**ABSTRACT**

**Objective:**

This project is aimed at providing APSRTC Local Bus information system in Hyderabad. This is a mobile based application that can be accessed in any android mobile phone. This application can be used to get the information about APSRTC city buses in Hyderabad. There are features like complete information of every bus and the rout information using Google maps.

**Existing System:**

* This System has no facilities of Communication through mobiles.
* Existing System has no facility providing the info about city buses in mobiles.
* Existing System has no facility for root map in mobile.
* In the existing system we need to communicate with people and get information otherwise we need to search in the internet.

**Proposed System:**

The development of this new system contains the following activities, which try to recover the problems from the previous system:

1. Providing complete information about city busses through mobiles.
2. Providing complete root map for desired stations using Google maps.
3. We can get the total information between two stations.
4. By using this application we need not to communicate with any one directly we can get the information about required busses in our mobile phones.

**Software Requirements for Designing:**

Operating System : Windows XP/above or Linux

Programming Language : Android Application Development

Web Applications : Google Android2.2

IDE/Workbench : Eclipse

Database : SQLlite

Sdk : Android Sdk 8 or above.

**Hardware Requirements for Designing:**

Processor : Pentium IV

Hard Disk : 40GB

RAM : 1 GB or more

**Software Requirements for Implementation:**

Operating System : Android 2 or above.

Mobile Phone : Any Android Mobile phone

**Hardware Requirements for Implementation:**

Processor : ARM 11or above

Internal Memory : 100 Mb or more

RAM : 126 Mb or more

Screen size : 480\*320

Pixel Density : 163ppi

**CHAPTER – I**

1. **INTRODUCTION**

Android is a software stack for mobile devices that includes an operating system, middleware and key application. Android is a software platform and operating system for mobile devices based on the Linux operating system and developed by Google and the Open Handset Alliance. It allows developer to write managed code in a java-like language that utilizes Google developed java libraries, but does not support programs developed in native code.

The unveiling of the android platform on 5 November 2007 was announced with the foundation of the Open handset Alliance, a consortium of 34 hardware, software and telecom companies devoted to advancing open standards for mobile devices. When release in 2008, most of the android platform will be made available under the apache free software and open –source license

.

**THE BIRTH OF ANDROID**

* 1. **Google Acquires Android Inc**.

In July 2005, Google acquired Android Inc., a small startup company based in Palo Alto, CA. Android's co-founders who went to work at Google included Andy Rubin {co-founder of Danger), Rich Miner {co-founder of Wildfire Communications, Inc), Nick Sears {once "\T at T-Mobile), and Chris White (one of the first engineers a! WebTV). At the time, little was known about the functions of Android Inc. other than they made software for mobile phones.

At Google, the team, led by Rubin, developed a Linux-based mobile device OS which they marketed to handset makers and carriers on the premise of providing a flexible, upgradeable system. It was reported that Google had already lined up a series of hardware component and software partners and signaled to carriers that it was open to various degrees of cooperation on their part.

**1.1.1. Open Handset Alliance Founded**

On 5 November 2007, the Open Handset Alliance, a consortium of several companies which include Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, Sprint Nextel and NVIDIA, was unveiled with the goal to develop open standards for mobile devices. Along with the formation of the Open Handset Alliance, the OHA also unveiled their first product. Android, an open source mobile device platform based on the Linux operating system.

**1 .1.2Hardware**

Google has unveiled at least three prototypes for Android, at the Mobile World Congress on February 12, 200B. One prototype at the ARM booth displayed several basic Google applications. A 'd-pad' control zooming of items in the dock with a relatively quick response.

A prototype at the Google IO conference on May 2B, 2QOS had a 52B MHz Qualcomm processor and a Synaptic capacitive touch screen, and used the UMTS cellular standard. It had 12BMB of RAM and 256MB of flash, showing that Android's memory requirements are reasonable. The demo was carried out using a 3.6 Mbit/s HSDPA connection.

* 1. **FEATURES**

**1.2 1 Application Framework**

It *is used* to write application for Android. Unlike after embedded mobile environments. Android applications are all equal, for instance, an applications which come with the phone are no different than those dial any developer writes. The framework is supported by numerous open source libraries such as openssl, SQLile and libc. It is also supported by the Android core libraries. Prom the point of security, the framework is based on UNIX file system; permissions that assure applications have only those abilities that mobile phone owner gave them al install time.

* + 1. **Dalvik Virtual Machine**

It is extremely low-memory based virtual machine, which was designed especially for Android to run on embedded systems and work well in low power situations. It is also tuned to the CPU attributes. The Dalvik VM creates a special file formal (DEX) that is created through build time post processing. Conversion between Java classes and .DEX format is done by included "'dx" tool.

**1.2.3 Integrated Browser**

Google made a right choice on choosing Web Kit as open source web browser. They added a two pass layout and frame flattening. Two pass layout loads a page without waiting for blocking elements, such as external CSS or external JavaScript and after a while renders again with all resources downloaded to the device. Frame flattening converts founded frames into single one and loads into the browser. These features increase speed and usability browsing the internet via mobile phone.

**1.2.4. Optimized Graphic**

As Android has *2D* graphics library and 3D graphics based on OpenGL ES 1.0, possibly we will see great applications like Google Earth and spectacular games like Second Life, which come on Linux version. At this moment, the shooting legendary 3D game Doom was presented using Android on the mobile phone.

12.5. **SQLite**

**Extremely small** (-500kb) **relational** database management system, which ***is* integrated** in Android. 11 is based on function **call and single file, where all definitions, tables** and data **are stored. This simple design is more** than **suitable** for a platform such as **Android.**

**1.2.6. Handset Layouts**

The platform is adaptable lo both larger, VGA, 2D graphics library, 3D graphics library based on OpenGL ES 1.0 specifications, traditional smart phone layouts. An underlying 2D graphics engine is also included. Surface Manager manages access to the display subsystem and seamlessly composites 2D and 3D graphic layers from multiple applications

**1.2.7. Data storage**

SQLite is used for structured data storage .SQLite is a powerful and lightweight relational database engine available to all applications.

**1.2.8. Connectivity**

Android supports a wide variety of connectivity technologies including GSM, CDMA, Bluetooth, EDGE, EVDO, 3G and Wi-Fi.

**1.2.9. Messaging**

SMS, MMS, and XMPP are available forms of messaging including threaded text messaging.

**1.2.10. Web Browser**

The web browser available in Android is based on the open-source Web Kit application framework. In includes Lib Web Core which is a modem web browser engine which powers both the Android browser and an embeddable web view.

**1.2.11. Java Virtual Machine**

Software written in Java can be compiled into Dalvik byte codes and executed in the Dalvik virtual machine, which is a specialized VM implementation designed for mobile device use, although not technically a standard Java Virtual Machine.

**1.2.12. Media Support**

Android will support advanced audio/video/still media formats such as MPEG-4, FL264.MP3. And AAC, AMR. JPEG. PNG, GIF.

**1.2.13. Additional Hard" are Support**

Android is fully capable of utilizing video '''still cameras, touch screens, GPS, compasses, accelerometers. And accelerated 3D graphics.

**1.2.14. Development Environment**

Includes a device emulator, tools for debugging, memory and performance profiling, a plug-in for the Eclipse IDE. There are a number of hardware dependent features, for instance, a huge media and connections support, GPS, improved support for Camera and simply GSM telephony. A great work was done for the developers to start work with Android using device emulator, tools for debugging and plug-in for Eclipse IDE.

**CHAPTER 2**

**2. DETAILED DESCRIPTION OF THE TOPIC**

**2.1. OPERATION**

**2.1.1. Android Runtime**

Android includes a set of core libraries drat provides most of the functionality available in the core libraries of the Java programming language Every Android application runs in its own process, with its own instance of the Dalvik virtual machine. Dalvik has been written so that a device can run multiple VMs efficiently.

The Dalvik VM executes files in the Dalvik Executable (.dex) formal which is optimized for minimal memory footprint. The VM is register-based, and runs classes compiled by a Java language compiler that have been transformed into the .dex format by the included "dx" tool. The Dalvik VM relies on the Linux kernel for underlying functionality such as threading and low-level memory management.

