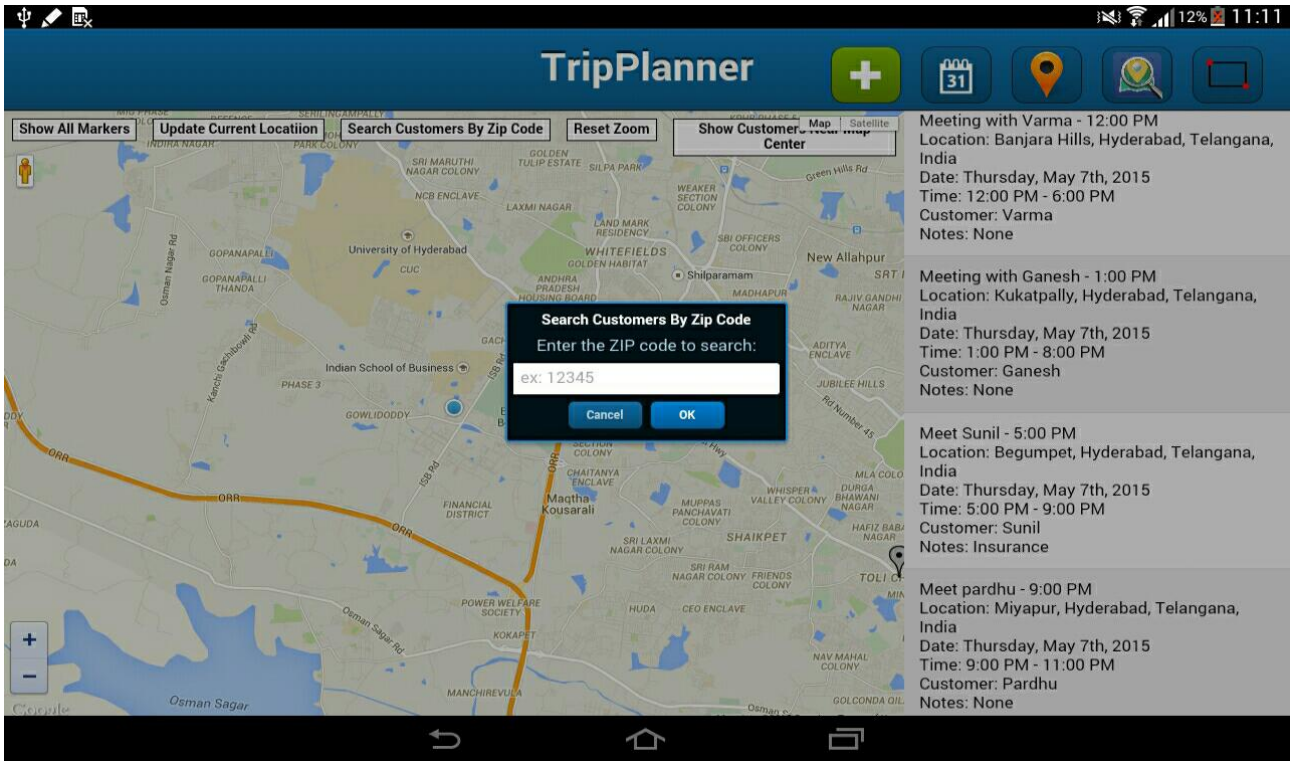

Fig 4.2.3.8.1 Search Customer by ZIP code

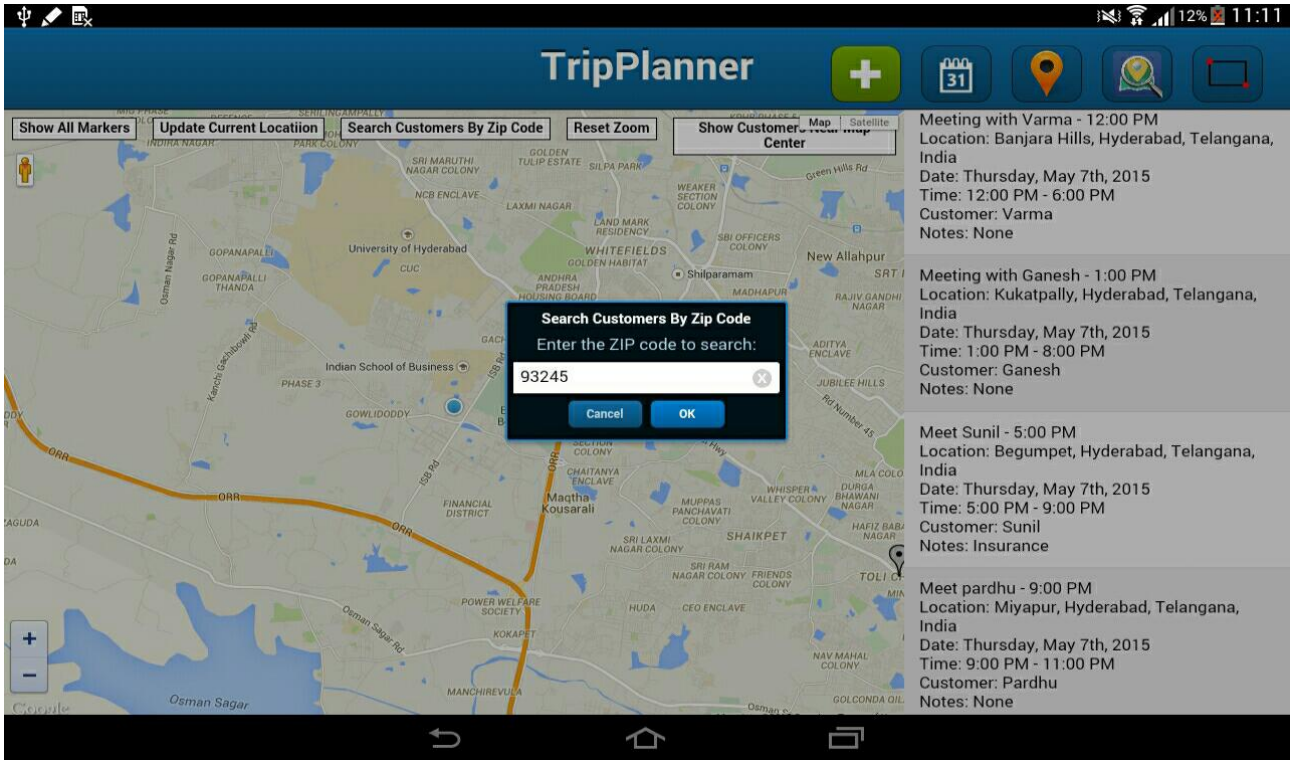

Fig 4.2.3.8.2 Search Customer by ZIP code

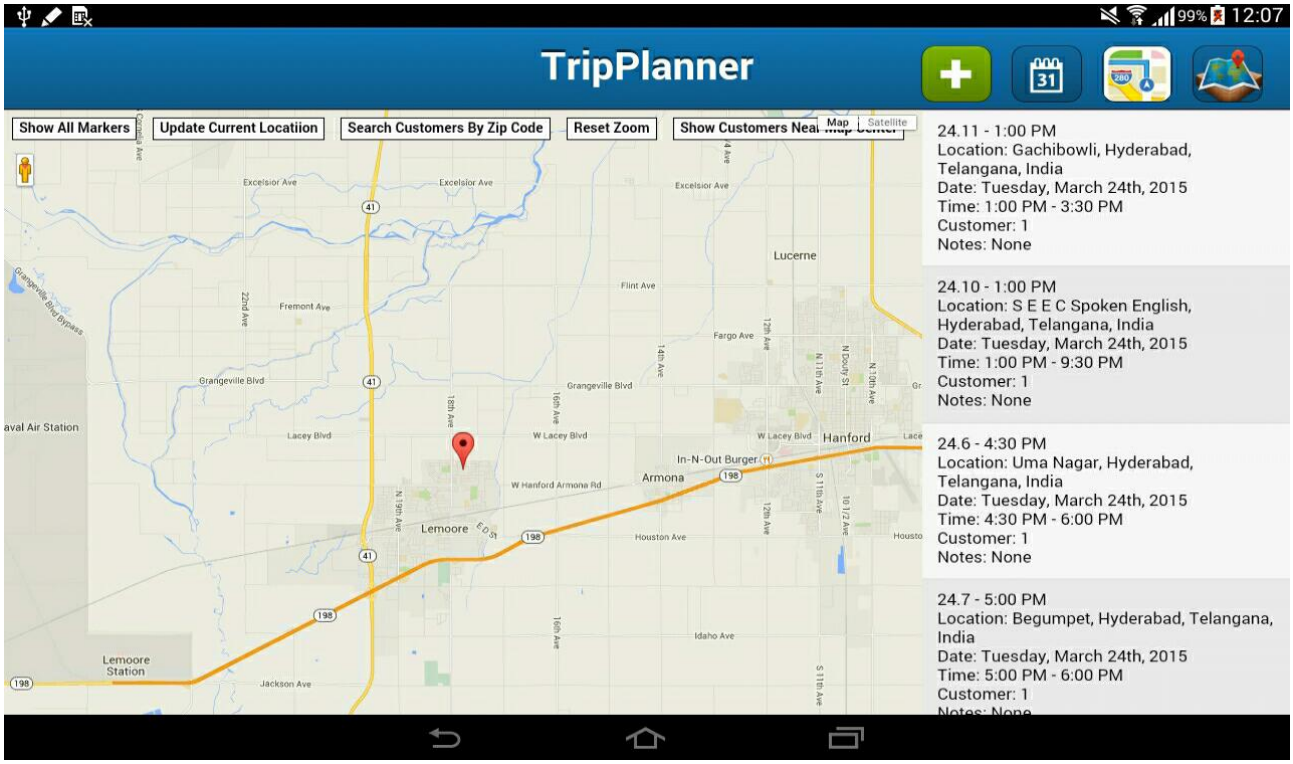

Fig 4.2.3.8.3 Search Customer by ZIP code

4.2.3.9 Display Customers in a selected area

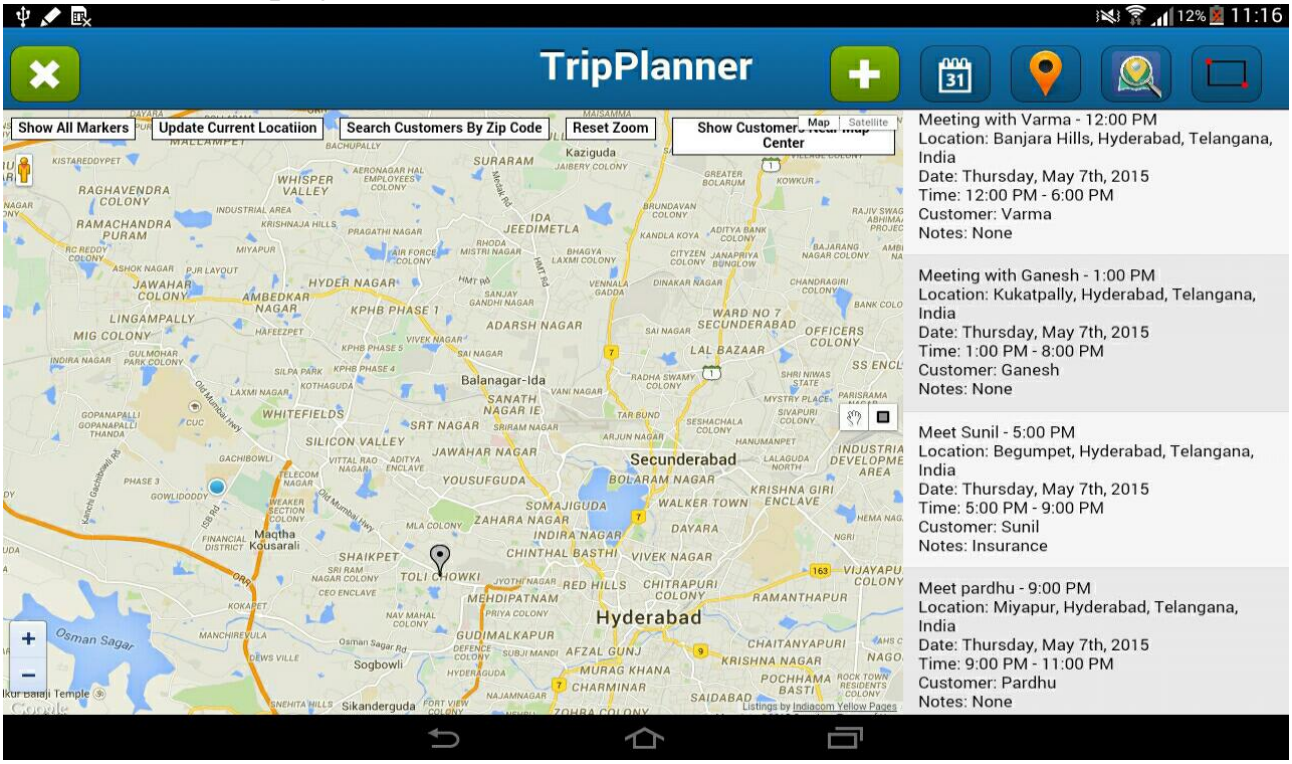

Fig 4.2.3.9.1 Draw Rectangle on Map

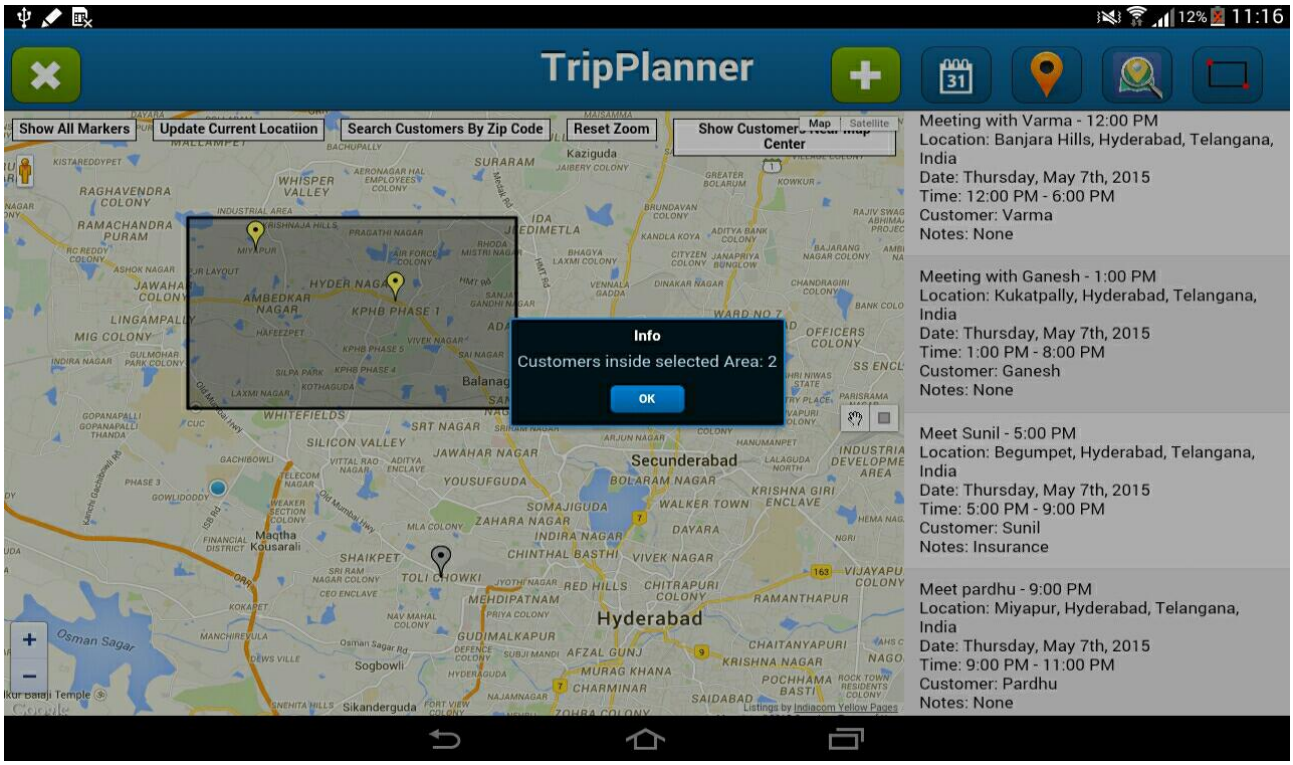

Fig 4.2.3.9.2 Displaying Customers in Selected Location

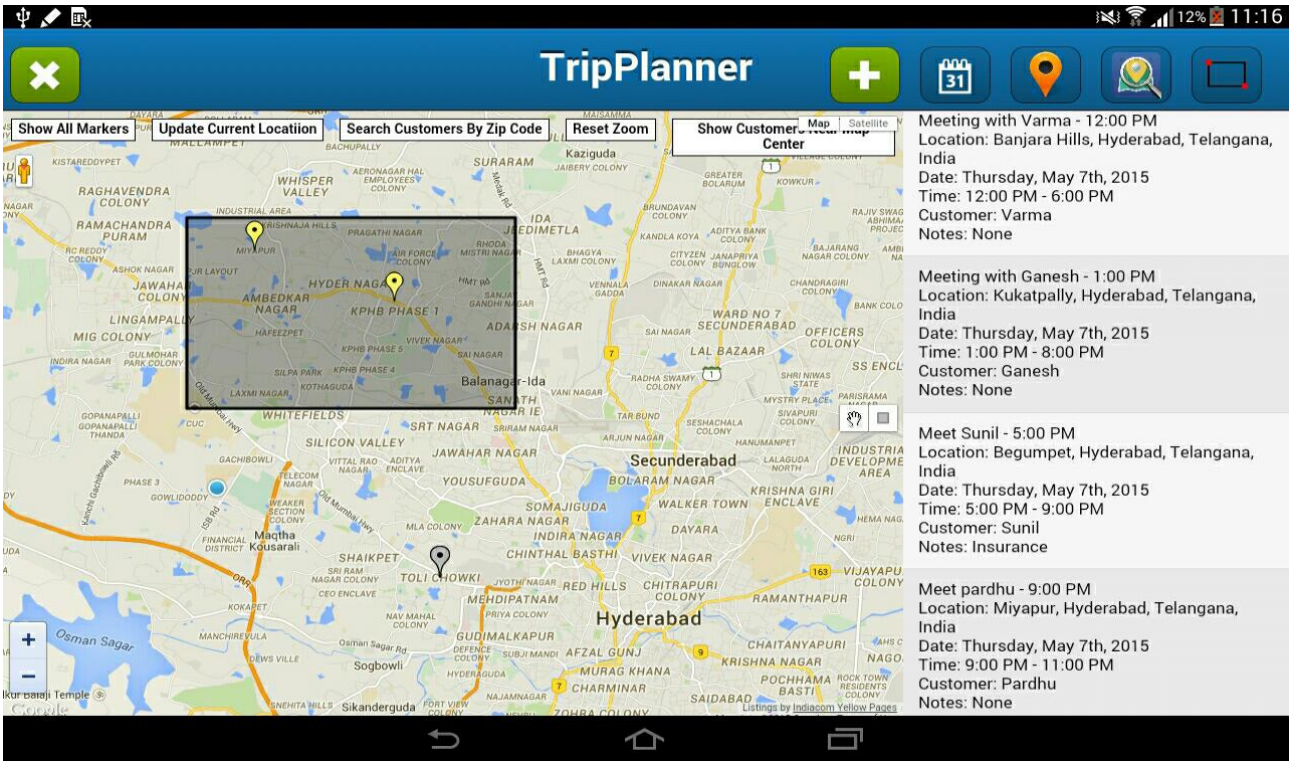

Fig 4.2.3.9.3 Displaying Customers in Selected Location

4.2.4. Calendar View

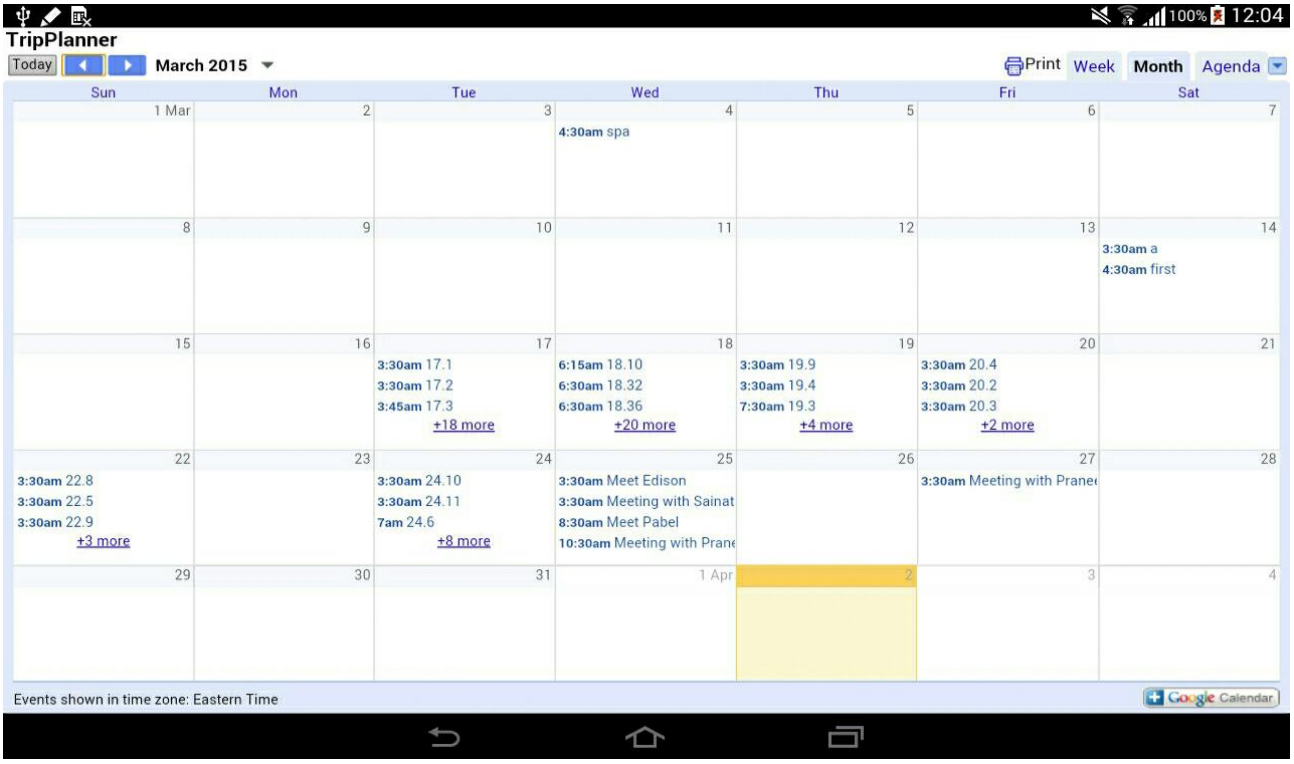

Fig 4.2.4. Calendar View

4.3 SAMPLE CODE

4.3.1 Verification View:-

```
Ext.define('TripPlanner.view.VerificationContainer', {
  extend: 'Ext.Container',

  requires: [
    'Ext.Panel',
    'Ext.Button',
    'Ext.Toolbar'
  ],

  config: {
    id: 'verificationContainer',
    items: [
      {
        xtype : 'titlebar',
        docked: 'top',
        title : 'TripPlanner',
        height: 70,
        style : {
          'font-size': '2em'
        },
        items : [
          {
            xtype: 'button',
            text: 'Back',
            id: 'cancelBtn',
            ui: 'back',
            margin : '12 0 0 10'
          }
        ]
      },
      {
        xtype: 'container',
        layout: 'vbox',
        margin: '40 20% 0 20%',
        pack : 'center',
        items: [
          {
            xtype: 'panel',
            id: 'verifcationPan',
```

```
          style : {
            'font-size' : '1.4em',
            'padding' : '1em'
          }
        },
        {
          xtype: 'button',
          text: 'Verify',
          id: 'verifyBtn',
          style : {
            'font-size' : '2em',
