VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Wubshet Solomon

Addis – City Guide Android Mobile Application

Technology and Telecommunication

2016

ACKNOWLEDGEMENTS

I would like to express my very sincere gratitude to my supervisor, Dr.Ghodrat Moghadampour, for his encouragement and support from the beginning to the end of this final project work.

I am so thankful to VAMK authorities for making it possible for me to study here and to all my instructors during my studies at VAMK.

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Degree Program in Information Technology

ABSTRACT

Author	Wubshet Solomon
Title	Addis – A City Guide Mobile Application
Year	2016
Language	English
Pages	48
Name of Supervisor	Dr.Ghodrat Mohgadampour

The rapid growth of tourism industry in Addis Ababa, the capital city of Ethiopia, attracts many tourist and today the country becoming one of the tourist destinations in Africa. In addition the city is the seat for African Union headquarters and home for many international organizations. Due to these facts of the city it is obvious to have a means to deliver enough information to the visitors using smart phone.

The main purpose of the project was to design a city guide mobile application under the Android platform which helps the user to navigate places within the city, assists in searching for hotels and historical places easily. In addition, the application provides the current currency exchange rate of Birr, the Ethiopian currency, in terms of international currency and also allowed easy reservation of hotels by converting the web view to a mobile application.

The project was designed and implemented by considering basic requirements for a city guide application showing a map, locating places with customized and clustered markers on a map and each marker representing a specific coordinate which is a combination of Latitude and Longitude values. In this project these values were saved in JSON file along with name of the location and accessed by the application from a server to plot a marker on Google Maps.

Overall, this project work was good exposure to understanding how Android mobile applications are developed to solve existing problems.

TABLE OF CONTENTS

LIST OF ABBREVIATION

API	Application Programming Interface
ADT	Android Development Tools
APK	Android Application Package
AVD	Android Virtual Device
JSON	JavaScript Object Notation
GUI	Graphical User Interface
JVM	Java Virtual Machine
SDK	Software Development Kit
XML	Extensible Markup Language
OS	Operating System
URL	Uniform resource Locator
HTTP	Hypertext Transfer Protocol
VAMK	Vassan ammattikorkeakoulu
UI	User Interface
IDE	Integrated Development Environment
Lat	Latitude
Lng	Longitude
HTML	Hypertext Markup Language
APP	Application Program
IP	Internet Protocol

JAVA EE	Java Platform Enterprise Edition
OOP	Object Oriented Programming

LIST OF FIGURE

1 INTRODUCTION

The rapid growth of economy and the movement of people in Addis Ababa, the capital city of Ethiopia show tremendous activity in the tourism industry. This attracts many tourists and today the country is becoming one of the tourist destination sites in Africa. In addition the city hosts many international meetings, festivals and cultural events that are registered by UNESCO. So, it is obviously of use to have a smart means to deliver enough information to visitors using what they have on their hand anyway, a smart phone. Moreover, it is better that all the service involved in the tourism industry are integrated and put on the same application, including hotels services , historical places or museum information ,banks and shopping locations etc.

1.1 Background of the topics

Currently there are many tour guide operators working in the city and almost more than half percent of the visitors who come to the city, use these operators also for the hotel reservation conducted through the hotel website or by contacting the reception. In addition there are some mobile applications designed for the city guide purpose by individuals and software companies. However all of these applications focus on listing the service providers and there are no instruction of how to get to the location of these places. The Google map is also included with the markers, however, the markers are not presented in a clustered format and this makes the map difficult to use for locating to a particular place.

The motivation of this project is to add some new feature in to the existing application. Moreover, the project explores Android technologies and puts them as a building component like Google Maps.

1.2 Objectives

Based on the background of the project, it has various objectives. The major objectives are listed below:

- Enable users to navigate places within the city easily, for example to hotels, tourist sites, banks and shopping places.
- Provide list of service providers with their service .In case of hotels, with a short description and the address.
- Use and customize Google Maps with cluster markers which used as the main features in the implementation of the project.
- Provide current currency rate exchange.
- Make it easy to reserve a hotel room.
- Save time and effort in searching for hotels and museums.

2 RELEVANT TECHNOLOGY

This section describes, technologies that are used to build this application and review its architectural structure.

2.1 Android Structure

Android is an open source and Linux-based Operating System for mobile devices such as smartphones and tablet computers. Android was developed by the Open Handset Alliance, led by Google, and other companies.

Android offers a unified approach to application development for mobile devices which means developers need only develop for Android, and their applications should be able to run on different devices powered by Android./2/

Android have many components which work with different APIs that are provided by Android SDK. These APIs are source code used as an interface in application development. The main component of Android is explained in the following.

2.1.1 Activities

An Android activity is one screen of the Android that represents user interface. Android activity is in many ways similar to Windows in a desktop application. An Android app may contain one activity or more, meaning one or more screens. The main activity is the launcher of an Android application and it triggers other activity.

An Android activity extends from Activity class and it has different methods to define the state of activity. The methods **onCreate, onStart** and **onResume** called when the main activity is created .When a new activity opened, the method **onPause** is called and the newly opened one will be active and the previous will be paused.When going back to the previous activity the method o**nResume** is called. During the exit of the application the method **onDestroy** will be called. The relationship of these methods is shown in Figure 1.

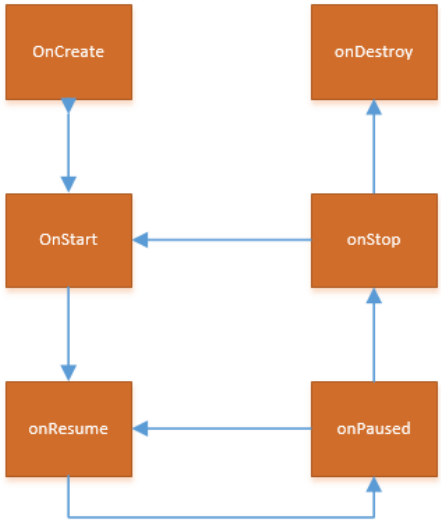

Figure 1 Activity Lifecycle

2.1.2 XML Layout

The Android xml layout is used to define each element of the user interface which is displayed in the activity and instantiate layout element at runtime. In this project the following layouts are used

- LinearLayout
- RelativeLayout
- TableLayout

2.1.3 Android Manifest File

The AndroidManifest.xml file defines the whole hierarchical structure of the application and it is used to declare permissions the application must have in order to access protected parts of API and interact with other application like internet access, or PS tracker. It also contains lists of classes that provide profiling and other information as the application is running.

2.1.4 Google Maps APIS

The Google Maps API is used to customize and embed a Map within the application. This API has a unique key which is stored in AndroidManifest.xml file and is used to access the Google Maps server.

2.1.5 MarkerClusterer

The MarkerClusterer is a client side utility library that applies grid-based clustering to a collection of markers. It works by iterating though the markers in the collection that you wish to cluster adding each one into the closest cluster if it is within in a minimum square pixel bounds./12/.Figure 2 demonstrates how the marker look before and after clustering.

