JUIT NAVIGATION SYSTEM

Submitted in partial fulfilment of the Degree of

Bachelor of Technology (E.C.E.)

under the supervision of

Dr. Davinder Singh Saini

By

Gagandeep Singh Gujral (091103)

Priya Kukreja (091116)

Sakshi Kukreja (091118)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY,

WAKNAGHAT

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY

WAKNAGHAT

SOLAN, HIMACHAL PRADESH

CERTIFICATE

This is to certify that project report entitled **"JUIT Navigation System"**, submitted by **Gagandeep Singh**, **Priya Kukreja** and **Sakshi Kukreja** in partial fulfillment for the award of degree of Bachelor of Technology in Electronics and Communication Engineering to **Jaypee University of Information Technology**, Waknaghat, Solan has been carried out under my supervision.

This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

Date:

Project Supervisor

Dr. D.S. Saini

Associate Professor

E.C.E. Deptt.-JUIT

ACKNOWLEDGEMENT

We are highly grateful to **Prof. Dr. Sunil Bhooshan**, Professor and Head, Dept. of Electronics and Communication Engineering, for providing us the opportunity to work on the project.

With great pleasure we express our gratefulness to our guide and mentor **Dr. D.S. Saini,** Associate Professor, Dept. of Electronics and Communication Engineering, for his valuable and sustained guidance and careful supervision during the project.

We express our sincere thanks to all the faculty members of **JAYPEE UNIVERSITY OF INFORMATION AND TECHNOLOGY** who have helped us directly or indirectly. Without their help and guidance we would not have been able to successfully complete our project.

Date:

Group Members:

Gagandeep Singh Gujral-091103

Priya Kukreja-091116

Sakshi Kukreja-091118

ABSTRACT

The availability of powerful mobile phones along with advances in software development platform and databases have led us to create an ANDROID application. The Application which we have built will be useful for any user visiting the University in order to help them to find landmarks inside and outside. Once this application is installed in a mobile phone, it starts locating the location of the handset by fetching the satellite signals with the help of GPS. The application that we have built would be concerned with the location of the user. It will be able to fetch the user its current location on the map. The user can also get the direction from his source to destination path. The application will help the user to search for any place or location within the campus and the result will be shown on the map. It gets the updated location frequently as the location changes and by which fetching the address and awaking the user about the location change. The user also gets the related information about the destination. The user can also get the idea of various departments and facilities in and around the university that is the user also has an option of getting to know about the nearby places such as Temple, Laundry, ATM, Cafe etc...

Since our application is based on GOOGLE Android operating system our system would send the complete address (postal address) as to where the user is and make an easy way to the visitors. After fetching GPS co-ordinates from satellite, our system would check if the mobile is connected to internet. If yes then mobile would send those GPS co-ordinates to GOOGLE MAPS and fetch the postal address from Google maps. This postal address is displayed on the screen.

Table of Contents

LIST OF FIGURES

LIST OF SNAPSHOTS

Chapter 1

Introduction

1.1 JUIT NAVIGATION SYSTEM

JUIT Navigation System would provide user its current location providing all possible directions from source to destination. The related information about the destination and the places of interest in the surroundings such as nearby Temple, Restaurants, ATM, etc... It should provide user the directions from one place to another and provide user to explore the map of any location or places.

On location change our system would receive the signals from satellite and the updated location to the user there by the user need no search the address frequently. Then our system would retrieve GPS co-ordinates from satellites and fetches the location. Then JUIT Navigation System would connect to internet and send those GPS co-ordinates to GOOGLE MAPs and then using Google maps API, it would retrieve the postal address. This functionality is not found on any of the existing system.

Location Based Information Systems combines all the technological advances to create a new breed of applications known as LOCATION BASED SERVICES (LBS).Global demand for location-based services continues to skyrocket because of the availability of cellular phones, the new level of service these applications provide, and the important role of LBS will play in future software systems. Creating LBS is very challenging, as there are many players involved and many issues still unsolved. LBS have to deal with erroneous and variable information, as the accuracy of GPS fixes depends on the positioning system, user location, weather conditions, interferences, and others. Cellular communication networks also introduce challenges due to the nature of the wireless transmission where signals fade, transmissions disconnect, and errors in the data occur. LBIS need to be conscious about the limited resources available in the mobile devices. LBS, as a part of LBIS, interact with other systems and databases to retrieve context information and therefore, be able to provide better information to the user. Finally, cellular phone applications need to have a standard way to access position information and interact with the rest of the system, regardless of the cellular phone manufacturer and cellular system.

1.2 PROBLEM STATEMENT

To design a Location Based Information System that would provide user its current location. Depending upon the user location it should also provide information such as nearby Restaurants, ATMs, etc... It should provide user the directions from one place to another and provide user to explore the map of any location or places.

LAYOUT OF PROJECT

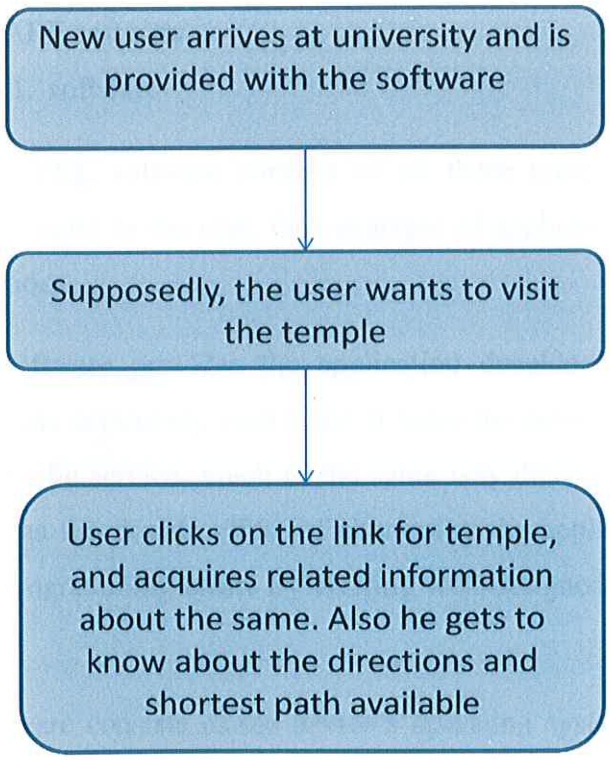

New user arrives at university and is provided with the software
Supposedly, the user wants to visit the temple
User clicks on the link for temple, and acquires related information about the same. Also he gets to know about the directions and shortest path available

Figure 1.1: Layout of the Project.

"THE FUNDAMENTAL PIECE OF INFORMATION IS THE LOCATION OF THE USER"

1.3 COMPLETE LBIS REAL-TIME TRACKING SYSTEM

Four Major Components

1) THE POSITIONING SYSTEM—GPS systems
2) THE CLIENT DEVICE
3) THE TRANSPORT NETWORK—the cellular network (GPRS)
4) THE SERVERS- Google maps

1.4 MOBILE DEVICE ARCHITECTURE

Cellular phones, as any other computer, contain the following three layers of software:

- APPLICATION-LEVEL software
- MIDDLEWARE software
- LOW-LEVEL software

APPLICATION-LEVEL software consists of all those programs running in the device that are meaningful to the user. One example of application level software is the tracking application.

MIDDLEWARE software provides the application developer with easy to use interfaces that automate commonly used tasks. It hides the developer from the details of providing the specific service, much in the same way drivers do, but at a higher level. Middleware has the double effect of shortening the application development time and reducing programming errors by offering well-designed and proved service interfaces.

LOW-LEVEL software consists of the device's operating system, the drivers, and virtual machines. The operating system is in charge of coordinating the activities and assigning the resources of the computer in a controlled manner to all running applications. As such, the operating system is directly attached to the hardware of the computer. Drivers are hardware- dependent-specific programs utilized by the operating system to communicate with a particular hardware device, such as a printer or a network interface card. Virtual machines are an additional software layer on top of the operating system that also executes computer programs. The importance of virtual machines is that they make the applications operating system independent allowing real portability of applications among computers. Virtual machines offer applications the same services and interfaces and hide the particularities of each operating system. Of course, there must be a specific virtual machine for each operating system. Since applications developed using the java programming language run on java virtual machines.

1.5 SOFTWARE ARCHITECHTURE

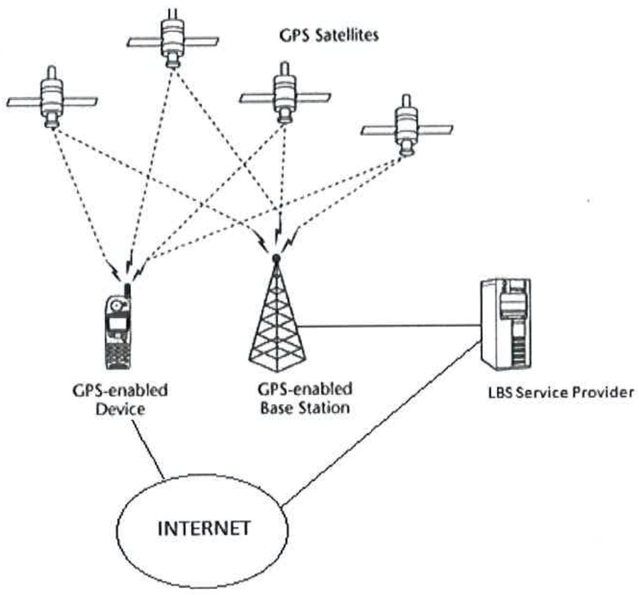


Figure 1.2: Software Architecture of the Application.

Here the above figure represents the software architecture of the application. The GPS satellites track the location of the user i.e. a GPS enabled device. The LBS Service provider with the help of network provider sends the information to the user through the internet.

1.5.1 LOCATION PROVIDER ARCHITECHTURE USED

MOBILE-BASED location provider architecture----Here, the LBS provider develops the application running in the user device, and the client device has the capability of obtaining its own information through the use of an embedded GPS, in collaboration with the cellular network provider. Once the client uses its location, it uses data connectivity through the cellular network via GPRS, to send it to the LBS provider server for storage and further updates and processing .The server receives location updates and queries from the user and then sends back the required information. It is worth mentioning that under this architecture, clients are not limited to cellular phones. Any GPS-enabled client with network connectivity can be part of the LBS. The main disadvantage of the mobile-based location provider architecture is that it has the potential to flood the network with unnecessary information, as different LBS providers implement their own applications and do not share the location information. On the plus side, mobile-based location provider architecture favours the rapid development of LBS applications, as it imposes neither major financial nor technical barriers.

1.5.2CLIENT-SIDE SOFTWARE ARCHITECTURE

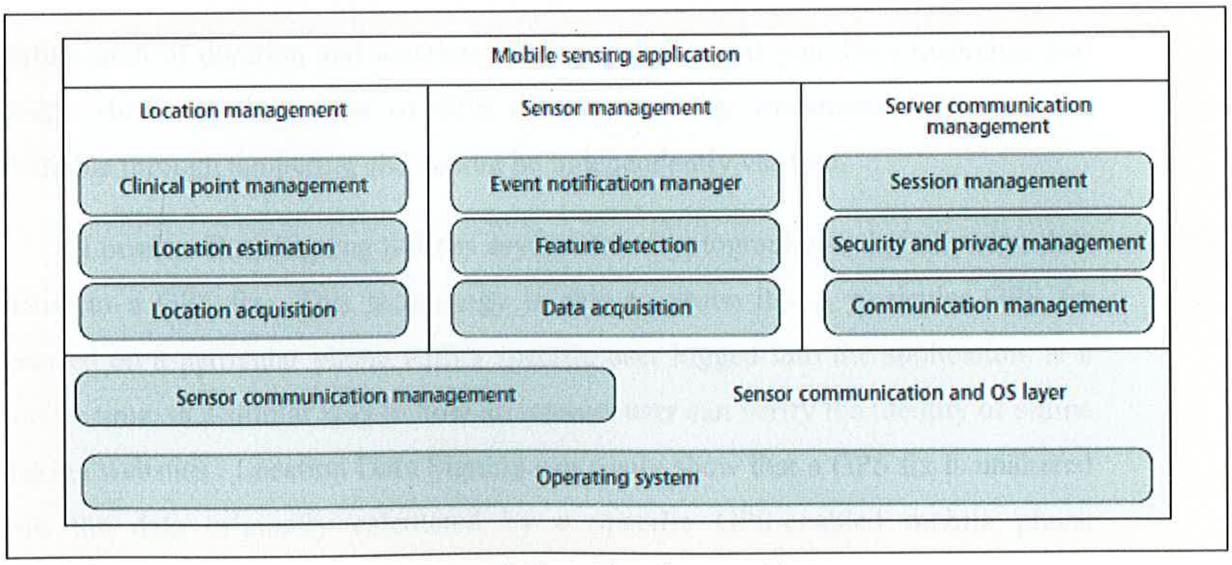

Mobile sensing application
Location management
Sensor management
Server communication management
Clinical point management
Event notification manager
Session management
Location estimation
Feature detection
Security and privacy management
Location acquisition
Data acquisition
Communication management
Sensor communication management
Sensor communication and OS layer
Operating system

Figure 1.3: Components of client side software architecture.

