UNIVERSITY OF NAIROBI

SCHOOL OF COMPUTING AND INFORMATICS

A MODEL FOR LOCATION BASED SERVICES
APPLICATION DEVELOPMENT

BY

LEONARD GICHUHO
P58/72874/2009

Supervised By

DANIEL ORWA

October 2012

Submitted in partial fulfillment of the requirements for the Master of Science in Computer Science
Declaration

The thesis, as presented in this report, is my original work and has not been presented for any other University award.

Sign: __________________________ Date: 22/10/2012

LEONARD GICHIUHO WAHU

P58/72874/2009

This project has been submitted as partial fulfillment of the requirements for the Master of Science degree in Computer Science of the University of Nairobi with my approval as the University supervisor.

Sign: __________________________ Date: 22/10/2012

DANIEL ORWA

Project supervisor

School of Computing and Informatics

University of Nairobi
Abstract

Location Based Services are one of the most rapidly expanding fields of the mobile communications sector, with a vast application range. A location based service is defined as an information or entertainment service, accessible with mobile devices through a mobile network and utilizing the ability to make use of the geographical position of the mobile device. Location based services draw upon a combination of location acquisition technologies, web services and spatial databases. They use knowledge of a mobile device's location to offer value to the mobile subscriber or a third party.

Even though location based services are a hot topic, there are very few publicly available frameworks for the rapid development of location based service applications. This thesis attempts to address this shortfall and come up with a theoretical framework for the rapid development of location based service applications. This is achieved by studying the strengths and weaknesses of, the few, currently available frameworks. The study is further complimented by carrying out a survey on the factors that influence the choice of a framework by ICT professionals. The study then proposes an open framework for rapid development of mobile location based service applications. It describes positioning prospects using GSM networks, Wi-Fi networks and the GPS system. It further describes a location aware application prototype built based on the proposed framework. This demonstrates that building applications based on the proposed framework is feasible. The said prototype was build based on the Rapid Application Development (RAD) methodology. The prototype as currently build has room for improvement. Further work on the prototype could include making it more interactive by letting users add their own points of interest on the map. The prototype could also be extended to allow users add location based reminders. To allow users to add data to the prototype would mean that the framework would also need to be extended to include a web based database for storing this information centrally. This is achievable.
Acknowledgements

I would like to thank Mr. Daniel Orwa and Mr. Eric Ayienga for their direction, assistance, and guidance. In particular, Mr. Orwa’s recommendations and suggestions have been invaluable for this thesis. Further, I am indebted to many of my fellow classmates who in one way or another aided me in the pursuit of my M.Sc.

Lastly, I owe my deepest gratitude to my immediate family for staying the course and putting up with my late nights and absent weekends. Special thanks to the three outstanding pillars in my life, Milka Wamaitha, Anne Wambui and baby Tanei Gichuho without whose support I would be lost.
Table of Contents

DEFINITION OF TERMS vii
LIST OF FIGURES viii
LIST OF TABLES x

CHAPTER 1: INTRODUCTION 1
1.1. Background 1
1.1.1. Problem statement 2
1.1.2. Objectives 3
System Development Objectives 3
Research Objectives 3
1.1.3. Research outcome and their significance to key audiences 3
1.1.4. Assumptions and limitations 3
1.1.5. Contents of the Document 3

CHAPTER 2: LITERATURE REVIEW 5
2.1 Location Based Services 5
2.2 Related Work 7
2.3 Location Based Services Frameworks 7
2.3.1 Place Lab 7
Problems with the PlaceLab approach 12
2.3.2 A framework for developing mobile location based application, from the University of Ljubljana. 13

CHAPTER 3: METHODOLOGY 16
3.1 Research Design 16
3.1.1 Target population and sampling technique 16
Data collection 16
Primary data for the framework design: 16
Secondary data for the prototype 17
3.1.2 Mapping of objectives to research methodology 17
3.2 Data Processing and Analysis 18
3.3 Prototype development methodology 23
3.4 Requirements Planning (RP) 27
3.4.1 Research Current Situation

3.4.2 Define Requirements

3.4.3 Finalize Requirements

3.5 User Design (UD)

3.5.1 Produce Detailed System Area Model

The Context Diagram

Level Zero Diagram

3.5.2 Develop Outline System Design

Database Design

Screen Design

Function Design

Get current location

Get POI information

Get proximity alerts

Get current address

3.6 Proof of Concept

CHAPTER 4: IMPLEMENTATION

4.1 Theoretical Framework

4.2 Rapid Construction (RC)

4.1.1 Prepare for Rapid Construction

4.2 Construct System

4.2.1 Sample code

4.2.2 Sample Screens

4.3 Generate Test Data and System Documents

4.4 Verify System Construction

CHAPTER 5: CONCLUSION & RECOMMENDATIONS

5.1. Evaluation of Objectives

5.2. Conclusion

5.3. Limitations of the Research

5.4. Recommendations & Further Work
DEFINITION OF TERMS

GSM - Global System for Mobile.

**World Geodetic System** - A standard for use in cartography, geodesy, and navigation. It comprises a standard coordinate frame for the Earth, a standard spheroidal reference surface (the datum or reference ellipsoid) for raw altitude data, and gravitational equipotential surface (the geoid) that defines the nominal sea level.

LBS - Location Based Services.

Wi-Fi - Wireless Fidelity. Is a trademark of the Wi-Fi Alliance. A Wi-Fi enabled device such as a personal computer, video game console, smartphone, or digital audio player can connect to the Internet when within range of a wireless network connected to the Internet. It is based on the IEEE 802.11 standard.

GPS - Global Positioning System. A space-based global navigation satellite system (GNSS) that provides reliable location and time information in all weather and at all times and anywhere on or near the Earth when and where there is an unobstructed line of sight to four or more GPS satellites.

A-GPS - Assisted GPS. Is a system which can, under certain conditions, improve the startup performance, or TTFF (Time To First Fix) of a GPS satellite-based positioning system. It is used extensively with GPS capable cellular devices.

SOAP - Simple Object Access Protocol, a protocol specification for exchanging structured information in the implementation of Web Services in computer networks.

Android Activity - This is the equivalent of a form in other visual programming languages such as Visual basic.

Proximity Alerts - These are alerts that get generated when the user is physically located near a specific Point Of Interest (POI)

Geocoding - The process of finding associated geographic coordinates (often expressed as latitude and longitude) from other geographic data, such as street addresses, or postal codes. Reverse geocoding is the opposite: finding an associated textual location such as a street address, from geographic coordinates.
LIST OF FIGURES

Figure 1 The Place Lab Architecture. Source: placelab ..............................................................9
Figure 2 The Mapper Hierarchy. Source: placelab ...................................................................10
Figure 3 An Excerpt of the Tracker Hierarchy. Source: placelab .............................................10
Figure 4 Ljubljana University framework architecture. Source: Ljubljana University ..........13
Figure 5 Framework components. Source: Ljubljana University .............................................14
Figure 6 Respondents’ years of experience in ICT. Source: research .....................................20
Figure 7 Impact of frameworks on productivity. Source: research ..........................................20
Figure 8 Respondents views on the importance of the number of components in a framework. Source: research.................................................................21
Figure 9 Respondents views on the complexity of a framework. Source: research ...............21
Figure 10 Gender distribution of the respondents. Source: research.....................................22
Figure 11 Respondents previous experience with frameworks. Source: research ..................22
Figure 12 A comparison between traditional development and RAD. Source: CASEMaker 25
Figure 13 Context Diagram. Source: Research. ....................................................................29
Figure 14 Level zero diagram. Source: Research. ................................................................30
Figure 15 ERD model of the prototype’s database. Source: Research.................................32
Figure 16 Map and address screen. Source: Research ............................................................33
Figure 17 Lecturer details screen. Source: Research..............................................................34
Figure 18 Lecturers’ department screen. Source: Research....................................................34
Figure 19 POI details screen. Source: Research.................................................................35
Figure 20 Arbitrary notifications screen. Source: Research....................................................35
Figure 21 Get Current Location. Source: Research.................................................................37
Figure 22 Get POI Information. Source: Research.................................................................38
Figure 23 Get Proximity Alerts. Source: Research.................................................................39
Figure 24 Get current address. Source: Research.................................................................40
Figure 25 Diagram depicting our conceptual framework architecture. Source: Research ....42
Figure 26 Framework components. Source: Research............................................................43
Figure 27 Proximity Alert Screen. This is the screen displayed once a user enters the radius prescribed as a Point Of Interest (POI). It informs the user that he/she is near a POI. Source: Research. 

Figure 28 Department details screen. This screen displays CBPS departments stored in the prototype’s database. Source: Research. 

Figure 29 Lecturer details screen. This screen displays CBPS lecturers stored in the prototype’s database. Source: Research. 

Figure 30 Cafeteria Meals screen. This screen displays CBPS cafeterias with the meals available on each day. Source: Research. 

Figure 31 Lecturer Availability Times. Screen displays CBPS lecturers’ availability times i.e. times when they are available for consulting. Source: Research. 

Figure 32 Our accepted updates on Google Map Maker 1. 

Figure 33 Our accepted updates on Google Map Maker 2.
LIST OF TABLES

Table 1: Platform configurations that Place Lab currently runs on. Source: placelab ........... 11
Table 2: Mapping of objectives to research methodology ...................................................... 17
Table 3: Tabulation of research findings. Source: Research ................................................... 19
Table 4: Project costs. Source: Research. .............................................................. 28
CHAPTER 1: INTRODUCTION

1.1. Background
Location Based Services are one of the most rapidly expanding fields of the mobile communications sector, with a vast application range. A location-based service (LBS) is defined as an information or entertainment service, accessible with mobile devices through a mobile network and utilizing the ability to make use of the geographical position of the mobile device. LBS may also be defined simply as applications that utilize users’ current locations (Weinberg 2004). LBS applications include:

- Asset tracking: enable organizations or individuals to track the locations of their assets, e.g. companies tracking mobile employees.
- Way-finding: help people find their way, e.g. by showing their positions on maps or by providing directions.
- Traffic management: aggregate many users’ locations and then provide driving routes based on traffic patterns discerned.
- Emergency response: enable emergency responders to find out the locations of people requesting emergency response.
- Mixed-reality games: involve people playing games in the real physical world.
- Targeted shopping: provides users details about nearby shopping possibilities.
- Location-aware instant messaging: augment traditional instant messaging services with location information.
- Location-aware billing: enable services to bill users based on where they are using the service, e.g. mobile phone roaming.

From a commercial perspective, the turn of the millennium until the end of 2007 LBS had promised much, but delivered little in terms of commercial applications. Even in Japan and South Korea, the most advanced markets for commercial LBS, usage was disappointing. In 2008 and 2009 the LBS market underwent a dramatic change and it is fair to say that it will never be the same again. Now location enablement is rapidly becoming a de facto feature of
mobile applications - so much so that by 2014 location based services and location enabled apps will cease to be a special interest category (Gibson & Holden 2010).

Location-based services are a large and imperative subset of the overall set of ubiquitous computing applications, and have already demonstrated the advantages gained from the ability to perceive the surrounding environment. Such applications however remain difficult to develop and deploy, with no widely accepted programming model available. Programmers are often required to write large amounts of code and interact with sensor and actuator devices at a low level in order to develop relatively simple applications. We sought to explore the world of Location-based services and recommend a conceptual framework that was friendlier to programmers in terms of the time and effort required to develop LBS applications.

