End Semester Evaluation

Minor Project (IT-413) Report

On

Remote Control PC

Submitted in partial fulfillment for the award of the degree

of

Bachelor of Technology

In

Information Technology

National Institute of Technology, Kurukshetra

Project By:

Varun Kumar - 1130328, IT - 2
Kuldeep Kumar - 1130828, IT - 6
Parmod Kumar - 1130644, IT - 4
B.Tech. 7th semester, Project Group: 7

Supervisor:

Dr. Mantosh Biswas
Department of Computer Engineering
NIT Kurukshetra

Table of Contents

1. Introduction

Android is a mobile operating system developed by Google, based on the Linux kernel and designed primarily for touchscreen mobile devices such as smartphones and tablets. Android's user interface is mainly based on direct manipulation, using touch gestures that loosely correspond to real-world actions, such as swiping, tapping and pinching, to manipulate on-screen objects, along with a virtual keyboard for text input. In addition to touchscreen devices, Google has further developed Android TV for televisions, Android Auto for cars, and Android Wear for wrist watches, each with a specialized user interface. Variants of Android are also used on notebooks, game consoles, digital cameras, and other electronics[1].

A socket is the mechanism that most popular operating systems provide to give programs access to the network. It allows messages to be sent and received between applications (unrelated processes) on different networked machines. The sockets mechanism has been created to be independent of any specific type of network. IP, however, is by far the most dominant network and the most popular use of sockets[2].

Remote Control PC uses TCP Socket to control PC using Android Phone. It consists of two parts. - an android app running on Phone and a desktop app running on PC. User performs any action on phone. This data goes to server i.e. desktop app and same action is simulated on PC. We have used Java technology in this project.

2. Motivation

There exist several situations where we want to wirelessly and comfortably operate a computer, where the computer screen is projected onto a big screen through a projector or big-screen television, such as classrooms, conference/meeting rooms, mobile, workgroup project environments and modern office environments, and even living rooms. Several specifically designed devices are available on the market for the purpose of operating computers remotely and wirelessly. Wireless keyboard, uses either Bluetooth or wireless USB mini-receiver plugged into the USB port of computer for the communication between the keyboard and the computer. Some wireless keyboards have a touchpad for controlling the mouse cursor. Wireless presentation controller, allows user to operate his/her computer remotely for PowerPoint presentation through Bluetooth connection.

However, all those devices have certain drawbacks. Wireless keyboard has limited flexibility and is not convenient for a presenter to carry it around in the room during the presentation. Presenters usually like to walk around while presenting. Carrying a wireless keyboard is definitely not convenient. Wireless presentation controller does have good mobility. However, most of such devices do not allow user to have full operation on the computer, such as running a program, moving or closing an application window, etc. Even it has a small touchpad for moving mouse cursor, however it is very difficult for the presenter to use it to move the mouse cursor while he/she is walking around[3].

The widely used and very popular smart devices, such as iPads, smartphones, PDAs, and smart game controllers, can be the excellent alternatives as computer remote controllers if we develop appropriate apps for them. This motivated us to develop an app for most popular mobile os android to control action on PC.

3. Problem Statement

To smoothly control cursor and keyboard activities on laptop/pc using android phone. To send and receive data from phone and laptop. To transfer android files to desktop as well as to download desktop files to android. To play media player on fly and to view gallery controlled by android. To control presentation on desktop and to enable shutdown of computer from android.

4. Objectives

To provide following features-

A. Control Left Click, Right Click, Mouse Scroll
B. Type text
C. Transfer files from phone to laptop
D. Download files from laptop to phone
E. Use laptop as speaker to play mp3 files of phone
F. See images of phone on laptop
G. Control presentation on laptop via phone
H. Suspend, Restart or Shutdown laptop using phone

App must be reliable and there must be nice user interface and user experience (UI and UX). Android / Desktop app must be lightweight and must not consume excess resources (memory, CPU, power etc).

5. Brief Description of project

Remote Control PC consists of following characteristics-

A. It has been developed in Java
B. It use TCP Socket (Via wifi) to establish connection between phone and laptop.
C. It consists of two parts- android app for phone and desktop app for laptop.
D. It is platform independent.