The client-side software architecture includes those software components that reside in the mobile device, the cellular phone. The architecture sits on top of the mobile device's operating system. The architecture has the platform at the bottom and the location-based application at the top, which is the application that runs in the mobile device and possibly interacts with the user. Between these two layers, the architecture presents several modules, each in charge of performing several functions. For example, it can be seen how the positioning data is obtained using the API included in the platform we use and how these GPS fixes are passed up to the application, either directly or through some other modules in charge of recalculating and estimating the position and making the position private. The GPS fixes can be sent from the application to the server directly using the UDP transport layer protocol, or they can be passed through some other modules in charge of saving energy not transmitting unnecessary fixes. Finally, the architecture also shows a Session management module, which is in charge of getting the log-in information from the user and establishing a new session with the server to store all the data from this particular device and user. This information is sent to the system's database using the TCP transport layer protocol for reliability purposes.

1.6 LOCATION DATA SIGNING

GPS data are increasingly being used by businesses and governments to support key operations (e.g. verification of mileage and time for workers, confirmation of duration and location of car use for pay-as-you-drive insurance and taxes). However, these uses of GPS data have a key weakness: GPS data are falsifiable through tampering and cannot be independently verified.

Location Data Signing utilizes asymmetric cryptography to digitally sign data related to a GPS fix. This technology is able to prove that a particular GPS fix occurred on a particular phone with a specific user logged into the application, at a specific time, in a similar way to how an internet user can verify the identity of online banking websites. Location Data Signing can easily show that a GPS fix is unaltered from the data originally calculated by a specific GPS-enabled mobile phone application.

1.6.1 CRITICAL POINT ALGORITHM

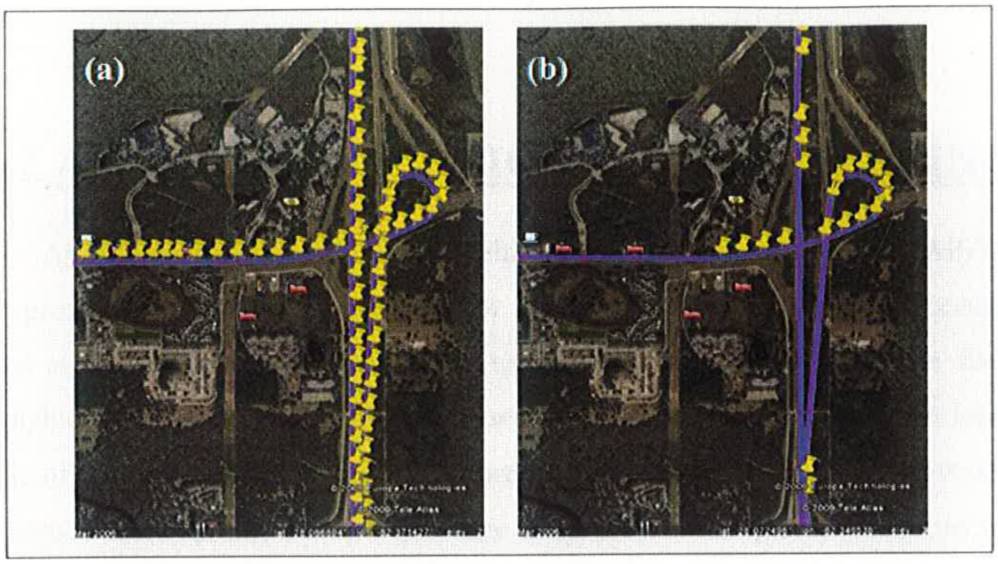


Figure 1.4 Critical Point Algorithm. (a)Tracking positions at all the points. (b) Tracking positions at only critical points.

GPS generates a large amount of location data, this data must be carefully managed to avoid wasting resources (e.g. battery energy) by transferring fixes to a server that may not contain useful information (e.g. repeated GPS fixes when the user is standing still, fixes lying upon the same vector when travelling in a straight line). Traditional LBS transfer fixes to the server regularly based on a fixed time period, which can result in transferring some of these irrelevant fixes, as well as accidentally discarding fixes that actually contain useful information.

The path of the user can be accurately represented by using only small portions of GPS data generated by mobile phones. The Critical Point Algorithm (CPA) uses the change in direction between sequential points as well as the user's speed to filter non-critical points from a set of GPS data, therefore, only critical points representing the path of the user remain (Figure 4). Only these critical points are transferred to a server for storage and analysis. By pre-filtering GPS data before it leaves the device, CPA saves battery energy, reduce data transfer costs, and saves network bandwidth, while still accurately representing the user's path. Table 1 shows the storage and financial savings per trip for transferring only critical points to the server. Assuming that each user takes around 4 trips per day, the savings quickly compound for thousands of users across long periods of time.

1.6.2 ADAPTIVE LOCATION DATA BUFFERING

As UDP is used to send location data to the server very efficiently, LAISYC 2-layer protocol by itself does not confirm that every location data fix successfully arrives at the server. In real-time tracking, the loss of occasional location fixes is acceptable since another location update will soon follow. However, since location data is often referenced after-the-fact to provide metrics (e.g. distance-travelled) and reconstruct users' paths, the loss of large numbers of contiguous fixes can cause significant problems for applications. Extended gaps can result from lack of support for simultaneous voice and data services on mobile phones or "dead zones" with no cellular signal. Adaptive Location Data Buffering increases the probability that most location data points will arrive at the server by performing occasional reliability checks. Before each location data UDP transmission, device-side APIs are checked to assess the current level of cellular signal and determine if a successful UDP transmission is probable. If not, the location data is buffered to either main memory or persistent storage on the mobile device. Once it is detected that UDP transmissions are likely to succeed, the buffered data is then sent via UDP and deleted on the device. Similarly, the device also occasionally checks with the server via the TCP protocol to ensure that it still has an open line of successful communication. If TCP fails, location data is buffered on the device until another future TCP check succeeds. Adaptive Location Data Buffering therefore increases the chance that most location data will successfully arrive at the server, while preserving the efficiency of using UDP to send most location data.

1.6.3 LOCATION DATA ENCRYPTION

The interception of location data during transfer over the Internet is a significant security threat to LBS. While secure TCP connections are implemented within Java through Secure Socket Layer (SSL), the implementation of secure UDP is left to the application developer. We currently use the Advanced Encryption Standard (AES) to encrypt location data as it is transferred using UDP to ensure a secure LBS.

1.6.4 POSITION RECALCULATION

Dynamically varies the interval of time between location updates depending on the real-time need of the system, and an estimation of whether the user is currently moving or is standing still. Numerous states, and gradual state transitions, are used to reduce the impact of GPS outliers. Figure 3 shows the battery life benefits of this technology, while still preserving the ability to quickly sample a user's location while they are moving. The current core technology estimates the correct state (e.g., moving or stationary) around 89% of the time, and recent experiments with modified Kalman Filters extend this to 93%.

1.6.5 SESSION MANAGEMENT

In human-computer interaction, session management is the process of tracking the activity of the user across sessions of interaction with the computer system. Typical session management tasks in a desktop environment includes the user's activity and tracking of open applications and documents so that whenever the user logs in sometime later, the same state can be restored. In case the session has expired, session management requires the user to login again. Information of different kind is stored on the server-side between HTTP requests as well.

1.6.6 DESKTOP SESSION MANAGEMENT

Saving and restoring of desktop sessions can be done by desktop session manager. A desktop session means all the currently running windows and its content. Session manager on Linux-based systems is provided by X session manager. On Microsoft Windows systems, no session manager is included in the system. There are numerous third-party applications which provide session management. For example- twins play.

1.6.7 BROWSER SESSION MANAGEMENT

Session management is particularly useful in a web browser when it comes to saving of all the open pages and settings and their restoration at a later stage. In order to recover from a system or application crash, restoration of pages and settings can be done on next run. The browsers which support session management are Google Chrome, Omni Web and Opera. Other browsers such as Mozilla Firefox support session management through third-party plug- in or extensions.

1.6.8 WEB SERVER SESSION MANAGEMENT

Hypertext Transfer Protocol (HTTP) is stateless: a client computer running a web browser must establish a new Transmission Control Protocol (TCP) network connection to the web server with each new HTTP GET or POST request. The web server, therefore, cannot rely on an established TCP network connection for longer than a single HTTP GET or POST operation. Session management is the technique used by the web developer to make the stateless HTTP protocol support session state. For example, once a user has authenticated oneself to the web server, his/her next HTTP request (GET or POST) should not cause the web server to ask him/her for him/her account and password again. For a discussion of the methods used to accomplish this please see HTTP cookie.

The session information is stored on the web server using the session identifier (session ID) generated as a result of the first (sometimes the first authenticated) request from the end user running a web browser. The "storage" of session IDs and the associated session data (user name, account number, etc.) on the web server is accomplished using a variety of techniques including, but not limited to: local memory, flat files, and databases.

In situations where multiple web servers must share knowledge of session state (as is typical in a cluster environment—see computer cluster) session information must be shared between the cluster nodes that are running web server software. Methods for sharing session state between nodes in a cluster include: multicasting session information to member nodes.

1.7 TYPE of SERVICES

- Finding current location of the user.
- Obtaining directions to a place from current location.
- Finding nearby places e.g. ATMs, Restaurants, etc.
- Search map of any place e.g. Delhi, Chandigarh, New York, etc.

These services are of the request in which user queries the system, including the current location and the system responds with the specific information after searching in other systems and databases.

1.8 SCOPE

- Emergency Services and Disaster Management.
- Fleet and Logistics Management
- Navigation
- Vehicle Tracking
-

1.9 RESOURCES REQUIREMENTS

There are two types of requirements in this project:

- Software Requirements
- Hardware Requirements

1.9.1 SOFTWARE REQUIREMENT

- We need **Eclipse Platform** with **Android SDK** plug in to develop an application which will be hosted on client mobile device.

- **AVD Emulator Plug in**

1.9.2 HARDWARE REQUIREMENT

Computer system with

- 512 MBRAM
- Min 1 GB of free hard disk space.

A **Mobile device,** Android platform and enabled with GPS.

1.10 LIMITATIONS

There are different parties involved in the complete application.

- GPS enabled Handset.
- Network Provider.
- LBS Provider.
- GPS Satellites.
- Internet.

Therefore the performance of the application depends on a lot of things.

1. Application will not run on every handset. It should be GPS enables, GPRS enabled and Android based.
2. Application will not perform efficiently in low signals, or will not perform at all in absence of network signals.
3. Application will not perform in case there is some problem with LBS service provider.

Chapter 2

GLOBAL POSITIONING SYSTEM

2.1 GPS

The global positioning system is perhaps the most widely used and ubiquitous system to obtain users' positions. GPS is a complex and expensive system consisting of three main segments namely, the user segment, the space segment, and the control segment. These three segments give an opportunity to the GPS receivers to determine their speed, location, time and direction.

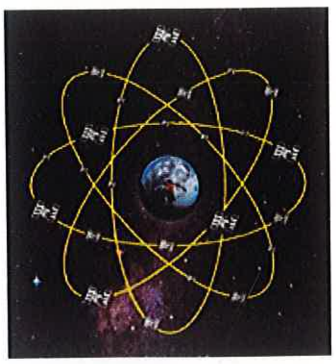
The GPS program is used to provide critical capabilities to military and commercial users across the globe. Moreover, GPS has been the driving force behind the modernizing the global air traffic system.

In order to overcome the earlier navigation system's limitations, the GPS project was developed in 1973. Integrating ideas from several predecessors, including a number of classified engineering design studies from the 1960s. GPS was created and realized by the **U.S.** D o D and was run with the help of 24 satellites, originally. In the year 1994, GPS became fully operational

It consists of three segments:

- Space Segment
- Control Segment
- User Segment

2.1.1 SPACE SEGMENT



- 24+ satellites
  - 6 planes with 55° inclination
  - Each plane has 4-5 satellites
  - Broadcasting position and time info on 2 frequencies
  - Constellation has spares

Figure 2.1: Space segment of GPS.

The space segment consists of the orbiting GPS satellites. A total of 24 satellites, four in six orbital planes centred on the earth, are needed so that at least six satellites can be detected by a GPS receiver from almost anywhere on earth. Currently, the constellation of GPS satellites consists of 31 satellites; seven more have been added to provide redundant signals and improve the precision of GPS receivers and the reliability and availability of the system.

