ANDROID APPOINTMENT MANAGER APPLICATION DEVELOPMENT WITH GOOGLE APIs

O. Osunade[1], A. J. Osho[2] and S. O. Oyebamiji[3]

[1,2]Department of Computer Science, University of Ibadan, Nigeria

[3]Department of Library, Archical Records and Information Studies, University of Ibdan, Nigeria

Abstract:

People are involved in many activities that are planned, unplanned, routine and emergency in nature. The ability to manage all these activities without conflict is desired by all persons, because time management is one of the attributes of successful people. With the proliferation of mobile devices in our society, this work seeks to develop an appointment management application for mobile devices using the android platform. The developed application utilized two application programming interfaces (APIs) from Google for the map and calendar. Other parts of the application were developed in Java. The results show a functional mobile application for appointment management.

Key Words: Google API, Android, mobile application, time management, Java

Introduction

Time is a universal resource to all men. The use of time is defined by activities, tasks and procedures. Time is finite and thus limited for the accomplishment of activities, tasks and procedures. Individuals have designed several ways to keep track, handle and manage the use of time. Such designs include clocks, wrist watch, sand dial, diary, and appointment books.

In this digital age, several of the designs developed for time have been digitized and integrated into other devices such as smartphones, tablets and microwave ovens. As the information phase of the digital age progresses, time devices are being translated into software applications to facilitate time management.

Time Management refers to planning, organizing, allocating, using and controling time effectively so that all ideintified activities, tasks and procedures are executed at the right time.

It is also making the best use of time so as to increase effectiveness, efficiency or productivity. Individuals assign specific time slots to activities based on their importance -this is called an appointment. Appointments were traditionally managed using diaries and appointment books, however in the information age, mobile software programs called apps are being used. A number of such appointment managers exist but with limited features and functionality.

The aim of this project is to develop a time management application for mobile device users on the android platform. The objectives of the project are: to integrate applicable Google APIs; design the user interface to work the Google APIs; and deployment of the mobile application to an android mobile device.

Literature Review

Choudhari et al (2014) developed a system that will ease the process of booking appointments with the doctor. The patient will book the appointment through his/her mobile phone. The doctor will come to know the number of patients he has to attend in the day. The system will save patient's as well as doctor's time. It will save the receptionist's paper work. The system will prove to be useful for the doctor as he can check his appointments whenever and from wherever he wants to using his mobile phone. The proposed system consists of two panels: Doctor and Patient. The users will first have to download the application and install it in their mobile devices. Once installed, this application will remain into the device permanently until the user deletes it or uninstalls it. The patient will have to register into the application for the first time. On registering, the patient will receive a username and password. The patient can use this username and password for logging into the app each time he uses it. After logging in, the patient will have to select a filtration type. The filtration is done on two bases: Area wise and Specialty wise. After selecting the filtration type, the doctors list will be displayed. The patient can select any particular doctor and view his profile. Also the patient can view the doctor‟s schedule and look for an appointment according to his convenience. The patient will then send a request for appointment. The doctor can either accept the appointment or reject it. The database will get updated accordingly and the patient will get a confirmation message. The add-on to this system is that the patient will receive a notification 2 hours before the actual appointment. This will be very useful in case the patient tends to forget the appointment.

Symey et al (2013) proposed to develop an alternate patient appointment system using Near Field Communication (NFC) technique and Android enabled mobile application with a view to redefining the core of hospital waiting time towards appointment and also collection of medicines. These were carried out in practice using appropriate NFC hardware, Android SDK, PHP and MySQL database.

A work was carried out towards scheduling appointment for students using Agents from Android handset recently. There were a few drawbacks in the existing system, like no provision of scheduling between lecturer and Lecturer, and it did not take into consideration the time span between the scheduling, rescheduling and cancellation of appointment and the actual start of the appointment. Another drawback of the previous work was the fact that the appointment diary of the lecturer could not be seen. Last but not the least negotiation between Scheduler agent and lecturer agent can be carried out only if the lecturer's mobile handset is on as the fuzzy preference logic for appointment negotiation resides on the mobile side which is a bit of drawback. Parchment and Sankaranarayanan (2013) worked to alleviate the above mentioned problems by incorporating software agents on Android enabled handset into the educational arena in an effort to solve scheduling appointments woes. Also in the work, it allows the scheduling and cancellation of appointments based on some time period validation. The Smart agents utilize the properties of autonomy and mobility to intelligently schedule appointments on behalf of the lecturer. JADE-LEAP on the latest Android handset was the choice to develop the proposed system.