Working Model:

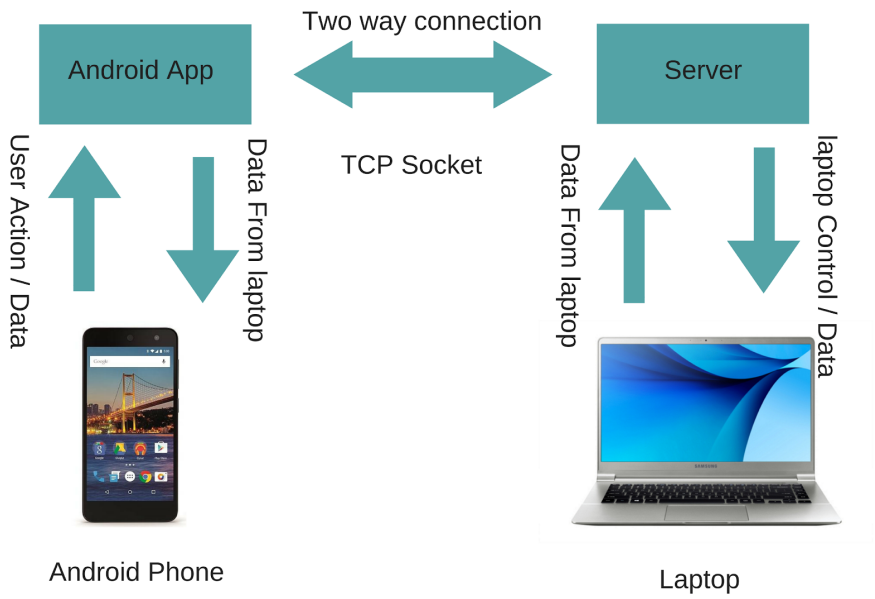

Fig 6: Working Model of Remote Control PC

A socket connection is established between android device and desktop. This two way connection is used to send as well as receive data from phone and PC. Any user activities on phone is simulated on PC using desktop app.

