A Thesis
titled

Location Based Services to Improve Public Transportation

by

Anandkrishna Srinivasan

Submitted to the Graduate Faculty as partial fulfillment of the requirements for the Master of Science Degree in Electrical Engineering

________________________
Dr. Jackson Carvalho, Advisor

________________________
Dr. Ezzatollah Salari, Committee Member

________________________
Dr. Devinder Kaur, Committee Member

________________________
Dr. Patricia R. Komuniecki, Dean
College of Graduate Studies

The University of Toledo

May 2011
The increasing demand for energy is a world concern which has motivated the search for optimized use of available resources. The transportation in North America is dependent on a complex structure which relies on the use of fossil fuels. One way to optimize the consumption of such source of energy is to encourage the use of public transportation. This may be obtained by either an increase in the number of units available or by improving the system that is already in use or by a combination of both approaches. The approach in this thesis is to develop an interactive mechanism to supply a mobile user with current information about the state of the buses for the selected routes. This research paper addresses the use of LBS systems to optimize city bus services.
For my parents, brother, and friends
Acknowledgements

I sincerely thank my advisor Dr. Jackson Carvallho for giving me the opportunity to pursue my research under his guidance. It has been a great learning experience working with him. The financial support from the EECS Department in the form of a graduate assistantship is gratefully acknowledged. Special thanks to all my friends for their enthusiastic and generous support through my research study. Above all, I thank my parents and brother who have been a constant source of encouragement and support for me throughout my journey in the Masters Degree program.
Contents

Abstract iii
Acknowledgements v
Contents vi
List of Tables 10
List of Figures 10i
List of Abbreviations 10iii
Introduction 1

1.1 Problem Statement ........................................................................................................... 1
1.2 Research Approach............................................................................................................. 2
1.3 Thesis Scope..................................................................................................................... 3
1.4 Organization .................................................................................................................... 3

Literature Review 5

2.1 Related Technology .......................................................................................................... 5
2.1.1 Global Positioning System (GPS) ................................................................................ 5
2.1.2 Cell ID ......................................................................................................................... 7
2.1.3 Enhanced-observed time difference (E-OTD) .......................................................... 8
2.2 Related Work ................................................................................................................... 9
2.2.1 Travel Assistant Device .......................................................... 9
2.2.2 MoBraille Framework ............................................................ 11
2.2.3 OneBusAway ................................................................. 12
2.2.4 Dynamic Message Sign ........................................................ 13
2.2.5 WaitLess Bus Tracking Device ........................................... 14

Development Platforms .......................................................... 17

3.1 Available Mobile Platforms .................................................. 17
  3.1.1 Android ............................................................... 17
  3.1.2 Iphone OS ............................................................ 18
  3.1.3 Symbian OS ........................................................... 20
  3.1.4 Windows Mobile ...................................................... 20
  3.1.5 Palm OS .............................................................. 21
  3.1.6 Blackberry ............................................................ 22

3.2 Why Develop for Android ..................................................... 22
  3.2.1 Technical Advantages ................................................ 22
  3.2.2 Business Advantages ............................................... 23

What Is Android? ................................................................. 25

4.1 Android Architecture .......................................................... 26
  4.1.1 Applications ............................................................ 27
  4.1.2 Application Framework ............................................. 27
  4.1.3 Libraries ............................................................... 28
  4.1.4 Android Runtime .................................................... 28
  4.1.5 Linux Kernel .......................................................... 29
4.2 Android Applications ................................................................. 29

4.2.1 Android Databases ............................................................. 30

4.3 Developing in Eclipse with ADT ............................................... 31

4.3.1 Android Development Tools ............................................... 31

Implementation Technique .......................................................... 33

5.1 Application Layer ..................................................................... 35

5.1.1 Enabling the GPS in the mobile device .................................... 37

5.1.2 Reverse Geo-coding ............................................................. 38

5.1.3 Map Interface ..................................................................... 39

5.1.4 Google Maps for Mobile ..................................................... 41

5.2 GIS Layer ................................................................................. 42

5.3 Database Layer ........................................................................ 43

Implementing a Multi-Modal Interface ............................................. 46

6.1 Barrier free interface ................................................................ 47

6.1.1 Vision .................................................................................. 47

6.1.2 Speech ................................................................................ 48

6.1.3 Tactile ................................................................................ 49

6.2 Example Scenario .................................................................... 50

Conclusion and Future Work .......................................................... 57

7.1 Conclusion ............................................................................... 57

7.2 Future Work ............................................................................. 58
References 59

Appendix A 63

Source Code for the Android Application 63

A.1 User Interface Implementation
A.1.1 Source code for the Implementation of map view in XML
A.1.2 Source code for displaying the users current location on the map
A.2 Source code for initializing Android TTS (text to speech)
A.3 Source code for using the Google Maps API
A.4 Source code for implementing the database

Appendix B 72

Various Screenshots of Android 72

B.2 Android Application running on the Emulator
B.3 DDMS Perspective
B.4 Emulator Controls to Provide Mock GPS Locations
List of Tables

5.1: Bus Routes ....................................................................................................................... 44
5.2: Bus Schedules.................................................................................................................. 44
List of Figures

1-1: OIL AND GAS DEMAND.................................................................1

2-1: GPS POSITION COMPUTATION USING (A) ONE SATellite (B) TWO SATellites (C) THREE
.................................................................................................................................................6

2-2: BASE TRANSCEIVER STATION SIZE VARIATION (A) AND SECTORIZATION (B)..............8

2-3: E-OTD’S USE OF TRIANGULATION FOR A LOCATION FIX IS DEPENDENT UPON LMUs (A)
......................................................................................................................................................9

2-4: TRAVEL ASSISTANT DEVICE (TAD) .............................................................................11

2-5: MOBRILLE ARCHITECTURE .........................................................................................12

2-6: FEATURES OF THE ONEBUSAWAY APPLICATION..................................................13

2-7: LONDON BUS INFORMATION ....................................................................................14

2-8: FUNCTIONAL BLOCK DIAGRAM OF WAITLESS SYSTEM DESIGN .........................15

2-9: WAITLESS BUS TRACKING DEVICE ........................................................................16

3-1: ARCHITECTURE OF THE IPHONE ........................................................................18

3-2: OPERATING SYSTEM SHARE FOR SMARTPHONE’S IN 2010 ..................................23

4-1: RELATIONSHIP BETWEEN THE ANDROID ENVIRONMENT, LINUX-KERNEL AND THE

BUILT ......................................................................................................................................26

4-2: MAJOR COMPONENTS OF THE ANDROID OPERATING SYSTEM .........................27

5-1: COMMUNICATION FLOW BETWEEN THE USER’S MOBILE PHONE, BUSES AND THE .......34
5-2: SYSTEM ARCHITECTURE ........................................................................................................35
5-3: CURRENT POSITION OF THE USER IN TERMS OF LATITUDE AND LONGITUDE ....38
5-4: CURRENT LOCATION ALONG WITH THE STREET ADDRESS ........................................39
5-5: VARIOUS VIEWS OF THE GOOGLE MAPS ..................................................................41
5-6: DIRECTION LIST ................................................................................................................42
5-7: BUS ROUTES AND SCHEDULES .......................................................................................45
6-1: SCREEN SHOT OF THE ICON USED TO INITIATE THE APPLICATION ..........................50
6-2: SCREENSHOT OF THE INITIAL INTERACTION STATE OF THE APPLICATION ..........51
6-3: SCREENSHOT OF THE APPLICATION DISPLAYING THE ADDRESS OF THE BUS STOP ....52
6-4: SCREEN SHOT OF THE INFORMATION ON BUS ROUTES AND SCHEDULES ...............53
6-5: SCREEN SHOT OF THE STARTUP SCREEN ....................................................................54
6-6: SCREEN SHOT OF THE GOOGLE MAPS ROUTE FINDER .............................................55
6-7: SCREEN SHOT OF THE WALKING DIRECTIONS .............................................................56

B-1: SCREEN SHOT OF THE ANDROID EMULATOR .............................................................72
B-2: SCREEN SHOT OF THE APPLICATION RUNNING ON THE ANDROID EMULATOR .........73
B-3: SCREEN SHOT OF THE DDMS TOOL .............................................................................74
B-4: SCREEN SHOT OF THE EMULATOR CONTROLS THAT PROVIDE MOCK GPS LOCATIONS 75
List of Abbreviations

ADT....... Android Development Tools
ADA....... Americans with Disabilities Act
API......... Application Programming Interface
AVD....... Android Virtual Devices
AAPT..... Android Asset Packaging Tool
ADB....... Android Debug Bridge
BTS........ Base Transceiver Station
DMS....... Dynamic Message Sign
DVM....... Dalvik Virtual Machines
DDMS...... Dalvik Debug Monitoring Service
E-OTD..... Enhanced-observed time difference
GPS....... Global Positioning Systems
IDE........ Integrated Development Environment
J2ME...... Java 2 Micro Edition
JVM........ Java virtual machine
LBS........ Location Based Services
LMU....... Location Measurement Unit
LED....... Light Emitting Diode
MS......... Mobile Station
MVC....... Model View Controller
OHA....... Open Handset Alliance
OS......... Operating System
PDA....... Personal Digital Assistant
POSIX.... Portable Operating System for Unix
RAM....... Random Access Memory
ROM....... Read Only Memory
RIM....... Research in Motion
SDK....... Software Development Kit
TAD....... Travel Assistant Device
TTS....... Text to Speech
URL....... Uniform Resource Locator
UI......... User Interface
Chapter 1

Introduction

1.1 Problem Statement

There are two vital reasons to encourage the use of public transportation. Firstly, the increasing demands for oil and gas in the world. Figure 1.1 indicates that the oil and gas demand is expected to rise through the year 2010. It is expected that by 2010, about half of the daily volume needed to meet projected demand is not on production today [1].