Grover et al (2013) looked to resolve the challenges faced by the sales industry by developing a "Salesman Application" an android mobile app that provides various hands-on services to a salesman thereby minimizing the reporting time and increasing the efficiency . The app allows a salesman to manage his appointments with clients , submit immediate orders , generate & print receipt via bluetooth printer , track his/her performance index , maintain a products catalogue, also features different payment modes (cheque , DD, Cash) and maintains client's history along with the feedback & a picture of the client (by creating phonegap camera plugin) on their android smartphones and tablets.

Hylton and Sankaranarayanan (2012) discovered that to make an appointment with the hospital staff, it becomes really tedious and time consuming. Over the past considerable amount of work have been done by using software Agents in areas like m-commerce, e-commerce, telemedicine etc. Agent based systems have also been developed for the hospital service, for searching and fixing appointment over mobile phones which gives a direct reply

when the appointment is made or the next available date(s) or cancelled. However, no facility like priority appointment of patients has been developed. Also the appointment does not take into consideration emergency situations like accidents and so on and the scheduling reported is only for general patient appointment only. Taking these important aspects into consideration, we here have developed an intelligent agent based system towards negotiating and collaborating with the agents of doctors and the hospital for the appropriate appointment time for the patient which would take the above factors into consideration. In addition the meetings of the junior staff like the duty doctor and nurse with the chief doctor regarding patients would also carried out again while taking into consideration the medical condition of the patient admitted and so on. These agents developed would function based on fuzzy preference rules, to make a proper decision regarding making an appointment for patient and other hospital staff, which is very unique and first of its kind. The system validated used ANDROID 2.2 and JADE-LEAP, for providing a robust, user friendly solution for the patient and doctor.

Pocatilu (2012) stated that almost all mobile applications use persistency for their data. A common way for complex mobile applications is to store data in local relational databases. Almost all major mobile platforms include a relational database engine. These databases engines expose specific API (Application Programming Interface) to be used by mobile applications developers for data definition and manipulation. This paper focus on database - based application models for several mobile platforms (Android, Symbian, Windows CE/Mobile and Windows Phone). For each selected platform the API and spec ific database operations are presented

Methodology

This is a design and programming work. Programming guidance was obtained from Felker and Dobbs (2011).

The android appointment management application consists of five modules: The modules are

- User Interface
- Appointment Manager Module
- Google Calendar API
- Google Map API
- Google Latitude API

User interface: this consists of the user and the graphical interface on the mobile device. The graphical interface makes the task of putting appointments and time allocation easy.

Appointment Manager Module: this serves the hub of communication with the APIs used in this work. Sample codes used for the integration are included in this paper.

Google Calendar API: lets users incorporate calendar functionality into personal applications or website. Users can edit calendars, create and delete events, query for events that match particular criteria, send invitations and more.

Google Map API: allows developers to integrate Google Maps into applications. Google Maps is a web mapping service application and technology for global use.

Google Latitude API: is used to access and update locations using the latitude, longitude and timestamp. It works with the Map API to produce results.

The model of the Android appointment management is shown in Figure 1.