**2.12. Linux Kernel**

Android relies on Linux version Z6 for core system services such as security, memory management, process management, network slack, and driver model. The kernel also acts as an abstraction layer between the hardware and the rest of the software slack.

It helps to manage security, memory management, process management, network stack and other important issues. Therefore, the user should bring Linux in his mobile device as the main operating system and install all the drivers required in order to run it. Developers have full access to the same framework APIs used by [he core applications. The application architecture is designed to simplify the reuse of component; any application can publish its capabilities and any other application may (hen make use of those capabilities (subject to security constraints enforced by the framework). This same mechanism allows components to be replaced by the user. Underlying all applications is a set of services and systems.

**ARCHITECTURE**

The following diagram shows the major component of the Android operating system. Each section is described in more detail below.



**Linux Kernel**

Android Architecture is based on Linux 2.6 kernel. It helps id manage security, memory management, process management, network stack and other important issues. Therefore, (he user should bring Linux in his mobile device as the main operating system and install all the drivers required in order to run it. Android provides the support for the Qualcomm MSM7K chipset family. For instance, the current kernel bee supports Qualcomm MSM 7200A chipsets, but in the second half of 200S we should see mobile devices with stable version Qualcomm MSM 7200, which includes major features:

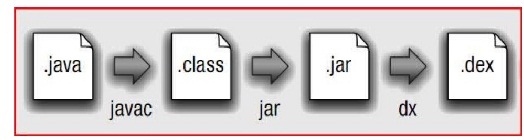
* WCDMA/HSUPA and ECPRS network support
* Bluetooth 1.2andWi-Fi support
* Digital audio support formp3 and other formats
* Support for Linux and other third-party operating systems
* Java hardware acceleration and support for Java applications
* camera up lo 6.0 megapixels
* GpsOne - solution for GPS
* And lots of other

**2.1.4. Libraries**

In the next level there are a set of native libraries written in C-'C—, which are responsible for stable performance of various components. For example, Surface Manager is responsible for composing different drawing surfaces on the mobile screen. It manages the access for different processes to compose 2Dand3D graphic layers. OpenGL ES and SGL make a core of graphic libraries and are used accordingly for 3D and 2D hardware acceleration. Moreover, it is possible to use 2D and 3D graphics in the same application in Android. The media framework was provided by Packet Video, one of the members of OHA. It gives libraries for a playback and recording support for all the major media and static image files. Free Type libraries are used to render all the bitmap and vector fonts. For data storage. Android uses SQLite. As mentioned before, it is extra light rational management system, which locates a single file for all operations related to database. Web Kit, the same browser used by Apples" Safari, was modified by Android in order to fit better in a small size screens.

**2.1.5. Android Runtime**

At the same level there is Android Runtime, where the main component Dalvik Virtual Machine is located. It was designed specifically for Android running in limited environment, where the limited battery, CPU, memory and data storage are the main issues. Android gives an integrated tool "dx", which converts generated.



byte code from .jar lo .dex file, after this byte code becomes much more efficient to run on the small processors.

Figure *2.2* : Conversion from Java lo .dex file

As the result, it is possible lo have multiple instances of Dalvik virtual machine running on the single device at the same time. The Core libraries are written in Java language and contains of the collection classes, the utilities, IO and other tools.

**2.1.6. Application Framework**

After that, there is Application Framework, written in Java language. His toolkit that all applications use, ones which come with mobile device like Contacts or SMS box. Or applications written by Google and any Android developer. It has several components.

The Activity Manager manages the life cycle of [he applications and provides a common navigation back stack for applications, which are running in different processes. The Package Manager keeps track of the applications, which are installed in the device. The Windows Manager is Java programming language abstraction on the top of lower level services that are provided by the Surface Manager.

The Telephony Manager contains of a set of API necessary for calling applications.

Content Providers was built for Android to share a data with other applications, for instance, the contacts of people in the address book can be used in other applications too. The Resource Manager is used to store localized strings, bitmaps, layout file descriptions and other external parts of the application. The View System generates a set of buttons and lists used in UJ. Other components like Notification manager is used to customize display alert and other functions.

**2.1.7. Application Layers**

` At the top of Android Architecture we have all the applications, which are used by the final user. By installing different applications, the user can turn his mobile phone into the unique, optimized and smart mobile phone. All applications are written

using the Java programming language..

**2.2. DEVELOPING APPLICATIONS**

**2.2.1. Application Building Blocks**

We can think of an Android application as a collection of components, of various kinds. These components are for the most pail quite loosely coupled, lo the degree where you can accurately describe them as a federation of components rasher than a single cohesive application.

Generally, these components all run in the same system process. It's possible (and quite common) to create multiple threads within that process, and it's also possible to create completely separate child processes if you need to. Such cases are uncommon. Because Android Cries very hard to make processes transparent to your code. Google provides three versions of SDK for Windows, for Mac OSX and one for Linux. The developer can use Android plug-in for Eclipse IDE or other IDEs such as InlelliJ. First step for Android developer is lo decomposing the application into the component. supported by the platform. The major building blocks are these:

* Activity
* Intent Receiver
* Service
* Content Provider

**2.2.1.1. Activity**

User interface component, which corresponds to one screen at time. It means that for the simple application like Address Book, the developer should have one activity for displaying contact, another activity component for displaying more detailed information of chosen name and etc.

**2.2.1.2. Intent Receiver**

Wakes up a predefined action through the external event. For example, for the application like Email Inbox, the developer should have intent receiver and

register code through XML to wake up an alarm notification, when the user receives email.

**2.2.13. Service**

A (ask, which *is done* in the background. Is means (hat (he user can start an application from ihe activity window and keep the service work, while browsing other applications. For instance, he can browse Google Maps application while holding a call or listening music while browsing other applications.

**2.2.1.4. Content Provider**

A component, which allows sharing some of the data with other processes and applications. It is the best way to communicate the applications between each other. Android will ship with a set of core applications including an email client, SMS program, calendar, maps, browser, contacts, and others. All applications

are written using the Java programming language.

**2.2.2. AndroidManifest.xml**

The AndroidManifest.xml file is the control file that tells the system what Ho do with all the top-level components (specifically activities, services, intent receivers, and content providers described below) you've created. For instance, this is the "glue" that actually specifies which Intents your Activities receive.

A developer should predefine and list all components, which he wants to use in the specific AndroidManifest.xml fit. It is a required file for all the applications and is located in the root folder. It is possible to specify all global values for the package, all the components and its classes used, intent fillers, which describe where and when the certain activity should start, permissions and instrumentation like security control and t sting.

Here is an example of AndroidManifest.xml file:

1. <?xml version="1.0" encoding="utf-8"?>
2. <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3. package="com.safeways"
4. android:versionCode="1"
5. android:versionName="1.0" >
6. <uses-sdk android:minSdkVersion="15" />
7. <application
8. android:icon="@drawable/icon"
9. android:label="welcome" >
10. <activity
    1. android:name=".Sample1Activity"
    2. android:label="@string/app\_name" >
    3. <intent-filter>
    4. <action android:name="android.intent.action.MAIN" />
    5. <category android:name="android.intent.category.LAUNCHER" />
    6. </intent-filter>
11. </activity>
12. </application>
13. </manifest>

The line *2 is* a namespace declaration, which makes a standard Android attributes available for that application. In the line 4 there is a single <application> element, where the developer specifies all application level component and its properties used by the package. Activity class in the line 5 represents the initial

**2.2.3. Application Lifecycle**

In Android, every application runs in its own process, which gives better performance in security, protected memory and other benefits. Therefore, Android is responsible Ho run and shut down correctly these processes when it is needed.

It is important that application developers understand how different application components (in particular Activity, Service, and Broadcast Receiver) impact the lifetime of the application's process. Not using these components correctly can result in the system killing the application's process while it is doing important work.

To determine which processes should be killed when low on memory. Android places each process into an "importance hierarchy" based on the components running in them and the state of those component. These process types are (in order of importance).