Figure 1-1: Oil and Gas demand
Secondly, a sizeable section of population still prefers public transportation to reach their work spot or procure goods and services and for social visits as well. The type of public transportation that I am going to consider in this thesis is the bus service. Generally, we see that bus networks can be confusing. To a traveler in a new city it is very difficult to access information on bus routes and their schedules. In fact even a regular commuter in a city would be familiar only with a few routes. Weather and traffic jams can render printed bus time tables useless or of little use. New transit technology must accept a broadened and more comprehensive role in the planning and development of transport systems. One way to improve the current bus service is to introduce a system with up to the minute real time data about the current location of the user, bus stops, routes and timetables. Such functionality may be implemented through the support of Location Based Services (LBS). A LBS is a wireless IP-based service that uses geographic information to serve a mobile user [2]. Global Positioning Systems (GPS) receivers will be used so that accurate information can be obtained.

1.2 Research Approach

My research addresses the use of LBS systems to optimize city bus services. It takes advantage of the fact that technology has transformed the cell phone from being a simple medium of voice communication to becoming an indispensable medium for contact information and entertainment. It addresses the use of off-the-shelf mobile devices such as mobile phones and Personal Digital Assistant (PDAs) which are being widely used and are easily available. The approach in this thesis develops an application based on a mobile device running the Google Android platform. The device used is 3G technologies enabled and has
a built in GPS receiver. This application provides the mobile user with information about the state of the buses for all the routes available at the user’s current location.

1.3 Thesis Scope

The main goal of this thesis is to facilitate the use of public transportation for visually impaired users. My approach to this is to develop an interactive mobile application which has a barrier free interface. By barrier free interface I mean an interface that makes use of multimodal communication [39] and hence it can be used by one and all irrespective of their physical disabilities. Vision, speech and tactile modes of interaction will be available to the user. This application includes the following general features:

- A GPS based system that can determine the current location of the user. Such information will be available to the user as speech output;
- A map interface that displays the potential bus stops;
- A list of directions to a bus stop of the user’s choice from his/her current location;
- A database that provides the user with information about the state of the buses for all the routes.

1.4 Organization

The remainder of this document is organized as follows. Chapter 2 discusses the literature review. Chapter 3 gives a detailed introduction to the Android platform. Chapter 4 gives an overview of the existing mobile platforms and compares them. Chapter 5 consists of implementation of the LBS application for Android. Chapter 6 gives the details and
screenshots of the barrier free user interface used for the application. Chapter 7 provides the conclusion of the work done and the future work.
Chapter 2

Literature Review

Advances in Internet, network technology and the rapidly growing number of mobile personal devices have resulted in the fast growth of Location Based Services (LBS). A location-based service can be described as an application that is dependent on a certain location [3]. The core of any location based service is the positioning technology involved in tracking the current location of the user. Positioning technology refers to the various approaches used to approximate the location of a mobile device and thereby also its user.

2.1 Related Technology

This section addresses positioning related technologies.

2.1.1 Global Positioning System (GPS)

The most well known and widely used satellite positioning system is GPS, operated by the United States Department of Defense. According to [3], in April 1995 the whole system containing 24 operational satellites in an 11000 nautical mile orbit at a cost of $12billion was declared fully operational. The principle behind the functioning of a GPS is that a receiver
measures the time a pseudo-random code takes to travel from the GPS satellite to the receiver which is around 0.1 seconds in practice. From this data the receiver can compute the distance \( X \) to the satellite. The satellite now places the receiver somewhere on the surface of an imaginary sphere centered on the satellite with radius \( X \). Figure 2.1(a) [3] illustrates this arrangement. Then the distance of the receiver from a second satellite is measured, narrowing the potential locations of the receiver to an elliptical ring at the intersection of the two spheres as shown in Figure 2.1(b)[3]. By including measurements from a third satellite, the potential locations of the receiver can be further reduced to just two possible points as shown in Figure 2.1(c) [3]. One of these positions is then ignored due to it being either too far from the Earth’s surface, or it is moving at an unrealistic velocity. Readings from a fourth satellite can then be taken so that the receiver can now be positioned in three dimensions which are latitude, longitude and altitude. Figure 2.1 [3] only shows a simplified 2-dimensional model.

Figure 2-1: GPS position computation using (a) one satellite (b) two satellites (c) three satellites.

For most of the applications used in the present day the accuracy of standard GPS is sufficient, but there are some applications that require even greater accuracy. The differential GPS was developed in order to satisfy this requirement. Differential GPS provides a
correction for errors that may have occurred in the satellite signal due to slight delays as the signal passes through the ionosphere and troposphere [3].

2.1.2 Cell ID

Cell ID [3] is the most basic form of cellular location and works simply by detecting the base transceiver station (BTS) with which the telephone is registered. The mobile telephone/station (MS) is registered to a BTS at all times. This is usually the nearest BTS, but may seldom be the BTS of a neighboring cell due to terrain, cell overlap or due to congestion at the nearest BTS. The size of a cell depends on the terrain and also on the possible number of users. Hence, the size of cells is much smaller in city centers than in rural locations as shown in Figure 2.2(a). The accuracy of a position fix greatly depends on the size of the cell because the location reported is in fact the location of the BTS and the MS may be anywhere within the boundary of the cell. Typically the extent of error in urban locations may be around 500 m, but in rural locations this can increase up to about 15 km. Each base-station will have multiple antennas, each covering a sector of the cell. So a BTS with three antennas will produce a cell with three 120 degrees sectors. By detecting the antenna with which the MS is registered, the location of the MS can be narrowed down to somewhere within a sector of the cell with the BTS at its apex as shown in Figure 2.2(b) [3].
2.1.3 Enhanced-observed time difference (E-OTD)

E-OTD uses triangulation between BTSs to provide a more accurate location fix. The distance of the MS from the BTS is calculated by comparing the difference in time taken for a control signal sent by the BTS to arrive at the MS and also to a fixed location known as Location Measurement Unit (LMU) [4]. The distance from the LMU to the BTS is known. This situation is shown in Figure 2.3(a) [3]. E-OTD can improve the accuracy of standard Cell ID by up to a factor of ten. The disadvantage of E-OTD is that it requires considerable investment by the network operator in installing the LMUs and a minor software upgrade to the handsets to facilitate them to calculate the signal delays [3].

Figure 2-2: Base transceiver station size variation (a) and sectorization (b).
2.2 Related Work

This section briefly describes the work done in the field of location based services for the visually impaired people.

The Americans with Disabilities Act (ADA) [5] requires the transportation systems to eliminate barriers to mass transit for persons who are blind or visually impaired, primarily by making visible information accessible and usable. Existing applications which facilitate the use of public transportation for the visually impaired are discussed in the subsections that follow.

2.2.1 Travel Assistant Device

The rapidly evolving technology has given birth to innovations that assure to enhance transit accessibility for people who are blind or visually impaired. One such application is the Travel Assistant Device (TAD) [15]. The TAD application is the primary system interface for the user, as well as the component liable for real-time navigation. With the cell phone application the user accesses the TAD system using the web services. The application then prompts the user for transit information. The user has to log into the system only for the
first time the application is put to use as shown in screen A of Figure 2.4 [15]. The sequence of screens in Figure 2.4 demonstrates the interaction states associated with the user’s interaction to request information about a bus service. The interaction is initiated after the user provides the correct username and password. The user is then navigated to another screen as shown in screen B of Figure 2.4 [15]. The user is now given an option to select a trip created earlier. All the necessary data about the buses and their routes are retrieved by the phone. Subsequently, the cell phone exhibits a screen with a green or red square and the distance to the next stop as shown in screen C of Figure 2.4. The green square indicates that the cell phone will consistently provide services to the user, and he/she will also obtain the exit notification when appropriate. However, if the square changes to red color it indicates that there is a problem with the GPS signal and hence the user will not receive a notification when they get close to the stop. When the user is approximately 150 meters from their destination, TAD notifies the user to get ready. When the user reaches 75 meters, TAD requests the user to pull the cord and the cell phone vibrates and displays the same message as shown in screen D of Figure 2.4. This announcement continues until the user confirms he/she has disembarked the vehicle by pressing a cell phone key [15].