1.1.1. Problem statement
The study was designed to formulate a suitable framework, then build a proto-type based on it (as a proof-of-concept), for location based services for handheld devices. The resulting proto-type sought to address the following problems:

- The lack of easily accessible information about lecturers' offices within departmental buildings in the University of Nairobi
- The lack of easily accessible information regarding lecturer availability times i.e. the time slots within which a lecturer may be privately consulted by students
- The lack of real time information about meals being served at the campus cafeterias within the University of Nairobi
- The lack of an easily accessible forum for students/visitors to give feedback about a location or a service within the campus
- The lack of a real time customized map to guide visitors navigating the University of Nairobi
- The lack of an easily implementable framework to build location based applications
1.1.2. Objectives

**System Development Objectives**

- To provide users with information based on their present location. This included information about location of lecture halls, cafeterias, laboratories among other points of interest.

- To provide students/visitors with an alert when they approach a point of interest.

**Research Objectives**

- To compare and contrast available location based application frameworks.

- To design a simple framework/model for location based services application development.

- To develop a prototype based on our framework, as a proof-of-concept.

1.1.3. Research outcome and their significance to key audiences

- A better understanding of the available frameworks for location based application development.

- The development of a prototype based on our framework. The prototype sought to empower its users by offering them relevant information regarding their surroundings where and when they need it.

1.1.4. Assumptions and limitations

- We limited our prototype coverage area to the University of Nairobi College of Biological and Physical Sciences (CBPS).

- We also assumed that students, staff and visitors at CBPS possess mobile devices that have a GPS receiver preferably coupled with Wi-Fi capability.

- The said mobile devices must have been running the android operating system. This was because the prototype was built on this platform.

1.1.5. Contents of the Document

The rest of this document is structured as below: Chapter two is the literature review. It presents the current work that has been done by other researchers within the problem domain. This chapter looks at the concept of location based services as a whole and the technology
behind it. It goes further and looks at the available frameworks for location based service
design. The chapter concludes by outlining the strengths and weaknesses of the identified
frameworks, thus setting the pace for a new framework for location based services
application design. Next is the chapter on the methodology that was used in undertaking the
research. This chapter covers the research design, the sampling techniques employed, data
collection, requirements planning and user design. The section on user design covers the
context diagram, the database design, screen design and function design. Chapter four details
the implementation phase of the research. The chapter introduces the tools and environment
used to develop the prototype. It walks the reader through the process of developing the
prototype as it depicts both sample code and the input and output screens of the prototype.
Test data is also discussed here as are the testing methods used to test the prototype. Finally
comes the conclusion and recommendations chapter which summarizes the findings and
experiences of the research effort as a whole.
CHAPTER 2: LITERATURE REVIEW

This chapter documented the available literature concerning the problem domain. The implication was that the researcher devoted sufficient time to reviewing research already undertaken on related problems. This was done to find out what data and other materials are already available from earlier research, and identify any gaps that the present research may fill.

2.1 Location Based Services

Location based services (LBS) draw upon a combination of location acquisition technologies, web services and spatial databases. LBS use knowledge of a mobile device's location to offer value to the mobile subscriber or a third party. Various wireless location acquisition technologies exist for implementing LBS. These include (Wang et al 2008):

- Basic positioning methods such as:
  - Dead reckoning
  - Proximity sensing
  - Trilateration
  - Multilateration
  - Triangulation
  - IEEE 802.11 (Wi-Fi)
  - Bluetooth

- Satellite positioning systems. Include:
  - Global Positioning System (GPS)
  - Galileo
  - A-GPS

- Positioning in 3G networks. Includes below technologies:
  - Mobile-based technologies: Cell-ID, time advance
  - Network-based technologies: TDOA, AOA
  - Mobile-assisted technologies: A-GPS, AFLT, OTD.

The most widely recognized system, by and large, is the Global Positioning System (Wang et al 2008). It comprises of 24 satellites orbiting around the Earth and enables us to determine our location anywhere on our planet with an accuracy of roughly 10 meters. To determine their location, GPS receivers need to obtain signals from at least four different satellites. Four satellites define four spheres defined by the difference between send time (from satellite) and
reception time (GPS receiver). The intersection of these spheres presents the current location of the GPS receivers.

Mobile-based technologies: Mobile devices, especially mobile phones are gaining on their popularity and have become a part of the average person’s life. Most of these mobile phones are connected to GSM networks. GSM networks are cellular networks by design and every cell in a cellular network has its own base station with a unique base station identifier. Since our mobile phones always know which base station they are connected to, we can use this information to determine the location of our mobile phone (Wang et al 2008). However, because areas covered by single base station vary in both shape and size it is hard to determine mobile device location accurately. While the distances between base stations in urban regions are between 200 and 500 meters, they can grow to a couple of kilometers in rural regions. Accuracy can be increased by considering time advance and hand over time information, but this can only be obtained from the mobile network operators who usually charge for such services (Andrej K et al 2006). Positioning based on Bluetooth is still at its infancy. The challenge with Bluetooth positioning is that extra hardware needs to be deployed in the form of Bluetooth beacons. A mobile device then approximates its position within a room based on the location of the Bluetooth beacon it can communicate with (Hallberg J et al 2003). Wi-Fi positioning on the other hand, uses the same principle as that of triangulation in determining location. Skyhook Wireless, a technology company in the USA, has created a large database of Wi-Fi access point locations which they use to determine a user’s location based on the Wi-Fi access point the user is connected to (Skyhook 2011).

Positioning based on Wi-Fi, GSM and/or Bluetooth has one important advantage over GPS. Besides determining our location outside, in open areas, it can also help us with determining our location inside buildings. This is because GPS performs poorly indoors. On the other hand GPS performs excellently, outdoors, even in rural areas.

In our case, we primarily used GPS, Wi-Fi and GSM (Cell-Id). The choice of which one to use in a given instance was transparent to the user as the application automatically determined this. Nonetheless, for the most accurate location determination users should activate both GPS and Wi-Fi on their devices.
2.2 Related Work

The launch of the Apple iPhone in 2007 and the announcement that it would support third party applications changed the way developers and users viewed applications for the mobile phone. These events also introduced a large number of consumers to the existence and capability of location enabled applications and associated services. Fast forward that to the year 2011 where LBS are part of everyday life for many people in the developed countries. Applications such as facebook places, gowalla and foursquare are currently among the most popular location based applications in the developed countries. All three applications are more or less social networking applications. Again all three are proprietary and thus the architecture behind them is not in the public domain.

Another most interesting application is Places, which runs on android mobile devices. With this application you can search for places that are near your current location (Google Places 2011). However, preliminary tests with this application indicated that it searches for nearby places using the keyword supplied. For instance, our search for nearby hospitals had a school named 'Hospital Hill Primary School' appearing in the results. This implies that it actually searches for the word hospital instead of the institution hospital. Such a model is thus capable of giving very misleading results. Again the underlying framework for this application is not in the public domain.

2.3 Location Based Services Frameworks

2.3.1 Place Lab

Place Lab, by Intel Research Seattle, is one of the earlier pioneers in location based applications. Place Lab utilizes Wi-Fi access points, GSM network base stations and Bluetooth devices to determine location. It relies on pre-defined spatial databases, holding radio beacon locations (GPS coordinates of GSM network base stations, Wi-Fi access points and stationary Bluetooth devices). It thus assumes that one has collected spatial data and loaded it onto a database. Building such a database is no mean feat, both in terms of effort and cost. Such data can be acquired by war-driving. War-driving is essentially a process of collecting radio beacon locations by driving around cities equipped with different wireless receivers and a GPS device. Every time a radio beacon is discovered, the current location, signal strength and beacon identification number are logged. These logs are then filtered and inserted into beacon location databases (Lamarca A. et al 2003). Place Lab falls into the
general category of fusion architectures. Conceptually, fusion architecture refines raw streams of data from possibly many sources into a sequence of high-level inferences. Fusion architectures have a place in wide-scale defense systems, context-aware computing, and sensor networks. The purpose of such architecture is to separate the different aspects of the data processing into logical algorithmic components that can be independently improved, replaced, or composed. A dominant theme in fusion architectures is the pipelining, stacking, or layering of the components into a sequence of processing stages that successively refine a data stream into inferences. Place Lab borrows from ActiveCampus, a server-centric database-oriented fusion architecture for extensible, integrated application design. It employs a multi-stage mediator-observer design pattern to create the stages of processing. The event-driven database model provides for decoupling of components yet tight integration: the storing of a lower-level data element into the database triggers an event that causes the next stage of processing to begin; the storing of that stage’s results triggers another event that starts the next stage of processing. New inference components can be added by registering for the appropriate events. Normalization of the database tables supports incremental extension of the objects being modeled and the components that process those objects. The Context Toolkit is a small set of generic base classes from which a programmer can derive specific subclasses for the development of a streaming peer-to-peer networked context-aware application. The primary classes are a Context Widget, which abstracts away a sensor as a data stream, a Context Interpreter, which provides a mapping of one type of context element to another and a Context Aggregator, a context widget that fuses data streams from multiple widgets (Lamarca A. et al 2003). The data element streamed between widgets is an aggregation of generic key-value pairs. The Context Toolkit’s primary value lies in the generic services of storing and forwarding data between peers, as well as the flexible interoperability of the classes that are developed by the programmer.

Place Lab follows the general lines of a layered event-streaming fusion architecture. Like ActiveCampus, it makes heavy use of the hybrid-mediator design pattern. Place Lab’s components map on to those of the Context Toolkit. What distinguishes it from these systems is its focus on location sensing, client-side inference, and the expected presence of the application itself on the client. These unique characteristics yield a distinct set of requirements, and enable the deployment of not only a toolkit, but also an infrastructure that can infer location on a wide variety of today’s client computing platforms (LaMarca A et al 2003).
The research community is particularly active in three aspects of location-based computing: sensing, sensor fusion in positioning algorithms, and applications. There is also substantial innovation in the personal computing devices that might deploy location based applications.

Figure 1 The Place Lab Architecture. Source: placelab

Spotters are the components that abstract away the hardware that senses the environment. The four standard spotters implemented in Place Lab are 802.11 (Wi-Fi), GSM, Bluetooth, and GPS. Mappers are static databases of information that are used by trackers to retrieve location information for spotter measurements. The data stored in a mapper always includes a location coordinate, but may include other useful information such as coverage radius. The data to populate a mapper can come from a mapping database, or user-defined files containing known beacon locations. Mappers can also be populated by war-driving data. Mappers that reside on different systems will require a different method of persistent storage. For example, a mapper using the Java Database Connectivity (JDBC) or Java Database Manager (JDBM)
libraries would work well on a PC, but would not function on a mobile phone. The Mapper interface defines the methods a class must implement to insert, query, and retrieve data from the persistent store. The AbstractMapper class implements the Mapper interface to provide a superclass for all Mapper classes to extend. The superclass also implements caching of data for quick accesses.

Figure 2 The Mapper Hierarchy. Source: placelab

Trackers are the system components that produce position estimates. The tracker utilizes the stream of spotter observations as Measurement objects, together with persistent data from Mappers, to calculate a single position Estimate. In doing so, Trackers may perform sensor fusion by combining data from multiple types of sensors with different characteristics.

Figure 3 An Excerpt of the Tracker Hierarchy. Source: placelab
The Tracker class defines the methods that all trackers must implement. Each tracker must implement a method to update its position estimate when receiving a new spotter measurement, filtering out any unwanted measurements that may be provided.

