BOUNDARY DISTRIBUTION USING ANDROID GEOGRAPHICAL MAPPING

(A CASE STUDY OF ILORIN EAST LOCAL GOVERNMENT AREA OF KWARA STATE)

BY

JIMOH AISHAH FEYISHAYO
HND/23/COM/FT/0121

A PROJECT REPORT SUBMITTED TO THE

DEPARTMENT OF COMPUTER SCIENCE

INSTITUTE OF INFORMATION AND COMMUNICATION TECHNOLOGY, KWARA STATE POLYTECHNIC ILORIN

IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE AWARD OF HIGHER NATIONAL DIPLOMA (HND) IN COMPUTER SCIENCE

2025

CERTIFICATION

This is to certify that this project was carried out by **Jimoh Aishah Feyishayo** with Matriculation Number **HND/23/COM/FT/0121** in the department of Computer Science, Institute of Information and Communication Technology, Kwara State Polytechnic, Ilorin.

---------- ----------

MR. ISIAKA O.S. Date
(Project Supervisor)

---------- ----------

MR. OYEDEPO F.S. Date
(*Head of Department*)

---------- ----------

External Supervisor Date

DEDICATION

This Project is dedicated to Almighty Allah the beginer and the end of the universe.

ACKNOWLEDGEMENT

I give all the glory, honour, adoration and Majesty to Almighty Allah for his mercies, faithfulness and protection over my life, academically, physically, financially and spiritually.

I will be ungrateful if I fail to acknowledge the contribution of my supervisor in person of Mr. MR. ISIAKA O.S. and my able H.O.D MR. OYEDEPO F.S. for their numerous helpful and assistance towards the successful of this project wishing you more long life and prosperity (Amen).

My profound gratitude goes to my parent Mr and Mrs Jimoh, thanks for always be their for me if not you where would I be by now. Thank you sir and ma.

My gratitude also goes to all my friends and love once for their pieces of advice and cooperation which will for ever remain ever green in my memories.

TABLE OF CONTENTS

ABSTRACT

Boundary distribution is a critical aspect of geographical mapping, essential for effective land management, resource allocation, and administrative planning. This study explores the development and implementation of an Android-based application for efficient boundary distribution. Leveraging the geospatial capabilities of Android devices, the application provides an intuitive platform for real-time mapping, visualization, and analysis of boundary data. Key features include GPS integration for accurate location tracking, interactive tools for boundary adjustments, and seamless data storage and retrieval. The solution addresses challenges in traditional mapping methods by offering portability, ease of use, and cost-effectiveness. This approach demonstrates the potential of mobile technology in enhancing boundary management processes, enabling precise and scalable solutions for diverse applications such as urban planning, agricultural zoning, and environmental monitoring. Finally, a review of the system confirms the system's usefulness within the context, particularly in the case study area of Ilorin East Local Government, Ilorin Kwara State, where the implementation is centered. This project was created with Android Studio 2.3.

CHAPTER ONE
GENERAL INTRODUCTION

1.1 INTRODUCTION

Mapping is defined as the art, science, and technology of creating and studying maps, and it has seen a resurgence in popularity as computers have made it possible to create maps and spatial interpretations. These computer programs and applications are not difficult to use, and they do not require any special data or systems to generate excellent and informative maps. Given the widespread availability of personal computers and the widespread distribution of data in electronic formats, there is no reason why data pertaining to standard geographic areas cannot be quickly mapped and included in almost any type of printed report or visual medium.

Mapping has changed dramatically in the twenty-first century as a result of technological advances. Technology has enabled the mapping of numerous locations that were previously thought to be impossible. Most maps rely on clearly defined boundaries to be recognizable and meaningful. These boundaries can be physical, such as coastlines or rivers, or political, such as state and county lines. Although drawing boundaries may appear to be a simple and fundamental task to the average map user, it is one of the more complex tasks involved in computer mapping. The majority of the data involved in the creation of a computer-drawn map is comprised of boundary definitions, and the quality of any map is determined by the quality of the boundary data.

According to Isenberg et al. (2011), shared cognition can occur in a variety of scenarios within two types of collaborative visualization environments: distributed and co-located. A distributed collaborative visualization is concerned with how collaborative contributions can be effectively structured and integrated into a shared visualization aimed at the division or spread of resources such as design artifacts, design knowledge, or design team to form common ground. These activities are carried out through video conferences and email-based data discussions. A co-located collaborative visualization, on the other hand, is one in which these collaborative works are initiated into a shared visualization in the same locations, where the design team can conduct further discussion via wall displays or shared tables. This study aims to identify the implications of different visualization strategies and techniques of shared visualization applications, as well as how they can be applied to designing collaborative visualization systems, in order to understand the implications of strategies demonstrated in both types of collaborative visualization environments. We are particularly interested in determining the strategies and techniques used by researchers in handling datasets in shared visual representation to achieve a specific result of shared cognition.

1.2 STATEMENT OF THE PROBLEM

The geographic precision of the reports that did have coordinates appeared to vary. Because of the varying level of detail in attribute data, it was necessary to visit the locations in order to determine the possible causes of the traffic. The study gave me the impression that there must be room for improvement in terms of data quality and quantity by improving the production process.

1.3 AIM AND OBJECTIVES

The aim of this project is to create an android-based geographical mapping system in terms of boundary distribution, with the following objectives:

i. Use of GIS, GPS, and RS to implement the system.
ii. Increasing and improving the value of Android phones, which are the most popular smartphones in Nigeria.
iii. Using smartphones for geographic modeling.

1.4 SIGNIFICANCE OF THE PROJECT

This system will demonstrate how powerful and accurate Android phones can be when used properly. It will show how GIS and GPS in an Android phone can provide accurate mapping that can be used to guide all map users. The study will also have a significant impact on cartography in Nigeria.

1.5 SCOPE OF THE STUDY

The study's scope is to implement a geographical mapping system within an environment focused on the territory of Ilorin East Local Government Area, Kwara State.

1.6 ORGANIZATION OF THE REPORT

The first chapter contains the study's introduction, problem statement, aim and objectives, significance of the study, scope of the study, organization of the study, and definition of terms. The second chapter examines previous works and conducts a literature review. The third chapter describes the methodology, the existing system's problem, and the existing and proposed systems. The fourth chapter presents the results of the input, output data of the proposed system, and system requirements. The fifth chapter contains the study's summary, conclusion, and recommendations.

CHAPTER TWO

LITERATURE REVIEW

2.1 REVIEW OF RELATED PAST WORK

Arun et al. (2016) emphasized the importance of combining technologies, services, and data sources in order to build credible and legitimate NRT forest monitoring systems at the appropriate scale, whether at the local, provincial, or national levels. These systems provide data on deforestation hotspots. Such information has enabled law enforcement, civil society, and the media to respond quickly to illegal activities, thereby slowing the rate of deforestation.

Makhtar et al. (2016) created the Mobile Attendance System (MAS), which includes an Android smartphone, GPS technology, a Wi-Fi access point, and a server. The overall system's function is very simple. It uses a GPS receiver embedded in a smart phone to locate employees and automate clocking in and clocking out attendance in real time by pressing a button on the phone. The smart phone's IMEI number and GPS information are saved in a database. The system could be used to replace the current attendance methods. The system has been successfully tested in a real-world situation outside the building since its implementation. When the user is away from the office or outstation, it facilitates the process of taking staff attendance in an efficient and cost-effective manner. MAS is a must-have application for employees who value every minute of their workday.

Samia et al. (2015) discussed the lessons learned during the development of the Immunization Information System's Android application. The app not only assists in the creation of digital health records in the field, but it also assists in the monitoring and validation of vaccinators' performance using real-time data. However, the success of the vaccination program is dependent not only on the performance of the vaccinators, but also on parental awareness, availability, and compliance.

2.2 REVIEW OF GENERAL TEXT

The first step in determining the architecture of an appropriate system is to look at previous examples. Mansouri et al. (2006) proposed a web-based system in an SDI (Spatial Data Infrastructure) framework for reliable information collection and sharing to aid disaster management, a topic arguably similar to that of this project. The web-based system includes the essential components of a web GIS, such as a user interface, a web server and application server, a map server, a data server, and a database. A Web-based system would have a similar structure to what this project is attempting to achieve, with the exception of the additional user interface, which is the added mobile platform for data collection.