The main functionality of the TAD application is to alert a visually impaired user to get off a bus at a particular stop. The TAD application will not be able to help a user in new surroundings where the user has no information about his current location.
2.2.2 MoBraille Framework

Another similar work is a tool which is built on a novel framework called MoBraille [7] that makes it possible for a Braille display to benefit from numerous features in an Android phone without knowledge of proprietary, device-specific protocols. This tool is very useful for a deaf-blind person who depends on public transportation to move from one place to the other. This tool provides the user with information on how much longer he or she must wait for the bus to arrive. The user points the Android phone towards the street and presses a button. A Braille display is provided where the user enters the route number. The application utilizes the GPS and compass to identify the current stop and queries the OneBusAway Application Programming Interface (API) [6] to acquire arrival information for the given route at the current stop. The arrival information is also displayed in Braille. The concept involved in the working of MoBraille is that it enables an Android application to interface with a Braille display through HTTP requests over a Wi-Fi connection [7]. Firstly, with the help of Wi-Fi the Braille display is tethered to the Android phone. Tethering
is the use of a mobile device with internet access such as 3G cellular service to serve as an internet gateway or access point for other devices [40]. Once the Braille display is tethered to the Android phone, the Braille display user loads a MoBraille webpage on the display’s built-in browser as shown in Figure 2.5 [7]. The webpage sends requests to the Android phone that are processed by a Java servlet running on the phone. The servlet can access the Android’s sensors like any other Android application, thereby permitting the Braille display user access to the various phone features like a 3G network, GPS, a compass etc. The MoBraille tool would probably not be very helpful in a situation where the user is not exactly at the bus stop. It works under the assumption that the user is exactly at the bus stop which need not be true in all the cases.

Figure 2-5: MoBraille architecture

2.2.3 OneBusAway

OneBusAway [6] is a collection of transit traveler information tools developed at the University of Washington. It provides real-time bus arrival information, a trip planner, a schedule and route browser, and a transit-friendly destination finder for Seattle area bus riders as shown in Figure 2.6 [6]. This application primarily targets the group of users who use the bus service quite frequently. It has been implemented on a smart phone so that it can
exploit its localization framework and its built-in multi-touch map support. The application communicates with the OneBusAway’s back-end server over the phone’s network connection to request information about stops in a given area, information about particular routes and ultimately real-time arrival information for specific stops. The various functionalities that the application offers are provided in the tab bar at the bottom of the screen that provides entry points for a map screen, a bookmark screen, a recent-stops screen, and a search screen.

![Features of the OneBusAway application](image)

Figure 2-6: Features of the OneBusAway application

### 2.2.4 Dynamic Message Sign

Western Europe and Japan have adopted real time passenger information systems for over a decade now. These systems are seen in many different forms which ranges from
traditional static displays to those wireless intelligent information systems providing pretty accurate arrival times. Dynamic Message Sign (DMS) [9] is a popular way to display transit information and are commonly known as electronic sign. Figure 2.7 [9] shows a Light Emitting Diode (LED) sign used in London to display real time arrival information of various buses at a particular bus stop.

![Figure 2-7: London Bus information](image)

**2.2.5 WaitLess Bus Tracking Device**

The WaitLess bus tracking device [10] is a system intended to display the real-time locations of the buses on Georgia Tech’s campus. The system comprises of a solar panel and backup battery, wireless module, microprocessor, and a LED embedded map of the Georgia Tech bus transportation routes.
Figure 2-8: Functional block diagram of WaitLess system design

Assembly of these components as shown in Figure 2.8 [11] facilitates the tracking device to hook up to the internet in order to obtain GPS data of the buses, which is portrayed by triggering LEDs in the approximate geographic positions of the buses on the route map. A prototype of the WaitLess Bus Tracking Device is shown in Figure 2.9 [10].

Both the WaitLess bus tracking device and the DMS do not address our main concern of facilitating the use of public transportation for visually impaired users. They also do not provide any speech output or a Braille interface. Even if an extension to the DMS system introduces speech output, the information rendered may just not be sufficient for a user. The other limitation of these systems is that, they cannot be customized according to the user’s choice.
Figure 2-9: WaitLess Bus Tracking Device
Chapter 3

Development Platforms

3.1 Available Mobile Platforms

This section discusses the various mobile development platforms that are presently available.

3.1.1 Android

Android is a new mobile device operating system first developed by Android Inc., a firm purchased by Google in 2005. It was first announced by Open Handset Alliance (OHA) in 2007. The first commercial product of Android was T-Mobile G1, released in September 2008 along with its software development kit (SDK). Android mobile operating system is built on top of Linux kernel version 2.6. The Linux kernel provides the interfaces to access low level hardware control functionalities. While the Android Operating System (OS) is built on top of a Linux kernel, all Android applications, including core applications, are developed in Java. Each Java program is run as a single process on Linux with its own instance of a Dalvik Virtual Machines (DVM). A DVM is an optimized java virtual machine for Mobile Devices [12]. A set of native C/C++ libraries is provided to support various additional
functions for software developers. For example, OpenGL 3D is provided for graphic engine and SQLite is provided for data storage.

Android supports many features required for mobile software development. Third party applications can perform multithreading and can access the same low-level resource as core applications. The application framework is so flexible that it allows all software components and extensions to be reused or customized. Also, Android is an open source platform, unlike most popular operating systems for mobile devices. An open-sourced mobile platform is more likely to attract programmers to contribute to this platform which is expected to play an important role in the future market of the mobile devices [13].

3.1.2 Iphone OS

Apple's mobile operating system is called iOS. It was first developed for the iphone. The iPad and the IPod Touch use this iOS as well now. The iOS is a Unix-like operating system. There are four abstraction layers in the iOS namely, the Core OS layer, the Core Services layer, the Media layer, and the Cocoa Touch layer [14].

![iPhone Architecture](image)

Figure 3-1: Architecture of the iphone
The iphone OS is a derivative of the Darwin open source POSIX compliant computer operating system developed by Apple Inc. The architecture of the iphone is shown above. The iphone OS has seen a rapid rise in popularity and garnered a large and dedicated user base in spite of being a relatively new product to enter the mobile market in comparison to other mobile operating systems. The rapid rise of the iphone OS could be primarily attributed to the innovations on its user interface and the availability of 3rd party applications. The new multi-touch features which are now patented by Apple, Inc., along with the large screen space, greatly improved the interaction levels between users and the mobile devices. The iTunes store, which is centralized, collects and publishes the comprehensive list of Apple and 3rd party applications, providing the entire iphone OS user community, easy access to any and all enhancements available for their iphones. The iphone hardware is tightly controlled and integrated with the iphone OS. With this, Apple Inc. can more easily assemble the various components of the hardware into one cohesive unit. The seamless utilization of the multi-touch, accelerometer, microphone and camera technologies in concert to provide sophisticated input data, to the mobile system is one of the most distinguishing features of the iphone. Other important features include a full-featured SDK and the new Core Location suite that can use any of GPS, cell tower triangulation and Wi-Fi signals to determine the device location. Still, with many positives, the iphone OS does come with its share of limitations, which importantly include:

- The single-vendor hardware/software availability through Apple Inc;
- The single-provider phone and internet service through AT&T;
- The lack of external memory expansion, easy battery replacement, and the heavily restricted multi-tasking functionality [16].
3.1.3 Symbian OS

One of Nokia’s mobile operating system is known as Symbian. The Symbian OS was designed specifically for mobile devices. It has very small memory footprint and low power consumption. Third party developers are authorized to develop and install applications independently from the device manufacturers. The Symbian OS was designed in a manner that applications could consistently work for years without losing the user data. Also, the OS can run on more than one hardware platform. Symbian OS is a highly optimized, heavily asynchronous operating system. Symbian features pre-emptive multitasking and memory protection like any other operating system [17]. The Symbian OS supports the client-server architecture. All the system services are run from Read Only Memory (ROM) directly. The user applications are located in Random Access Memory (RAM). The user application level does not permit the usage of multi-threading and context switching [16].

