Niroj Khadka

A Remote Desktop Application for Android

Helsinki Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Thesis

20 March 2013

Helsinki Metropolia University of Applied Sciences

Author(s) Title Number of Pages Date	Niroj Khadka A Remote Desktop Application for Android 47 pages 20 March 2013
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering, Information Systems
Instructor(s)	Peter Hjort, Senior Lecturer

The main goal of the project was to develop a remote control system for computers in an Android platform using Bluetooth 4.0 and Java. The objectives were that the applications would enable connection with the computer via Bluetooth from an Android phone and once connected it would give the user full control of the computer's mouse and keyboard.

The project consisted of two applications, server and client. For the client part in an Android device, the development was carried out using Eclipse Integrated Development Environment (IDE) and Android Software Development Kit (SDK) with an Android Development Tools (ADT) plug-in. A Samsung Galaxy S III phone was used for testing and running the client application. The development of the server part of the project in the computer was carried out the using the NetBeans IDE and Bluecove libraries.

Bluetooth can be the best option for communication between the server and the client as users do not have to pay for the data transfer as in the Internet use. However using Bluetooth as the communication media limits the scope of the project as the Bluetooth communication range is from 10 meters to 100 meters depending on the version of Bluetooth.

The developed client application in an Android phone can be used to control any Bluetooth-integrated computer running server application using the Bluetooth technology. The applications were well tested in different devices and the client application can run in phones running an Android operating system 2.0 and above 2.0. This project can be further modified by replacing Bluetooth with the Internet for communication. This would allow a user to access the computer from any part of the world.

Keywords	Android, Bluetooth, ADT, IDE, Bluecove, Remote Control

Contents

1 Introduction 1

2 Background to Android, Bluetooth and Java 2

2.1 Android 2
2.1.1 Android Architecture 3
2.1.2 Android Application Components 4
2.1.3 Application Life-cycles 5
2.1.4 Android Software Development Kit and Environment 8
2.1.5 Android User Interface and Touch Screen 9
2.1.6 Implementation of Touch in an Android Phones 11
2.2 Bluetooth 12
2.2.1 Bluetooth Specifications 14
2.2.2 Bluetooth Profile 15
2.2.3 Bluetooth Protocol 16
2.3 Java 17

3 Implementation of Android and Desktop Applications Development 19

3.1 Eclipse 19
3.1.1 Eclipse Architecture 20
3.1.2 Overview of Eclipse Plug-ins 22
3.2 NetBeans 22
3.3 Android Virtual Device 24
3.4 Project Components 25
3.4.1 Android Project Components 25
3.4.2 Java Project Components 26

4 Application Requirements and Description 27

4.1 Client Application Requirements and Descriptions 27
4.1.1 Android Application Background and Specifications 27
4.1.2 Architectural Design 28

4.2 Server Application Requirements and Descriptions 34
4.2.1 Desktop Application Background and Specification 34
4.2.2 Architectural Design 35
4.3 Sequence Diagram 38
4.4 Project Development Process 39

5 Testing of the client and server applications 41

6 Results 43

7 Discussion 44

8 Conclusion 47

References 48

1 Introduction

Mobile phones now a days are not just devices that are used for making calls. With the advancement of the technologies the ways mobile phones are used have changed. Mobile phones now a days can be used for various purposes. For example cameras, gaming devices, calculators, remote controls or sensors. As new technologies are being discovered day by day, tough competition is going between mobile companies to produce better phones. Those who cannot stand up to the expectations of users disappear slowly. The factor that makes mobile phones popular is the applications that allow users to do various surprising tasks. These applications have been among the important reasons for the growth of the mobile market. The revenue collected by phone applications in 2010 was 1.6 billion euros and it is expected to rise to 8.8 billion euros by 2015. [1]

In the battle of operating systems for mobiles, Android has emerged as a competitor or dominator to all existing mobile companies including Nokia and Apple. So choosing Android application development as a profession in today's competitive world seems to be secure. The Android platform, which collected 80 million euros in 2010 alone, is expected to collect 1.5 billion euros in 2015 [1]. Also the tools and environments used for application development are free and easily available from the Android developer's site. Good sets of instructions regarding the installation and developing applications are available on the site.

The goal of this final year project was to develop a remote desktop application allowing a user to have full control over the computer's mouse and the keyboard in an Android platform. The Android phone in the project was the client application acting as remote control for controlling the server desktop application. The Bluetooth 4.0 technology was used for the communication between the server and the client. The applications allow a user to control a computer without getting physically involved. The scope of the project can be extended by replacing the Bluetooth with the Internet as a communication mediator. This project then can be used to access the computer from any part of the world using the phone, provided the server and the client be in the same network.

2 Background to Android, Bluetooth and Java

2.1 Android

Android is one of today's most popular terms in the world of mobile operation systems. Android phones and Android applications are so popular among people that they have established a secured position among users in the world of mobile phones. Android is the software stack for a mobile device that includes an operating system, middleware and key applications .It is a Linux based operating system. Android Inc was founded in 2003 in Palo Alto, California, the United States in October, 2003 by Andy Rubin, Rich Miner and some other workers. In 2005 August Android Inc was acquired by Google with the key employees of Android Inc moving to Google. At Google Andy Rubin and his team developed a mobile device platform based on the Linux kernel known as Android and was introduced in 2007 when a group of industries came together to form Open Handset Alliance. Open Handset Alliance is an association of firms to develop open standards for mobile devices. Since then thousands of Android-based phones and applications had been launched to the market and have achieved tremendous success. [2,16-26]

In the market many programming languages are available such as C, C++, PHP, Flash and Java. Among these available languages Android uses Java as a native programming language. All the applications for Android are created using Java and then compiled using the javac compiler. Those compiled files are then converted into Dalvik Executable (.dex) file and ran in Dalvik Virtual Machine. Dalvik Virtual Machine is a virtual machine just like in Java which is used to test and run applications. The Android applications developed using Java use Android Software Development Kit (SDK) whereas the applications developed using C/C++ programming languages use Android Native Development Kit (NDK). [2,16-26]

The reason for Android to choose Java over other programming languages is because Java is an object oriented programming language, simple, secure, dynamic, portable, has high performance, supports multithreading and has automatic garbage collection. Besides Java C/C++, Flash can also be used to create Android applications. [2,16-26]

2.1.1 Android Architecture

The Android operating system can be divided into five major layers: Application, Application Framework, Libraries, Android Runtime and Linux kernel. These are the basic components that an Android application consists of. Applications is the top layer of the architecture. This is the layer where the core applications of a device such as phone calls, an email client, SMS program, calendar, maps, browser, contacts, and others can be found. These applications are written in Java and other languages. The Application Framework is the second layer of the architecture. This is the framework or the outline that a developer has to follow during application development. Developers are given full access to the same framework Application Programming Interface (API) as used by the Applications layer.This layer is just like a basic tool that can be used by a developer to develop much more complex applications. [3]

The third layer of the architecture is libraries. It consists of sets of C/C++ libraries used by various components of the Android system. These libraries are available to a developer for use through Application Framework. Some of these libraries are Surface Manager, Media Framework and WebKit. These libraries provide the facility to use in-built functions to carry out common tasks, or to reuse the functions for other purposes. The fourth layer in the architecture is the Android Runtime. It consists of sets of core libraries that are available in the Java language and Dalvik Virtual Machine. Dalvik Virtual Machine is software that behaves like a real device and is used for testing and running an application as its own process. [3]

The last layer of the architecture is Linux Kernel. Android depends on Linux version 2.6 for the core system services such as memory management, security settings, power management software and several drivers for hardware, file system access, networking and inter-process-communication. The Kernel also acts as an abstraction layer between hardware and the rest of the software stack. [3] Figure 1 shows the above discussed layers of the Android operating system.

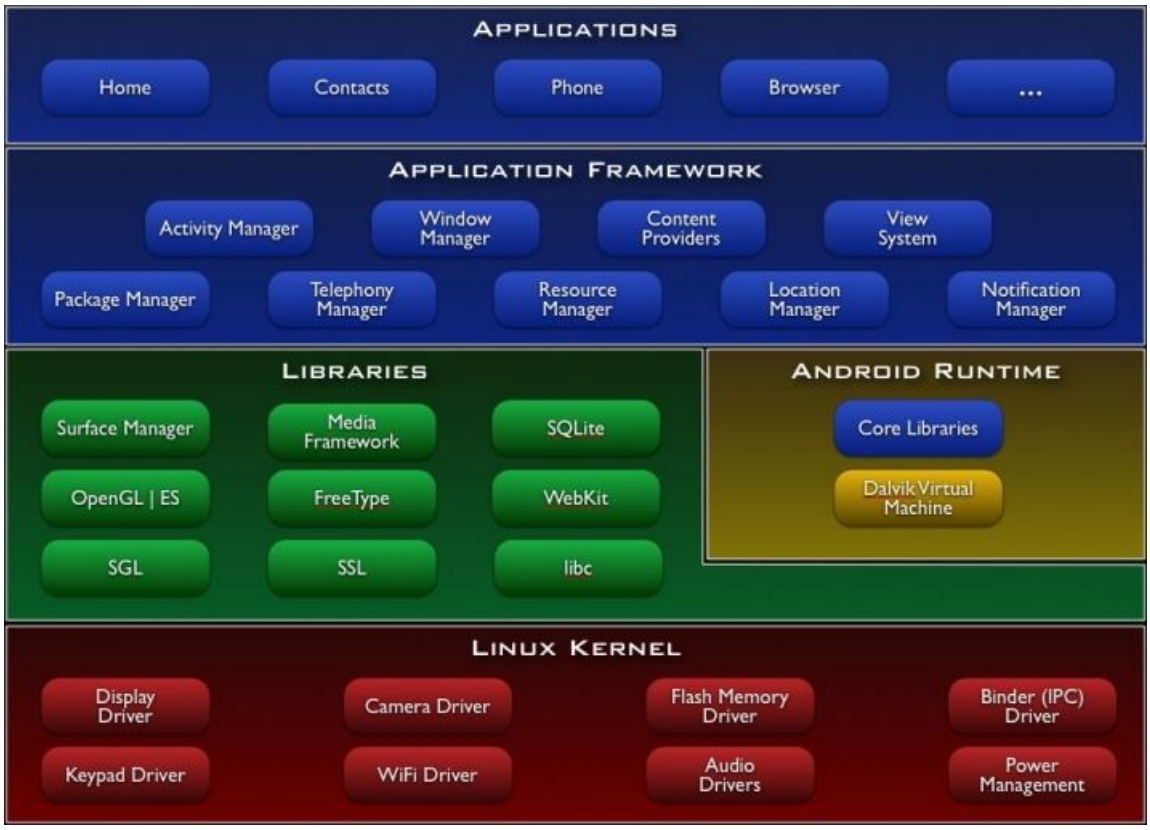

Figure 1. Android Operating System‘s Architecture. Copied from Android Architecture [3]

As shown in figure 1, the five layers are the basic components required to form the Android operating system. Each layer has its own function and groups various programs together to run specific functions of the operating system.

2.1.2 Android Application Components

Android application components are the basic building blocks required to develop an Android application. Through these components a system can have a connection with an application. The integral building blocks are Activities, Services, Content Provider and Broadcast Receiver. Each of these components has its own role and is used to start an application or connect one application with the other application. The first component is Activities which is also the starting point of an application. As single activity can be an application or a single application can sometimes contain many activities. Each activity represents a single screen and has a user interface. This component interacts with a user responding to the events. For example a contact application can have one activity for calling people, second activity for viewing people and third for sending a Short Message Service (SMS). The second component is Services which represents the background process without a user interface. It also has its own life cy-

cle. Generally operations taking a long time such as fetching data over the Internet, loading games or playing music can be done using the Services. This component enables one to perform different operations even when some other applications are active. [4]

The third component of an application is the Content Provider. This component is created in order to share data between the Activities and Services that are stored in devices by any means. Moreover using this component a user can query and even modify the data For example contact names in the phones are available to all applications. The last component is the Broadcast Receiver which acts as a system's event listener. This component creates alerts or broadcasts whenever it detects a change in the system such as battery low or screen turn off. These components can also be manually included in an application to create an alert to a user such as to notify that the data has been downloaded or a connection established to a network. Among all these components, three components, Activities, Services and Broadcast Receiver are activated using the Intent Service. The Intent is just an asynchronous message to an application about the action to be performed. Content Provider is implemented using Content Resolver. It can be activated by requesting permission in the MANIFEST file of the Android project. [4]

2.1.3 Android Application Life-cycles

The activity in Android is responsible for creating a window of an application. An application can have any number of activities. An activity in an Android application has its own life cycle. An activity lifecycle has the starting and the ending state with many other states in between. When an activity lifecycle method is called, there is a change in the state of an application which is handled by the application components. Android manages its resources strictly in order to keep a device in the normal working state. Any processes can be killed or paused without notifying to free the memory for higher priority applications. An activity stack in Android ensures that the new launched activity is always at the top. Any previously launched activity will be below the current one and will not come to the top until the newly launched activity finishes. An Android application has its own process and its own instance in Dalvik. Every Android application saves its state when launched, so that when an application is resumed, it will acquire its previous state. [5;10]

An Android activity can be in four states. The states are the active/running state, pause and resume state, stop and restart state and finally dead/destroy state. The ac-

tive/running state is the state where an activity has been started. An activity is currently active and is always at the foreground or at the top of the activity stack. The highest priority in the stack is always the currently running stack. Such an activity is only killed if it tries to cause an abnormal error to a device by occupying the excess memory. During the pause and resume state an activity is not active but is partially visible. This state is usually caused when a newly launched activity does not occupy the whole screen or is semi-transparent. This activity is still alive and has the second highest priority in the activity stack. An activity gets killed by Android if sufficient memory is not available for the application to be in other states. An activity resumes with the previous state when the semi-transparent activity finishes. [6;7;10]

The stop and restart state is the state where an activity is completely invisible in the view. Another activity completely obstructs its view. The activity is still alive and does not lose its progress. In this state an activity has the lowest priority in the activity stack and can be killed by the operating system just to provide memory to the higher priority activity. If a user returns to an application by any means, then the activity restarts. The dead/destroy is the final state of an activity. An activity in Android can be manually destroyed by a user by pressing the back button or by a system if required. An activity will try to reinstate its previous state but is usually destroyed by the Android system in order to free memory for other activities. All these all states form a lifecycle for an activity in Android. [7;8;9] Figure 2 shows how the lifecycle of an Android activity occurs.