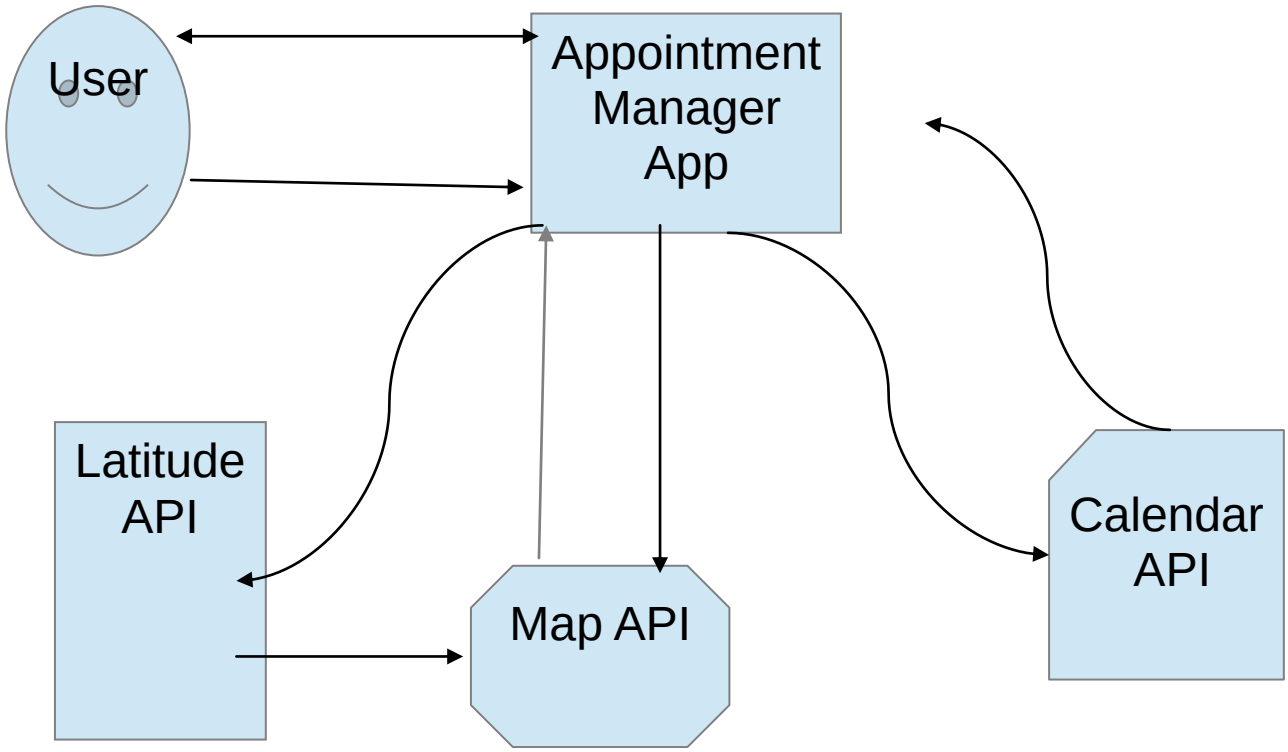

Figure 1: Model of the Android Apointment Manager

In Figure 1, the interation of all the modules of the appointment manager is shown. The user provides details about an appointment with the user interface. The details are stored by the Appointment Manager App which sends the date or time of the appointment to the Calendar API and the location of the appointment to the Map API. The Latitude API obtains the user co-ordinates which are sent to the Map API and then used to load the corresponding map on the user interface.

Hardware Requirements

An android device with the following specifications was used to test the application

- Chipset ARM-based
- Memory 128 MB RAM; 256 MB Flash External
- Storage Mini or Micro SD
- Primary Display QVGA TFT LCD or larger, 16-bit color or better
- Navigation Keys 5-way navigation with 5 application keys, power, camera and
- Camera 2MP CMOS
- USB Standard mini-B USB interface
- Bluetooth 1.2 or 2.0

Software Development Tools

The following software tools were used during the development process.

- Eclipse IDE (Eclipse 3.5 (Galileo)) and Eclipse JDT plug-in
- Eclipse IDE for Java EE Developers, Eclipse IDE for Java
- JDK 5 or JDK 6
- Android Development Tools – SDK, plug-in,
- Apache Ant 1.6.5 or later for Linux and Mac, 1.7 or later for Windows
- APIs: Calendar API, the Google Maps API and the Google Latitude API