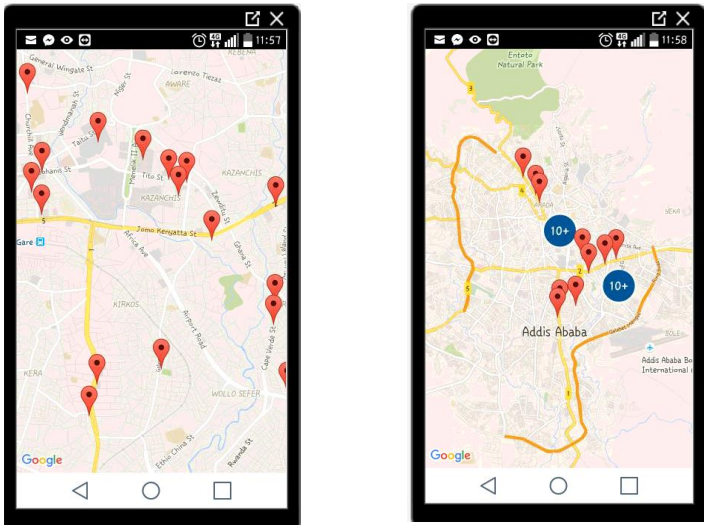

Figure 2 Before and After Clustering

2.1.6 JSON

JSON, which stands for JavaScript object notation is a readable format for structuring data. It is used to send and retrieve data from an application to a server. It is like a

human readable language format that is easy to understand. In this application the JSON file is stored on a server and the application retrieves data from the file and after parsing the application used parsed data for map customization and adding hotel locations.

2.2 External Library

The application uses external libraries to access a web service .The first one is a httpclient and it is used to transmit and deceive HTTP messages. The second one is httpcore and it is used to build custom client and server side http service. The third library is httpmime and it is responsible for loading and processing the setting of a related application or window function./13/

3 APPLICATION DESCRIPTION

This chapter presents briefly the project requirements in terms of how the users communicate with the system and define specific features and expectations. It also describes the behavior of the application.

3.1 Quality Function Deployment

The quality function deployment identify the priority needs and expectations from this project. It also describe the interactions between the system and its environment independent of its implementation.

According to functionalities and priority level the requirement are categorized into three groups.

3.1.1 Normal requirements (must-haves)

These are the least requirements the application performs and they are:

- Have a map with customized marker with their location name.
- List all the available hotels starting from five star rating.
- Make it easy to navigate to places within the city.
- Make easy to reserve hotels services.
- Display the current currency exchange rate.

3.1.2 Expected requirements (should-haves)

These are the requirements that are important features on the application.

- To avoid an overload of markers on map the application implements in marker clustering.
- The application must have user friendly interface.
- Descriptions and physical addresses of hotels and museums are included in the application.

3.1.2 Optional Requirements (Nice to Have)

These are the extra requirements which make the application to have more features.

- Markers in the map are represented by a unique picture for a better representation of their target.
- The application GUI implement animation

3.2 Use case Diagram

A use case diagram is a graphic presentation of the system interaction with external environment and the relationship between the main functionalities of the system.

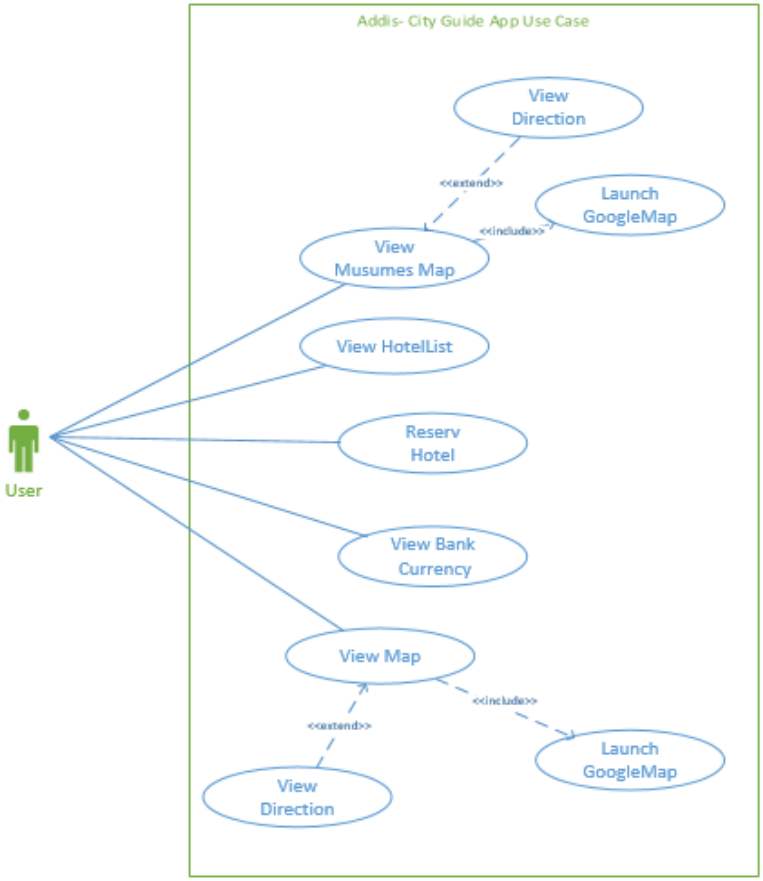

Figure 3 Use Case Diagram

3.2.1 View Museums Map Use case

Textual description for the use case “View Museums Map”

1. Use case name: View Museums Map
2. Participating actor: users
3. Entry condition: click or tap the button Museums.
4. Event flow

    4.1 zoom out the clustered marker

    4.2 click or tap the customized Marker

    4.3 Click the direction icon in the map bottom apart

5. Exit condition: - Click the back arrow in the bottom part

3.2.2 View Hotel List Use case

Textual description for the use case “View Hotel List”

1. Use case name: View Hotel List
2. Participating actor: users
3. Entry condition: click or tap the button Hotel List.
4. Event flow

    4.1 The user select the hotel from the list

    4.2 Click the hotel from the list

5. Exit condition: - Click the back arrow in the bottom part

3.2.3 Reserve Hotel Use case

Textual description for the use case “Reserve Hotel”

1. Use case name: View Reserve Hotel
2. Participating actor: users
3. Entry condition: click or tap the button Hotel List.
4. Event flow

    4.1 The user select the hotel from the list

4.2 Click the hotel from the list

4.3 click the reserve button

5. Exit condition: - Click the back arrow in the bottom part

3.2.4 View Bank Currency Use case

Textual description for the use case “View Bank Currency”

1. Use case name: View Bank Currency
2. Participating actor: users
3. Entry condition: click or tap the button Bank.
4. Event flow

   4.1 Click the button Bank

   4.2 view the current currency rate of Birr in terms of the other international currency

5. Exit condition: - Click the back arrow in the bottom part

3.2.5 View City Map Use case

Textual description for the use case “View City Map”

1. Use case name: View city Map
2. Participating actor: users
3. Entry condition: click or tap the button Map
4. Event flow

   4.1 zoom out the clustered marker

   4.2 click or tap the customized Marker

   4.3 Click the direction icon in the map bottom apart

5. Exit condition: - Click the back arrow in the bottom part

3.3 Sequence Diagram

The sequence Diagram models the relationship of objects based on a time sequence. It shows how the objects interact with others in a particular scenario of a use case with advanced visual modeling capability./14/

