MY TRIP PAL: AN ANDROID APPLICATION FOR TRACKING TRAVEL

By

SWAPNA BOJANKI

B.E, Andhra University, India, 2011

A REPORT

Submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Computing and Information Sciences

College of Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

2014

Approved By:

Major Professor

Mitchell L. Neilsen

Copyright © 2014 by Swapna Bojanki

All rights reserved.

No part of this Report may be replicated in any form or by any electronic or mechanical means. The only exception is that it can be replicated by a reviewer, who may quote short excerpts in a review. This Report is a description of work of the Project MyTripPal and its implementation.

Abstract

Smartphones have become an integral part of everyday life and a study conducted by Kantar World panel ComTech (11/2012 – 02/2013) showed sales of all Android phones outpaced the iPhone by a hefty margin: 52.1 percent to 43.5 percent. Moreover, Android is the OS for most of the mobiles like HTC, Google, Samsung, Sony, Motorola etc.

As Steve Jobs said "It's really hard to design products by focus groups. A lot of times, people don't know what they want until you show it to them." There are a lot of common people out there who travel often and book tickets on various websites; i.e., a person like me travels very often and books ticket on a website which provides for the cheapest price. Most of the time the destinations might match and they want to keep track of the websites the tickets are booked, track the dates travelled to a place, track the price and various other details as it gets difficult to remember all details. Moreover, it is hectic to browse through each and every website and check the previous travel. Similarly, if one travels on quite a number of trips via car, one might want to keep track of those details, too. Hence to track all of these, it would be very convenient if there was an application which can take in all details of personal trips so that you can refer back to them whenever you want to do so.

MyTripPal is an Android application where a user can save past trips and future trips via flight or car and a user can enter in all details and save trips. For a future trip, users can be reminded via alarm on a date and time he chooses while entering the trip details in the application. Autofill option is given while entering the source and destination where these are prompted from the values that are entered earlier in previous trips. Google maps navigation option is given which provides the route from the source to the destination. The user can also search for the trips and there is an Upcoming Trips tab which lists all future trips via car or flight.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to express my sincere gratitude to my major professor, Dr.Mitch Neilsen, who has provided me with constant guidance and encouragement throughout my project development. He has always been a source of valuable suggestions and corrections which helped me gain immense knowledge.I would like to thank my committee members, Dr.John Hatcliff and Dr.Torben Amtoft, for serving on my committee and also I feel honored to have taken courses under you.

It was a pleasure taking courses under all the other Professors of the CIS Department. They provided me a lot of real world experience and knowledge. I also thank all the staff of CIS Department for being such warm people and helping me whenever needed.Special thanks to Dr.Gurdip Singh for having faith in me as a Graduate Teaching Assistant which helped me gain a lot of experience.

I thank all my friends who provided me constant support and Pranay Roy Matcha for being with me through thick and thin and for being a constant source of encouragement and motivation.

Finally, I would like to thank my parents Mr.Rambabu Bojanki and Mrs.Aruna Bojanki for everything they have done for me and for giving their unconditional love and support, ‘You made me who I am today’.

Chapter 1 - Introduction

MyTripPal is an Android application which targets any common user with an Android smartphone who travels via flight or car. This application helps the user track travel including both previous and future trips. The application mainly consists of three modules; Trips via Flight, Trips via Car, and Upcoming Trips. For ‘Trips via Flight’, as a user he can add/modify/delete a trip. The add or edit feature involves various fields needed by user like the Source, Destination, Trip date and time which in turn disables the alarm option; i.e., if it is a previous date the user won’t be able to add an alarm option, Airlines, website booked and flight information, cost incurred for the trip, people travelled with and alarm option where you can specify exact and date and time to be reminded. For ‘Trips via Car’, a user is able to add/edit the trip source, destination and time details. Users can also add the car rental information like the company car is booked from and total number of people travelling and alarm option. Users can also use the navigation button present on the trip information screen to get google map navigation for the trip. For each of these he can track and search previous and future trips. The ‘Upcoming Trips’ module displays a list of all future trips that are lined up via flight or car.

Chapter 2 - Motivation

The main motivation behind the project idea is my high frequency of travelling. I am a person who likes to travel a lot. Let it be travel to visit relatives located in different parts of USA or India, or travel involving road trips with friends on a car. While booking my air travel, I browse through all available websites and book the ticket that is the cheapest. Hence, whenever I want to book a ticket for the same destination or so, I go through the hassle of searching previous data in all sites. It would be very convenient if I could store all my travel information in one place so that I can store my previous travel history and at the same time other details so that it would be handy to check them whenever required. At the same time I can have an alarm on my mobile device before some time or days to notify me that I have a trip soon. The same for road trips, I can track all my trips and it is handy to have the navigation from a past or future trip. Hence, in creating a solution to all of those problems, plus some extra features emerges the application ‘MyTripPal’.

Chapter 3 – Background: Android Design and Architecture

3.1 Background

Android Operating System (AOS) is a Linux-based Operating System which can be used on smartphones and tablets. AOS is initially developed by Android Inc. and is later acquired by Google. It is an open source development platform powered by a modified Linux 2.6 Kernel [1, pp.1].

3.2 Android Architecture

The layered Android Architecture can be seen in Figure 1 where the modified Linux 2.6 Kernel acts as the Hardware Abstraction Layer (HAL) thus providing the memory management, device drivers, process management and networking functionalities.

The libraries layer is interfaced by Java and contains the Android Specific Bionic libc which is a lightweight, embedded version of Android's own C library developed by the Android Community. The Surface manager form the Libraries layer handles the User Interface windows.

Next is the Android Runtime Layer which contains the Core Libraries and the Dalvik Virtual Machine (DVM). Android systems use the DVM which uses a special form of byte-code due to which java byte-code cannot be executed on Android Systems but the java class files can be converted to Dalvik executables (dex) using dx tool.

The next layer is the Android Application Framework which is responsible for the Application Life cycle. The Content Provider is responsible for accessing the data from other applications and also to share an applications' own data. The Resource Manager helps accessing non code resources like graphics whereas the Notification Manager helps displaying custom alerts.

The top layer contains the Applications like Calculator, Clock, and Calendar etc. [3, pp.2].

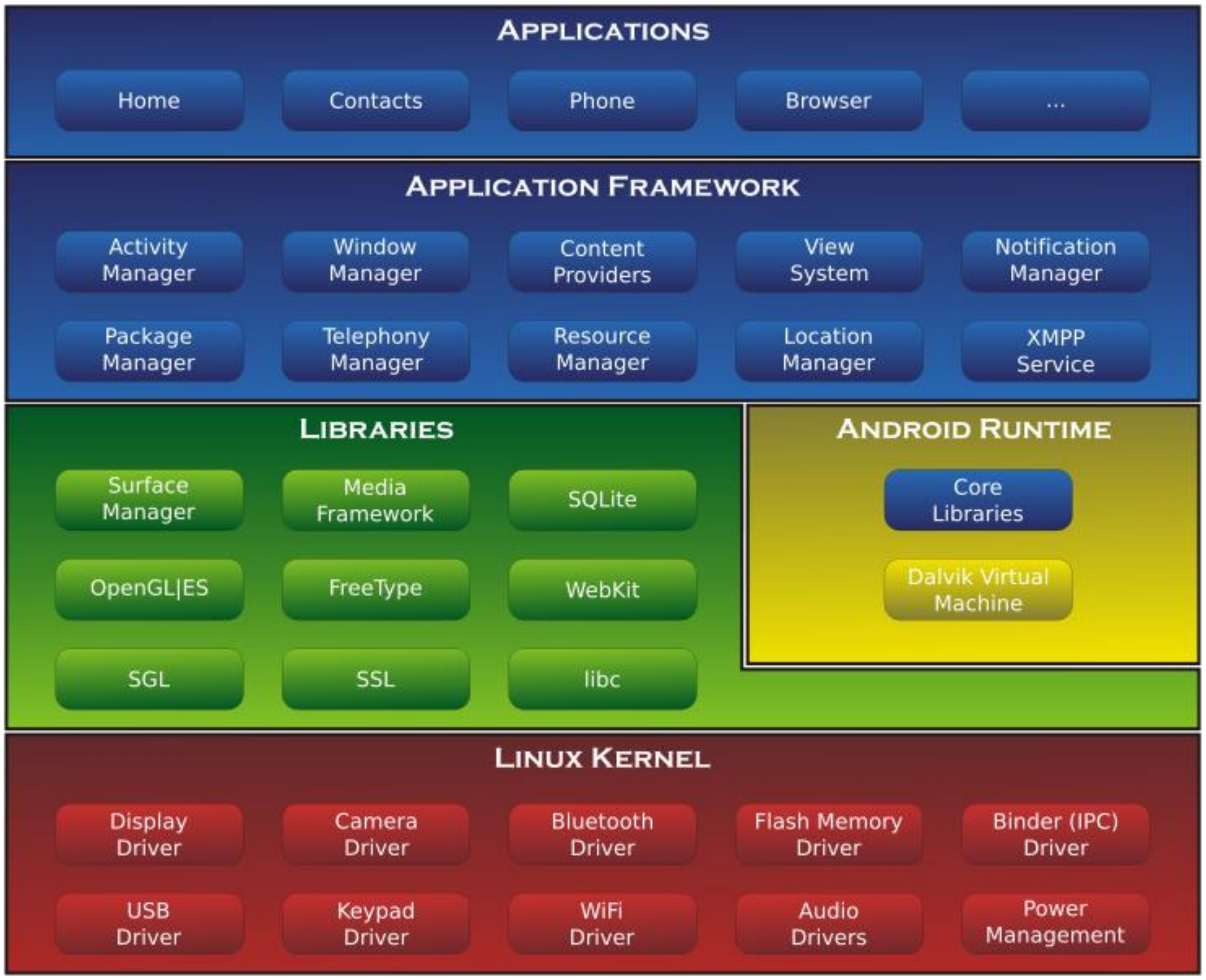


Figure 3.1 Android Layered Architecture, [1, Fig.1]

3.3 Target Platform

As mentioned earlier, Android development platform powered by a modified Linux 2.6 Kernel and most of the systems based on it are X86 based systems. But the mobile devices are ARM based which is a 32-bit Reduced Instruction Set Computer Architecture (RISC). On the other hand, X86 is mainly base on Complicated Instruction Set Architecture (CISC). As the name suggests, CISC instructions are more complicated than RISC. This gives rise to issue with size, cost and power and to resolve this, ARM provides a second 16-bit instruction set which is labelled as Thumb which can be interleaved with the actual 32-bit ARM instructions. The main advantage of ARM design is it focuses on lower power consumption. The major chipsets deployed in Android devices these days are Qualcomm Snapdragon

(HTC), Texas Instruments OMAP (Motorola) and Hummingbird (Samsung), all three are based on the ARM Cortex-A8 Architecture.

3.4 Kernel and Startup Process

Though Android is based on Linux 2.6, it doesn't have the standard Linux Kernel. It has other improvements like alarm driver, a shared memory driver, power management feature, kernel logger and debugger and a binder for inter-process communication. On power, Boot ROM code starts executing and loads the BootLoader into RAM and starts the execution where the BootLoader is a program which runs before the Android OS. During the android boot process as shown in Figure2, the init process is called the Android Linux Kernel which in turn accesses the init.rc and init.device.rc files.

In Java, a separate Virtual Machine instance will pop up in memory for each application. But since we want the Android application to launch in the least possible time, launching different instance of the Dalvik Virtual Machine for every application consumes lot of memory and time. Hence, this problem is resolved by Android OS by a system named "Zygote" which enables sharing code across Dalvik Virtual Machines, and lowering the memory footprint. Thus, the overall startup process is expedited. The System Servers are then launched which contains the system services like starting the Power manager, Alarm manager, Battery service, Activity manager, etc. Once all services are launched, the Startup process is completed [2].

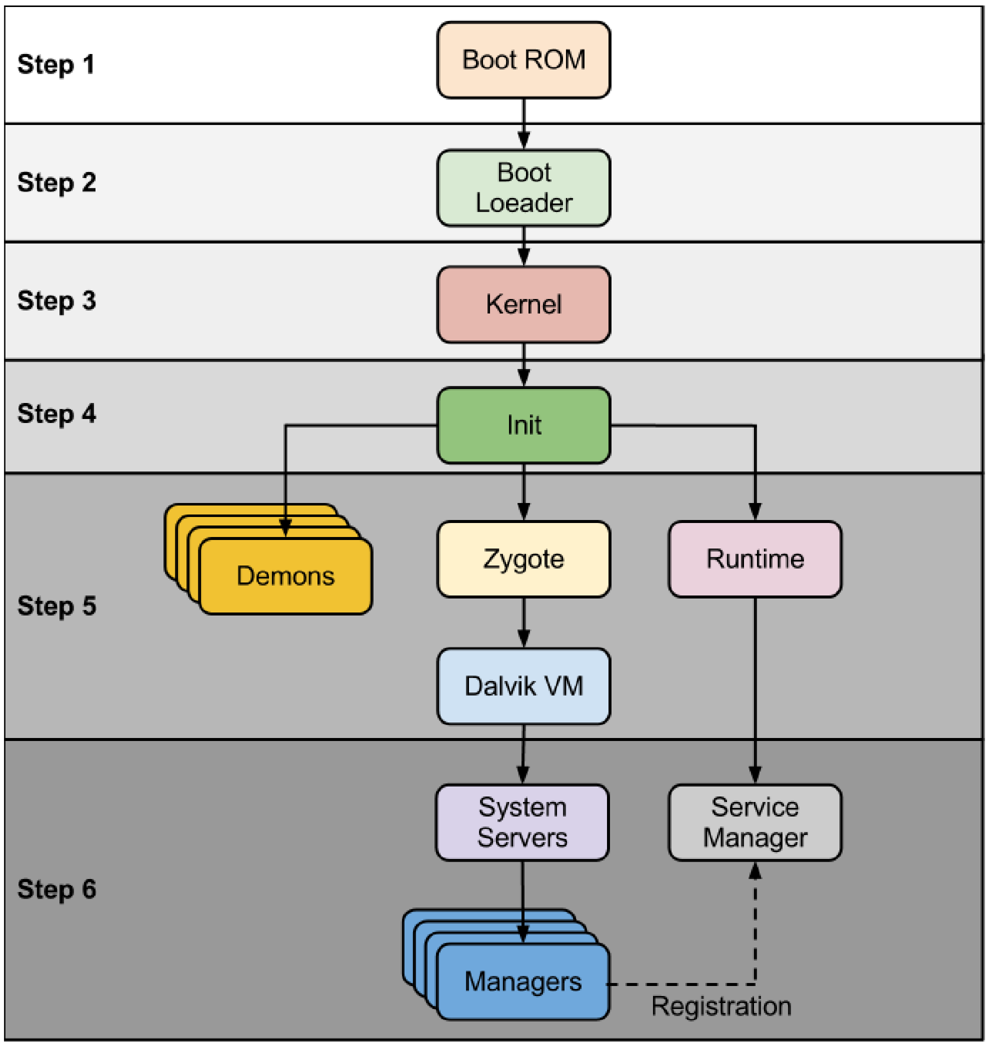


Figure 3.2 Android Boot Sequence [2, Fig.1]

3.5 Data Storage

Android provides various options to save persistent data.

- Shared Preferences - Stores private primitive data in the form of Key-Value pairs.
- Internal Storage - Stores private data directly on the device.
- External Storage - Stores public data on a shared external storage.
- SQLite Database – Stores structured data on a private database.

- Network Connection – Stores data on the web with our own network server.

This application uses SQLite database to store all of its data.

3.6 Power Management

Android based systems have their own power management infrastructure called Power Manager. The applications and the services request CPU resources through wake locks via the Android application framework and the native Linux libraries. Thus a processor will not consume power if the applications or a service does not require any power. If there are no active wake locks, Android will simply shutdown the processor [3].

3.7 Related Work

There are mobile applications for almost all of the travel or airline websites like apps for www.orbitz.com, www.southwest.com, www.priceline.com etc. But there isn't one which holds all the data in one place and moreover, most of the mobile apps of these travel websites do not have the option to view or search through all reservations. There aren't many applications to store our travel via car, too. Hence, users can make good use of an application like MyTripPal to track their travel information.