            'margin' : '0 0 0 30%'
            // background: -webkit-linear-gradient(top, #a2e306,#7eb105
3%,#5b7f03);
          },
          ui: ['confirm', 'forward'],
          width: '30%'
        }
      ]
    }
  ]
 }
});
```

4.3.2 Main Container View:

```
Ext.define('TripPlanner.view.MainContainer', {
 extend: 'Ext.Container',

 requires: [
  'Ext.Toolbar',
  'Ext.Button',
  'Ext.Panel',
  'Ext.dataview.List',
  'Ext.Map',
  'TripPlanner.view.Map',
  'TripPlanner.view.TimePickerField',
  'TripPlanner.view.ApptList',
       'TripPlanner.view.ApptListPopUp'
 ],

 config: {
  fullscreen: true,
```

```
    id: 'mainContainer',
    layout: 'vbox',
    items: [
      {
        xtype : 'titlebar',
        docked: 'top',
        title : 'TripPlanner',
        height: 70,
        style : {
          'font-size': '2em'
        },
        items : [
          {
            xtype: 'button',
            id: 'TrafficInfoBtn',
            ui: 'normal',
            align: 'left',
            height: 60,
            width : 70,
            margin : '12 20 0 0',
            padding: '0',
            iconMask: true,
            iconAlign:'center',
                        hidden:true,
                        html:'<img src="resources/icons/Cars2.png"
height="50" width="50" align="middle">',
          }, {
            xtype: 'button',
            id: 'WeatherInfoBtn',
            ui: 'normal',
            align: 'left',
            height: 60,
            width : 70,
            margin : '12 20 0 0',
            padding: '0',
            iconMask: true,
            iconAlign:'center',
                        hidden:true,
                        html:'<img src="resources/icons/weather3.png"
height="50" width="50" align="middle">',
          },{
            xtype: 'button',