3.1.4 Windows Mobile

This platform is based on Windows CE (WinCE). WinCE is a compact OS specifically designed, for devices that have minimal storage. Windows CE is a distinct operating system and kernel rather than a trimmed down version of desktop Windows [18]. With the help of the Win32 API the WinCE platform maintains a consistent interface for applications on various hardware platforms and thus emphasizes on portability. The hardware platforms on which the WinCE can be used on ranges from Pocket PCs, Smartphone’s, Portable Media Centers, to even onboard computers in automobiles. Initially the Windows Mobile platform was running on the Pocket PC and was brought into existence in 2000. It then moved to the Smartphone arena. The first version was Windows Mobile 2003. Visual C++ is used for developing software for the Windows Mobile OS making use of Microsoft’s .NET
framework. The SDK is setup to work using Visual Studio as the Integrated Development Environment (IDE).

One of the most important characteristics of the Windows Mobile platform is that it was designed for flexibility keeping the developer in mind. For that reason it was designed to support lots of preemptive multitasking. It supports a massive 256 priority levels for threads and up to 32 processes. It also supports all of the standard mutual exclusion and synchronization methods you would expect from a desktop PC. This functionality makes it ideal for a Smartphone because the users normally prefer multitasking in order to be as productive as possible [16].

3.1.5 Palm OS

Palm OS Garnet (v5.4.x) is a proprietary operating system originally developed by Palm Inc. In the early versions (pre-Garnet), the Palm OS was primarily utilized in Palm developed Personal Digital Assistant (PDA) mobile hardware units. Palm OS is designed for ease of use with a touch screen based graphical user interface [19]. At one point, Palm PDAs with the Palm OS held 85% of the market share in the mobile device market [20, 16]. The two features that make the Palm OS different from the other existing ones are the built-in, tightly-integrated handwriting recognition input suite called Graffiti 2 and the highly efficient power management suite that achieves up to twice the battery life as compared to the devices which used the other existing operating systems. Still, Garnet had a significant number of limitations including its format of installable software modules and its lack of multi-tasking even though the kernel is multi-task capable.
3.1.6 Blackberry

BlackBerry is a line of mobile e-mail and Smartphone devices developed and designed by Canadian company Research in Motion (RIM) for 10 years. The most significant feature of BlackBerry is its ability to integrate existing e-mail systems on mobile phones. BlackBerry is primarily a messaging phone which consists of the largest number of messaging features in a Smartphone today [16]. A user with a BlackBerry phone has a mailbox which is “Always On” since it can perform as a wireless agent. This feature helps people to send messages rapidly and securely with the help of BlackBerry and greatly improves his/her working efficiency. The present BlackBerry has the ability to do much more than the ones before, including push-to talk, internet surfing, and other features which are also supported by other mobile operating systems. Besides, BlackBerry provides associate Java software development kit in order to encourage programmers to develop BlackBerry Applications [21].

3.2 Why Develop for Android

The Android platform varies in some technical and business aspects from its competitors. We will now look at the advantages of using Google’s Android operating system in the subsections that follow.

3.2.1 Technical Advantages

Unlike most of the competitors, Android is built upon an open-source platform, and most of the Android code is released under the open source Apache License. Android applications are written in the Java programming language, which is a powerful, mature and extensively adopted language in the global development community. Android’s Java is not
the same as J2ME, however, most of the existing J2ME applications can be ported to Android with small modifications.

### 3.2.2 Business Advantages

The iTunes App Store is Apple's dedicated online service for distribution of applications developed for its iPhone, iPod, iPod Touch and iPad. The Android market is Google's open source initiative to provide applications for Android based smart phones. Apple strives to ensure applications meet its defined standards of quality, utility and appropriateness whereas Google follows a very simple procedure. Google only sees to that the applications meet the terms the developers agree to when they sign up for the Android Market [22]. Applications are not evaluated on any other grounds, so developers will have more freedom to create the type of application or content they like [22]. Android market lets developers directly list their applications, which speeds up the launch of the apps. Sales of Google Android phones in the United States are rising so rapidly, that the devices have outsold Apple handsets for the first time on record [23]. Figure 3.2 shows the operating system share of the Smartphone’s sold in the year 2010 in the United States.

![Operating System Share: 6 Month Recent Acquirers](image)

*Figure 3.2: Operating System share for Smartphone’s in 2010*
Canalys, in 2010 estimated that Android handset sales grew 886 percent worldwide from the year ago quarter [24].

Taking all the information regarding the available mobile platforms into consideration, I decided to build my application on the android mobile platform. Chapter 4 focuses on the various details of Android mobile platform.
Chapter 4

What Is Android?

Android is a software platform developed by Google and The Open Handset Alliance which is a group of approximately 30 organizations [25]. Android is an open-source software stack that includes the operating system, middleware and key mobile applications along with a set of API libraries for writing mobile applications that can shape the look, feel and function of mobile handsets [31].

In July 2005, Google bought Android, Inc., a tiny startup company based in Palo Alto, California, USA. At Google, a team led by Andrew Rubin developed a mobile device platform powered by the Linux kernel. Their main marketing targets were hardware component manufacturers and software developers with an easy and flexible operating system. On the 5th of November 2007, several companies including Google, HTC, Motorola, Intel and 26 other firms joined together to form the “Open Handset”, with the stated objective of developing open standards for mobile devices. Subsequently they unveiled the new product labeled “Android”- a mobile device platform built on the Linux kernel version 2.6 [26]. Android has been available as open-source software since October 2008.
4.1 Android Architecture

Android is not only a mobile operating system that uses a modified version of the Linux kernel, but also a software stack for mobile devices that includes an operating system, middleware and key applications [27, 28, 29]. The components of Android’s underlying operating system are written in C or C++, but user applications are built for Android using the Java programming language [30]. Figure 4.1 below shows the relationship between the android environment, Linux-Kernel and the built in applications.

Figure 4-1: Relationship between the android environment, Linux-Kernel and the built in applications

The major components of the Android operating system are show in Figure 4.2 [30].
The five components illustrated in Figure 4.2 are described below in detail.

4.1.1 Applications

Android has a set of applications that consists of an email client, calendar, maps, browser, SMS program and contacts. All the applications are written in Java.

4.1.2 Application Framework

Android provides an open development platform. This characteristic favors the development of extremely rich and creative applications. The architecture of the android application component is designed such that the reuse of components is greatly simplified.

A set of services and systems underlies all the android applications. They include:

- A extensible and a rich set of Views that can be used to build an application, including lists, grids, text boxes, buttons, and even an embeddable web browser;
- An *Activity Manager* that provides a common navigation back-stack and mainly manages the lifecycle of applications;

- There are *Content Providers* that enable applications to connect with and access data from other applications (Ex: Contacts) or also share their own data;

- A *Resource Manager* that provides access to non-code resources such as localized strings, layout files and graphics, and

- A *Notification Manager* that enables all applications to display custom alert messages in the status bar [31].

4.1.3 Libraries

Apart from Java, Android includes a set of C/C++ libraries that is used by various components of the Android system. The Android application framework makes these available to developers.

4.1.4 Android Runtime

Most of the functionality in the core libraries of the Java programming language is made available in Android with its own set of Core libraries. Android applications are developed using java as the programming language. However they are executed by a custom virtual machine called the ‘Dalvik’ rather than the traditional java virtual machine (JVM). During execution, each Android application would run in its own process, and would have its own instance of the Dalvik virtual machine. One important characteristic of Dalvik is that it has been written so that a device can run multiple VMs efficiently. The Dalvik VM relies on the Linux kernel for underlying functionality such as threading and low-level memory management [31].
4.1.5 Linux Kernel

Android relies on Linux version 2.6 for core system services and it also acts as an abstraction layer between the hardware and the rest of the software stack.

Each Android application runs in a separate process within its own Dalvik instance and hence hands over all responsibility for memory and process management to the android runtime. The android runtime then stops and kills processes according to the priority assigned in order to manage resources.

4.2 Android Applications

Android applications consist of loosely coupled components. These components are activities, services, content providers, intents, broadcast receivers, widgets and notifications. The following six components provide the building blocks for an android application.

- Activities are the application’s presentation layer. Every screen in the application will be an extension of the Activity class. Activities use Views to form graphical user interfaces that display information and respond to user actions. In terms of desktop development, an Activity is equivalent to a Form [31].

- Services are the invisible workers of the application. Service components either inform the system about something the application wants to be doing in the background or expose some of its functionality to other applications.

- Content Providers are shareable data stores. Content Providers are used to manage and share application databases. They’re the preferred means of sharing data across application boundaries. One can configure his or her own Content Providers to
permit access from other applications as well as use Content Providers exposed by others to access their stored data. Android devices include several native Content Providers that expose useful databases like the media store and contact details [31].

- Intents are an inter-application message-passing framework. Using Intents one can broadcast messages system-wide or to a target Activity or Service, stating the intention to have an action performed. The system will then determine the target(s) that will perform actions that are appropriate.