Chapter 4 - Requirements Analysis

In the requirements analysis we need to elicit the requirements by performing the requirements gathering by analyzing the stakeholders and obtain a clear set on unambiguous requirements and record them.

4.1 Requirements Gathering

As this is an application for a common user, the graphical design that is the front end is a main aspect and it should be taken care that it will be easy to use for any user and navigation shouldn't be any confusing. The application doesn't target users from a particular region. Hence it can be used by any person all throughout the world. Main requirements were gathered after brainstorming with couple of people like my friends and my colleagues at my internship who provided me an insight that I should save the website information for easier tracking. Some other requirements are obtained from my Major Professor Dr.Mitch Neilsen. The main requirements suggested are that I can provide an auto-fill of the source and destination from the list of places which were previously entered by the user and the other one is to provide navigation via google maps from the source and destination entered. The main requirements of this application are

- Provide a button view to add Trips via Flight or Trips via Car or view Upcoming trips from both of these.
- For each of the buttons Trips via Flight and Trips via car provide views which enable user to add, edit or delete a trip.
- For Future Trips provide an alarm option and let the user specify the exact date and time when he needs to be reminded.
- Provide navigation via Google maps from the source and destination entered by the user.
- View list of all trips once they are entered.
- Provide a tab view to view past, future trips and all trips and search through them.

4.2 Requirements Specification

4.2.1 Software Requirements

The set of software requirements for the application are listed below,

Development Perspective:

Operating System: Windows 8

Language: Android SDK, Java

Database: SQLite

Tools: Eclipse IDE

Technologies: Java, SQLite, Android, XML, Google Maps API

Debugger: Dalvik Debug Monitor Server (DDMS), Genymotion Simulator, Android mobile device (HTC One M8).

Application Perspective:

Framework: Android SDK Version 3.2

Network Required: Mobile network and Internet (cellular or Wi-Fi)

4.2.2 Hardware Requirements

Development Perspective:

Processor: Pentium IV or higher

RAM: 256 MB

Space on disk: 250MB or higher

Application Perspective:

Device: Android phone with version 3.2 or higher

4.3 Feasibility Analysis

4.3.1 Economic Feasibility

This application is economically feasible as it requires an Android device with Android SDK is 3.2 or higher which can be downloaded for free. But in order to download the application, the users need Wi-Fi or cellular network. Hence it is economically feasible.

4.3.2 Technical Feasibility

To develop this application we need a system to install Android SDK and develop the application and a device to test or it can also be tested on an emulator. This application has been tested on HTC One M8 and Genymotion Emulator. Hence it is Technically Feasible.

Chapter 5 - System Architecture and Design

5.1 System Architecture

The System architecture of an application gives us an overview of its interaction. The following is the figure which illustrates a high level architecture of the application MyTripPal. The user provides input to the application. This can be while Creating a tripviaFlight and tripviaCar, or while searching or editing the trips mainly. Once the user provides the input to the User Interface, it interacts with the logic present in the Java classes and other API's and then later interacts with the database and provides the result back to the User Interface for the user to view.

For example, in this application let us consider the case of creating a tripviaFlight. The user chooses the button TripsviaFlight from Homescreen.java which provides the user interface to fill in the details from MainActivity.java and in turn interacts with the database using TravelDatabase.java and then creates a trip in the database and the confirmation is sent back to the user via the User Interface.

For the fields, the user input can be filled in the textarea provided. For source and destination and airline name, there is an autocomplete option where the suggestions pop up from the values the user types in. Similarly, the user can mention data and time of his trip using the DatePicker. Similarly once the user checks the Alarm on checkbox, a DatePicker will be provided where he can mention the exact date and time. There is also a navigation option, if the user selects the navigation option on the top right corner of the trip, he will be given the navigation options from the source to destination he entered in the fields. This has been done using Google MapsV2 API. Figure 5.1 shows the system architecture of MyTripPal application.

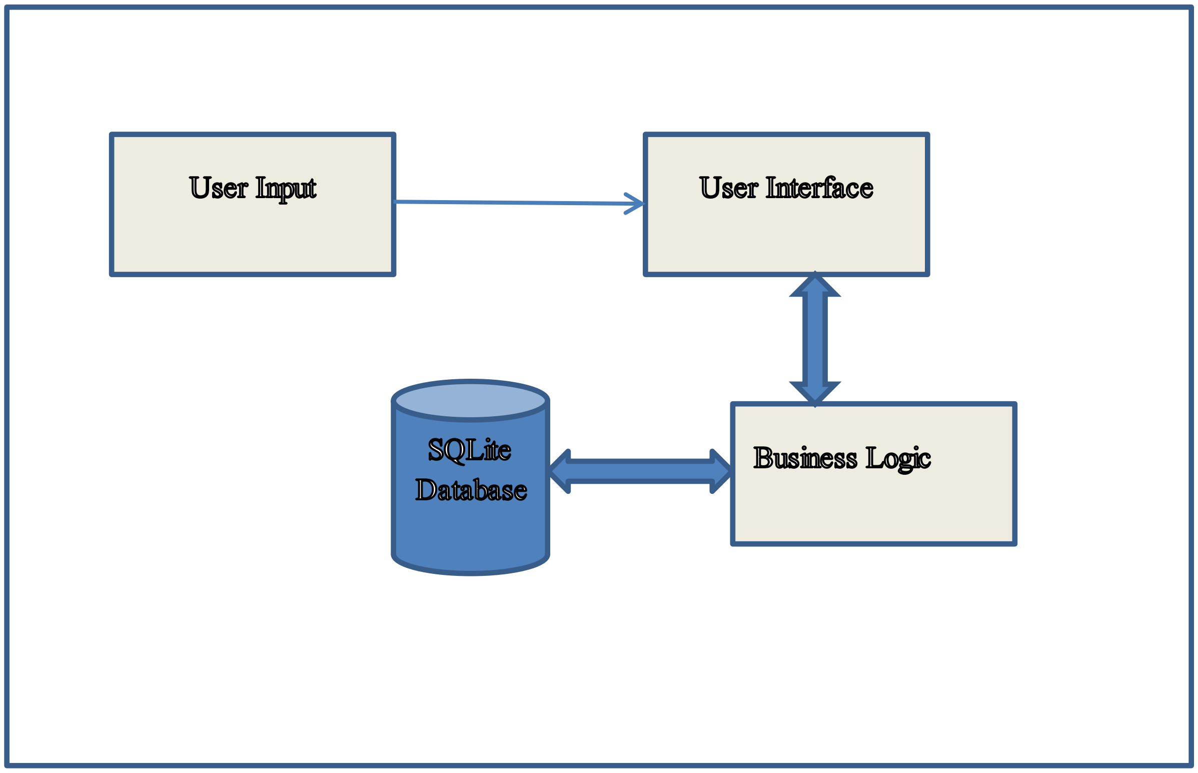


Figure 5.1 MyTripPal System Architecture

5.2 Use Case Diagram

Use Case Diagrams in UML can help us to describe the functionality of any system in a horizontal way. That is, instead of representing the details of individual features of a system, Use Case Diagrams can be helping us to see all of its available functionality. Use case Diagrams basically vary from sequence diagrams or flow charts because they do not represent the order or number of times that the systems actions and sub-actions must be executed. Use case Diagrams doesn't provide the exceptional cases. It helps us provide a high level view of the system.

The major elements of a Use Case Diagram are explained below.

Actors: Specifies the role played by a user or it is any system that we are describing interacts with.

Use cases: Use case is a list of steps which typically defines the interactions between a role and a system, to achieve a goal.

Lines: Lines help us in representing the relationships between the above elements.

The use case interactions of the application MyTripPal can be seen in the Figure 5.2 below. [5]

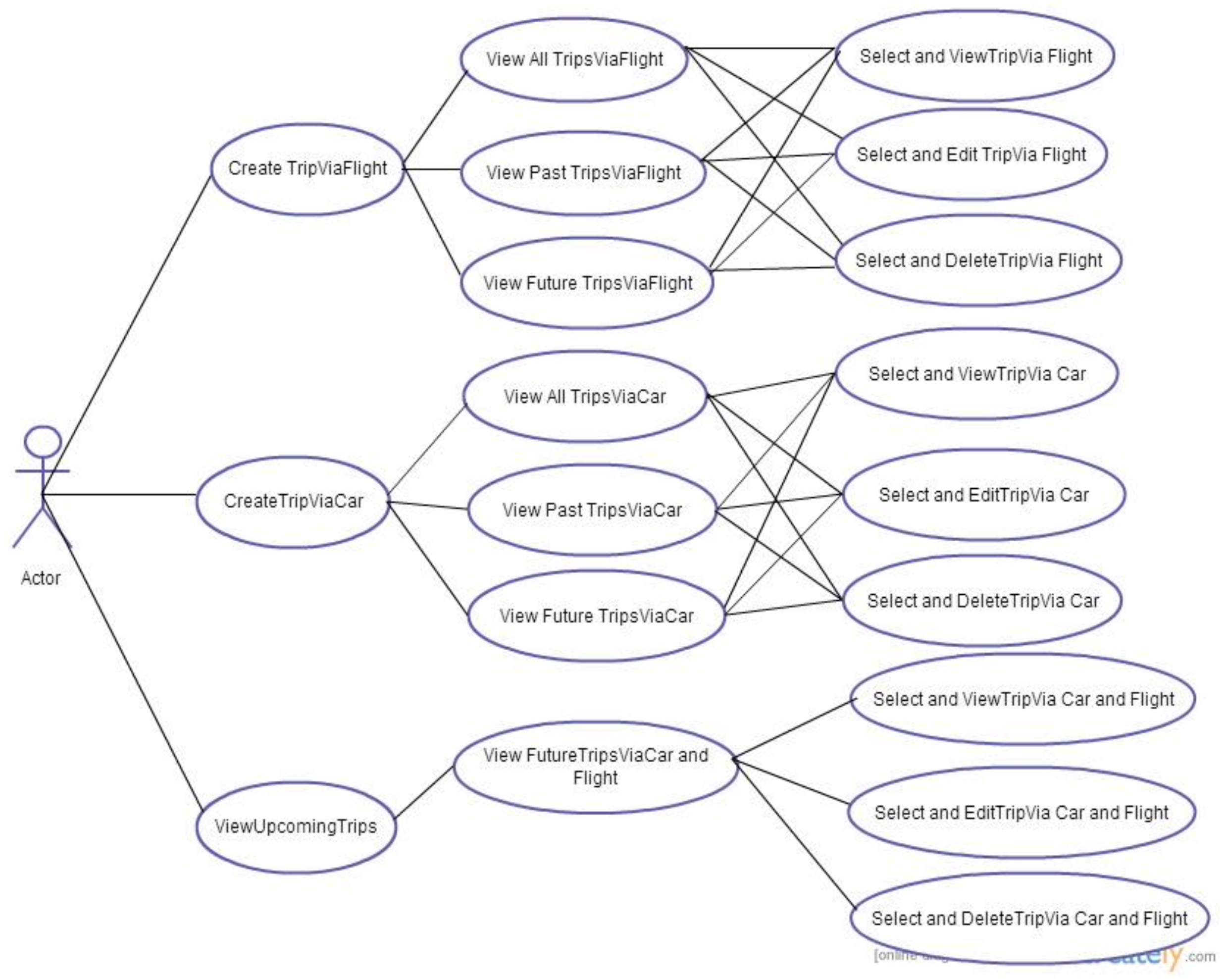
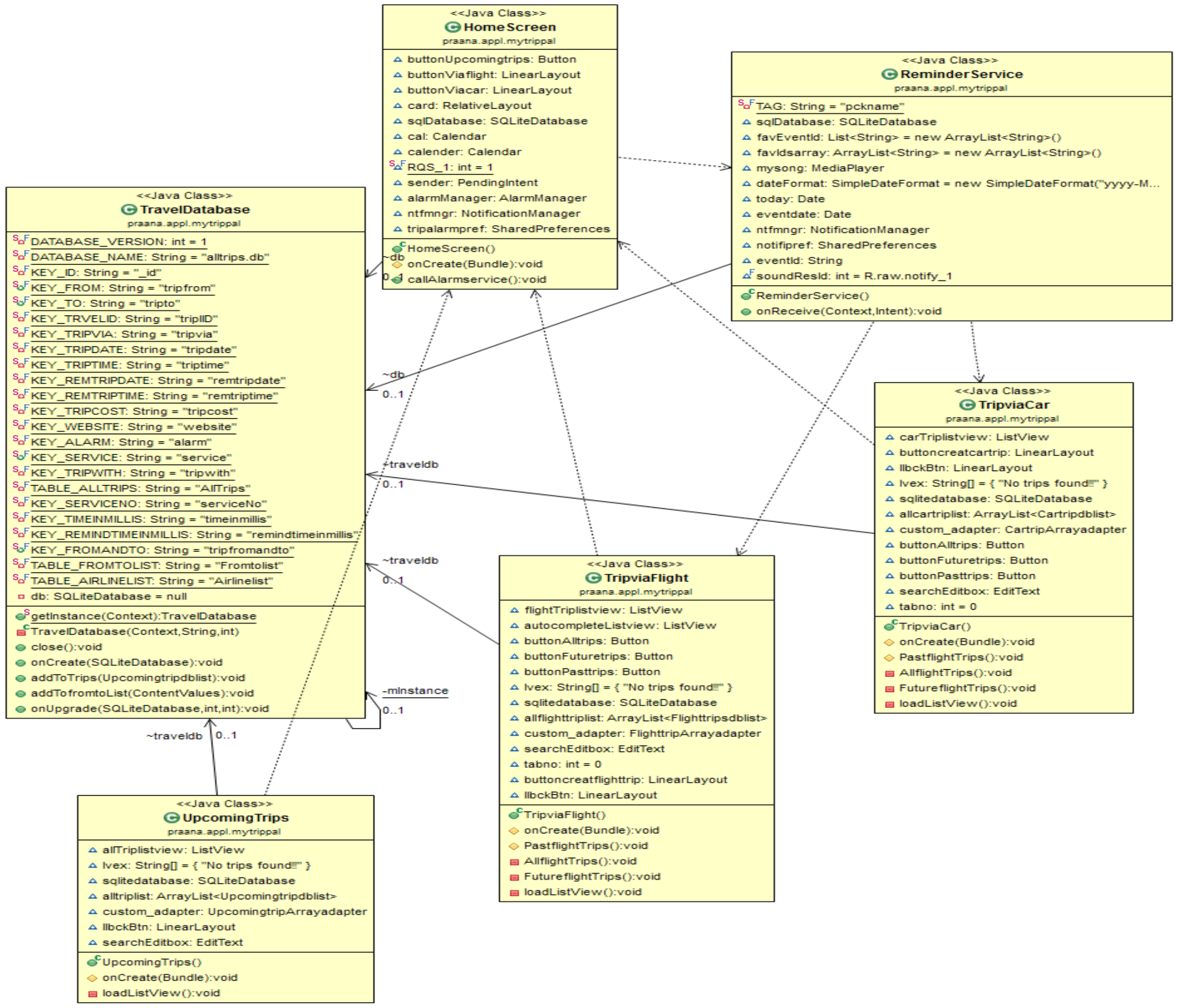

Actor
Create TripViaFlight
View All TripsViaFlight
View Past TripsViaFlight
View Future TripsViaFlight
Select and ViewTripVia Flight
Select and Edit TripVia Flight
Select and DeleteTripVia Flight
CreateTripViaCar
View All TripsViaCar
View Past TripsViaCar
View Future TripsViaCar
Select and ViewTripVia Car
Select and EditTripVia Car
Select and DeleteTripVia Car
ViewUpcomingTrips
View FutureTripsViaCar and Flight
Select and ViewTripVia Car and Flight
Select and EditTripVia Car and Flight
Select and DeleteTripVia Car and Flight

Figure 5.2 MyTripPal Use Case Diagram

5.3 Class Diagrams

Class diagrams in the Unified Modeling Language help us understand the structure of a system by showing us the system's classes, their attributes, methods and the relationships among objects. It is the main building block of Object Oriented Modelling. Figures 5.1 to 5.6 illustrates System design in the form of the class diagrams. [6]