Integration Code Snippets for the Google APIs used

Snippets for the integration of the Google maps modules

```
import com.google.android.maps.MapView;
import android.os.Bundle;
public class LatMapView extends MapActivity{
MapView locMapViewe;
@Override
protected boolean isRouteDisplayed() {
return false;
} @Override

 protected void onCreate(Bundle latmap) {
// TODO Auto-generated method stub

 super.onCreate(latmap);
setContentView(R.layout.lat_map_layout);

 locMapViewe = (MapView) findViewById(R.id.locmapview);
//make available zoom controls
locMapViewe.setBuiltInZoomControls(true);
locMapViewe.invalidate();
```

Snippets for the integration of the Google calendar modules

```
import java.text.Format;
import android.app.Activity;
import android.database.Cursor;
import android.os.Bundle;
import android.provider.CalendarContract;
import android.text.format.DateFormat;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;
```

```
public class calendarThings extends Activity implements OnClickListener {

 private Cursor calcus;
private static final String[] calstring = new String[]{
CalendarContract.Events.TITLE, CalendarContract.Events.DTSTART};

 @Override

 protected void onCreate(Bundle calend) {
// TODO Auto-generated method stub
super.onCreate(calend);
setContentView(R.layout.caledar_layout);
calcus = getContentResolver().query(

CalendarContract.Events.CONTENT_URI,calstring ,null,null,null);
calcus.moveToFirst();
```

Snippets for the integration of the Google latitude modules

```
com.google.api.client.http.HttpResponseException: 401 Unauthorized
at com.google.api.client.http.HttpRequest.execute(HttpRequest.java:380)
at
com.google.api.services.latitude.Latitude$RemoteRequest.execute(Latitude.java:550)
at
com.google.api.services.latitude.Latitude$CurrentLocation$Get.executeUnparsed(Lati
tude.java:222)
at
com.google.api.services.latitude.Latitude$CurrentLocation$Get.execute(Latitude.java:
207)
at
com.ecs.android.sample.oauth2.AndroidOauthGoogleApiJavaClient.getCurrentLocati
on(AndroidOauthGoogleApiJavaClient.java:107)
at
```

```
com.ecs.android.sample.oauth2.AndroidOauthGoogleApiJavaClient.performApiCall(
AndroidOauthGoogleApiJavaClient
Latitude latitude = new Latitude(transport, accessProtectedResource, jsonFactory);
latitude.apiKey=OAuth2ClientCredentials.API_KEY;
LatitudeCurrentlocationResourceJson currentLocation =
latitude.currentLocation.get().execute();
```

Results And Discussion

The screenshots were taken during the implementation of the designed appointment manager for Adroid mobile devices.

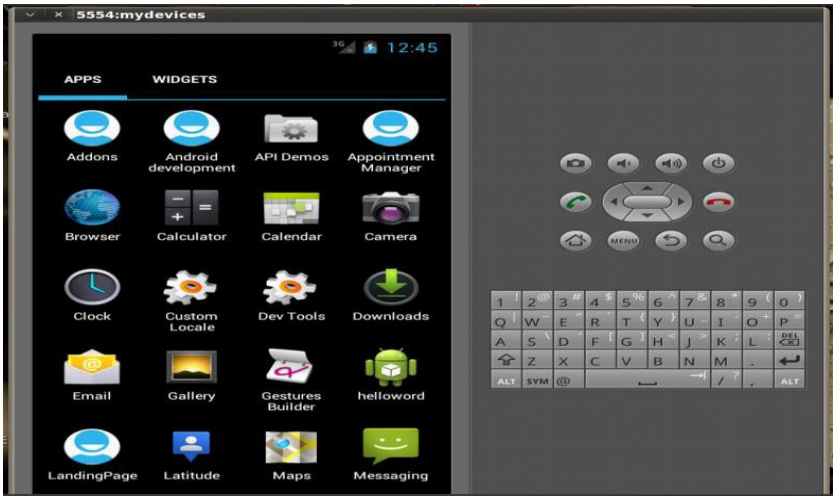

Figure 2: Home Screen with Icon for Appointment Manager

Figure 2 shows the home screen of the Android mobile device simulator. The icon for the appointment manager is the fourth icon on the first row.