2.1.2 CONTROL SEGMENT

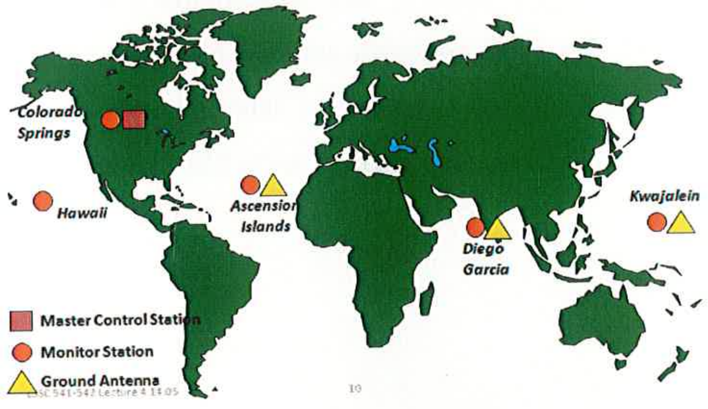


Figure 2.2: Control segment of GPS.

This segment comprises several ground stations that track and monitor the space segment and a main control station. The main control station is in charge of updating the on-board atomic clocks of the satellites and their ephemerides, or a table with the exact position of the satellites in the sky.

2.1.3 USER SEGMENT

•Dual Use System Since 1985 (civil & military)

•Civilian community was quick to take advantage of the system

- Hundreds of receivers on the market
- 3 billion in sales, double in 2 years
- 95% of current users

Figure 2.3: User segment of GPS.

The user segment is made up of all GPS receivers, a high-level architecture of a GPS receiver consists of the receiving part, with an antenna tuned to the frequencies of the GPS satellites, a main processor, and a crystal oscillator. Depending on the receiver, they can monitor anywhere from 4 to 20 channels .The calculated position along with additional information derived from the satellite may then be further processed to build other systems such as stand-alone navigational systems or tracking applications.

2.2 COORDINATION OF THREE SEGMENTS OF

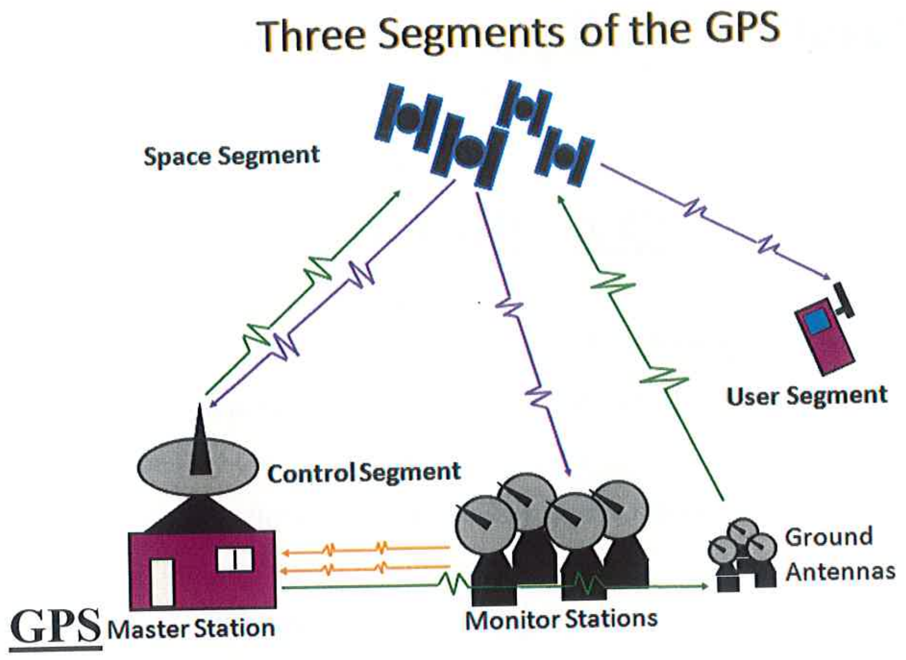


Figure 2.4: How the three segments of the GPS coordinates with each other.

Chapter 3

Android Application Development

3.1 WHAT IS ANDROID – A GPHONE?

The weeks and months before Google released the Android SDK there had been a lot of rumours about a so called GPhone. It was said to be a mobile device manufactured by Google providing free communication by showing context-sensitive advertisements to the user on the device itself. Render of a potential GPhone But on November 5th 2007 Andy Rubin2 announced:
- [The] Android [Platform] – is more significant and ambitious than a single phone.

Google within the Open Handset Alliance (OHA) delivers a complete set of software for mobile devices: an operating system, middleware and key mobile applications. What was released a week later was not a final product, but a –First Look SDK‖ what many did not realize. Major news sites grabbed the discomforts of some developers who said that Android is full of bugs and heavily lacks of documentation. But the majority says that Android is not buggier than any other software at this stage.

Andy Rubin – Google Director of Mobile Platforms and book - Android Programming Let's take a look at what the OHA emphasizes on its Android Platform:

3.2 OPENNESS

Android was built from the ground-up to enable developers to create compelling mobile applications that take full advantage of all a handset has to offer. It is built to be truly open. For example, an application could call upon any of the phone's core functionality such as making calls, sending text messages, or using the camera, allowing developers to create richer and more cohesive experiences for users.‖

This is true, as a developer you can do everything, from sending short messages with just 2 lines of code, up to replacing even the HOME-Screen of your device. One could easily create a fully customized operating system within weeks, providing no more of Google's default application to the user.
–Android is built on the open Linux Kernel. Furthermore, it utilizes a custom virtual machine that has been designed to optimize memory and hardware resources in a mobile environment. Android will be open source; it can be liberally extended to incorporate new cutting edge technologies as they emerge. The platform will continue to evolve as the developer community works together to build innovative mobile applications.‖

Here Google is talking of the so called Dalvik virtual machine (DalvikVM), which is a register based virtual machine, designed and written by Dan Bornstein and some other Google engineers, to be an important part of the Android platform. In the words –register based‖ we find the first difference to normal Java virtual machines (JVM) which are stack based.

3.3 All APPLICATIONS ARE CREATED EQUALLY

Android does not differentiate between the phone's core applications and third-party applications. They can all be built to have equal access to a phone's capabilities providing users with a broad spectrum of applications and services. With devices built on the Android Platform, users will be able to fully tailor the phone to their interests. They can swap out the phone's home screen, the style of the dialler, or any of the applications. They can even instruct their phones to use their favourite photo viewing application to handle the viewing of all photos.

Once again this is all true. Developers can 100% customize their Android-Device. The Android System Communication is based on so called Intents, which are more or less just a String (with some data attached) which defines an action that

needs to be handled. An example for this is:
‖android.provider.Telephony.SMS_RECEIVED‖ one can simply listen on that Intent by writing about 5 lines of definitions. The system would then recognize that there is more than one application that wants to handle that Intent and ask the user to choose which one he or she would like to handle the Intent.

3.4 BREAKING DOWN APPLICATION BOUNDARIES

Android breaks down the barriers to building new and innovative applications. For example, a developer can combine information from the web with data on an individual's mobile phone - such as the user's contacts, calendar, or geographic location - to provide a more relevant user experience.

With Android, a developer could build an application that enables users to view the location of their friends and be alerted when they are in the vicinity giving them a chance to connect. Fast & easy application development -Android provides access to a wide range of useful libraries and tools that can be used to build rich applications.

For example, Android Programming enables developers to obtain the location of the device, and allows devices to communicate with one another enabling rich peer-to-peer social applications. In addition, Android includes a full set of tools that have been built from the ground up alongside the platform providing developers with high productivity and deep insight into their applications. Since the Web 2.0 revolution, making content rich applications within minutes is no more illusion. Android has brought developing to unknown speeds.

3.5 ANDROID DEVLOPEMENT

Android applications are written in the Java programming language. The compiled Java code along with any data and resource files required by the application is bundled by the apt tool into an Android package, an archive file marked by an .apk suffix. This file is the vehicle for distributing the application and installing it on mobile devices; it's the file users download to their devices. All the code in a single .apk file is considered to be one application. In many ways, each Android application lives in its own world:

- By default, every application runs in its own Linux process. Android starts the process when any of the applications code needs to be executed, and shuts down the process when it's no longer needed and system resources are required by other applications.
- Each process has its own Java virtual machine (VM), so application code runs in isolation from the code of all other applications.
- By default, each application is assigned a unique Linux user ID. Permissions are set so that the application's files are visible only that user, only to the application itself although they can be exported to other applications also.

It's possible to arrange for two applications to share the same user ID, in which case they will be able to see each other's files. To conserve system resources, applications with the same ID can also arrange to run in the same Linux process, sharing the same VM.

3.5.1 APPLICATION COMPONENTS

A central feature of Android is that one application can make use of elements of other applications (provided those applications permit it). For example, if your application needs to display a scrolling list of images and another application has developed a suitable scroller and made it available to others, you can call upon that scroller to do the work, rather than develop your own. Your application doesn't incorporate the code of the other application or link to it. Rather, it simply starts up that piece of the other application when the need arises.

For this to work, the system must be able to start an application process when any part of it is needed, and instantiate the Java objects for that part. Therefore, unlike applications on most other systems, Android applications don't have a single entry point for everything in the application (no main () function, for example). Rather, they have essential components that the system can instantiate and run as needed.

A. ACTIVITIES

An activity presents a visual user interface for one focused endeavour the user can undertake. For example, an activity might present a list of menu items users can choose from or it might display photographs along with their captions. A text messaging application might have one activity that shows a list of contacts to send messages to, a second activity to write the message to the chosen contact, and other activities to review old messages or change settings. Though they work together to form a cohesive user interface, each activity is independent of the others. Each one is implemented as a subclass of the Activity base class.

An application can comprise only one activity or, it may contain several like the text messaging application. What the activities are, and how many there are depends, of course, on the application and its design. Typically, one of the activities is marked as the first one that should be presented to the user when the application is launched. Moving from one activity to another is accomplished by having the current activity start the next one.
Each activity is given a default window to draw in. Typically, the window fills the screen, but it might be smaller than the screen and float on top of other windows. An activity can also make use of additional windows for example, a pop-up dialog that calls for a user response in the midst of the Activity or a window that presents users with vital information when they select a particular item on- screen.

The visual content of the window is provided by a hierarchy of views — objects derived from the base View class. Each view controls a particular rectangular space within the window. Parent views contain and organize the layout of their children. Leaf views (those at the bottom of the hierarchy) draw in the rectangles they respond to user actions which are directed at that space.

Thus, views are where the activity's interaction with the user takes place. For example, a view might display a small image and initiate an action when the user taps that image. Android has a number of ready-made views that you can use — including buttons, text fields, scroll bars, menu items, check boxes, and more.

A view hierarchy is placed within an activity's window by the Activity.setContentView() method. The content view is the View object at the root of the hierarchy. (See the separate User Interface document for more information on views and the hierarchy.)

B. SERVICES

A service doesn't have a visual user interface, but rather runs in the background for an indefinite period of time. For example, a service might play background music as the user attends to other matters, or it might fetch data over the network or calculate something and provide the result to activities that require it. Each and every service also extends the Service base class.

A prime example is a media player playing songs from a play list. The player application would probably have one or more activities that allow the user to choose songs and start playing them.

However, the music playback itself would not be handled by an activity because users will expect the music to keep playing even after they leave the player and begin something different. To keep the music going, the media player activity could start a service to run in the background. The system would then keep the music playback service running even after the activity that started it leaves the screen. It's possible to connect to (bind to) an on-going service (and start the service if it's not already running). While connected, you can communicate with the service through an interface that the service exposes. For the music service, this interface might allow users to pause, rewind, stop, and restart the playback.

Like activities and the other components, services run in the main thread of the application process. So that they won't block other components or the user interface, they often spawn another thread for time-consuming tasks (like music playback).

C. BROADCAST RECEIVRS

A broadcast receiver is a component that is responsible for the receiving and giving reactions to broadcast announcements. Many broadcasts originate in system code — for example, announcements that the time zone has changed, that the battery is low, that a picture has been taken, or that the user changed a language preference. Applications can also be used for initiating broadcasts such as, letting the other applications know that some data has been downloaded to the device and is available for them to use. An application can have any number of broadcast receivers to respond to any announcement it considers important. All receivers extend the Broadcast Receiver base class. Broadcast receivers do not display a user interface. However, they may start an activity in response to the information they receive, or they may use the Notification Manager to alert the user. Notifications can get the user's attention in various ways — flashing the backlight, vibrating the device, playing a sound, and so on. They typically place a persistent icon in the status bar, which users can open to get the message.

D. CONTENT PROVIDERS

A content provider makes a specific set of the application's data available to other applications. The data can be stored in the file system, in a SQLite database, or in any other manner that makes sense. The content provider extends the Content Provider base class to implement a standard set of methods that enable other applications to retrieve and store data of the type it controls. However, applications do not call these methods directly. Rather they use a Content Resolver object and call its methods instead. A Content Resolver can talk to any content provider; it cooperates with the provider to manage any interposes communication that's involved.