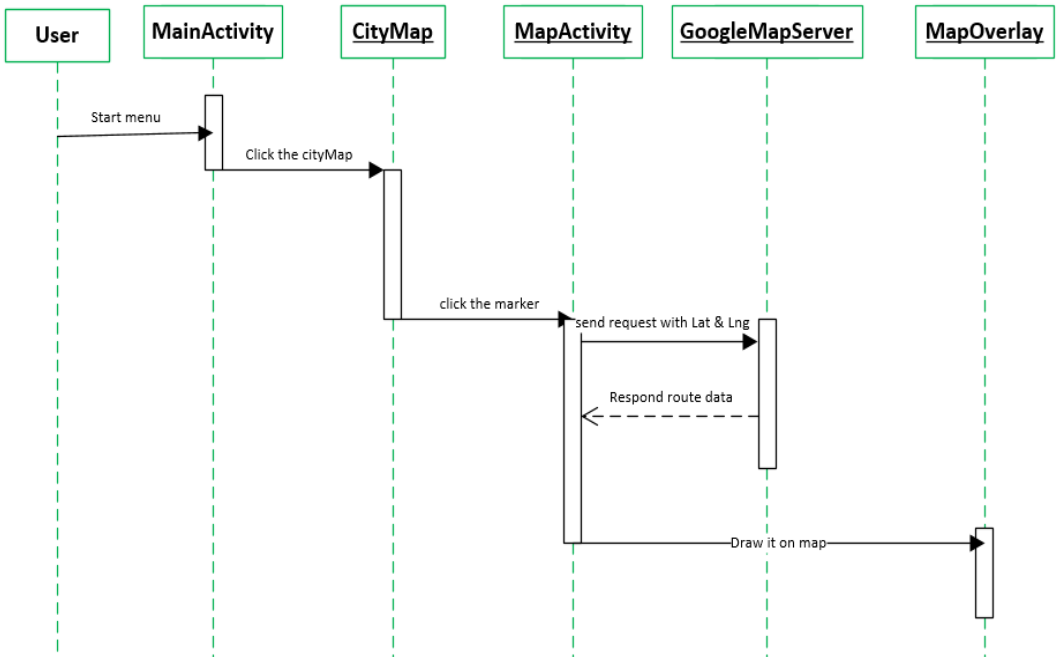

Figure 4 City Map Sequence Diagram

The above sequence diagram in Figure 4 shows the function flow of the city map functionality. The user gets the city map button from the menu list by clicking on the marker. The selected request is send to MapActivity, which then sends it to Google Map Server with latitude and longitude

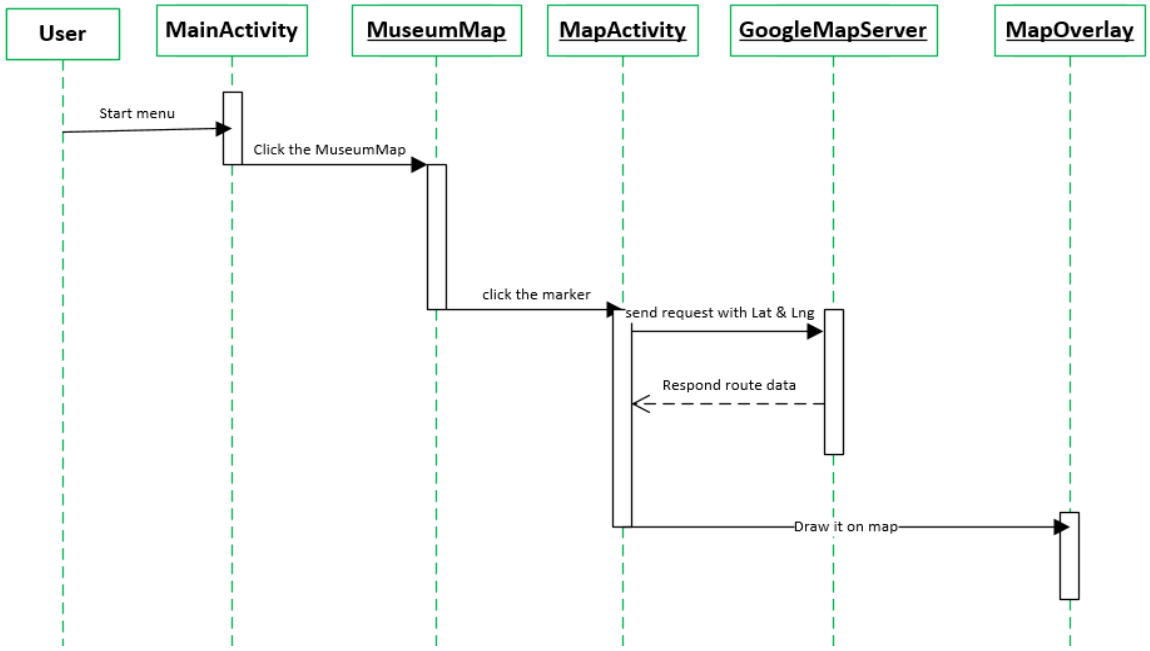

Figure 5 Museum Map Sequence Diagram

Figure 5 shows the function flow of the museum map functionality. The user gets the museum map button from the menu list and by clicking on the marker. The selected request is sent to MapActivity, which then sends it to Google Map Server with latitude and longitude.