5.1 Screenshots-

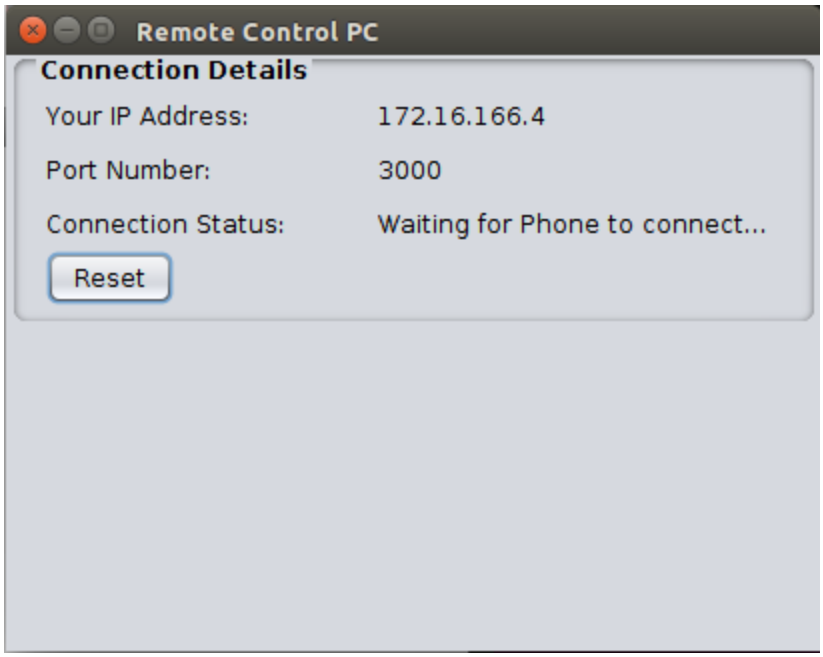

Fig 1: Server running on Desktop

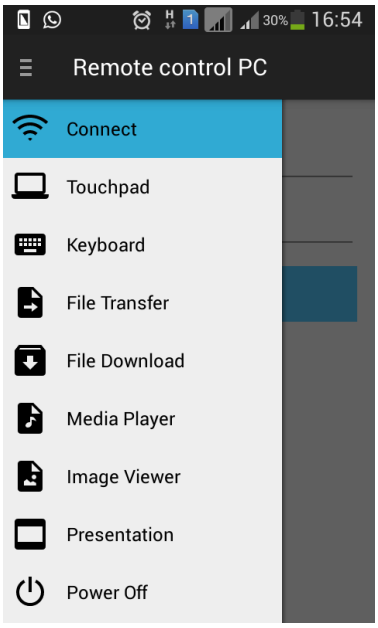

Fig 2: Navigation (Android)