1. A foreground process is one that is required for what the user is currently doing. Various application components can cause its containing process to be considered foreground in different ways. A process is considered to be in the foreground if any of the following conditions hold:

* It is running an Activity at the top of the screen that the user is infracting with (its onResume() method has been called),
* ii It has a BroadcastReceiver that is currently miming (its BroadcaslReceiver.onReceiveO method is executing).
* It *has a* Service that *is* currently executing code in one of *its* callbacks (Service.onCreateQ, Service.onStartQ, or Service.onDesIroyQ). There will only ever be a few such processes in the system, and these will only be killed as a *last* leant if memory *is so* low mat not even these processes can continue to run. Generally, at this point, the device *has* reached a memory paging slate, so this action *is r*equired in order to keep the user interface responsive.

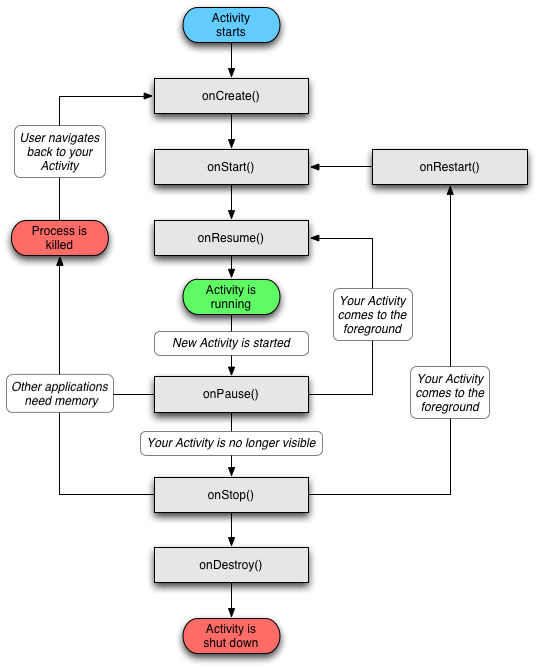
2. A visible process *is* one holding an Activity [hat *is* visible to the user on­screen but not in the foreground {its onPauscQ method has been called). This may occur, for example, if the foreground Activity is displayed as a dialog that allows the previous Activity to be seen behind it. Such a process is considered extremely important and will not be killed unless doing so is required to keep all foreground processes running.

3. A service process is one holding a Service that has been started with the start Service() method. Though these processes are not directly visible to the user, they are generally doing things that the user cares about {such as background mp3 playback or background network data upload or download), so the system will always keep such processes running unless there is not enough memory to retain all foreground and visible process.

4. A background process *is one* holding an Activity that *is not* currently visible

to the *user* {its onStop() method has been called). These processes have no direct impact on the user experience. Provided they implement their Activity life-cycle correctly (see Activity for more details), the system can kill such processes at any time to reclaim memory for one of the three previous processes types. Usually there are many of these processes miming, so they are kept in an LRU list to ensure the process [hat was most recently seen by the user is the last to be killed when running low on memory.

5. An empty process is one that doesn't hold any active application components. The only reason to keep such a process around is as a cache to improve startup time the next lime a component of its application needs to run. As such, the system will often kill these processes in order to balance overall system resources between these empty cached processes and the underlying kernel caches.



In the following example we will display a process flow from the Android System point of view *to* get a clear idea how the application behave. Let assume the

Possible 5cenario: A User talks to his friend via mobile phone and he is asked to browse the citymetro (a talk is hold for a moment), find the root of particular station via Maps back to his friend and resume .In this situation, there are 4 different applications and 4 different processes running, but from the user point of view none of them are important, as Android manages CPU work and memory usage by itself. It means the user can travel through the applications forward and back without thinking about how much memory is left or which processes are run at the time-Firstly, as the user is talking to his friend, a specific Talk application is opened, which contains the activity manager. In the following slack we can see two processes running, the main system process and Talk application process. Moreover, before going to

**Al this** point, as a user holds a **talk** and opens citymetro application**, the** system creates a **new process** and new citymetro **activity is launched in it.** Again, the staff of **last activity is** saved (W):

**After that, the** user browses **the city metro,** finds **his desired place. He does not close** city metro, instead he select a maps option to find the desired place in maps.

**Al this point, the** user finds the station and he request to open the maps**. The last state is** seeing. Now assume [hat the **mobile phone is** out of **the** memory and **there is** no room **to create a new process for maps application. Therefore,** Android **looks to kill a process.**

It cannot destroy search process, as it was used **previously and could** be reused again, so it kills not useful anymore and locates new map process instead:

The user opens maps application and search for a location .back to the Talk application and to resume a talk to his friend. Because of the previously saved states, this work is done last and easily. In this example. Maps application is popped out and the user sees a previous search application:

**2.2.4. Application Framework**

Developers have full access to the same framework APIs used by the core applications. The application architecture is designed to simplify the reuse of components: any application can publish its capabilities and any other application may then make use of those capabilities (subject lo security constraints enforced by the framework). This same mechanism allows components to be replaced by the user.

Underlying all applications is a set of services and sys terns, including:

1. A rich and extensible set of Views *that* can be used to build an application, including lists, grids, text boxes, buttons, and even an embeddable web browser

2. Content Providers dial enable applications lo access data from other applications (such as Contacts), or to share their own data

3. A Resource Manager, providing access to non-code resources such as localized strings, graphics, and layout files

4. A Notification Manager that enables all applications lo display custom alerts in the status bar

5. An Activity Manager that manages the life cycle of applications and provides a common navigation back stack.

**2.2.5. Library**

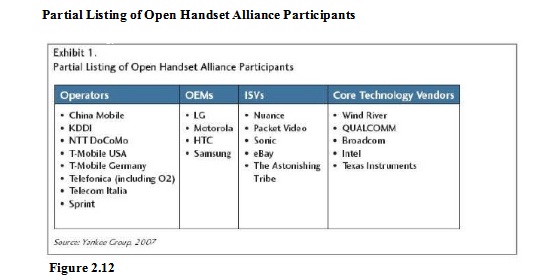
* Android includes a set of **c, c++** libraries used by various components of me Android system. These capabilities are exposed lo developers through me Android application framework. Some of the core libraries are listed Below:
* System C library - a BSD-derived implementation of the standard C system library (libc), tuned for embedded Linux-based devices
* Media Libraries - based on Packet video's open Core: the libraries
* Support playback and recording of many popular audio and video formals.
* Surface Manager - manages access Ha the display subsystem and seamlessly composites 2D and 3D graphic layers from multiple applications.
* LibWebCore - a modem web browser engine which powers both the Android browser and an embeddable web view
* SGL - the underlying 2D graphics engine.
* 3D libraries - an implementation based on openGL ES 1.0 APIs; the libraries use either hardware 3D acceleration (where available) or the included, highly optimized 3D software rasterizer.
* Free Type - bitmap and vector font rendering
* SQLite - a powerful and lightweight relational database engine available fi> all applications

**2.3. SOFTWARE DEVELOPMENT**

The feedback on developing applications for the Android platform has been mixed. Issues cited include bugs, lack of documentation, inadequate QA .The first publicly available application was the Snake game.

**Software Development kit**

It includes development and debugging tools, a set of libraries, a device emulator, documentation, sample projects, tutorials, and FAQs. Requirements also include Java Development Kit, Apache Ant, and Python *2.2* or later. The only officially supported integrated development environment (IDE) is Eclipse *32* or later, through the Android Development Tools Plug-in, but programmers can use command line tools to create, build and debug Android applications.



**2.4. SECURITY ISSUES**

Android mobile phone platform is going to be more secure than Apple's iPhone or any other device in the long run. There are several solutions now a day’s to protect Google

Phone from various attacks. One of them is security vendor McAfee, a member of Linux Mobile (LiMo) Foundation. This foundation joins particular companies to develop an open mobile-device software platform. Many of the companies listed in the LiMo foundation have also become members of the Open Handset Alliance (OHA).

As a result, Linux secure coding practice should successfully be built into the Android

Development process. However, open platform has its own disadvantages, such as source code vulnerability for black-hat hackers. In parallel with great opportunities for mobile application developers, there is an expectation for exploitation and harm. Stealthy Trojans hidden in animated images, particular viruses passed from friend to friend, used for spying and identity theft, all these threats will be active for a long run.