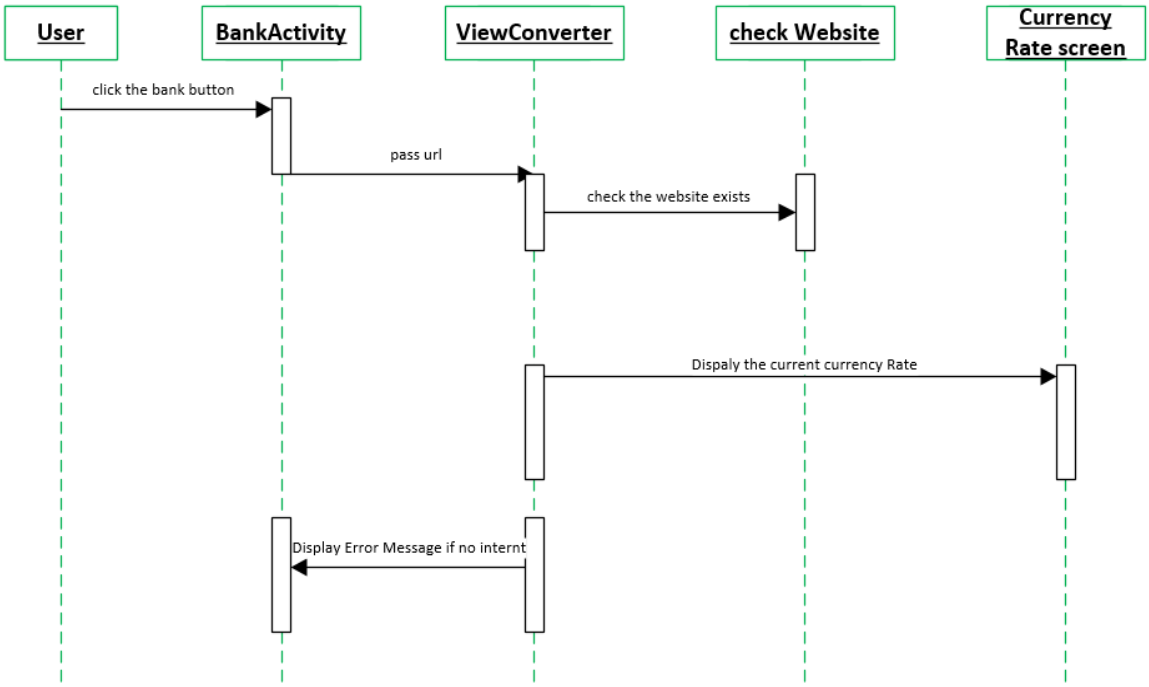

Figure 6 Rate exchange Sequence Diagram

Figure 6 shows the function flow of the rate exchange functionality. The user clicks Bank button from the menu list. The selected request is sent to viewconverter, finally the current exchange rate will display to user.

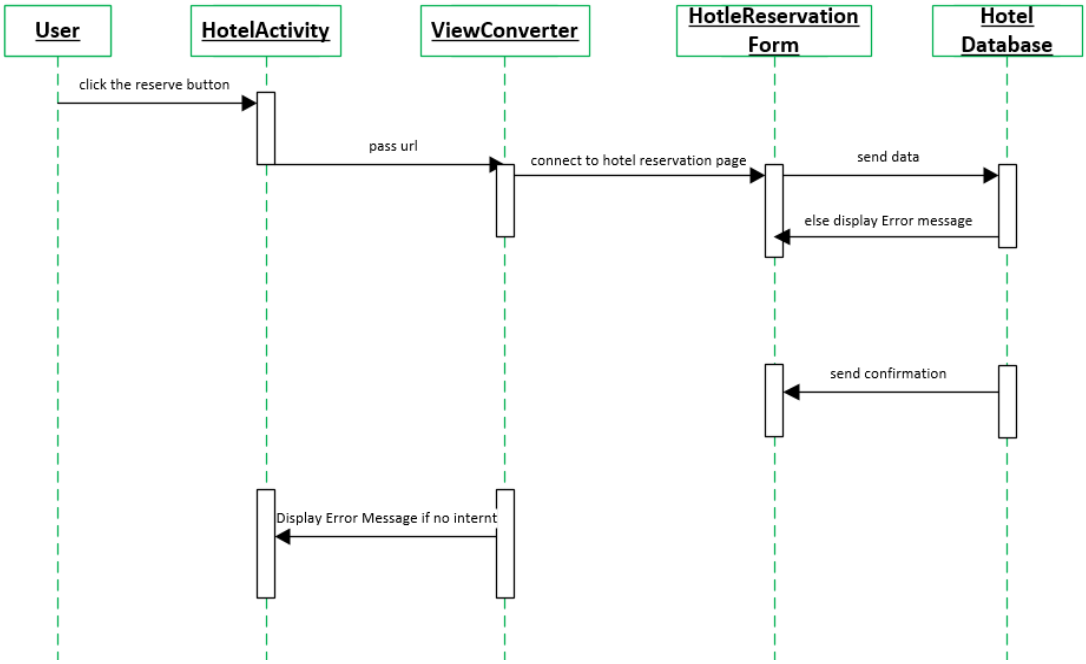

Figure 7 Hotel Reservation Sequence Diagram

Figure 7 shows the function flow of how the webview is converted to mobile app. The user clicks reserve button from the menu list. The selected request is sent to viewconverter. Finally the reservation form will display to user.

3.4 Class Diagram

A class diagram is a representation of the relationships and source code dependencies among classes in the Unified Modeling Language.

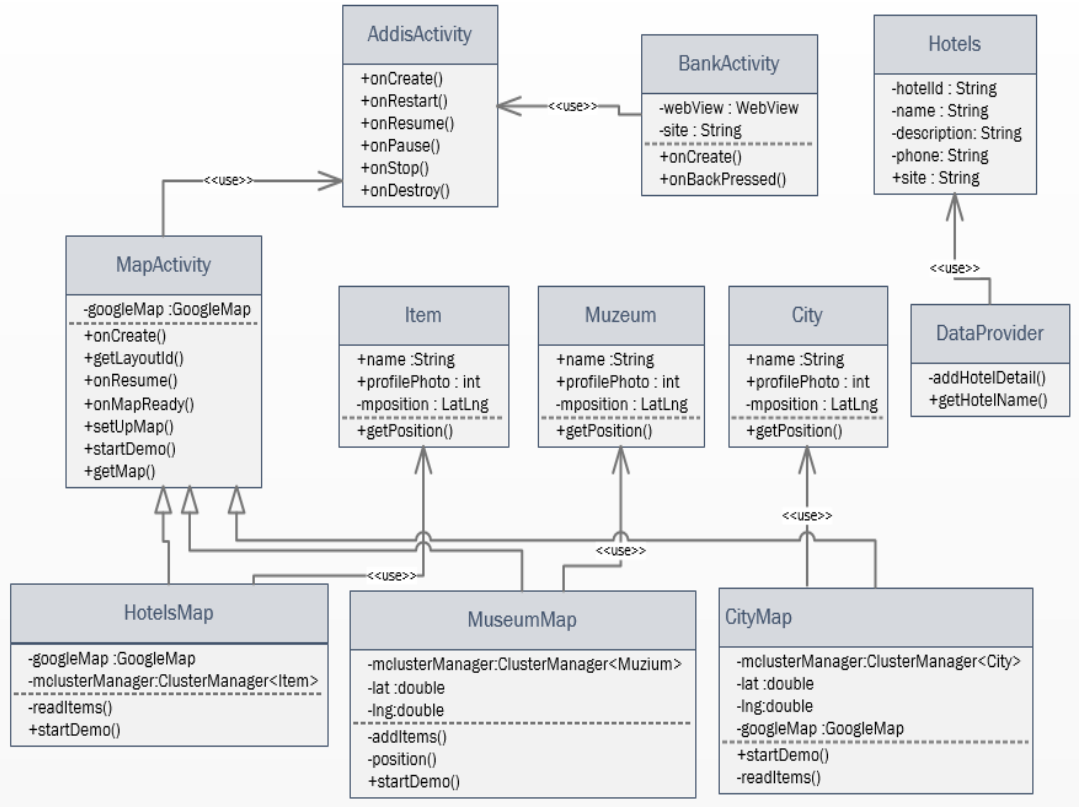

Figure 8 Class Diagram

Figure 8 above presents the class diagram of this project, AddisActivity is the launcher of the application and the two activities, MapActivity and BankActivity use the MainActivity, which means these activities can be accessed through AddisActivity. The three classes HotelsMap, MuseumMap and CityMap extend to MapActivity .The purpose of these classes are to read the JSON file, which contains latitude, longitude and location name from the server and place a marker on the map using the clustering utility.