**Application Adapter – Façade:**

When Place Lab is instantiated, it must be adapted to the platform, available sensors, and the application. In a few cases runtime checks are used to detect the available sensors, but generally the configuration is determined by how the Place Lab adapter object is sub classed and instantiated. Place Lab currently runs in many different platform configurations, these are:

<table>
<thead>
<tr>
<th>Operating Systems</th>
<th>Architectures</th>
<th>802.11</th>
<th>GSM</th>
<th>Bluetooth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows XP</td>
<td>x86</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Linux</td>
<td>x86, ARM, XScale</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mac OS X</td>
<td>Power PC</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Windows Mobile</td>
<td>ARM, XScale</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Place Lab provides five interfaces for communicating location information to applications:

- **Direct Linking** - Applications may communicate with the PlaceLab object directly. For applications that use a preconfigured Place Lab object, they can invoke a single method to start the location tracking service. The application can use either an asynchronous or synchronous interface to obtain position estimates from Place Lab.

- **Daemon** - For some applications, it may be desirable or necessary to not link them directly to Place Lab. To support such applications, Place Lab can be run as a daemon and be queried via a simple HTTP interface. This interface allows programs written in a wide range of languages and styles to use Place Lab.

- **Web Proxy** - A web proxy interface uses Place Lab functionality to support location-enhanced web services by augmenting outgoing HTTP requests with extension headers that denote the user’s location. By configuring web browsers to use this proxy (in the same way one uses a corporate firewall’s proxy), web services that understand the extension headers can provide location-based service to the user.

- **JSR 179** - To support existing Java location-based applications, Place Lab can provide location through the JSR-179 Java location API.

- **NMEA 0183** - Place Lab provides a virtual serial-port interface that mimics an external GPS unit by emitting NMEA 0183 navigation sentences in the same format generated by GPS hardware. Since many applications (e.g., Microsoft MapPoint) already understand NMEA, they can seamlessly take advantage of location functionality developed using Place Lab (which might operate indoors, unlike GPS).

**Problems with the PlaceLab approach**

To use PlaceLab there must exist a number of entries in a centralized database for landmarks in the user’s locality. For users in well developed, technologically advanced areas this may not be a problem but for the majority of people this is unlikely. It is thus crucial for PlaceLab to quickly develop a large community base to bootstrap the database and encourage use of the platform. This community already exists somewhat in the form of War
Drivers who contribute to global databases of Wi-Fi access points such as Wigle.net. Information from this database integrates easily with PlaceLab which partly solves this problem. Some functionality such as altitude and estimation accuracy is currently not provided although these issues will very likely be addressed in future versions.

### 2.3.2 A framework for developing mobile location based application, from the University of Ljubljana.

The framework comprises of three elements (Fig 4): location aware client (a mobile device), the application server and the database server that holds spatial data.

![Figure 4 Ljubljana University framework architecture. Source: Ljubljana University](image)

The spatial data is stored in a relational database on the database server. It contains both the locations of radio beacons with their associated geographical positions and user defined data such as maps, object location information, path information etc. Because of its simple design, the relational database can be easily extended to hold any kind of location data, meta-data or even sensor data.
The application server provides web services for mobile clients and communicates with the database server. Client communication is based on standard SOAP messages while native communication is used when querying the database server.

The mobile device obtains either its geographical coordinates or a list of radio beacons (base stations, access points or Bluetooth devices) in its vicinity from one of its wireless receivers (GPS receiver, Wi-Fi receiver, Bluetooth receiver). If the mobile device obtains a list of radio beacons it first encapsulates these data in a SOAP envelope and sends it to the appropriate web service on the application server. The web service executes a query against the spatial database and returns the approximate location to the mobile client. Now the mobile application can query the web services for interesting information based on its current location. Again the query is encapsulated in a SOAP envelope and sent to the web service, which queries the spatial database server for the required data and returns it to the client. The mobile device displays received location information over a map if one is available from the web service. Location information can contain interesting places, objects, paths etc. in the area around the current location (ANDREJ K et al. 2006).

Figure 5 Framework components. Source: Ljubljana University

Fig 5 above shows how the Ljubljana University framework interconnects with other components of the system. The framework abstracts methods for access to I/O devices (most...
commonly different sensors) via serial or Bluetooth interfaces. It also provides methods for obtaining our current location and for obtaining location information. Methods for accessing web services are provided and also used internally for obtaining location information from the application servers. The framework is utilized by a mobile application which does not have to deal with the implementation details of obtaining location information, sensor communication and web services access.
CHAPTER 3: METHODOLOGY

This chapter describes the methodology and approach adopted for the research. Research methods may be understood as all those techniques that are used for conducting the research. In other words, all those methods which are used by the researcher during the course of studying his research problem are termed as research methods.

Our research methodology was two-fold: The first was for investigation of LBS frameworks, while the second one was used for the development of the prototype.

3.1 Research Design

Research design may be understood as the planning of any scientific research from the first to the last step. In this sense it is a program to guide the research in collecting, analyzing and interpreting observed facts (Nyandemo, 2007). It may also be defined as the arrangement of conditions for collection and analysis of data in a manner that aims to combine relevance to the research purpose with economy in procedure. It is the conceptual structure within which research is conducted; it constitutes the blueprint for the collection, measurement and analysis of data (Kothari, 2004).

3.1.1 Target population and sampling technique

For purposes of the design of a framework our sample population (sampling unit) was restricted to individuals in the ICT field. This was so that we could get input from persons who have studied or applied frameworks within the context of ICT. The type of universe, in our case, was finite. However, it was not possible for the researcher to establish the total number of ICT professionals neither in Kenya nor in Africa as a whole.

Data collection

Data collection for this research was two-fold:

Primary data for the framework design:
A review of works already done by other researchers was conducted. Weaknesses and strengths of their approaches were then highlighted. Further, an experience survey was conducted via structured questionnaires. An experience survey is defined as a survey of people who have had practical experience with the problem to be studied (Kothari, 2004). The questionnaires were availed to a population of individuals with varied backgrounds in ICT. A structured questionnaire is defined as one in which there are definite, concrete and pre-determined questions. The questions are presented
in exactly the same wording and in the same order to all respondents (Kothari, 2004). Resort is taken to this sort of standardization to ensure that all respondents reply to the same set of questions. Our questionnaire had both open and closed questions and can be found in appendix section of this document.

Our sampling unit, as mentioned earlier consisted of professionals in the ICT industry. Among the respondents was the M.Sc. Computer Science class of 2009, at the University of Nairobi, among other ICT professionals. An e-mail was sent to each of the said individuals requesting their feedback and onward transmission of the questionnaire to their colleagues and friends unknown to the researcher but working within the ICT field. This was to get as varied a view as possible on the topic at hand. Our questionnaire sought to find out:

- First, do ICT professionals actually use frameworks?
- Why do ICT professionals use frameworks?
- What is the impact of using a framework?
- What are some of the reasons that would make an ICT professional pick one framework over another?
- Do factors like complexity and number of components influence choice of a framework?
- What would be some of the considerations they would have in mind were they in a position to develop a framework.

We sent out a total of 100 questionnaires.

Secondary data for the prototype
This was obtained from Google map servers via their free plugin available on the World Wide Web for developers to use. The said data consisted of mapping information.

3.1.2 Mapping of objectives to research methodology

Table 2: Mapping of objectives to research methodology

<table>
<thead>
<tr>
<th>Research objectives</th>
<th>How they are achieved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Study current LBS frameworks</td>
<td>Literature review</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Development/Design of a conceptual framework for LBS development</td>
<td>Analysis of current frameworks</td>
</tr>
<tr>
<td>To develop a prototype based on our framework</td>
<td>- Base architecture of the prototype on the proposed conceptual framework.</td>
</tr>
<tr>
<td>- Programming in java.</td>
<td>- Unit &amp; Integration testing for the code.</td>
</tr>
<tr>
<td>Test the developed prototype</td>
<td>- System testing, via walking around CBPS and monitoring the prototype's response to movement.</td>
</tr>
<tr>
<td>Provide users with information based on their present location.</td>
<td>Development of a prototype</td>
</tr>
</tbody>
</table>

3.2 Data Processing and Analysis

The data, after collection, has to be processed and analyzed in accordance with the outline laid down for the purpose at the time of developing the research plan. This is essential for a scientific study and for ensuring that we have all the relevant data for making contemplated comparisons and analysis. Technically speaking, processing implies editing, coding, classification and tabulation of collected data so that they are amenable to analysis. Analysis then refers to the computation of certain measures along with searching for patterns of relationship that exist among data-groups (Nyandemo, 2007).

i. Editing

This is a process of examining the collected raw data to detect errors and omissions and to correct these when possible. It involves a careful scrutiny of the completed questionnaires and/or schedules. Editing is done to assure that the data are accurate, consistent with other facts gathered, uniformly entered, as complete as possible and have been well arranged to facilitate coding and tabulation. There are two forms of editing:

- Field editing – consists in the review of the reporting forms by the investigator for completing (translating or rewriting) what the latter has written in abbreviated and/or in illegible form at the time of recording the respondents’ responses.
Central editing – should take place when all forms or schedules have been completed and returned to the office. This type of editing implies that all forms/questionnaires should get a thorough editing by a single editor in a small study and by a team of editors in case of a large inquiry. Editor(s) may correct the obvious errors such as an entry in the wrong place, entry recorded in months when it should have been recorded in weeks, and the like. At times, the respondent can be contacted for clarification. This was the type of editing this research undertook.

ii. Classification

Most research studies result in a large volume of raw data which must be reduced into homogenous groups if we are to get meaningful relationships. This fact necessitates classification of data which happens to be the process of arranging data in groups or classes on the basis of common characteristics.

Table 3: Tabulation of research findings. Source: Research

<table>
<thead>
<tr>
<th>Work Experience (in years)</th>
<th>1 - 2</th>
<th>2 - 5</th>
<th>More than 5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>III</td>
<td>III</td>
<td>III</td>
</tr>
<tr>
<td></td>
<td>II</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>NO</th>
<th>YES</th>
<th>NO ANSWER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ever used a framework</td>
<td>III I</td>
<td>III III</td>
<td>III III</td>
</tr>
<tr>
<td></td>
<td>III</td>
<td>III III</td>
<td>III III</td>
</tr>
<tr>
<td></td>
<td>III</td>
<td>III</td>
<td>III</td>
</tr>
<tr>
<td>Does framework complexity matter?</td>
<td>III I</td>
<td>III III</td>
<td>III III</td>
</tr>
<tr>
<td>Productivity improved?</td>
<td>III I</td>
<td>III III</td>
<td>III III</td>
</tr>
<tr>
<td></td>
<td>III II</td>
<td>III</td>
<td>III</td>
</tr>
<tr>
<td>Number of components</td>
<td>III III</td>
<td>III</td>
<td>III I</td>
</tr>
</tbody>
</table>
As can be seen from above tables, our classification was based on attributes. The said attributes were descriptive in nature. Descriptive characteristics refer to qualitative phenomenon which cannot be measured quantitatively; only their presence or absence in an individual item can be noticed. Data obtained this way on the basis of certain attributes are known as statistics of attributes and their classification is said to be classification according to attributes (Nyandemo, 2007). We further used pie charts to simplify analysis as below:

Figure 6 Respondents' years of experience in ICT. Source: research

![Respondents' Years of Experience in ICT](image)

Figure 7 Impact of frameworks on productivity. Source: research

![Impact of frameworks on productivity](image)
Impact of frameworks on productivity

- Frameworks improved productivity (85%)
- Frameworks did not improve productivity (15%)

Figure 8 Respondents views on the importance of the number of components in a framework. Source: research.