- Broadcast Receivers are Intent broadcast consumers. If one creates and registers a Broadcast Receiver, the application can listen for broadcast Intents that match specific filter criteria. Broadcast Receivers will automatically start the application to respond to an incoming Intent, making them perfect, for creating event-driven applications.

- Notifications are the preferred technique for getting a user’s attention from within a Service or Broadcast Receiver [31].

4.2.1 Android Databases

A persistent data structure is a data structure which constantly preserves the earlier version of itself when it is modified. Such data structures are effectively immutable, as their operations do not update the structure in-place, but instead always yield a new updated structure [41]. Structured data persistence in Android is provided through SQLite Databases [32] and Content Providers. When managed, structured data is the best approach. Android offers the SQLite relational database library. Every application can create its own databases over which it has complete control.
4.3 Developing in Eclipse with ADT

The Android Development Tools (ADT) plugin for Eclipse adds potent extensions to the Eclipse integrated development environment (IDE). It allows the developer to create and debug Android applications easier and faster. The Eclipse ADT plugin gives some additional capabilities to help in developing Android applications.

4.3.1 Android Development Tools

The Android SDK consists of several tools and utilities to assist users create, test and debug their projects. The ADT plug-in easily integrates most of these tools into the Eclipse IDE, where one can access them from the DDMS perspective. These include the following.

- The Android SDK and Virtual Device Manager: Used to create and manage Android Virtual Devices (AVD) and SDK packages.
- The Android Emulator: An implementation of the Android virtual machine designed to run within a virtual device on your development computer. One can use the emulator to test and debug the Android applications.
- Dalvik Debug Monitoring Service (DDMS): Can use the DDMS perspective to monitor and control the Dalvik virtual machines on which one is debugging his or her applications.
- Android Asset Packaging Tool (AAPT): Constructs the Android package file which is recognized by the .apk string of characters attached to the package name.
- Android Debug Bridge (ADB): A client-server application that gives a link to a running emulator. It lets the developer copy files, install compiled application packages and run shell commands [33].
Logcat: Provides the user with system log messages that are output on an android device or emulator [33].
Chapter 5

Implementation Technique

This work addresses the use of LBS systems to optimize city bus services. It takes advantage of the fact that technology has converted the mobile phone from being a simple medium of voice communication to become an essential medium for contact information and entertainment. The main goal of this research is to facilitate the use of public transportation for visually impaired users. The main characteristics of the application are:

- A map interface that displays the current location of the user and the potential bus stops. The current location of the user is given as a voice output;

- A list of directions to a bus stop of the user's choice from the current location using Google mobile maps; and

- A database that provides the user with information about the buses for all the routes.

My approach to this is to develop an interactive mobile application which has a barrier free interface. Figure 5.1 shows a simple solution to our problem scenario. It illustrates the communication flow between the user's mobile phone, buses and the database.
Figure 5-1: Communication flow between the user’s mobile phone, buses and the database

It is assumed that both the user’s mobile phone and the buses are equipped with GPS receivers. The GPS receivers in the buses should have the ability to send their current location information to the database. Figure 5.2 shows the vital role that the GPS information plays in the overall system architecture. Figure 5.2 also provides a high level description of the structure I propose. As it can be seen it is composed of four components. They are the User, the Application layer, the GIS layer and the database layer. The three layers carry out different tasks and obtain results which are used and shared by the other layers depending on the requirement. The moment the application is initiated, the current location of the user is obtained from the GPS receiver in the mobile device. The application layer invokes the GIS layer (Geo-spatial Information System) and the database layer to gather all the information required and returns it to the application layer. The application layer then displays the information to the user. From Figure 5.2 we can also see that the main layers are the application layer, the database layer and the GIS layer which consists of
the map server. The section that follows discusses the different layers and how they help in building the whole application.

![System architecture](image)

Figure 5-2: System architecture

### 5.1 Application Layer

The application layer is the central part of the application. This layer is responsible for gathering information from the GIS and the database layer. The information obtained is then sent to the user in a form that can be easily comprehended. The various functionalities accomplished by the application layer are:

- Enabling the GPS receiver in the mobile device and also implementing reverse geocoding to convert the latitude/longitude pair into a physical address;
- Converting the physical address obtained into speech by using Android TTS (text to speech) API;
- Creating a map interface; and
• Accessing the Google maps for mobile by using a URL (Uniform Resource Locator) connection.

I have used several important classes in the application to enable the GPS and to use the android maps, which includes:

• The Criteria class: This class helps in selecting the most suitable location provider by specifying the applications criteria. Providers are ordered according to accuracy, power usage, ability to report altitude, speed, bearing, and monetary cost.

• The LocationManager class: It provides access to the system location services.

• The MapView class: It gives the developer control over the map view.

• MapActivity: It is the base class that is extended to create a new Activity that can include a MapView. The MapActivity class handles the application life cycle and background service management required for displaying maps.

• Overlay class: This is the class used to annotate maps. Using Overlays, we can use a Canvas to draw onto any number of layers that are displayed on top of a Map View [31].

• MapController class: It is used to control the map. It enables the user to set the center location and zoom levels.

• MyLocationOverlay class: It is a special Overlay that can be used to display the current position and orientation of the device.

• ItemizedOverlays and OverlayItems class: They are used together to let the user create a layer of map markers, which are displayed using Drawables and associated text.
5.1.1 Enabling the GPS in the mobile device

Android provides many technologies to determine the location of the user which generally depends on the device being used. In most scenarios it’s unlikely that the user will want to explicitly choose the `LocationProvider` to use. More commonly, the user will specify the requirements that a provider must meet and let Android determine the best technology to use. This is done by using the `Criteria` class with which we can state the requirements of a provider in terms of accuracy (fine or coarse), power use (low, medium, high), financial cost, and the ability to return values for altitude, speed, and bearing. Once the user has defined the required criteria, he or she can either get the best provider which would match our criteria or he or she can get a list of all the possible matches. One with the greatest accuracy is returned when the result obtained consists of more than one provider. I have used the Global Positioning System as the location provider for my application.

The next task is to find the physical location of the user/device. This can be achieved by gaining access to the `LocationManager` class, which is responsible for the location based services in android. One or more permissions have to be acquired for supporting access to the LBS hardware before one can use the `LocationManager` class. The GPS provider requires `fine` permission, while the Network (Cell ID/Wi-Fi) provider requires only `coarse`. Figure 5.3 shows the result of what we have obtained by using the `LocationProvider` and the `LocationManager` system service. It shows the current position of the user in terms of latitude and longitude.
Most of the location-sensitive applications are expected to be responsive to user movement. We use the `requestLocationUpdates` method to get updates whenever the current location changes. The `requestLocationUpdates` method accepts either a specific `LocationProvider` name or a set of Criteria to determine the provider to use. To optimize efficiency and minimize cost and power use, we also specify the minimum time and the minimum distance between location change updates. Therefore the new location is traced whenever the minimum time and distance values are exceeded.

### 5.1.2 Reverse Geo-coding

The next task is to convert the current position of the user, which is in the form of a latitude/longitude pair to an existing physical address. This process of converting a latitude/longitude pair to a street address is called reverse geo-coding. To carry out a reverse lookup, the target latitude and longitude are presented to a Geocoder’s `getFromLocation` method. The `getFromLocation` method returns a list of possible matching addresses. The accuracy and granularity of reverse lookups are completely dependent on the quality of data in the geo-coding database. Therefore, the quality of the results obtained may vary.
extensively between different countries and locales [31]. Figure 5.4 displays the result of using reverse geo-coding. The latitude/longitude pair has been converted into a street address.

![Image of Where Am I? app](image)

Figure 5-4: Current location along with the street address

We then use android text to speech (TTS) application programming interface (API) to translate the address into speech for the user [34].

### 5.1.3 Map Interface

The `MapView` class in android offers a perfect user interface option for presenting geographical data. A `MapView` can be used to create activities that feature an interactive map. `MapViews` support annotation using `Overlays` and by pinning views to geographical locations. `Overlay` is the class used to annotate maps. Using `Overlays` we can use a canvas to draw on top of a `MapView`. `MapViews` offer full programmatic control of the map display letting the user control the zoom level, location, and the various display modes. The interactive map can be seen in three different views which include the satellite, street, and traffic views.