Kresse & Danko (2012) described the implementation of a map server while presenting a similar architecture for a Web-GIS. A map server, which generates maps from spatially referenced data, is an essential component of a Web-GIS. Danko and Kesse's architecture is described as a three-tier architecture with a client, a server, and a geospatial database access system. The server tier is divided into three components: a web server, an application server, and a map server (Kresse & Danko, 2012). Despite some differences in presentation, the architectures described by Mansourian et al and Danko & Kesse are essentially very similar. The descriptions of how the components interact and function together are, most importantly, the same.

The user interface, also known as the client, is where data is displayed for visual analysis. The user interface of a web-based GIS is a graphical environment that is viewed in a web browser. The graphical environment is made up of several documents that contain HTML tags and JavaScript elements (Mansourian et al., 2006; Kresse & Danko, 2012). The web server receives client requests in the form of URLs. With the assistance of an application server, the request is routed to a map server. As previously stated, the map server will generate maps from spatial data based on the web server's request. The map server's spatial data is served by the data server, which retrieves it from the database in response to a request.

Mansourian et al. (2006) recommend including mirror databases for data backup to ensure that the system is always operational. When the original database fails, the mirror databases are activated. When the mirror databases are not active, they replicate all changes made to the original database. The user should not notice any difference regardless of which database is active at the time.

Selecting platforms and tools to work with is part of the developer's task when planning the architecture of a system. With limited personal experience, such as creating mobile applications and servers and databases, it becomes necessary that resources such as extensive instructions and documentation are available for the chosen software and platforms. Furthermore, the project's budget is constrained. The conclusion is that the architecture should include as much free and open source software (FOSS) and platforms as possible. The goal of open source is to collectively create and improve source code for rapid software evolution with the help of an engaged community (Open Source Initiative 2012, n.d.-e). A program written in its original programming language, such as C or Java, is known as source code. It is the language in which the program is written and will later be transformed for execution by the machine.

In some cases, distributing an application programming interface is a solution for increasing the popularity of a product (API). An API is a set of guidelines and specifications for a system that make programming for that system easier. Using API, programmers can create a program

without having a deep understanding of the system (Merriam-Webster n.d.). Google did this early on when they distributed an API for Google Earth, making it useful to millions of users, a feat dubbed "the democratization of GIS" by some (Goodchild, 2007). Google Maps API can now be used to embed and customize maps on websites or mobile applications, further increasing the app's popularity.

Figure 2.1: ESRI's Collector for ArcGIS.

With large user communities, web evolution, and lower thresholds for creating customized tools, it is now possible to build a system entirely of free and open source software, if desired. An example is Brovelli et al. (2014)'s proposed web-based prototype architecture for reporting pavement damages. The data collection process is carried out on a mobile device on the client side, and the output can be viewed on desktop computers, tablets, and mobile phones. This project, on the other hand, only necessitates data collection on mobile devices and output to desktop computers. Brovelli et al. (2014)'s prototype architecture is similar to what this project aims to achieve.

CHAPTER THREE

METHODOLOGY AND ANALYSIS OF THE EXISTING SYSTEM

3.1 RESEARCH METHODOLOGY

The research methodology is based on a solution developed with a license from open source services provided by geographical information system companies such as Google (Google Maps), among others. Materials were also used to extract pertinent information on the use of GIS and its connection to an Android phone via quantitative techniques that included the use of the internet.

This project was carried out using traditional research methods. There were four levels of geographic information services used. Data streams will be collected using community-based and satellite monitoring via Android phones. The project makes extensive use of satellite imagery. GPS coordinates will also be used to improve precision. The use of time-based modulation is critical because it will be used in the studies. A location-based disturbance detection system based on satellite time series will be implemented. The internet has advanced to the point where we can now use web-based applications rather than just device or desktop-based applications.

Initially, the geographical area was sketched out. After mapping it out, a platform was built with the help of a global positioning system, or GPS. Remote sensing (RS) and GIS (geographical imaging systems) technology will also be used. All of these factors combined to create a well-balanced and highly enhanced system. This system provided precise information about the entire geographical area. Control strategies were implemented to improve accuracy even further. Android phones are currently the most popular phones in the world. They control more than 70% of the smartphone market. Android phones were used for geographic modeling.

3.2 ANALYSIS OF THE EXISTING SYSTEM

The existing system describes a proof-of-concept study for the design and implementation of an android-based geographical boundary distribution system for the Ilorin East Local Government in Kwara State. Our implementation has undergone an initial fitness for purpose evaluation, and the results are being prepared for this. Overall, the evaluation results are positive, but more research over a longer period of time with more participants is needed to realize its true benefits and potentialities in terms of boundary distribution.

At the moment, the existing system has two major design limitations that cause some time lag in boundary sharing. First, for Landsat data processing, the system employs a semi-automatic process chain. Our system searches for and downloads newly available Landsat images from the server manually. This manual approach prevents us from establishing an

automatic Landsat data process chain. The local government has recently improved their data distribution infrastructures and now allows automatic downloads of Landsat scenes via external platforms such as the Google Earth Engine. This manual interaction is introduced for a short period of time by disseminating alerts to local stakeholders. This limitation could be overcome by sending location change alerts directly to mobile devices and utilizing geo-fence services to notify people about the change locations, as proposed in the system.

3.3 PROBLEM OF THE EXISTING SYSTEM

i. The existing system cannot function in places that are not clearly defined. Coordinates and mapping must be error-free in order to function properly. It is unable to properly define overlapped areas.
ii. The data captured from the existing system contains numerous errors. The android-based system has a lot of features that will make it more precise and reduce errors significantly.
iii. The current system is not portable. Android systems are lightweight and portable, allowing them to be carried anywhere. This will make mapping very simple and affordable for everyone. Because it will be accessible to a large number of people, the geographical mapping system's boundary distribution will be well defined.
iv. The existing system is costly to operate. The old system was prohibitively expensive. The proposed system will be less expensive to operate. Every component is modular and can be swapped out at any time.
v. Data collection is simple. To collect data in the old system, GIS and GPS systems had to be combined with other systems. The new system simplifies data collection.

3.4 DESCRIPTION OF THE PROPOSED SYSTEM

The proposed system will carry out geographical mapping in terms of boundary distribution using an Android smartphone. As the system must outperform existing systems, the boundary distribution must be unambiguously accurate. GIS and GPS will be used to demonstrate the ever-changing world of cartography.

3.5 ADVANTAGE OF PROPOSED SYSTEM

i. The proposed system will function in a variety of settings. It makes no difference if the area to be mapped is highly contorted, hilly, or rough. The new system will function properly.
ii. The new system will be able to map areas that overlap.
iii. The data captured from the existing system contains numerous errors. The android-based system has a lot of features that will help it be more precise and reduce errors significantly.

iv. The current system is not portable. Android systems are lightweight and portable, allowing them to be carried anywhere. This will make mapping very simple and affordable for everyone. Because it will be accessible to a large number of people, the geographical mapping system's boundary distribution will be well defined.

v. The proposed system will be less expensive to operate. Every component is modular and can be swapped out at any time. Android phones are widely available, and they all include GPS for simple triangulation.

vi. New data can be collected and analyzed effectively using new digital methods. Android smartphones are capable of performing some level of analysis. As a result, new technology will elevate this system to new heights.

vii. Data collection will be a breeze with the new system. GIS and GPS systems can be easily combined to create a highly accurate system. Smartphones enable all new data collection systems to collect data digitally. The new system simplifies data collection.

viii. The new system will teach a large number of people about geographical mapping. It was previously complicated, but the new system makes it very simple.

CHAPTER FOUR

DESIGN AND IMPLEMENTATION OF THE STUDY

4.1 DESIGN OF THE SYSTEM

System design is the gradual refinement of data structure, program structure, and procedural details, which are developed, reviewed, and documented. System design can be approached from either a technical or a project management standpoint. Architectural design, data structure design, interface design, and procedural design are the four activities that comprise design from a technical standpoint.

4.1.1 OUTPUT DESIGN

Computer system outputs are primarily used to communicate the results of processing to users. Figure 4.1 depicts the outputs to be obtained from the proposed system.