            id: 'DeleteRoute',
```

```
            ui: 'confirm',
            align: 'left',
            height: 60,
            width : 70,
            margin : '12 20 0 0',
            padding: '0',
            iconMask: true,
            iconAlign:'center',
                        hidden:true,
                        iconCls:'delete'
                        //html:'<img src="resources/icons/Cars2.png"
height="50" width="50" align="middle">',
          },{
            xtype: 'button',
            id: 'newApptBtn',
            ui: 'confirm',
            itemId: 'newApptBtn',
            align: 'right',
            height: 60,
            width : 70,
            margin : '12 20 0 0',
            padding: '0',
            iconCls: 'add',
            iconMask: true,
            iconAlign:'center'
          },
          {
            xtype: 'button',
            id: 'calendarViewBtn',
            align: 'right',
            height: 60,
            width : 70,
            margin : '12 20 0 0',
            padding: '0',
            ui: 'normal',
            iconCls: 'calendar',
            iconMask: true,
            iconAlign:'center'
          },
                        {
            xtype: 'button',
            id: 'routeBtn',
            align: 'right',
```

```
          height: 60,
          width : 70,
                        html:'<img src="resources/icons/navigation.png"
height="50" width="50" align="middle">',
          margin : '12 20 0 0',
          padding: '0',
          ui: 'normal',
          iconMask: true,
          iconAlign:'center'
         },
                  {
          xtype: 'button',
          id: 'nearbyBtn',
          align: 'right',
          height: 60,
          width : 70,
          margin : '12 20 0 0',