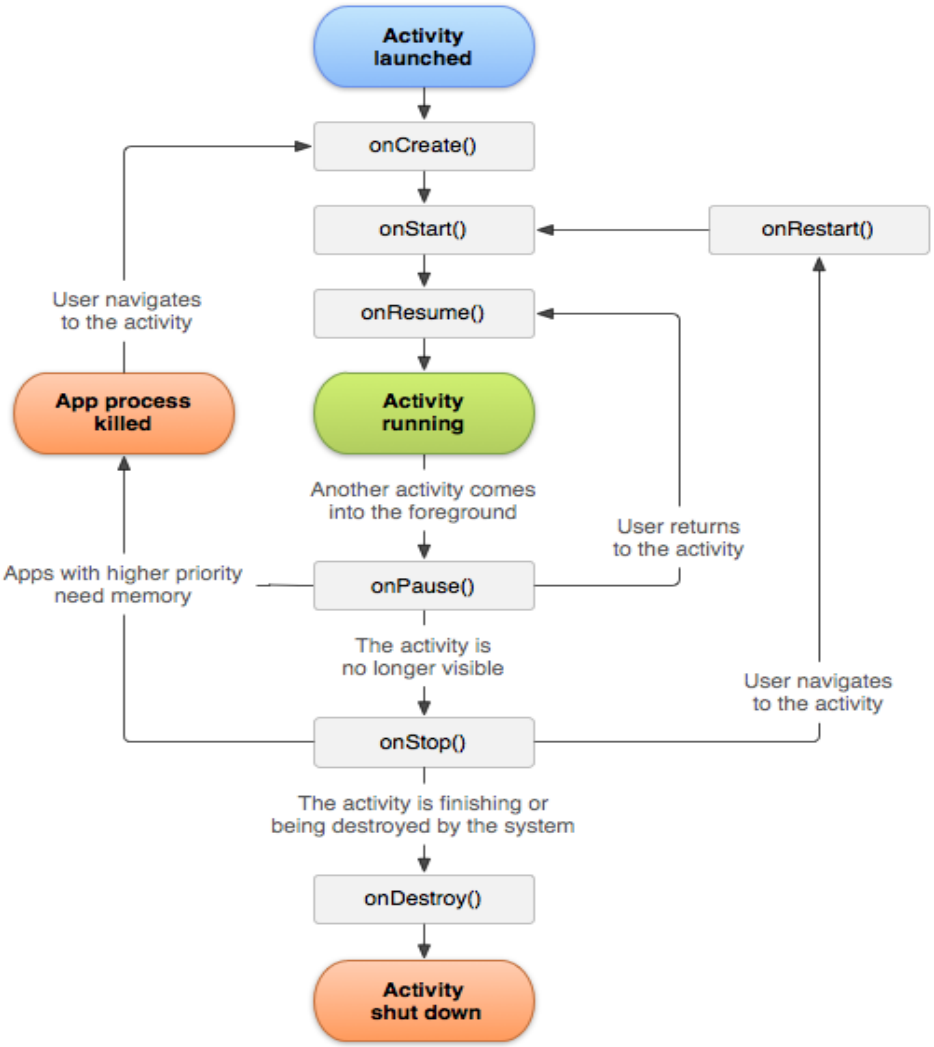

Figure 2. Lifecycle of an Android activity. Copied from Activity [5]

Figure 2 shows how an activity is started, paused, resumed and destroyed in its lifecycle. Android provides methods that can be called to handle the changes when a state in an activity changes. As the activity is started, the OnCeate() method is called. This method initiates an activity and performs the tasks such as creating a view, reading layouts, initializing variables or binding data. Then the OnStart() method is called, which makes an activity visible .An activity is still inactive during the OnStart() call. An activity becomes visible in the view when the OnResume() method is called. During this call an activity becomes active and users can interact with an activity. [5;10]

An activity is at the top of the activity stack during the OnResume() call. If the current activity is sent to the background by any means, then the OnPause() method is called. There are two possible scenarios during this call. An activity is either destroyed or resumed by a user. An activity can be destroyed by the operating system to free the

memory. In this case an activity restarts from the OnCreate() method with the BundlesavedInstanceState parameter which store the previous state of an activity. The next scenario would be to close the new activity and resume an activity from the OnResume() method. [5;10]

If none of the above mentioned cases occur, an activity is passed to the OnStop() method. This happens when a user has pressed the back button or a new activity has started that completely covers the previous one. There are three possible scenarios during this call. The system can kill an activity to free the resources and the cycle is started again from OnCreate(). Another case is that an activity is resumed from OnStart() when a user brings an activity from the background to the foreground. The final case is the OnDestroy() method is called and an activity is destroyed. This happens when an application is terminated by a user or the system. The entire life cycle of an activity is in between the OnCreate() and OnDestroy() methods. An activity is visible during the OnStart() and OnStop() methods. Finally an activity is in the foreground between the OnPasue() and OnResume() methods. [5;10]

2.1.4 Android Software Development Kit and Environment

The Android SDK is a collection of software development kits used to develop applications in the Android platform. The Android SDK is available for free to download and use from the official Android developers' website. The SDK is available for all operating systems. It can be used alone to develop an application with a command prompt or can be used with IDEs such as Eclipse and NetBeans. The IDE provides a graphical interface for easier and faster development of an application. The Android SDK consists of the following features:

- Required libraries
- Debugger
- An emulator
- Documentation for the Android application program interfaces (APIs)
- Sample source code
- Tutorials for the Android operating System [11].

For each release of a new version of Android, a corresponding SDK is released. A separate SDK is available for a separate version of Android. In order to use the latest feature in Android, developers need to install the latest SDK depending on the phone's Android version. To develop an application Android provides a custom plug-in for

Eclipse called ADT. ADT provides an integrated environment for development of an application by extending the abilities of Eclipse. This helps developers to create an Android project, design an application user interface, debug an application and export and import an application package. The native language used for the Android application development is Java for which Java Development Kit (JDK) is needed. JDK can be downloaded for free from official Sun Microsystems website. All the above mentioned plug-ins and IDEs are available for free downloads, and complete sets of instructions on installation are available on the Android official website. [11;12]

2.1.5 Android User Interface and Touch Screen

With more and more phone manufacturing companies evolving today, there has always been competition between vendors regarding the technology provided to customers. The factors affecting the popularity of phones are their looks, performance, user experience, price and applications. One of the important factors affecting user experience is a user interface. The user interface is a component of an application that a user uses to interact with a device. Designing a better user interface and providing better technology to interact with the interface has been an essential task in designing phones. One of the ways of interacting with a user interface is using the touch screen. Touch screens can be used to feed inputs and to produce outputs. A simple and more intuitive interaction is provided to a user allowing direct and natural interaction with the devices. [13]

The touch screen is an electronic visual display with which a user interacts by touching the screen. A human finger or a touch pen can be used to interact with the screen. It is fast, simple, easy and natural compared to other input devices, so many companies manufacturing electronic devices have acquired the technology, including Android. Several of issues must be kept in mind before designing User Interface (UI) for touch screens. [14] Some of the basic principles to be followed during designing a UI for the touch screen are as follows:

- The UI should allow immediate access to the components.
- Gestures such as a tap, click or flicker should kept simple and smart.
- The UI should be designed for real world use (considering the size of finger)
- The UI should be kept simple and clear [15]

Designing a UI in Android has its own techniques and components. Android uses Views, View Groups and Activities for designing a UI. Views are the class containing all

the basic building blocks or component for a UI design. They contain all the UI controls, layouts and widgets. It is not always necessary to use the components provided by the Views class for the UI design. A new view or components can be created by subclassing the existing class. Views can be comprised of multiple views known as View Groups. A View Group itself is a subclass of Views and can contain multiple child Views. An activity is the screen displayed to the user. To display the UI, View is assigned to an activity. Generally XML files are used to design a UI in Android applications. An XML file must contain a root layout and this root layout can contain many nested layouts and view to design the necessary UI. Figure 3 shows the different types of View components used in Android. [16;17]

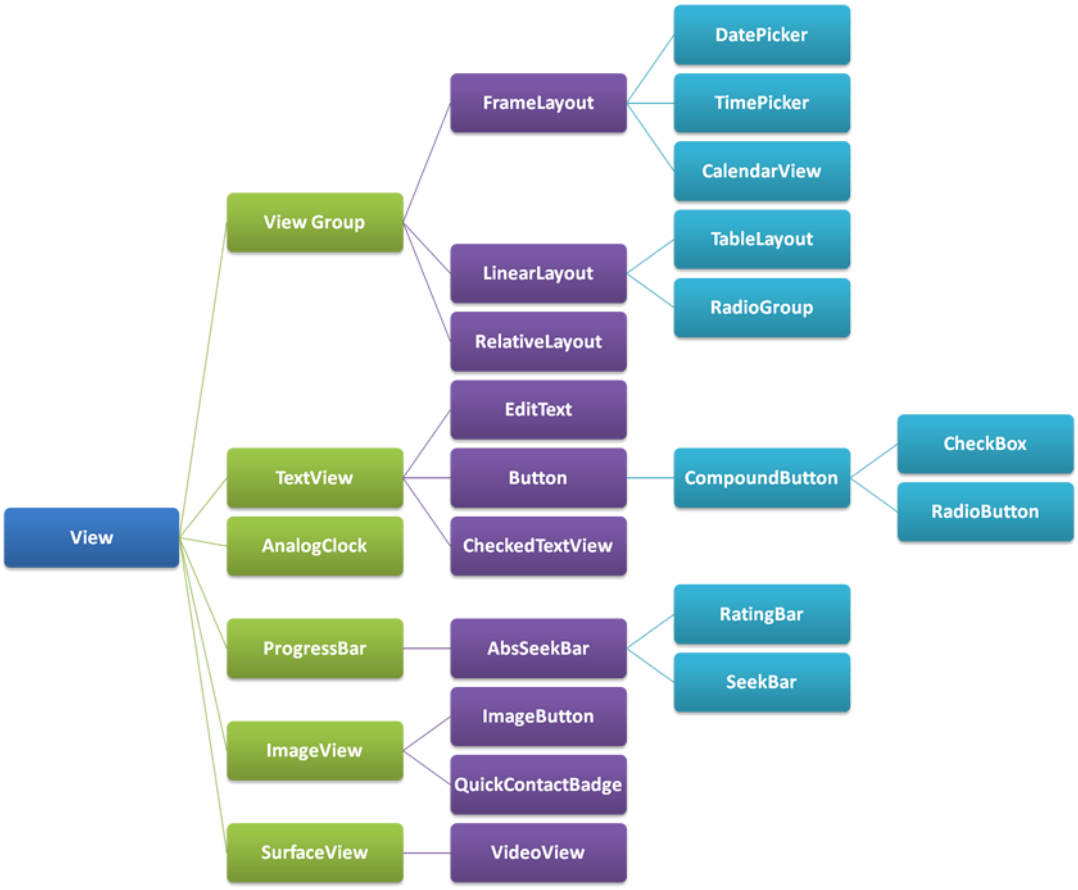

Figure 3. View class and its sub classes. Copied from the Procedural vs. Declarative Design of User Interfaces [17]

Figure 3 shows the View roots class and its subroots. View class is sub divided in View Group and other components. These View Group and the other components are further divided into layouts and other widgets. They are the building components of a UI. They

are all usually included in the Extensible Markup Language (XML) file and used in an Activity. Figure 4 shows a sample user interface in Android.

Figure 4. Different components of the View class. Copied from Building a simple Android app [18]

Figure 4 illustrates different components of the View used for designing a user interface. The whole figure represents the activity of an application. This is the part of an application that appears on the screen. The figure consists of different elements of View such as TextView, ImageView and Button. All of these components are included inside a root layout defined in the XML file. An application can contain any layout, such a linear layout, relative layout, List View or Grid View.

2.1.6 Implementation of Touch in an Android Phone

Most Android phones use the touch screen as the primary medium to interact with a user. The screen of a phone is used to interact with a user interface of an application. Android supports both a single touch and multi touch. Touch events in an Android phone are supported by view, custom views and an activity. Touches are interpreted in an Android phone in the form of gestures, such as swipe, flicker, scroll, fling or drag. Android has provided proper classes and methods to handle the touch events. Some of

the classes and the methods that handle touch are MotionEvent, TouchEvent, GestureDetector and VelocityTracker. [19,591]

To detect a touch in Android the MotionEvent class is passed to a view using the OnTouchEvent method. The MotionEvent class contains information such as coordinates of the touched point, the number of pointers, size and pressure of each pointer. The OnTocuhEvent method then can be used to use the points. The MotionEvent class provides some constants that can be used to determine the action caused by a touch. ACTION_DOWN detects a new touch, ACTION_MOVE detects finger moving on the screen and ACTION_UP detects the removal of the finger from the screen. The two constants ACTION_POINTER_DOWN for pointer down and ACTION_POINTER_UP for pointer up are used for detecting a multitouch. [19,592]

The two other classes used to detect a touch are GestureDetector and VelocityTracker. GestureDetector must use GestureOverlayView in the view to detect gestures. The class detects gestures such as swipes and flings. This class can be used to draw figures and write by a moving finger on the screen. Likewise the VelocityTracker class can be used to track the velocity of the touch events. [19,604-619] This project also uses the touch feature of the phone. The touch screen and keyboard of the phone acts as a mouse and keyboard for the server simultaneously.

2.2 Bluetooth

With the world making remarkable progress in technology humans are knowingly or unknowingly associated with the wireless technology. There are no such fields where the wireless technology has not been used. The wireless technology is especially designed for communication and data transfer. One of such wireless technologies that has been an important aspect of communication is Bluetooth.