Another solution for such attacks is SMobile Systems mobile package. Security Shield -an integrated application that includes anti-virus, anti-spam, firewall and other mobile

Protection is up and ready to run on the Android operating system. Currently, the main

Problem is availability for viruses to pose as an application and do things like dial phone

Numbers send text messages or multi-media messages or make connections to the Internet during normal device use. It is possible for somebody louse the GPS feature to Track a person's location without their knowledge. Hence SMobile Systems is ready to notify and block these secure alerts. But the truth is that it is not possible to secure your mobile device or personal computer completely, as it connects to die internet. And neither the Android phone nor other devices will prove to be the exception.

**2.5. COMPARISON**

**2.5.1. Speculations with Cellular Carriers**

Google Android enters a tangled mess of cellular carrier world. As a new player in the mobile market. Android brings an open platform with the new rules. On the one hand there is OHA with major companies and carries, such as T-Mobile and Sprint. On the one hand, there are two largest cellular carries AT&T and Verizon Wireless in United Slats, which have a vested interest in operating systems of their own. It is predictable, dial Sprint or T-Mobile will be first earners providing devices with Google Android. This ensures equal development time for the networks, GSM side and CDMA. But the main problem, which laces all the cellular carriers around the world, is the availability to download and use free applications that could block almost every communications product they sell. A user does not need to pay for GPS mapping service anymore. He can simply download a free one that taps into Google Maps.

In fact. Why pay for cellular minutes at all when a user can download Skype, GTalk or other client and just use his data plan? OS's such as Android threaten carriers with a loss of control over the applications on the phones on their network and they may find themselves becoming nothing more than wireless Internet service providers, forced to complete on price and band width.

Another aspect is hardware cost Google Android owns 10 percent of the local cost of a phone, which combined with falling hardware prices could eventually result a free unlocked handset market. In conclusion, Google has a better start in this race than any company had before to bring new rules to the mobile market with all carriers, mobile devices and its customers.

**2.5.2. Manufacturers War**

Presently, Google main competitors like Nokia, Microsoft and Apple do not see Google Android as a serious rival or real to their business strategies. However, the cement situation is not so unsophisticated. There is a huge flunky in the companies, which are not in the list of OHA.

Par instance, Nokia, which is the largest handset manufacturer in the world, nowadays owning some 39% market share, was one of the companies snubbed on the invitation list to the34-parly Open Handset Alliance that is growing daily.

In contrast, Nokia is buying companies and dumping cash into development, while Google is releasing an open platform hoping the applications and services will build themselves with the help of a strong develop community, development contests and large alliance of grand companies.

Despite of this, Nokia is ready to come back whatever Google has to throw with Google Android in 2008. Another company Apple has already stroked the market with iPhone and its closed operating system. Accordingly, iPhone in the US remains loyal to AT&T mobile carrier for five years.

That is plenty of time for Google to conquer the market with open Android-Obvious advantage of Android is cost: while iPhone is priced at a weighty $400, Google says it hopes to reach a more mainstream market by pricing Android-powered devices at around $200. Microsoft, selling 21 million copies of Windows Mobile software, stays calm at this point waiting for some particular results from Google Android.

This nice and healthy competition is just what the mobile industry needs at the moment, at least for the consumers. The wars being waged between Google and the field will only create better, cheaper handsets and more advanced applications.

**2.5.3. Market Research**

A new generation of mobile device users is coming in the next decade. These users

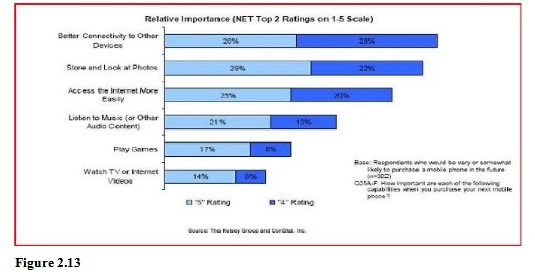
are going to explore the mobile internal afresh with its new features, compatible mobile phones, new services and applications.

This is a huge leap for mobile advertisement business, where revenue could rise 8 times more by 2012. Google Android is going to present new solutions through the fast search engine, open source applications and other services.

The Kelsey Group, which works with public opinion polls and statistics, published

the results released October 11 2007 .which say, that one hundred out of 500, or 20 percent of people would be interested in purchasing a Google phone. Despite the fact, that Google Android is in alpha version and it is unknown for the customers and mobile market, the results look promising.

The diagram below shows the study, which was conducted in September 2007 via an online 30-question survey of 500 U.S. mobile phone users aged IB and older. People do not find a good Internet experience in their phones today, so they are more interested in gravitating toward an Internal or technology company telephone



because they think connectivity between devices and to the Internet is going to be much better on those phones. They use Google search, Gmail, Google Maps. Picasa albums and other popular services on their computers, and this is what they expect to have in their mobile devices in die close future.

**2.5.4. Mobile Ads**

Jaiku - an activity stream and sharing service that works from the Web and mobile

phones was bought by Google as important investment into the mobile advertisement. People wondered why Google preferred the micro-blogging service to Twitter, which is much more popular nowadays.

The answer lies in Jaiku's unique ability to combine micro-blogging with user's location. An integral part of the service is a Jaiku client application for Symbian S60 platform mobile phones, which should come lo Android platform as well. The client uses location APIs within device to get the handset and the users" location based on nearby cellular network towers.

Though the location is not very precise, the mobile phone is able to broadcast it

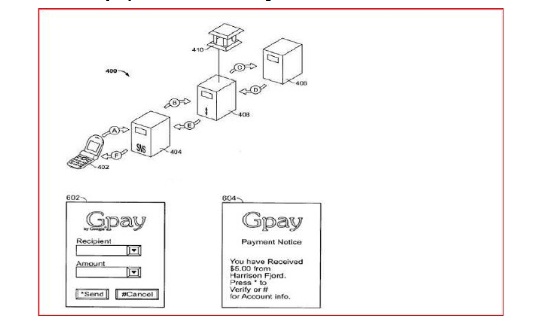
automatically. At dial point the text can be connected lo users^ location and create a

list of preferences for each place the user frequently visits.

Using such a technology, it is simple id track down a user via phone's IP address, whenever he comes into McDonald or is silting in the airport. Google is not a million miles away from being able to push advanced advertising to individuals based on their profile, their location and their availability. They already offer regional and local targeting for ads for desktop users, but this could be much more useful for a mobile phone. And if the ads are Truly relevant, interesting and unobtrusive, people might actually start lo like them.

**2.5.5. Mobile Services**

Adding to its fast growing suite of mobile applications and services, Google has applied for a patent for a mobile payments service that would allow users to make payments at retail shops using their mobile phones. The Text Message payment patent describes a system where Google offers mobile focused payments called GPay. This describes a system where a SMS message would be sent containing a payment amount and other information. That payment amount would then be validated, debited from the user's account, and communicated from server to server. Payment confirmation that had been received would also simultaneously be sent to the relevant party, as illustrated in the diagram below:



**Figure 2.14: Figure describe? Google's mobile focused payment? Called GPay**

Described as "a computer-implemented method of effectuating an electronic on-line payment," the system mentioned in the patent application is similar to existing mobile payment services.

These services like mobile version of PayPal have been available for some time but have had little success bursting with merchants and with customers. The main difference between existing mobile payment systems and GPay is, of course, that GPay is created by Google and will be easily adopted by Android Platform. The more issues regarding Gpay are yet to be released.

**2.5 .6. What makes Android special?**

There are already many mobile platforms on the market today, including Symbian, iPhone, Windows Mobile. BlackBerry, Java Mobile Edition. Linux Mobile (LiMo), and more.