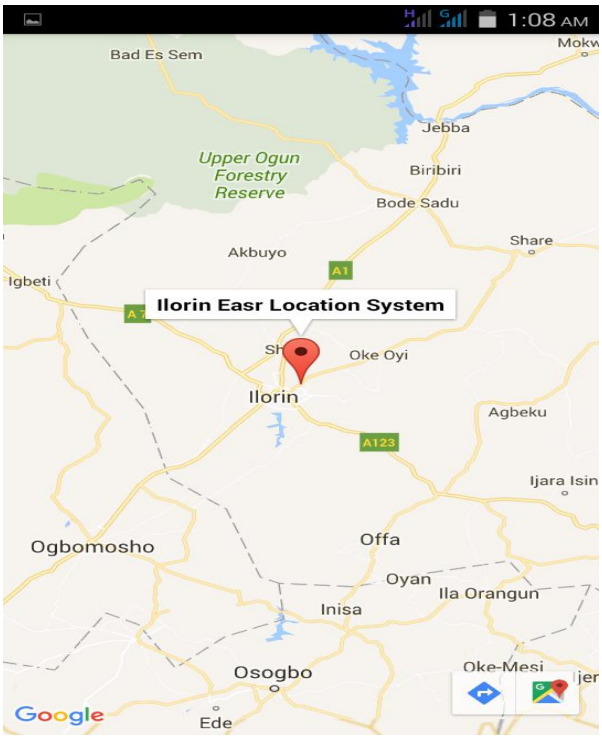


Figure 4.1: Google map showing propose location

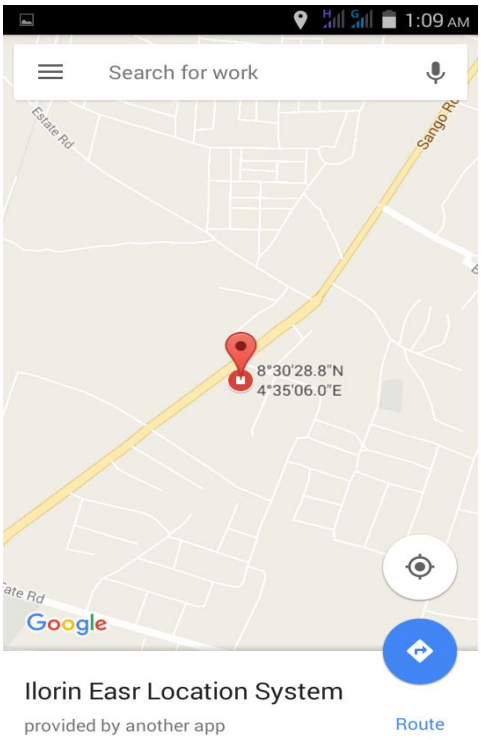


Figure 4.2: Showing longitude and latitude

4.1.2 INPUT DESIGN

The input design is a component of the overall system design. Figures 4.3 and 4.4 depict the inputs to be extracted from the proposed system.

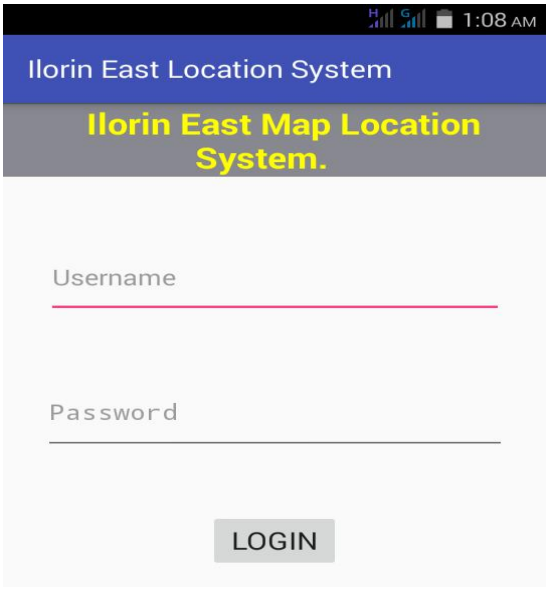


Figure 4.3: Login Page

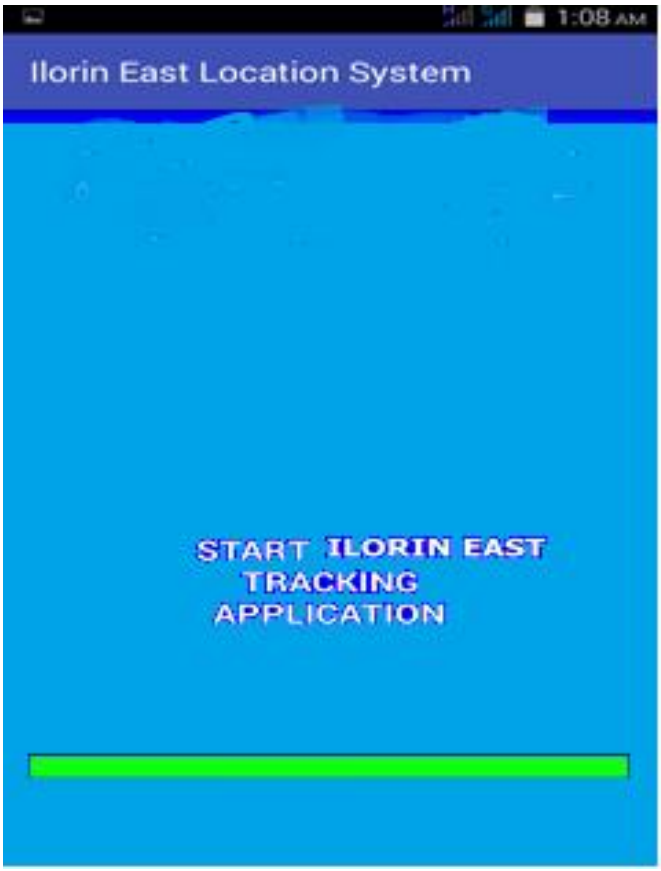


Figure 4.4: Launch Application

4.1.3 DATABASE DESIGN

Most Android apps require data saving, even if it is only to save app state information during onPause() so that the user's progress is not lost. Most non-trivial apps must also save user preferences, and some must manage large amounts of data in files and databases. This class will introduce you to the main data storage options in Android, such as:

i. Keeping simple data type key-value pairs in a shared preferences file.
ii. Saving arbitrary files to the Android file system.
iii. Using SQLite-managed databases.

4.1.4 PROCEDURE DESIGN

The arrangement, steps, and flow of a program are referred to as procedure design. It is concerned with how the commands and instructions are grouped together to form the entire program. The program's methodology is an event-driven programming language, which makes it more efficient in terms of execution, development time, storage utilization, and maintenance.

4.2 IMPLEMENTATION OF THE SYSTEM

This section is concerned with the new system's implementation and documentation. It is concerned with the physical requirements of the machines on which it can run, such as memory

capacity and operating system. It also includes a comprehensive user's manual or guide, as well as various screens (forms) as they appear to the user during run-time.

4.2.1 CHOICE OF PROGRAMMING

The programming language used in the development of this Android app is Android Studio 2.3, which is also coupled with device local storage for the backend.

Language support

Language: android studio and emulator

Local storage (Backend)

4.2.2 SOFTWARE AND HARDWARE REQUIREMENT

The following are the software requirement of the proposed system.

i. Operating system at least window XP and above
ii. 512mb RAM and above
iii. Android studio
iv. emulator
v. Local storage (Backend)

4.2.3 CHANGE OVER TECHNIQUES

The changeover technique used for this project's implementation is a parallel technique in which the new and old systems run concurrently.

4.3 SYSTEM DOCUMENTATION

4.3.1 DOCUMENTATION OF THE PROGRAM

i. GPS Tracker lite Application enabled
ii. You will install the develop application
iii. Prompt all instruction

4.3.2 OPERATING THE SYSTEM

i. Download Ilorin East Andriod file
ii. Locate the download file
iii. Install and find the icon on your android phone
iv. Then you can find or get the location on the map

4.3.3 MAINTAINING THE SYSTEM

i. Charge your phone regularly
ii. Always connect to internet
iii. Have strong antivirus
iv. Check your data and time

CHAPTER FIVE

SUMMARY, CONCLUSION AND RECOMMENDATION

5.1 SUMMARY

Someone might wonder why and how there could be a malicious app on my phone attempting to steal data. As previously stated, mobile phones are no longer just mobile phones. Users can download and install applications from a variety of sources onto their mobile devices. And not all of them can be relied on. According to a recent study on Android applications, approximately 20% of the 48,000 apps in the Android marketplace allow a third-party application access to sensitive or private information.