Bluetooth is a wireless technology capable of data and voice transmission. Bluetooth is designed for short range transmission from 5 meters to 100 meters. The transfer range of Bluetooth depends on its version. Bluetooth uses short wavelength radio transmission for communication in the Industrial, Scientific and Medical radio bands (ISM) band from 2400–2480 MHz. Bluetooth was invented in 1994 by Ericson as a wireless replacement for data cables. The name Bluetooth comes from the 10th-century king Harald I of Denmark. Later a group called Bluetooth Special Interest Group (SIG) was formed to manage the development of the Bluetooth technology. The group now con-

sists of renowned companies such as Ericson, Nokia, Lenovo, Microsoft, Motorola, Intel and Toshiba with about 17000 member companies involved in it. [20;21,3-4]

Bluetooth devices require less power to operate and hence they are affordable. They are also completely automatic. Any Bluetooth device can sense another Bluetooth device in the surrounding if it is within the detection range. A master Bluetooth device can make communication with seven devices at a time. Also Bluetooth can be used to form a Personal Area Network with a high level of security. A Bluetooth device usually ignores any interference from other wireless devices as it uses frequency hopping. Frequency hopping resists radio transmission interference by ignoring the use of crowded frequencies in a fast and short sequence. With the implementation of Bluetooth in an electronic device a user does not have to be physically involved to do certain tasks such as listening to music, making voice call and downloading any form of data. Also the availability and the mobility feature of Bluetooth makes it even better over other wireless devices for use. [20;21,3-4]

Among the different versions of Bluetooth, the latest version available is the Bluetooth version 4.0. The main feature of Bluetooth 4.0 is its low energy technology allowing a Bluetooth to run for years on button-cell batteries. Also Bluetooth 4.0 is cheap with an extended range and an ultra-low peak. Bluetooth 4.0 is especially designed for Bluetooth smart devices. Some of Bluetooth smart devices are heart-rate monitors, blood-glucose meters, door security sensors and smart watches. Also most smart phones, such as Iphone 4s and Samsung Galaxy III, have Bluetooth 4.0 in them. Except for Bluetooth version 1.1 and 1.2, the rest of the versions of Bluetooth are capable of transmitting data along a range more than 10 meters. Bluetooth versions 2, 2.1, 3.0 and 4 .0 are capable of transmitting data within a range of 100 meters. [20;21,3-4]

Bluetooth on one hand has advantages while on the other hand also disadvantages. Bluetooth has very low data transfer compared to other wireless devices. The transfer rate is only one megabyte per second while infrared has four megabytes per second and Wi-Fi has even higher speeds. Also the transfer range of Bluetooth is very low compared to Wi-Fi. Bluetooth uses radio waves to transfer data, which makes it easier for hacker to break into the system. Bluetooth uses the battery of a device in order to operate and hence it can drain or reduce the battery life easily. [20;21,3-4]

The development process of the Bluetooth technology is still going on. With the advancement in the technology some day Bluetooth might be an alternative to all the

wireless devices. This thesis project also uses Bluetooth to make communication between a mobile and a computer. The documentation of Bluetooth provides information about the Bluetooth specifications and the Bluetooth profiles. The Bluetooth specifications give information on how a technology works and the Bluetooth profiles about how to use the core technology to build a device. [20;21,3-4]

2.2.1 Bluetooth Specifications:

The Bluetooth specifications provide all the necessary information to all the Bluetooth manufacturers to ensure the compatibility of a Bluetooth device. The Bluetooth specifications available in the Bluetooth documentation are Radio specification, Baseband, Link Manager Protocol, Logical Link Protocol and Adaptation Control (L2CAP) and Service Discovery Protocol (SDP). [22;23]

The Radio specification is the lower layer defined in the Bluetooth specifications. A Bluetooth transceiver operates in a 24 GHz ISM band. The specification defines requirements for the transceiver operating under this band. The specifications are defined to ensure the compatibility of use of radio in Bluetooth and to ensure quality. Not all countries use the same band frequency for Bluetooth. To compensate this issue, the frequency hopping algorithm has been implemented in the countries with a different band frequency for Bluetooth. The Baseband provides information about the Bluetooth link control layer. This layer lies on top of the radio layer. The main purpose of the baseband is to manage the physical channel and links. It can be used as a link controller to manage synchronous connection oriented (SOC) and asynchronous connectionless (ACL) physical links. [22;23]

The Link Manager Protocol is used for link setup, control and security. It interprets and filters messages, hence ensuring authentication and encryptions. The Link Manager uses the Link Controller to communicate with other Link Manager via the Link Manager Protocol. The L2CAP layer is the layer above the baseband protocol. It acts as a barrier between the low level protocol layer and upper layer protocol. Connection oriented and connectionless data services are provided by the Logical Link Protocol and Adaptation Control to the upper layer protocol. The SDP provides a medium to discover the Bluetooth services. The SDP is located on top of the L2CAP. The Service discovery in Bluetooth is different compared to the service discovery in a traditional network-based environment as the services in Bluetooth change with the device motion. [22;23]

2.2.2 Bluetooth Profiles

The Bluetooth profiles, which are managed by SIG, define how to use the core technology to build a device. It helps a Bluetooth device to work with or use itself with other parts of a system. If a Bluetooth device is to function properly, then it must work smoothly and easily with other devices. A device can use the Bluetooth technology if a device is compatible with the subsets of the Bluetooth profiles. To be compatible with the subsets of the Bluetooth profiles a device using the Bluetooth technology must use the desired services provided by a Bluetooth device. [24;25]

The Bluetooth profiles depend on the Bluetooth specifications and may use features provided by Bluetooth specifications. Different core specifications are available for different versions of Bluetooth. It is not necessary that a specific profile should follow a specific specification. A Bluetooth profile can depend on the core specifications of a single or multiple versions of Bluetooth. The Profiles define how the Bluetooth technology is used for a device. So the specifications provided by the Bluetooth profiles must be followed by vendors in order to let the devices use Bluetooth as desired. A list of some of the Bluetooth profiles provided by SIG is given bellow.

- Advanced Audio Distribution Profile (A2DP)
- Audio/Video Remote Control Profile (AVRCP)
- File Transfer Profile (FTP)
- Generic Audio/Video Distribution Profile (GAVDP)
- Human Interface Device Profile (HID)
- LAN Access Profile (LAP)
- Message Access Profile (MAP)
- Personal Area Networking Profile (PAN)
- Serial Port Profile (SPP)
- Service Discovery Application Profile (SDAP)
- Video Distribution Profile (VDP) [24;25]

Each of the Bluetooth profile has its own use. For example, Human Interface Device Profile (HID) which is used in keyboard, mouse and remote control must know to handle devices that control the computers and televisions. Dependency of Bluetooth on the other devices, user interface format and the protocol stack used by a profile are the basic information that should be contained in the Bluetooth profiles. The main target of

the Bluetooth profile is to help interoperability. It does so by providing guidelines for user interface, defining parameters, reducing implementation options and specifying mechanism to combine different standards. This ensures uniformity, allows devices to use the same feature and to operate in the same way. [24;25]

2.2.3 Bluetooth Protocol

Bluetooth system can be subdivided into layers and protocols on the basis of its implementation. Bluetooth protocols are used to manage transfer of data wirelessly through links, services and applications. The Bluetooth protocol can be subdivided into a Bluetooth host and a Bluetooth controller. The Bluetooth host is the upper layer of the stack and is present in the form of software in a device. Bluetooth host is generally installed in the operating system such as laptops and phones. The Bluetooth controller is the layer below the Bluetooth host. It communicates with the upper layer via Host Controller Interface (HCL). The Bluetooth controller is present in the form of hardware such as a personal computer card which is connected to target devices. Figure 5 shows the block diagram of the Bluetooth protocol stack. [22,9-11]

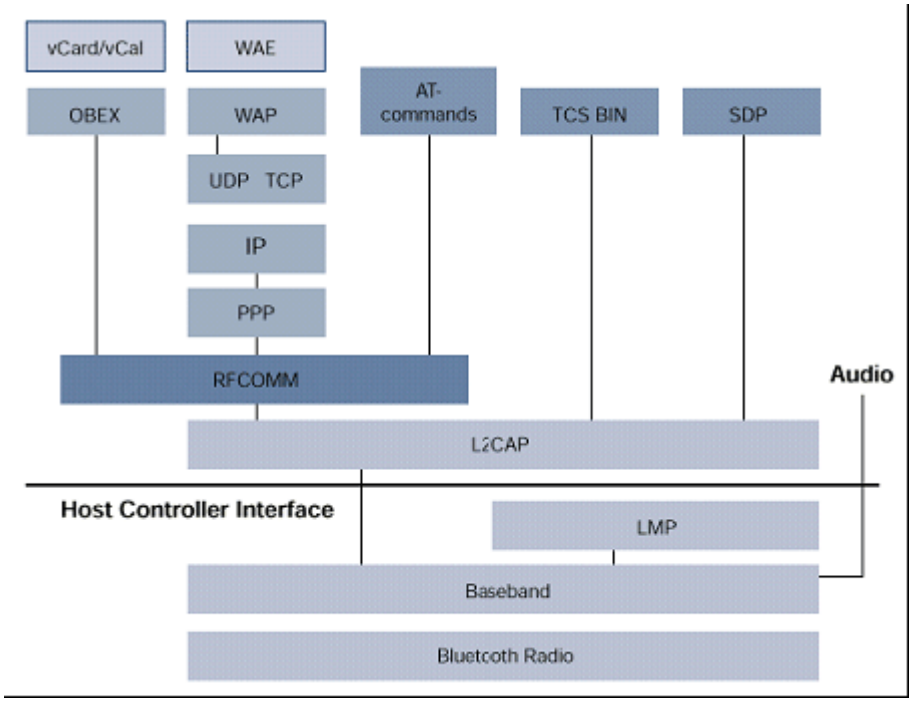

Figure 5. Different layers of the Bluetooth Protocol. Copied from Bluetooth Protocol [26]

Figure 5 illustrates the arrangement of different layers of the protocols and the layers in the protocol stack. Most of the protocol layers shown in figures are defined by SIG whereas a few of them are adopted from standards-making organizations. A device must follow the stack architecture for the desired and proper functioning of Bluetooth. The lowest layer the Bluetooth radio defines the requirements for operating devices in the proper frequency. The Baseband and link controller controls the radio frequency links for making connections. The Link Manager Protocol is responsible for the links setup and configuration. The HCL provides a medium for communication between the host and the controller layer. The L2CAP protects the lower layer details from the higher layer. RFCOMM provides serial port communication. OBject Exchange (OBEX) and Wireless Application Protocol (WAP) are built on one of the protocols specified by SIG. SDP helps to discover a service provided by Bluetooth. Telephony Control Protocol Specifications (TCS) provides communication service between Bluetooth devices. [22,9-11]

2.3 Java

Java is an object-oriented programming language introduced by Sun Microsystems. James Glosing and his team developed Java in 1995. Most of the syntax in Java is derived from the C and C++ programming languages but it is simple and easy to use compared to C and C++. Java programs can be run on any device capable of running the Java Virtual Machine (JVM) and do not need to be recompiled. The JVM is a virtual machine which executes Java byte codes. Currently Java runs on about 850 million computers and 3 billion mobiles worldwide. Java was made free and open source software in 2006 by Sun Microsystems. [27;28]

The features that make Java a powerful and popular programming language are the presence of a compiler and interpreter, platform independence, object oriented, robust, secure, multithreaded and interactive, easily distributed, portable and high performance. Source codes in Java are first compiled and then interpreted by JVM. This allows for error reduction and security. Platform independence allows Java to be portable. Once the Java byte code has been compiled and interpreted, it can be run on any device having the JVM without recompiling the bytecode. Java is an object oriented programming language which uses classes and objects, providing it with features such as code reusability and maintenance. The Robust feature of Java makes it reliable. A compiler and interpreter help to reduce code errors and runtime errors. Java also has an automatic garbage collection and memory allocation mechanism. [27;28]

Java does not use memory pointers and always checks an array index limit. Java is a multithreaded programming language. This feature allows a single program to have multiple tasks to execute independently at the same time. The libraries for Hypertext Transfer Protocol (HHTP) and File Transfer Protocol (FTP) protocols are provided by Java, which allows a developer to use these functions to transfer files over the Internet. Since the Java code is compiled to Java bytecodes rather than machine language, the Java programs are fast to run on any devices running JVM. [27;28]

Java programming language is used to develop desktop applications, web applications and mobile applications. Java is also used to make an Android application. There is a difference in pure Java and Java used to program an Android application. The virtual machine used by Java is a stack machine but a virtual machine used by Android is register-based architecture. Most of the libraries of Java are used in an Android for programming, hence Android is Java-based. For the rest of the libraries Android either has its own replacement libraries or does not require them at all. Unlike Java, which has the main function as the starting point of a program, Android has oncreate function. [27;28]

3 Implementation of Android and Desktop Applications Development

3.1 Eclipse

Eclipse is an open-source IDE donated to the community by International Business Machine (IBM). Java and the Android programs are written using Eclipse. Eclipse can also be used to program in other programming languages by means of external plug-ins. Also the plug-ins can extend the usability of Eclipse. Eclipse is developed by an open source community which has about 2000 open source projects related to different aspects of software development. Syntax-highlighting editor, incremental code compilation, a thread-aware source-level debugger, a class navigator, a file/project manager, and interfaces to standard source control systems are the features provided by Eclipse, once combined with JDT. [29;30]

Eclipse was started as project for IBM which later in 2001 was released as an open-source project. An association was then formed for further development of Eclipse as open source, and the original board members were Borland, IBM, Merant, QNX Software Systems, Rational Software, Red Hat, SusE, TogetherSoft and WebGain. Later in 2004 the Eclipse foundation was formed. It is a nonprofit organization responsible for governing the Eclipse projects. Figure 6 shows the basic window of the Eclipse. [29;30]