While some of its features have appeared before. Android is the first environment that combines:

* A truly open, free development platform based on Linux and open source. Handset makers like it because they can use and customize the platform without paying a royalty. Developers like it because they know that the platform '"'has legs" and is not locked into any one vendor that may go under or be acquired.
* A component-based architecture inspired by Internet mash-ups. Parts of one application can be used in another in ways not originally envisioned by the developer. You can even replace built-in Components with your own improved versions. This will unleash a new round of creativity in the mobile space.
* Tons of built-in services out of the box. Location based services use GPS or cell tower triangulation to let you customize the user experience depending on where they are. A full-powered SQL database lets you harness the power of local storage for occasionally connected computing and synchronization. Browser and Map views can be embedded directly in your applications. All these built-in capabilities help lo raise the baron functionality while lowering your development costs.
* Automatic management of the application life cycle. Programs are isolated from each other by multiple layers of security, which will provide a Level of system stability not seen before in smart phones. The end user will no longer have to worry about that applications are active, or close some programs so that others can run. Android is optimized for low-power, low-memory devices in a fundamental way that no previous platform has attempted.
* High quality graphics and sound. Smooth, anti-aliased 2D vector graphics and animation inspired by Flash is melded with 3D.Accelerated OpenGL graphics to enable new kinds of games and business applications. Codecs for the most common industry standard audio and video formats are built right in, including H.264 (AVC), MP33 and AAC.
* Portability across a wide ranges of current and future hardware All your programs are written in Java and executed by Android's Dalvik virtual machine so your code will be portable across ARM.X.S6, and other architectures. Support for a variety of input methods is included such as keyboard, touch, tilt, camera, voice, and trackball. User interlaces can be customized for any screen resolution and orientation .Android is a fresh take on the way mobile applications

**2.5.7. Android and Java ME**

Java Platform, Micro Edition or Java ME (previously known as Java 2 Platform, Micro Edition or J2ME) is a specification of a subset of the Java platform aimed at providing a certified collection of Java APIs for the development of software for small, resource-constrained devices. Though, do not confuse it with Google Android, even mere are some similarities:

* Eclipse plug-ins for J2ME and Android look very similar and interface very well with their respective SDKs:
* Both J2ME and Android seem to share the same core Java APJs, such as java.util and java.net. But their APJs for graphics, UIs, etc. are very dissimilar and philosophies for developing applications are very different;
* Android seems to be more tightly integrated (up to even the OS services provided and how they interact with the APIs) while J2ME is far manufactures.

A slower application development and performance - these are the main disadvantages Java's J2ME have for today. J2ME apps are second-rate citizens in the phones. They do not have an access lo most of the low-level features, like call API, external connectivity (USB) and other. There is no way to replace or extend built-in phone apps like contact, calendar and calls.

For instance, J2ME applications in Nokia devices with S50 work great for standard tasks. But more advanced users find difficulties handling Wi-Fi access points with S60. because APIs simply do not seem to be exposed to J2ME.A user may find difficulties synchronizing Google Calendar with his device - nobody seems to have been able to figure out how to make the J2ME calendar interlaces work correctly on S60. There are lots of problems with Java applications on 550, even though S60 probably has one of the best Java implementations.

Android fills a void in Java Mobile applications by providing AFJ lo build richer applications - more useful for Smart Phones which contain the ability to provide these types of functionalities. If J2ME filled every void. Android as an APJ wouldn't be needed (though Android as an OS could still fill avoid).

Google has written its own virtual machine for Android most likely as a way to get around licensing issues with Sun. However, Android does not include a complete and compliant Java slack (neither JME nor JSE); only a subset and therefore it is technically not the Java platform, it just looks a lot like it

.

**2.5 8. Openness of the Platform**

The open source school of thought implies that differentiation and competitive advantage come from innovation on top of the underlying platform rather than the platform itself. The robustness and scalability of the platform is secured by the community's stewardship, and open access to a central repository of updated code. Beyond this, a strong third -party development environment and software development kit (SDK) are critical lo attracting innovation.

"Open" is an invariably subjective term. Symbian and Microsoft can claim a degree of openness for their mobile platforms, for example, but ultimate control of API access and source code remains with a single entity. On the contrary, as Google has pointed out, there's nothing keeping any of the alliance members from using Android id build a Yahoo Go phone.

Motorola has had some success delivering high-volume Linux-based devices such as the Ming and RAZR II to market. But mobile Linux initiatives have failed lo scale on the basis of attractiveness lo third-party developers; it's been supply-push with the development focus in Java ME or other application framework components.

Importantly, Android includes almost the entirely of the applications-related software stack, less key technical pieces such as telephony protocol stacks, which are left to silicon vendors. Android bundles critical components such as a Linux kernel from Wind River, various optimized graphics engines, codec, and notification software, a '"'clean room" JVM implementation, and the KHTML open source browser. The latter forms the basis of Apple's Safer! And Nokia’s S60 offerings.

**2.5 9. Advantages**

* Open - Android allows you to access core mobile device functionality through standard API calls.
* All applications are equal - Android does not differentiate between the phone's basic and third-party applications — even the dialer or home screen can be replaced.
* Breaking down boundaries - Combine information from the web with data on the phone -- such as contacts or geographic location -- to create new user experiences.
* Fast and easy development - The SDK contains what you need to build and run Android applications, including a [rue device emulator and advanced debugging tools.

**2.5 .10. Disadvantages**

* Security - Making source code available to everyone inevitably invites the attention of black hat hackers.
* Open Source - A disadvantage of open-source development is that anyone can scrutinize the source code to find vulnerabilities and write exploits.
* Login - Platform doesn't run on an encrypted file system and has a vulnerable log-in.
* Incompetence – Google’s dependence on hardware and carrier partners puts the final product *out* of their control.

**Database Tables**

**Search:**

The search table consists of all the stages information. Like what are the root busses are going on a particular station all those information will be stored in the Search table.

**Stages:**

The stages table consist of all the stages information i.e the user selected root information like the root starts from which station to which station and the all intermediate stations in between the root . all the information will be stored in the Stages table.

**Search Table:**

**Create Table “search” (“station” TEXT, “Stages” VARCHAR);**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Column id** | **Name** | **Type** | **Not Null** | **Default Value** | **Primary Key** |
| **0** | **Station** | **Text** | **1** | **O** | **O** |
| **1** | **Stages** | **Varchar** | **1** | **0** | **0** |

**Stages Table:**

**CREATE TABLE "stages" ("rootno" VARCHAR, "wtw" TEXT, "s1" TEXT, "s2" TEXT, "s3 TEXT, "s4" TEXT, "s5" TEXT, "s6" TEXT, "s7" TEXT, "s8" TEXT, "s9" TEXT, "s10" TEXT, "s11" TEXT, "s12" TEXT, "s13" TEXT, "s14" TEXT, "s15" TEXT, "s16" TEXT, "s17" TEXT, "s18" TEXT, "s19" TEXT, "s20" TEXT)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Column id** | **Name** | **Type** | **Not Null** | **Default Value** | **Primary Key** |
| **0** | **Rootno** | **Varchar** | **1** | **0** | **0** |
| **1** | **Wtw** | **Text** | **1** | **0** | **0** |
| **2** | **S1** | **Text** | **1** | **0** | **0** |
| **3** | **S2** | **Text** | **1** | **0** | **0** |
| **4** | **S3** | **Text** | **1** | **0** | **0** |
| **5** | **S4** | **Text** | **1** | **0** | **0** |
| **6** | **S5** | **Text** | **1** | **0** | **0** |
| **7** | **S6** | **Text** | **1** | **0** | **0** |
| **8** | **S7** | **Text** | **1** | **0** | **0** |
| **9** | **S8** | **Text** | **1** | **0** | **0** |
| **10** | **S9** | **Text** | **1** | **0** | **0** |
| **11** | **S10** | **Text** | **1** | **0** | **0** |
| **12** | **S11** | **Text** | **1** | **0** | **0** |
| **13** | **S12** | **Text** | **1** | **0** | **0** |
| **14** | **S13** | **Text** | **1** | **0** | **0** |
| **15** | **S14** | **Text** | **1** | **0** | **0** |
| **16** | **S15** | **Text** | **1** | **0** | **0** |
| **17** | **S16** | **Text** | **1** | **0** | **0** |
| **18** | **S17** | **Text** | **1** | **0** | **0** |
| **19** | **S18** | **Text** | **1** | **0** | **0** |
| **20** | **S19** | **Text** | **1** | **0** | **0** |
| **21** | **S20** | **Text** | **1** | **0** | **0** |

**Design Document**

* + The entire system is projected with a physical diagram which specifics the actual storage parameters that are physically necessary for any database to be stored on to the disk. The overall systems existential idea is derived from this diagram.
  + The relation upon the system is structure through a conceptual ER-Diagram, which not only specifics the existential entities but also the standard relations through which the system exists and the cardinalities that are necessary for the system state to continue.
  + The content level DFD is provided to have an idea of the functional inputs and outputs that are achieved through the system. The system depicts the input and output standards at the high level of the systems existence.