4. Database and GUI design

4.1 JSON File structure

The application uses JSON file for retrieving data which stored on a server. The Figure 9 shows the structure of JSON file used in this project.

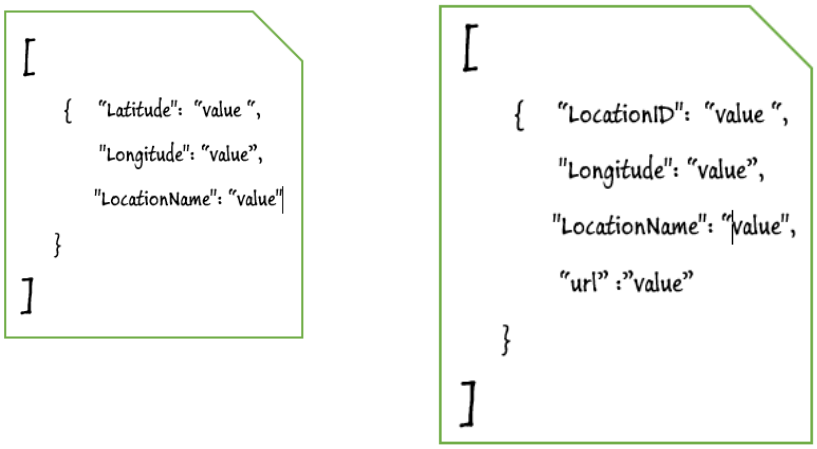

Figure 9 JSON File Structure

4.2 GUI design

The GUI for this application is built using a collection of activities which are the basic unit of an Android application. All the components of the GUI are defined in XML file and this file is loaded during a runtime.

4.2.1 Launch Icon

The application is launched by clicking or tapping the icon showed in the red colored rectangle and the icon picture is selected intentionally to represent the city.

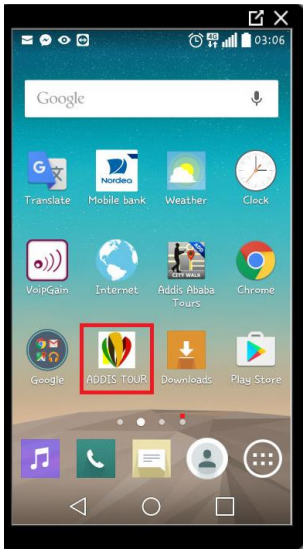

Figure 10 Launcher Icon Page

4.2.2 Main Menu

The menu page is displayed after the launch icon is clicked and it contains six buttons which are used to trigger the core functionalities of the application. The Figure 11 shows the main menu page.

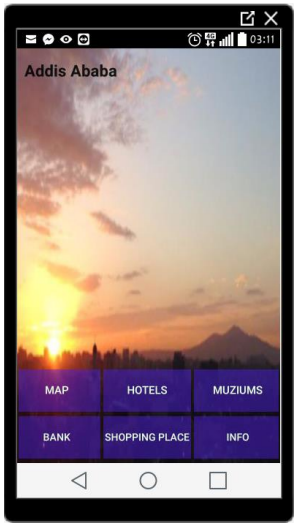

Figure 11 Menu Page

4.2.3 City Map Page

The City Map displayed by clicking the button Map from the menu. It shows all the available hotels in the city rating from three stars to five and all the available tourist sites with customized markers.

The markers are presented in a grid-based or a distance-based collection called clustering, which is used to avoid marker overload and it bring more insights to the locations. By zooming in the cluster, the markers can be easily detached.

To navigate to a hotel or a museum ,first select the marker from the map and the location name will be displayed automatically.Then the arrow will appear at the bottom,this leads to Google Maps server.Finally, the map with a route connecting marker location and the current user position appeares. Figure 12 shows the City Map page.

Figure 12 City Map

Figure 13 shows the sample navigation of the hotel Capital & Spa

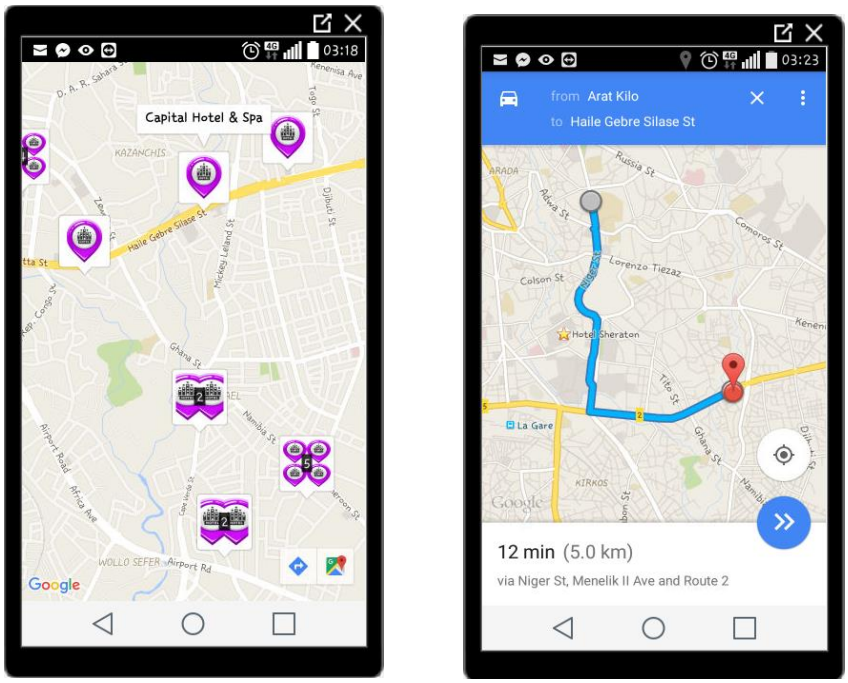

Figure 13 City Navigation Sample page

4.2.4 Hotel List

The hotel list shows the available hotels according to their rating rank starting from five stars to four stars. When the user selects one of the hotel from the scrollable list, the detail page will appear. This page contains a short text description about the hotel. The page also contains two buttons .The first button is for triggering a map of the selected hotel with a customized marker and a name. The second button is to launch the reservations form, which is converted to an application view from the web view for the selected hotel.

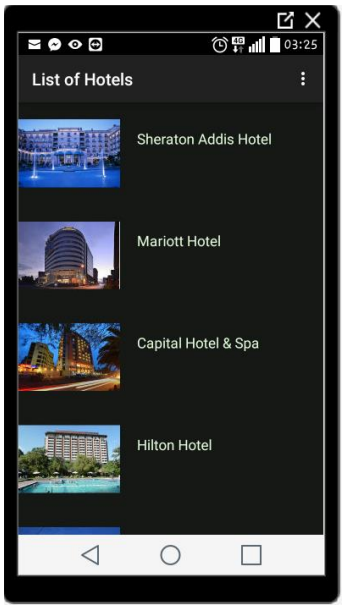

Figure 14 Hotel List

Figure 15 shows the details about the selected hotel from the list and the user can access two other pages by using the two buttons that are Map and Reserve. For the sample purpose here the Capital Hotel & spa are selected and both the detail page and a map is showed in Figure 15.