                        html:'<img src="resources/icons/search-nearby.png"
height="50" width="50" align="middle">',
                        padding: '0',
                        ui: 'normal',
          iconMask: true,
          iconAlign:'center'
         }  ,
                  {
          xtype: 'button',
          id: 'drawRectBtn',
          align: 'right',
          height: 60,
          width : 70,
          margin : '12 20 0 0',
                        html:'<img src="resources/icons/drawrect.png"
height="50" width="50" align="middle">',
                        padding: '0',
                        ui: 'normal',
          iconMask: true,
          iconAlign:'center'
         }

       ]
     },
     {
       xtype: 'container',
```

```
        layout: 'hbox',
        items: [
          {
            xtype: 'panel',
            html: '<h1 id="auth-msg"> You must authorize with your Google account by clicking the button to the right in order to use TripPlanner </h1>',
            id: 'authRequiredMsg',
            flex: 3
          },
          {
            xtype: 'button',
            id: 'authBtn',
            text: 'Authorize',
            ui : 'decline',
            style : {
              'font-size' : '2em'
            },
            flex : 2
          }
        ]
      },
      {
        xtype: 'container',
        layout: 'hbox',
        items: [
          {
            xtype: 'tripmap',
            flex: 7
                        //width:'60%'
          },
          {
            xtype: 'panel',
            id: 'apptListPanel',
            flex: 3,
                        //width:'40%',
            layout: 'fit',
                        emptyText: 'No Appointmetns found for the day!! #1',
            items: [
              {
                xtype: 'widget.apptList',