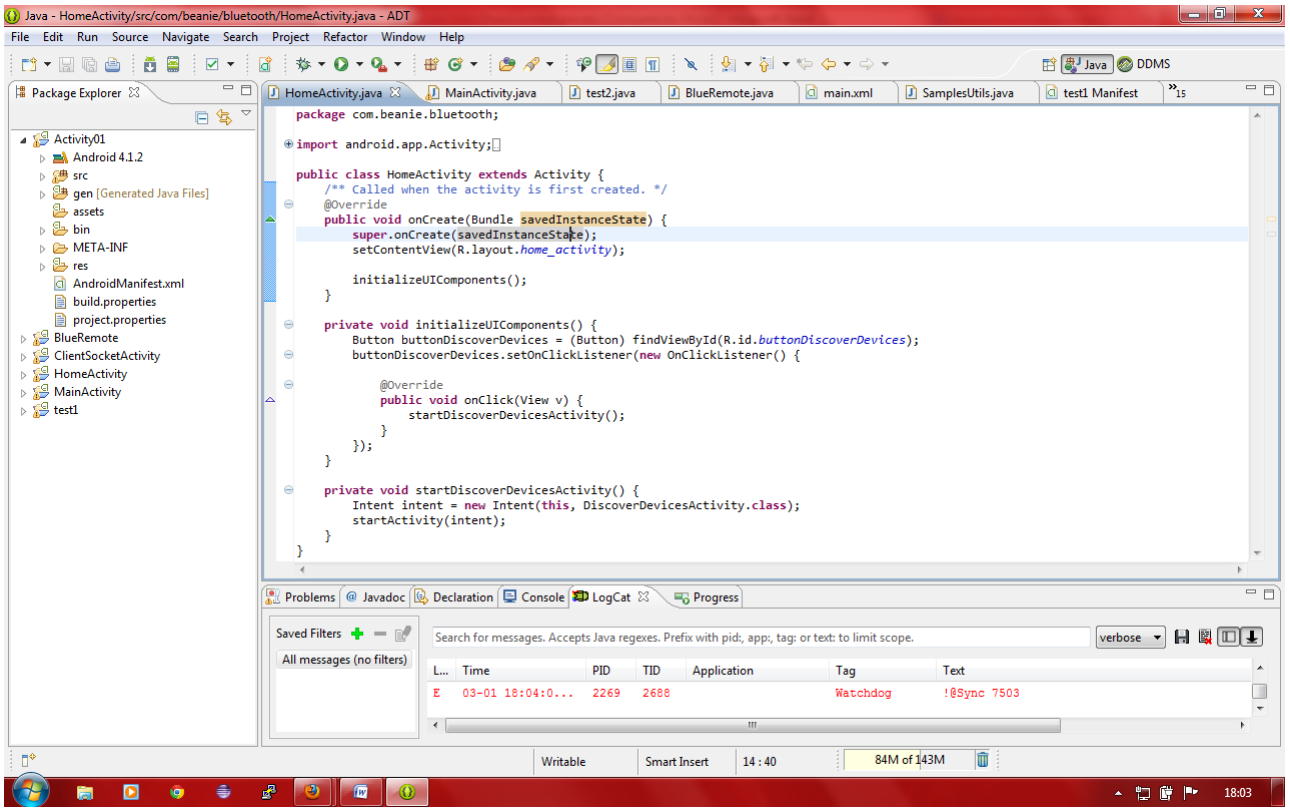

Figure 6. User interface of Eclipse IDE. Screenshot taken from laptop

Figure 6 shows a simple user interface of Eclipse IDE. The whole figure is known as the work bench window. The window consists of a package explorer view, a type hierarchy view, a Java outline view, wizards for creating the Java elements, a Java editor, a short cut bar, a menu bar and a tool bar, as shown in figure 6. The Package explorer shows the entire project that is currently in the workspace and the project highlighted in the package explorer is the project that is currently in use. The Type hierarchy view shows the sub and super hierarchies whereas the Java outline view shows the classes and the Java units. Wizards for creating the Java elements consist of the Java classes, packages and interfaces. The Java editor is where all the codes for application development are written. The codes can be highlighted, edited or fixed in this area. All the commands such as edit, run, build and others are included in the shortcut bar, menu bar and tool bar. [29;30] To develop the client-side application of the project in the Android platform using the Java programming language, Eclipse was used.

3.1.1 Eclipse Architecture

The architectural structure of Eclipse is completely comprised of plug-ins. Eclipse itself does not contribute much to end products. plug-ins provides functionality and services in Eclipse. A suitable environment for programming is created when a specific lan-

guage based plug-in and the Eclipse plug-in are combined. The basic platform consists of a platform runtime, a workspace, a workbench and a help system. A simple architecture of Eclipse is shown in figure 7.

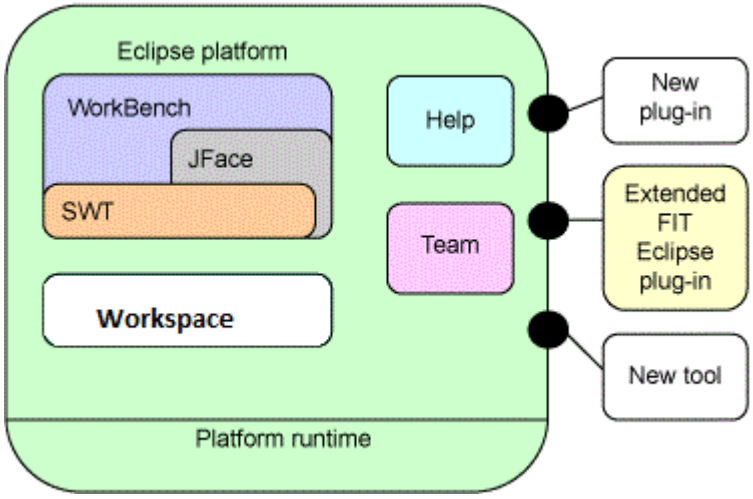

Figure 7. Architecture of Eclipse. Copied from Fit and Eclipse [32]

Figure 7 illustrates how Eclipse is comprised of different components and plug-ins. The platform runtime is the essential part of the architecture. Loading and binding of plug-ins and creating and managing a suitable environment for development tools are all done by the platform runtime. The workspace manages all the projects created in Eclipse. Each of the projects has its own folder containing all the related files and resources. Any changes with the files in a project are handled and updated by the workspace. Users can have multiple workspaces at a time and can switch between the workspaces without exiting Eclipse. New updates even allow creating and importing projects from outside the workspace. The help system provides the documentation to the online books and APIs used for development. Plug-in Development Environment (PDE) provides all the necessary tools to run, build, deploy, develop and test the plug-ins. [31]

JDT provides all the required plug-ins for creating an environment for the Java application development. Java editors, wizards, debuggers, compilers are all provided for development in Eclipse by JDT. Workbench is the user interface element of the architecture that determines how Eclipse appears to users. It consists of two toolkits, the Standard Widget Toolkit (SWT) and JFace. The SWT provides independent graphical widget toolkit for application development. It is used to portrait a user's information in a graphical form. JFace consists of the tools necessary for user interface programming. It

is designed to work with SWT. With features such as actions and viewers JFace is used for programming frameworks and wizards. [31]

3.1.2 Overview of Eclipse Plug-ins

Eclipse itself is not capable of application development without any plug-ins being installed in it. In general plug-ins are a group of codes that are used to extend functionality of any IDE. Depending on the situation plug-ins can be added and removed any time. They are extendable. One or more plug-ins can be added or can be connected to each other via extension points provided by each plug-in. A plug-in can be added to an application as a directory or a jar file and is easily available for sharing. The Eclipse plug-ins implement Open Services Gateway initiative (OSGi) bundles which are used to manage plug-ins in an Eclipse application. [33]

A valid OSGi header must be available in the manifest file with the plug-in name and version. The manifest file tells Eclipse about the time for activation of the plug-ins. Creating, developing, debugging, testing building and deploying of the plug-ins in Eclipse are all done by PDE. [33]

Eclipse is not a single program, but rather software surrounded by hundreds of plug-ins to form a complete IDE. A single plug-in is sometimes enough to build an application. However sometimes a plug-in may rely on other plug-ins for services or may provide services to other plug-ins depending on the situation. All the functions required to be performed by a plug-in are included in its code while its dependencies and services are included in MANIFEST.MF and plugin.xml files. MANIFEST.MF contains runtime information of a plug-in whereas plugin.xml contains extension and an extension point. The plug-in used for this project in Eclipse is the ADT. [33]

3.2 NetBeans

NetBeans is also an IDE used mainly for Java programming. Other languages such as JavaScript, PHP, Python, Ruby, Groovy, C, C++, Scala and Clojure can also be used to program in NetBeans. NetBeans is written in Java and is compatible with all the operating systems capable of running the Java Virtual Machine. Initially developed in 1996 by a student of Java IDE, NetBeans was later acquired by Sun Microsystems and was released as an open source project. NetBeans is available for free download from the NetBeans official website. The download includes documentation, sample projects,

tools and a module to integrate with Java. Like Eclipse NetBeans functionality can also be extended by external plug-ins. To program in Java using Netbeans JDK (Java Development kit) plug-ins must be installed. The latest version available of NetBeans is Version 7.0. [34] Figure 8 shows the window of the NetBeans IDE.

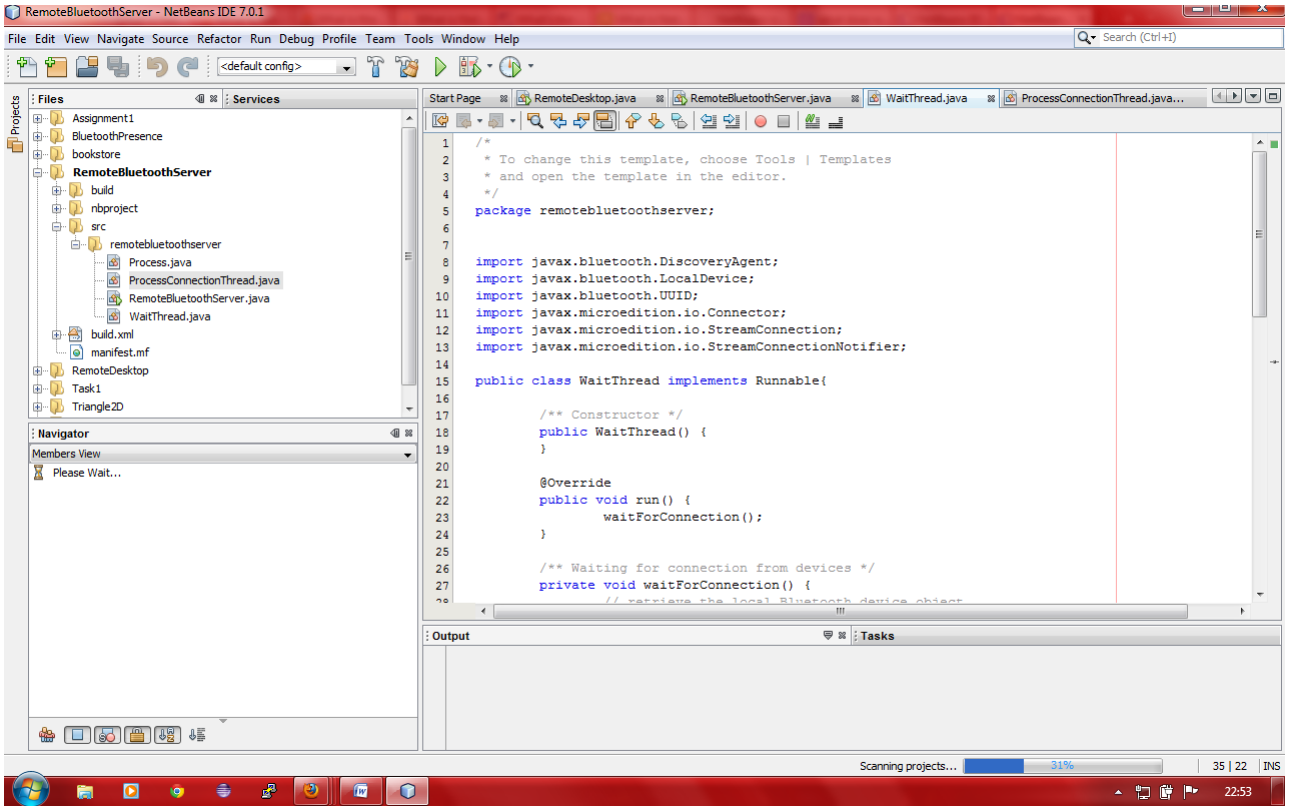

Figure 8. User Interface of NetBeans. Screenshot taken from laptop

Figure 8 shows a figure of NetBeans with the basic content of the IDE. Like Eclipse Netbeans also contains package explorer, navigator, editor and tool bars. All the functionalities of NetBeans is provided by modules. Modules are sets of modular software components. These modules are already included in NetBeans during download of NetBeans. Additional modules can be installed depending on the requirements. The module consists of three components, NetBeans Profiler, Graphical User Interface (GUI) design tool and NetBeans JavaScript editor. The monitoring of an application by finding memory leaks and an optimizing speed is done by NetBeans Profiler. NetBeans profiler is a tool used for monitoring Java applcaitions. The GUI design tools allow designing an application. The drag and drop functionality of GUI components are available in NetBeans. The JavaScript editor of NetBeans provides support for Cascading Style Sheets (CSS), JavaScript and Ajax. [34] To create a server-side application of the project in the desktops using the Java programming language, NetBeans was used. Eclipse could also be used for the purpose. The reason behind choosing NetBeans

over Eclipse for desktop application programming is because of previous experience and familiarity with the IDE.

3.3 Android Virtual Device

The Android Virtual Device (AVD) is an emulator included in the Android SDK. The AVD lets a user to run, build and test an application without the need for any physical devices. AVD can be considered an exact replica of a real Android device, except that lacks a few features. The AVD cannot be used to make calls. The Android market is not accessible from the AVD. The applications that use sensor and Bluetooth cannot be tested in AVD. Except for this, testing an application in the AVD appears similar as in a physical Android device. Android mobile phones can also be used as emulators instead of using emulator provided by SDK. All the instruction sets required for using an Android device as an emulator can be found in the Android developer's official web-site.[35] Also using an Android device as an emulator is faster compared to using AVD. Figure 9 shows the Android Virtual Device provided by an Android SDK.