**Data Flow Diagrams**

* + This Diagram server two purpose.

⮲ Provides an indication of how date is transformed as it moves through the system.

⮲ Disputes the functions and sub functions that transforms the dataflow.

* + The Data flow diagram provides additional information that is used during the analysis of the information domain, and server as a basis for the modeling of functions.
  + The description of each function presented in the DFD is contained is a process specifications called as PSPEC

**ER-Diagrams**

* + The entity Relationship Diagram (ERD) depicts the relationship between the data objects. The ERD is the notation that is used to conduct the date modeling activity the attributes of each data object noted is the ERD can be described resign a data object descriptions.
  + The set of primary components that are identified by the ERD are

◆ Data object ◆ Relationships

◆ Attributes ◆ Various types of indicators.

* + The primary purpose of the ERD is to represent data objects and their relationships.

**Unified Modeling Language Diagrams**

* + The unified modeling language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules.
  + A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows.
  + User Model View
    1. This view represents the system from the users perspective.
    2. The analysis representation describes a usage scenario from the end-users perspective.

**Structural model view**

◆ In this model the data and functionality are arrived from inside the system.

◆ This model view models the static structures.

**Behavioral Model View**

◆ It represents the dynamic of behavioral as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

**Implementation Model View**

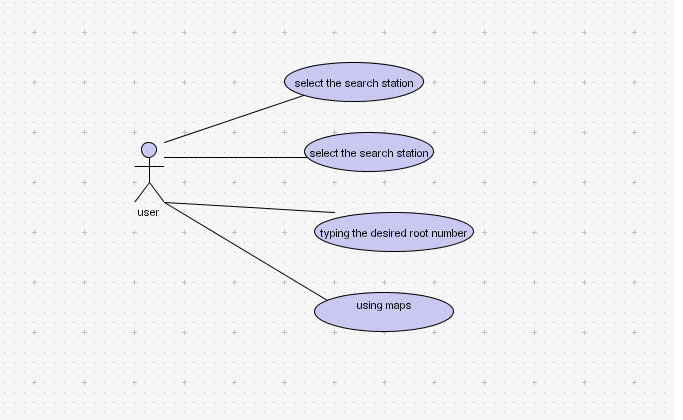
* + In this the structural and behavioral as parts of the system are represented as they are to be built.

**Environmental Model View**

In this the structural and behavioral aspects of the environment in which the system is to be implemented are represented.

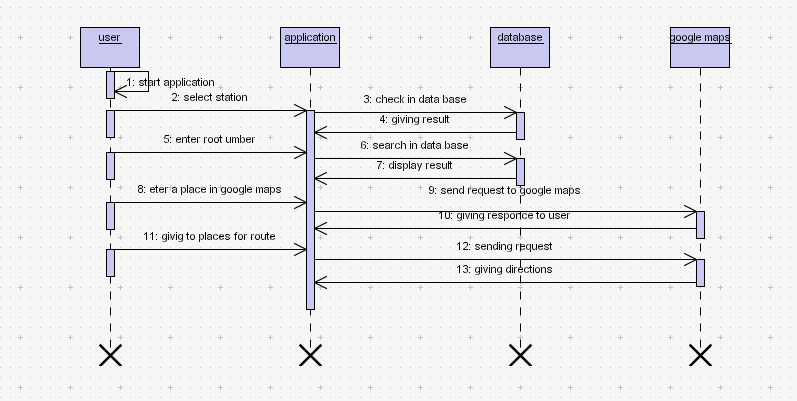
**UML Diagrams**

**Use case Diagram:**

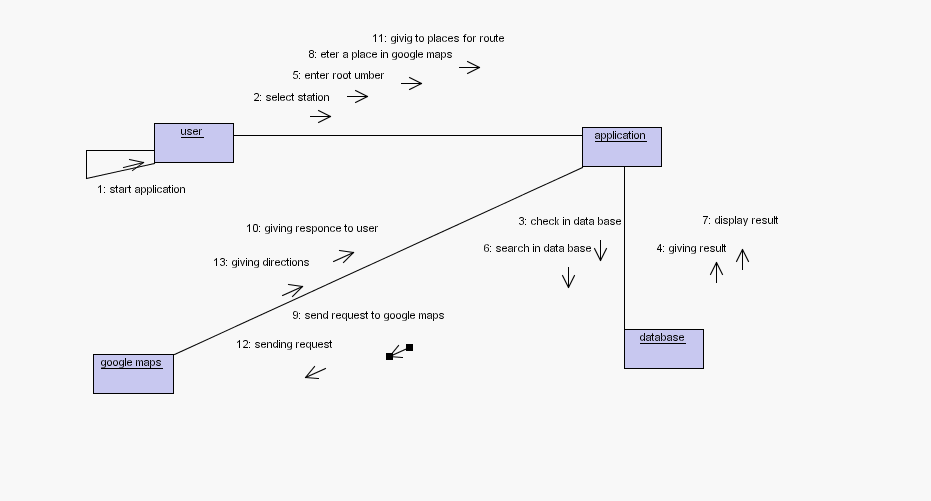


**Class Diagram:**

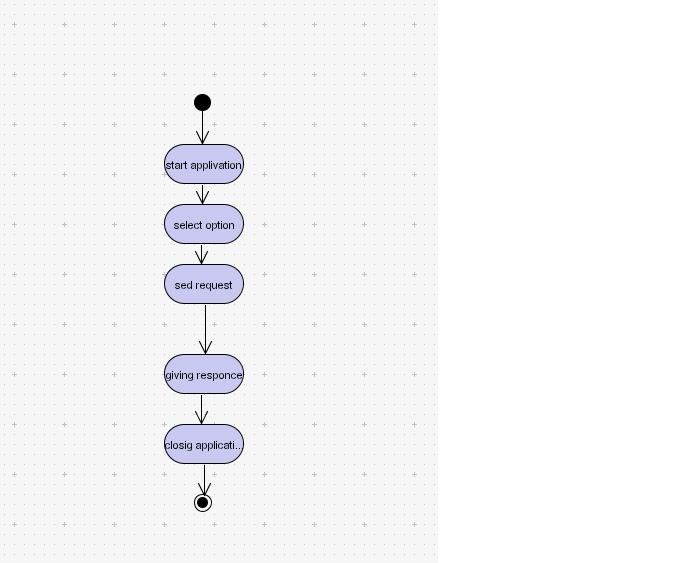
**Sequence Diagram:**

****

**Collaboration Diagram:**

****

**Start chart Diagram:**

****

**Testing**

Testing is the process of detecting errors. Testing performs a very critical role for quality assurance and for ensuring the reliability of software. The results of testing are used later on during maintenance also.

Psychology of Testing

The aim of testing is often to demonstrate that a program works by showing that it has no errors. The basic purpose of testing phase is to detect the errors that may be present in the program. Hence one should not start testing with the intent of showing that a program works, but the intent should be to show that a program doesn’t work. Testing is the process of executing a program with the intent of finding errors.

Testing Objectives

The main objective of testing is to uncover a host of errors, systematically and with minimum effort and time. Stating formally, we can say,

* Testing is a process of executing a program with the intent of finding an error.
* A successful test is one that uncovers an as yet undiscovered error.
* A good test case is one that has a high probability of finding error, if it exists.
* The tests are inadequate to detect possibly present errors.
* The software more or less confirms to the quality and reliable standards.

Levels of Testing

In order to uncover the errors present in different phases we have the concept of levels of testing. The basic levels of testing are as shown below…

Acceptance Testing

System Testing

Integration Testing

Unit Testing

Client Needs

Requirements

Design

Code

*System Testing*

**The philosophy behind testing is to find errors. Test cases are devised with this in mind. A strategy employed for system testing is code testing.**

*Code Testing:*

This strategy examines the logic of the program. To follow this method we developed some test data that resulted in executing every instruction in the program and module i.e. every path is tested. Systems are not designed as entire nor are they tested as single systems. To ensure that the coding is perfect two types of testing is performed or for that matter is performed or that matter is performed or for that matter is performed on all systems.