In order to use a `MapView` in an application the developer first needs to obtain an API key from the Android developer web site. An API key (Application Programming Interface key) is a code generated by websites that allow users to access their API. API keys are used to track how the API is being used in order to prevent malicious use or abuse of the terms
of service [43]. Without an API key the MapView will not download the tiles used to display the map. To obtain a key one needs to specify the MD5 fingerprint of the certificate used to sign the application. A public key fingerprint is a short sequence of bytes used to authenticate or look up a longer public key [35]. Fingerprints are created by applying a cryptographic hash function such as MD5 to a public key. Generally, an application is signed using two certificates, a default debug certificate and a production certificate. If a developer uses Eclipse with the ADT plug-in to debug the applications, he or she will be signed with the default debug certificate. To view map tiles while debugging the user will need to obtain a Maps API key registered via the MD5 fingerprint of the debug certificate. Each computer that we use for development will have a different debug certificate and MD5 value. If we want to debug and develop map applications across multiple computers we will need to generate and use multiple API keys [31].

MapActivity has to be extended in order to use maps in an application. The layout for the new class then contains a MapView to display a Google Maps interface element. The Android maps library is not a standard Android package. As an optional API, it must be explicitly included in the application manifest before it can be used. It is provided within the Android SDK by Google and is available on most Android devices. Google Maps downloads the map tiles on demand as a result. It implicitly requires permission to use the Internet. The MapView class displays the Google map; and it includes several options for specifying how the map is displayed. By default the MapView will show the standard street map, as shown in Figure 5. In addition, you can choose to display a satellite view, Street View, and expected traffic, as shown in the Figure 5.5. We then use the MyLocationOverlay class to show the current location of the user on the map.
Specific map locations in the android mapping classes are represented by GeoPoint objects which contain latitude and longitude measured in micro degrees. Markers are then placed at these specific GeoPoints as our potential bus stops.

5.1.4 Google Maps for Mobile

The URLConnection abstract subclass defines methods for managing HTTP connections. We open a URL connection with the Google maps API through which we gain complete access to Google mobile maps. By doing this we can obtain the walking and driving directions to the bus stop of our choice. We can also get voice guided turn by turn walking and driving directions by using the Google Maps Navigation. Figure 5.6 displays the direction list obtained by using the Google Maps Navigation service in the application.
5.2 GIS Layer

Often, GIS’s are misconstrued as a mere geographic tool for producing maps. A more complete and comprehensive definition which characterizes the functions of GIS is as follows: “It stores geographic data, retrieves and combines this data to create new representations of geographic space. It also provides tools for special analysis and performs simulations to help users organize their work in many areas. These areas include public administration, transportation networks, military applications, and environmental information systems [36]”. The application layer invokes the GIS layer every time the application needs information regarding the GIS layer. The functions of the GIS layer are:
- To obtain the current location of the user when the GPS receiver is enabled by the application layer;
- To download the map tiles based on the current location of the user; and
- To fix the positions of the bus stops on the map tiles as requested by the application layer.

5.3 Database Layer

Mobile phone applications which implement a database ensure that the end users always have access to important information at any point of time in order to complete a job. The database layer in my work consists of information regarding the bus routes and their schedules which is easily accessed by the user. Information from the database layer is sent to the application layer as and when requested. The application provides real time information regarding the current location of the user and the route to any bus stop. However, the database created in this application is not dynamic in nature and hence does not provide real time information on the position of the buses and their arrival times. The dynamic database feature has not been implemented due to the unavailability of access to any of the public transportation system’s server. The very fact that we are able to pull information from a database created proves that the application is also capable of retrieving information from a back-end server which holds real time information of the state of the buses. By implementing a real time database in the application we can say that a mobile application which can facilitate the use of public transportation for the visually impaired can be developed.

The pre-requisites for maintaining the database that has been implemented are:
- Data regarding the total number of buses and their registration numbers;
- The specific route that each bus takes; and
- Schedule of all the buses.

The model of a portion of the database is shown in table 5.1 and table 5.2:

Table 5.1: Bus Routes

<table>
<thead>
<tr>
<th>Bus Number</th>
<th>Bus Route</th>
<th>Via</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bus Stop A to Bus Stop E</td>
<td>Bus Stop B, Bus Stop C, Bus Stop D</td>
</tr>
<tr>
<td>2</td>
<td>Bus Stop F to Bus Stop J</td>
<td>Bus Stop G &amp; Bus Stop H, Bus Stop I</td>
</tr>
</tbody>
</table>

Table 5.1 consists of the data regarding the bus numbers and their routes.

Table 5.2: Bus Schedules

<table>
<thead>
<tr>
<th>Bus Number</th>
<th>Bus Route</th>
<th>Arrival Times</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bus Stop A to Bus Stop E</td>
<td>Bus Stop A at 10 AM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bus Stop B at 10:10 AM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bus Stop C at 10:20 AM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bus Stop D at 10:30 AM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bus Stop E at 10:40 AM</td>
</tr>
</tbody>
</table>

Table 5.2 consists of data regarding the bus numbers and their arrival times at various bus stops. All the information from Tables 5.1 and 5.2 are displayed in the application on demand. A screen shot of the application displaying the information is shown in Figure 5.7.
The database that would be ideal for this application would be dynamic in nature. An example of one such application is OneBusAway [6]. The OneBusAway application pulls information from a backend server which is equipped with information regarding the arrival times of buses and their routes.

Figure 5-7: Bus routes and Schedules.
Chapter 6

Implementing a Multi-Modal Interface

Software applications and/or computing systems that combine multiple modes of input and output are said to be multimodal in nature. Multimodal interfaces [39] are user computer interfaces which make use of the various human mechanisms of perception. Such interfaces are characterized by the use of information that is encoded in some communication channel and involves a combination of vision, hearing, touch smell, taste and balance. They are a new class of interfaces that aim to recognize naturally occurring forms of human language and behavior, and which incorporate one or more recognition-based technologies (e.g. speech, pen, vision) [38]. For example, most of the current Smartphone’s incorporate the capabilities of a screen & keyboard interface, a touch interface, and a speech interface. The objective of multimodal interfaces is twofold. Firstly, to support and accommodate users’ perceptual and communicative capabilities. Secondly, to integrate computational skills of computers in the real world, by offering more natural ways of interaction to humans [37].

For users, multimodality represents a fast and efficient way to interact with a mobile device. Generally, multimodal applications are more compliant to the needs of different users in varying circumstances.
6.1 Barrier free interface

The communication between user and application in this research has been designed to support barrier free interfaces. A barrier free interface makes use of a combination of several modes of communication in a manner that it facilitates its use by one and all irrespective of their physical disabilities. Such interfaces are said to be multi-modal in nature. In barrier free interfaces the cognitive load associated with the task to be accomplished by means of the interface is either partially or in the ideal scenario totally absorbed by the system. As a result the communication between user and system is simplified. The various modes of interaction that will be presented in the android application are:

- Vision;
- Speech; and
- Tactile.

The subsections that follow will look into a few details of the three modes of interaction.

6.1.1 Vision

Some important terminology involved with manipulation of creating user interfaces in android are:

- Views: They are the base class for all visual interface elements (commonly known as controls or widgets). All User Interface (UI) controls, including the layout classes, are derived from View.

- ViewGroup: They are extensions of the View class that can contain multiple child Views. They extend the ViewGroup class to create compound controls made up of
interconnected child Views. The ViewGroup class is also extended to provide the layout managers that help the lay out controls within your Activities.

- **Activities:** They represent the window, or screen, being displayed. Activities are the Android equivalent of Forms. To display a user interface one needs to first assign a View (usually a layout) to an Activity [32].

The application consists of two activities. The first activity is the start up screen of the application which displays the current location of the user and the potential bus stops on an interactive map. The second activity is the screen which displays information about the buses. Details of the implementation of the two activities in the application have been explained in Chapter 5.

### 6.1.2 Speech

The text to speech (TTS) applications are a fairly new territory in the smart phone market. There are very few applications that can convert text of any kind into audio. Android has now introduced the Text To Speech engine. I have used the android TTS API to produce speech synthesis from within the application, allowing the mobile device to talk to users. Due to storage space constraints on some Android devices, the language packs are not always preinstalled on each device. Before using the TTS engine, it’s good practice to confirm whether the language packs have been installed. Once the availability of the voice data has been confirmed, one then needs to create and initialize a new text to speech instance. Note that one cannot use the new Text To Speech object until initialization is complete. When Text To Speech has been initialized we can use the `speak` method to synthesize voice using the default device audio output. The `speak` method lets us specify a parameter to either add the new voice output to the existing queue, or flush the queue and start speaking straight away. One can affect the way the voice output sounds using the
setPitch and setSpeechRate methods. Each of these methods accepts a float parameter that modifies the pitch and speed, respectively, of the voice output. More importantly, one can change the pronunciation of the voice output using the setLanguage method. This method takes a Locale value to specify the country and language of the text being spoken. This will affect the way the text is spoken to ensure the correct language and pronunciation models are used. When the application has finished speaking, one then uses the stop method to halt voice output and shutdown to free the TTS resources. I have used the android TTS API to covert the street address we obtain from reverse geo-coding into speech. This process is done in real time. The application provides the current location of the user as a speech output. I have also used the android TTS to convert the name of the buttons to speech when they have been selected by the user.