Importance of Number of Components

- Undecided (23%)
- Important (15%)
- Not Important (62%)

Figure 9 Respondents views on the complexity of a framework. Source: research.
Does Framework’s Complexity Matter?

- Undecided
- Complexity does not matter
- Complexity matters (the simpler the better)

Figure 10 Gender distribution of the respondents. Source: research.

Gender distribution of respondents

- Female
- Male

Figure 11 Respondents previous experience with frameworks. Source: research.
iii. Analysis

It was instructive to note that 85% of the respondents (figure 7) agreed that frameworks are an important tool in ICT for productivity improvement. This was a major impetus for us to come up with a framework for location based services application design and development. Secondly, there was a consensus among the respondents that the number of components in a given framework may ultimately be used as a measure for deciding whether to use the said framework. 62% (figure 8) of the respondents believed that the fewer the number of components in a framework, the higher its chances of being adopted by ICT professionals. It was further deduced that they assumed that the fewer the components the easier a framework would be to implement. That might not be necessarily true. Nonetheless, it is important to note. Complexity was another aspect of frameworks that the respondents considered important. 69% (figure 9) of those polled believe that the simpler the framework the better; and the higher its chances of being adopted. Figures (6, 10 and 11) show the distribution of the respondents by the number of years in ICT (figure 6), gender (figure 9) and whether they have previously used a framework (figure 11).

3.3 Prototype development methodology

We chose the Rapid Application Development (RAD) methodology for our prototype. Rapid Application Development (RAD) refers to a development life cycle designed to give much
faster development and higher quality systems than the traditional life cycle. It is designed to take advantage of powerful development software like CASE tools, prototyping tools and code generators. RAD is a people-centered and incremental development approach. Active user involvement, as well as collaboration and co-operation between all stakeholders are imperative. Testing is integrated throughout the development life cycle so that the system is tested and reviewed by both developers and users incrementally. Professor Clifford Kettetemborough of Whitehead College, University of Redlands, defines Rapid Application Development as “an approach to building computer systems which combines Computer-Assisted Software Engineering (CASE) tools and techniques, user-driven prototyping, and stringent project delivery time limits into a potent, tested, reliable formula for top-notch quality and productivity. RAD drastically raises the quality of finished systems while reducing the time it takes to build them (CASEMaker 1997). Rapid Application Development thus enables quality products to be developed faster, saving valuable resources.

Traditional lifecycles devised in the 1970s, and still widely used today, are based upon a structured step-by-step approach to developing systems. This rigid sequence of steps forces a user to “sign-off” after the completion of each specification before development can proceed to the next step. The requirements and design are then frozen and the system is coded, tested, and implemented. With such conventional methods, there is a long delay before the customer gets to see any results and the development process can take so long that the customer’s business could fundamentally change before the system is even ready for use. In response to these rigid, cascading, one-way steps of Stage-wise or Waterfall Models of development, Barry Boehm, Chief SW Engineer at TRW, introduced his Spiral Model. The Spiral Model is a risk-driven, as opposed to code-driven, approach that uses process modeling rather than methodology phases. Through his model, Boehm first implemented software prototyping as a way of reducing risk. The development process of the Spiral Model separates the product into critical parts or levels while performing risk analyses, prototyping, and the same steps at each of these levels. Similarly, Tom Gilb’s Evolutionary Life Cycle is based on an evolutionary prototyping rationale where the prototype is grown and refined into the final product. The work of Boehm and Gilb paved the way for the formulation of the methodology called Rapid Iterative Production Prototyping (RIPP) at DuPont in the mid-to-late 1980s. James Martin (in 1991) then extended the work done at DuPont and elsewhere into a larger, more formalized process, which has become known as Rapid Application Development (RAD). RAD compresses the step-by-step development of conventional methods into an
iterative process. The RAD approach thus includes developing and refining the data models, process models, and prototype in parallel using an iterative process. User requirements are refined, a solution is designed, the solution is prototyped, the prototype is reviewed, user input is provided, and the process begins again (CASEMaker 1997). The pros and cons of RAD are (TestPlant 2011):

RAD Benefits

- Time needed to develop is lesser.
- Development cycles are shorter which will result in a cheaper system.
- User involvement is broader

RAD Detriments

- Planning is not a main concern
- It mainly focuses on resources and time as its functionality reduces
- Elements and prototypes which are reused again may cause irregularity in the designs

The key objectives of RAD are:

- High Speed;
- High Quality; and
- Low Cost.
- High emphasis on simplicity and usability of GUI design

Figure 12 A comparison between traditional development and RAD. Source: CASEMaker
The RAD life cycle composes of four stages:

- Requirements Planning;
- User Design;
- Rapid Construction; and
- Transition.
3.4 Requirements Planning (RP)

The objectives of this stage are:

- To establish a general understanding of the business problems that surround its development and eventual operation;
- To become familiar with existing systems and;
- To identify the business processes that will be supported by the proposed application.

During this stage, an outline of the system area and definition of the system scope were developed. The tasks of this stage were:

3.4.1 Research Current Situation

This task initiates the Requirements Planning stage for the proposed system by researching the current environment. In researching the current environment (with regard to location based service frameworks) we looked at existing models for building location based service applications. We singled out two well documented models for location based services design:

- The place lab model
- The University of Ljubljana model

These frameworks have been discussed in depth in chapter one of this document.

3.4.2 Define Requirements

The outline system area model and scope of the proposed prototype are developed in this task. For our case the scope of the prototype was as below:

- The prototype’s geographical area was limited to only cover the college of biological & physical sciences (CBPS), University of Nairobi.
- The prototype’s Points of interest (POIs) were limited to the following (all within CBPS):
  - Lecture rooms/halls
  - Cafeterias
  - Departmental buildings
- The prototype was developed to run only on mobile devices running on the android platform.

The prototype was required to do the following:
3.4.3 Finalize Requirements

In this task, the scope of the proposed system is formally documented. An estimate of the cost and duration to implement the system was prepared.

Project Costs

Table 4: Project costs. Source: Research.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Cost (in KES)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Development PC running Windows Vista</td>
<td>60000</td>
</tr>
<tr>
<td>2 Android GPS enabled phone (Huawei Ideos)</td>
<td>8500</td>
</tr>
<tr>
<td>3 SQLite database management system</td>
<td>0</td>
</tr>
<tr>
<td>4 Android SDK Release 11</td>
<td>0</td>
</tr>
<tr>
<td>5 Java SDK 1.5</td>
<td>0</td>
</tr>
<tr>
<td>6 Java Runtime Environment (JRE) v6 update 25</td>
<td>0</td>
</tr>
<tr>
<td>7 Eclipse Java EE IDE for Web Developers (Galileo)</td>
<td>0</td>
</tr>
<tr>
<td>8 Android ADT plugin for eclipse (with Google APIs)</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>68500</td>
</tr>
</tbody>
</table>

3.5 User Design (UD)

The objectives of the user design stage are

- to analyze in detail business activities associated with the proposed system area;
- to analyze in detail the business data associated with the proposed system area;
- to develop the system structure in terms of the automated and manual functions that will comprise the system;
- to develop proposed screen layouts for the most important automated functions;
to select the appropriate construction approach for the system and; 

The UD stage produces a detailed system area model, an outline system design, and an implementation plan. The tasks of this stage are:

3.5.1 Produce Detailed System Area Model

JAD workshops are conducted to complete the analysis of the business activities and data associated with the proposed system, and produce a detailed system area model. Once the detailed system area model is completed, it is possible to refine the scope to ensure that the critical functions will be delivered in the required time frame.

We used *Data Flow Diagrams* to define the requirements of our prototype.

**The Context Diagram**

Figure 13 Context Diagram. Source: Research.
Level Zero Diagram

Figure 14 Level zero diagram. Source: Research.
3.5.2 Develop Outline System Design

An outline system design is developed by the JAD workshop participants using the system area model that has been completed and confirmed. Upon completion of the outline design, interactions between procedures (series of functions) and data are identified. That is, data usage per function is defined. The deliverables produced in this task include functions required for the system, reusable design components to be incorporated into the system design, system structures, database structures and tentative layouts of critical screens and reports to be supported by the system.

Our outline system design was thus:

**Database Design**

Our prototypes 'database was thus:
Screen Design
The prototypes’ screens were as below:

\[ a) \quad \text{Output Screens:} \]

Figure 16 Map and address screen. Source: Research.
Figure 17 Lecturer details screen. Source: Research.

Figure 18 Lecturers' department screen. Source: Research.
Figure 19 POI details screen. Source: Research.

Figure 20 Arbitrary notifications screen. Source: Research.
Function Design

Functions necessary for our prototype's operation were:

Get current location

The flowchart for this function was as below
Figure 21 Get Current Location. Source: Research.
Get POI information

Figure 22 Get POI Information. Source: Research.
Get proximity alerts

Figure 23 Get Proximity Alerts. Source: Research.

Get current address
Figure 24 Get current address. Source: Research.
3.6 Proof of Concept
We developed a prototype application that implements the functionality described in chapter one (under objectives). Some of the functionality went unimplemented due to time constraints, but remains feasible to achieve. The prototype is derived from our framework architecture. We have a mobile device accessing a GPRS/EDGE/3G network to gain access to the internet. It then accesses the spatial database server via this link. The device has in-built GPS and WiFi receivers. It uses these receivers to obtain its current location in terms of geographic coordinates. This location is mapped onto the map downloaded to the device from the spatial database server. We then overlay our points of interest on the downloaded map. We further create proximity alerts based on our custom points of interest.
CHAPTER 4: IMPLEMENTATION

In this phase the designs from the previous phase are translated into code. Computer programs are written using a conventional programming language or an application generator. This is the phase where the programmers swing into action and get their hands 'dirty' with code. The various forms of testing are also done in this phase. The framework is also developed in this phase.

4.1 Theoretical Framework

We came up with a conceptual framework for the development of Location based applications. A conceptual framework is defined as a set of theories widely accepted enough to serve as the guiding principles of research within a particular discipline. It is used in research to outline possible courses of action or to present a preferred approach to an idea or thought. Our motivation for coming up with a framework was guided by the need to simplify development of LBS applications for the application developer.

The architecture of our framework was as below.

Figure 25 Diagram depicting our conceptual framework architecture. Source: Research.

The architecture of our framework was much simpler (both conceptually and in terms of effort required to implement it) than both PlaceLab and Ljubljana frameworks. Simplicity
was among our objectives when designing the framework. Components of our framework include:

- A mobile device – a laptop, netbook, palmtop, mobile phone etc
- A wifi receiver
- A gps receiver
- A network that allows connection of the mobile device to the internet
- A spatial database server

The choice of the mobile device or how it is connected to the wifi and gps receivers is trivial. What was of concern to us in this work was for it to have a way of detecting its location. Besides wifi and gps, cell-id (discussed earlier in this thesis) may also be used to determine the devices location. The network was another important component of our framework. This network must be one that allows the device to connect to the internet. The connection maybe through wifi, GSM or even via a wired network. However, in practice mobile devices typically connect to the internet via wireless technologies (wifi, GSM, 3G etc) for mobility reasons. The last component was the spatial database. This was basically the database that holds locational data i.e. maps. Again the type of spatial database is trivial. The application developer may choose to connect to one of the many free map servers such as google maps, yahoo maps, ovi maps, bing maps etc. the other option would be for the programmer to build their own spatial database, a most expensive and tedious process.