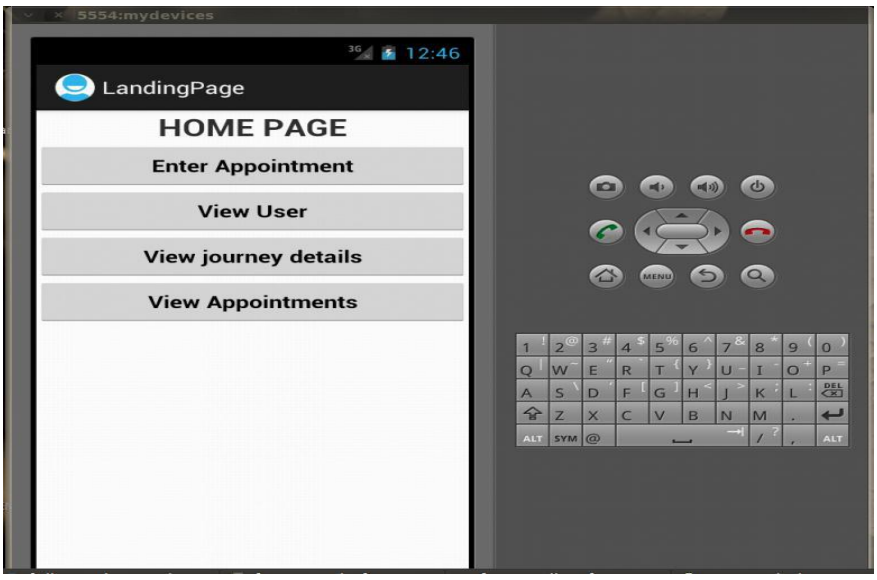

Figure 3: Home Page of Appointment Manager

The user interface for the appointment manager is shown in Figure 3.

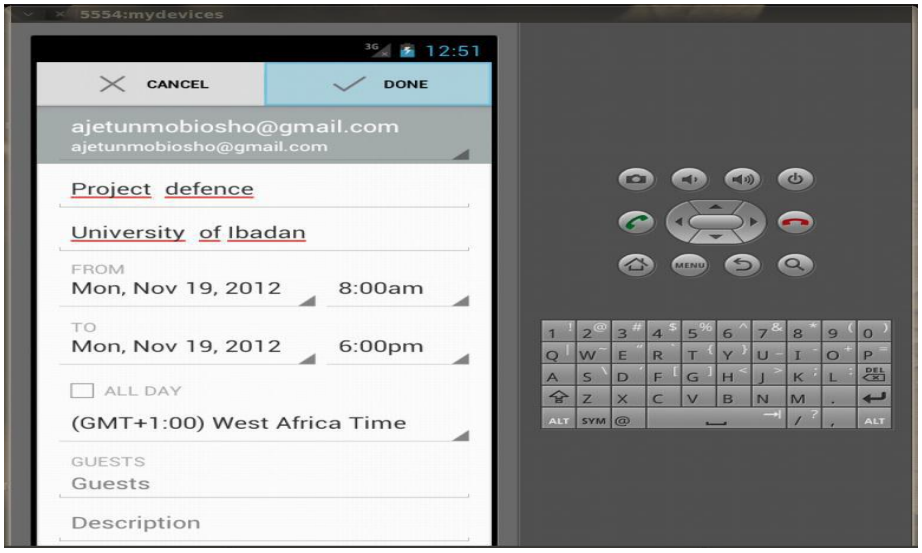

Figure 4: Appointment Data entry screen

The user provides details such as place, time (start and end), reminder of an appointment using the screen in Figure 4. The users Gmail address is used for identification of all operations carried out with the Google APIs.