Figure 9. Android emulator. Screenshot taken from laptop

Figure 9 shows an emulator used to run, build and test an Android application. Also multiple emulators can be run at a time without affecting any other emulators. This feature is possible because of the Dalvik Virtual Machine (DVM). Each Android application has its own instance of DVM. When Java classes are written in Eclipse, the class files are compiled and converted into the .dex file by the dx tool. The .dex files are Dalvik Executable files that are executed by DVM. Issues such as threading and low memory management in DVM are all managed by the Linux kernel. [35]

3.4 Project Components

3.4.1 Android Project Components

The application package file (.apk) is the file that is installed in a device to run an application. It contains the program code, resources, assets and manifest file. Some files are created by default whereas some are created as required. An Android project is comprised of the Android version directory, src, gen, assets, bin, res, AndroidManifest.xml and project.properties. The Android version directory tells about the current version used for application development. The file android.jar is included in this directory which is used in the project. The Java files that contain all the source code can be found in the src directory. The gen folder consists of all the Java files automatically created by ADT. The most important file in the gen directory is the R.java. The R.java file contains the resources such as drawables, layouts, strings, arrays, and anything declared in the resources. The Assets directory in an Android project is generally empty but can be used to store raw asset files. These raw files can be later accessed from the Java file if required. The Bin directory contains other compiled resources and the final .apk file that is installed in a device. [36]

The res directory is one of the most important components of a project. The res directory further contains subdirectories such as drawable, layout, menu, values, color, raw and xml. The images, layouts, menu items, strings, colors, files and xml codes are simultaneously contained in these subdirectories. AndroidManifest.xml is also a very important file of an Android project. This xml file consists of all the information about an application. Furthermore the xml file describes the activities, services, intent receivers, and content providers used in an application. The information about the permissions requested, the external libraries used and the API level compatibility are also included in the file. The project.properties file contains the project setting about the build target and can be edited as required. [36]

3.4.2 Java Project Components

The Java Archive (.jar) file is an executable file for Java projects. The .jar file is an archived file format consisting of the Java classes, metadata and resources. This file can be run on any computer with JVM. In order to access the file content of the .jar file any zip tools can be used. This file is installed in a computer in order to run an application. During the development of an application, some files are created by a user whereas some are created by default. The Java project consists of files such as build, nbproject, src, build.xml and manifest.mf. [37]

The build directory in the Java project consists of Java files with .class extension. The .class files are the compiled Java binary classes. These files can be decompiled to get a readable .java class. The nbproject directory consists of the NetBeans project metadata and resources. Some of the files in the nbproject contains data specific to the system. The src directory contains all the files where all the coding is done. The files in this directory are the files with .java extension. Buid.xml contains information about the version of JDK used for the development of the project. The manifest.mf file contains all the information about how the .jar file can be used. It also contains information about all the files packaged in the .jar file. Using the information provided in manifest.mf, the .jar file can be used for many purposes. [37]

4 Project Requirements and Description

4.1 Client Application Requirements and Description

4.1.1 Android Application Background and Specifications

As mentioned in the introduction of this thesis, an Android application acts as a remote control device for a Bluetooth-enabled computer. For this project the Android device acts as a client to the computer. The touch screen feature of the Android device is used to control the mouse in the computer whereas the keyboard of the phone is used to control the keyboard of the computer. Since the screen of the phone is used to simulate the mouse and keyboard, all the information from the phone needs to be sent to the server application. In the case of the server side, everything is taken care of by the computer application. For the exchange of information between the client and server, Bluetooth is used as a mediator. The functions such as enabling the Bluetooth in a device, discovering Bluetooth devices, creating a connection to the server and manipulating the touch and keyboard events are all done in the client side. Figure 10 shows the actions performed by the client side application in the Android device.

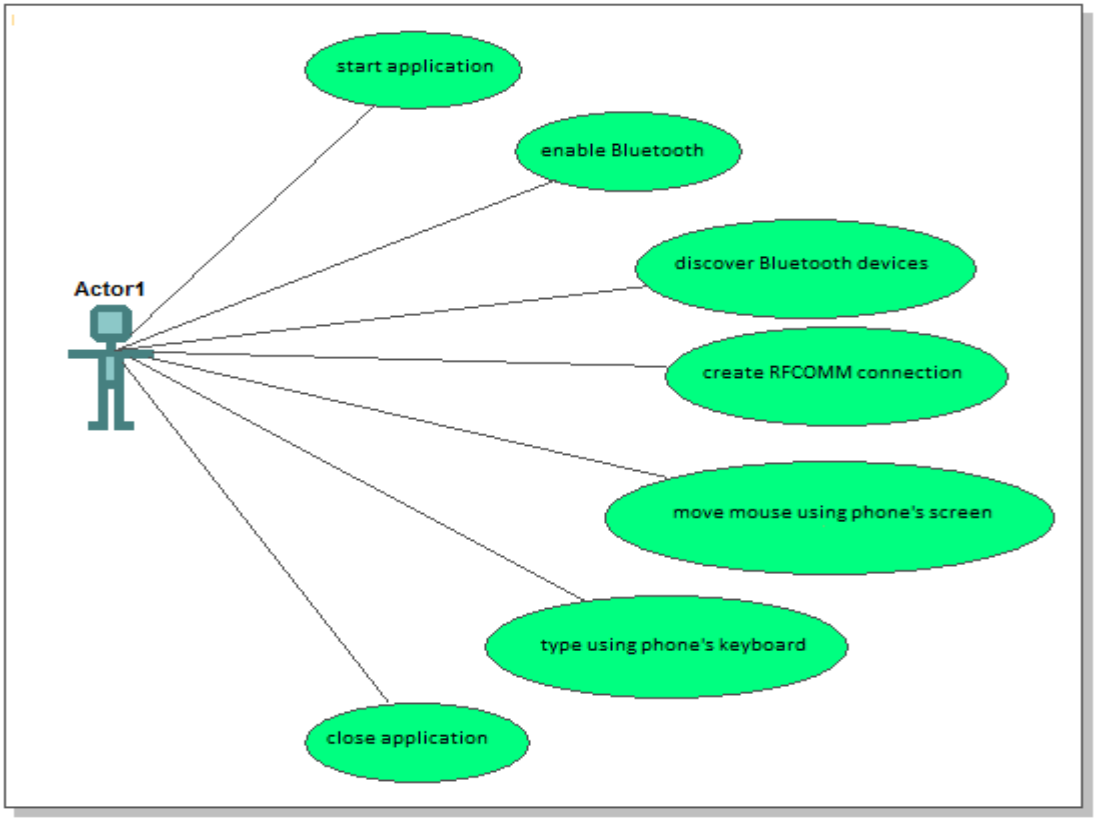

Figure 10. Use case diagram of the client application

As illustrated in figure 10, the Android application should use the Bluetooth wireless technology to create RFCOMM Bluetooth communication with the server in the computer. RFCOMM is a serial port communication which provides a simple data stream to a user. It should be kept in mind that not every Android device has Bluetooth hardware installed in it. An application will not work for the phones with an Android version earlier than 2.0. The Bluetooth API has been available in Android only since Android version 2.0. The application is currently available for computers with an English keyboard layout only. Once the application is started it queries the Bluetooth devices in the surrounding within the range and sends requests for connection. While sending the request for connection with the other devices for the first time, a user will be prompted to accept connection with a pin code. [38,339-341]

The specialty of the application is that once it has been paired with the device, it stores the paired device's name. So every time when the application is rerun, the user will not have to rescan the devices if the user intends to use the same one. Also a user will not be prompted to accept a request for connection with the pin code for previously bonded devices. The client side application is responsible for sending requests to the server application. For the devices to make a connection between each other, it is very important for the server side to open a server socket before sending a request from the client. A connection between the server and the client can only be established when they have connected the Bluetooth socket in the same RFCOMM channel. Once the connection is established, all the data sent from the client are received and manipulated by the server application, hence allowing a user to accomplish what the application is intended to do. On closing the application the Bluetooth socket will be closed and Bluetooth will automatically be turned off in the Android device.

4.1.2 Architectural Design

The Android application for this project acts as the client to the server located in the computer. The client application contains the user interface of the application. Most of the time during the use of application the user interacts with the client part than the server. All the works such as connection to the server, sending data to the server, controlling the keyboard and the mouse in the server wirelessly via Bluetooth are done by the client application. Figure 11 shows the flow chat of the client application.

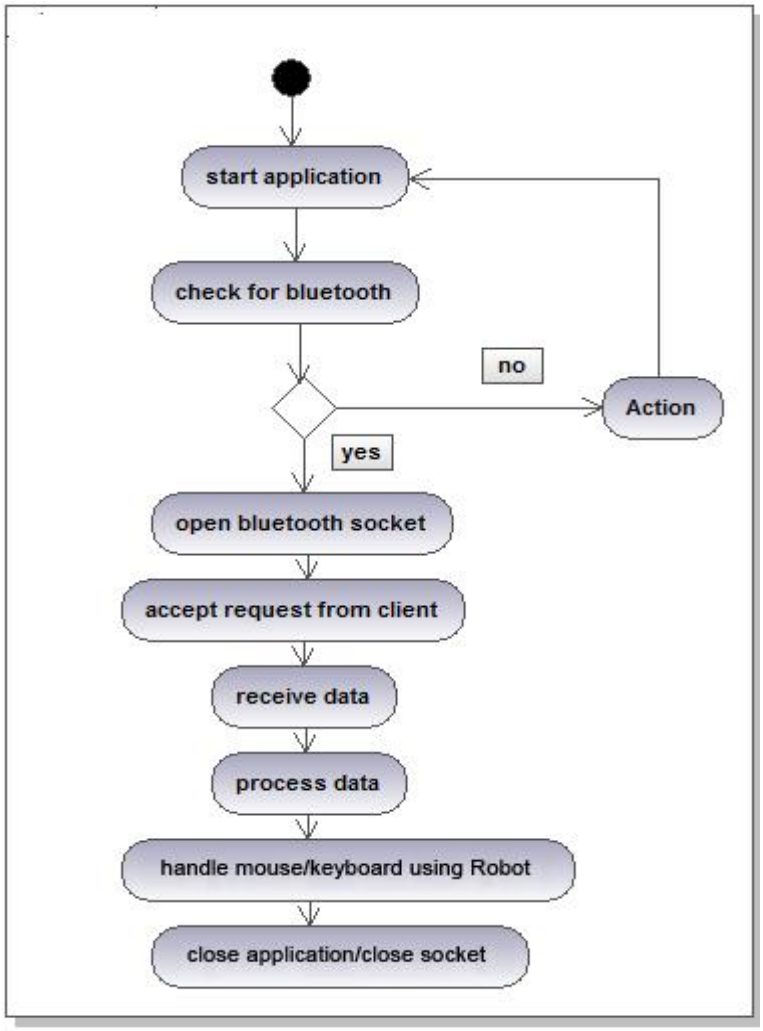

Figure 11. Flow chart of the client application

Figure 11 shows the steps that take place during the use of the application. When the application is started the availability of Bluetooth is checked. Before using this application, it should be kept in mind that not every Android device necessarily has Bluetooth in it. If available the application prompts to scan the Bluetooth devices present in the surrounding within its searchable range. In this application, a device means the computer running the server application. If Bluetooth is not available, the application provides a dialog box asking a user to open Bluetooth and moves forward to scan the Bluetooth device after the user has turned the Bluetooth on. The application then scans the Bluetooth device and lists all the available devices as a list. Then a user has to select a device from the list where the server is already running. It should be noted that the server must already be active when the application sends the request for connection.

After the selection of the device, the application sends a request for the RFCOMM Bluetooth connection. The application will ask for a pin code for pairing if the server is connected to the client for the first time. Once the devices are paired a user will not have to rescan the devices when reusing the application as the application stores the paired device's name. The screen and the keyboard of the phone are used for controlling the server computer. After a successful connection between the server and the client, the data received when a user touches the screen and types in the keyboard of the client are sent to the server. Finally when a user closes the application Bluetooth is turned off closing the RFCOMM connection and the application terminates.

The user interface in an Android application can be designed by coding in an xml file. Different types of layouts, buttons, test fields, colors and other user interface elements can be added to the xml file. An Android application can contain multiple views at a time. Different other elements such as dialog boxes, widgets, images and videos can also be added to the views. The xml provides the views that appear to a user. Figure 12 shows the user interface of the different stages of the application.