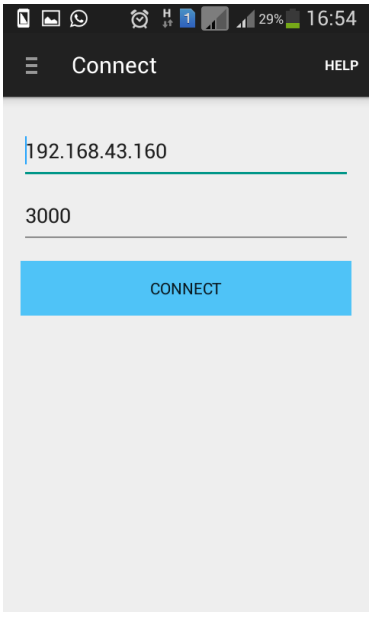

Fig 3: Connect Screen on Android

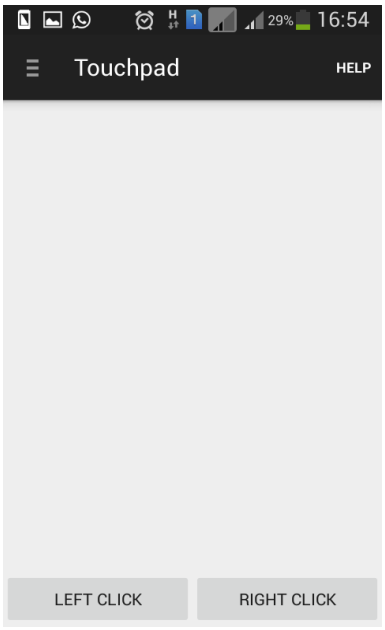

Fig 4: Touchpad to control mouse

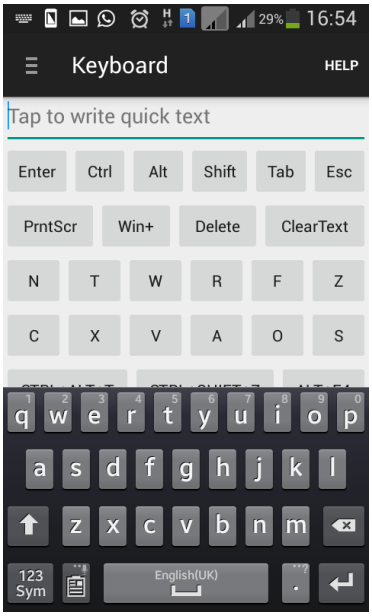

Fig 5: Keyboard Screen

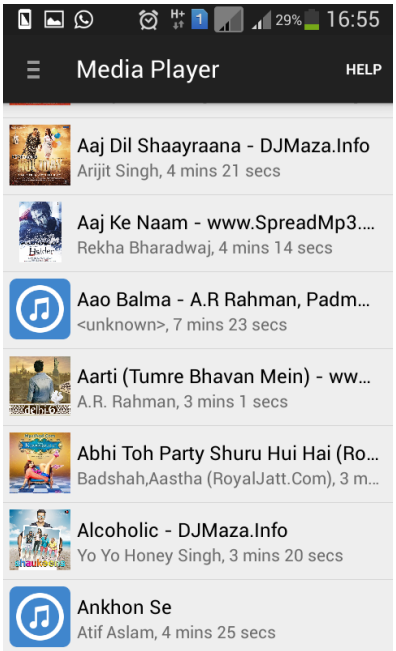

Fig 6: Media Player Screen

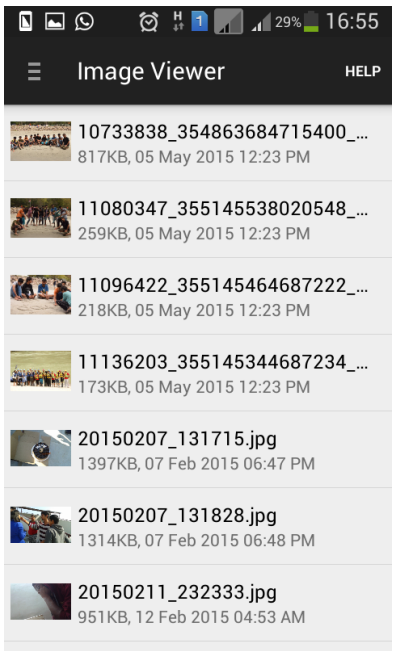

Fig 7: Image Viewer Screen

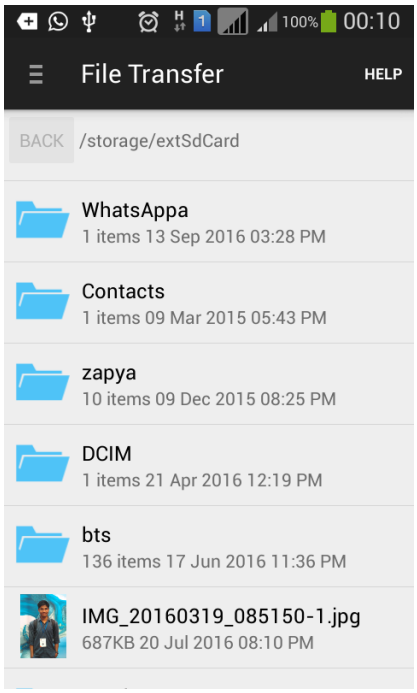

Fig 8: File Transfer Screen

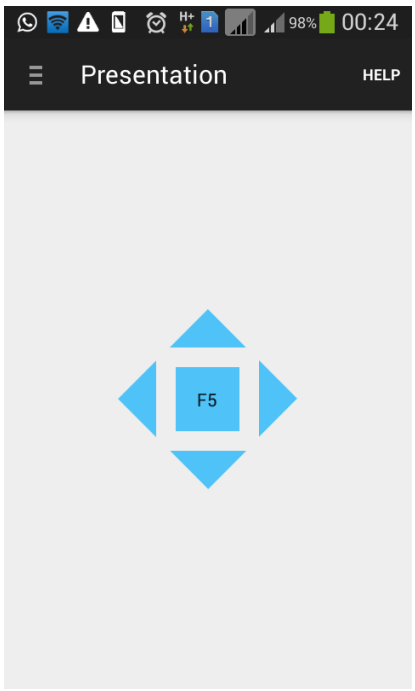

Fig 9: Presentation Control Screen

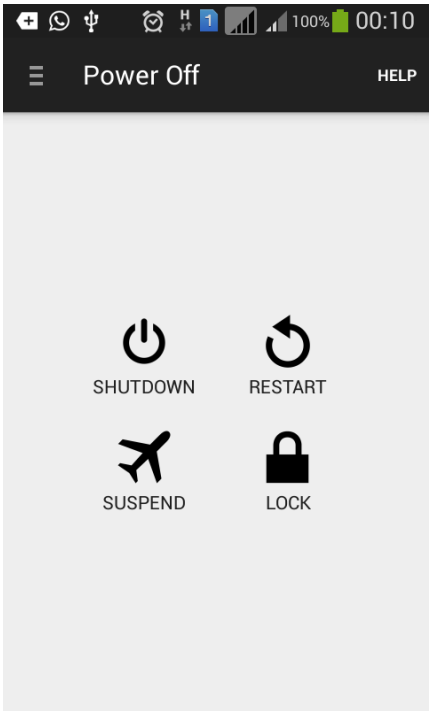

Fig 10: Power Off Screen

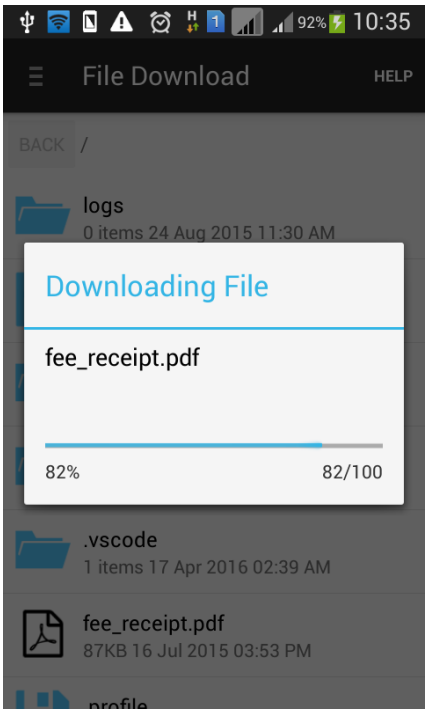

Fig 11: File Download Screen

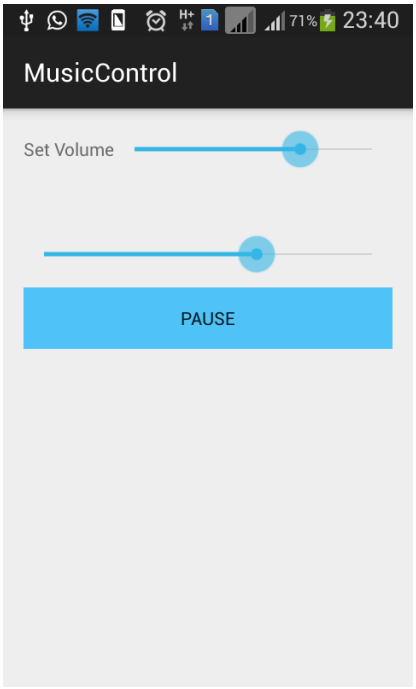

Fig 12: Music Control Screen

5.2 How to connect-

A. Start hotspot on phone and connect your laptop via wifi

B. Open the desktop app on your laptop

C. Open Android app and enter connection details provided by desktop app to connect

5.3 App Dependencies

A. User must have JRE 8 installed

B. Only Android mobiles are supported

C. Working wifi/hotspot is necessary

5.4 Limitations-

A. Media Player and Image Viewer work only after transferring complete file. There is no live streaming. This cause some delay depending on file size.

B. Shutdown, Restart and Suspend features do not work on Linux due to security issues.

6. Implementation Details

The complete project is divided into three parts-

A. RemoteControlPC-Android: Android part of the project which has been developed in Eclipse.
B. RemoteControlPC-Desktop: Desktop part of the project which has been developed in Netbeans.
C. RemoteControlPC-Libraries: Library (jar file) used by both android as well as desktop part. It has been developed in Netbeans.

6.1 Packages and Classes structure-

Desktop Part-

Java Packages	**Java Classes**
image	ImageViewer
music	MusicPlayer
remotecontrolpc.desktop	MainScreen Utility
remotecontrolpc.desktop.filesharing	FileAPI ReceiveFile SendFile SendFilesList
remotecontrolpc.desktop.ipaddress	GetFreePort GetMyIpAddress
remotecontrolpc.desktop.mousekeyboardcontrol	MouseKeyboardControl
remotecontrolpc.desktop.poweroff	PowerOff
remotecontrolpc.desktop.server	Server