Figure 26 Framework components. Source: Research.
The mobile application sat on our framework, it provided methods for obtaining our current location and for obtaining location information (country, city, street and address). The application connected to the spatial database via a network which could be GPRS/EDGE/3G/CDMA etc. again the type of network is trivial, the important thing was that it be a network the ‘moves’ around with the mobile device.

To implement our above framework, the programmer needed to first have in place the below components.

- A spatial database server i.e. a map server
- A mobile device with a GPS and Wi-fi receivers
- A network that allows connection of the mobile device to the internet
- An Integrated Development Environment (IDE) such as eclipse, visual studio, netbeans etc. The development environment has to be compatible with the mobile device chosen. For instance, if it is a windows mobile device then one would use the Windows Phone Application Platform for development of the application (Microsoft 2011). On the other hand if it was an android mobile device then one could, for instance, use the Eclipse Java IDE. These tools are freely downloadable from the internet. For any given mobile platform, the programmer will find necessary information regarding IDE compatibilities from the manufacturers website.

The programmer also has to make sure that the map server of choice is compatible with the mobile device. For instance, the android platform is not compatible with the Nokia’s ovi maps server. To use some of the freely available map servers, e.g. google maps, in one’s application requires permission from the map owners to avoid infringing on copy rights. For google, there is a simple online registration which is free of charge.

4.2 Rapid Construction (RC)

The objectives of the Rapid Construction stage are to:
- complete the detailed design of the proposed system;
- create and test the software that implements the proposed system;
- generate a system that operates at an acceptable level of performance;
- prepare documentation necessary to operate the proposed application and;
design, develop, and test the required transition software.

The tasks of the RC stage are:

4.1.1 Prepare for Rapid Construction

The development environment is finalized including workstations and workspace for the system developers, and the CASE software for their use. Additionally, the database is designed based on the preliminary data structure developed in the UD stage. Finally, the testing strategy for the system is completed and the acquisition of facilities necessary to operate the system after it is constructed is initiated.

Our development environment consisted of the following:

- Eclipse Java EE IDE for Web Developers SR2 - Galileo version
- SDK for Android version 11
- ADT plugin for Eclipse version 11
- Huawei Ideos 8150 phone
- Windows vista with SP1 and Windows 7 Enterprise SP1
- SQLite Database

Eclipse Java and the Android SDK were installed on the Windows machines and then the ADT was installed and used to link the two. This allowed the development to be done on Eclipse and tested via the ADT on both an android emulator on the SDK and also on a physical Android device (Huawei Ideos in our case).

4.2 Construct System

The detailed definition of the design of each function is completed, based on the requirements of the future end-users. Software to implement the automated functions is developed and tested.

Having all components from section 4.1 above at hand, the next step was to develop the application that connects the various components. In our case we had to give our application permission to access the mobile devices internet connection (android.permission INTERNET) and locational services.
(android.permission.ACCESS_FINE_LOCATION). This permission allows an application to access fine (e.g. GPS) location (Android 2011). To access google maps we needed the ‘com.google.android.maps’ library. Classes for the display and control of the map on the user’s device (i.e. moving the map, panning & zooming etc) were also necessary. A class for getting the devices location and constantly keeping it updated as the device moves was also needed. We extended android’s LocationManager class for this and listened for location changes every two seconds. We however only updated our location after detecting movement of more than 10 meters. A class for overlaying our points of interest on the map was also necessary. We extended android’s ‘ItemizedOverlay’ class, and modified it for our purposes. A graphics class was also created for managing the graphics associated with the various points of interest. Lastly, an SQLite database was created for holding our overlays, lecturer information, departmental information, cafeteria information and any other information that the application presented to its users. See sample code below.

### 4.2.1 Sample code

```java
import java.io.IOException;
import java.util.List;
import java.util.Locale;
import android.content.Context;
import android.database.Cursor;
import android.graphics.drawable.Drawable;
import android.location.Address;
import android.location.Criteria;
import android.location.Geocoder;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.net.Uri;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.ListView;
import android.widget.SimpleCursorAdapter;
import android.widget.TextView;
import android.widget.Toast;
```
import com.google.android.maps.GeoPoint;
import com.google.android.maps.MapActivity;
import com.google.android.maps.MapController;
import com.google.android.maps.MapView;
import com.google.android.maps.Overlay;
import com.google.android.maps.OverlayItem;

public class Nilipo extends MapActivity {
    // class for displaying & controlling map

    MapController mapController;
    MyOverlay positionOverlay;

    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.main);

        MapView myMapView = (MapView) findViewById(R.id.myMapView);
        mapController = myMapView.getController();

        myMapView.setBuiltInZoomControls(true);
        // myMapView.setSatellite(true);
        // myMapView.setStreetView(true);
        myMapView.setTraffic(true);

        // Zoom in
        mapController.setZoom(19);

        // Add the MyPositionOverlay
        positionOverlay = new MyOverlay();
        List<Overlay> overlays = myMapView.getOverlays();
        overlays.add(positionOverlay);
    }
}
//adding itemized overlay markers

List<Overlay> mapOverlays = myMapView.getOverlays();
Drawable drawable = this.getResources().getDrawable(R.drawable.marker);
MyItemizedOverlay itemizedoverlay = new MyItemizedOverlay(drawable, this);

mapOverlays.add(itemizedoverlay);

LocationManager locationManager =
(LocationManager) getSystemService(Context.LOCATION_SERVICE);

Criteria criteria = new Criteria();
criteria.setAccuracy(Criteria.ACCURACY_FINE);
criteria.setAltitudeRequired(false);
criteria.setBearingRequired(false);
criteria.setCostAllowed(true);
criteria.setPowerRequirement(Criteria.POWER_LOW);
String provider = locationManager.getBestProvider(criteria, true);

More code is available in the appendix section of this document.

4.2.2 Sample Screens

The prototypes screens were as below.

Figure 27 Proximity Alert Screen. This was the screen displayed once a user entered the radius prescribed as a Point Of Interest (POI). It informed the user that he/she was near a POI. Source: Research.
Proximity Alert!
You are near a point of interest.
8:12 PM
Figure 28 Department details screen. This screen displayed CBPS departments stored in the prototype’s database. Source: Research.

<table>
<thead>
<tr>
<th>Auto Id</th>
<th>Dept Id</th>
<th>Dept Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BIOC</td>
<td>Biochemistry</td>
</tr>
<tr>
<td>2</td>
<td>SCI</td>
<td>Computing</td>
</tr>
<tr>
<td>3</td>
<td>PHY</td>
<td>Physics</td>
</tr>
<tr>
<td>4</td>
<td>ZOO</td>
<td>Zoology</td>
</tr>
</tbody>
</table>
Figure 29 Lecturer details screen. This screen displayed CBPS lecturers stored in the prototype’s database.

Source: Research.

<table>
<thead>
<tr>
<th>Auto Id</th>
<th>Lect Id</th>
<th>Lect Name</th>
<th>Dept Id</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SCI001</td>
<td>Phillip Okumu</td>
<td>SCI</td>
</tr>
<tr>
<td>2</td>
<td>BIOC001</td>
<td>Marigat Kipkorir</td>
<td>BIOC</td>
</tr>
<tr>
<td>3</td>
<td>SCI002</td>
<td>Joyce Kamau</td>
<td>SCI</td>
</tr>
<tr>
<td>4</td>
<td>ZOO001</td>
<td>Peter Ojwang</td>
<td>ZOO</td>
</tr>
</tbody>
</table>
Figure 30 Cafeteria Meals screen. This screen displayed CBPS cafeterias with the meals available on each day. Source: Research.

<table>
<thead>
<tr>
<th>Auto Id</th>
<th>Location Id</th>
<th>Day</th>
<th>Vegetables</th>
<th>Carbohydrate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SWA</td>
<td>mon</td>
<td>kales</td>
<td>ugali</td>
</tr>
<tr>
<td>3</td>
<td>UNES</td>
<td>mon</td>
<td>cabbage</td>
<td>rice</td>
</tr>
<tr>
<td>4</td>
<td>MESS</td>
<td>mon</td>
<td>cabbage</td>
<td>chapati</td>
</tr>
<tr>
<td>2</td>
<td>SWA</td>
<td>tue</td>
<td>kales</td>
<td>chapati</td>
</tr>
<tr>
<td>5</td>
<td>MESS</td>
<td>tue</td>
<td>kales</td>
<td>rice</td>
</tr>
<tr>
<td>6</td>
<td>UNES</td>
<td>wed</td>
<td>Cauliflower</td>
<td>chapati</td>
</tr>
</tbody>
</table>
4.3 Generate Test Data and System Documents

This task involves developing the necessary test data that will verify the operational capacity of the system. This data will be used during the integration, system and acceptance tests.

Test data for the prototype mainly consisted of coordinates of the various Points of Interest within the college. We used a free GPS tool downloaded from the android market, *GPS Status*. The tool is developed by eclipsim. The tool displays your GPS and sensor data. It also

![Lecturer Availability Times](image-url)
shows the position, number and signal strength of GPS satellites. It checks your position, GPS accuracy, speed, acceleration and bearing. A magnetic compass is included which displays both magnetic and true north. The format of the coordinates collected was of the form DD.DDDDDD°, where D is for degrees. Further, we did some updates to Google maps via the Google Map Maker tool. Google Map Maker is a tool from Google that lets you add to and update the map, for millions of people to see in Google Maps and Google Earth. However, before an addition is published online it has to go through a vetting process by other mappers. Our published updates included:

- Science Crescent
- Alala Road
- Wasawo drive
- Mungai Road
- School of Computing & Informatics
- Millennium halls I & II
- Physics Department

Samples of our published updates are shown below.

Figure 32 Our accepted updates on Google Map Maker I. Source: Google Maps
4.4 Verify System Construction

The system must be put through a series of tests to ensure that each component of the system and the complete system performs according to the user requirements.

We chose black-box testing as our preferred testing approach. Black-box testing is a method of software testing that tests the functionality of an application as opposed to its internal structures or workings. This testing methodology looks at what are the available inputs for an application and what the expected outputs are that should result from each input. It is not concerned with the inner workings of the application, the process that the application undertakes to achieve a particular output or any other internal aspect of the application that may be involved in the transformation of an input into an output (TestPlant 2011, p.1). This
method of test can be applied to all levels of software testing: unit, integration, functional, system and acceptance.

In this regard we conducted the following tests on our prototype.

- **Unit testing** – which is a method by which individual units of source code are tested to determine if they are fit for use. In our case a unit was in the form of a method, which in turn is a part of an activity. An activity in android is the equivalent of a form in other visual programming languages. Unit tests aid in finding problems early in the development cycle. Tests were conducted on each method as it was built and after it satisfactorily passed its test, it was integrated into the android activity.

- **Integration testing** – which is the phase in software testing in which individual software modules are combined and tested as a group. It was done after unit testing.