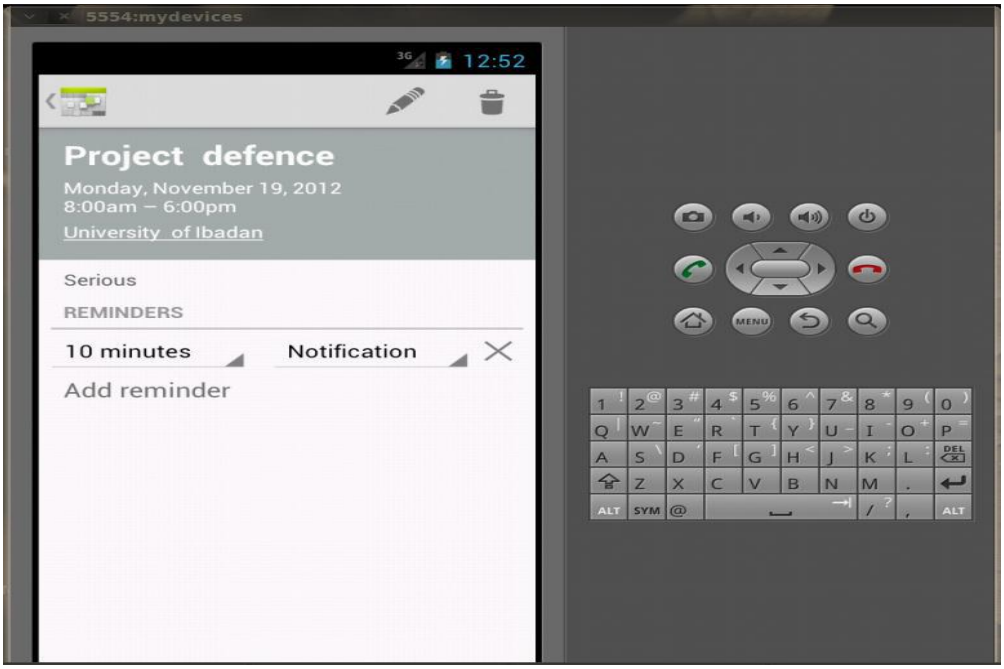

Figure 5: Details of an Appointment

Figure 5 is another view of the appointment as stored in the calendar. It provides the information about reminders. The user can edit the appointment from this view.

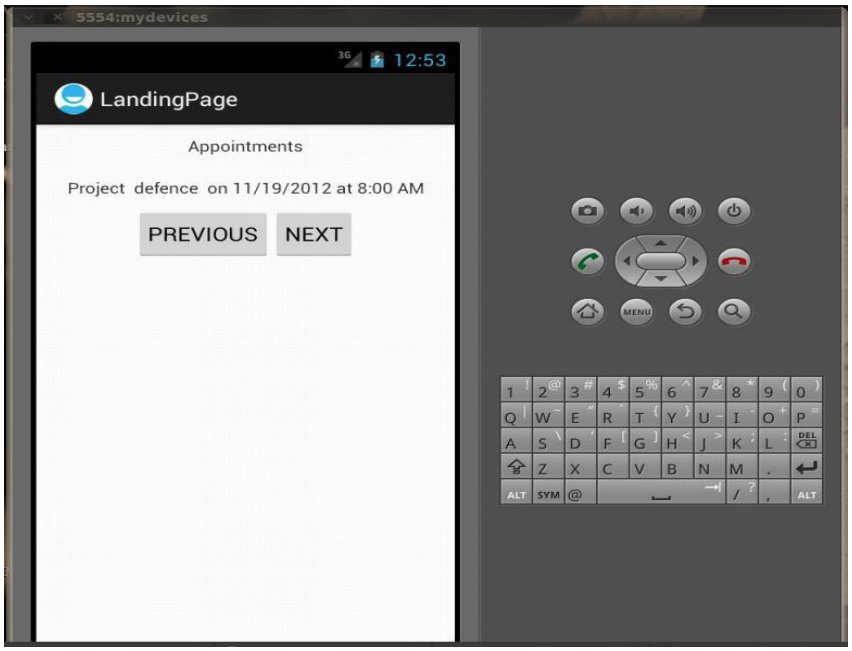

Figure 6: List of Apointments made

A list of all appointments made can be viewed as shown in Figure 6.