5.3.1 Class Diagram for the Homescreen:



<<Java Class>>
HomeScreen
praana.appl.mytrippal
buttonUpcomingtrips: Button
buttonViaflight: LinearLayout
buttonViacar: LinearLayout
card: RelativeLayout
sqlDatabase: SQLiteDatabase
cal: Calendar
calender: Calendar
RQS_1: int = 1
sender: PendingIntent
alarmManager: AlarmManager
ntfmngr: NotificationManager
tripalarmpref: SharedPreferences
HomeScreen()
onCreate(Bundle):void
callAlarmservice():void
<<Java Class>>
ReminderService
praana.appl.mytrippal
TAG: String = "pckname"
sqlDatabase: SQLiteDatabase
favEventId: List<String> = new ArrayList<String>()
favIdsarray: ArrayList<String> = new ArrayList<String>()
mysong: MediaPlayer
dateFormat: SimpleDateFormat = new SimpleDateFormat("yyyy-M...
today: Date
eventdate: Date
ntfmngr: NotificationManager
notifipref: SharedPreferences
eventId: String
soundResId: int = R.raw.notify_1
ReminderService()
onReceive(Context,Intent):void
<<Java Class>>
TravelDatabase
praana.appl.mytrippal
DATABASE_VERSION: int = 1
DATABASE_NAME: String = "alltrips.db"
KEY_ID: String = "_id"
KEY_FROM: String = "tripfrom"
KEY_TO: String = "tripto"
KEY_TRVELID: String = "tripID"
KEY_TRIPVIA: String = "tripvia"
KEY_TRIPDATE: String = "tripdate"
KEY_TRIPTIME: String = "triptime"
KEY_REMTRIPDATE: String = "remtripdate"
KEY_REMTRIPTIME: String = "remtriptime"
KEY_TRIPCOST: String = "tripcost"
KEY_WEBSITE: String = "website"
KEY_ALARM: String = "alarm"
KEY_SERVICE: String = "service"
KEY_TRIPWITH: String = "tripwith"
TABLE_ALLTRIPS: String = "AllTrips"
KEY_SERVICENO: String = "serviceNo"
KEY_TIMEINMILLIS: String = "timeinmillis"
KEY_REMINDTIMEINMILLIS: String = "remindtimeinmillis"
KEY_FROMANDTO: String = "tripfromandto"
TABLE_FROMTOLIST: String = "Fromtolist"
TABLE_AIRLINELIST: String = "Airlinelist"
db: SQLiteDatabase = null
getInstance(Context):TravelDatabase
TravelDatabase(Context,String,int)
close():void
onCreate(SQLiteDatabase):void
addToTrips(Upcomingtripdblist):void
addTofromtoList(ContentValues):void
onUpgrade(SQLiteDatabase,int,int):void
~db
0..1
~db
0..1
~traveldb
0..1
~traveldb
0..1
-mInstance
0..1
~traveldb
0..1
<<Java Class>>
TripviaCar
praana.appl.mytrippal
carTriplistview: ListView
buttoncreatcartrip: LinearLayout
llbckBtn: LinearLayout
lvex: String[] = { "No trips found!!" }
sqlitedatabase: SQLiteDatabase
allcartriplist: ArrayList<Cartripdblist>
custom_adapter: CartripArrayadapter
buttonAlltrips: Button
buttonFuturetrips: Button
buttonPasttrips: Button
searchEditbox: EditText
tabno: int = 0
TripviaCar()
onCreate(Bundle):void
PastflightTrips():void
AllflightTrips():void
FutureflightTrips():void
loadListView():void
<<Java Class>>
TripviaFlight
praana.appl.mytrippal
flightTriplistview: ListView
autocompleteListview: ListView
buttonAlltrips: Button
buttonFuturetrips: Button
buttonPasttrips: Button
lvex: String[] = { "No trips found!!" }
sqlitedatabase: SQLiteDatabase
allflighttriplist: ArrayList<Flighttripsdblist>
custom_adapter: FlighttripArrayadapter
searchEditbox: EditText
tabno: int = 0
buttoncreatflighttrip: LinearLayout
llbckBtn: LinearLayout
TripviaFlight()
onCreate(Bundle):void
PastflightTrips():void
AllflightTrips():void
FutureflightTrips():void
loadListView():void
<<Java Class>>
UpcomingTrips
praana.appl.mytrippal
allTriplistview: ListView
lvex: String[] = { "No trips found!!" }
sqlitedatabase: SQLiteDatabase
alltriplist: ArrayList<Upcomingtripdblist>
custom_adapter: UpcomingtripArrayadapter
llbckBtn: LinearLayout
searchEditbox: EditText
UpcomingTrips()
onCreate(Bundle):void
loadListView():void

Figure 5.3 Class Diagram for HomeScreen

5.3.2 Class Diagram for Trip via Flight:

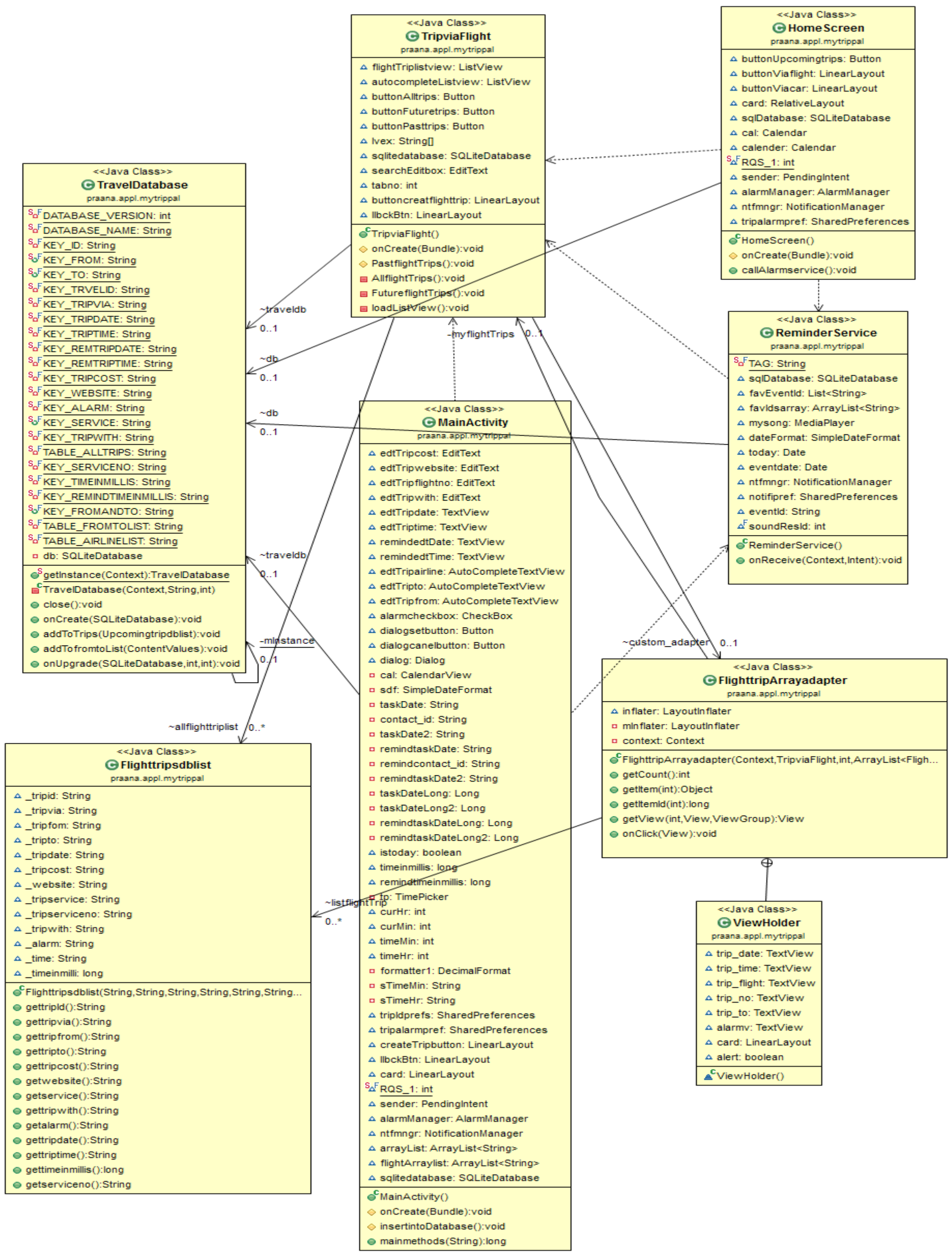
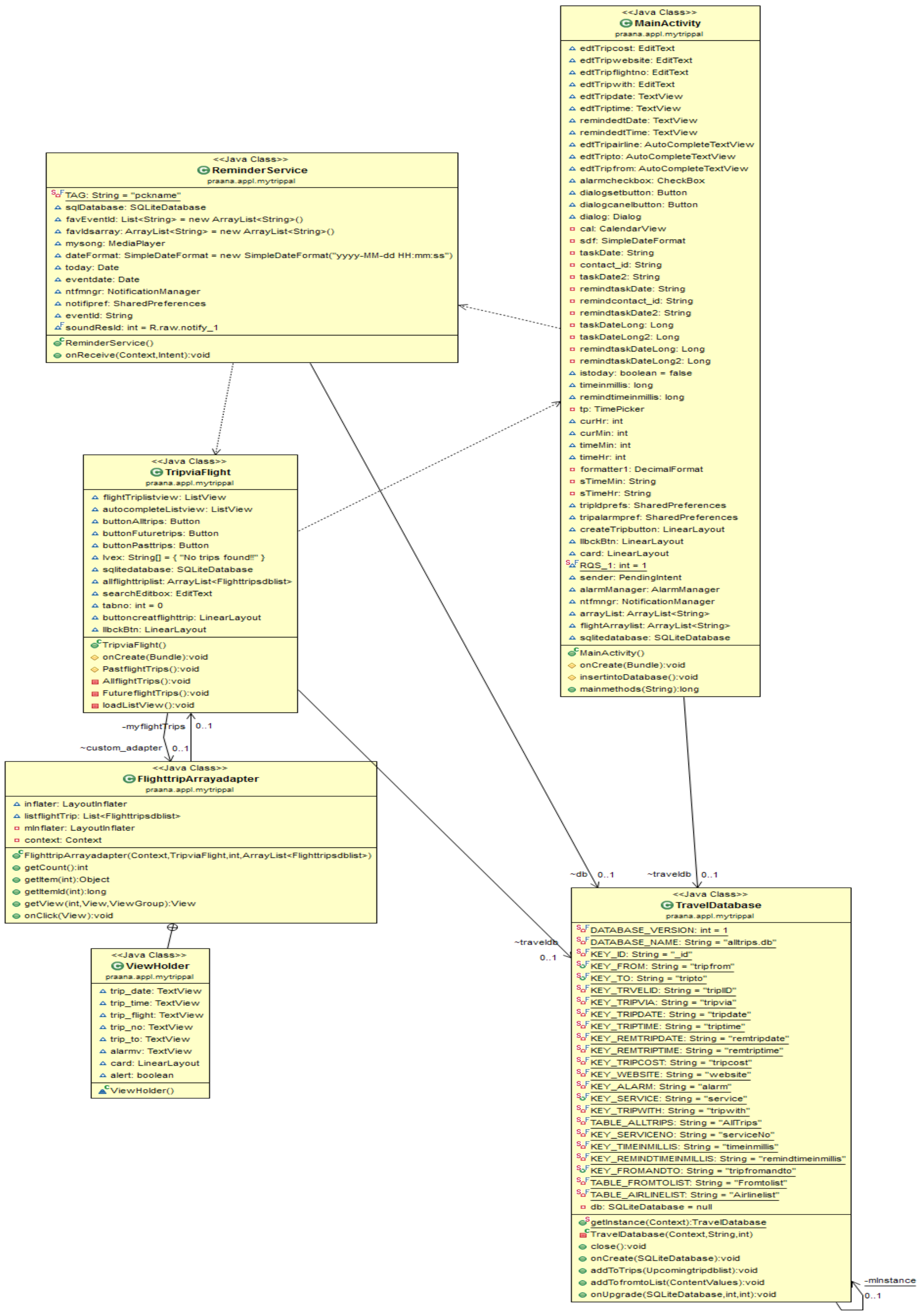


Figure 5.4 Class diagram for Trip via Flight

5.3.3 Class Diagram for the Creating a new Trip via Flight:



<<Java Class>>
MainActivity
praana.appl.mytrippal
edtTripcost: EditText
edtTripwebsite: EditText
edtTripflightno: EditText
edtTripwith: EditText
edtTripdate: TextView
edtTriptime: TextView
remindedtDate: TextView
remindedtTime: TextView
edtTripairline: AutoCompleteTextView
edtTripto: AutoCompleteTextView
edtTripfrom: AutoCompleteTextView
alarmcheckbox: CheckBox
dialogsetbutton: Button
dialogcanelbutton: Button
dialog: Dialog
cal: CalendarView
sdf: SimpleDateFormat
taskDate: String
contact_id: String
taskDate2: String
remindtaskDate: String
remindcontact_id: String
remindtaskDate2: String
taskDateLong: Long
taskDateLong2: Long
remindtaskDateLong: Long
remindtaskDateLong2: Long
istoday: boolean = false
timeinmillis: long
remindtimeinmillis: long
tp: TimePicker
curHr: int
curMin: int
timeMin: int
timeHr: int
formatter1: DecimalFormat
sTimeMin: String
sTimeHr: String
tripIdprefs: SharedPreferences
tripalarmpref: SharedPreferences
createTripbutton: LinearLayout
llbckBtn: LinearLayout
card: LinearLayout
RQS_1: int = 1
sender: PendingIntent
alarmManager: AlarmManager
ntfmngr: NotificationManager
arrayList: ArrayList<String>
flightArraylist: ArrayList<String>
sqlitedatabase: SQLiteDatabase
MainActivity()
onCreate(Bundle):void
insertintoDatabase():void
mainmethods(String):long
<<Java Class>>
ReminderService
praana.appl.mytrippal
TAG: String = "pckname"
sqlDatabase: SQLiteDatabase
favEventId: List<String> = new ArrayList<String>()
favIdsarray: ArrayList<String> = new ArrayList<String>()
mysong: MediaPlayer
dateFormat: SimpleDateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss")
today: Date
eventdate: Date
ntfmngr: NotificationManager
notifipref: SharedPreferences
eventId: String
soundResId: int = R.raw.notify_1
ReminderService()
onReceive(Context,Intent):void
<<Java Class>>
TripviaFlight
praana.appl.mytrippal
flightTriplistview: ListView
autocompleteListview: ListView
buttonAlltrips: Button
buttonFuturetrips: Button
buttonPasttrips: Button
lvex: String[] = { "No trips found!!" }
sqlitedatabase: SQLiteDatabase
allflighttriplist: ArrayList<Flighttripsdblist>
searchEditbox: EditText
tabno: int = 0
buttoncreatflighttrip: LinearLayout
llbckBtn: LinearLayout
TripviaFlight()
onCreate(Bundle):void
PastflightTrips():void
AllflightTrips():void
FutureflightTrips():void
loadListView():void
~myflightTrips
0..1
~custom_adapter
0..1
<<Java Class>>
FlighttripArrayadapter
praana.appl.mytrippal
inflater: LayoutInflater
listflightTrip: List<Flighttripsdblist>
mInflater: LayoutInflater
context: Context
FlighttripArrayadapter(Context,TripviaFlight,int,ArrayList<Flighttripsdblist>)
getCount():int
getItem(int):Object
getItemId(int):long
getView(int,View,ViewGroup):View
onClick(View):void
<<Java Class>>
ViewHolder
praana.appl.mytrippal
trip_date: TextView
trip_time: TextView
trip_flight: TextView
trip_no: TextView
trip_to: TextView
alarmv: TextView
card: LinearLayout
alert: boolean
ViewHolder()
~db
0..1
~traveldb
0..1
~traveldb
0..1
<<Java Class>>
TravelDatabase
praana.appl.mytrippal
DATABASE_VERSION: int = 1
DATABASE_NAME: String = "alltrips.db"
KEY_ID: String = "_id"
KEY_FROM: String = "tripfrom"
KEY_TO: String = "tripto"
KEY_TRVELID: String = "tripID"
KEY_TRIPVIA: String = "tripvia"
KEY_TRIPDATE: String = "tripdate"
KEY_TRIPTIME: String = "triptime"
KEY_REMTRIPDATE: String = "remtripdate"
KEY_REMTRIPTIME: String = "remtriptime"
KEY_TRIPCOST: String = "tripcost"
KEY_WEBSITE: String = "website"
KEY_ALARM: String = "alarm"
KEY_SERVICE: String = "service"
KEY_TRIPWITH: String = "tripwith"
TABLE_ALLTRIPS: String = "AllTrips"
KEY_SERVICENO: String = "serviceNo"
KEY_TIMEINMILLIS: String = "timeinmillis"
KEY_REMINDTIMEINMILLIS: String = "remindtimeinmillis"
KEY_FROMANDTO: String = "tripfromandto"
TABLE_FROMTOLIST: String = "Fromtolist"
TABLE_AIRLINELIST: String = "Airlinelist"
db: SQLiteDatabase = null
getInstance(Context):TravelDatabase
TravelDatabase(Context,String,int)
close():void
onCreate(SQLiteDatabase):void
addToTrips(Upcomingtripdblist):void
addTofromtoList(ContentValues):void
onUpgrade(SQLiteDatabase,int,int):void
-mInstance
0..1