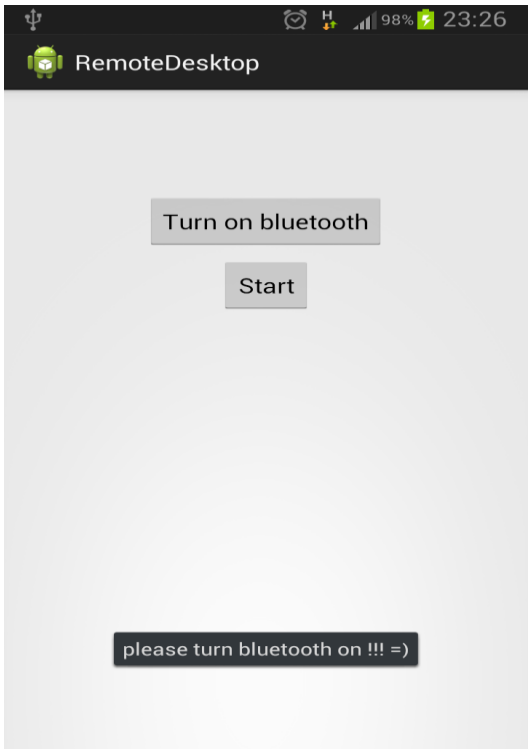

Figure 12.1 First screen of the client application

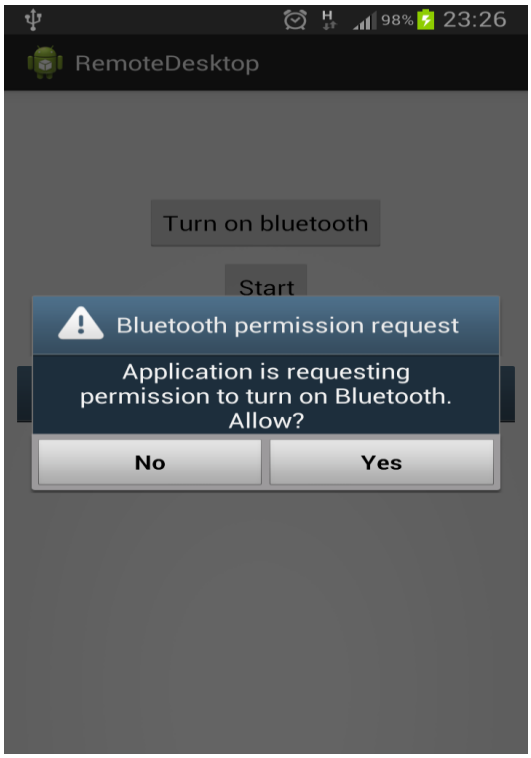

Figure 12.2 Second screen of the client application

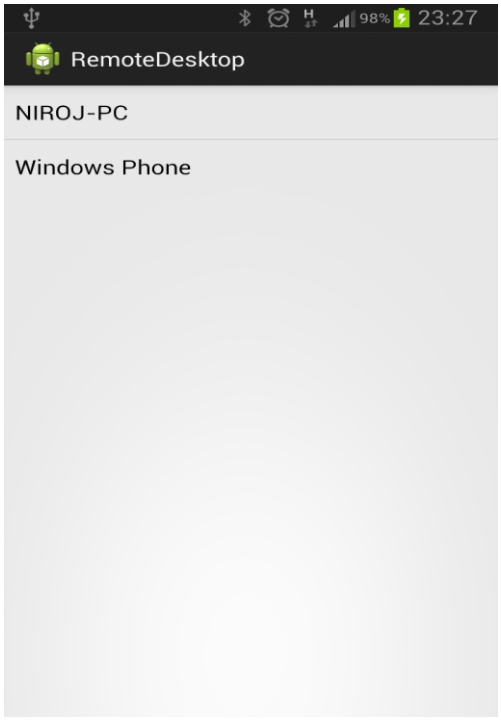

Figure 12.3 Third screen of the client application

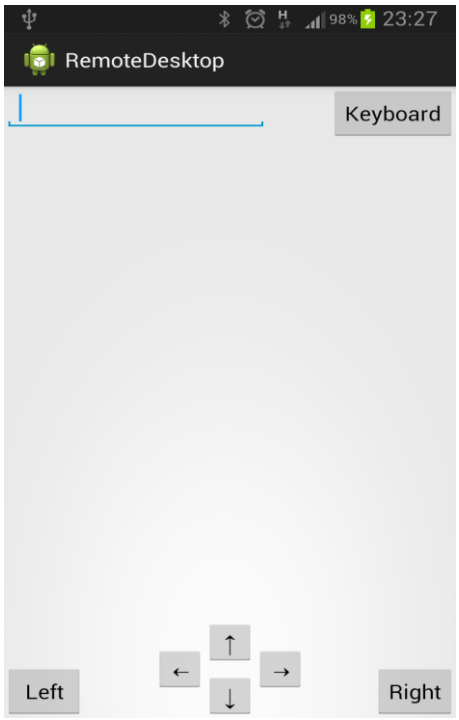

Figure 12.4 Fourth screen of the client application

Figure 12. User Interface for different stages of the client application

Figure 12 shows the user interface for different stages of the client application as it appears to a user. Figure 12.1 shows the starting screen of the application. It is the view where a user is prompted to enable Bluetooth if Bluetooth is turned off. The same view has a button named start which starts the scanning of the Bluetooth devices when pressed as shown in the figure 12.3. Figure 12.3 appears when the start button is pressed, here all the Bluetooth devices are listed in a list view. Figure 12.2 shows an alert dialog asking a user to turn on the Bluetooth in the phone.

In the figure 12.3 when a user selects an item from the list the figure 12.4 appears in the screen, which is the main view for the application. This view consists of an edit text field where a user can type. The characters typed in the field are the characters that appear in the server application. There are seven buttons in the view. Two buttons at the bottom corner represent the right and the left button of the mouse of the computer. The button at the side of the text field named as the keyboard is used to show and hide the keyboard of the phone as required. The remaining four buttons are used as arrows to navigate in the computer. The rest of the screen of the phone represents the screen of the server computer. Moving along the screen of the phone by touching the screen will move the mouse pointer on the server screen.

The Android application is developed using the Java programming language. It contains objects, classes and variables. Classes in Java are templates describing the behavior of objects. Objects in Java are instances of classes and variables in Java are used to store data. An application can contain any number of classes; classes can contain any number of objects and variables. Figure 13 shows the high level class diagram for the Android application.

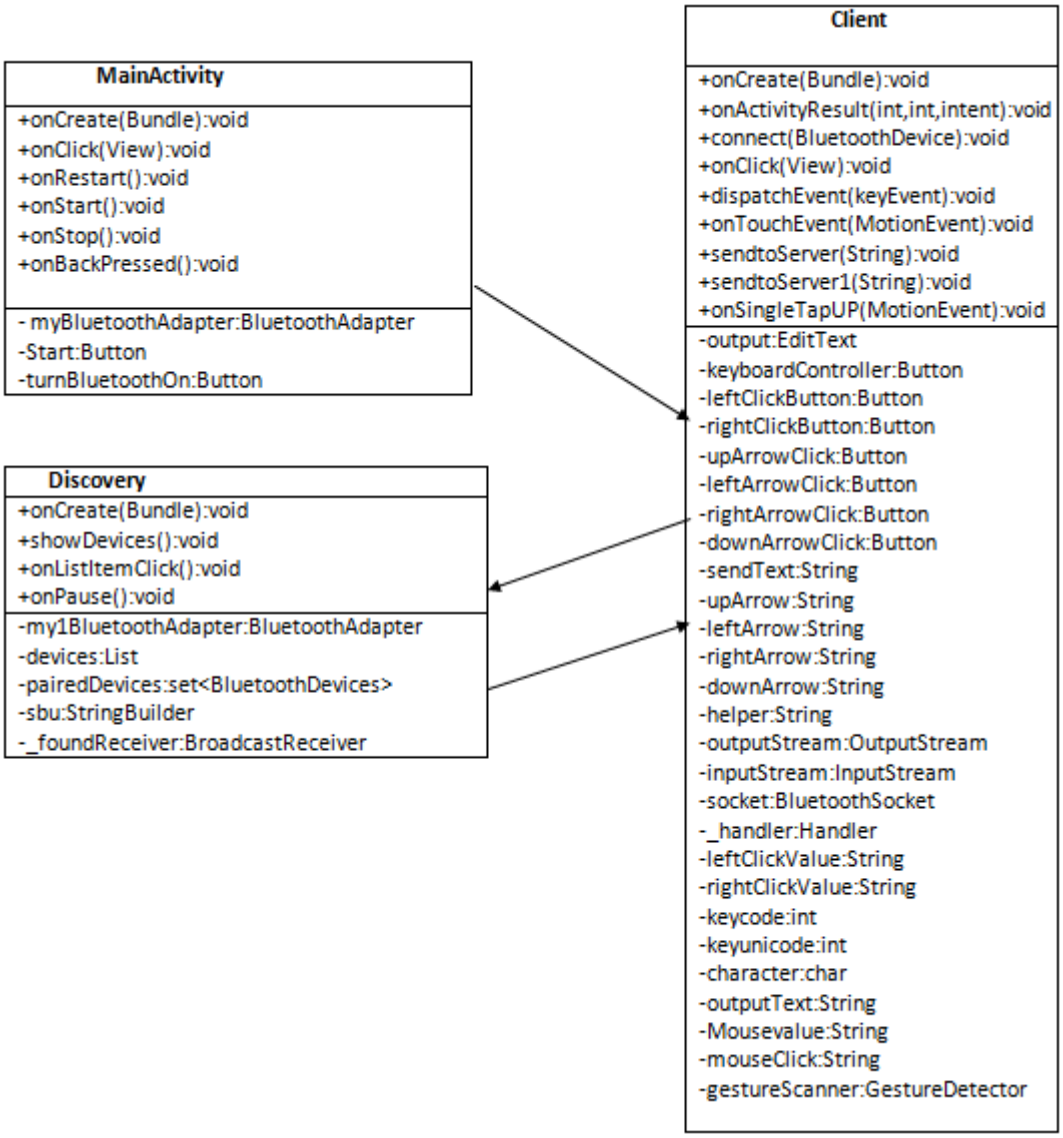

Figure 13. High level class diagram of the client application

Figure 13 shows the classes used in the client application and the relationship between them. The client application consists of three classes, MainActivity, Client and Discovery. MainActivity is the starting activity which starts the Bluetooth service in the application. This class then calls the Client class. Before doing anything the Client class calls the Discovery class which will scan the Bluetooth devices and list all the available devices. The Discovery class lies on the top of the view with the Client class still being

semi-visible. On selecting the Bluetooth device focus comes back to the Client class which will then make the Bluetooth connection and send the data to the server. On pressing the back button in an Android device the application is terminated. During the termination function like closing the Bluetooth socket connection, turning off the Bluetooth and closing the application are all handled by the Client class.

4.2 Server Application Requirements and Descriptions

4.2.1 Application Background and Specification

As mentioned in the introduction of this thesis, a computer application acts as a server for the client Android application. To run the application the computer must have the Bluetooth devices installed in it. A Universal serial Bus (USB) Bluetooth device can also be used for the application if a computer does not have Bluetooth preinstalled in it. The server application opens a socket and waits for a client to send the request for the connection. Once the request is accepted, the RFCOMM channel is established between the server and the client. The reason for using RFCOMM over other Bluetooth protocols is the common use of serial communication and easy use of API. Bluecove was used in the application to deal with all the Bluetooth issues which is a Java library for Bluetooth. Figure 14 shows the actions performed by the server side application in the computer.

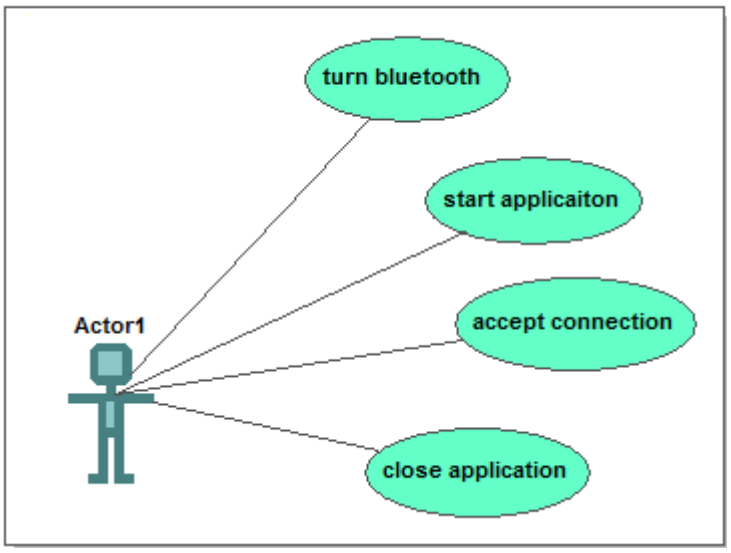

Figure 14. Use case diagram of the server application

As illustrated in figure 14, the computer application uses the Bluetooth wireless technology to create a RFCOMM Bluetooth communication with the client in Android. Once a connection has been established between the server and the client, the data sent from the client are received by the server. The data are then processed using the Java codes. The data then are passed to a class where all the data are handled by the Java class called Robot. Robot is the Java class used to test, run and code the applications where the use of the mouse and keyboard is needed. The task of simulating the mouse movement and the keyboard press in the computer is handled by the Robot class.

The application can also be used as a chat application with Bluetooth as a communication mediator. The application properly functions for only a keyboard with the English layout. The reason for this is that Java does not provide classes to handle key events for the keyboard with other layouts except English. If a developer has to create such an application for the keyboard with another layout, all the key events for the keyboard must be created manually by coding.

4.2.2 Architectural Design

The computer application in this project acted as the server for the client application in the Android phone. This server-side application of the project is where a user has to interact less with the application. Once the server is started, the rest of the work is done by the client application. Server application opens the Bluetooth socket for communication with the client. All the data sent by the client are processed in the server application. The server uses Bluecove to perform the functions related to Bluetooth. Figure 15 shows the sequence diagram of the server application.

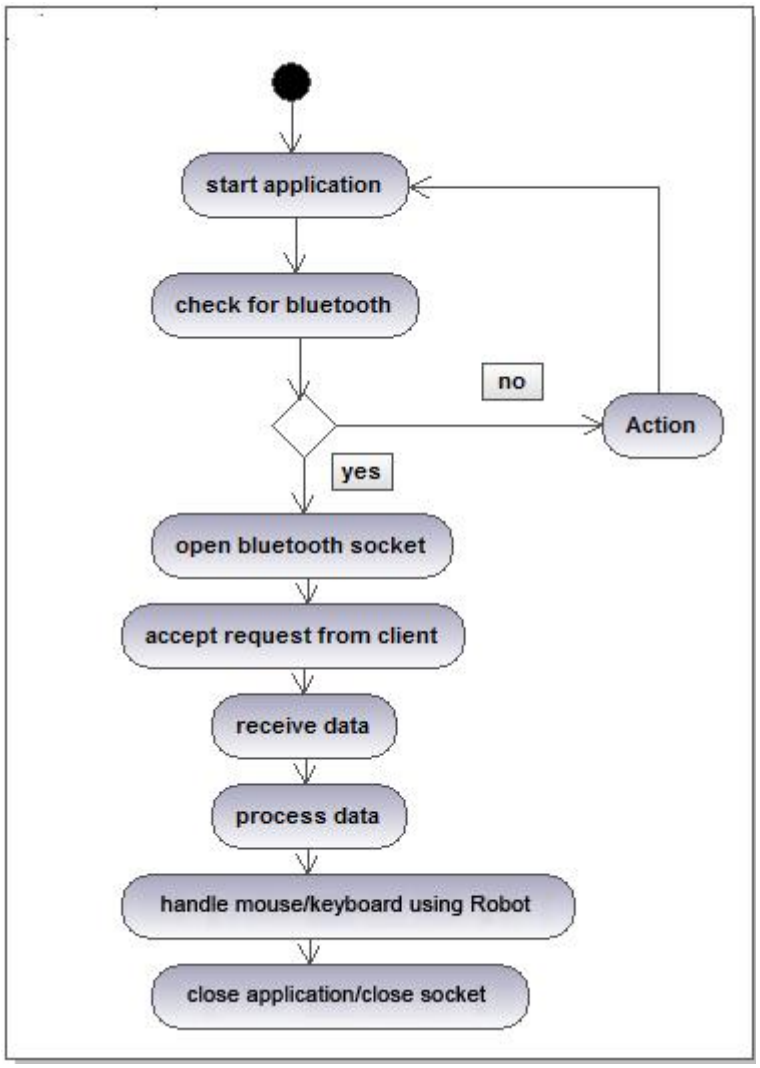

Figure 15. Flow chart of the server application

Figure 15 shows the steps that take place during the use of an application. When the application is started, the state of the Bluetooth device is checked. If Bluetooth is turned on, it will prompt the application to open the Bluetooth socket for the client. If not, the application terminates and the application has to be restarted after turning on Bluetooth. When the request from the client application is accepted, then the RFCOMM communication takes place between the server and the client. On receiving the data by the server, the data is processed applying the mathematical formulas and the Java codes. The data is then used by the Robot library provide by Java to simulate the mouse movement and the keyboard press in the server. Finally the Bluetooth socket connection is closed on termination of the application.