See the separate Content Providers document for more information on using content providers. Whenever there's a request that should be handled by a particular component, Android has the duty of making sure that the component's application process is running, if it is not then starting it, and that an appropriate instance of the component is available, creating the instance if necessary.

Activating components: intent Content providers are activated when they're targeted by a request from a Content Resolver. The other three components activities, services, and broadcast receivers are activated by asynchronous messages called intents. Intent is an Intent object that holds the content of the message. For activities and services, it names the action being requested and specifies the URI of the data to act on, among other things. For example, it might convey a request for an activity to present an image to the user or let the user edit some text. For broadcast receivers, the Intent object names the action being announced. For example, it might announce to interested parties that the camera button has been pressed.

There are separate methods for activating each type of component:

- An activity is launched (or given something new to do) by passing an Intent object to Context.startActivity() orActivity.startActivityForResult(). The responding activity can look at the initial intent that caused it to be launched by calling its getIntent() method. Android calls the activity's onNewIntent() method to pass it any subsequent intents. One activity often starts the next one. In case it is expecting the activity it's starting to give back the result, it calls startActivityForResult() instead ofstartActivity(). For example, if it starts an activity that lets the user pick a photo, it might expect to be returned the chosen photo. The result is returned in an Intent object that's passed to the calling activity's onActivityResult() method.
- A service is started (or new instructions are given to an on-going service) by passing an Intent object to Context.startService(). Android calls the service'sonStart() method and passes it the Intent object. Similarly, an intent can be passed to Context.bindService() to establish an ongoing connection between the calling component and a target service. The service receives the Intent object in an onBind() call. (If the service is not already running, bindService() can optionally start it.) For example, an activity might establish a connection with the music playback service mentioned earlier so

that it can provide the user with the means (a user interface) for controlling the playback. The activity would call bindService() to set up that connection, and then call methods defined by the service to affect the playback. A later section, Remote procedure calls, has more details about binding to a service.

- An application can be used for initiating a broadcast by just passing an Intent object to methods like Context.sendBroadcast(), Context.sendOrderedBroadcast(), Context.sendStickyBroadcast() in any of their variations. Android has the duty of delivering the intent to all interested broadcast receivers by calling their onReceive()methods. For more on intent messages, see the separate article, Intents and Intent Filters.

E. SHUTTING DOWN COMPONENTS

A content provider is active only while it's responding to a request from a Content Resolver. And a broadcast receiver is active only while it's responding to a broadcast message. So there's no need to explicitly shut down these components. Activities, on the other hand, provide the user interface. They're in a long-running conversation with the user and may remain active, even when idle, as long as the conversation continues. Similarly, services may also remain running for a long time. So Android also comprises the methods which could shut down activities and services in an orderly way:

- An activity can be shut down by calling its finish () method. One activity can shut down another activity (one it started with startActivityForResult()) by calling finishActivity().
- A service can be stopped by calling its stopSelf() method, or by calling Context.stopService(). Components might also be shut down by the system when they are no longer being used or when Android must reclaim memory for more active components. A later section, Component Lifecycles, discusses this possibility and its ramifications in more detail.

F. THE MANIFEST FILE

Before Android can start an application component, it must learn that the component exists. Therefore, applications declare their components in a manifest file that's bundled into the Android package, the .apk file that also holds the application's code, files, and resources. The manifest is a structured XML file and is always named AndroidManifest.xml for all applications. It does a number of things in addition to declaring the application's components, such as naming any libraries the application needs to be linked against (besides the default Android library) and identifying any permission the application expects to be granted. But the principal task of the manifest is to inform Android about the application's components. For example, an activity might be *declared as follows:*

```
<?xml version=

"1.0" encoding="UTF-8"?>

-<manifest android:versionName="1.0" android:versionCode="1"
package="com.juit.activities"

xmlns:android="http://schemas.android.com/apk/res/android"> <uses-sdk
android:minSdkVersion="8"/> <uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION"/> <uses-
permission

android:name="android.permission.INTERNET"/> -<application
android:label="@string/app_name"

android:icon="@drawable/icon"> <activity android:name=".MainActivity"

android:label="@string/app_name"> </activity> <activity
android:name=".AboutUsActivity"/> <activity

android:name=".WhereAmIActivity"/> <uses-library
android:name="com.google.android.maps"/> -<activity

android:name=".SplashScreen"> -<intent-filter> <action
android:name="android.intent.action.MAIN"/>

<category android:name="android.intent.category.LAUNCHER"/> </intent-filter>
</activity> <activity
```

```
android:name=".CollegePhotos"/> <activity android:name=".CollegeMap"/>
</application>

</manifest>
```

The name attribute of the <activity> element names the Activity subclass that implements the activity. The icon and label attributes point to resource files containing an icon and label that can be displayed to users to represent the activity. The other components are declared in a similar way —
<service> elements for services, <receiver> elements for broadcast receivers, and <provider> elements for content providers. Activities, services, and content providers that are not declared in the manifest are not visible to the system and are consequently never run. The broadcast receivers can either be declared in the manifest, or they can also be created in the code dynamically (as BroadcastReceiver objects) and registered with the system by calling Context.registerReceiver().

G. INTENT FILTERS

An Intent object can explicitly name a target component. If it does, Android finds that component (based on the declarations in the manifest file) and activates it. But if a target is not explicitly named, Android must locate the best component to respond to the intent. It does so by comparing the Intent object to the intent filters of potential targets. A component's intent filters inform Android about the various kinds of intents that the component can be used to handle. Like other essential information about the component, they're declared in the manifest file. Here's an extension of the previous example that adds two intent filters to the activity:

```
<application . . . >

  <activity
       android:name="com.example.proj
       ect.FreneticActivity"
       android:icon="@drawable/small_
       pic.png"
       android:label="@string/freneticL
       abel"
```

```
        .          .

        .

        .

        >

      <intent-filter . . . >

        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER"

/>

      </intent-filter>

      <intent-filter . . . >

        <action android:name="com.example.project.BOUNCE" />

        <data android:mimeType="image/jpeg" />

        <category android:name="android.intent.category.DEFAULT" />

      </intent-filter>

    </activity>

    . . .

  </appli
  cation>
  </manif
   est>
```

The first filter in the example — the combination of the action "android.intent.action.MAIN and the category android.intent.category.LAUNCHER" — is a common one. It marks the activity as one that should be represented in the application launcher, the screen listing applications users can launch on the device. In other words, the activity is the entry point for the application; the initial one user would see when they choose the application in the launcher.

The second filter declares an action that the activity can perform on a particular type of data. A component can have any number of intent filters, each one declaring a different set of capabilities. If it doesn't have any filters, it can be activated only by intents that explicitly name the component as the target. For a broadcast receiver that's created and registered in code, the intent filter is instantiated directly as IntentFilter object. All other filters are set up in the manifest.

H. AVD EMULATOR

The Android SDK includes a mobile device emulator — a virtual mobile device that runs on your computer. The emulator lets you develop and test Android applications without using a physical device.

The moment at which the emulator is running, one can do interaction with the emulated mobile device just as you would an actual mobile device, except that you use your mouse pointer to "touch" the touchscreen and can use some keyboard keys to invoke certain keys on the device.

Chapter 4

Project Development

4.1. Java Development Kit- JDK

Java software development environment that is the kit is provided by Sun. It includes the JVM, compiler, debugger and other tools for developing Java applets and applications. Each new version of the JDK adds features and enhancements to the language. When Java programs are developed under the new version, the Java interpreter (Java Virtual Machine) that executes them must also be updated to that same version. On 17 November 2006, Sun announced that it would be released under the GNU General Public License (GPL), thus making it free software.

4.2 Android Software Development Kit (SDK)

A software development kit (SDK) is typically a set of software development tools that allows for the creation of applications for a certain software package, software framework, hardware platform, computer system, video game console, operating system, or similar platform. It may be something as simple as an application programming interface (API) in the form of some files to interface to a particular programming language or include sophisticated hardware to communicate with a certain embedded system. Common tools include debugging aids and other utilities often presented in an integrated development environment (IDE). SDKs also frequently include sample code and supporting technical notes or other supporting documentation to help clarify points from the primary reference material. SDKs may have attached licenses that make them unsuitable for building software intended to be developed under an incompatible license. For example, a proprietary SDK will probably be incompatible with free software development, while a GPL-licensed SDK could be incompatible with proprietary software development. LGPL SDKs are typically safe for proprietary development. A software engineer typically receives the SDK from the target system developer. Often the SDK can be downloaded directly via the Internet. Many SDKs are provided for free to encourage developers to use the system or language.

4.3 ECLIPSE .

Eclipse is a multi-language software development environment comprising an workspace and an extensible plug-in system. It is written mostly in Java. It can be used to develop applications in Java and, by means of various plug-ins, other programming languages including C, C++, PHP, Python etc. It can also be used to develop packages for the software Mathematics. Development environments include the Eclipse Java development tools (JDT) for Java, Eclipse CDT for C/C++ and Eclipse PDT for PHP, among others.

4.4 Android Development tool- ADT Plug-in

Android Development Tools (ADT) is a plug-in for the Eclipse IDE that is designed to give you a powerful, integrated environment in which to build Android applications. ADT extends the capabilities of Eclipse to let you quickly set up new Android projects, create an application UI, add packages based on the Android Framework API, debug your applications using the Android SDK tools, and even export signed (or unsigned) .apk files in order to distribute your application. Developing in Eclipse with ADT is highly recommended and is the fastest way to get started. With the guided project setup it provides, as well as tools integration, custom XML editors, and debug output pane, ADT gives you an incredible boost in developing Android applications.

4.5Android Architecture

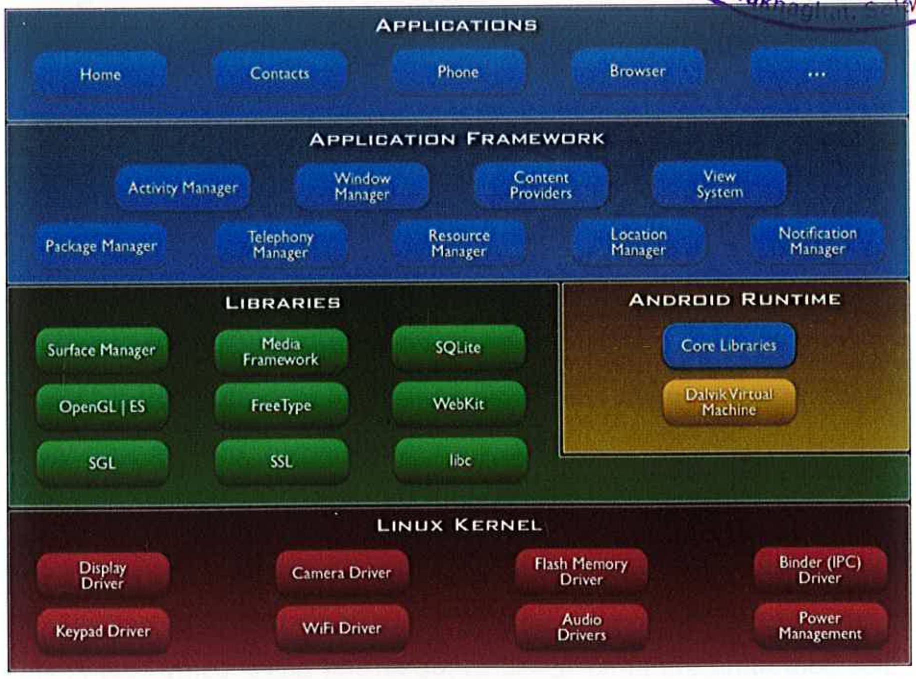


Figure 4.1: Android Architecture

Chapter 5

Application Testing

5.1 THE GOAL OF TESTING

In different publications, the definition of testing varies according to the purpose, process, and level of testing described. Miller gives a good description of testing.The general aim of testing is to affirm the quality of software systems by systematically exercising the software in carefully controlled circumstances.

The most important will be performance testing as when so many users will be simultaneously using the application and the database will be accessed as such frequent high rates it is must that performance should not deteriorate.

5.2 THE TESTING SPECTRUM

Testing is involved in every stage of our software, but the testing done at each level of software development is different in nature and has different objectives.

5.3 UNIT TESTING

This is d o n e at the lowest level. It tests the basic unit of software, which is the smallest testable piece of software, and is often called –unit, –module, or –component interchangeably.

5.4 INTEGRATION TESTING

This is performed when two or more tested units are combined into a larger structure. The test is often done on both the interfaces between the components and the larger structure being constructed, if its quality property cannot be assessed from its components.