Figure 5.5 Class Diagram for Create a Trip via Flight

5.3.4 Class diagram for Trip via Car:

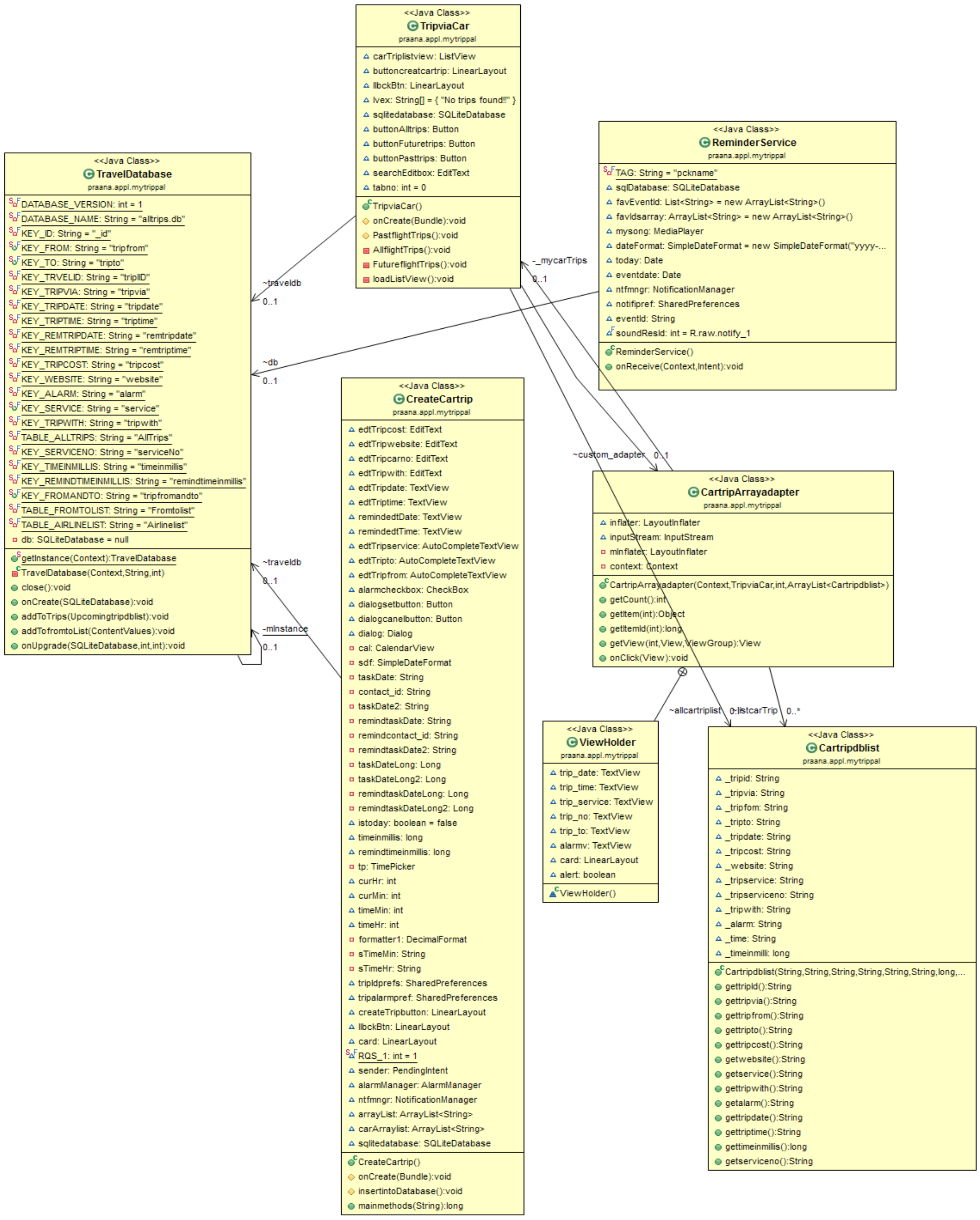


Figure 5.6 Class Diagram for Trip via Car

5.3.5 Class Diagram for the Creating a new Trip via Car:

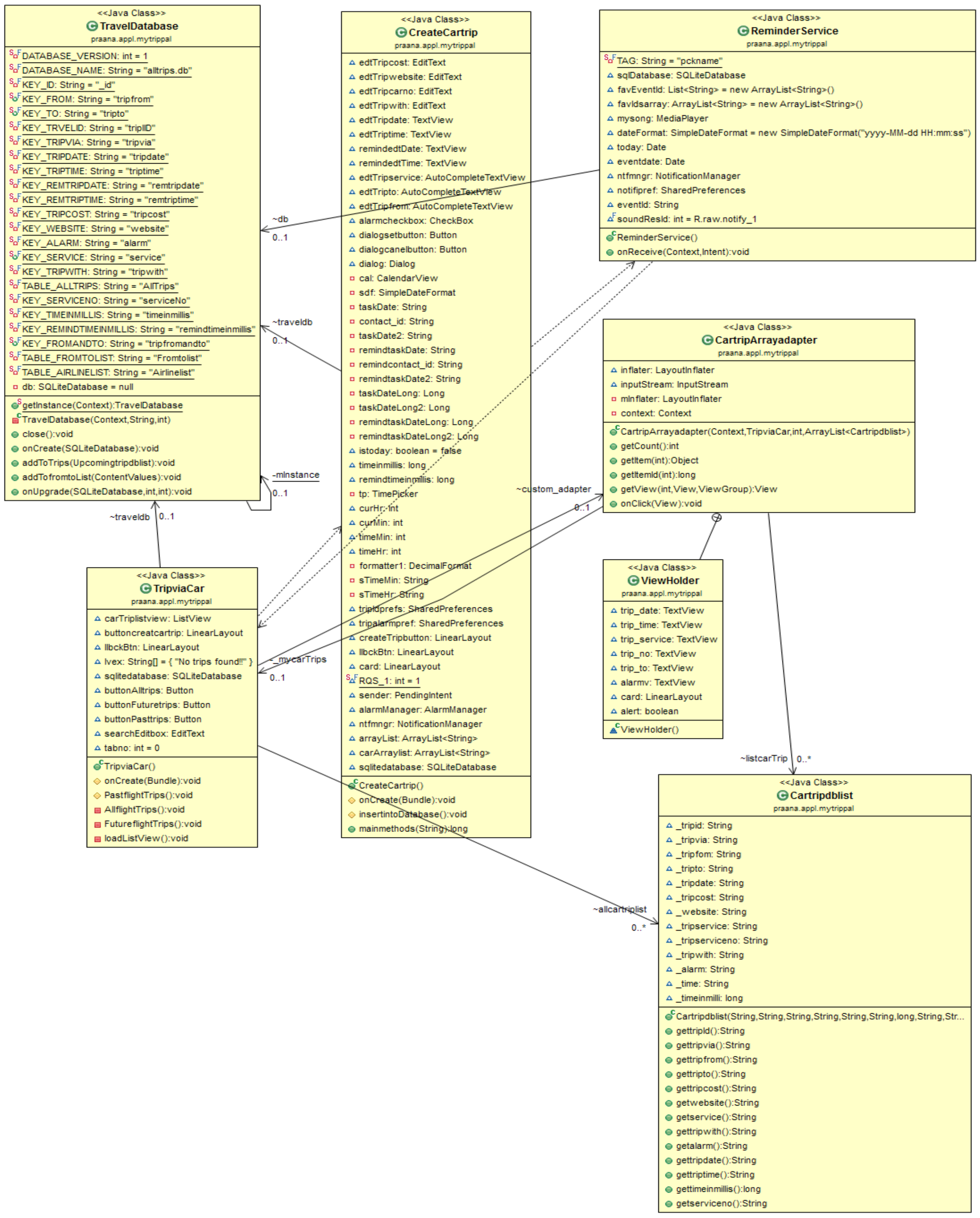


Figure 5.7 Class Diagram for Create a Trip via Car

5.3.6 Class Diagram for Upcoming Trips:

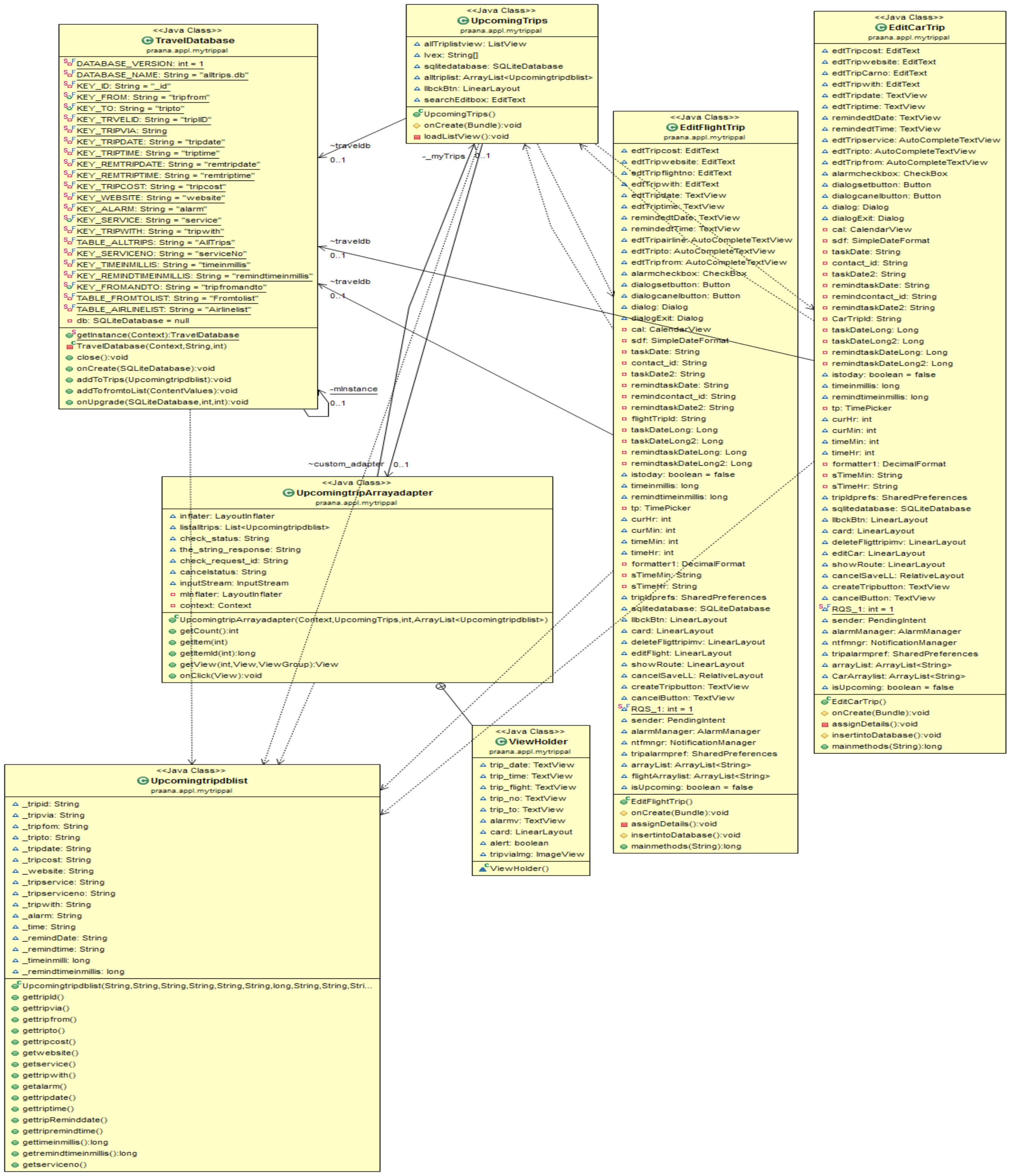


Figure 5.8 Class diagram for Upcoming Trips

5.4 Sequence Diagram

A sequence diagram shows the interaction of various processes. It shows the interaction of objects in an arranged time sequence. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development [13]. The sequence diagram for the events that occur when the user selects the Trip via Flight and chooses to view a trip and edit a trip.

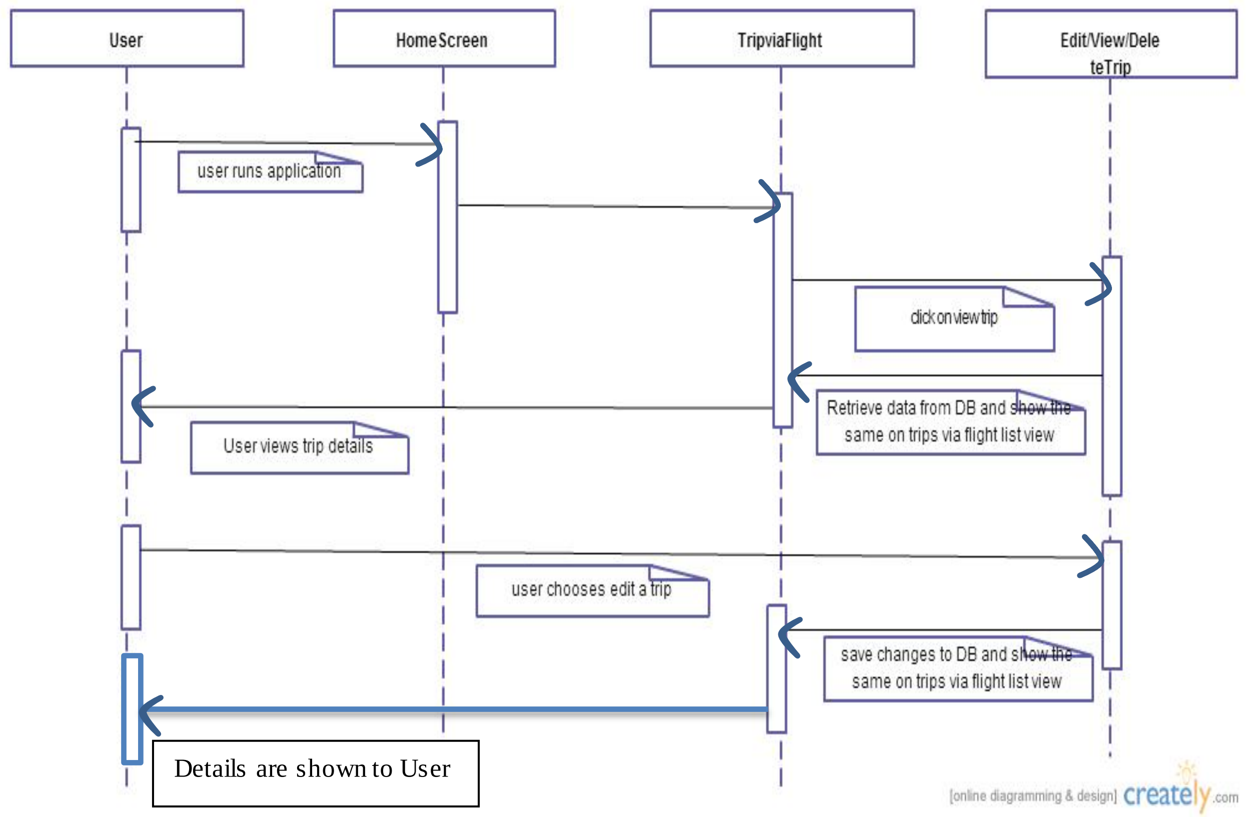

User
HomeScreen
TripviaFlight
Edit/View/DeleteTrip
user runs application
click on view trip
Retrieve data from DB and show the same on trips via flight list view
User views trip details
user chooses edit a trip
save changes to DB and show the same on trips via flight list view
Details are shown to User
[online diagramming & design] creately.com

Figure 5.9 Sequence Diagram for Create a Trip via Flight

Chapter 6 - Android Framework Components

The Android Software Development Kit (SDK) provides us all the required API libraries and developer tools to build, test, and debug apps for Android. The application MyTripPal has been developed in Eclipse Integrated Development Environment (IDE) with the Android Developer Tool (ADT) plugin. The application can be run on an emulator or an .apk file can be generated which can be installed on an android smartphone or tablet. [9]

6.1 AndroidManifest.xml

Every android application should have an AndroidManifest.xml file to work in its root directory. This file provides the essential information about the application to the Android system. It provides the point from where the application should start that is

```
<category android:name="android.intent.category.LAUNCHER" />
```

Among the other things, the manifest file does the following:

It names the Java package for the application which serves as a unique identifier for the android application. It also provides a description of the components of the application that is the activities, services, broadcast receivers, and content providers comprises. These declarations enable the Android system to know the list of components and the conditions under which they can be launched.