The server application is also coded using the Java programming language. Like the client application it contains classes, objects and variables. In addition to that it uses Bluecove and Robot APIs to perform functions related to Bluetooth and the mouse

/keyboard events simultaneously. The server application does not have any user interface. The server application is just a .jar file for the server with JVM. Figure 16 shows the high level class diagram for the server application.

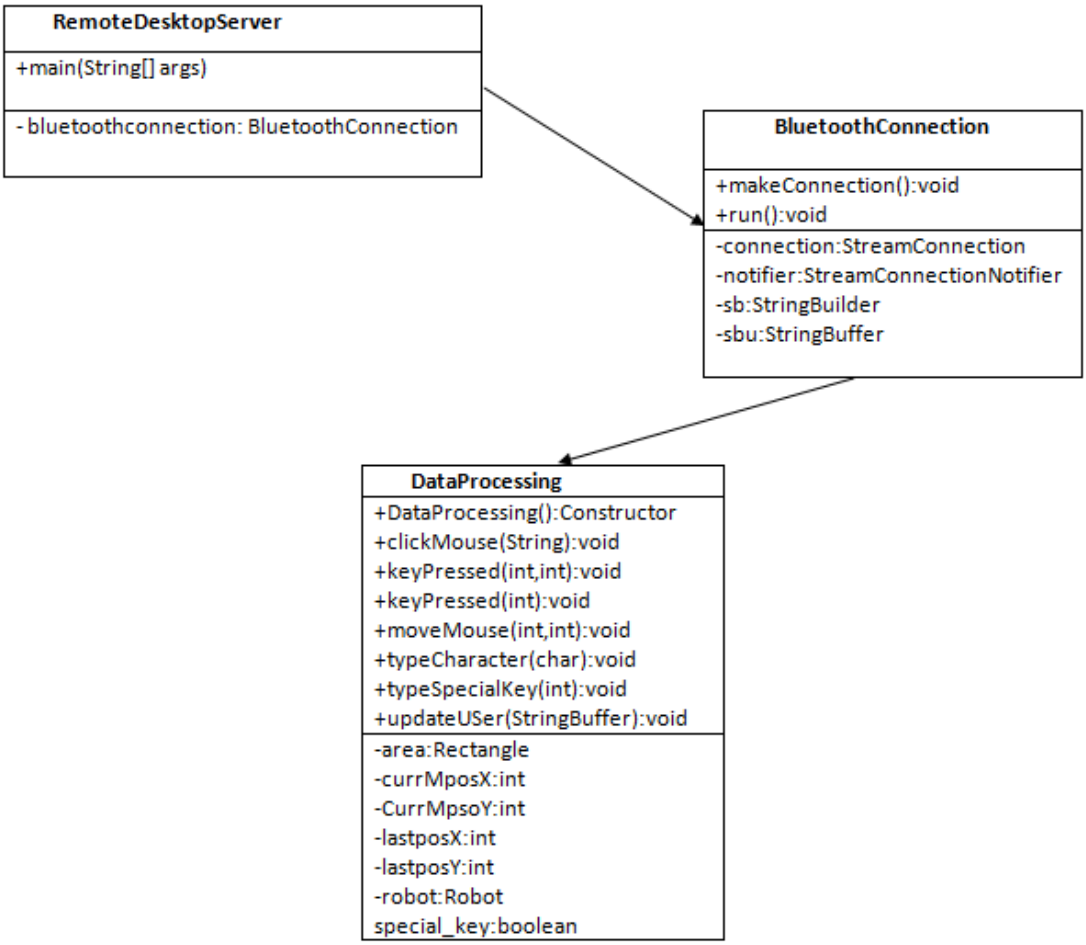

Figure 16. High level class diagram of the server application

Figure 16 shows the classes used in the server application and the relationship between them. The server application consists of three classes, RemoteBluetoothServer, BluetoothConnection and DataProcessing. RemoteBluetoothServer contains the main function and is the starting class of the application. This class calls the BluetoothConnection class where functions for opening the Bluetooth socket and connecting to the client are coded. On establishing the RFCOMM connection with the client successfully, the BluetoothConnection class receives the data from the client in a byte form and changes it into the Stringbuilder format. The data is then passed to the DataProcessing class from the BluetoothConnection class. In the DataProcessing class all the data are processed and then handled by the Robot API. The movement of the mouse and the

keyboard pressed are all controlled by the Robot. Finally on terminating the application the Bluetooth socket is closed for the connection.

4.3 Sequence Diagram

The sequence diagram is an interaction diagram that shows in what order and how processes are carried out. The classes contained in each application are somehow interconnected with each other. A class might call a function in another class to do a specific task or can send data to another class asking it to perform some tasks. Figure 17 shows the sequence diagram for the project.

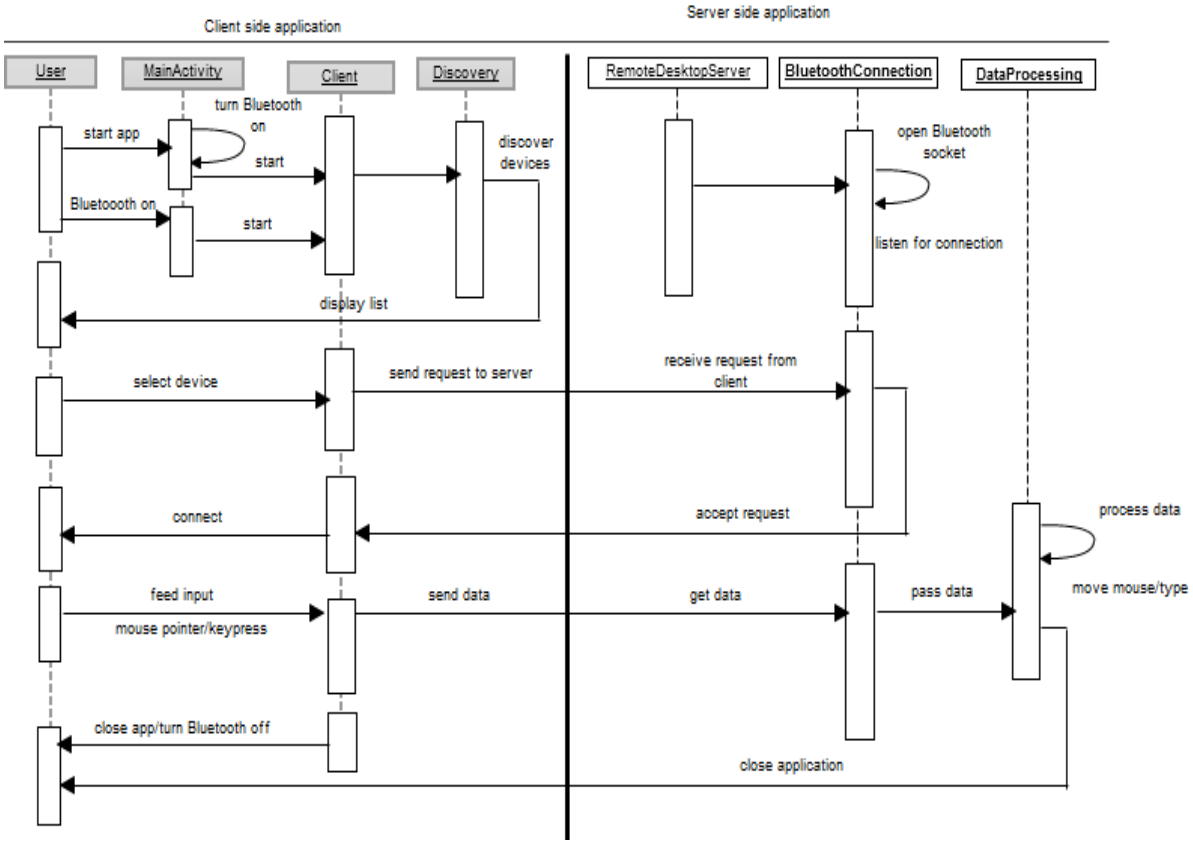

Figure 17. Sequence diagram of the project

Figure 17 illustrates different tasks performed in the project and their order of execution. A user starts the project by running both the client and the server applications. On the client side the MainActivity checks if Bluetooth is turned on or not. If Bluetooth is on, a user starts the scanning process by clicking the start button which initiates the Client class. If Bluetooth is not on, a dialog box appears asking the user to turn on Bluetooth. Then the user starts the scanning process. The Client class then calls the Discovery class to initiate the scanning process for the Bluetooth devices. On finishing the discovery of the Bluetooth devices the Discovery class sends the list of the discov-

ered devices to the Client class. The user is then asked to select the desired device where the server for the project is already running. The client class then sends the request to the server for the Bluetooth communication.

The RemoteDesktopServer class of the already running server application calls the BluetoothConnection class. The BluetoothConnection class is responsible for opening a socket connection for the incoming Bluetooth connection. This class then accepts the request from the client application sent by a user. After this both the server and client applications are connected to each other via Bluetooth. The Client class then accepts the connection and sends the input produced by a user by moving the finger on the phone's screen and by typing in the phone's keyboard. These data are sent to the server application where the data are handled by the DataProcessing class. The DataProcessing class is responsible for manipulating all the data and giving the output as desired by a user. This class gives the user full control over the mouse and the keyboard of the server. On terminating both the server and client applications the socket connection is closed.

4.4 Project Development Process

Various software development models are available, and are generally followed during the software designing process such as agile, spiral and incremental. No particular model was followed for the development of this project. The general steps were followed for the development of this project .The steps followed were requirements analysis, specification, design, code, implementation, testing, documentation and maintenance. Upon finalizing the topic for the thesis project the above mentioned steps were followed simultaneously for the development process. Research work was done to get the ideas on remote desktop application before starting the project. After collecting ideas on remote desktop application, the requirements for the project were analyzed.

The devices, tools, environments and plug-ins required for the development of the project were collected. The specification of the project was that it should be able to control the server computer's mouse and keyboard using Bluetooth. The condition for the usability of the project was that the server and client devices should be within Bluetooth's communication range. The range is generally up to 100 meters depending on the version of Bluetooth. A prototype user interface for the project was designed and the coding was started.

The applications of this project do not follow any software architecture patterns. The client application contains three classes with each class acting as an activity, whereas the server side application is just a .jar file containing three classes running in the computer. The client application was implemented using a Samsung Galaxy III phone while the server application using a Hewlett-Packard laptop. The whole project was then documented and the maintenance process was started. Some of the bugs were fixed whereas some bugs are still present in the application.

5 Testing of the client and server applications

Testing of the project was done by using the applications in different phones and computers. Furthermore the applications were given to my colleagues for testing and the feedback was received. The Android application of this project cannot be used for Android phones with Android version earlier than 2.0. It should be kept in mind that both the server and the client must have properly functioning Bluetooth before testing the project. Also the server and the client must be within the range depending on the version of Bluetooth available in the phone. The maximum working range of recently designed Bluetooth devices are 100 meters. Keeping these facts in mind the testing was done. The test cases are as follows. Table 1 shows the different test cases done for the client application.

Table 1. Test cases for the client application

Devices	**Application's Compatiblity**	**Bluetooth Communication**	**Using the App**	**Range (5 m to 10 m)**	**Range (5 m to 10 m)**
Samsung Galaxy III	Yes	Yes	Works fine	Yes	fluctuation
Samsung Galaxy II	Yes	Yes	Works fine	Yes	fluctuation
Samsung Galaxy S Plus	Yes	Yes	Works fine	Yes	fluctuation
HTC Desire HD	Yes	Yes	Works fine	Yes (a few disturbances)	did not work

Table 1 illustrates the different test cases conducted by using different Android phones. First of all the server was run on the same computer and the client application was tested using different Android phones such as HTC Desire HD, Samsung Galaxy III, Samsung Galaxy II and Samsung Galaxy S Plus. Regarding the compatibility, there were no problems encountered in running the application on different phones. The phones were able to make successful communication and send data to the server.

Using the application in different phones was easy and almost the same as they all had almost the same screen sizes, so there was no alteration in the user interface of the client application. While testing the application by putting the server and the client within a certain range, 10-20 meters, there was an interruption encountered. The connection was weak when the distance was more than 10 meters. The phones were having difficulty connecting to the server. This project worked well within the range of 10 meters between the server and the client. Table 2 shows the different test cases conducted for the server application in different laptops.