Library-

Java Packages	**Java Classes**
file	AvatarFile

Android Part-

Java Packages	Java Classes
com.example.remotecontrolpc	AvatarFile AvatarFileAdapter CallbackReceiver FileAPI HelpActivity MainActivity MusicControlActivity MusicImageAvatar MusicImageAvatarAdapter NavigationDrawerFragment NavigationDrawerItem NavigationDrawerItemAdapter Utility
com.example.remotecontrolpc.connect	ConnectFragment MakeConnection ValidateIP
com.example.remotecontrolpc.filedownload	DownloadFileFromServer FileDownloadFragment GetFilesListFromServer
com.example.remotecontrolpc.filetransfer	FilesList FileTransferFragment TransferFileToServer
com.example.remotecontrolpc.help	HelpFragment
com.example.remotecontrolpc.imageviewer	ImagesList ImageViewerFragment
com.example.remotecontrolpc.keyboard	KeyboardFragment
com.example.remotecontrolpc.mediaplayer	MediaPlayerFragment SongsList
com.example.remotecontrolpc.poweroff	PowerOffFragment
com.example.remotecontrolpc.presentation	PresentationFragment
com.example.remotecontrolpc.touchpad	TouchpadFragment

6.2 Activity Diagram-

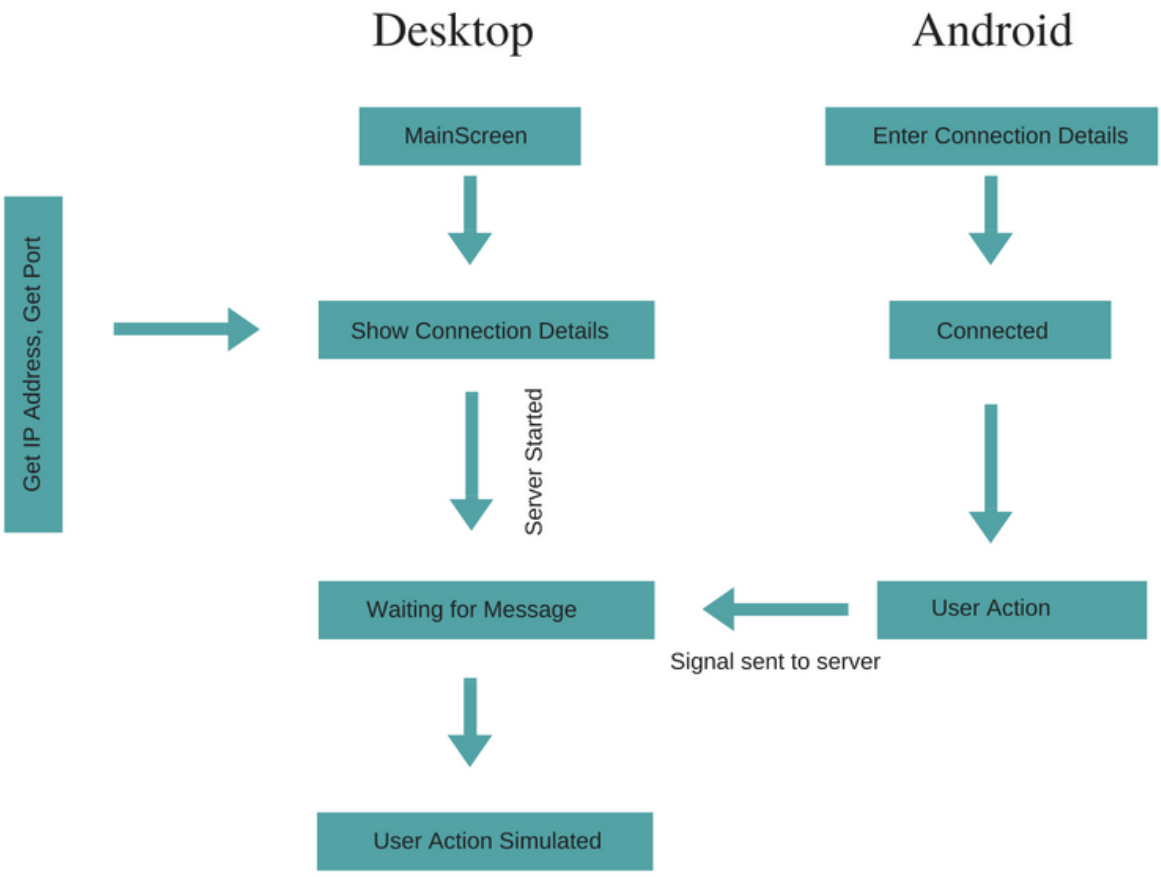

Fig 13: Activity Diagram

6.3 Socket Streams-

A socket is an abstraction that we use to talk to something across the network. In Java, to send data via the socket, we get an OutputStream from it, and write to the OutputStream (we output some data). To read data from the socket, we get its InputStream, and read input from this second stream. We can think of the streams as a pair of one-way pipes connected to a socket on the wall. What happens on the other side of the wall is not our problem! In our case, the server has another socket (the other end of the connection) and another pair of streams. It uses its InputStream to read from the network, and its OutputStream to write the same data back across the network to our client, which reads it again via its InputStream completing the round trip[7].