It tells about the permissions an application has to access protected parts of the API and its permissions to interact with other applications. Similarly tells us about the permissions other applications have to interact with the current application components.

It also talks about the minimum sdk version and the target sdk version of the application. The manifest file for MyTripPal can be seen below:

```
<?xml  version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="praana.appl.mytrippal"
   android:versionCode="1"
   android:versionName="1.0" >

   <uses-sdk
       android:minSdkVersion="14"
       android:targetSdkVersion="20" />

   <application
       android:allowBackup="true"
       android:icon="@drawable/logo_mtp"
       android:label="@string/app_name"
       android:theme="@style/NoActionBar" >
       <activity
           android:name="praana.appl.mytrippal.MainActivity"
           android:label="@string/app_name"
           android:screenOrientation="nosensor"
           android:windowSoftInputMode="stateHidden|adjustResize" >
           <intent-filter>
               <action android:name="android.intent.action.MAINACTIVITY" />

               <category android:name="android.intent.category.DEFAULT" />
           </intent-filter>
       </activity>

       <receiver
           android:name="praana.appl.mytrippal.ReminderService"
           android:label="@string/app_name" >
           <intent-filter>
               <action android:name="android.intent.action.REMINDERSERVICE" />

               <category android:name="android.intent.category.DEFAULT" />
           </intent-filter>
 >
       </receiver>

       <activity
           android:name="praana.appl.mytrippal.HomeScreen"
           android:label="@string/app_name"
           android:screenOrientation="nosensor"
           android:windowSoftInputMode="stateHidden|adjustResize" >
           <intent-filter>
               <action android:name="android.intent.action.MAIN" />

               <category android:name="android.intent.category.LAUNCHER" />
           </intent-filter>
       </activity>
       <activity
           android:name="praana.appl.mytrippal.TripviaFlight"
           android:label="@string/app_name"
           android:screenOrientation="nosensor"
           android:windowSoftInputMode="stateHidden|adjustResize">
           <intent-filter>
```

```
            <action android:name="android.intent.action.TRIPVIAFLIGHT"  />

            <category android:name="android.intent.category.DEFALT" />
          </intent-filter>
      </activity>
      <activity
          android:name="praana.appl.mytrippal.TripviaCar"
          android:label="@string/app_name"
          android:screenOrientation="nosensor"
          android:windowSoftInputMode="stateHidden|adjustResize">
          <intent-filter>
            <action android:name="android.intent.action.TRIPVIACAR" />

            <category android:name="android.intent.category.DEFALT" />
          </intent-filter>
      </activity>
      <activity
          android:name="praana.appl.mytrippal.CreateCartrip"
          android:label="@string/app_name"
          android:screenOrientation="nosensor"
          android:windowSoftInputMode="stateHidden|adjustResize" >
          <intent-filter>
            <action android:name="android.intent.action.CREATECARTRIP" />

            <category android:name="android.intent.category.DEFALT" />
          </intent-filter>
      </activity>
      <activity
          android:name="praana.appl.mytrippal.UpcomingTrips"
          android:label="@string/app_name"
          android:screenOrientation="nosensor"
          android:windowSoftInputMode="stateHidden|adjustResize">
          <intent-filter>
            <action android:name="android.intent.action.UPCOMINGTRIPS" />

            <category android:name="android.intent.category.DEFALT" />
          </intent-filter>
      </activity>
      <activity android:name="praana.appl.mytrippal.EditFlightTrip"
          android:screenOrientation="nosensor"
          android:windowSoftInputMode="stateHidden|adjustResize">
          <intent-filter>
            <action android:name="android.intent.action.EDITFLIGHTTRIP"  />
          </intent-filter>
      </activity>

       <activity android:name="praana.appl.mytrippal.EditCarTrip"
          android:screenOrientation="nosensor"
          android:windowSoftInputMode="stateHidden|adjustResize">
          <intent-filter>
            <action android:name="android.intent.action.EDITCARTRIP" />
          </intent-filter>
      </activity>
    </application>

</manifest>
```

From the manifest file, we can see that the minimum sdk version for this application is 13. Thus the devices below sdk version 13 that is android Honeycomb_MR2 will not be able to run this application. Different activities and their intents are mentioned in this file for MyTripPal [11].

6.2 AndroidDependencies

Android Dependencies is a virtual folder where we find JAR files that Eclipse uses for the project. It is a virtual folder and will not be found on the hard disk. For MyTripPal android-support-v7-appcompat.jar has been added to enable ActionBar and it in turn depends on v4 Support Library.

6.3 Activity

Activity is an important component in an application which helps providing screen for the users to interact. All the activities in a system are managed as an activity stack that is the last in first out procedure. So if a new activity is started, it is placed on the top and will be the current activity that is running and the previous activity will be in the background.

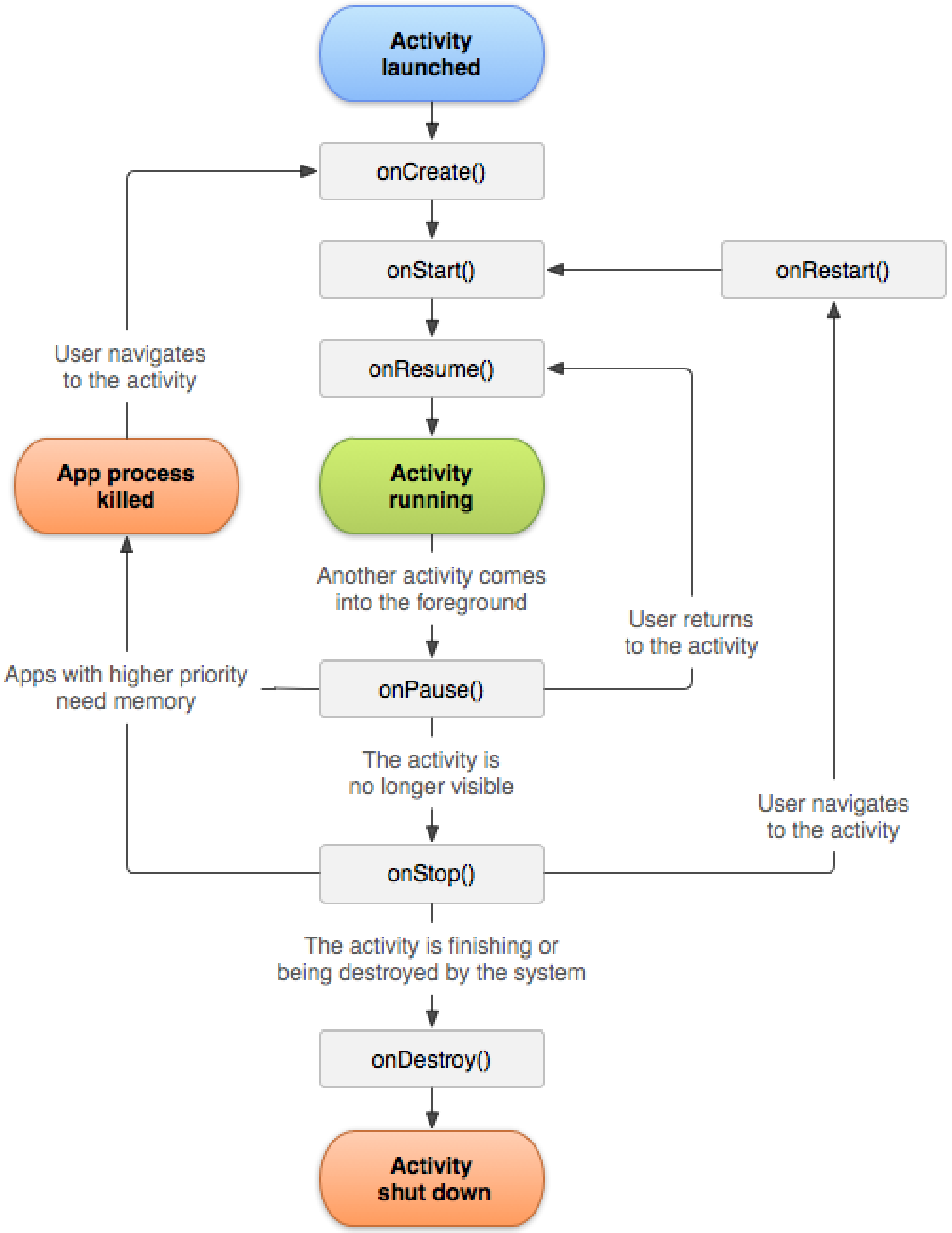
- An activity has essentially four states:
- An activity which is in foreground and is running can be called active state.
- An activity that has lost focus but is still visible can be called a paused state
- An activity that is paused or stopped can be dropped by the system from memory by either asking it to finish, or by killing its process.

An activity moves between states. For example while filling data when we open a calendar to fill in the date the calendar activity is the current activity that is created and runs.

```
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.create_trip);
        traveldb = TravelDatabase.getInstance(MainActivity.this);

        arrayList = new ArrayList<String>();
        flightArraylist = new ArrayList<String>();
```

The previous activity with all the TextArea will be the background activity which is in onpause() state. The following diagram shows the important state paths of an Activity. [12]



Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Figure 6.1 Activity Life Cycle [12, Fig.1]

6.4 Intent

Intent is an android component which provides an abstract description of an operation which is to be performed. The two primary forms of intents are

- **Explicit Intents:** These intents have a specified component that is via setComponent (ComponentName) or setClass(Context, Class), thus providing the exact classes to run.
- **Implicit Intents:** These intents will not have specified a component. Thus they include enough information for the system to determine which of the available components suits the best to run for that intent [13].

Thus these intents helps in launching activities. You can call one activity from another. For example in the code below, ManiActivity.this activity calls TripviaFlight.class.

```
Intent i = new Intent(MainActivity.this, TripviaFlight.class);
```

6.5 Layout Inflater

Layout Inflater android component helps loading the layout XML file into its view objects such as ProgressBar, TextView etc. It is used in conjunction getLayoutInflater() or getSystemService(String) to retrieve a standard LayoutInflater instance which is already hooked up to the current context [14].

```
LayoutInflater li = (LayoutInflater) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
```

Chapter 7 - Graphical User Interface

The front end of the application that is the User Interface has been developed using XML. It is simple to create and has richness of the data structure.

7.1 Home Page:

The Figure 7.1 below shows the HomeScreen for MyTrippal application. It contains buttons for TripviaFlight to perform CRUD (Create Update and Delete) operations related to trips via flight. Similarly it has the button Tripviacar to track the trips via car and there is an Upcoming trips tab which lists all the past and future dated trips via flight and car.

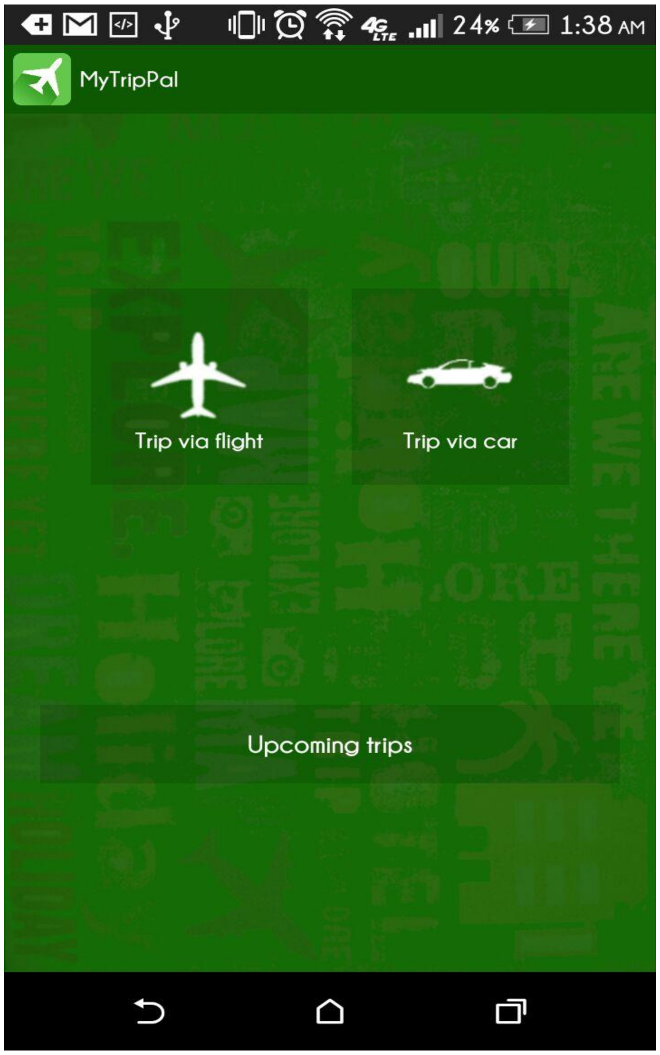


Figure 7.1 HomeScreen View of the Application-MyTripPal

7.2 Trips via Flight

Once the user presses on the Trips via Flight button, he will be navigated to the screen where he will be presented with a screen with tabs AllTrips, PastTrips and FutureTrips and he can choose to create a new trip by clicking on the "+" button on the top-right corner of the screen, for which the fields are listed as shown below in the diagram. Once all values are entered, he can click on the Save button present on the top-right corner of the screen.