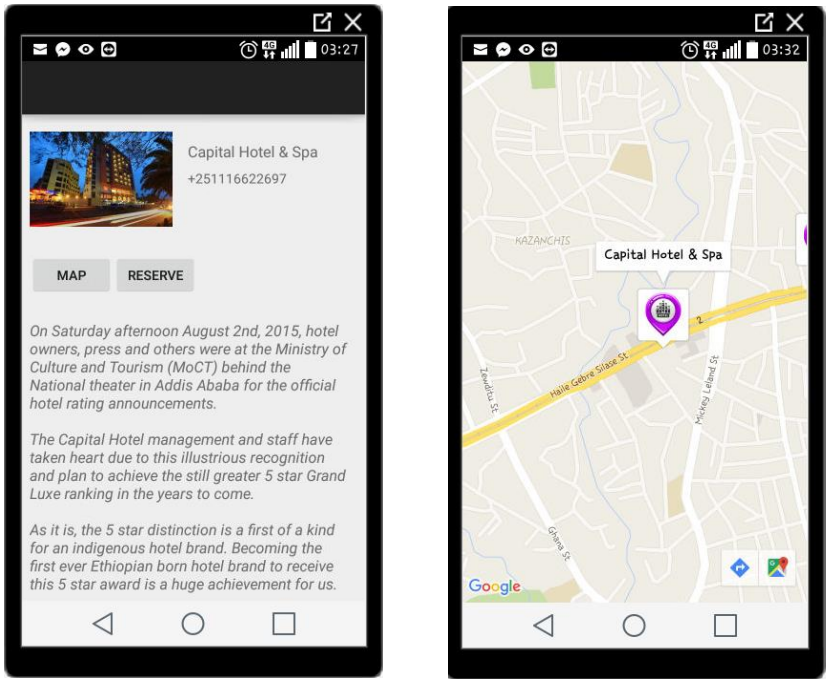

Figure 15 Hotel Detail Page

4.2.5 Museum Map

The museum Map page displays a map which uses a cluster for the customized and organized markers. All the tourist sites including statues and museums are included in this map. In addition, the markers are designed by the image of the respected location.

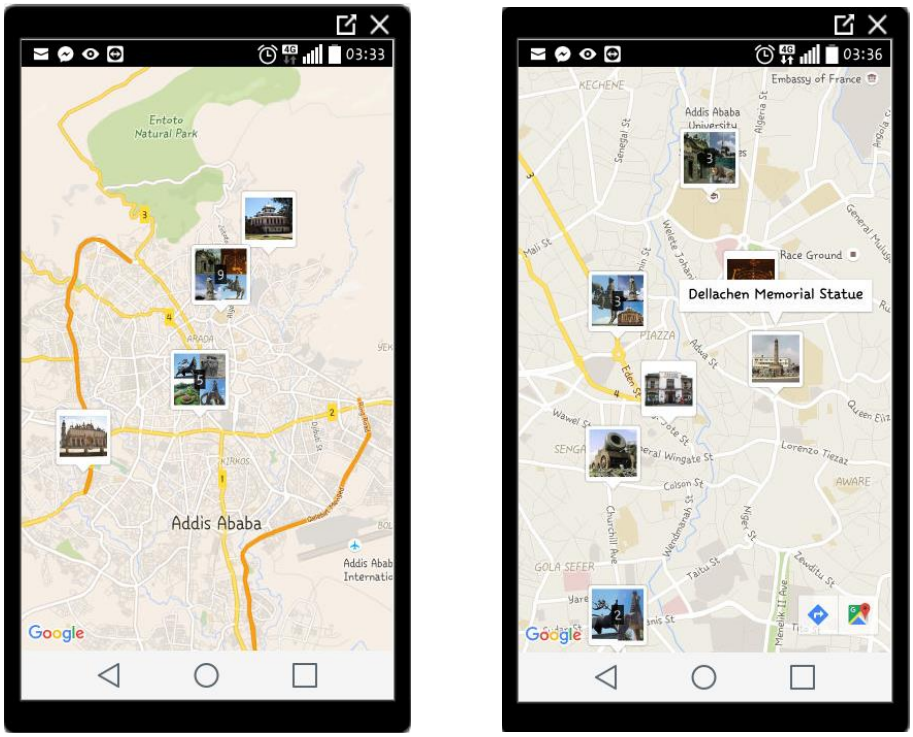

Figure 16 Museum Map

After selecting a specific tourism site and clicking the pictured marker, it will be easy to navigate that specific place from the current position. Figure 16 shows the museum pages.

4.2.6 Bank Page

The Bank page is accessed from the main menu by clicking the Bank button. The page displays the current currency exchange rate of Ethiopian currency called Birr relatively to USD, EURO, GBP, CAD, CHF, AED, SAR and JPY. An image of the display is shown in Figure 17.

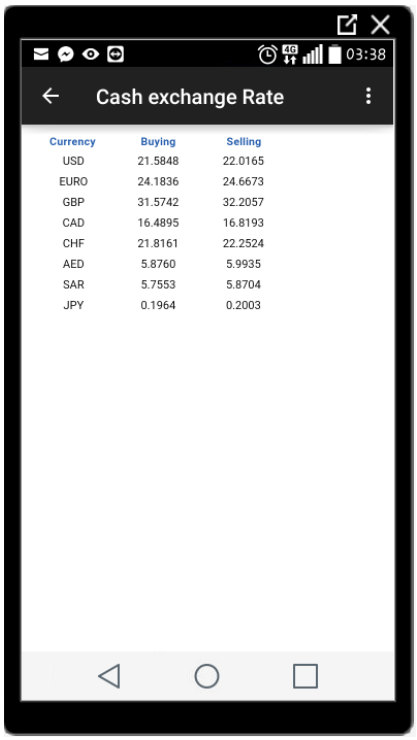

Figure 17 Rate Exchange Page

5. IMPLEMENTATION

The application is implemented using an object oriented programming model using the concept of OOP. This section shows how the project is implemented.

5.1 Main Activity

The MainActivity organizes all the other activities within the application and it is also a launcher. In this application case the main activity is called AddisActivity. The activities which use this MainActivity are defined in AndroidManifest xml file. In addition, the GUI part is defined in activity_addis xml file. The relation between activities is shown in Code snippet 1.

```
<activity
        android:name=".AddisActivity"
        android:label="ADDIS TOUR" >
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity android:name=".HotelsMap"
            android:label="Addis Ababa" > </activity>
    <activity android:name=".HotelsList"
            android:label="List of Hotels" />
    <activity android:name=".CityMapActivity"
            android:label="" > </activity>
    <activity android:name=".MuziumActivity"
            android:label="" />
    <activity android:name=".HotelDetail"
            android:label="@string/title_activity_hotel_detail" />
    <activity android:name=".Site_converter"
```

```
            android:label="" />
        <activity android:name=".BankActivity"
            android:label="Cash exchange Rate "
            android:parentActivityName=".AddisActivity" >
          <meta-data
            android:name="android.support.PARENT_ACTIVITY"
            android:value="com.example.wubshet.sample.AddisActivity" />
        </activity>
```