*Types Of Testing*

* **Unit Testing**
* **Link Testing**

*Unit Testing*

Unit testing focuses verification effort on the smallest unit of software i.e. the module. Using the detailed design and the process specifications testing is done to uncover errors within the boundary of the module. All modules must be successful in the unit test before the start of the integration testing begins.

In this project each service can be thought of a module. There are so many modules like Login, HWAdmin, MasterAdmin, Normal User, and PManager. Giving different sets of inputs has tested each module. When developing the module as well as finishing the development so that each module works without any error. The inputs are validated when accepting from the user.

In this application developer tests the programs up as system. Software units in a system are the modules and routines that are assembled and integrated to form a specific function. Unit testing is first done on modules, independent of one another to locate errors. This enables to detect errors. Through this errors resulting from interaction between modules initially avoided.

*Link Testing*

Link testing does not test software but rather the integration of each module in system. The primary concern is the compatibility of each module. The Programmer tests where modules are designed with different parameters, length, type etc.

*Integration Testing*

After the unit testing we have to perform integration testing. The goal here is to see if modules can be integrated proprerly, the emphasis being on testing interfaces between modules. This testing activity can be considered as testing the design and hence the emphasis on testing module interactions.

In this project integrating all the modules forms the main system. When integrating all the modules I have checked whether the integration effects working of any of the services by giving different combinations of inputs with which the two services run perfectly before Integration.

*System Testing*

Here the entire software system is tested. The reference document for this process is the requirements document, and the goal os to see if software meets its requirements.

Here entire ‘ATM’ has been tested against requirements of project and it is checked whether all requirements of project have been satisfied or not.

*Acceptance Testing*

Acceptance Test is performed with realistic data of the client to demonstrate that the software is working satisfactorily. Testing here is focused on external behavior of the system; the internal logic of program is not emphasized.

In this project ‘Network Management Of Database System’ I have collected some data and tested whether project is working correctly or not.

Test cases should be selected so that the largest number of attributes of an equivalence class is exercised at once. The testing phase is an important part of software development. It is the process of finding errors and missing operations and also a complete verification to determine whether the objectives are met and the user requirements are satisfied.

*White Box Testing*

This is a unit testing method where a unit will be taken at a time and tested thoroughly at a statement level to find the maximum possible errors. I tested step wise every piece of code, taking care that every statement in the code is executed at least once. The white box testing is also called Glass Box Testing.

I have generated a list of test cases, sample data. which is used to check all possible combinations of execution paths through the code at every module level.