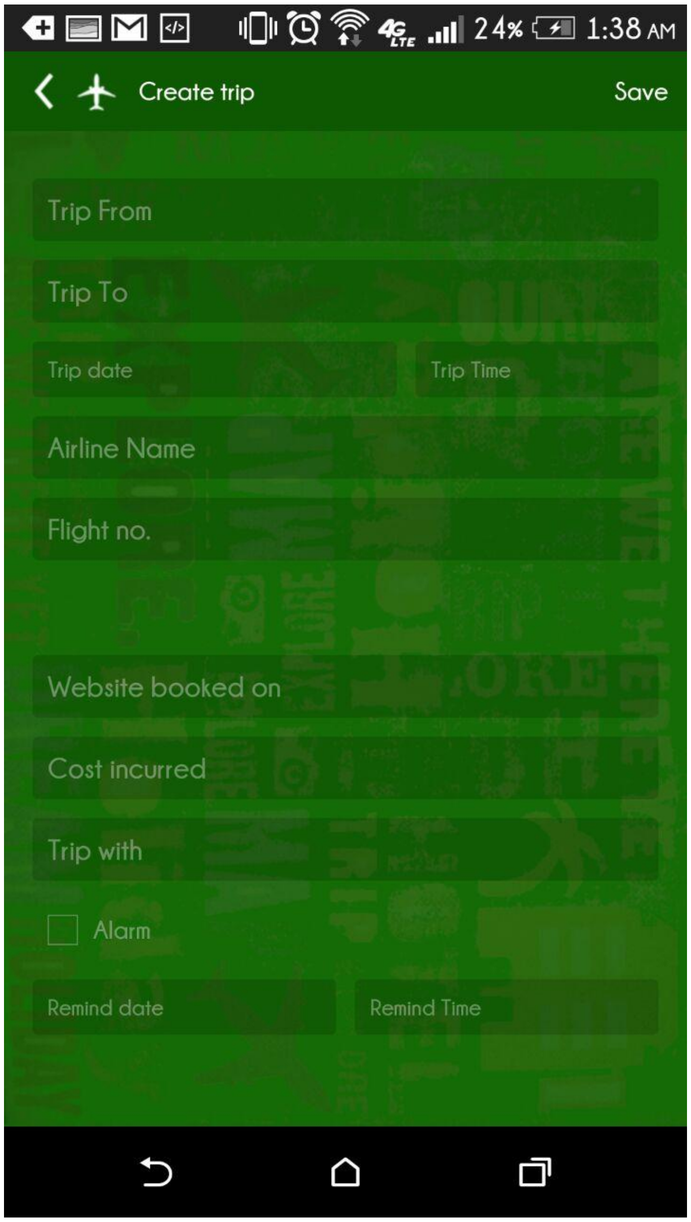


Figure 7.2 Creating a Trip via Flight

7.3 Autocomplete Option:

Allowing the application to be generic, I haven't instilled any constraints on the source, destination and the airlines. But at the same time to allow the user to have a feasibility of choosing the source, destination or airlines from the values he previously entered, an autocomplete option has been provided which provides the i=user suggestion by populating the values from the SQLite database.

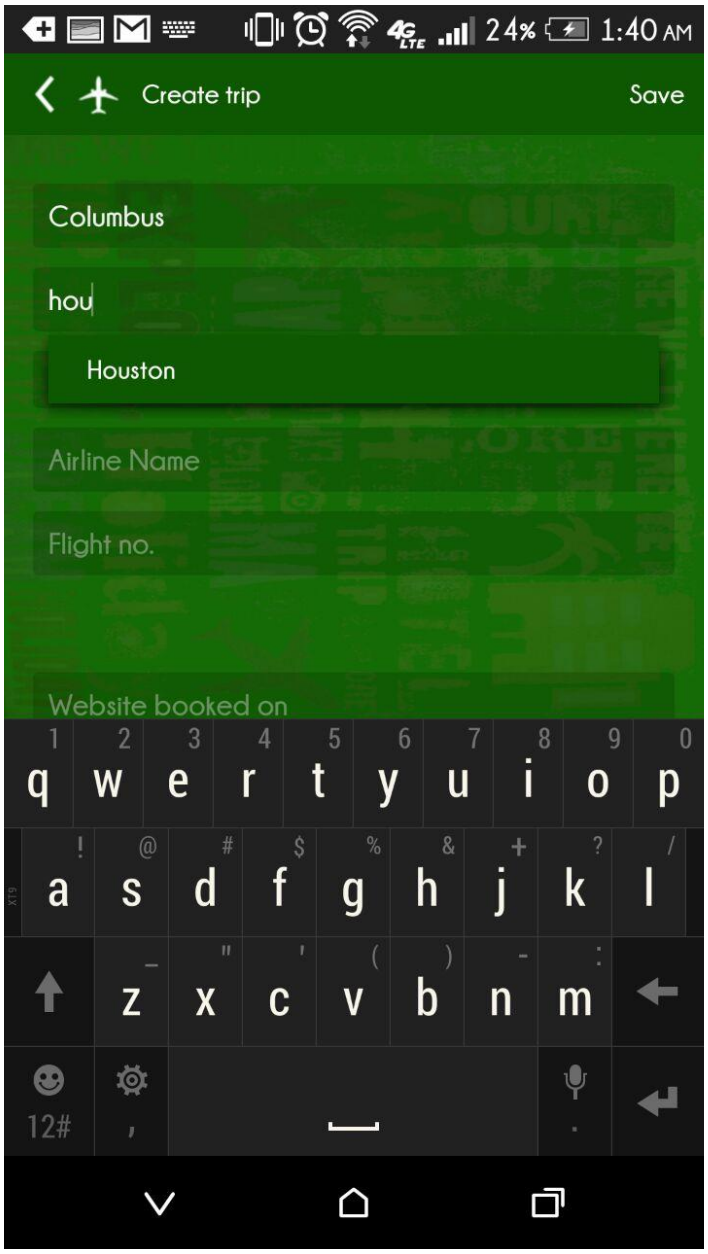


Figure 7.3 Autocomplete Option

7.4 Calendar view to choose date and Alarm option

Instead of making the user go through the hassle of entering the date and confuse among various date formats (as different countries have different date formats) , I have given a Datepicker to the user which enables him to pick the date and at the same time traverse through various years or months. Another time widget is added which enables the user to choose a time in 24 hour format in order to set the alarm. The same widgets are used for the alarm option too.

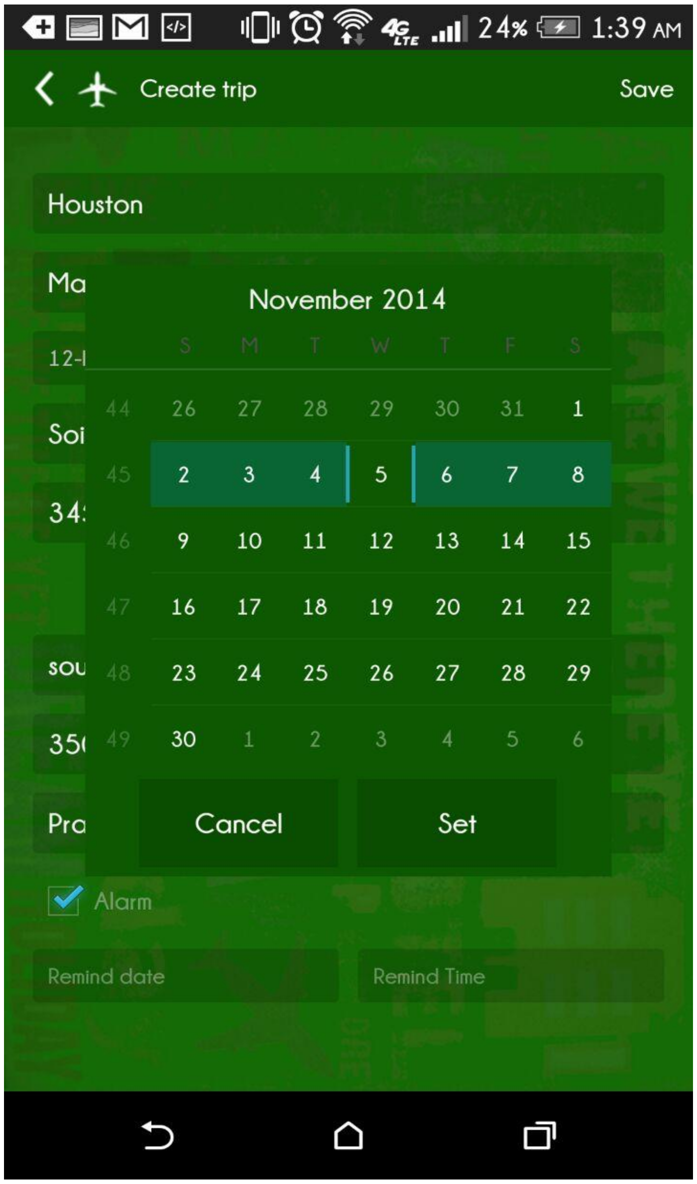


Figure 7.4 Date Picker

7.5 Alarm Validations

For any future trip; i.e., in both cases of trip via car or trip via flight, the alarm can be set and cannot be set for past dates. Just in cases the user tries to do say, toast messages are displayed to let the user that it is an invalid operation. Then, the user can save a trip.

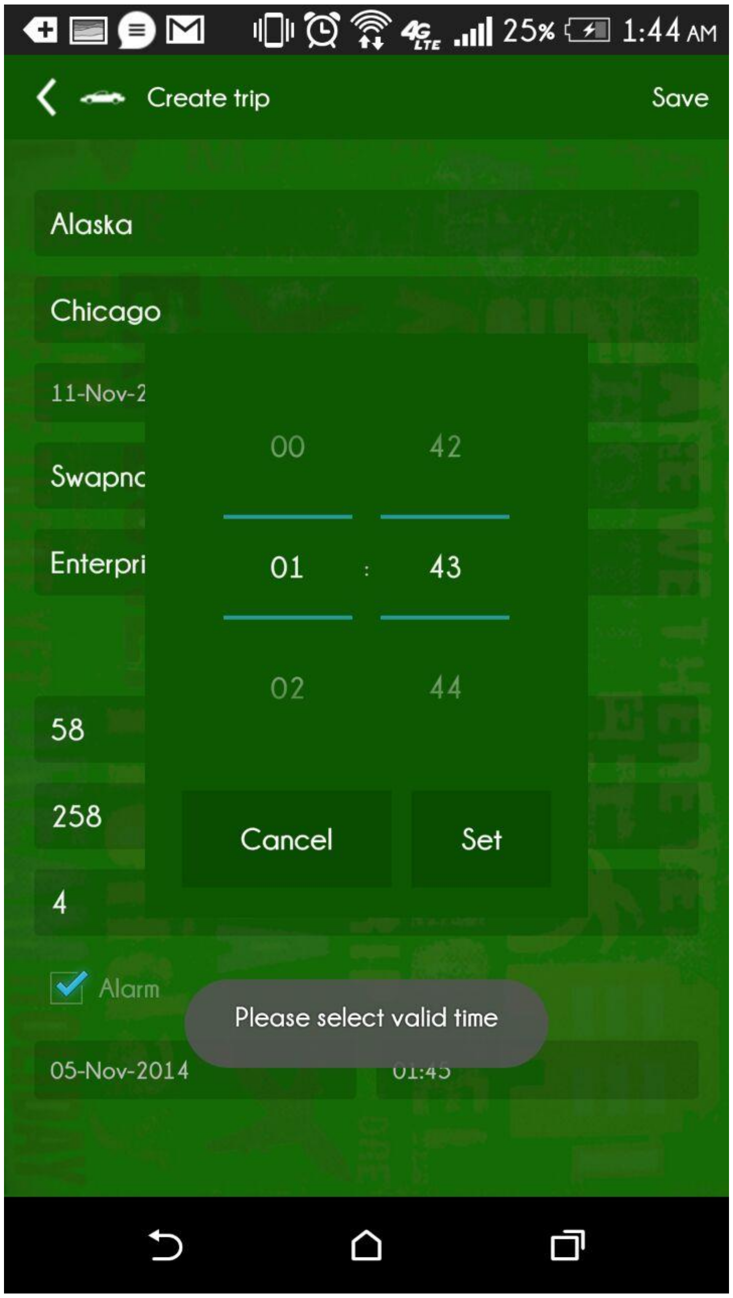
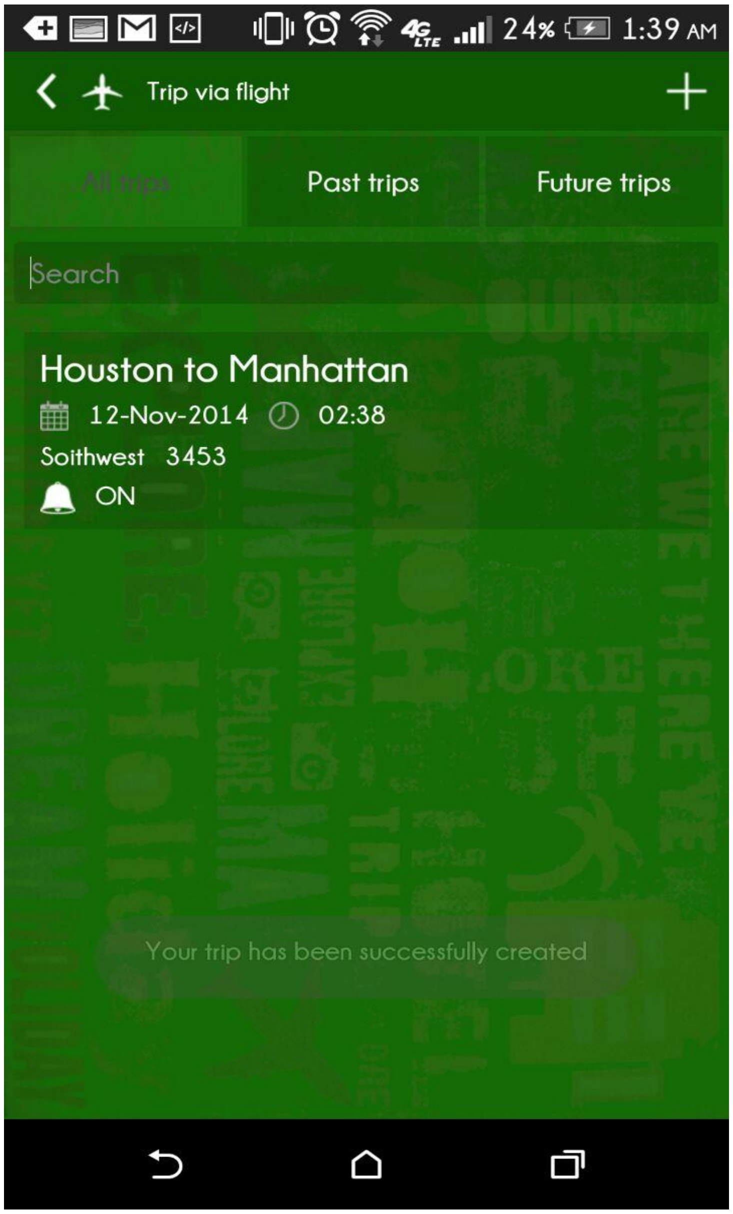


Figure 7.5 Alarm Time Validations

7.6 Toast Messages

Toast messages are added for all required users to communicate with the user. Whenever a trip is saved or alarm is turned on or off, when trips are edited and so on. These messages help communicate with the user and at the same time do not require any input from the user to disable them. They just provide feedback to the user and disappear.