                id: 'apptList',
                                    emptyText: 'No Appointmetns found for the
```

```
day!! #2'
            }
          ]
        }
      ]
    },
        {
                    xtype:'panel',
                    id:'popup',
                    hidden:true,
        //floating: true,
        centered: true,
        modal: true,
        width: 350,
        height: 500,
                    ui:'confirm',
                    layout:'fit',
        styleHtmlContent: true,
                    hideOnMaskTap:true,
          items: [{
            xtype: 'widget.apptListPopUp',
                                id: 'apptListPopUp',
          }
          ]

      },
            {
                    xtype:'panel',
                    id:'popup2',
                    hidden:true,
        //floating: true,
        centered: true,
        modal: true,
         width: 350,
        height: 500,
                    ui:'confirm',
                    layout:'fit',
        styleHtmlContent: true,
                    hideOnMaskTap:true,
          items: [{
            xtype: 'list',
                                data:[
                                {text:' Search Nearby
```

```
Restaurants',id:'restaurant'},
                                {text:' Search Nearby ATMs',id:'atm'},
                                {text:' Search Nearby Parking
Garages',id:'parking'},
                                {text:' Search Nearby Banks',id:'bank'},
                                {text:' Search Nearby
Hospitals',id:'hospital'}

                                ],
                                 id: 'PlacesListPopUp',
                                 itemTpl:'{text}'
            }
            ]

      }
   ] // end items
  } // end config

});
```