- **System testing** - is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. This was done after integration testing.
CHAPTER 5: CONCLUSION & RECOMMENDATIONS

5.1. Evaluation of Objectives

From the research end of our thesis our contribution was the framework. Our framework abstracts location acquisition techniques from the programmer to leave the programmer with the single task of concentrating on building the application itself. All the programmer needs to ensure is that the various components of the framework are in place i.e. a mobile device with capability to connect to the internet, a GPS receiver, a Wi-Fi receiver and a map server. The programmer can then jump into the Integrated Development Environment of choice and come up with a location based service application in a couple of hours. Our research objectives at the onset of this thesis were:

- To compare and contrast available location based application frameworks – we achieved this, as illustrated in the literature review (chapter 2) section of this thesis.
- To design a simple framework/model for location based services application development – we achieved this objective as documented in the implementation section of this thesis.
- To develop a prototype based on our framework, as a proof-of-concept. We also achieved this objective and developed, in Java, a prototype running on the android platform.

From the application development end of our research our contribution was a working prototype that meets the below objectives as set out in the introduction section of this thesis.

- To provide users with information based on their present location. This includes information about location of lecture halls, cafeterias, laboratories among other points of interest. Our prototype provided the below information.
  - Information about the various departments at the University of Nairobi including a brief of their areas of specialty.
  - Information about lecturers at the University. This included information about their specialization, their office locations and their availability days and times, for student consultations.
Information about the locations and the menus of the various cafeterias within the campus.

To provide students/visitors with an alert when they approach a point of interest. This was also achieved via proximity alerts in the prototype.

5.2. Conclusion
We have established, via this research, that Information & Computing Technology professionals use frameworks for improved productivity. However, there are certain elements of a framework that would make an ICT professional prefer one framework over another. Chief among these are:

- The complexity of a framework. This may either be the perceived complexity or the actual complexity of a framework. From our research it became apparent that the more complex a framework is the higher its chances of NOT being adopted and vice versa.
- The number of different components present in a framework. Again from our research we established that the more the number of components in a framework the more it is perceived as complex and the higher it chances of being rejected. The reverse is also true.

This research has further shown that building mobile location aware applications is achievable by using open standards. Our framework abstracted location acquisition techniques from the programmer to leave him/her with the single task of concentrating on building the application itself. The framework reduces the number of components the developer needs to implement and therefore its complexity. This is ideally what every programmer seeks, to improve their productivity i.e. a simple and easy to implement framework. By using this framework we can integrate any platform (Apple’s IOS, Nokia’s Symbian, Microsoft’s Windows etc.) with any spatial database may it be Oracle’s 9i, 10g, 11g or Microsoft SQL server, MySQL, or any third party specialized geographic database.

Nonetheless, if our prototype was to be extended so that users could interact (add data to the database) with the data then its database would have to be hosted on a webservice for universal access. The prototype, as it is currently, runs the database on the local device. Running the database from a webservice would create additional latency, but is the only way to allow user feedback to be universally accessible.
5.3. Limitations of the Research

The main limitation of our study was the rather small sample group of ICT professionals polled. The sample group consisted of only one hundred professionals and might not be as representative as it could be. This is especially so as we were unable to establish the exact number of ICT professionals globally.

5.4. Recommendations & Further Work

Given the experience gained during the research, the prototype can be extended to include the below features:

- Add functionality to let users interactively add locations to the application i.e. their preferred points of interest.
- Allow users to add comments for a given point of interest.
- Allow users to add proximity alerts for their preferred points of interest.
- Explore how to let users add location based reminders i.e. the user sets a reminder based on a location so that they get a prompt when they are in the said location.
- Host the prototype’s database on a webserver to allow centralized access.

Further research may also consider other innovative uses for location based technologies. This may include using location based technologies for:

- Traffic management: aggregate many users’ locations and then provide driving routes based on traffic patterns discerned. i.e. a location where many users are located, at some given time, should be avoided as a route because it would mean that the users are stuck in traffic
- Emergency response: enable emergency responders to find out the locations of people requesting emergency response i.e. once a user places a call then the system automatically tracks their location.
- Targeted shopping/advertising: provides users details about nearby shopping possibilities and also send relevant adverts to users at a specific location. E.g. send a
user a message regarding a new shampoo if the user is near a salon for a say more than 15 minutes.
REFERENCES

ANDREJ K et al. (2006). A framework for developing mobile location based applications [WWW] University of Ljubljana and Slovenia Institute of Quality and Metrology. Available from:


BERNARDOS A M et al. (2009). Deploying context-aware services: A case study of rapid prototyping [WWW] Universidad Politecnica de Madrid. Available from


APPENDIX I: USER MANUAL

Installation

The prototype compiles the first time it is run from the development environment. Plans are also underway to launch it on the android market where users can download it from. It automatically installs itself on the device once it is downloaded. Once installed, the user clicks on the application icon to launch it.
Home Screen

On launching the application, via clicking on its icon, the below screen is displayed. The screen shows your current location on a map, and your current address as text.

Menus

On clicking on the menu button, the below screen is displayed. The various menus allow you to access information from the application’s database.
My Location Menu

On clicking on this menu option, the application finds your current location and displays it on the centre of the screen.
Lecturer Details Menu

On clicking on this menu option, the application loads the below screens with lecturer information.
On clicking on one of the lectures, further information about the lecturer is displayed as below.
Miss Kamau specializes in Artificial Intelligence, her office is in the SCI building room 10.

<table>
<thead>
<tr>
<th>Lect Id</th>
<th>Lect Name</th>
<th>Dept Id</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCI002</td>
<td>Joyce Kamau</td>
<td>SCI</td>
</tr>
<tr>
<td>BIOC001</td>
<td>Mangat Kipkor</td>
<td>BIOC</td>
</tr>
<tr>
<td>ZOO001</td>
<td>Peter Ojwang</td>
<td>ZOO</td>
</tr>
<tr>
<td>SCI001</td>
<td>Phillip Okumu</td>
<td>SCI</td>
</tr>
<tr>
<td>Lect Id</td>
<td>Lect Name</td>
<td>Dept Id</td>
</tr>
<tr>
<td>---------</td>
<td>---------------</td>
<td>---------</td>
</tr>
<tr>
<td>SCI002</td>
<td>Joyce Kamau</td>
<td>SCI</td>
</tr>
<tr>
<td>BIOC001</td>
<td>Marigat Kipkorir</td>
<td>BIOC</td>
</tr>
<tr>
<td>ZOO001</td>
<td>Peter Oiwang</td>
<td>ZOO</td>
</tr>
<tr>
<td>SCI001</td>
<td>Phillip Okumu</td>
<td>SCI</td>
</tr>
</tbody>
</table>
Departments Menu

On clicking on this menu option, the application loads the below screens with departmental information.

<table>
<thead>
<tr>
<th>Lect Id</th>
<th>Lect Name</th>
<th>Dept Id</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCI002</td>
<td>Joyce Kamau</td>
<td>SCI</td>
</tr>
<tr>
<td>BIOC001</td>
<td>Marigat Kipkorir</td>
<td>BIOC</td>
</tr>
<tr>
<td>ZOO001</td>
<td>Peter Ojwang</td>
<td>ZOO</td>
</tr>
<tr>
<td>SCI001</td>
<td>Phillip Okumu</td>
<td>SCI</td>
</tr>
</tbody>
</table>

Mr Okumu specializes in Mobile Computing, his office is in the SCI building room 4.
On clicking on one of the departments, further information about the department is displayed as below.
The Department handles over 300 students- undergraduate and postgraduate- registered for different degree programmes for which physics is core. This number excludes the serviced courses belonging to other schools and faculties. It offers both undergraduate & postgraduate courses.
<table>
<thead>
<tr>
<th>CBPS Departments</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Dept Id</td>
<td>Dept Name</td>
</tr>
<tr>
<td>BIOC</td>
<td>Biochemistry</td>
</tr>
<tr>
<td>SCI</td>
<td>Computing</td>
</tr>
<tr>
<td>PHY</td>
<td>Physics</td>
</tr>
<tr>
<td>ZOO</td>
<td>Zoology</td>
</tr>
</tbody>
</table>

![Image of a device interface](image_url)
Cafeterias Menu

On clicking on this menu option, the application loads the below screens with CBPS cafeteria information. One may scroll sideways in case all contents of the menu are not visible on the screen.

### CBPS Departments

<table>
<thead>
<tr>
<th>Dept Id</th>
<th>Dept Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIOC</td>
<td>Biochemistry</td>
</tr>
<tr>
<td>SCI</td>
<td>Computing</td>
</tr>
<tr>
<td>PHY</td>
<td>Physics</td>
</tr>
<tr>
<td>ZOO</td>
<td>Zoology</td>
</tr>
</tbody>
</table>

The department offers courses to undergraduate students in six schools and two faculties namely: Schools of Medicine, Pharmacy, Dental Sciences, Physical Sciences, Biological Sciences, of Nursing; Faculty of Veterinary Medicine and Faculty of Agriculture.
### Lecturer Availability Menu

On clicking on this menu option, the application loads the below screens with lecturer availability information.

<table>
<thead>
<tr>
<th>Location</th>
<th>Day</th>
<th>Vegetables</th>
<th>Carbohydrate</th>
<th>Protein</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWA</td>
<td>mon</td>
<td>kales</td>
<td>ugali</td>
<td>beef</td>
</tr>
<tr>
<td>UNES</td>
<td>mon</td>
<td>cabbage</td>
<td>rice</td>
<td>chicken</td>
</tr>
<tr>
<td>MESS</td>
<td>mon</td>
<td>cabbage</td>
<td>chapati</td>
<td>beef</td>
</tr>
<tr>
<td>SWA</td>
<td>tue</td>
<td>kales</td>
<td>chapati</td>
<td>mutton</td>
</tr>
<tr>
<td>MESS</td>
<td>tue</td>
<td>kales</td>
<td>rice</td>
<td>beef</td>
</tr>
<tr>
<td>UNES</td>
<td>wed</td>
<td>Cauliflower</td>
<td>chapati</td>
<td>lamb</td>
</tr>
</tbody>
</table>
### Lecturer Availability Times

<table>
<thead>
<tr>
<th>Lecturer Id</th>
<th>Day of Week</th>
<th>TimeFrom</th>
<th>TimeTo</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCI001</td>
<td>Mon</td>
<td>9:00</td>
<td>11:00</td>
</tr>
<tr>
<td>ZOO001</td>
<td>Mon</td>
<td>8:00</td>
<td>9:30</td>
</tr>
<tr>
<td>BIOC001</td>
<td>Thu</td>
<td>14:00</td>
<td>15:30</td>
</tr>
<tr>
<td>SCI002</td>
<td>Thu</td>
<td>8:00</td>
<td>9:00</td>
</tr>
<tr>
<td>SCI001</td>
<td>Tue</td>
<td>9:00</td>
<td>12:00</td>
</tr>
<tr>
<td>ZOO001</td>
<td>Wed</td>
<td>16:00</td>
<td>17:30</td>
</tr>
</tbody>
</table>