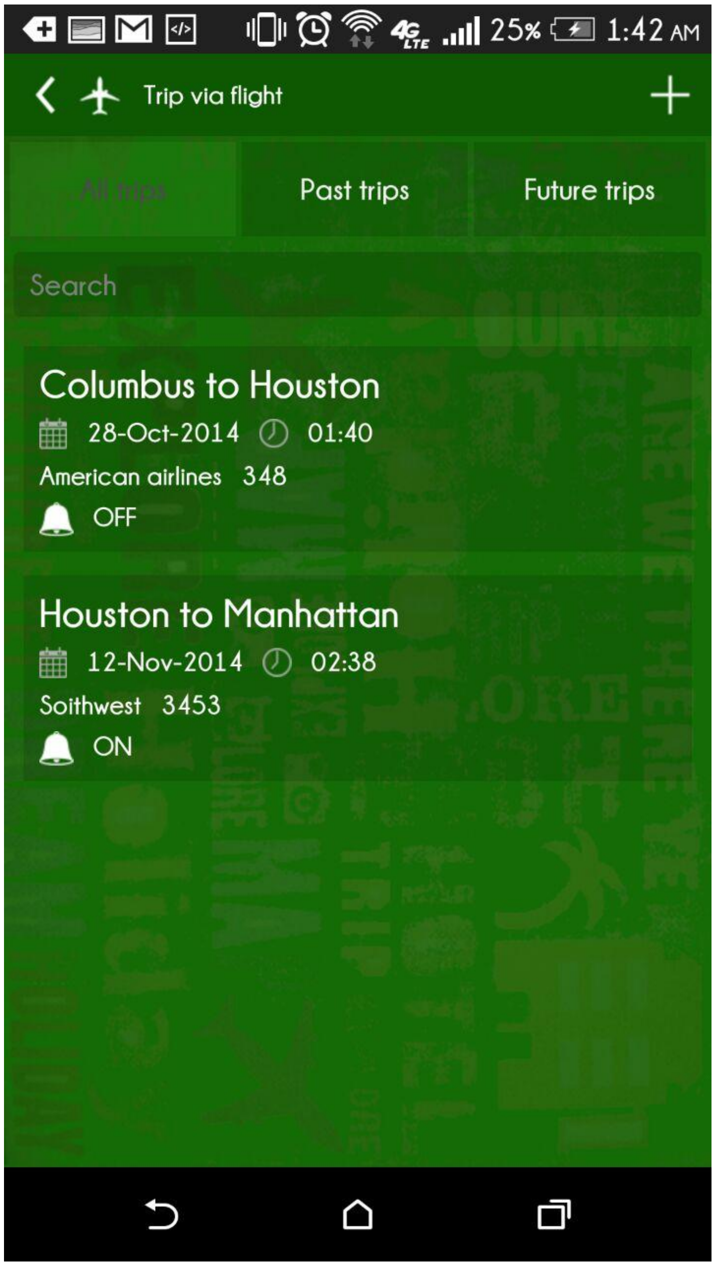
Houston to Manhattan
12-Nov-2014 02:38
Soithwest 3453
ON

Your trip has been successfully created

Figure 7.6 Toast Messages

7.7 All Trips via Flight

The user can is provided with three tabs to view the trips. One of them is All Trips. This tab is present in both Trips via Flight and Trips via Car modules. For each of these modules, the tab All Trips provides the list of all trips both past and future in a view as shown below in the figure.



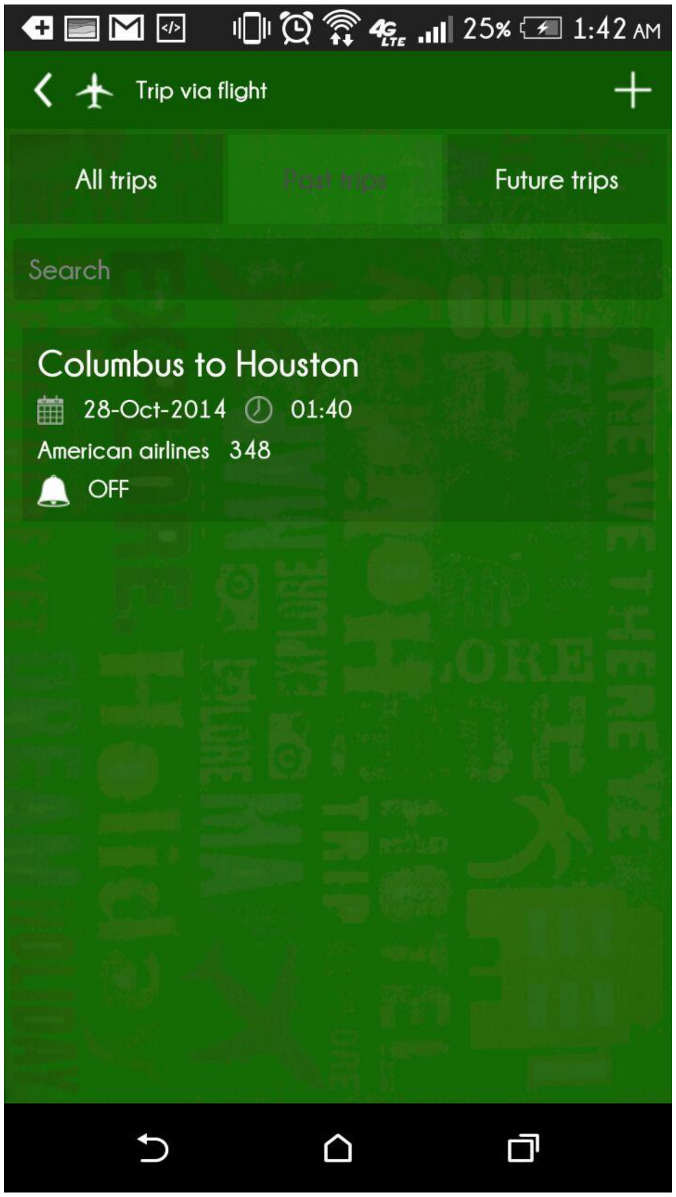
Columbus to Houston
28-Oct-2014 01:40
American airlines 348
OFF

Houston to Manhattan
12-Nov-2014 02:38
Soithwest 3453
ON

Figure 7.7 All Trips

7.8 Past trips via Flight

This is one of the three tabs and it presents the list of all past Trips for each of the module, Trips via Flight and Trips via Car. An overview of the trip is shown which depicts information like the Source, Destination, Trip date and time Airline and Flight Information and whether the alarm is turned on or off. The rest of the details can be seen when the particular trip is selected.

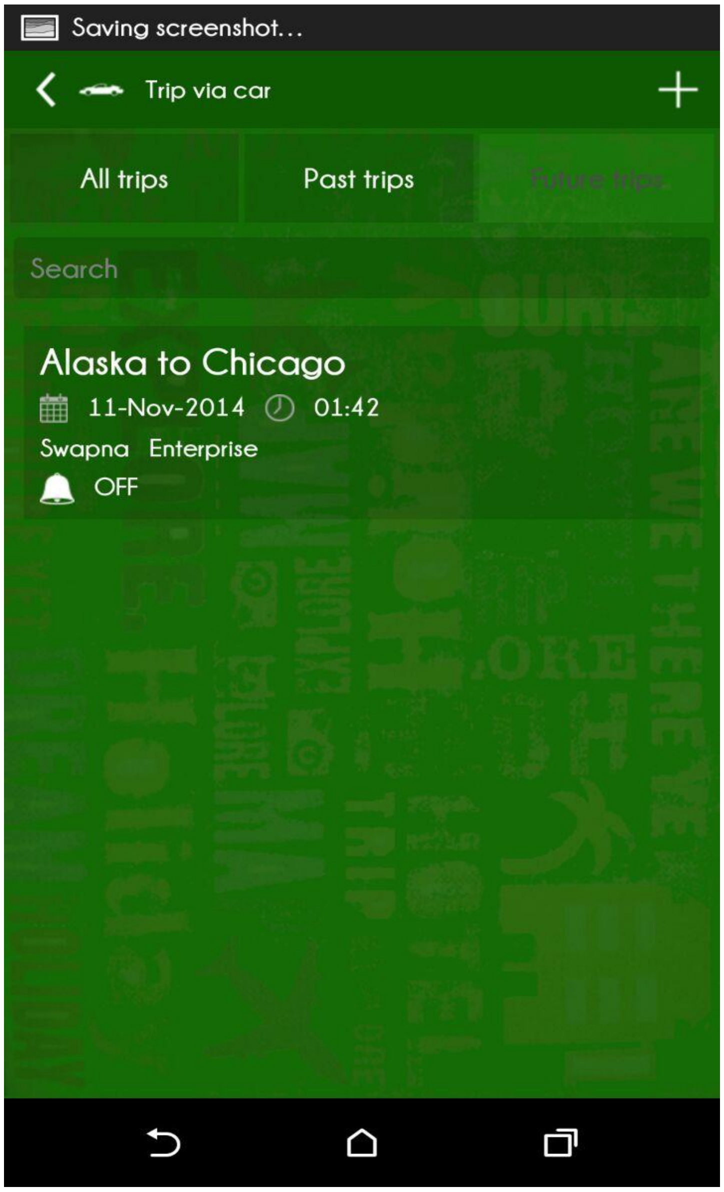


Columbus to Houston
28-Oct-2014 01:40
American airlines 348
OFF

Figure 7.8 Past Trip

7.9 Future Trips via Car

The last tab is Future Trips where the trip details are displayed and this helps us identify all the future trips which are in line and also it displays then in the order on their occurrence. For all the three tabs a search option has been given where the trips can be searched using the source or destination or even the airline name.



Alaska to Chicago
11-Nov-2014 01:42
Swapna Enterprise
OFF

Figure 7.9 Future Trips

7.10 Create a Trip via Car

A user can create a trip via car and the figure below shows all of the fields provided. As a car can be a rented one, an option to rental information is included and options to enter the number of people travelled or the food expenses are added along with the common fields.

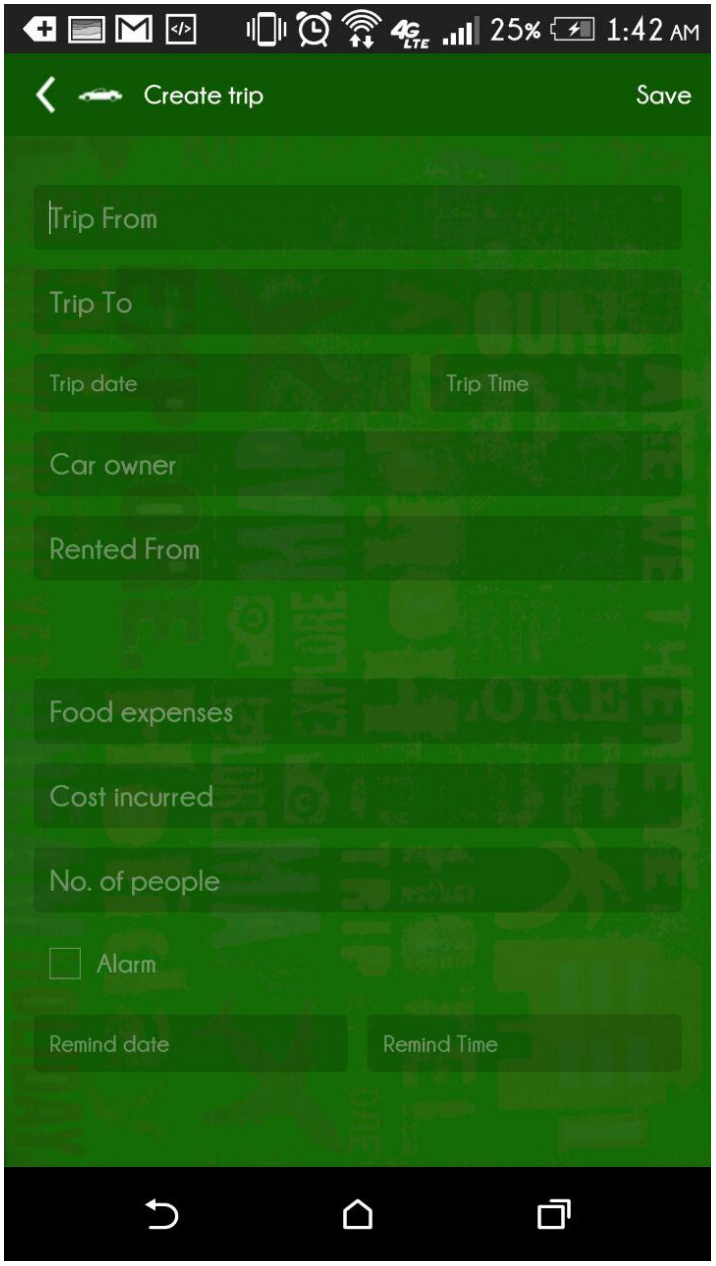


Figure 7.10 Creating Trip via Car

7.11 Enter Expense

To enter expenses for food or price of the flight or the car rental, the user can enter the value and since it shouldn't be a string, the input is limited to numbers.

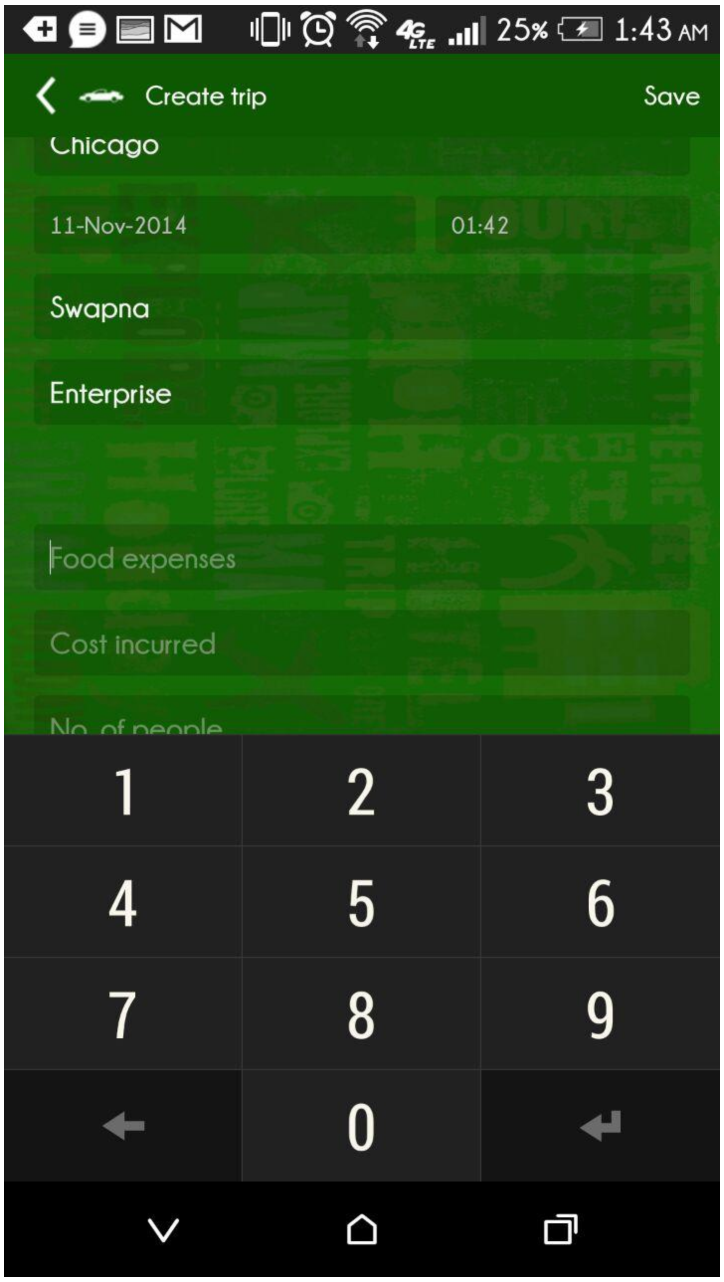
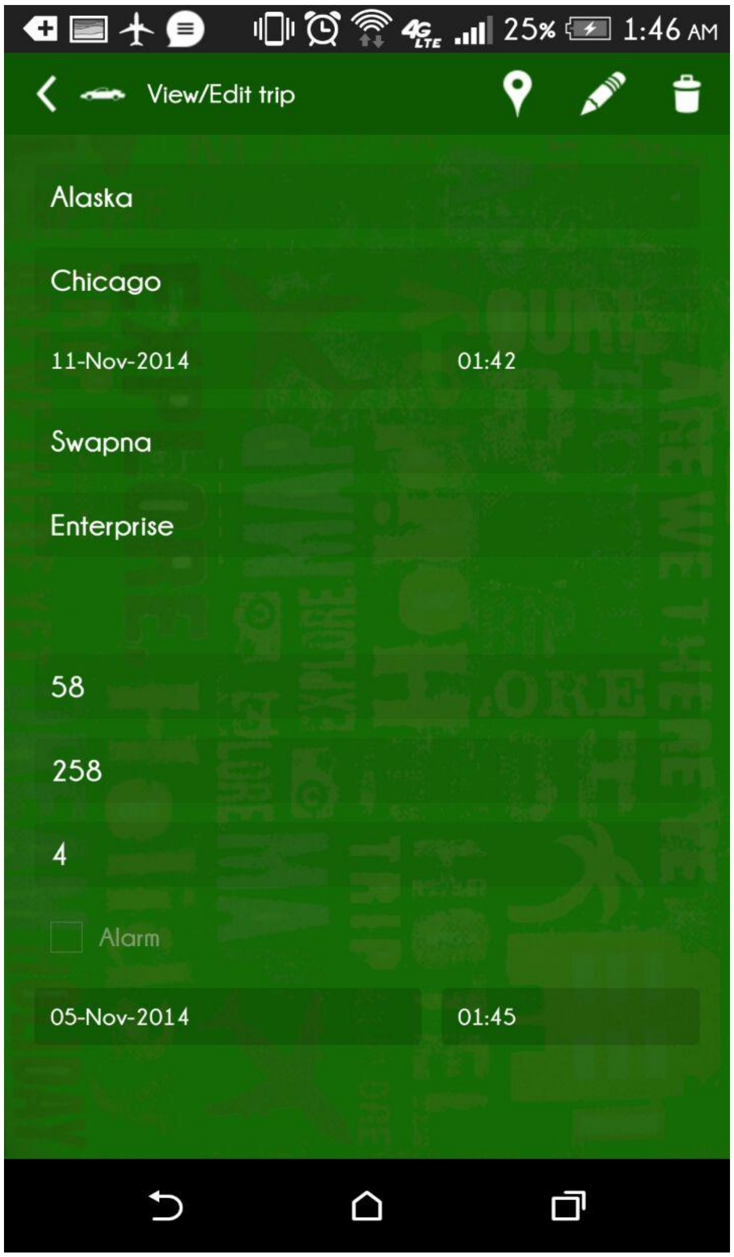


Figure 7.11 Expenses

7.12 View or Edit Trip via Car

For any of the listed trips, the user can view all of the details by selecting them or edit them by selecting the small pen shaped icon and can save back the values which will be updated in the database.