5.5 SYSTEM TESTING

This type of testing is required to confirm the end-to-end quality of the entire system. It is based on the functional specification of the system. It is also useful when it comes to the checking of non-functional quality attributes, such as security, reliability and maintainability.

5.6 ACCEPTANCE TESTING

It is done when the completed system is handed over from the developers to the customers or users. The purpose of acceptance testing is rather to give confidence that the system is working than to find errors.

All the aforementioned techniques will be implemented by us in our next semester on the software that we aspire to make.

5.7 PERFORMANCE TESTING

Special tools are available which enable us to performance test any application with (n) number of users. Here it is very important.

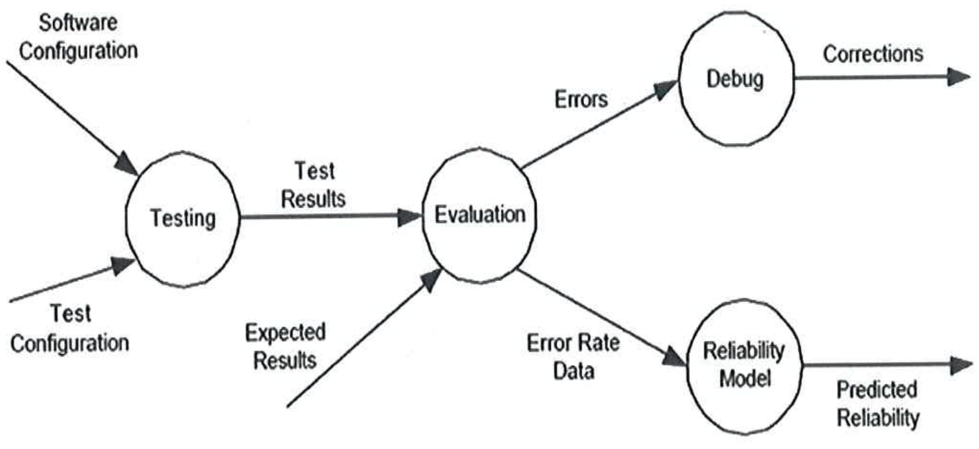


Figure 1. Testing Information Flow

Fig 5.1: Testing Information Flow

Chapter 6

METHODOLOGY

6.1 Understanding of ANDROID

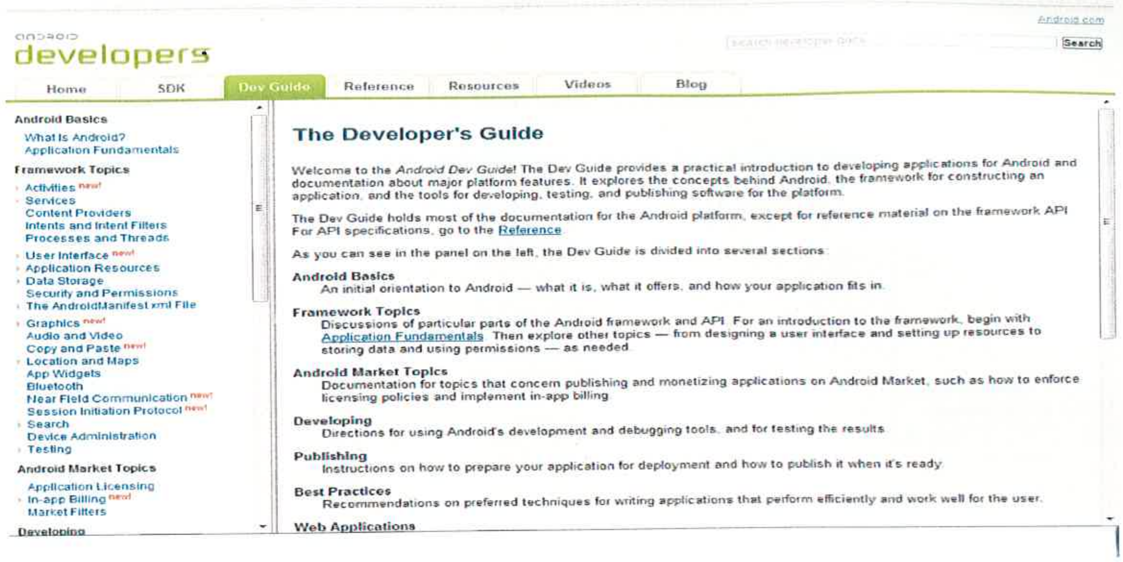
Android is an operating system for mobile devices such as "smart phones" and "tablet computers". Android is unique because Google is actively developing the platform but giving it away for free to hardware manufacturers and phone carriers who want to use Android on their devices. It was developed by the Open Handset Alliance led by Google. It is a group of 54 companies which includes hardware, software, and telecommunication companies called the Open Handset Alliance with the goal of contributing to Android development. Of particular note are the prominent mobile technology companies Motorola, HTC, T-Mobile, and Qualcomm etc...Most members also have the goal of making money from Android, either by selling phones, phone service, or mobile applications.

6.2 Advantages of Android

- Open Source
- Free
- Open API's
- It is currently the fastest growing phone platform in the world.
- It was reported that in Q4 2010 the Android operating system was the world's best-selling smart phone platform,
  dethroning Nokia's Symbian from the 10-year top position

Documentation of Android

1. Go to Android SDK folder → Docs folder → index.html



Android Basics
What Is Android?
Application Fundamentals
Framework Topics
Activities new!
Services
Content Providers
Intents and Intent Filters
Processes and Threads
User Interface new!
Application Resources
Data Storage
Security and Permissions
The AndroidManifest.xml File
Graphics new!
Audio and Video
Copy and Paste new!
Location and Maps
App Widgets
Bluetooth
Near Field Communication new!
Session Initiation Protocol new!
Search
Device Administration
Testing
Android Market Topics
Application Licensing
In-app Billing new!
Market Filters
Developing

The Developer's Guide

Welcome to the *Android Dev Guide*! The Dev Guide provides a practical introduction to developing applications for Android and documentation about major platform features. It explores the concepts behind Android, the framework for constructing an application, and the tools for developing, testing, and publishing software for the platform.

The Dev Guide holds most of the documentation for the Android platform, except for reference material on the framework API. For API specifications, go to the Reference.

As you can see in the panel on the left, the Dev Guide is divided into several sections:

Android Basics
An initial orientation to Android — what it is, what it offers, and how your application fits in.

Framework Topics
Discussions of particular parts of the Android framework and API. For an introduction to the framework, begin with Application Fundamentals. Then explore other topics — from designing a user interface and setting up resources to storing data and using permissions — as needed.

Android Market Topics
Documentation for topics that concern publishing and monetizing applications on Android Market, such as how to enforce licensing policies and implement in-app billing.

Developing
Directions for using Android's development and debugging tools, and for testing the results.

Publishing
Instructions on how to prepare your application for deployment and how to publish it when it's ready.

Best Practices
Recommendations on preferred techniques for writing applications that perform efficiently and work well for the user.

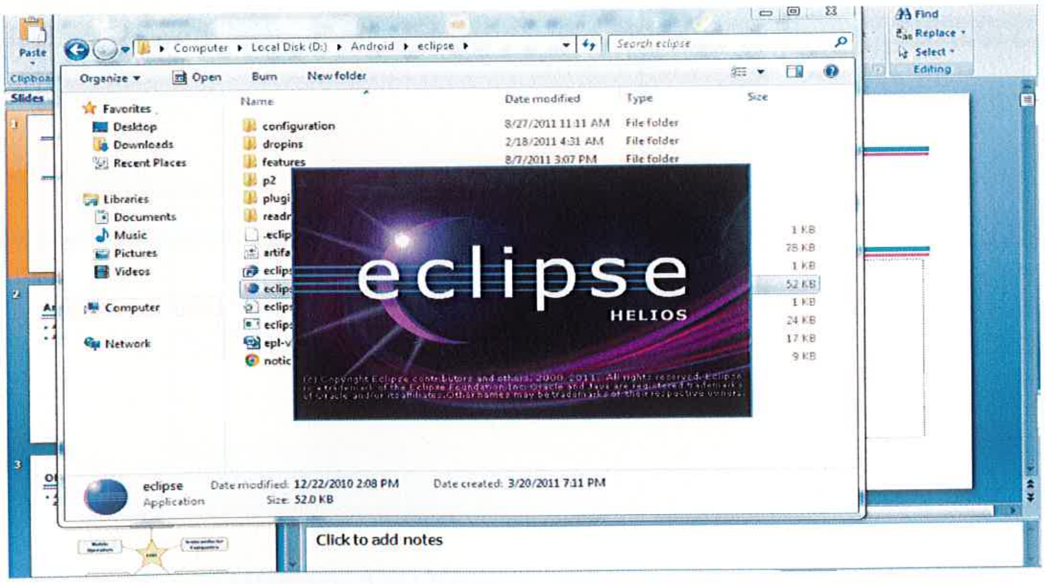
Web Applications

Snapshot1: Documentation of Android

Setting up workbench

Fire up/ start Eclipse by going to

i. Eclipse folder → eclipse (Application) and double click on it
ii. The Eclipse IDE starts as shown in below screen shot
iii. Once if Eclipse starts, it means to say that the JDK (Java Development Kit) has been installed successfully.
iv. If you couldn't fire up eclipse, then install JDK first.



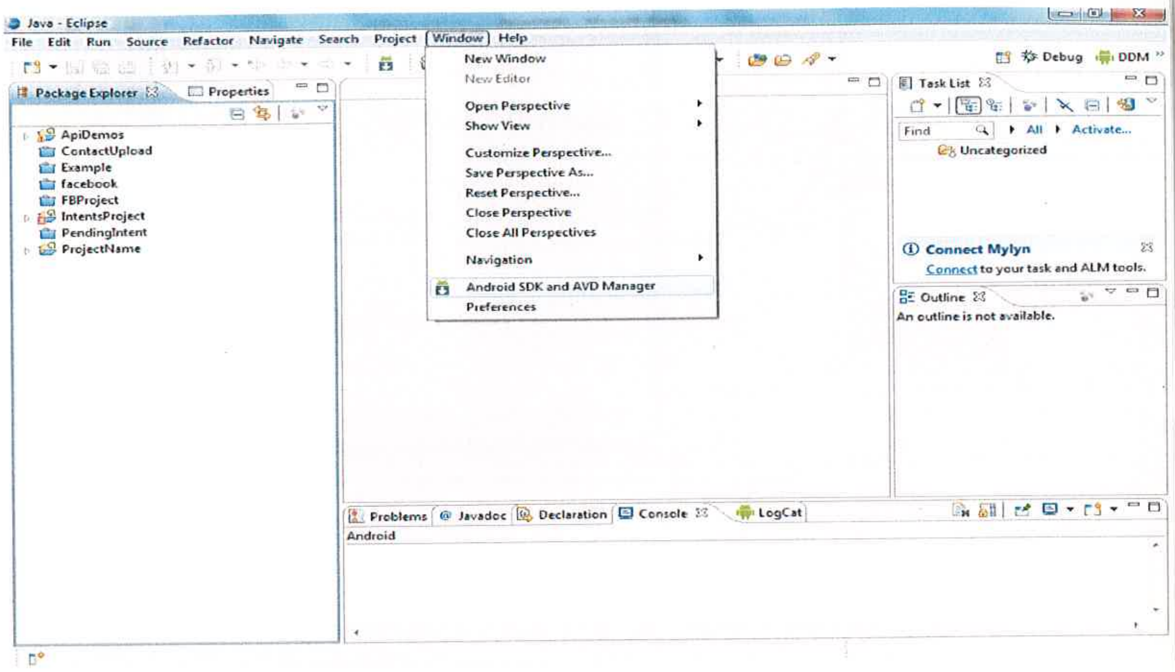
Name	Date modified	Type	Size
configuration	8/27/2011 11:11 AM	File folder	
dropins	2/18/2011 4:31 AM	File folder	
features	8/7/2011 3:07 PM	File folder	

eclipse Application Date modified: 12/22/2010 2:08 PM Date created: 3/20/2011 7:11 PM Size: 52.0 KB