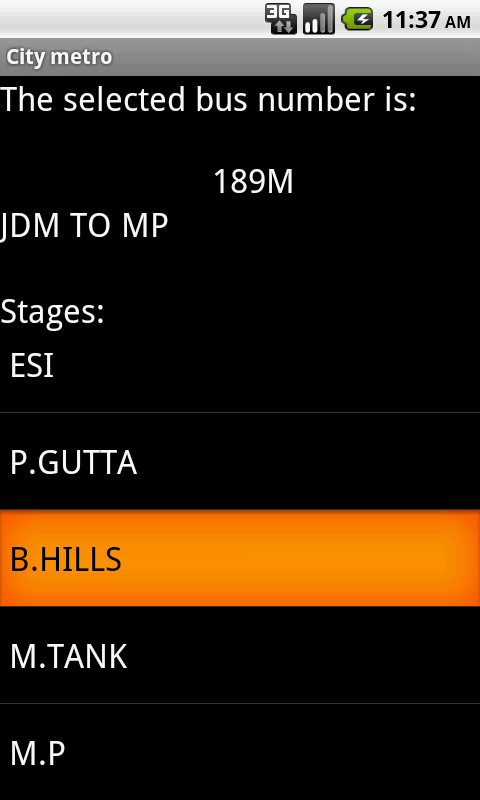
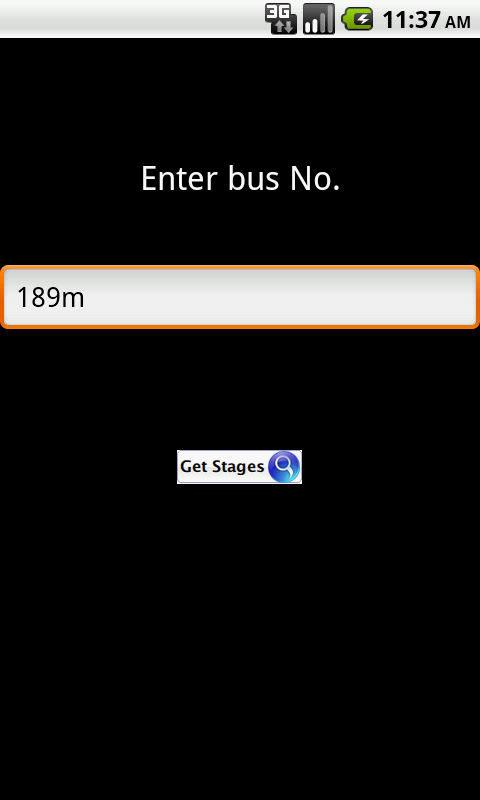
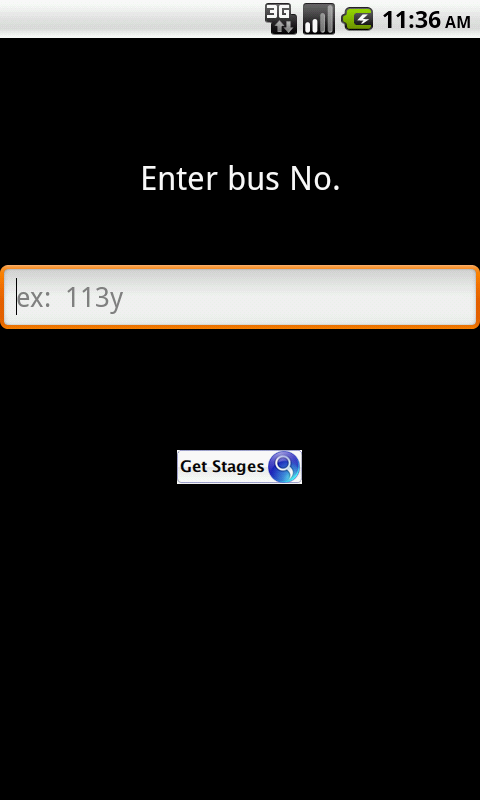
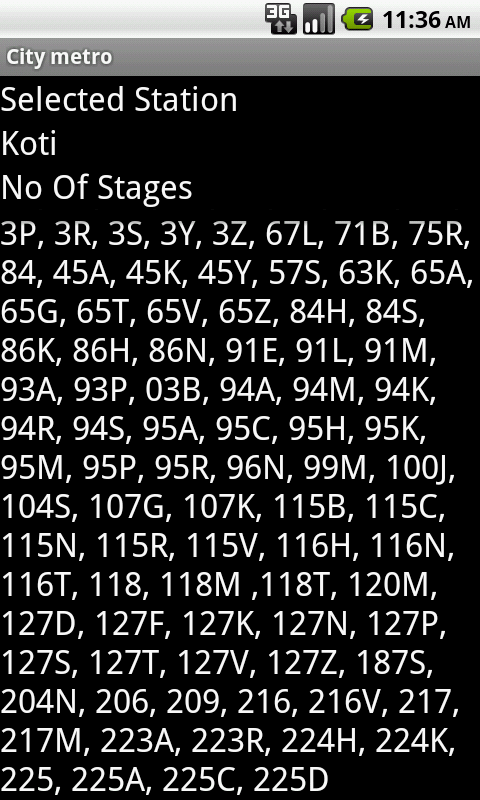
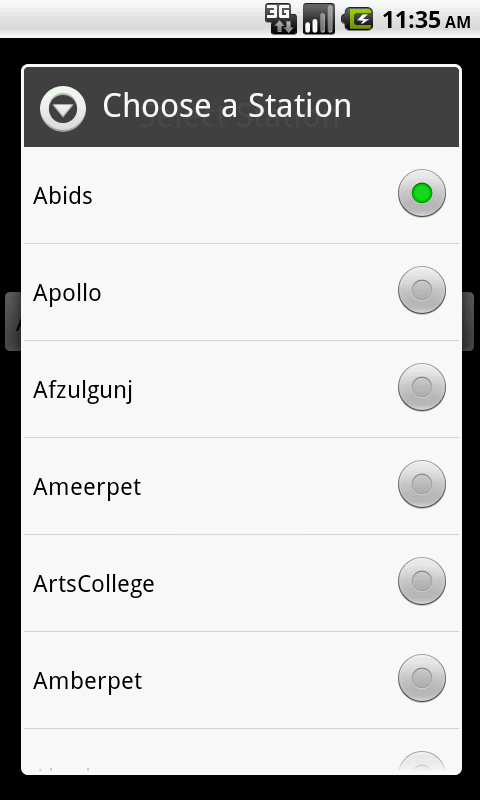
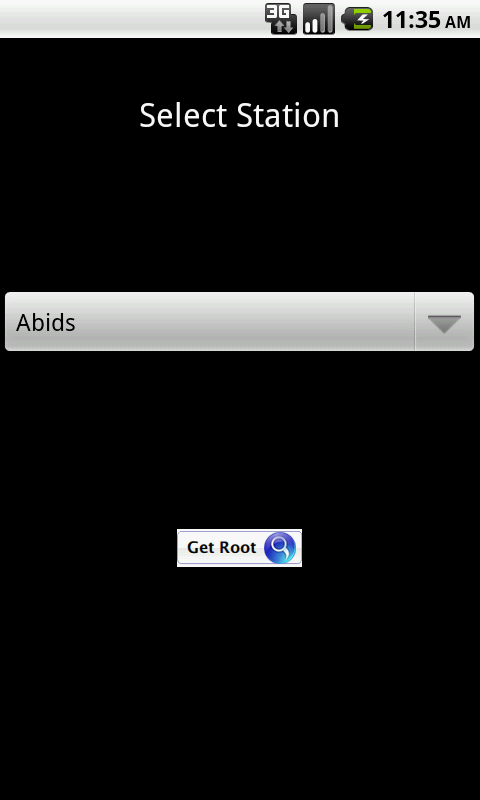
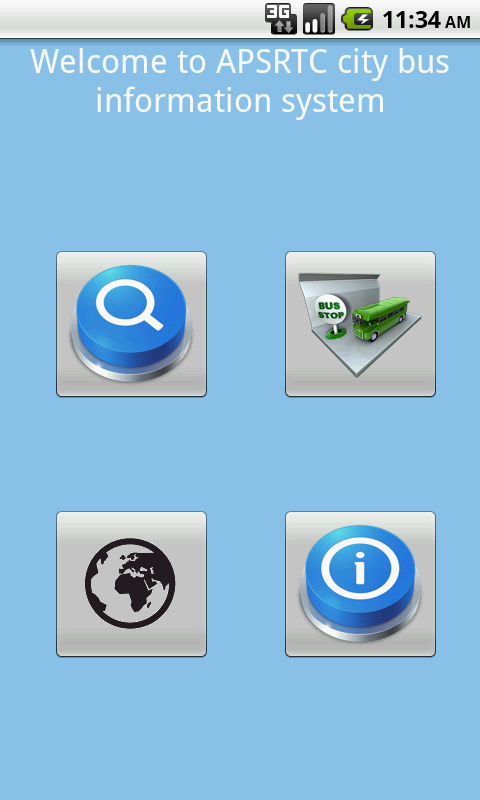
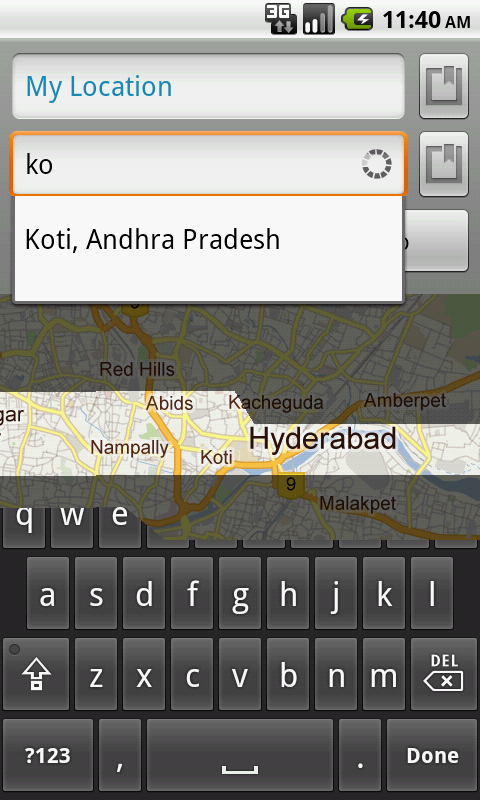
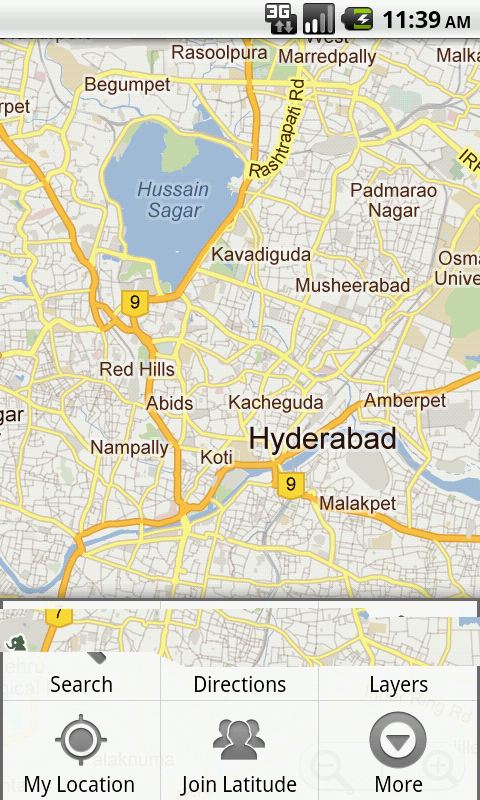
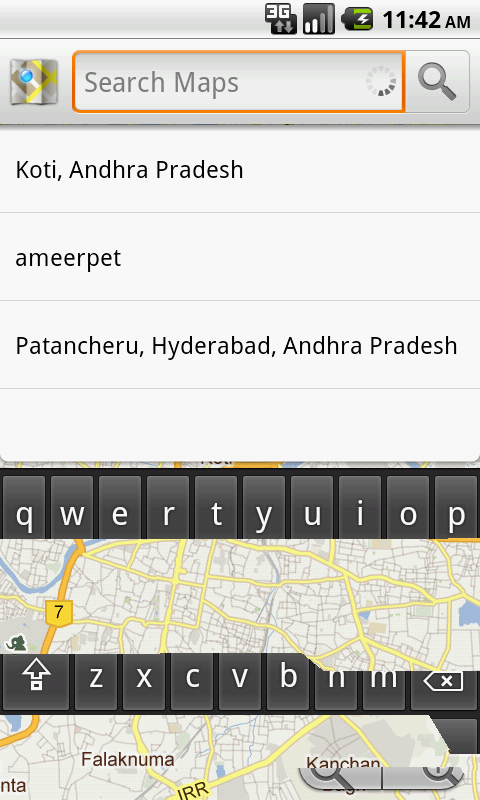
*Black Box Testing*

This testing method considers a module as a single unit and checks the unit at interface and communication with other modules rather getting into details at statement level. Here the module will be treated as a block box that will take some input and generate output. Output for a given set of input combinations are forwarded to other modules.

*Criteria Satisfied by Test Cases*

* + - * 1. Test cases that reduced by a count that is greater than one, the number of additional test cases that much be designed to achieve reasonable testing.
        2. Test cases that tell us something about the presence or absence of classes of errors, rather than an error associated only with the specific test at hand.



**Conclusion**

The analysis, requirements gathering and designing phases has been completed successfully. The coding part can be done easily according to the designs, and it can be tested by taking the “test cases”. It is user-friendly, and has required options, which can be utilized by the user to perform the desired operations. The software can be developed using android java libraries as front end and SQLite as back end in windows environment.

**References**

[1]Beginning android application development. Wei Meng Lee

[2] Software Engineering 5th edition - Roger Pressman

[3] <http://www.stackoverflow.com>

[4] <http://www.google.co.in>

[5] http://www.developer.android.com