Table 2. Different test cases for the server application in different laptops

Devices	**Application's Compatiblity**	**Bluetooth Connection**	**Using the App**	**Range (5 m to 10m)**	**Range (10 m to 20m)**
Hp Pavillion 6799E0	Yes	Yes	Works fine	Yes	fluctuation
Samsung	Yes	Yes	Works fine	Yes	fluctuation

Table 2 illustrates different test cases conducted by using different laptops for the server application. The server application was run on different computers and the project was tested by using the Samsung Galaxy III as the client. This project worked well in all the computers and there was no problem regarding the issues mentioned in the test cases when the range was within 10 meters. For the distance more than 10 meters this project did not work well or did not work at all.

In aggregate this project worked well in all devices within some range. There was no problem in running the server application in the computers. However the client application crashed a few times during the use. The crashing might have been caused by an error occurred in the Bluetooth connection or because of the weak signal. Also the project was given to different users for testing purposes. The users found the applications to be simple and easy to use.

6 Results

The main goal of the project was to develop a remote control system for computers in an Android platform using Bluetooth 4.0 and Java. The client and server applications of this thesis project would enable Bluetooth connection between the phone and computer, and once connected the applications would give the user full control of the computer's mouse and keyboard. The mobile application was developed for Android version 4.2. However it is compatible with the devices with Android version 2.0 and above 2.0. The server application was developed for the computer and is compatible with any computers integrated with Bluetooth. As the applications use Bluetooth to communicate between the client and server devices, the maximum distance between the client and server device should not be more than 100 meters.

The client application was developed using Samsung Galaxy III phone. However different brands of Android phones have different features that might cause the application to behave differently. The client and server applications have been tested using different Android phones and laptops simultaneously. The applications worked well in all the devices when the distance between the client and server was from 1-10 meters. However when the distance between the client and server was from 10 -20 meters, there were a few interruptions encountered in Bluetooth communication between the client and server devices. This project can only be used for the computers that use English language layout keyboard. The application sometimes can crash as Bluetooth might not work properly. Also the controlling of mouse of the server device with phone is not as smooth as it is expected to be.

Thus, despite of several limitations, a functional remote desktop application was developed. However further modification can be done to the server and client applications to fix the errors and to provide additional features.

7 Discussion

A few years ago developing application was meant only for PCs. Hardware and software requirements for PC applications to be able to run are very high. Furthermore applications development requires dedication and involvement of a large number of skilled people. However with the launch of the smartphone the concept has changed. Now applications are also meant for mobile phones. To make mobile applications dedication and involvement of a large number of skilled people is not necessarily required. Such applications can be made by a single person and a application developer can get involved in such development as a profession or as a hobby.

Android has a good and well conducted market for developers to sell their applications. These types of platform have given people chances to expose their skills. Moreover Android being an open source people gets more interested in application development. Open source means that the source code of the applications are available to public for use and modification from its original design. Android has got plenty of guides on setting up the environment for application development, getting stated with the application development and putting developed applications in the market for selling. So applications can be developed easily if developers' guides are followed properly. Also several sensors are available on the Android phones. As this project used Bluetooth, it was found out that Bluetooth can be used for different purposes other than using it as a communicator for a remote control for computers.

This project was developed for an Android device using the software version 4.2. However this application can be used on the phones with the Android version above 2.0. So in the future this project can be further used or modified to run the application on any Android phones. Moreover the application can be modified to a Bluetooth chat application and other applications that use Bluetooth. Also this project can be ported to mobile platforms such as Nokia or Apple by using the high class diagram, sequence diagram and use case diagram following the same logic of the application from the documentation of the project.

The project was completed within the projected time. Though the project is ready for use, it still contains some bugs. The application sometimes can crash as Bluetooth might not work properly. Besides this the controlling of the mouse with the phones is

not as smooth as it is expected to be. The main drawback of the project is that it cannot be used for other layout keyboards except the one in English. This is because Java does not provide key events for the keys for keyboard layouts in language other than English. If the application has to be made for such keyboards, then the key events must be created in Java manually through coding. These bugs can be fixed during the process of debugging. Since the project uses Bluetooth, the battery of the Android phones can drain out quickly. Moreover the client application cannot be used in the phones without having Bluetooth in the phone and also phones using the Android operating system version earlier than 2.0.

Developing the server application was easy compared to the client application as Java has more online as well as printed materials for reference than Android . In the case of the client application, the developers' guide provided by Android was not enough to learn some specific task, so helps had to be taken through third party sites. Android does not have enough and good quality printed books for reference. Developing applications can be a time consuming process. Therefore weekly planning and time management was required to complete the project within the expected time.

Android phones are very popular but still have some drawbacks. The battery drains out rapidly on using the Internet or while using an application that uses the resources of a mobile. So whenever a developer develops an application he or she has to be careful about the battery power drainage. Also in long use of such applications the battery gets slightly heated. Implementing long vibration in games, providing enough brightness during the use of an application and downloading data from the Internet, using Bluetooth can easily drain the battery. Also the emulator or Dalvik VM cannot be used for testing the application that uses sensors and Bluetooth. To test the application that uses sensors and Bluetooth a developer must have Android phone. Regarding these defect developers must have Android phones for testing and running the applications that use sensors and Bluetooth.

More and more new Android phones are being introduced. Every phone does not have the same configurations. So a developed application may not work for all devices. This slightly decreases the possibility of people taking application development as a profession. For those people who work as part time developers or who work alone, selling applications in the Android market does not cover their living expenses. Also regarding the security of the Android market, there have been a few cases where malicious software was hidden in an application and launched to the market. There applications were

capable of sending SMSs, making calls or extracting device-specific information and personal details of the user from the phone without the user's knowledge. These kinds of issues make people cautious and they may not want to download applications from the Android market. As people are showing more interest in application development for mobile phones, Android can have a better future if Google can guarantee the developer to solve the issues discussed above. [39]

The project can be a good reference for developing any Bluetooth related applications. Additional features can be added to the applications. A feature such as viewing the server's screen on the phone can be added. Also this project can be made available for multi-lingual keyboard layouts. The untimely crashing of the application and irregularity in the movement of the mouse can be fixed during future development. Additional features such as copying text between the server and client zoom in/out of the display, onscreen keyboard for missing characters and a multitouch for handling the mouse can be added in the project. During modification issues related to security can also be considered.

During this project useful knowledge and experiences were gained. There was success as well as failure. Despite the defects discussed above, this project was successfully completed. Finally the project was shown to many people, requesting for suggestions necessary to make project better. In the future considering the suggestion the project can further be modified and can be made available for downloads through the Android market.

8 Conclusion

The goal of the project was to develop a remote desktop application using Java with Bluetooth as a communicating medium. The project gives a user full access to the mouse and the keyboard of the server computer to be handled by using Android phones. This project was mainly targeted to acquire ideas on Android and computer applications development using Java. The project was successfully carried out and was completed within the schedule. Despite being available for use, the application has some bugs in it, such as untimely crashing and irregularity in mouse movement and works only for a keyboard with an English layout. Also this project cannot be used on the phones without Bluetooth and the phone with the Android operating system earlier than version 2.0. Bluetooth was successfully implemented into the phone to communicate between the server and the client application.

Simple remote control client and server applications were made using simple Java codes for Android and the computer. In the future this project can further be implemented or modified to create a Bluetooth chat application. Also this project can be developed using Wi-Fi instead of Bluetooth as a communicator between the server and client applications. Furthermore implementing the same logic the project can be used for other purposes such as Bluetooth GPS, Bluetooth multiplayer games, a call making application using Bluetooth or any applications using Bluetooth. Also referring to the high level class diagram and sequence diagram, this project can be ported to other platforms using different programming languages, such as C/C++, QT and Flash etc.

References

1 The Mobile Application Market [online].February 2013
URL: http://www.berginsight.com/ReportPDF/ProductSheet/bi-app1-ps.pdf.
Accessed 26 February 2013.

2 Shane Conder, Lauren Darcey. Android Wireless Application Development,Second Edition. Boston, Addison-Wesley Professional, December 2010

3 Android Developers Guide. Android Architecture. [online].2013
URL: http://developer.android.com/about/versions/index.html.
Accessed 26 February 2013.

4 Android Developers Guide. Application Fundamentals [online].2013
URL: http://developer.android.com/guide/components/fundamentals.html.
Accessed 26 February 2013.

5 Android developers Guide. Activity [online].2013
URL: http://developer.android.com/reference/android/app/Activity.html.
Accessed 27 February 2013.

6 Android Developers Guide. Starting an Activity [online].2013
URL: http://developer.android.com/training/basics/activity-lifecycle/starting.html.
Accessed 27 February 2013.

7 Android Developers Guide. Pausing and Resuming Activity [online].2013
URL: http://developer.android.com/training/basics/activity-lifecycle/pausing.html.
Accessed 27 February 2013.

8 Android Developers Guide. Stopping and Restarting Activity [online].2013
URL: http://developer.android.com/training/basics/activity-lifecycle/stopping.html.
Accessed 27 February 2013.

9 Android Developers Guide. Recreating and Actvity [online].2013
URL: http://developer.android.com/training/basics/activity-lifecycle/recreating.html.
Accessed 27 February 2013.

10 Android Programming: Understanding the Activity Life Cycle [online].March 2011
URL: http://answers.oreilly.com/topic/2692-android-programming-understanding-the- activity-life-cycle/.
Accessed 27 February 2013.

11 technopedia. Android SDK [online].2013
URL: http://www.techopedia.com/definition/4220/android-sdk.
Accessed 1 March 2013.

12 Android Developers Guide. Get the Android SDK[online].2013
URL: http://developer.android.com/sdk/index.html.
Accessed 1 March 2013.

13 Den McKenzie. Designing For Android[online].June 2011
URL: http://coding.smashingmagazine.com/20 11/06/30/designing-for-android/.
Accessed 1 March 2013.

14 FreeWiMAXInfo.com. What is Touch Screen Technology [online].2012
URL: http://freewimaxinfo.com/touch-screen.html.
Accessed 1 March 2013.

15 Punchcut. Design consideration for UI [online].April 2010
URL: http://punchcut.com/perspectives/videos/design-considerations-touch-ui.
Accessed 1 March 2013.

16 Techotopia. Understanding Android Views, View Groups and Layouts [online].May 2012
URL: http://www.techotopia.com/index.php/Understanding_Android_Views,_View_Groups_and_Layouts.
Accessed 2 March 2013.

17 IT&C Solutions. Procedural vs. Declarative Design of User Interfaces [online].2010
URL: https://publications.theseus.fi/handle/10024/15152.
Accessed 2 March 2013.

18 APC. Building a simple Android app [online].October 2011
URL: http://apcmag.com/building-a-simple-android-app.htm.
Accessed 2 March 2013.

19 Hashimi S, Komatineni S, MacLean D. Pro Android 2. New York, NY: Apress; 2010

20 Techradar.pro. What is Bluetooth? [online].February 2012
URL: http://www.techradar.com/news/phone-and-communications/mobile-phones/what-is-bluetooth-1063913.
Accessed 4 March 2013.

21 C Bala Kumar, Paul J. Kline, Timonthy J. Thompson. Bluetooth Application Programming with the Java APIs. San Francisco, 2004.

22 Specification of the Bluetooth Systems, Volume 1 [online].February 2001
URL: http://www.inf.ethz.ch/personal/hvogt/proj/btmp3/Datasheets/Bluetooth_11_Specifications_Book.pdf.
Accessed 4 March 2013.

23 Palowireles. Bluetooth Tutorial –specifications [online].
URL: http://www.palowireless.com/infotooth/tutorial.asp.
Accessed 5 March 2013.

24 Specification of the Bluetooth Systems, Volume 2 [online].December 1999
URL: http://grouper.ieee.org/groups/802/15/Bluetooth/profile_10_b.pdf.
Accessed 6 March 2013.

25 Palowireles. Bluetooth Tutorial –Profiles [online].
URL: http://www.palowireless.com/infotooth/tutorial/profiles.asp.
Accessed 6 March 2013.

26 Nokia Developer. Bluetooth Protocol [online].July 2012
URL: http://www.developer.nokia.com/Community/Wiki/Bluetooth_Protocol.
Accessed 7 March 2013.

27 OReillyonJava.com, What is Java [online].2006
URL: http://onjava.com/pub/a/onjava/2006/03/08/what-is-java.html.
Accessed 8 March 2013.

28 Javatech, Features [online].October 2003
URL: http://www.particle.kth.se/~lindsey/JavaCourse/Book/Part1/Java/Chapter01/features.html.
Accessed 8 March 2013.

29 IBM, What is Eclipse, and how do I use it? [online].November 2001
URL: http://www.ibm.com/developerworks/opensource/library/os-eclipse/index.html.
Accessed 8 March 2013.

30 Eclipse, What is Eclipse? [online].
URL: http://help.eclipse.org/galileo/index.jsp?topic=/org.eclipse.platform.doc.isv/guide/int_eclipse.htm.
Accessed 9 March 2013.

31 The Architecture of Open Spurce Applications, Eclipse [online].September 2012
URL: http://www.aosabook.org/en/eclipse.html.
Accessed 9 March 2013.

32 IBM, Fit and Eclipse [online].June 2006
URL: http://www.ibm.com/developerworks/aix/library/au-fiteclipse2/.
Accessed 9 March 2013.

33 Eclipse Platform Technical Overview [online].2006
URL: http://www.eclipse.org/articles/Whitepaper-Platform-3.1/eclipse-platform-whitepaper.html.
Accessed 9 March 2013

34 Joomla, NetBeans overview [online].January 2013
URL: http://docs.joomla.org/NetBeans_overview.
Accessed 10 March 2013.

35 The Dalvik Virtual Machine Architecture [online].March 2010
URL: http://elastos.org/elorg_files/FreeBooks/dalvik/The_Dalvik_Virtual_Machine.pdf.
Accessed 10 March 2013.

36 Android Developer Guide, Managing Projects [online].2013
URL: http://developer.android.com/tools/projects/index.html.
Accessed 10 March 2013.

37 Oracle, JAR file specification [online].2011
URL: http://docs.oracle.com/javase/6/docs/technotes/guides/jar/jar.html.
Accessed 10 March 2013.

38 Meier R . Professional Android 2 application development. Indianapolis, IN: Wiley publishing, Inc.;2010

39 Geere D. Google Android: Its History and uncertain future [online]. April 2010 Accessed 12 April 2010.