5.2 CONCLUSION

It has been a pleasure to work on this exciting and challenging project. The project provides knowledge about the most recent technology used in developing Android Location applications, which will be in high demand in the future. This will provide better opportunities and guidance in the future when developing projects on your own.

5.3 RECOMMENDATION

Based on these observations, it is suggested that the following (paraphrased) recommendations be made to three relevant parties:

i. Novice Smartphone Users: Download applications only from official markets because they are less likely to be malicious than APPs from unofficial markets. While it is impossible to guarantee that all applications found on official markets are clean, there is always the possibility that any malicious applications will be removed from the market if reported to the appropriate authorities.
ii. Device Manufacturers: Instead of relying on the smartphone operating system, smart-device manufacturers can provide users with pre-installed applications, giving them more control over their private information.
iii. Academia/Industry: Researchers from academia and industry form an open-source research community to develop open-source applications that will help to compensate for security vulnerabilities found in existing applications offered by official application markets.

REFERENCES

Brodeur, P (2012). Zero-permission android applications Part 2. Publication of Leviathan Security Group; accessed August 7, 2015, leviathansecurity.com/blog/zero-permission-android-applications-part-2

Jae-Jung, K. and Seng-Phil, H. (2010). A Method of Risk Assessment for Multi-Factor Authentication. Journal of Information Processing Systems, Vol.7, No.1.

Kevin, S. and Justin, H. (2008). An Examination of Information Security in Mobile Banking Architectures in Proceedings of Conference on Information Systems Applied Research, v1, Page 1.

Lineberry, A., Richardson, D.L. and Wyatt, T. (2010). These aren't the permissions you're looking for, 2010. presentations/Lineberry/DEFCON-18-Lineberry-Not-The-Permissions-You-Are-Looking-For.pdf

Egele, M., Scholte, T., Kirda E. and Kruegel, C. (2012). A survey on automated dynamic malware-analysis techniques and tools. ACM Comput. Surv 44(2), 6. Arizona, USA.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. and Witten, I.H. (2009). The WEKA data mining software: an update. SIGKDD Explorations, 11(1).

Moonsamy, V., Alazab, M. and Batten, L.M. (2012). Towards an understanding of the impact of advertising on data leak. Int. J. Secur. Networks, 7(3), 181–193. Inderscience Publishers, London, England.

Moonsamy, V. and Batten, L.M. (2012). Zero permission android applications—attacks and defenses. In: 3rd Applications and Technologies in Information Security, pp. 5–9. School of Information Systems, Deakin University Press, Australia

Park, Y., Lee, C. Lee, C., Lim, J., Han, S., Park, M. and Cho, S. (2012). RGB Droid: a novel response-based approach to android privilege escalation attacks. In: 5th USENIX conference on Large-Scale Exploits and Emergent Threats (LEET'12), 8 pp. Berkeley, California, USA .

Pearce, P., Felt, A. P., Nunez, G., and Wagner, D. (2012). Android: privilege separation for applications and advertisers in android. In: 7th ACM Symposium on Information, Computer and Communications Security, pp. 71–72. ACM Digital Library, Arizona, USA .

Peng, G., Shao, Y., Wang, T., Zhan, X. and Zhang, H. (2012). Research on android malware detection and interception based on behavior monitoring. Wuhan Univ. J. Nat. Sci.17.

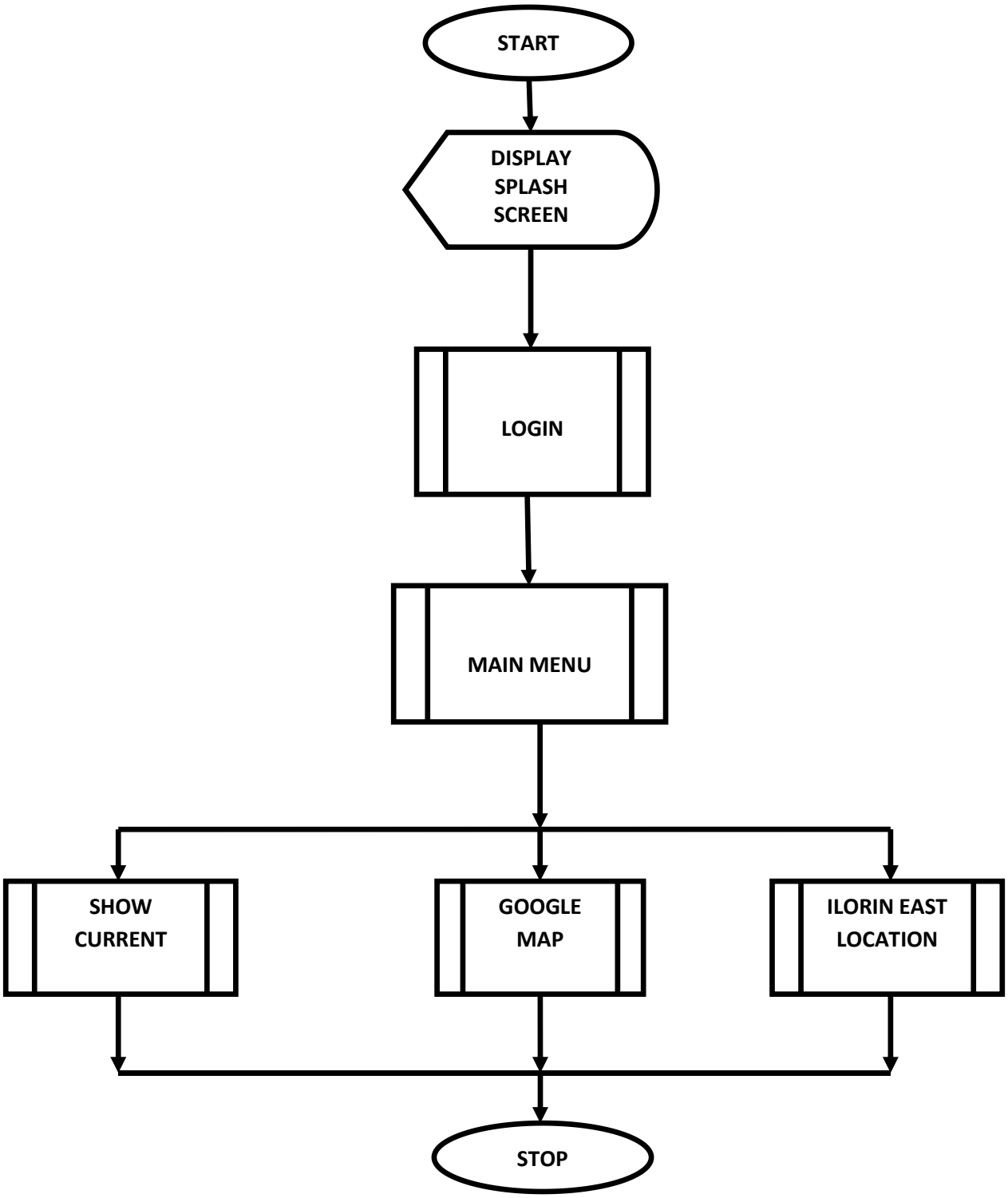
Rastogi, V., Chen, Y., Jiang, X. and Droid, C. (2013). Evaluating android anti-malware against transformation attacks. In: 8th ACM Symposium on Information, Computer and Communications Security (ASIACCS 2013), pp. 329–334. ACM Digital Library, Arizona, USA.

Rajnish, T., and Stephan, B. (2007). The Mobile Commerce Prospects: A Strategic Analysis of Opportunities in the Banking Sector. Hamburg University Press.