4.3.3 Calendar Container View:

```
Ext.define('TripPlanner.view.CalendarContainer', {
  extend: 'Ext.Container',

  requires: [
    'Ext.Panel',
    'Ext.Button'
  ],

  config: {
    id: 'calendarContainer',
    items: [
      {
        xtype: 'panel',
        id: 'calendarPanel'
      },
      {
        xtype: 'backBtn',
        text: 'Back'
      }
    ]
  }
```

```
});
```

4.3.4 Appointment View:

```
Ext.define('TripPlanner.view.ApptFormContainer', {
  extend: 'Ext.Panel',

  requires: [
    'Ext.form.Panel',
    'Ext.form.FieldSet',
    'Ext.field.DatePicker',
    'Ext.picker.Date',
    'Ext.Button',
    'Ext.MessageBox',
    'TripPlanner.view.BackBtn',
          'Ext.dataview.List',
          'Ext.field.Search' ,
          'TripPlanner.view.SearchListDisplay'
  ],

  config: {
    layout: 'fit',
    items: [
      {
                    region:'north',
        xtype : 'titlebar',
        id    : 'apptFormTitleBar',
        docked: 'top',
        title : 'New Appointment',
        height: 70,
        style : {
          'font-size': '2em'
        },
        items : [
          {
            xtype: 'backBtn',
            text: 'Back',
            ui: 'back',
            margin : '12 0 0 10'
          }
        ]
      },
```

```
{
            margin : '12 0 0 10',
     width:'65%',
     height:'100%',
            region: 'west',
  xtype: 'formpanel',
  id: 'apptFormPanel',
  items: [
    {
     xtype: 'fieldset',
     id: 'apptFormFields',
     items: [
                        {
       xtype: 'textfield',
       id: 'apptFormTitleField',
       label: 'Title',
       name: 'title',
       required: true
      },{
       xtype: 'textfield',
       id: 'CustomerNameField',
       label: 'Customer Name',
       name: 'customer'
      },
      {
       xtype: 'textfield',
       id: 'apptFormLocationField',
       label: 'Location',
       name: 'location',
       required: true
      },
      {
       xtype: 'datepickerfield',
       id: 'apptFormDatePicker',
       label: 'Date',
       name: 'date',
       required: true,
       picker : {
        yearFrom : new Date().getFullYear(),
        yearTo   : new Date().getFullYear() + 2
       }
      },
      {
```

```
            xtype: 'timepickerfield',
            id: 'apptFormStartTimePicker',
            label: 'Start',
            name: 'start',
            required: true
          },
          {
            xtype: 'timepickerfield',
            id: 'apptFormEndTimePicker',
            label: 'End',
            name: 'end',
            required: true
          },
          {
            xtype: 'textfield',
            id: 'customerNoteField',
            label: 'Notes',
            name: 'note',
            // Scroll so keyboard on Android doesn't block input field
            listeners: {
                focus: function(comp, e, eopts) {
                    var ost = comp.element.dom.offsetTop / 2;

this.getParent().getParent().getScrollable().getScroller().scrollTo(0, ost);
                }
            }
          }
        ]
      },
      {
        xtype: 'button',
        id: 'apptFormSubmitBtn',
        ui : 'confirm',
        width: '50%',
        margin:'0 25% 0 25%',
        style : {
          'font-size' : '2em'
        },
        text: 'Submit'
      },
      {
        xtype: 'button',
        id: 'apptFormDeleteBtn',
```

```
          text: 'Delete',
          ui: 'decline',
          width: '50%',
          margin:'10 25% 0 25%',
          style : {
            'font-size' : '2em'
          }
        }
      ]
    },
        {
                margin : '12 0 0 65%',
      xtype: 'panel',
                layout:'vbox',
      items: [
        {
          xtype: 'fieldset',
                        layout:'hbox',
          items: [
            {
              xtype: 'searchfield',
              id: 'SearchCustomerField',
              label: 'Search',
                                iconCls:'search',
                                autoComplete:true,
                                clearIcon:true,
                                placeHolder:'Customer Name',
                                width:'80%',
                                listeners :{
                                        change : function( me, newValue,
oldValue, eOpts )
                                        {
                                        var a=me.getValue();
                                        if(a>0)
                                        {

        Ext.getCmp('listid').setHidden(false);
                                        }
                                        else
                                        {

        Ext.getCmp('listid').setHidden(true);
                                        }
```

```
                                    }

                                }
            },
                        {
             xtype: 'button',
             id: 'SearchCustomerButton',
             label: 'Search',
             name: 'Search',
                                iconCls:'search',
                                ui: 'confirm',
                                iconMask: true,
                                width:'20%'
            }

                        ]
                        },
                        {
                                margin : '-6% 0',
             xtype: 'widget.SearchListDisplay',
             id: 'SearchList',
                                layout: 'fit',
            }
                        ]
                        }
    ]
  }
});
```