Snapshot2: Setting up workbench

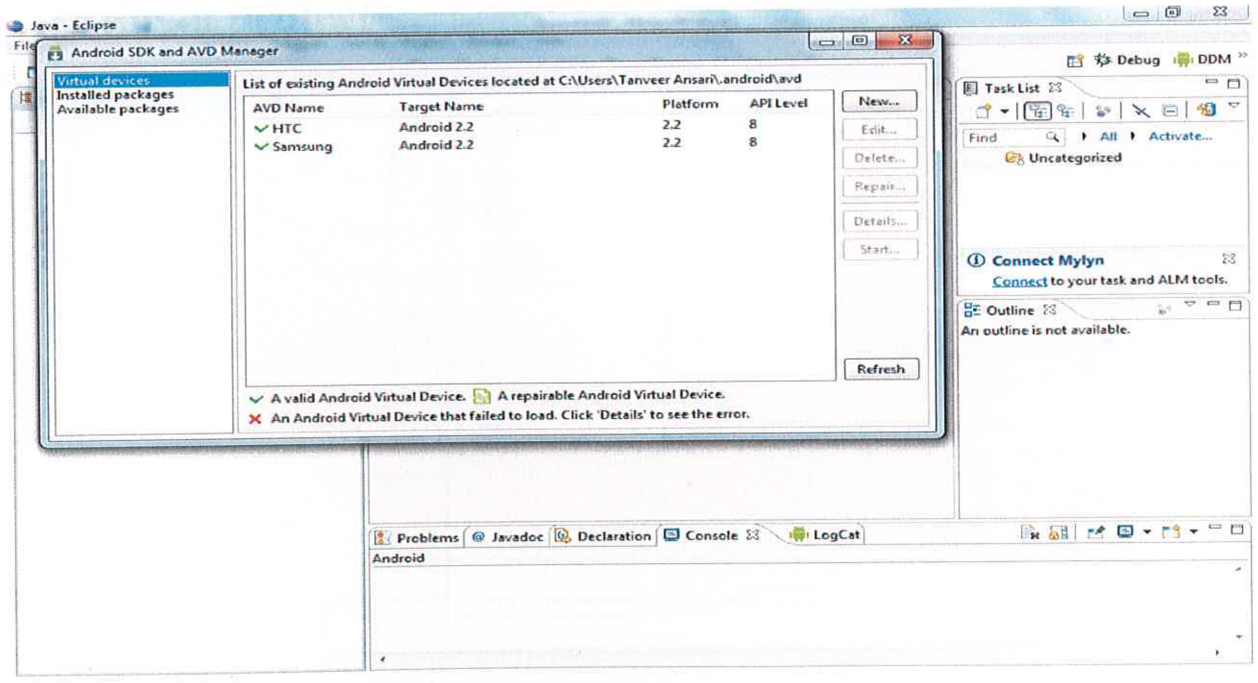
Creating AVD

i. Go to Window → Click on Android SDK and AVD Manager



Snapshot 3: Creating AVD

1) Click on New Button



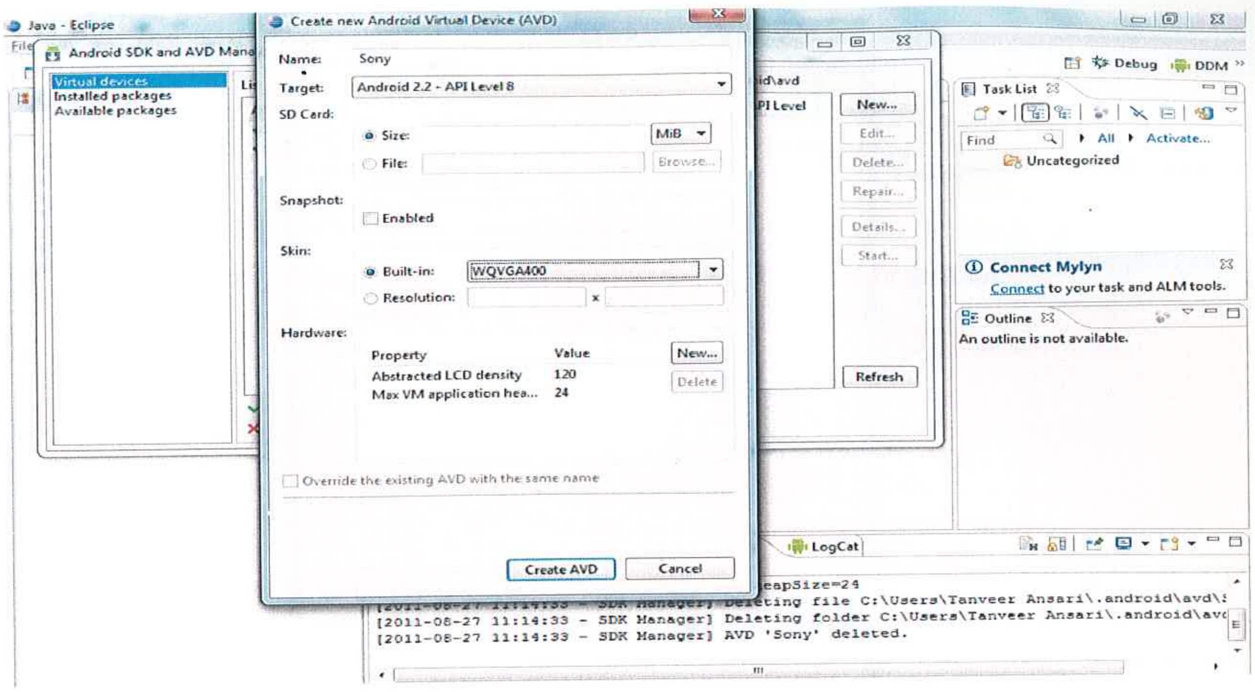
List of existing Android Virtual Devices located at C:\Users\Tanveer Ansari\.android\avd

AVD Name	Target Name	Platform	API Level
✓ HTC	Android 2.2	2.2	8
✓ Samsung	Android 2.2	2.2	8

✓ A valid Android Virtual Device. A repairable Android Virtual Device.
✗ An Android Virtual Device that failed to load. Click 'Details' to see the error.

Snapshot 4: AVD Manager

Enter the name and select the Target (platform) and skin→ Click on Create AVD. Then select the AVD created and click on Launch.


Create new Android Virtual Device (AVD)

Name: Sony
Target: Android 2.2 - API Level 8
SD Card:
Size: MiB
File: Browse...
Snapshot: Enabled
Skin:
Built-in: WQVGA400
Resolution: x
Hardware:

Property	Value
Abstracted LCD density	120
Max VM application hea...	24

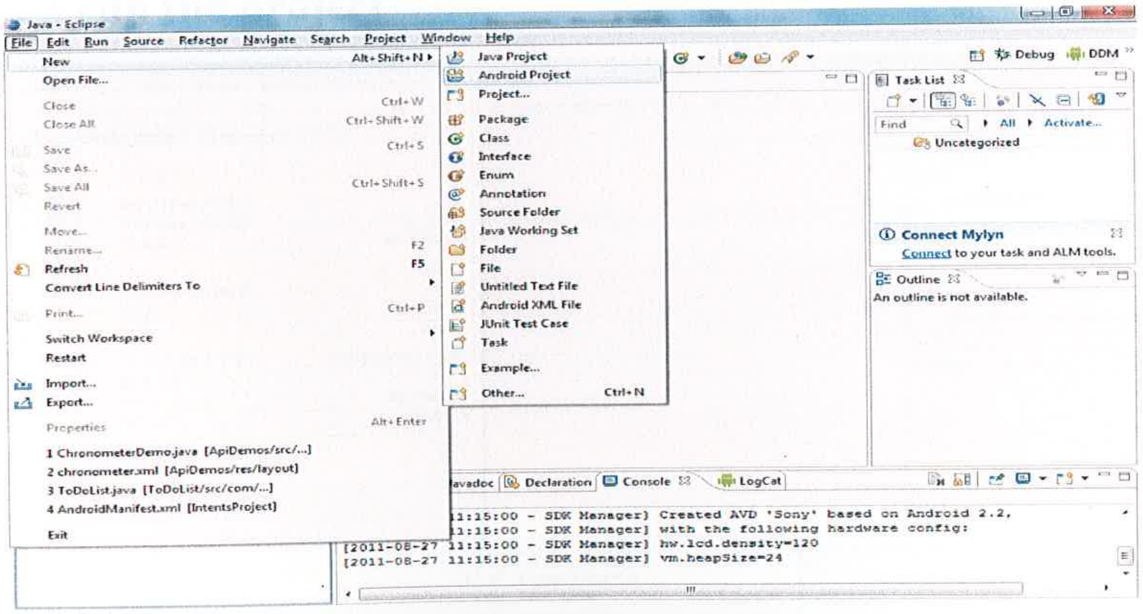
Override the existing AVD with the same name

Create AVD   Cancel

```
eapSize=24
[2011-08-27 11:14:33 - SDK Manager] Deleting file C:\Users\Tanveer Ansari\.android\avd\
[2011-08-27 11:14:33 - SDK Manager] Deleting folder C:\Users\Tanveer Ansari\.android\av
[2011-08-27 11:14:33 - SDK Manager] AVD 'Sony' deleted.
```

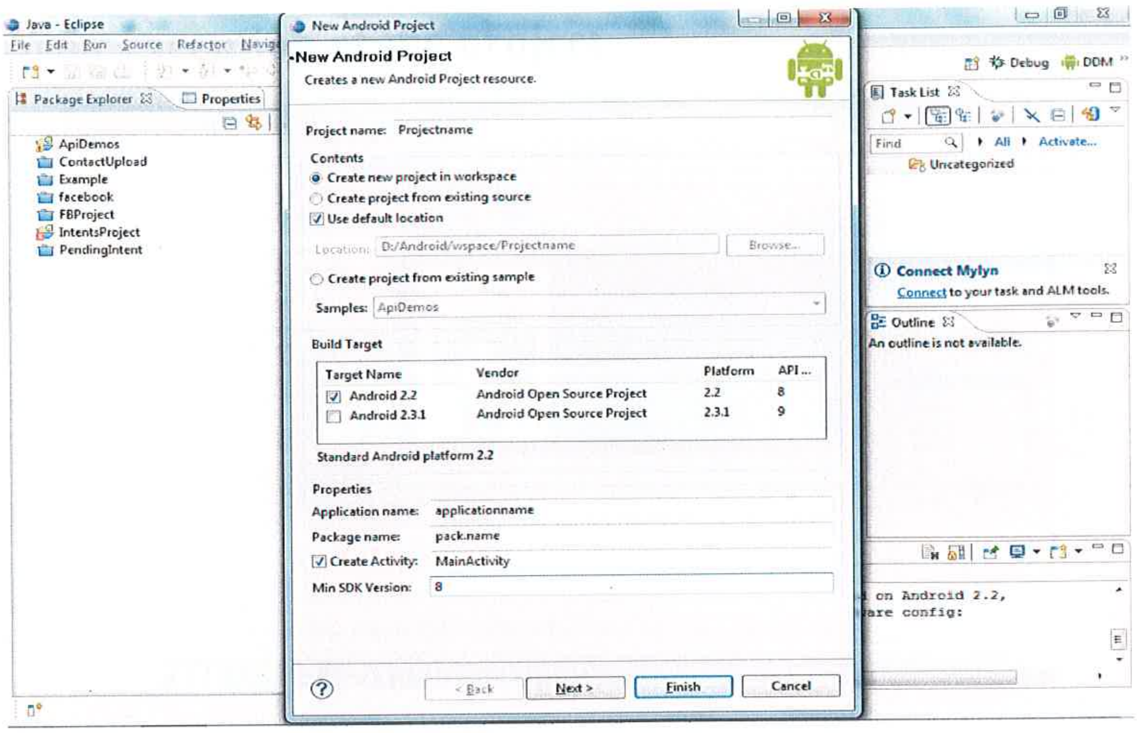
Snapshot 5: Create new AVD

Creating a project



```
11:15:00 - SDK Manager] Created AVD 'Sony' based on Android 2.2,
11:15:00 - SDK Manager] with the following hardware config:
[2011-08-27 11:15:00 - SDK Manager] hw.lcd.density=120
[2011-08-27 11:15:00 - SDK Manager] vm.heapSize=24
```

Snapshot 6: Create a Project


New Android Project

Creates a new Android Project resource.