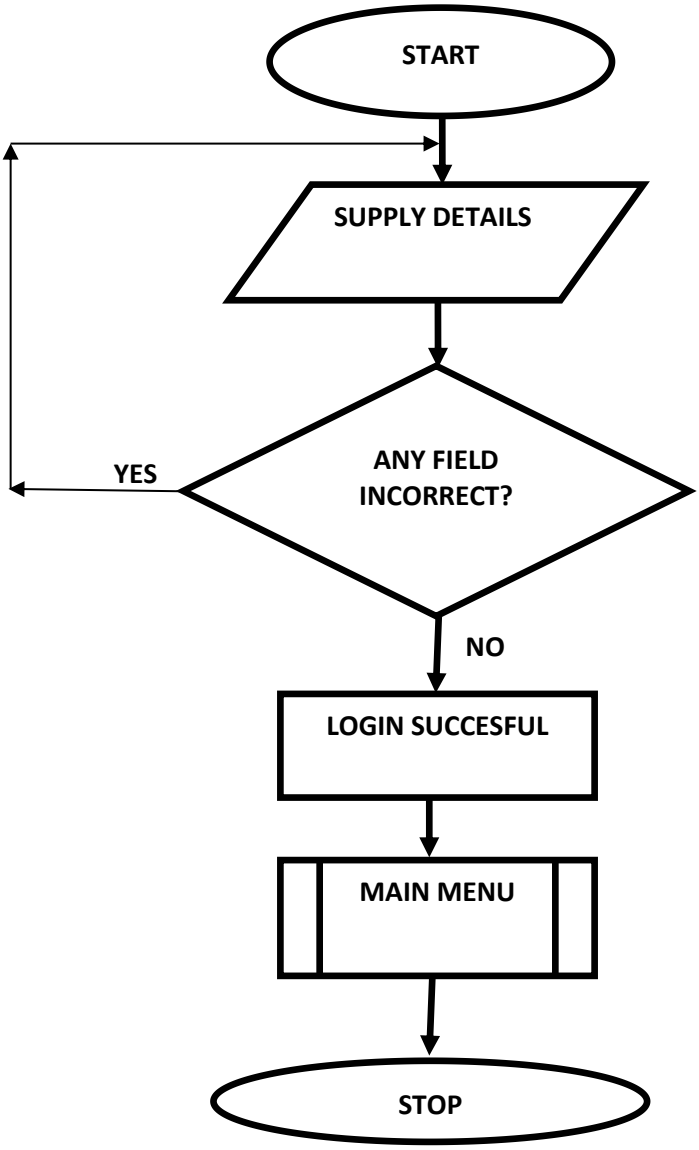
Riivari, J. (2005). Mobile banking: A powerful new marketing and CRM tool for financial services companies all over Europe. Journal of Financial Services Marketing.

Thomas, W., (1998). The Secure Remote Password Protocol in Proceedings of the Internet Society Network and Distributed System Security Symposium, pages 97-111, San Diego, CA, March.