5.TESTING

5.1 TESTING OVERVIEW:

- Testing is a process of executing a program with intent of finding an error.
- Testing presents an interesting anomaly for the software engineering.
- The goal of the software testing is to convince system developer and the customers that the software is good enough for operational use. Testing is a process intended to build confidence in the software.
- Testing is a set of activities that can be planned in advance and conducted systematically.
- Software testing is often referred to as verification & validation.

5.2 DIFFERENT TYPES OF TESTING

Types of testing:

- Unit Testing
- Integration Testing
- Functional Testing
- System Testing

5.2.1 Functional Test:

Functional tests provides systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation and user manuals.

Functional testing is centred on the following items:

Valid Input : Identified classes of valid input must be accepted.
Invalid Input : Identified classes of invalid input must be rejected.
Functions : Identified functions must be exercised.
Output : Identified classes of application outputs must be exercised
Systems/Procedures: Interfacing systems or procedures must me invoked.

Organizations and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify business process lows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

5.2.2 System Testing

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flow, emphasizing pre-driven process links and integration points. System testing falls within the scope of black box testing and as such, should require no knowledge of the inner design of the code or logic.

5.2.3 White Box Testing

White-box testing is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality. In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. This is analogous to testing nodes in a circuit.

5.2.4 Black Box Testing

Black box testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a defective source document, such as specification or requirements document. It is a testing in which the software under testing is treated as a black box. We cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works. It’s also called as behavioural tests as the focus is on functional requirements of the software.

5.2.5 Acceptance Testing

Acceptance testing is a test conducted to determine if the requirements of specification or contract are met. Software developers often distinguish acceptance testing by the system provider from acceptance testing by the customer (the user or client) prior to accepting transfer of ownership. In the case of software, acceptance testing performed by the customer is known as user acceptance testing.

5.4 Test Report

5.4.1) Authorization Module:

Table 5.4.1 Test Cases for Authorization Module

Test Case Id	Test Case Name	Description	Expected Result	Actual Result	Status
1	Authorize	Correct usename and correct password	Authorizes with Google API key	Authorizing with Google API key	Pass
2	Authorize	Correct usename and wrong password	Displays error message	Displaying error message	Pass
3	Authorize	Wrong usename and correct password	Displays error message	Displaying error message	Pass
4	Authorize	Wrong usename and wrong password	Displays error message	Displaying error message	Pass
5	After Authorization	Display of Appointment, Calendar Buttons	Displays Appointment, Calendar Buttons	Displaying Appointment, Calendar Buttons	Pass
6	Before Authorization	Display of Appointment, Calendar Buttons	Should not Display Appointment, Calendar Buttons	Not Displaying Appointment, Calendar Buttons	Pass

5.4.2) Appointment List Module:

Table 5.4.2 Test Cases for Appointments Module

Test Case Id	Test Case Name	Description	Expected Result	Actual Result	Result
1	Create New Appointment Button	Displays Appointment form on clicking	Display Appointment Form	Displaying Appointment Form	Pass
2	Appointment Form – Location Field	Displays Locations on entering name- Auto-complete	Shows list of locations	Showing list of locations	Pass
3	Appointment Form – Date Field	User selects Date from Date picker	Date populates in date field	Date is populating in date field	Pass
4	Appointment Form – Start-time Field	User selects Time from Time picker – No manual Entry	Time populates in Time field- No manual Entry	Time populating in Time field	Pass
5	Appointment Form – End-time Field	User selects Time from Time picker- No manual Entry	Time populates in Time field- No manual Entry	Time populating in Time field	Pass
6	Display Appointments	After authorization – Present day appointments are displayed along with markers	All appointments should be displayed with markers	All appointments are visible on map along with markers	Pass

7	Display Appointments	On single tap on a particular appointment it should be focused on the center of map	Appointment should be focused on center of map	Appointment is getting focused on the center of map	Pass
8	Display Appointments	On double tap of particular appointment screen pops up allowing to update or delete an appointment	On double tap Screen should navigate to create new appointment form with the particular appointment already pre filled and display Delete and Update Buttons	On double tap fields are pre filled and Update and Delete buttons are getting displayed	Pass
9	Update Appointment Button	On clicking any changes to appointments that are made are saved and stored in calendar and updated appointment list is displayed	Changes should be stored and updated appointment list is displayed	Changes are stored and updated appointment list is being displayed	Pass