Project name: Projectname

Contents
- (●) Create new project in workspace
- ( ) Create project from existing source
- [✓] Use default location

Location: D:/Android/wspace/Projectname

- ( ) Create project from existing sample

Samples: ApiDemos

Build Target

	Target Name	Vendor	Platform	API ...
[✓]	Android 2.2	Android Open Source Project	2.2	8
[ ]	Android 2.3.1	Android Open Source Project	2.3.1	9

Standard Android platform 2.2

Properties

Application name: applicationname

Package name: pack.name

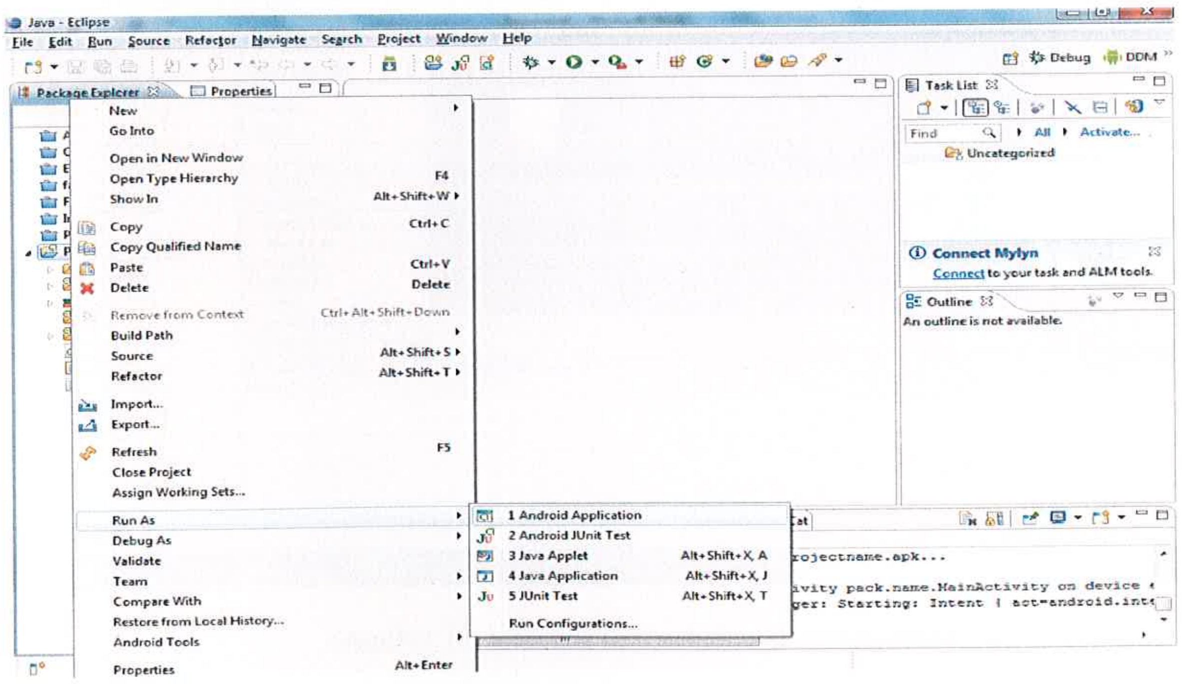
[✓] Create Activity: MainActivity

Min SDK Version: 8

```
on Android 2.2,
are config:
```

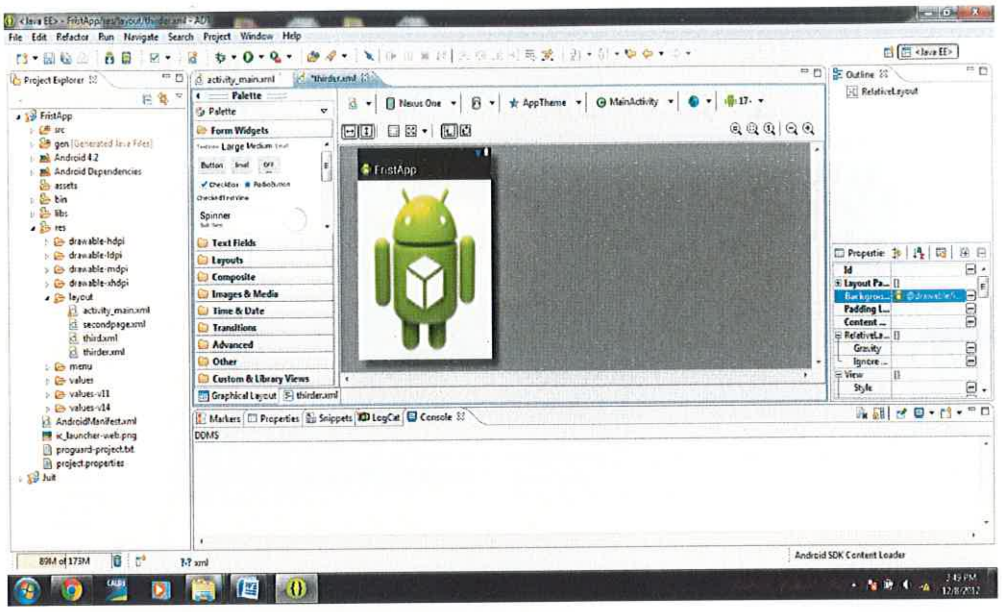
Snapshot 7: New Project

To run the project


```
rojectname.apk...
ivity pack.name.MainActivity on device
ger: Starting: Intent { act=android.int
```

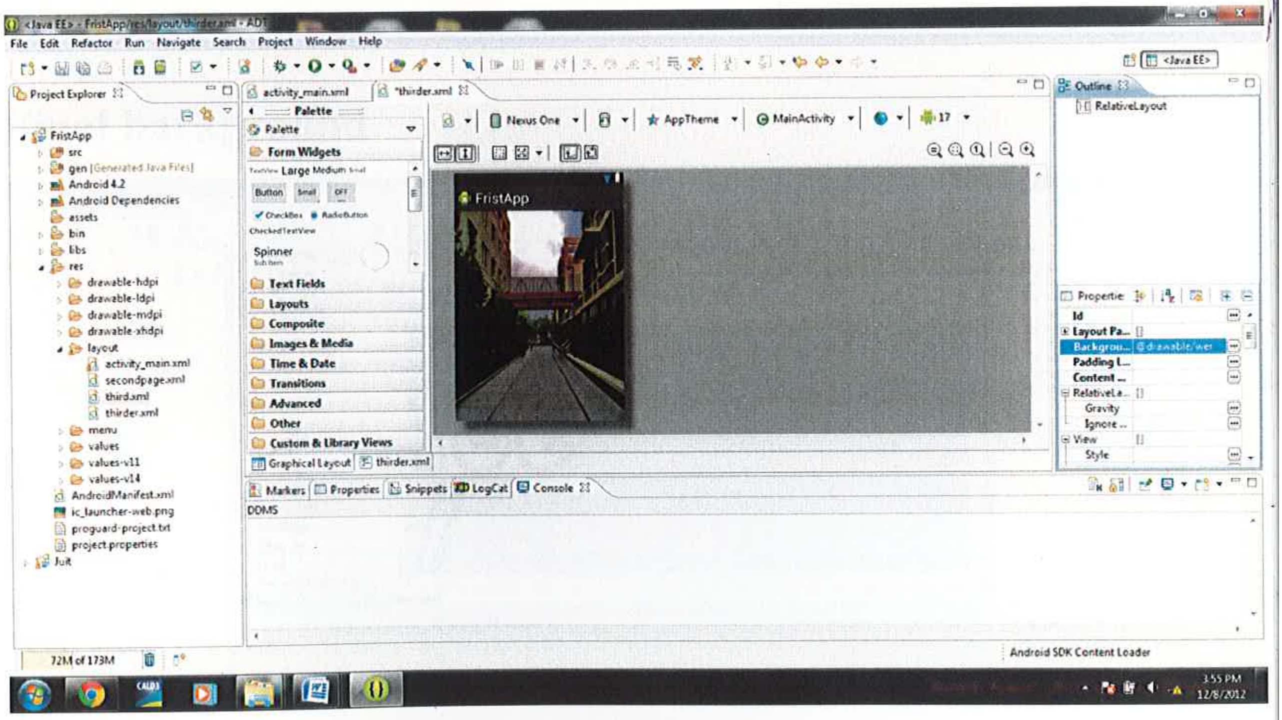
Snapshot 8: To run the project

After execution of the project



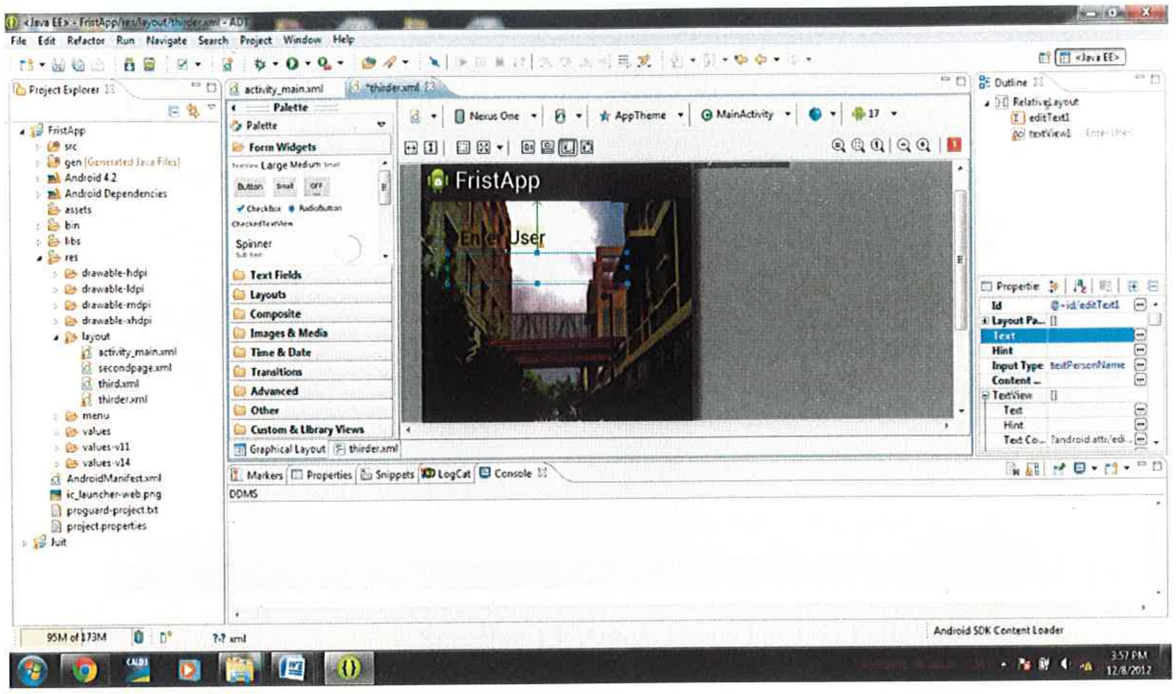
Snapshot 9: After execution of the project

Inserting the background



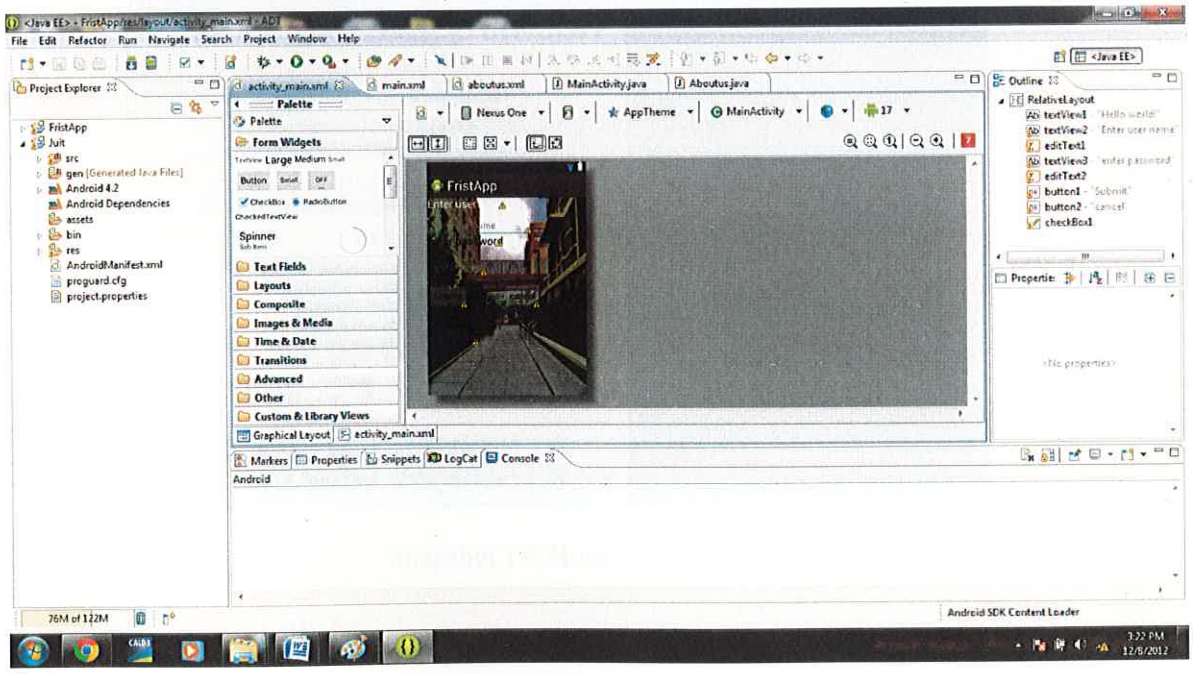
Snapshot 10: Inserting the background

Inserting text .