Code Snippet 1 AndroidManifest xml

In addition, the GUI part is defined in activity_addis xml file. These relation is shown in the following code snippets.

```
<TableLayout
    android:layout_width="match_parent"
    android::layout_height="wrap_content"
    android:layout_alignParentEnd="true"
    android:layout_alignParentBottom="true"
    android:layout_alignParentStart="true"
    android:layout_marginRight="2dp"
    android:layout_marginLeft="2dp"
    android:id="@+id/tableLayout">
    <TableRow
      android:layout_width="match_parent"
      android:layout_height="60dp"
       android:contextClickable="false"
       android:gravity="center">
       <Button
          android:layout_width="120dp"
          android:layout_height="60dp"
          android:id="@+id/imageButton"
          android:text="Map"
```

```
        android:onClick="showMap"
        android:textSize="15dp"
        android:background="#6f5722f8"
        android:layout_marginRight="4dp"
        android:layout_marginBottom="4dp"
        android:textColor="#e5ffde"
         />
    <Button
        android:layout_width="120dp"
        android:layout_height="60dp"
        android:id="@+id/imageButton2"
       android:text="Hotels"
        android:onClick="setHotelList"
        android:background="#6f5722f8"
        android:layout_marginRight="4dp"
        android:layout_marginBottom="4dp"
        android:textColor="#e5ffde"
        android:textSize="15dp" />
    <Button
        android:layout_width="120dp"
        android:layout_height="60dp"
        android:id="@+id/imageButton3"
        android:text="Muziums"
       android:onClick="showMuzium"
        android:background="#6f5722f8"
        android:layout_marginRight="4dp"
        android:layout_marginBottom="4dp"
        android:textColor="#e5ffde"
        android:textSize="15dp" />
  </TableRow>
```

Code Snippet 2 Main Activity XML

The class AddisActivity defines the functions which trigger the other activities and the code is shown in the following code snippet.

```
public void setHotelList(View view)
{
    Intent intent = new Intent(this,HotelsList.class);
    startActivity(intent);
}

public void showMap(View view) {

    Intent intent = new Intent(this,MuziumActivity.class);
    startActivity(intent);
}
public void showMuzium(View view) {

    Intent intent = new Intent(this,CityMapActivity.class);
    startActivity(intent);
}

public void showBank(View view) {

    Intent intent = new Intent(this,BankActivity.class);

 startActivity(intent);
}
```

Code Snippet 3 Declaring buttons event

5.2 MapActivity class

This class defines the content of the main Map. It derived from android.app.Activitiy and it handles management of the mapEngine. The following Code snippet shows what function is defined to display and manipulate Map.

```
@Override
 protected void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(getLayoutId());
     setUpMap();
 }

 @Override
 protected void onResume() {
```

```
    super.onResume();
    setUpMap();
}

@Override
public void onMapReady(GoogleMap map) {
    if (myMap != null) {
        return;
    }
    myMap = map;
   startMap();
}

private void setUpMap() {
    ((SupportMapFragment)
getSupportFragmentManager().findFragmentById(R.id.googleMap)).getMapAsync(this);
 }
 protected abstract void startMap();

 protected GoogleMap getMap() {
    return myMap;
 }
```

Code Snippet 4 Map Activity Definition

5.3 CityMap class

This class determine what to display on city map. It also defines a function which reads data from the JSON file and puts markers according to reading the latitude, longitude and location name from the file.

The following code snippet shows how to grab the contents accessed by the URL using HttpClient and create a stream, then read the data from the file. The file is located on a server.

```
public static String getHttp(String server_url) {

    StringBuilder strbuilder = new StringBuilder();

    HttpClient client = new DefaultHttpClient();

    HttpGet httpGet = new HttpGet(server_url);

    try {
```

```
        HttpResponse response = client.execute(httpGet);
        StatusLine statusLine = response.getStatusLine();
        int statusCode = statusLine.getStatusCode();
           HttpEntity entity = response.getEntity();
           InputStream contents = entity.getContent();
           BufferedReader reader = new BufferedReader(new InputStreamReader(contents));
           String line;
           while ((line = reader.readLine()) != null) {
              str.append(line);
           }
        } else {
           Log.e("Log", "can't download result..");
        }
     } catch (ClientProtocolException e) {
        e.printStackTrace();
     } catch (IOException e) {
        e.printStackTrace();
     }
     return str.toString();
}
```

Code Snippet 5 connecting and Reading Data from Server

The function startMap () defines the zoom level of Google MapS to the specified bounds for the city of Addis Ababa and organizes loading the JSON file information from the given URL placed at “http://www.cc.puv.fi/~e0900775/output.json”. The data

is retrieved from URL, added to HashMap collection type object and parsed, then converted to a string. Finally these values are used to add a marker on the map as shown in the following Code Snippet 6.

```
protected void startMap() {

    getMap().moveCamera(CameraUpdateFactory.newLatLngZoom(new
                        LatLng(9.005401,38.763611), 10));
    ArrayList<HashMap<String, String>> location = new ArrayList<HashMap<String, S
                            tring>>();
    String server_url = "http://www.cc.puv.fi/~e0900775/output.json";

    try {

        String result = getHttp(server_url);

        JSONArray data = null;
        try {
            data = new JSONArray(result);
        } catch (JSONException e1) {
            e1.printStackTrace();
        }


        HashMap<String, String> myMap;


        for (int i = 0; i < data.length(); i++) {

            JSONObject object = data.getJSONObject(i);


            myMap = new HashMap<String, String>();

            myMap.put("LocationID", object.getString("LocationID"));

            myMap.put("Latitude", object.getString("Latitude"));

           myMap.put("Longitude", object.getString("Longitude"));

            myMap.put("LocationName", object.getString("LocationName"));

            location.add(map);

        }

    } catch (JSONException e)
```

```
        e.printStackTrace();
    }

    for (int i = 0; i < location.size(); i++) {
        Latitude = Double.parseDouble(location.get(i).get("Latitude").toString());
        Longitude = Double.parseDouble(location.get(i).get("Longitude").toString());
        String name = location.get(i).get("LocationName").toString();
        MarkerOptions marker = new MarkerOptions().position(new LatLng(Latitude,
                Longitude)).title(name);
        googleMap.addMarker(marker);
    }
```

Code Snippet 6 Maker adding

5.4 Bank Activity Class

This class handles displaying the current currency rate which is accessed from external source using an application view.

The onClick method is defined in activity_addis.xml redirecting the Bank button to BankActivity which handles the functionality stated above.

```
public void showBank(View view) {
    Intent intent = new Intent(this,BankActivity.class);
    startActivity(intent);
}
```

Code Snippet 7 Show Bank Activity

The following Code Snippet 8 shows the initializing of BankActivity and handles converting the web view to the Android application which use the URL of the web view.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    site = "https://www.dashenbanksc.com/cashexchangerate/rate.php";
    setContentView(R.layout.activity_bank);
    webView = (WebView)findViewById(R.id.webView);

    WebSettings webSettings = webView.getSettings();
    webSettings.setJavaScriptEnabled(true);
    webView.loadUrl(site);
    webView.setWebViewClient(new WebViewClient());
}
```

Code Snippet 8 Converting Web View

To return to the main menu, onBackPressed () function implemented as show below in Code Snippet 9.