6.1.3 Tactile

The tactile or touch interface on most of the mobile devices uses a layer of capacitive material below a protective covering. The basic idea behind the working of a touch screen interface is that it involves the electrical properties of the human body. When a person touches a capacitive surface, the amount of charge it holds changes. Regardless of which method the screen uses to measure changes in electrical states, the user changes the electrical properties of the screen every time it is touched. The device records this change as data, and it uses mathematical algorithms to translate the data into an understanding of where the fingers are [42]. Most of the features available in the application can be accessed through the touch interface. For example, I have used the MapController method in android to pan and zoom a map view. The actual zoom level available for a specific location depends on the resolution of Google’s maps and imagery for that area. We can also use zoomIn and zoomOut features to change the zoom level.
6.2 Example Scenario

Consider a scenario where a user wants to travel from bus stop A to E. Let us also assume that the user is relatively new to the locality and hence does not know the current location, route to the nearest bus stop, and the schedule of the buses. Taking the above assumptions into consideration, let us now look at how my application will guide the user from bus stop A to E with ease.

STEP 1: The user launches the application by tapping on the “whereami” icon as shown in Figure 6.1. As soon as the application is initiated, the current location of the user is provided on the screen in text as well as through a voice output. The screen shot of the application displaying the user’s current location and the potential bus stops is shown in Figure 6.2. Figure 6.2 displays the initial interaction state of the application.

![Figure 6.1: Screen shot of the icon used to initiate the application](image)

The current location of the user is displayed on the interactive map as a green dot with a “user” named tag to it as shown in Figure 6.2.
Figure 6-2: Screenshot of the initial interaction state of the application

The address of any bus stop can be obtained by tapping on the bus icon as shown in Figure 6.3.
STEP 2: The user now might want to obtain information on the bus schedules. On tapping the “Schedules” button the user is guided to the next interaction state where all the information regarding the routes and status of the buses are available as shown in Figure 6.4. The Screen in Figure 6.4 has been divided into two scrollable parts. The top half displays all the bus numbers along with their specific routes. The bottom half displays the time tables of all the buses at their various bus stops. From the information in Figure 6.4 the user infers that bus number one has to be taken in order to reach bus stop E which is his/her destination. The user now has arrival information of the bus at the various bus stops.
Figure 6-4: Screen shot of the information on bus routes and schedules

STEP 3: The user now needs to obtain the walking or driving directions to the bus stop to embark his journey. On pressing the “back” button on the mobile device the user is steered back to the initial interaction state of the application as shown in Figure 6.5.
To proceed with the task of obtaining the address of bus stop A, the user is now expected to interact with the system by clicking the “Route to Bus Stop” button. The application responds by presenting the user with the Google maps for mobile service which has been integrated into the application through a URL connection as shown in Figure 6.6.
Figure 6-6: Screen shot of the Google maps route finder

The reason why I have used the Google maps service in the application is because of its popularity which has lead to familiarity with most of the mobile phone users. In order to proceed with the interaction, the user is expected to enter the address of the bus stop to obtain the walking or driving directions as shown in Figure 6.7. Bus stop addresses may be provided by the user either as a speech or by the touch interface.
The application will guide the user to the bus stop by rendering the directions through speech and the android display.

Figure 6-7: Screen shot of the walking directions
Chapter 7

Conclusion and Future Work

7.1 Conclusion

This research has introduced a novel approach in the field of LBS to facilitate the use of public transportation for people with or without visual impairments. The application has successfully integrated visual, speech and tactile modes of interaction which has resulted in creating a barrier free interface. A barrier free interface based application for the mobile devices which can possibly use real time bus information is presented in the document. The contribution of the present work to Public Transportation access for the visually impaired users is the following:

- A map interface to display the current location of the user and the potential bus stops;
- A speech output of the current location of the user;
- The visual availability of the addresses of the bus stops in response to user selection of the marker which is embedded in the map interface;
- Bus stop addresses may be provided by the user, either as speech or by the touch interface;
• A list of directions to a bus stop as response to the user’s choice from the current location using Google mobile maps;

• Both speech and visual turn by turn navigation from the user’s current location to the bus stop; and

• A database support that provides the user with information about the routes and schedules of the buses for all the routes.

### 7.2 Future Work

One of the important reasons to work in this area of research was to encourage future work in the same field so that applications that provide reliable real time information to the users can be developed. While this application was developed to aid transit riders with visual impairments to increase their level of independence, any user can benefit from its service. The various encouraging results obtained from the application indicate the following possible upgrades:

• To include real time information about the position of the buses and therefore predict their arrival times.

• The real time information of other modes of public transportation.

• The implementation of a notification system to inform the user where to disembark the bus.

• The implementation of a caching mechanism to cut down costs can be implemented so that the similar information need not be downloaded again.
References


15. The travel assistant device: Utilizing GPS-enabled mobile phones to aid transit riders with special needs. Sean J. Barbeau, Philip L. Winters, Nevine L. Georggi, Miguel A. Labrador, Rafael Perez.


22. AT&T Developer Program- Developing Applications for Android. s.l. : AT&T .


Appendix A

Source Code for the Android Application

A.1 User Interface Implementation

This section shows the source code of all the important components that make up the user interface.

A.1.1 Source code for the Implementation of map view in XML

```xml
<?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" />
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
        <Button
            android:id="@+id/displayMap"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Route to Bus Stop"
            android:layout_weight="1" />
    </LinearLayout>
</LinearLayout>
```
A.1.2 Source code for displaying the users current location on the map

```java
public class MyPositionOverlay extends Overlay {
    private final int mRadius = 5;
    Location location;

    public Location getLocation() {
        return location;
    }
    public void setLocation(Location location) {
        this.location = location;
    }

    @Override
    public boolean onTap(GeoPoint point, MapView mapView) {
        // Get the projection to convert to and from screen coordinates
        Projection projection = mapView.getProjection();
        // Return true if we handled this onTap()
        return true;
        //return false;
    }
}
```
public void draw(Canvas canvas, MapView mapView, boolean shadow) {
    Projection projection = mapView.getProjection();

    if (shadow == false) {
        // Get the current location
        Double latitude = location.getLatitude() * 1E6;
        Double longitude = location.getLongitude() * 1E6;
        GeoPoint geoPoint;
        geoPoint = new GeoPoint(latitude.intValue(), longitude.intValue());

        // 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 + mRadius);

        // Setup the paint
        Paint paint = new Paint();
        paint.setARGB(250, 127, 255, 112);
        paint.setAntiAlias(true);
        paint.setFakeBoldText(true);

        Paint backPaint = new Paint();
        backPaint.setARGB(175, 0, 0, 0);
        backPaint.setAntiAlias(true);

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

        // Draw the marker
        canvas.drawOval(oval, paint);
        canvas.drawRoundRect(backRect, 5, 5, backPaint);
        canvas.drawText("USER",
                        point.x + 2*mRadius, point.y,
                        paint);
    }
    super.draw(canvas, mapView, shadow);
}

A.2 Source code for initializing Android TTS (text to speech)

public void onInit(int version) {
    //talk !

    if (version == TextToSpeech.SUCCEED) {
        // Set preferred language to US english.
    }
int result = mTts.setLanguage(Locale.US);
if (result == TextToSpeech.LANG_MISSING_DATA ||
    result == TextToSpeech.LANG_NOT_SUPPORTED) {
} else {
    mTts.setLanguage(Locale.US);

    mTts.speak("Your current location is" + addressString,
        TextToSpeech.QUEUE_FLUSH, null);
}

@Override
public void onDestroy() {
    // Don't forget to shutdown!
    if (mTts != null) {
        mTts.stop();
        mTts.shutdown();
    }

    super.onDestroy();
}

A.3 Source code for using the Google Maps API

private boolean displayMap(int cellID, int lac) throws Exception {
    String urlString = "http://www.google.com/glm/mmap";

    //---open a connection to Google Maps API---
    URL url = new URL(urlString);
    URLConnection conn = url.openConnection();
    HttpURLConnection httpConn = (HttpURLConnection) conn;
    httpConn.setRequestMethod("POST");
    httpConn.setDoOutput(true);
    httpConn.setDoInput(true);
    httpConn.connect();

    //---write some custom data to Google Maps API---
    OutputStream outputStream = httpConn.getOutputStream();
    WriteData(outputStream, cellID, lac);