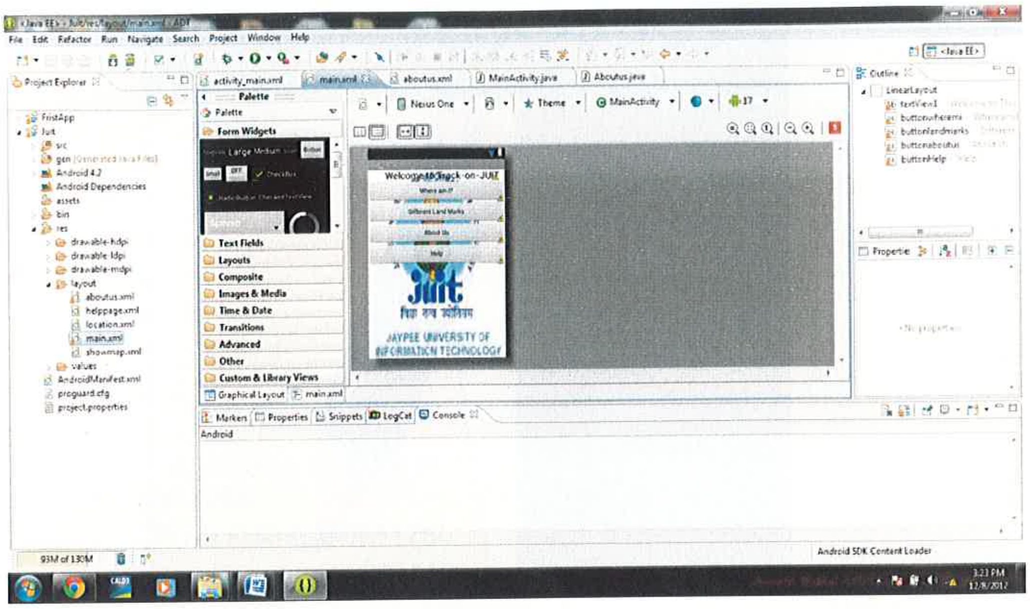
Snapshot 11: Inserting the text

Final background



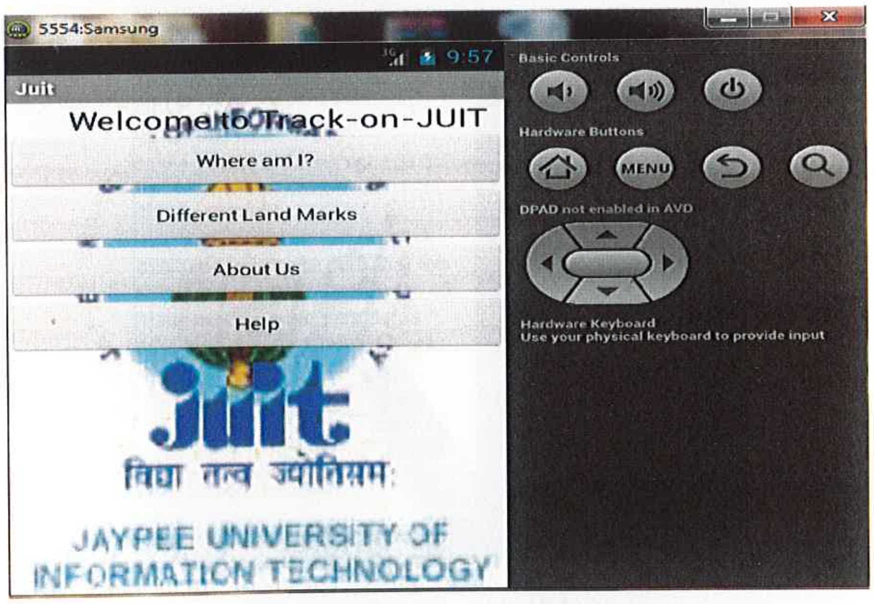
Snapshot 12: Final background

Actual Home Page on Eclipse



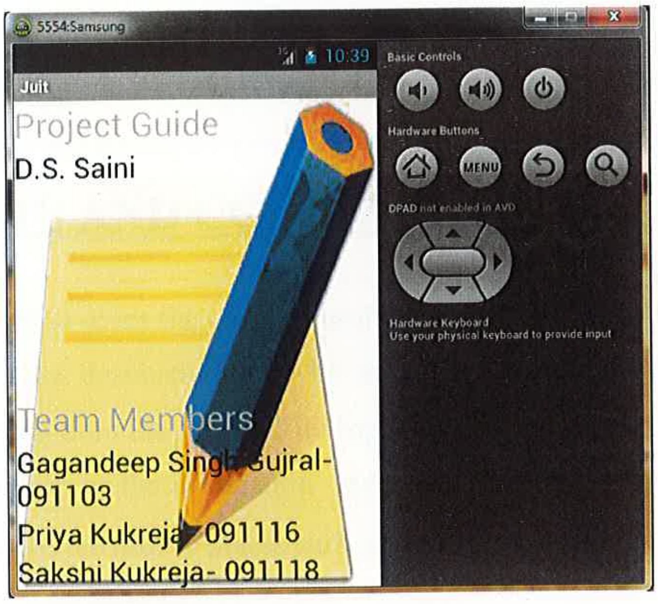
Snapshot 13: Actual Home Page on Eclipse

Home Page on Emulator



Snapshot 14: Home Page on Emulator

About Us option in our application



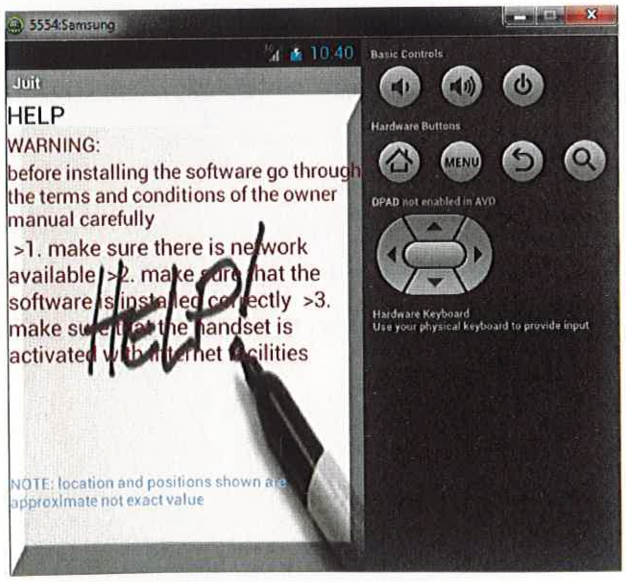
Juit

Project Guide
D.S. Saini

Team Members
Gagandeep Singh Gujral- 091103
Priya Kukreja- 091116
Sakshi Kukreja- 091118

Snapshot 15: About Us

Help option in our application



Juit

HELP
WARNING:
before installing the software go through the terms and conditions of the owner manual carefully
>1. make sure there is network available >2. make sure that the software is installed correctly >3. make sure that the handset is activated with internet facilities

NOTE: location and positions shown are approximate not exact value

Snapshot 16: Help Option

Chapter 7

CONCLUSION

7.1 RESULTS AND CONCLUSION

The application gives the desired result in the form of directions from a given place to another. The destination is to be mentioned by the user and he gets the direction at each step from the source. The application also keeps a track of the places visited while reaching to the destination. Moreover, the user also has an option of getting to know about the nearby places such as restaurant, ATM, hospitals etc.

7.1.1 MY LOCATION

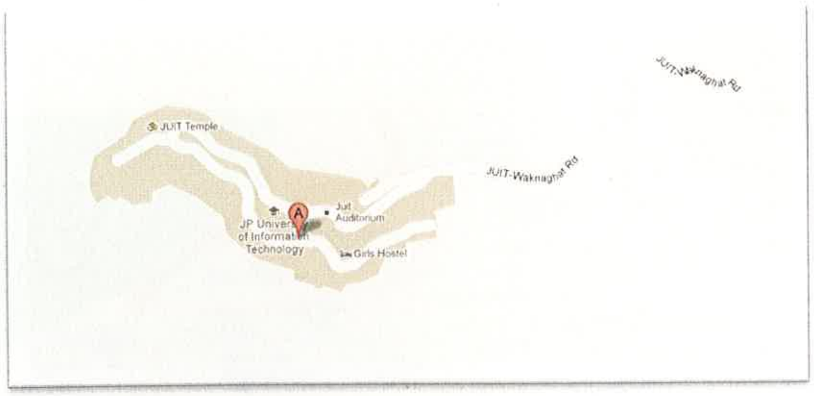


Figure 7.1: Show the current location of the user i.e. Jaypee University of Information Technology.

7.1.2 GET DIRECTION

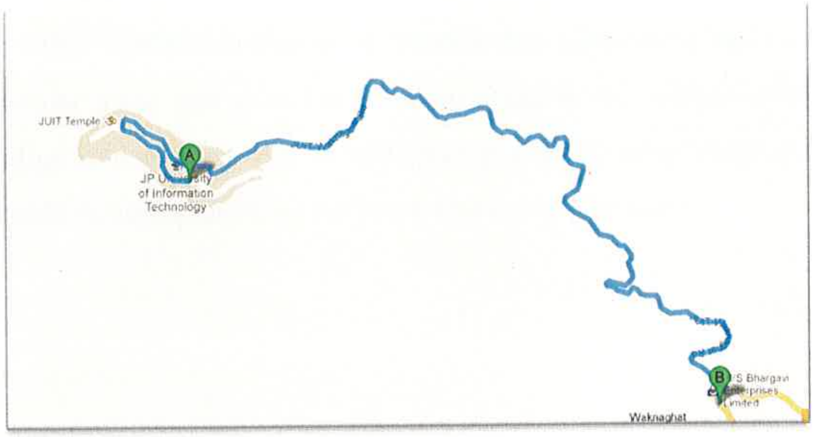

JUIT Temple
A
JP University of Information Technology
B
S Bhargavi Enterprises Limited
Waknaghat

Figure 7.2 Showing Directions from JUIT to Waknaghat

7.1.3 SHOW MAP

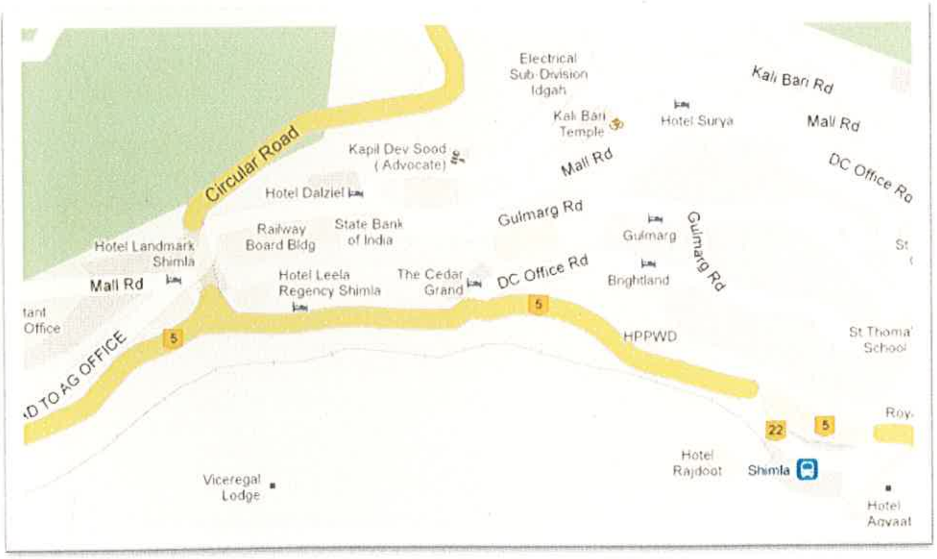


Figure 7.3: Showing the map of Mall Road Shimla. Also showing nearby places, hotels etc.

7.2 CONTRIBUTION OF THE PROJECT

There are many applications that exist in the market which provide the user with current location. But the application that we have built uses minimum resources when compared to other application that exist. Further this application can be extended to tag any particular place and write some memo about it. So, whenever the user is in particular radius of that place user would receive a notification about that place and the memo would remind user about the work that have to be done.

References

1. **LBIS (Developing Real Time Tracking Application)** - M. A. Labrador, Alfredo J., Pedro Wightman. (Pg: 4-16)
2. **A Scalable Architecture for Global Sensing and Monitoring,** IEEE Networks Magazine- A. J. Perez, M. A. Labrador, S. Barbeau. (Pg: 57-64, Issue Date: July- August 2010)
3. **Co-operative Location Update Algorithm for Mobiles in Next Generation Cellular Networks**-Samir K. Shah, Sirin Tekinay, Cem Saraydar.(Pg: 331)
4. **Foundation of Location Based Services-** Stefan Steiniger, Moritz Neun and Alistair Edwardes. (Volume-1, Pg: 12-21)
5. **What is Android?** Android Developers.
6. **"Philosophy and Goals"sss.** Android Open Source Project.
7. **Professional Android Application Development** - Reto Meier (Pg: 207)
8. **Hello, Android-** Ed Burnette
9. **Android Wireless Application Development**