Alaska

Chicago

11-Nov-2014 01:42

Swapna

Enterprise

58

258

4

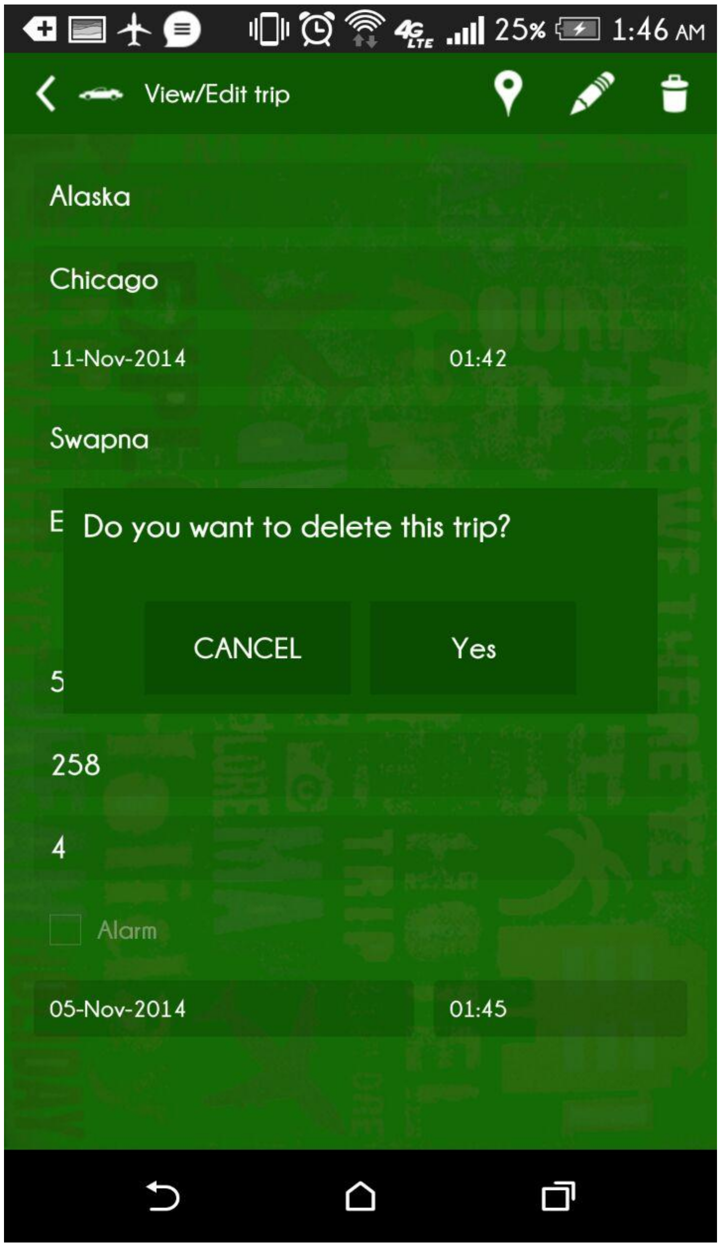
Alarm

05-Nov-2014 01:45

Figure 7.12 View or Edit Trip via Car

7.13 Delete Trip

Any trip can be deleted by choosing the delete (trashcan shaped) icon. Once the icon is selected, it asks for a confirmation if the user wants to delete or cancel. If Yes is chosen, the trip will be deleted else if cancel is chosen, the trip is retained back.



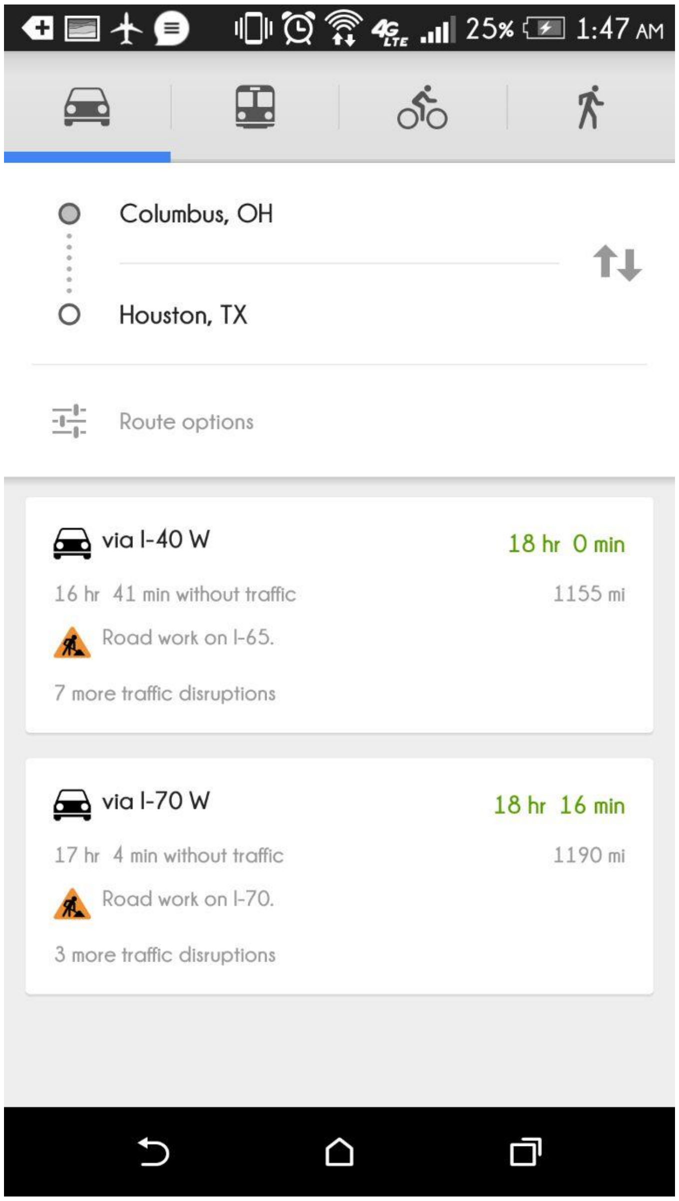
Do you want to delete this trip?

CANCEL Yes

Figure 7.13 Delete Trip

7.14 Navigation

For any trip, let it be Trip via flight or Trip via car, once the user clicks on the trip, he will be presented with a screen which shows all the details of the trip and on the same screen there is a navigation button on the top- right corner, which when clicked provided the google navigation map from the source and destination saved in the trip. The user can retain the same values or modify them for his convenience.



Columbus, OH

Houston, TX

Route options

via I-40 W — 18 hr 0 min
16 hr 41 min without traffic — 1155 mi
Road work on I-65.
7 more traffic disruptions

via I-70 W — 18 hr 16 min
17 hr 4 min without traffic — 1190 mi
Road work on I-70.
3 more traffic disruptions

Figure 7.14 Navigation from Source to Destination Values

Chapter 8 - Testing

Our primary goal behind testing is to find defects. It helps us verify if the system meets all the requirements like functional, reliability, usability and so on. It also helps us validate if the product we built is in accordance with the user requirements. Thus helping us improve and reduce the ambiguity of the product. There are various types of testing. Some of them are performed for the application and are explained below.

8.1 Unit Testing

Unit Testing is the process where we test individual units of the source code are tested to verify if they function as desired. While testing it is taken care that each and every activity and every textarea of the screens are thoroughly unit tested. Also the interaction between various activities and screen loading is tested. The unit tests performed and their results for three modules Trip via Flight, Trip via Car and Upcoming Trips are tabulated below.

8.1.1 Unit Test Cases for Trip via Flight

Table 8.1 Test Cases for Trip via Flight

S.no	Test Case	Expected Result	Result
1	On load Start up screen	Display three tabs Trip via Flight, Trip via car and Upcoming Trips	Pass
2	On click of Trip via Flight	Display the screen with the three tabs all trips, past trips and future trips and other icons to create a trip.	Pass
3	On click of + button on top-right corner	Open the screen with all the fields to fill in for a new trip creation.	Pass
4	On click of date field	Open up the date time picker widget	Pass

5	On type in source, destination and airlines field	If place is already in the database provide auto fill suggestions	Pass
6	On click on save button	Save the trip to database and show the list of trips.	Pass
7	On click of Past Trips button	List all the trips that are from a date lesser than current date	Pass
8	On click of Future trips button	List all the trips that are from a date greater than current date	Pass
9	On click on a listed trip	Display all the trip information.	Pass
10	On click on delete icon on top right corner of the trip	Delete the trip from the view and the database.	Pass
11	On click on edit icon on top right corner of the trip	Open up the trip with editable fields.	Pass
12	On click on navigate icon on top right corner of the trip	Provide google maps navigation for the Source to Destination as entered by the user.	Pass
13	Set an alarm and see if it notifies	Alarm notified at the specified time	Pass

8.1.2 Unit Test cases for Trip via car

For trips via car, we have the same test cases as the ones in Trip via Flight. Any additional test cases are tabulated below.

Table 8.2 Test cases for Trip via Car

S.no	Test Case	Expected Result	Result
1	On click of Trip via Car	Display the screen with the three tabs all trips, past trips and future trips and other icons to create a trip.	Pass
2	On click on Price field	Open the numeric keypad	Pass
3	On click of time field	Open up the date time picker widget with time in 24 hour format.	Pass

8.1.3 Unit Test cases for Upcoming Trips

Table 8.3 Test cases for Upcoming Trips

S.no	Test Case	Expected Result	Result
1	On click of upcoming Trips	List all the future trips form Trip via Car and Trip via Flight.	Pass

8.2 Compatibility Testing

This application has been installed on various devices to test its compatibility. It has been installed on HTC One m8, Google Nexus 4, Samsung Galaxy 4 and Samsung Galaxy Tab 10.1 and the application ran with proper resolution. To provide proper image resolution images are stored with various resolutions in folders hdpi, xhdpi, xxhdpi and so on.

8.3 Usability Testing

Two of my friends have installed MyTripPal application on their phone and tested all the modules and provided with some defects. All of these bugs/defects are corrected in the modified version of the application.

User 1 and User 2 Provided with the following inputs,

- Error checking for the Alarm was missing. User was able to add an alarm for past trip.
- User was able to set an alarm for a past date for a future trip but on a date prior to the tripdate.
- The field destination didn’t have error checking; i.e., it needs to be filled in compulsorily.
- I haven’t had an option to save the website booked on option.
- Upcoming trips had all trips instead of having just the future trips.

All of these have been incorporated and the application has been re-coded to include all these inputs received from Usability testing.

8.4 Performance Testing

Performance testing enables us to know how effective an application is performing. It can be measured in various ways like analyzing the thread execution, checking the battery consumption, reliability etc.

8.4.1 Traceview Analysis

Traceview is a graphical viewing tool which helps us view execution logs that are created by the Debug class to log tracing. The timeline panel describes when each method started and stopped as shown in the figure below. The below table gives the response times of various screens in the application.

Table 8.4.1 Response Time Analysis

Screen Name	**Response Time (ms)**
View All Trips	0.59
View Past Trips	0.57
View Future Trips	0.56
Upcoming Trips	0.61
Create Trip	0.24

Edit Trip	0.28
Trip Via Car	0.33
Trip Via Flight	0.36

8.4.2 Battery Consumption

The battery consumption has been tested using a HTC Onem8 device. The application was tested for the time where the battery percentage started at 100% and reduced to 10 %. In the first case the phone was used to perform normal operations like audio playback, voice calls, WhatsApp texting and voice calls. In the second case along with the normal operations, the MyTripPal application was running in the background. In both the cases, the phone was using Internet with Wi-Fi or 4G/LTE Networks.

Table 8.4.2 Battery Consumption

S.No	Applications Running	Time taken for the battery to reduce to 10%
1	Normal operations like audio playback, voice calls, WhatsApp texting and voice calls	117 minutes
2	MyTripPal Application along with other normal operations	105 minutes

Chapter 9 Conclusion

It is a great learning curve for me since this is my first attempt towards Android development. The application MyTripPal helps a user to track their travel history by providing a user friendly interface to create/view/edit/delete a trip and if the trip is a future-dated trip, the user has the option of setting an alarm if he desired. It also provides the user with the feasibility to get directions to navigate from the source to the destination of the trip by clicking an icon which redirects the user to Google maps.

This application followed the complete Software development life cycle with Analysis followed by Requirements gathering, Implementation which is done using Eclipse with ADT plugin and Testing which is done on real devices. Throughout this process I have learnt Android development and understood its various components and the functionality. It also helped me gain knowledge on the SQLite database and Google Maps API.

Chapter 10 Future Work

The application MyTripPal can be extended to incorporate the following features:

- Unit testing can be done using monkey runner instead of the manual testing. It can also be done using Junit tests.
- The application can be further enhanced and another module which helps tracking trips via Train can be added.
- The application can redirect to travel websites if needed and booking a ticket on one of those sites can automatically save the trip in the application.

Chapter 11 References

[1] "Mobile Devices - An Introduction to the Android Operating Environment", http://www.dhtusa.com/media/AndroidInternals.pdf , Retrieved October 8, 2014.

[2] "Android Boot sequence/ Process", [Online], http://www.kpbird.com/2012/11/in-depth-android-boot-sequence-process.html , Retrieved October 14, 2014.

[3] "Storage Options", [Online], http://developer.android.com/guide/topics/data/data-storage.html , Retrieved October 15, 2014.

[4] "Who's Winning, iOS or Android? All the Numbers, All in One Place", [Online], http://techland.time.com/2013/04/16/ios-vs-android/ , Retrieved October 14, 2014.

[5] "UML Use Case Diagrams", [Online], https://www.andrew.cmu.edu/course/90-754/umlucdfaq.html , Retrieved October 26, 2014.

[6] "Use Case", [Online], http://en.wikipedia.org/wiki/Use_case , Retrieved October 26, 2014.

[7] "Class Diagram", [Online], http://en.wikipedia.org/wiki/Class_diagram , Retrieved October 27, 2014.

[8] "To Draw Use Case Diagrams", [Online], https://creately.com/app/# , Retrieved October 29, 2014.

[9] "App Manifest", [Online], http://developer.android.com/guide/topics/manifest/manifest-intro.html , Retrieved October 29, 2014

[10] "Activity", [Online], http://developer.android.com/reference/android/app/Activity.html#Fragments , Retrieved

November 3, 2014

[11] “Intent”, [Online].

http://developer.android.com/reference/android/content/Intent.html , Retrieved

November 5, 2014

[12] “Layout Inflater”, [Online],

http://developer.android.com/reference/android/view/LayoutInflater.html , Retrieved

November 4, 2014

[13] “Sequence Diagrams”, [Online],

http://en.wikipedia.org/wiki/Sequence_diagram , Retrieved November 20, 2014.