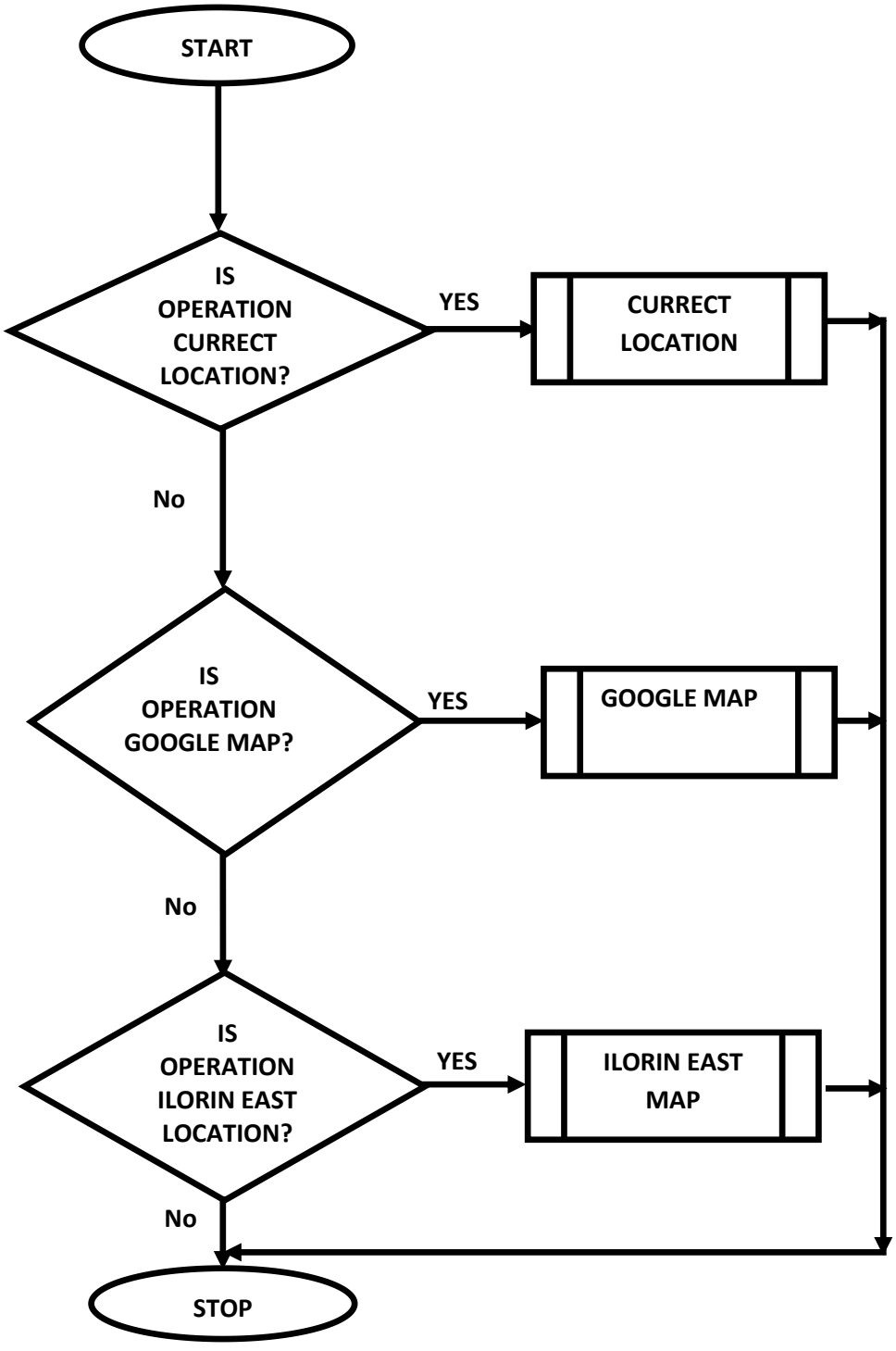
PROGRAM FLOW CHART



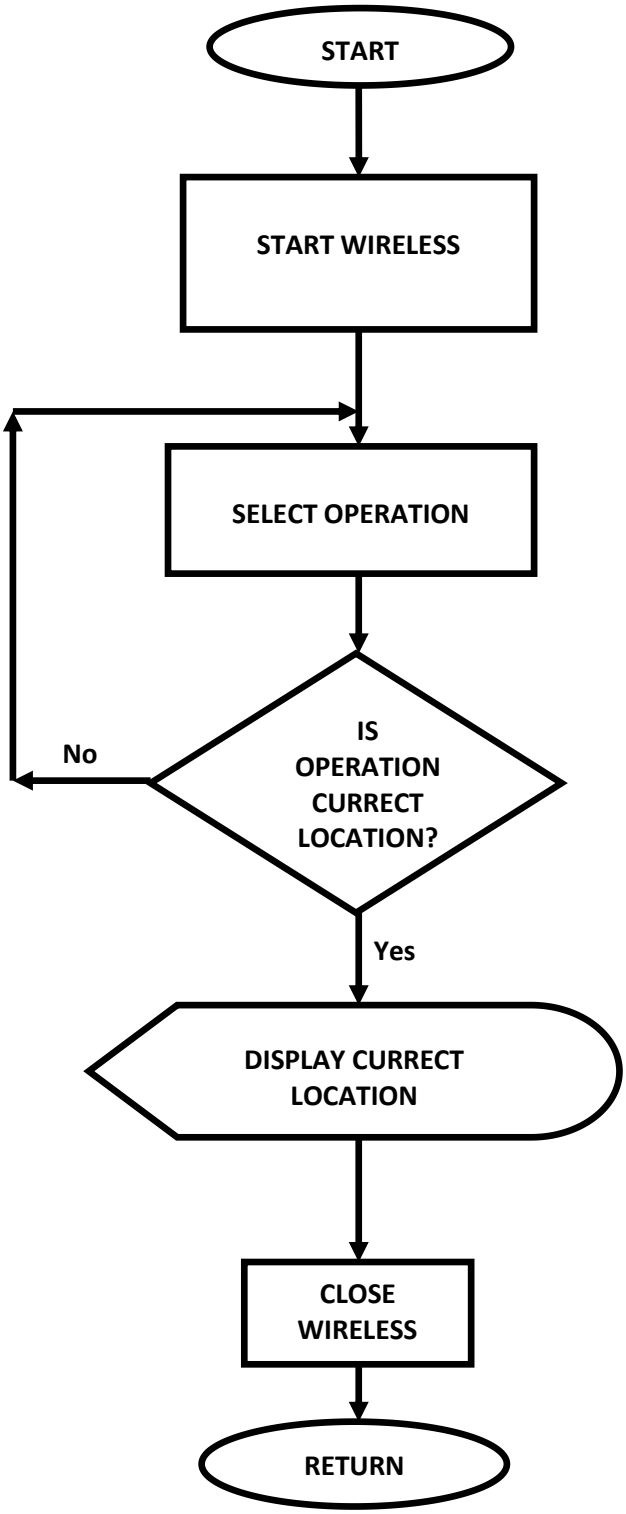
LOGIN FLOWCHART



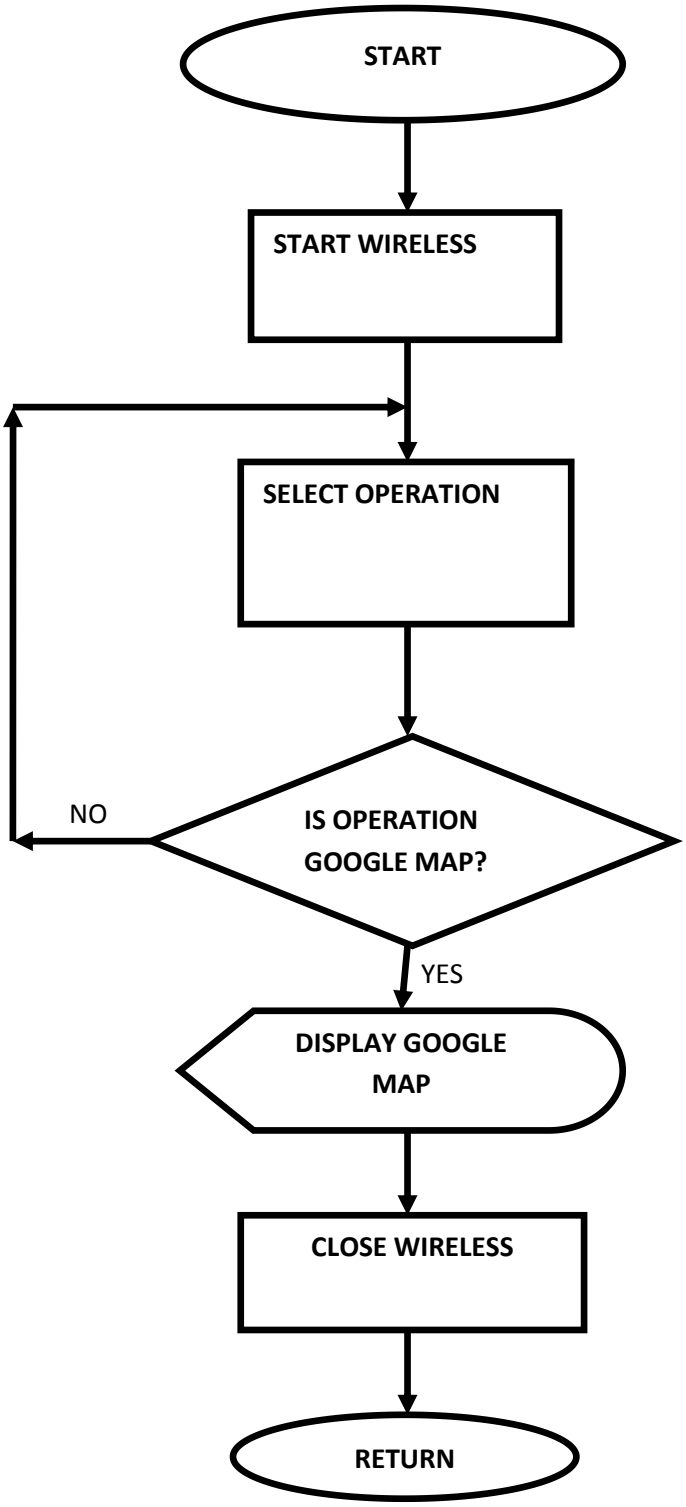
MAIN MENU FLOWCHART



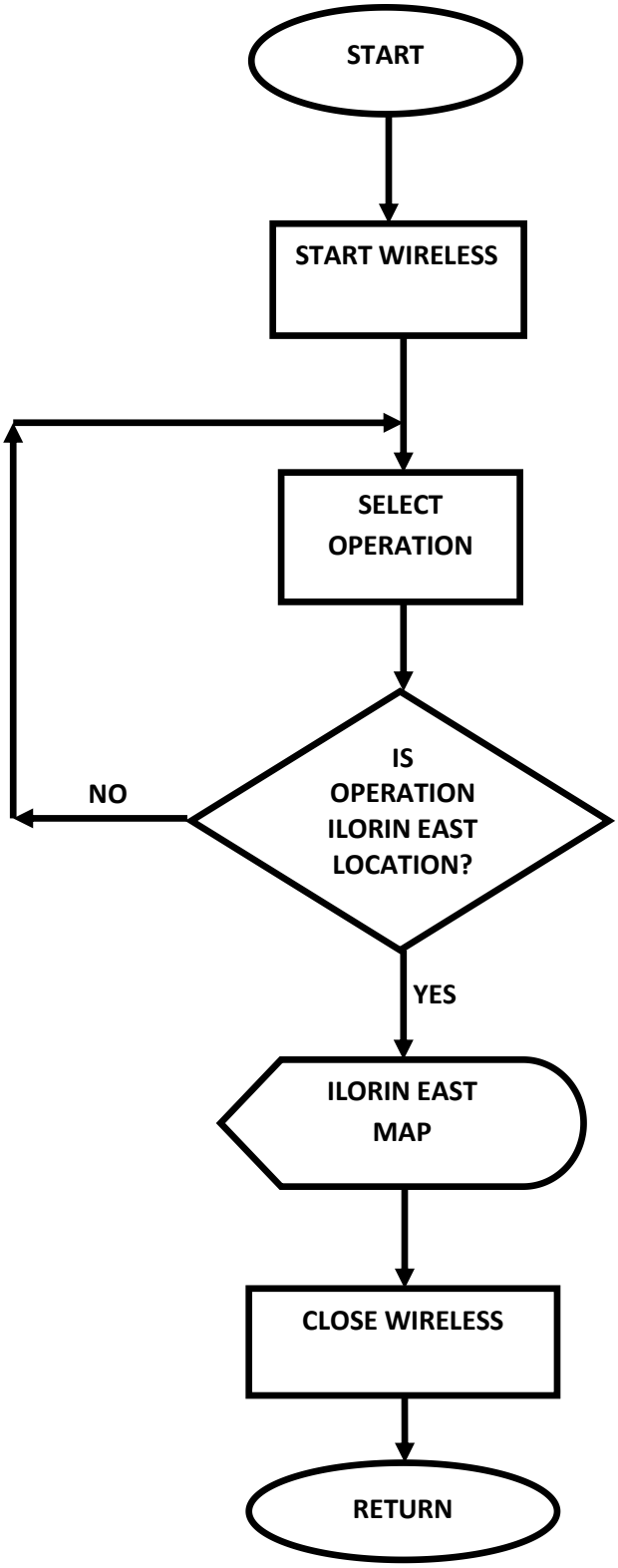
CURRECT LOCATION FLOWCHART



GOOGLE MAP FLOWCHART



ILORIN EAST LOCATION FLOWCHART



SOURCE CODE

```
package com.example.user.myapplication;


import android.location.Location;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v4.app.FragmentActivity;

import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;

public class MapsActivityextends FragmentActivityimplements OnMapReadyCallback {


private GoogleMapmMap;

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_maps);
// Obtain the SupportMapFragment and get notified when the map is ready to be used.
SupportMapFragmentmapFragment = (SupportMapFragment) getSupportFragmentManager()
        .findFragmentById(R.id.map);
mapFragment.getMapAsync(this);


   }


/**
    * Manipulates the map once available.
    * This callback is triggered when the map is ready to be used.
    * This is where we can add markers or lines, add listeners or move the camera. In this case,
    * we just add a marker near Sydney, Australia.
    * If Google Play services is not installed on the device, the user will be prompted to install
    * it inside the SupportMapFragment. This method will only be triggered once the user has
    * installed Google Play services and returned to the app.
    */
@Override

public void onMapReady(GoogleMapgoogleMap) {
mMap= googleMap;

// Add a marker in Sydney and move the camera
LatLngilorin = new LatLng(8.5, 4.55);
mMap.addMarker(new MarkerOptions().position(ilorin).title("Vehicle Location Tracking"));
mMap.moveCamera(CameraUpdateFactory.newLatLng(ilorin));


   }
@Nullable

public void handleNewLocation(Location location)
   {
//log.d(TAG, location.toString());
```

```
double currentLat= location.getLatitude();
double currentLon=location.getLongitude();

LatLngLatlat= new LatLng(currentLat, currentLon);
MarkerOptions options = new MarkerOptions()
            .position(Latlat)
            .title("I can Locate My Vehicle");
mMap.addMarker(options);
   }

   }


package com.example.user.myapplication;


import android.app.Activity;
import android.content.Context;
import android.net.wifi.WifiManager;
import android.os.Bundle;
import android.widget.CompoundButton;
import android.widget.Switch;
import android.widget.Toast;



public class TrackingActivityextends Activity {



@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_tracking);

      Switch toggle = (Switch) findViewById(R.id.wifi_switch);


toggle.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
public void onCheckedChanged(CompoundButtonbuttonView, booleanisChecked) {
if (isChecked) {
toggleWiFi(true);
Toast.makeText(getApplicationContext(), "Wi-Fi Enabled!", Toast.LENGTH_LONG).show();
            } else {
toggleWiFi(false);
Toast.makeText(getApplicationContext(), "Wi-Fi Disabled!", Toast.LENGTH_LONG).show();
            }
         }
      });
   }

public void toggleWiFi(booleanstatus) {
WifiManagerwifiManager = (WifiManager)  this.getSystemService(Context.WIFI_SERVICE);
if (status == true && !wifiManager.isWifiEnabled()) {
wifiManager.setWifiEnabled(true);
      } else if (status == false &&wifiManager.isWifiEnabled()) {
wifiManager.setWifiEnabled(false);
      }

   }
```

```
}
package com.example.user.myapplication;

import android.Manifest;
import android.app.ActionBar;
import android.app.Activity;
import android.app.DialogFragment;
import com.google.android.gms.maps.GoogleMap;
import android.app.Notification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.SharedPreferences;
import android.content.pm.ApplicationInfo;
import android.content.pm.PackageManager;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.net.ConnectivityManager;
import android.net.Uri;
import android.os.Bundle;
import com.google.android.gms.maps.MapView;
import android.preference.PreferenceManager;
import android.support.v4.app.ActivityCompat;
import android.support.v4.app.NotificationCompat;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewConfiguration;
import android.widget.Button;
import android.widget.CompoundButton;
import android.widget.Switch;
import android.widget.TextView;

import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.GooglePlayServicesUtil;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.MapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;
import com.example.user.myapplication.fragments.ChangePushTimeFragment;
import com.example.user.myapplication.services.PushLocationService;
import com.google.android.gms.maps.OnMapReadyCallback;


import java.lang.reflect.Field;

public class UbidotsActivityextends Activity implements
ChangePushTimeFragment.DialogListener {
// Preferences
private SharedPreferencesmSharedPreferences;
private SharedPreferences.EditormEditor;

// App info
private intmTimeToPush= 1;
private booleanmAlreadyRunning;
```

```
// Activity stuff
private Button mPushTimeButton;

private Switch mSwitch;
// Check connection
private ConnectionStatusReceivermReceiver= new ConnectionStatusReceiver();
private IntentFilteriFilter= new IntentFilter(ConnectivityManager.CONNECTIVITY_ACTION);

// Maps variables
private GoogleMapmGoogleMap;
private LatLngmUserLocation;
private Marker mUserMarker;

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_ubidots);
if (getActionBar() != null) {
// Modify the ActionBar
getActionBar().setDisplayShowHomeEnabled(false);
getActionBar().setHomeButtonEnabled(false);
getActionBar().setDisplayUseLogoEnabled(false);
getActionBar().setDisplayOptions(ActionBar.DISPLAY_SHOW_CUSTOM);
getActionBar().setCustomView(R.layout.action_bar);

// Set the options overflow menu always on top
try {
ViewConfigurationconfig = ViewConfiguration.get(this);
            Field menuKeyField = ViewConfiguration.class.getDeclaredField("sHasPermanentMenuKey");