    //---get the response---
    InputStream inputStream = httpConn.getInputStream();
    DataInputStream dataInputStream = new DataInputStream(inputStream);
//---interpret the response obtained---
dataInputStream.readShort();
dataInputStream.readByte();
int code = dataInputStream.readInt();
if (code == 0) {
double lat = (double) dataInputStream.readInt() / 1000000D;
double lng = (double) dataInputStream.readInt() / 1000000D;
dataInputStream.readInt();
dataInputStream.readInt();
dataInputStream.readInt();
dataInputStream.readUTF();

//---display Google Maps---
String uriString = "geo:" + lat + "," + lng;
Intent intent = new Intent(
    android.content.Intent.ACTION_VIEW,
    Uri.parse(uriString));
startActivity(intent);
return true;
} else {
    return false;
}

private void WriteData(OutputStream out, int cellID, int lac)
throws IOException
{
    DataOutputStream dataOutputStream = new DataOutputStream(out);
dataOutputStream.writeShort(21);
dataOutputStream.writeLong(0);
dataOutputStream.writeUTF("en");
dataOutputStream.writeUTF("Android");
dataOutputStream.writeUTF("1.0");
dataOutputStream.writeUTF("Web");
dataOutputStream.writeByte(27);
dataOutputStream.writeInt(0);
dataOutputStream.writeInt(0);
dataOutputStream.writeInt(3);
dataOutputStream.writeInt(0);
dataOutputStream.writeInt(0);
dataOutputStream.writeInt(0);
dataOutputStream.writeInt(0);
dataOutputStream.writeInt(0);
dataOutputStream.writeFileInputStream(out);
dataOutputStream.writeInt(cellID);
dataOutputStream.writeInt(lac);
dataOutputStream.writeInt(0);
dataOutputStream.writeInt(0);
dataOutputStream.writeInt(0);
dataOutputStream.writeString("\n");
dataOutputStream.writeInt(0);
dataOutputStream.writeInt(0);
dataOutputStream.writeInt(0);
dataOutputStream.writeInt(0);
dataOutputStream.writeInt(0);
dataOutputStream.flush();
}
A.4 Source code for implementing the database

```java
public class DBAdapter {
    public static final String KEY_ROWID = "_id";
    public static final String KEY_BUS_NO = "busNo";
    public static final String KEY_START_BUS_STOP_NAME = "startstopname";
    public static final String KEY_STOP1_NAME = "stop1name";
    public static final String KEY_STOP2_NAME = "stop2name";
    public static final String KEY_STOP3_NAME = "stop3name";
    public static final String KEY_END_BUS_STOP_NAME = "endstopname";
    public static final String KEY_START_BUS_STOP_TIME = "startstoptime";
    public static final String KEY_STOP1_TIME = "stop1time";
    public static final String KEY_STOP2_TIME = "stop2time";
    public static final String KEY_STOP3_TIME = "stop3time";
    public static final String KEY_END_BUS_STOP_TIME = "endstoptime";

    private static final String TAG = "DBAdapter";
    //TODO changed books to books2
    private static final String DATABASE_NAME = "BusDatabase";
    private static final String DATABASE_TABLE = "RoutesAndSchedule";
    private static final int DATABASE_VERSION = 5;

    private static final String DATABASE_CREATE =
        "create table RoutesAndSchedule (_id integer primary key autoincrement, 
        + "busNo text not null, startstopname text not null,stop1name text not null,stop2name text not null,stop3name text not null,endstopname text not null,"
        + " startstoptime text not null,stop1time text not null,stop2time text not null,stop3time text not null,endstoptime text not null)";

    private final Context context;
    private DatabaseHelper DBHelper;
    private SQLiteDatabase db;

    public DBAdapter(Context ctx) {
        this.context = ctx;
        DBHelper = new DatabaseHelper(context);
    }

    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);
}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
{
    Log.w(TAG, "Upgrading database from version " + oldVersion
    + " to " + newVersion + ", which will destroy all old data");
    db.execSQL("DROP TABLE IF EXISTS RoutesAndSchedule");
onCreate(db);
}

// opens the database
public DBAdapter open() throws SQLException
{
    db = DBHelper.getWritableDatabase();
    return this;
}

// closes the database
public void close()
{
    DBHelper.close();
}

// insert a RoutesAndSched into the database
public long insertRoutesAndSched(String busNo,
    String startstopname, String stop1name, String stop2name, String stop3name,
    String endstopname, String startstoptime, String stop1time, String stop2time,
    String stop3time, String endstoptime)
{
    ContentValues initialValues = new ContentValues();
    initialValues.put(KEY_BUS_NO, busNo);
    initialValues.put(KEY_START_BUS_STOP_NAME, startstopname);
    initialValues.put(KEY_STOP1_NAME, stop1name);
    initialValues.put(KEY_STOP2_NAME, stop2name);
    initialValues.put(KEY_STOP3_NAME, stop3name);
    initialValues.put(KEY_END_BUS_STOP_NAME, endstopname);
    initialValues.put(KEY_START_BUS_STOP_TIME, startstoptime);
    initialValues.put(KEY_STOP1_TIME, stop1time);
    initialValues.put(KEY_STOP2_TIME, stop2time);
    initialValues.put(KEY_STOP3_TIME, stop3time);
    initialValues.put(KEY_END_BUS_STOP_TIME, endstoptime);
    return db.insert(DATABASE_TABLE, null, initialValues);
}

// deletes a particular RoutesAndSched

//---opens the database---

//---closes the database---

//---insert a RoutesAndSched into the database---

//---deletes a particular RoutesAndSched---
public boolean deleteRoutesAndSched(long rowId) {
    return db.delete(DATABASE_TABLE, KEY_ROWID + "=" + rowId, null) > 0;
}

//---retrieves all the RoutesAndScheds---
public Cursor getAllRoutesAndScheds() {
    return db.query(DATABASE_TABLE, new String[] {
        KEY_ROWID,
        KEY_BUS_NO,
        KEY_START_BUS_STOP_NAME,
        KEY_STOP1_NAME,
        KEY_STOP2_NAME,
        KEY_STOP3_NAME,
        KEY_END_BUS_STOP_NAME,
        KEY_START_BUS_STOP_TIME,
        KEY_STOP1_TIME,
        KEY_STOP2_TIME,
        KEY_STOP3_TIME,
        KEY_END_BUS_STOP_TIME
    },
    null,
    null,
    null,
    null,
    null);
}

//---retrieves a particular RoutesAndSched---
public Cursor getRoutesAndSched(long rowId) throws SQLException {
    Cursor mCursor =
    db.query(true, DATABASE_TABLE, new String[] {
        KEY_ROWID,
        KEY_BUS_NO,
        KEY_START_BUS_STOP_NAME,
        KEY_STOP1_NAME,
        KEY_STOP2_NAME,
        KEY_STOP3_NAME,
        KEY_END_BUS_STOP_NAME,
        KEY_START_BUS_STOP_TIME,
        KEY_STOP1_TIME,
        KEY_STOP2_TIME,
        KEY_STOP3_TIME,
        KEY_END_BUS_STOP_TIME
    },
    KEY_ROWID + "=" + rowId,
    null,
    null,
    null,
    null);

    if (mCursor != null) {
        mCursor.moveToFirst();
    }
}
return mCursor;

//---updates a RoutesAndSched---
public boolean updateRoutesAndSched(long rowId, String busNo,
    String startstopname, String stop1name, String stop2name, String stop3name,
    String startstoptime, String stop1time, String stop2time, String stop3time,
    String endstopname, String endstoptime)
{
    ContentValues args = new ContentValues();
    args.put(KEY_BUS_NO, busNo);
    args.put(KEY_START_BUS_STOP_NAME, startstopname);
    args.put(KEY_STOP1_NAME, stop1name);
    args.put(KEY_STOP2_NAME, stop1name);
    args.put(KEY_STOP3_NAME, stop1name);
    args.put(KEY_END_BUS_STOP_NAME, endstopname);
    args.put(KEY_START_BUS_STOP_TIME, startstoptime);
    args.put(KEY_STOP1_TIME, stop1time);
    args.put(KEY_STOP2_TIME, stop1time);
    args.put(KEY_STOP3_TIME, stop1time);
    args.put(KEY_END_BUS_STOP_TIME, endstoptime);
    return db.update(DATABASE_TABLE, args,
        KEY_ROWID + "=" + rowId, null) > 0;
}
Appendix B

Various Screenshots of Android Emulator and DDMS Perspective

B.1 Home Screen of the Android Emulator

Figure B-1: Screen shot of the android emulator.
B.2 Android Application running on the Emulator

Figure B- 2: Screen shot of the application running on the android emulator.
B.3 DDMS Perspective

Figure B-3: Screen shot of the DDMS tool
B.4 Emulator Controls to Provide Mock GPS Locations

![Emulator Controls](image)

Figure B-4: Screen shot of the emulator controls that provide mock GPS Locations.