10	Delete Appointment Button	On clicking the respective appointment should be deleted from calendar and store and updated appointment list is displayed	On clicking appointment should be deleted from store and calendar and updated appointment list is displayed	Appointment is getting deleted from store and calendar and updated appointment list is displayed	Pass
11	Search Customer Button	On entering a name and clicking the button results are displayed based on keyword entered	Valid results should be displayed based on keyword entered	Valid results are getting displayed based on keyword entered	Pass
12	Search Customer Button	On selection of results the name of existing customer and his location should be automatically filled in appointment form	Fields should be automatically filled on selection of a particular customer	Fields are getting filled on selecting a customer	Pass

5.4.3) Map View Module:

Table 5.2.3 Test Cases for Map View Module

Test Case Id	Test Case Name	Description	Expected Result	Actual Result	Result
1	Show All Markers	Set the zoom of the map panel in such a way that all the markers on map should be visible	Should display all the markers on map	Displaying all the markers on map when tapped	Pass
2	Show Route	Draw route from first appointment to last appointment connecting the middle appointments as waypoints	Should display route from first to last appointment connecting the middle appointments	Displaying route from first to last appointment connecting the middle appointments	Pass
3	Show Near By	Get the appointments location and Search near by type from agents and display the selected	Should get the appointments location and Search near by type from agents and display the selected with markers	Taking the appointments location and Search near by type from agents and displaying the selected with markers	Pass
4	Update Current Location	display current agent location	Should Display current agents location when tapped	Displaying current agents location when tapped	Pass

5	Reset Zoom	Resets the zoom to default	Should Resets the zoom to default	When Tapped,Resets the zoom to default	Pass
6	Appointment Information	Displays appointments Details in a infowindow when tapped on the marker	Should Display appointments Details in a infowindow when tapped on the marker	Displaying appointments Details in a infowindow when tapped on the marker	Pass
7	Search Customer By ZIP	Gets ZIP code from agent and displays marker of the result	Should take ZIP from agent and search ,display the result with marker	Taking ZIP from agent and displaying the result with marker	Pass
8	Search Customer near map center	Get ZIP code from agent current locations and display near by customers with markers	Should get ZIP code from agent current locations and display near by customers with markers	Taking ZIP code from agent current locations and display near by customers with markers	Pass
9	Draw Markers for Current day	Displays red color marker for current days appointments on map	Should display red color marker for current days appointment	Is displaying red markers for current days appointments	Pass
10	Draw Markers for other days except current day	Displays Gray color marker for other days appointments on map	Should display Gray color marker for Other days appointment	Is displaying Grey markers for Other days appointments	Pass

5.4.4) Calendar View Module:

Test Case Id	Test Case Name	Description	Expected Result	Actual Result	Result
1	Calendar Button	On clicking the button Trip planner calendar will be displayed showing all the appointments	Should show all the appointments	Displaying all the appointments	Pass

Table 5.2.4 Calendar View Module

6. CONCLUSION AND FUTURE WORK

In this project, I have been assigned to develop a Cross Platform Application that is used by insurance agents which should simplify the work for agents to manage their client's information and appointments. We have successfully implemented the system which contains various functionalities that integrates Google Calendar, Google Maps.

In Future, We would like to do the following

- **UI Improvement**
  - Add a help button to inform users of functionality and available actions.
  - Improve appointment list look and feel
  - Appointment form
    - Improve look / feel
    - Improve validation messages (currently appear in an alert)
- **Google Calendar Events**
  - Updating events. Updates are done by deleting and then *recreating* the event instance in Google Calendar. This is because the property 'sequence' has to be properly updated when updating Calendar events. After updating or creating an event, Google Calendar will send a response object containing the next sequence value for updating the object. For some reason, this was not working when we tried to implement it. Thus our workaround was to simply delete and recreate the event in Google Calendar.

Known Bugs:

- Time picker field causes overflow error when it is opened (does not break app)
- Appointment list items have a fixed size
- Notes field in appointment form may be covered by Android keyboard. A workaround has been provided, to scroll the form, however it is not optimal.
- When deleting an appointment, a javascript error “Cannot call removeCls of null” appears.
- App does not update based on current time. For example, some markers may need to be switched from gray to red if an agent has the app open for awhile
- App does not work offline.
- Doing too much work at once may cause app to close / error
- Map needs to be dynamically sized for landscape / portrait and different devices.
- Splash screen and application icon need to updated

7. REFERENCES

[1] Implementation of Location based Services in Android using GPS and Web Services Manav Singhal1, Anupam Shukla2,ABV-Indian Institute of Information Technology and Management, Gwalior, India, IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 1, No 2, January 2012 .

[2]Introduction to iPad Application Development with PhoneGap,Guangyuan Piao and Wooju Kim, Member, IACSIT ,International Journal of Innovation, Management and Technology, Vol. 4, No. 1, February 2013.

[3]Performance Evaluation of RESTful Web Services for Mobile Devices Hatem Hamad, Motaz Saad, and Ramzi Abed ,Computer Engineering Department, Islamic University of Gaza, Palestine,International Arab Journal of e-Technology, Vol. 1, No. 3, January 2010 .

[4] http://phonegap.com/blog/2013/11/20/SenchaPhoneGap/

[5]http://docs.sencha.com/touch/2.4/

[6]http://docs-origin.sencha.com/touch/2.4/2.4.0-apidocs/

[7]https://developers.google.com/maps/documentation/javascript/

[8]https://developers.google.com/google-apps/calendar/

[9]https://developers.google.com/places/webservice/search

[10]https://developers.google.com/maps/documentation/javascript/examples/marker-simple

[11]https://developers.google.com/maps/documentation/javascript/examples/drawing-tools

[12]https://developers.google.com/maps/documentation/javascript/directions

[13]https://developers.google.com/maps/documentation/javascript/examples/place-search

[14]https://developers.google.com/maps/documentation/javascript/examples/layer-traffic

[15]https://developers.google.com/maps/documentation/javascript/examples/layer-weather

[16]https://developers.google.com/maps/documentation/javascript/examples/map-geolocation