if (menuKeyField != null) {
menuKeyField.setAccessible(true);
menuKeyField.setBoolean(config, false);
            }
        } catch (Exception e) {
e.printStackTrace();
        }
    }

// Instantiate layout widgets
mSwitch= (Switch) findViewById(R.id.toggleActivation);
mPushTimeButton= (Button) findViewById(R.id.push_times_button);

// Instantiate shared preferences
mSharedPreferences= PreferenceManager.getDefaultSharedPreferences(this);
mEditor= mSharedPreferences.edit();

// Get preferences variables
booleanfirstTime =
mSharedPreferences.getBoolean(Constants.FIRST_TIME, true);
mAlreadyRunning=
mSharedPreferences.getBoolean(Constants.SERVICE_RUNNING, false);
mTimeToPush= mSharedPreferences.getInt(Constants.PUSH_TIME, 1);

// Set the text at the left of the Switch
((TextView) findViewById(R.id.toggleText)).setText((mAlreadyRunning) ?
getString(R.string.enabled_text) : getString(R.string.disabled_text));
// Put the switch at its position
mSwitch.setChecked(mAlreadyRunning);

// If it's the first time the user access to the application we should put the preference
    // about it into false, so we can continue entering this activity immediately
```

```
if (firstTime) {
mEditor.putBoolean(Constants.FIRST_TIME, false);
mEditor.apply();
      }

// Check if Google Maps is installed
if (isGoogleMapsInstalled()) {
// Instantiate the fragment containing the map in the layout
        //mGoogleMap = ((MapFragment) getFragmentManager().findFragmentById(R.id.map)).getMapAsync();

        // Get the location given by the system
LocationManager location = (LocationManager) getSystemService(LOCATION_SERVICE);

// Create a location that updates when the location has changed
LocationListenerlocationListener = new LocationListener() {
@Override
public void onLocationChanged(Location location) {
mUserLocation= new LatLng(location.getLatitude(), location.getLongitude());
mGoogleMap.clear();
mUserMarker= mGoogleMap.addMarker(new MarkerOptions()
                    .position(mUserLocation)
                    .title(getString(R.string.location)));
mGoogleMap.moveCamera(CameraUpdateFactory.newLatLngZoom(mUserLocation, 17));
            }

@Override
public void onStatusChanged(String provider, intstatus, Bundle extras) {
            }

@Override
public void onProviderEnabled(String provider) {
            }

@Override
public void onProviderDisabled(String provider) {
            }
        };

// Set the listener to the location manager
if (ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED&&ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
// TODO: Consider calling
            //    ActivityCompat#requestPermissions
            // here to request the missing permissions, and then overriding
            //   public void onRequestPermissionsResult(intrequestCode, String[] permissions,
            //                                          int[] grantResults)
            // to handle the case where the user grants the permission. See the documentation
            // for ActivityCompat#requestPermissions for more details.
return;
        }
        location.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 0, 0, locationListener);
      }

// Update the text inside the button with the time to push
updatePushButton();

// Open the time dialog and
mPushTimeButton.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
ChangePushTimeFragment fragment = new ChangePushTimeFragment();
```

```
fragment.show(getFragmentManager(), "UBIDOTS_DIALOG_PUSH");
            }
        });

// Set the listener when clicked the switch button
mSwitch.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
@Override
public void onCheckedChanged(CompoundButtonbuttonView, booleanisChecked) {
if (!mAlreadyRunning) {
startRepeatingService();
mEditor.putBoolean(Constants.SERVICE_RUNNING, true);
                ((TextView) findViewById(R.id.toggleText)).setText(getString(R.string.enabled  text));
createNotification();
            } else {
mEditor.putBoolean(Constants.SERVICE_RUNNING, false);
                ((TextView) findViewById(R.id.toggleText)).setText(getString(R.string.disabled  text));
deleteNotification();
            }
mAlreadyRunning= !mAlreadyRunning;
mEditor.apply();
            }
        });
    }

// Start the service
public void startRepeatingService() {
startService(new Intent(this, PushLocationService.class));
    }

// Update the button to show the correct message
public void updatePushButton() {
        String buttonMsg;
if (mTimeToPush<60) {
buttonMsg = getResources()
                .getQuantityString(R.plurals.pushes_seconds, mTimeToPush, mTimeToPush);
        } else {
buttonMsg = getResources()
                .getQuantityString(R.plurals.pushes  minutes, mTimeToPush/ 60, mTimeToPush/ 60);
        }
mPushTimeButton.setText(buttonMsg);
    }

// Check if Google Maps is installed
public booleanisGoogleMapsInstalled()
    {
try {
ApplicationInfo info = getPackageManager().getApplicationInfo("com.google.android.apps.maps", 0);
return true;
        } catch(PackageManager.NameNotFoundException e) {
return false;
        }
    }

// Check if Google Play is available
public void checkGooglePlayAvailability() {
intrequestCode = GooglePlayServicesUtil.isGooglePlayServicesAvailable(this);
if (requestCode != ConnectionResult.SUCCESS) {
GooglePlayServicesUtil.getErrorDialog(requestCode, this, 1337).show();
        }
    }

// Create the notification to notify the user that the service is running
```

```
public void createNotification() {
      String ns = Context.NOTIFICATION_SERVICE;
      String notificationTitle = getString(R.string.notification  title);

      Intent intent = new Intent(this, UbidotsActivity.class);
PendingIntentpendingIntent = PendingIntent.getActivity(this, 0, intent, 0);

// We should use Compat, because we are using API 14+
NotificationCompat.Builder notification = new NotificationCompat.Builder(this)