### Lecturer Access Times

<table>
<thead>
<tr>
<th>Course Code</th>
<th>Day of Week</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCI001</td>
<td>Mon</td>
<td>9:00</td>
</tr>
<tr>
<td>SCI002</td>
<td>Mon</td>
<td>8:00</td>
</tr>
<tr>
<td>BIOC001</td>
<td>Thu</td>
<td>14:00</td>
</tr>
<tr>
<td>SCI002</td>
<td>Thu</td>
<td>8:00</td>
</tr>
<tr>
<td>SCI001</td>
<td>Tue</td>
<td>9:00</td>
</tr>
<tr>
<td>ZOO001</td>
<td>Wed</td>
<td>16:00</td>
</tr>
</tbody>
</table>
import android.graphics.Canvas;
import android.graphics.Paint;
import android.graphics.Point;
import android.graphics.RectF;
import android.location.Location;
import com.google.android.maps.GeoPoint;
import com.google.android.maps.MapView;
import com.google.android.maps.Overlay;
import com.google.android.maps.Projection;

public class MyOverlay extends Overlay {

    /** Get the position location */
    public Location getLocation() {
        return location;
    }

    /** Set the position location */
    public void setLocation(Location location) {
        this.location = location;
    }

    Location location;

    private final int mRadius = 5;

    @Override
    public void draw(Canvas canvas, MapView mapView, boolean shadow) {
        Projection projection = mapView.getProjection();

        if (shadow == false) {
            GeoPoint geoPoint = new GeoPoint(-1263030,
            36911300);

            // Draw the circle
            Paint paint = new Paint();
            paint.setColor(Color.BLUE);
            paint.setStrokeWidth(3f);
            paint.setStyle(Paint.Style.STROKE);
            canvas.drawCircle(geoPoint.getLongitude(), geoPoint.getLatitude(), mRadius, paint);
        }
    }
}
// Convert the location to screen pixels
Point point = new Point();
projection.toPixels(geoPoint, point);

RectF oval = new RectF(point.x - mRadius, point.y -
mRadius,
point.x + mRadius, point.y +
point.x + mRadius, point.y +
RectF backRect = new RectF(point.x + 2 + mRadius,
point.y - 3*mRadius,
point.x + 65, point.y +
mRadius);

// Setup the paint
Paint paint = new Paint();
paint.setARGB(255, 255, 0, 0);
paint.setAntiAlias(true);
paint.setFakeBoldText(true);

Paint backPaint = new Paint();
backPaint.setARGB(180, 50, 50, 50);
backPaint.setAntiAlias(true);

canvas.drawOval(oval, paint);
canvas.drawRoundRect(backRect, 5, 5, backPaint);
canvas.drawText("Am here", point.x + 2*mRadius,
point.y, paint);

super.draw(canvas, mapView, shadow);

@Override
public boolean onTap(GeoPoint point, MapView mapView) {
    return false;
}
package com.chuo.thesis;

import java.util.ArrayList;
import android.app.AlertDialog;
import android.content.Context;
import android.graphics.drawable.Drawable;
import com.google.android.maps.ItemizedOverlay;
import com.google.android.maps.OverlayItem;

public class MyItemizedOverlay extends ItemizedOverlay {

    private ArrayList<OverlayItem> mOverlays = new ArrayList<OverlayItem>();
    private Context mContext;

    public MyItemizedOverlay(Drawable defaultMarker, Context context) {
        super(boundCenterBottom(defaultMarker));
        mContext = context;
    }

    public void addOverlay(OverlayItem overlay) {
        mOverlays.add(overlay);
        populate();
    }

    @Override
    protected OverlayItem createItem(int i) {
        return mOverlays.get(i);
    }

    @Override
    public int size() {
        return mOverlays.size();
    }

    private static final String DATABASE_CREATE_VIEW_VIEWLEC

    private static final String DATABASE_CREATE_VIEW_VIEWLEC

    79
"CREATE VIEW viewLecs AS SELECT LectTable."+colLecId+" AS _id,"+
"LectTable."+colLecName+","+
"DeptTable."+colDeptName+"+"+
FROM "LectTable" JOIN "DeptTable"+
ON "LectTable."+colLecDeptId+
= DeptTable."+colDeptID

private static class DatabaseHelper extends SQLiteOpenHelper
{
    DatabaseHelper(Context context) {
        super (context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db)
    {
        db.execSQL(DATABASE_CREATE_DEPTTABLE);
        db.execSQL(DATABASE_CREATE_LECTABLE);
        db.execSQL(DATABASE_CREATE_TRIGGER_LECDEPT);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        Log.w("Content provider database",
        "Upgrading database from version " +
        oldVersion + " to " + newVersion +
        ", which will destroy all old data");

        db.execSQL("DROP TABLE IF EXISTS LecturerDetails");
db.execSQL("DROP TABLE IF EXISTS DepartmentDetails");
db.execSQL("DROP TRIGGER IF EXISTS fk_lecdept_deptid");

onCreate(db);
}

//To delete an item, override the delete() method:
@override
public int delete(Uri arg0, String arg1, String[] arg2) {
    int count=0;
    switch (uriMatcher.match(arg0)) {
        case LECTURERS:  
            count = Chuodb.delete(  
                LecTable,  
                arg1,  
                arg2);  
            break;
        case DEPTS:  
            count = Chuodb.delete(  
                DeptTable,  
                arg1,  
                arg2);  
            break;
    }
    getContext().getContentResolver().notifyChange(arg0, null);  
    return count;
}
public String getType(Uri uri) {
    switch (uriMatcher.match(uri)) {
        case LECTURERS:
            return "vnd.android.cursor.dir/vnd.learn2develop.Lecturers ";
        case DEPTS:
            return "vnd.android.cursor.dir/vnd.learn2develop.Depts ";
        default:
            throw new IllegalArgumentException("Unsupported URI: " + uri);
    }
}

//To allow new items to be inserted into the content provider, //override the insert() method:
@Override
public Uri insert(Uri uri, ContentValues values) {
    Uri _uri = null;
    switch (uriMatcher.match(uri)) {
        case LECTURERS:
            //---add a new lecturer---
            long rowIDl = Chuodb.insert("Lecture",
                values);
            //---if added successfully---
            if (rowIDl>0) {
            
        }
    }
}
_uri = ContentUris.withAppendId(CONTENT_URI1, rowID1);
getContext().getContentResolver().notifyChange(_uri, null);
}
break;

case DEPTS:
    //---add a new dept---
    long rowID2 = Chuodb.insert(DeptTable, "", values);
    //---if added successfully---
    if (rowID2>0) {
        _uri = ContentUris.withAppendId(CONTENT_URI2, rowID2);
        getContext().getContentResolver().notifyChange(_uri, null);
    }
break;

default: throw new SQLException("Failed to insert row into " + uri);
}
return _uri;

//open a connection to the database when the content provider is started
@Override
public boolean onCreate() {
    Context context = getContext();
    DatabaseHelper dbHelper = new DatabaseHelper(context);
    Chuodb = dbHelper.getWritableDatabase();
    return (Chuodb == null)? false:true;
}
public Cursor query(Uri uri, String[] projection, String selection,
        String[] selectionArgs, String sortOrder) {

    SQLiteQueryBuilder sqlBuilder = new SQLiteQueryBuilder();
    Cursor c = null;
    switch (uriMatcher.match(uri)) {

        case LECTURERS:

            sqlBuilder.setTables(LecTable);

            if (sortOrder == null || sortOrder =="")
                sortOrder = colLecName;

            c = sqlBuilder.query(
                Chuodb,
                projection,
                selection,
                selectionArgs,
                null,
                null,
                sortOrder);

            break;

        case DEPTS:

            sqlBuilder.setTables(DeptTable);

            if (sortOrder == null || sortOrder =="")

            break;

    }
sortOrder = colDeptName;

c = sqlBuilder.query(
    Chuodb,
    projection,
    selection,
    selectionArgs,
    null,
    null,
    sortOrder);

break;

//--register to watch a content URI for changes--
// c.setNotificationUri(getContext().getContentResolver(), uri);
}
return c;

public void DisplayLect(Cursor c)
{

Uri allLects = Uri.parse("content://net.learn2develop.provider2.Chuo/lecturers");
c = managedQuery(allLects, null, null, null, null);

startManagingCursor(c);
ListView listContent = (ListView) findViewById(R.id.contentlist);

SimpleCursorAdapter adapter = new SimpleCursorAdapter(
this,
R.layout.row1,
getContentResolver().query(allLects, null, null, null, null), // getting cursor for database query here.
new String[] { "_id", "LecId", "LecName", "LecDeptId" },
new int[] { R.id.text1, R.id.text2, R.id.text3, R.id.text4
});

//ListView listContent = (ListView) findViewById(R.id.contentlist);
// moved from up
listContent.setHeaderDividersEnabled(true); //added
listContent.addHeaderView(getLayoutInflater().inflate(R.layout.row1, null)); //added
//listContent.setAdapter(cursorAdapter);

listContent.setAdapter(adapter);

}

public void DisplayDept(Cursor c)
{

Uri allDepts = Uri.parse("content://net.learn2develop.provider2.Chuo/depts");
c = managedQuery(allDepts, null, null, null, null);

startManagingCursor(c);
ListView listContent = (ListView) findViewById(R.id.contentlist);

SimpleCursorAdapter adapter = new SimpleCursorAdapter(
    this,
    R.layout.row2,
getContentResolver().query(allDepts, null, null, null, null), // getting cursor for database query here.
    new String[] { "_id", "DeptID", "DeptName" },
    new int[] { R.id.text1, R.id.text2, R.id.text3 });

// ListView listContent = (ListView) findViewById(R.id.contentlist);

// moved from up
listContent.setHeaderDividersEnabled(true); // added
listContent.addHeaderView(getLayoutInflater().inflate(R.layout.row2, null)); // added

// listContent.setAdapter(cursorAdapter);

listContent.setAdapter(adapter);

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
/>

    <ListView
    android:id="@+id/contentlist"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"/>

</LinearLayout>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_height="wrap_content"
    android:id="@+id/rowLayout"
    android:orientation="horizontal"
    android:layout_width="fill_parent">
    <TextView android:layout_width="wrap_content"
              android:layout_alignParentLeft="true"
              android:layout_height="wrap_content"
              android:layout_marginRight="20px"
              android:id="@+id/text1"
              android:text="Auto Id" />
    <TextView android:layout_width="wrap_content"
              android:layout_toRightOf="@+id/text1"
              android:layout_height="wrap_content"
              android:layout_marginRight="20px"
              android:id="@+id/text2"
              android:text="Lecturer Id" />
    <TextView android:layout_width="wrap_content"
              android:layout_marginRight="20px"
              android:layout_toRightOf="@+id/text2"
              android:layout_height="wrap_content"
              android:id="@+id/text3"
              android:text="Lecturer Name" />
</RelativeLayout>

</RelativeLayout>

    <TextView android:layout_width="wrap_content" android:layout_height="wrap_content" android:id="@+id/text1" android:text="Auto Id" />
    <TextView android:layout_width="wrap_content" android:layout_height="wrap_content" android:id="@+id/text2" android:text="Lect Id" />
    <TextView android:layout_width="wrap_content" android:layout_height="wrap_content" android:id="@+id/text3" android:text="Lect Name" />
    <TextView android:layout_width="wrap_content" android:layout_height="wrap_content" android:id="@+id/text4" android:text="Dept Id" />
</LinearLayout>
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.chuo.thesis"
    android:versionCode="1"
    android:versionName="1.0">
    <uses-sdk android:minSdkVersion="8" />
    <uses-library android:name="com.google.android.maps"/>
    <application android:icon="@drawable/icon"
        android:label="@string/app_name">
        <activity android:name=".Nilipo"/>
    </application>
</manifest>
<intent-filter>
  <action android:name="android.intent.action.MAIN"/>
  <category android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>