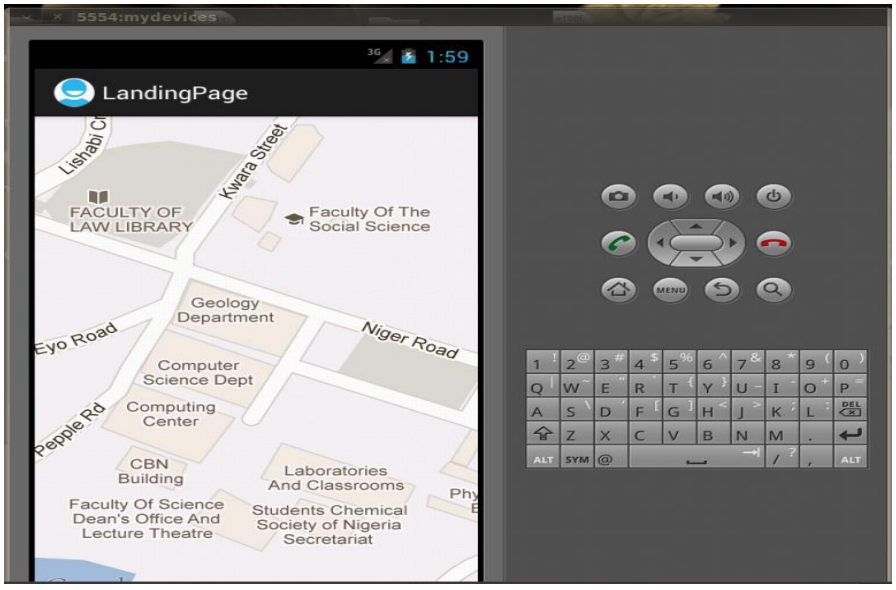

Figure 7: Map of Appointment location

The map of the Computer Science department is displayed in Figure 7 based on the appointment detail and request for a map of the appointment location.

Some scenarios the Android Application is designed to solve include

User 1 to meet an Appointment

The user inputs the location and time of the appointment e.g. a lecture. The application initiates the Google Map API via Wi-Fi, GPS or a cellular radio on the mobile phone to get the user's current location. Based on the information received, Google maps calculates the amount of time it would take for the user to get to the destination.

User 1 to meet User 2 for an appointment at User 2's locations

In this scenario each user have the application installed on the their devices. User 2 can monitor user 1's movement and user 2 will obtain the amount of time it would take user1 to get to the location of the appointment via the android application.

Conclusion

In this work a time management application was developed for the android platform by integrating two Google Application Programming Interfaces (API)- Google maps API and the Google Calendar API - with additional programming in Java. The mobile application developed accepts appointments and events by storing the appointment on the phone calendar which is synchronized with the Google calendar and alerts the user at a preset specified time before the appointment. The Google map incorporated into it allows the user view the location beforehand thereby reducing the time it takes to get the location. It would be recommended that future work be done to integrate the Google latitude API that will assist in distance calculation.

References

Choudhari, S. B.; Kusurkar, C.; Sonje, R.; Mahajan, P. and Vaz, J. (2014) Android Application for Doctor‟s Appointment. International Journal of Innovative Research in Computer and

Communication Engineering, 2(1): 2472-2474

Felker, D. and Dobbs, J. (2011) Android Application Development For Dummies. Published by Wiley Publishing, Inc. USA . ISBN: 978-0-470-77018-4

Grover, T.; Makhija, A.; Goyal, A. and Sharma, D. K. (2013) Salesman Mobile Application (on Android). International Journal of Scientific & Engineering Research, 4(9): 2344-2349.

Hylton III, A. and Sankaranarayanan, S. (2012). Application of Intelligent Agents in Hospital Appointment Scheduling System. International Journal of Computer Theory and Engineering, 4(4): 625-630.

Parchment, D. and Sankaranarayanan, S. (2013) Intelligent Agent based Student-Staff Scheduling System. International Journal of Computer Information Systems and Industrial Management Applications, 5:383-404

Pocatilu, P. (2012) Building Database-Powered Mobile Applications. Informatica Economică, 16(1): 132-142

Symey, Y.; Sankaranarayanan, S. and Sait, S. N. (2013) Application of Smart Technologies for Mobile Patient Appointment System, International Journal of Advanced Trends in Computer Science and Engineering, 2(4): 74-85