            .setContentTitle(notificationTitle)
            .setSmallIcon(R.drawable.ic_stat_notify_logo)
            .setContentIntent(pendingIntent);

// Build the notification
NotificationnotificationCompat = notification.build();

// Create the manager
NotificationManagernotificationManager = (NotificationManager) getSystemService(ns);
notificationCompat.flags |= Notification.FLAG_ONGOING_EVENT;

// Push the notification
notificationManager.notify(Constants.NOTIFICATION_ID, notificationCompat);
   }

// Delete the notification
public void deleteNotification() {
      String ns = Context.NOTIFICATION_SERVICE;
NotificationManagernotificationManager = (NotificationManager) getSystemService(ns);
notificationManager.cancel(Constants.NOTIFICATION_ID);
   }

// Method from the dialog to handle the click
@Override
public void onDialogPositiveClick(DialogFragment dialog, intwhich) {
if (which == 0) {
mTimeToPush = 1;
      } else if (which == 1) {
mTimeToPush = 10;
      } else if (which == 2) {
mTimeToPush = 30;
      } else if (which == 3) {
mTimeToPush = 60;
      }

// Add the time to push in the preferences
mEditor.putInt(Constants.PUSH_TIME, mTimeToPush);
mEditor.apply();

// Update the text from the button
updatePushButton();
   }

   @Override
protected void onPause() {
this.unregisterReceiver(mReceiver);
super.onPause();
   }

   @Override
protected void onResume() {
this.registerReceiver(mReceiver, iFilter);
checkGooglePlayAvailability();
```

```
super.onResume();
   }

   @Override
public booleanonCreateOptionsMenu(Menu menu) {
getMenuInflater().inflate(R.menu.ubidots, menu);
return true;
   }

   @Override
public booleanonOptionsItemSelected(MenuItem item) {
intid = item.getItemId();
if (id == R.id.action_change_token) {
mEditor.putBoolean(Constants.SERVICE_RUNNING, false);
mEditor.putBoolean(Constants.FIRST_TIME, true);
mEditor.putString(Constants.VARIABLE_ID, null);
mEditor.putString(Constants.TOKEN, null);
mEditor.putString(Constants.DATASOURCE_VARIABLE, null);
mEditor.putInt(Constants.PUSH_TIME, 1);
mEditor.apply();

          Intent i = new Intent(this, MainActivity.class);
startActivity(i);
          finish();
      } else if (id == R.id.action_help) {
          String urlHelp = Constants.BROWSER_CONFIG.HELP_URL;
          Intent i = new Intent(Intent.ACTION_VIEW);
i.setData(Uri.parse(urlHelp));
startActivity(i);
      }

return super.onOptionsItemSelected(item);
   }

public class ConnectionStatusReceiverextends BroadcastReceiver {
      @Override
public void onReceive(Context context, Intent intent) {
intconnectionStatus = NetworkUtil.getConnectionStatus(context);
if (connectionStatus == NetworkUtil.TYPE_MOBILE ||
connectionStatus == NetworkUtil.TYPE_WIFI) {
mPushTimeButton.setEnabled(true);
mSwitch.setEnabled(true);
          } else {
if (mSharedPreferences.getBoolean(Constants.SERVICE_RUNNING, false)) {
deleteNotification();
             }
mPushTimeButton.setEnabled(false);
mSwitch.setChecked(false);
mSwitch.setEnabled(false);
mEditor.putBoolean(Constants.SERVICE_RUNNING, false);
mEditor.apply();
          }
      }
   }
}
package com.example.user.myapplication;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.ImageView;
```

```
public class VerificationActivityextends Activity {
private ImageViewmMapImage;

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity  verification);

mMapImage= (ImageView) findViewById(R.id.continue_map);

mMapImage.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
            Intent i = new Intent(VerificationActivity.this, UbidotsActivity.class);
startActivity(i);
            finish();
        }
    });
  }
}
package com.example.user.myapplication;

import android.os.Bundle;

import android.app.Activity;
import android.app.Fragment;
import android.app.FragmentManager;
import android.content.Intent;
import android.content.SharedPreferences;
import android.preference.PreferenceManager;
import com.example.user.myapplication.fragments.BrowserFragment;
import com.example.user.myapplication.fragments.MainFragment;


public class Main2Activityextends Activity implements MainFragment.MainFragmentButtonsInterface {
// We want to know if the user has logged in before
private SharedPreferencesmSharedPreferences;
private booleanmUserFirstTime;
private FragmentManagermFragmentManager;

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main2);

mFragmentManager= getFragmentManager();
// Get the preference to check if the user has logged in previously
mSharedPreferences= PreferenceManager.getDefaultSharedPreferences(this);
mUserFirstTime= mSharedPreferences.getBoolean(Constants.FIRST_TIME, true);

if (savedInstanceState == null) {
if (mUserFirstTime) {
MainFragmentmainFragment = new MainFragment();
getFragmentManager()
                    .beginTransaction()
                    .add(R.id.fragment_container, mainFragment)
                    .commit();
        } else {
            Intent ubidotsIntent = new Intent(this, UbidotsActivity.class);
startActivity(ubidotsIntent);
```

```
            finish();
        }
    }
}

// Method from MainFragment
@Override
public void onLoginButtonClick(Fragment fragment) {
    Bundle arguments = new Bundle();
BrowserFragmentlogInFragment = new BrowserFragment();

arguments.putString(Constants.URL, Constants.BROWSER_CONFIG.LOGIN_URL);
logInFragment.setArguments(arguments);

getFragmentManager().beginTransaction()
        .addToBackStack(null)
        .add(R.id.fragment_container, logInFragment)
        .commit();
}

// Method from MainFragment
@Override
public void onSignUpButtonClick(Fragment fragment) {
    Bundle arguments = new Bundle();
BrowserFragmentsignUpFragment = new BrowserFragment();

arguments.putString(Constants.URL, Constants.BROWSER_CONFIG.SIGN_UP_URL);
signUpFragment.setArguments(arguments);

getFragmentManager()
        .beginTransaction()
        .replace(R.id.fragment_container, signUpFragment)
        .addToBackStack(null)
        .commit();
}
}
```