<activity android:name=".MyOverlay">
</activity>

</application>

<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_MOCK_LOCATION"/>

</manifest>

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android" 
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent">
  <TextView
android:id="@+id/myLocationText"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/hello"
/>
  <ListView
android:id="@+id/contentlist"
android:layout_width="fill_parent"
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:enabled="true"
    android:clickable="true"
    android:apiKey="OZb4bS9293QtrEIm6jNY7z0glqaiZNCRMVazmbQ" />

</LinearLayout>

import java.io.IOException;
import java.util.List;
import java.util.Locale;
import android.content.Context;
import android.database.Cursor;
import android.graphics.drawable.Drawable;
import android.location.Address;
import android.location.Criteria;
import android.location.Geocoder;
import android.location.Location;
import android.location.LocationManager;
import android.net.Uri;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.ListView;
import android.widget.SimpleCursorAdapter;
import android.widget.TextView;
import android.widget.Toast;
import com.google.android.maps.GeoPoint;
import com.google.android.maps.MapActivity;
public class Nilipo extends MapActivity {

    private static String TREASURE_PROXIMITY_ALERT = "com.chuo.treasurealert";

    MapController mapController;
    MyOverlay positionOverlay;

    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.main);

        MapView myMapView = (MapView) findViewById(R.id.myMapView);
        mapController = myMapView.getController();

        /*
         // Configure the map display options
         myMapView.displayZoomControls(true);
         myMapView.setSatellite(true);
         myMapView.setStreetView(true);

         // Zoom in
         mapController.setZoom(18);

         // Add the MyPositionOverlay
         positionOverlay = new MyOverlay();
         List<Overlay> overlays = myMapView.getOverlays();
         overlays.add(positionOverlay);
         */
// Configure the map display options
myMapView.displayZoomControls(true);
myMapView.setBuiltInZoomControls(true);
myMapView.setSatellite(true);
myMapView.setStreetView(true);
myMapView.setTraffic(true);

// Zoom in
mapController.setZoom(19);

// Add the MyPositionOverlay
positionOverlay = new MyOverlay();
List<Overlay> overlays = myMapView.getOverlays();
overlays.add(positionOverlay);

// Adding itemized overlay markers

List<Overlay> mapOverlays = myMapView.getOverlays();
Drawable drawable = this.getResources().getDrawable(R.drawable.marker);
MyItemizedOverlay itemizedoverlay = new MyItemizedOverlay(drawable, this);

// Specify POIs
GeoPoint point = new GeoPoint(-1273309, 36807176);
OverlayItem overlayitem = new OverlayItem(point, "School of Computing & Informatics", "Three storied building that holds lecture rooms; lecturer offices and a departmental library ");

GeoPoint point2 = new GeoPoint(-1272791, 36807042);
OverlayItem overlayitem2 = new OverlayItem(point2, "Central Examination Centre", "Five storied building that contains lecturers offices and several lecture rooms");
GeoPoint point3 = new GeoPoint(-1272971, 36807710);
OverlayItem overlayItem3 = new OverlayItem(point3, "Millenium Hall 1", 
"Single storied building that houses one large lecture room");

GeoPoint point4 = new GeoPoint(-1273631, 36807311);
OverlayItem overlayItem4 = new OverlayItem(point4, "Chemistry Department", 
"Two storied building that houses lecturers offices; small lecture rooms and labs");

GeoPoint point5 = new GeoPoint(-1273990, 36807539);
OverlayItem overlayItem5 = new OverlayItem(point5, "Physics & Maths Departments", 
"Two storied building that houses lecturers offices; small lecture rooms and labs");

GeoPoint point6 = new GeoPoint(-1272443, 36807321);
OverlayItem overlayItem6 = new OverlayItem(point6, "Science Labs", "Three storied building that houses computer science and chemistry labs. Has no lecturer offices");

GeoPoint point7 = new GeoPoint(-1273770, 36803923);
OverlayItem overlayItem7 = new OverlayItem(point7, "Chiromo Library", "Two storied building that houses the Main CBPS library. Has no lecture rooms nor lecturer offices");

GeoPoint point8 = new GeoPoint(-1273456, 36804486);
OverlayItem overlayItem8 = new OverlayItem(point8, "Human Anatomy", "Two storied building housing offices, student labs and the old chiromo mortuary");

GeoPoint point9 = new GeoPoint(-1273435, 36805127);
OverlayItem overlayItem9 = new OverlayItem(point9, "Vet Anatomy", "Two storied building housing offices and student labs");

GeoPoint point10 = new GeoPoint(-1274190, 36806330);
OverlayItem overlayitem10 = new OverlayItem(point10, "ICIPE", "Single storied building housing ICIPE offices");

GeoPoint point11 = new GeoPoint(-1274370, 36806830);
OverlayItem overlayitem11 = new OverlayItem(point11, "SWA Chiromo Cafeteria", "Student Welfare Authority Cafeteria");

GeoPoint point12 = new GeoPoint(-1274360, 36805020);
OverlayItem overlayitem12 = new OverlayItem(point12, "Biological Sciences Block 1", "Three storied building housing lecturer offices and student labs");

GeoPoint point13 = new GeoPoint(-1274100, 36805176);
OverlayItem overlayitem13 = new OverlayItem(point13, "Biological Sciences Block 2", "Three storied building housing lecturer offices and student labs");

GeoPoint point14 = new GeoPoint(-1274240, 36806870);
OverlayItem overlayitem14 = new OverlayItem(point14, "Photocopy & Binding Shops", "Offer photocopy services to students at a small fee");

GeoPoint point15 = new GeoPoint(-1273690, 36805020);
OverlayItem overlayitem15 = new OverlayItem(point15, "Fire Assembly Point", "Fire Assembly Point C");

GeoPoint point16 = new GeoPoint(-1273550, 36806380);
OverlayItem overlayitem16 = new OverlayItem(point16, "Institute of African Studies", "Single storied building housing the Institute of African Studies");

itemizedoverlay.addOverlay(overlayitem);
itemizedoverlay.addOverlay(overlayitem2);
itemizedoverlay.addOverlay(overlayitem3);
itemizedoverlay.addOverlay(overlayitem4);
itemizedoverlay.addOverlay(overlayitem5);
itemizedoverlay.addOverlay(overlayitem6);
itemizedoverlay.addOverlay(overlayitem7);
itemizedoverlay.addOverlay(overlayitem8);
itemizedoverlay.addOverlay(overlayitem9);
itemizedoverlay.addOverlay(overlayitem10);
itemizedoverlay.addOverlay(overlayitem11);
itemizedoverlay.addOverlay(overlayitem12);
itemizedoverlay.addOverlay(overlayitem13);
itemizedoverlay.addOverlay(overlayitem14);
itemizedoverlay.addOverlay(overlayitem15);
itemizedoverlay.addOverlay(overlayitem16);

mapOverlays.add(itemizedoverlay);

LocationManager locationManager =
(LocationManager) getSystemService(Context.LOCATION_SERVICE);

Criteria criteria = new Criteria();
criteria.setAccuracy(Criteria.ACCURACY_FINE);
criteria.setAltitudeRequired(false);
criteria.setBearingRequired(false);
criteria.setCostAllowed(true);
criteria.setPowerRequirement(Criteria.POWER_LOW);
String provider = locationManager.getBestProvider(criteria, true);

Location location =
locationManager.getLastKnownLocation(provider);

updateWithNewLocation(location);

locationManager.requestLocationUpdates(provider, 2000, 10,
locationListener);

private final LocationListener locationListener = new
LocationListener() {

public void onLocationChanged(Location location) {
    updateWithNewLocation(location);
}

}
public void onProviderDisabled(String provider) {
    updateWithNewLocation(null);
}

public void onProviderEnabled(String provider) {
}

public void onStatusChanged(String provider, int status, Bundle extras) {
}

/** Update the map with a new location */
private void updateWithNewLocation(Location location) {
    TextView myLocationText = (TextView) findViewById(R.id.myLocationText);
    String latLongString;
    String addressString = "No address found";

    if (location != null) {
        // Update my location marker
        positionOverlay.setLocation(location);
        // Update the map location.
        Double geoLat = location.getLatitude()*1E6;
        Double geoLng = location.getLongitude()*1E6;
        GeoPoint point = new GeoPoint(geoLat.intValue(),
            geoLng.intValue());
        mapController.animateTo(point);
        double lat = location.getLatitude();
        double lng = location.getLongitude();
        latLongString = "Lat:" + lat + "\nLong:" + lng;
        Geocoder gc = new Geocoder(this, Locale.getDefault());
        try {
            List<Address> addresses = gc.getFromLocation(lat, lng, 1);
            StringBuilder sb = new StringBuilder();
            if (addresses.size() > 0)
                Address address = addresses.get(0);
for (int i = 0; i < address.getMaxAddressLineIndex(); i++)
    sb.append(address.getAddressLine(i)).append("\n");
    sb.append(address.getLocality()).append("\n");
    sb.append(address.getPostalCode()).append("\n");
    sb.append(address.getCountryName());
    addressString = sb.toString();
}

} catch (IOException e) {} } else {
    latLongString = "No location found";
}

myLocationText.setText("Your Current Position is: \n" +
    latLongString + "\n" + addressString);
}

@Override
    protected boolean isRouteDisplayed() {
    return false;
}

static final private int MENU_ITEM = Menu.FIRST;
static final private int MENU_ITEM_1 = Menu.FIRST + 1;
static final private int MENU_ITEM_2 = Menu.FIRST + 2;

@Override
    public boolean onCreateOptionsMenu(Menu menu) {
    super.onCreateOptionsMenu(menu);
    // Group ID
    int groupId = 0;
    // Unique menu item identifier. Used for event handling.
    int menuItemId = MENU_ITEM;
    int menuItemId1 = MENU_ITEM_1;
    int menuItemId2 = MENU_ITEM_2;

    // The order position of the item
    int menuItemOrder = Menu.NONE;
// Text to be displayed for this menu item.
String menuItemText = "My Location";
String menuItemText1 = "Lecturers";
String menuItemText2 = "Departments";

// Create the location menu item and keep a reference to it.
MenuItem menuItem = menu.add(groupId, menuItemId,
menuItemOrder, menuItemText);
We are carrying out a research on ICT frameworks, their use and their importance (or lack of it thereof) in achieving results. This questionnaire seeks your opinion on ICT frameworks and their use. Kindly answer all the questions.

In this context, a framework is defined as a real or conceptual structure intended to serve as a support or guide for the building of something that expands the structure into something useful.

Section I – Demographics (please tick as appropriate)

a. Gender
   □ Male  □ Female

b. How many years have you worked in the ICT industry
   □ Less than a year  □ Between 1 and 2 years
   □ Between 2 and 5 years  □ More than 5 years

c. What is your professional background (e.g. oracle administrator, java developer, support engineer etc)

Section II – Frameworks

a. Have you in your professional capacity had a chance to use a framework?

b. If yes, which framework(s)?
c. Why did you choose to use a framework?

d. How did your use of the framework(s) affect your overall productivity, as relates to the task you used the framework to achieve?

e. Would you recommend the use of this framework(s) (from ‘b’ above) to other professionals in your field?

f. In your opinion, do you think the number of components a framework contains is important?

g. Why is the number of components important? (skip this question if you answered No to question f above)

h. Does the complexity of a framework influence your decision while choosing a framework?

i. If you were to develop a framework to improve/replace the framework(s) mentioned in ‘b’ above, what would be your three main considerations?