```
@Override
public  void onBackPressed(){

    if(webView.canGoBack()){
      webView.goBack();
    } else {
       super.onBackPressed();
    }
}
```

Code Snippet 9 Get back menu page

5.4 Hotels Activity Class

This class defines the structure of the hotel related functions within the project and the attributes and the method which belong to it is shown in the following Code Snippet 10.

```
public class Hotels {

    private  String hotelsId ;
    private String name;
    private  String description;
    private  String phone;
    private  String site;
    public  String getHotelsId()

    {
        return  hotelsId;
    }

    public String getName()
    {
        return name;
    }

    public  String getDescription() {return description;}

    public  String getPhone() {  return phone }

    public  String getSite()  {return site; }

    public Hotels(String hotelsId,String name,String description,String phone,String site) {

        this.hotelsId = hotelsId;
        this.name = name;
        this.description =description;
        this.phone = phone;
        this.site=site;
    }
}
```

Code Snippet 10 Get back menu page

The list of hotels with their name and imageView and the click event of the list is handled by the following the code shown in Code Snippet 11.

```
TextView nameTxt = (TextView) convertView.findViewById(R.id.nameTxt);
nameTxt.setText(hotel.getName());
ImageView view = (ImageView) convertView.findViewById(R.id.imageView);
Bitmap bitmaph = getBitmapFromAsset(hotel.getHotelsId());
view.setImageBitmap(bitmaph);
```

```
view.setOnClicker(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
    }
});
```

Code Snippet 11 ImageView List handling

The setOnClickListener link the button map to a HotelMapActivity which show all the available hotels with a customized marker in a Google MapS. The following Code Snippet 12 show this implementation.

```
Button map =(Button) findViewById(R.id.map);
map.setOnClickListener( new View.OnClickListener(){

    @Override
    public void onClick(View v) {
        Intent intent = new Intent(HotelDetail.this,HotelsMap.class);
        startActivity(intent);
    }
});
```

Code Snippet 12 Hotel map show button

5.4 Museum Class

The class museum code is defined as the following code snippet and this class implements the clusterItem class.

```
public class Muzium implements ClusterItem {
    public final String name;
    public final int profilePhoto;
    private final LatLng mapPosition;

    public Muzium(LatLng position, String name, int pictureResource) {
        this.name = name;
        profilePhoto = pictureResource;
        mapPosition = position;
```

```
    }

    @Override
    public LatLng getPosition() {
        return mapPosition;
    }
}
```

Code Snippet 13 Museum Class Definition

The MuseumActivity which use the museum class implement also clustering for the map of museum .The following Code Snippet 14 show the cluster algorithm in the background thread.

```
private class ClusterTask extends AsyncTask<Float, Void, Set<? extends Cluster<T>>> {
    @Override
    protected Set<? extends Cluster<T>> doInBackground(Float... zoom) {
        mapAlgorithmLock.readLock().lock();
        try {
            return mapAlgorithm.getClusters(zoom[0]);
        } finally {
            mapAlgorithmLock.readLock().unlock();
        }
    }
```

Code Snippet 14 Cluster infowindow click event

All the markers added to the map has click event and the following section handles this event.

```
@Override
public boolean onMarkerClick(Marker customizedMarker ) {

 customizedMarker.showInfoWindow();
    return true;
```

Code Snippet 15 Marker click event

6. TESTING

This section describes the testing of the project in terms of working environment platform, the main functional requirements and the non-functional requirements achievement according to the predefined objectives.

6.1 Platform Testing

The application is tested on Android 4.2 called Jelly Bean and on the current version of Android 6.0 called Marshamallow. It runs perfectly in all Android version which are between JellyBean and Marshamallow. This is because the Android SDK platform package includes API level 17 to API level 23.

During testing, different AVDs were created for different API levels and the application run and showed all the expected outcomes with the only exception that the map could not be displayed on the emulator. Then for this part the real Android mobile phone was used and the Google Maps displayed perfectly.

6.2 Requirement Testing

All the functional and nonfunctional requirements are tested. Regarding the functional requirements the application gives enough information and makes it easy to search and navigate to places in the city. The GUI is user friendly and used different images to represent places and markers.

The application is designed in a way to be flexible. That means it is easy to use it for another city with some customization on the implementation.

7. CONCLUSION

The project deals with the main functions of a city guide application as stated in the introduction. The ultimate goal of the project was designing an application which include the services working around tourism industry and as much as possible to provide many packages within one application. The second consideration was the drawback of the existing mobile application working in the city and this project tried to implement a new concept.

The main objective of the project were achieved and the user of the application can easily navigate any place with the city, easily search for hotels, easily find tourist sites using the map which has a customized marker, access the current currency exchange rate and finally the user can easily reserve a hotel room from this application.

The application is working on Android platform and can run on Android 4.2 to the current version of Android OS. During implementation it uses a lot of API and an external JAR file.

Finally, it was good exposure to work on this project. I got a lot of lessons working on Android programming, especially manipulating Google Maps for the application purposes. The challenging part was to get resources around implementing the clustering marker.

7.1 Future work

There are some things need to be included in this project in the future to make the application more efficient and fully functioning.

Firstly, Google Maps for the Bank, which has both clustered and customized markers must be included in the application.

Secondly, the marketplace handling functionality, which handles locating the marketplace and promoting the products are the ideas that will be implement in the future.

REFERENCES

/1/ Configure Android Studio. Accessed on 05.12. 2015
https://developer.android.com/studio/intro/studio-config.html

/2/ what is Android. Accessed on 5.12.2015
http://www.tutorialspoint.com/android/android_overview.htm

/3/ Get API key. Accessed on 20.12.2015
https://developers.google.com/maps/documentation/android-api/signup

/4/ Map Object. Accessed on 4.01.2016.
https://developers.google.com/maps/documentation/android-api/map

/5/ JSON Object. Accessed on 2.02.2016
https://developer.android.com/reference/org/json/JSONObject.html

/6/ Introduction to JSON. Accessed on 10.02.2016
http://json.org/?cm_mc_uid=31697722836114625105726&cm_mc_sid_50200000=1464007205

/7/ Markers. Accessed 20.03.2016
https://developers.google.com/maps/documentation/androidapi/marker#code_samples

/8/ Google Maps Android Marker Clustering Utility. Accessed on 20.03.2016.
https://developers.google.com/maps/documentation/android-api/utility/marker-clustering

/9/ Addis Ababa Map. Accessed on 10.4.2016
http://www.viamoro.com/Africa/Ethiopia/Addis-Ababa/latlong/

/10/ Map Style. Accessed on 15.04.2016
https://www.mapbox.com/android-sdk/#map_styles

/11/ Device Compatibility. Accessed on 5.12.2015
https://developer.android.com/guide/practices/compatibility.html
/12/ MrkerClusterer. Accessed on 5.5.2016
https://developers.google.com/maps/articles/toomanymarkers#fusiontables
/13/ Apache HttpComponent. Accessed on 20.4.2016
https://hc.apache.org
/14/ Sequence Diagram. Accessed on 3.5.2016
https://www.visual-paradigm.com/VPGallery/diagrams/Sequence.html