Code for socket stream in Java-

```
        MainScreen.clientSocket = MainScreen.serverSocket.accept();
        MainScreen.inputStream = MainScreen.clientSocket.getInputStream();
        MainScreen.outputStream = MainScreen.clientSocket.getOutputStream();
        MainScreen.objectOutputStream = new ObjectOutputStream(MainScreen.outputStream);
        MainScreen.objectInputStream = new ObjectInputStream(MainScreen.inputStream);
```

6.4 Server Code-

```
public void connect(JButton resetButton, JLabel connectionStatusLabel) {
        MouseKeyboardControl mouseControl = new MouseKeyboardControl();
        try {
            connectionStatusLabel.setText("Waiting for Phone to connect...");
            MainScreen.clientSocket = MainScreen.serverSocket.accept();
            resetButton.setEnabled(false);
            connectionStatusLabel.setText("Connected to: " +
                    MainScreen.clientSocket.getRemoteSocketAddress());
            MainScreen.inputStream = MainScreen.clientSocket.getInputStream();
            MainScreen.outputStream = MainScreen.clientSocket.getOutputStream();
            MainScreen.objectOutputStream = new ObjectOutputStream(MainScreen.outputStream);
            MainScreen.objectInputStream = new ObjectInputStream(MainScreen.inputStream);
            FileAPI fileAPI = new FileAPI();
            String message, filePath, fileName;
            int slideDuration;
            float volume;
            PowerOff  powerOff = new PowerOff();
            MusicPlayer musicPlayer = new MusicPlayer();
            ImageViewer imageViewer = new ImageViewer();
            while (true) {
                try {
                    message = (String) MainScreen.objectInputStream.readObject();
                    int keyCode;
                    if (message != null) {
                        switch (message) {
                            case "LEFT_CLICK":
                                mouseControl.leftClick();
                                break;
                            case "RIGHT_CLICK":
                                mouseControl.rightClick();
                                break;
                            case "MOUSE_WHEEL":
                                int scrollAmount = (int) MainScreen.objectInputStream.readObject();
                                mouseControl.mouseWheel(scrollAmount);
                                break;
                            case "MOUSE_MOVE":
                                int x = (int) MainScreen.objectInputStream.readObject();
                                int y = (int) MainScreen.objectInputStream.readObject();
                                Point point = MouseInfo.getPointerInfo().getLocation();
                                float nowx = point.x;
                                float nowy = point.y;
                                mouseControl.mouseMove((int) (nowx + x), (int) (nowy + y));
                                break;
                            case "KEY_PRESS":
                                keyCode = (int) MainScreen.objectInputStream.readObject();
                                mouseControl.keyPress(keyCode);
                                break;
                            case "KEY_RELEASE":
                                keyCode = (int) MainScreen.objectInputStream.readObject();
                                mouseControl.keyRelease(keyCode);
```

```
            break;
        case "CTRL_ALT_T":
            mouseControl.ctrlAltT();
            break;
        case "CTRL_SHIFT_Z":
            mouseControl.ctrlShiftZ();
            break;
        case "ALT_F4":
            mouseControl.altF4();
            break;
        case "TYPE_CHARACTER":
            char ch = ((String) MainScreen.objectInputStream.readObject()).charAt(0);
            mouseControl.typeCharacter(ch);
            break;
        case "TYPE_KEY":
            keyCode = (int) MainScreen.objectInputStream.readObject();
            mouseControl.typeCharacter(keyCode);
            break;
        case "LEFT_ARROW_KEY":
            mouseControl.pressLeftArrowKey();
            break;
        case "DOWN_ARROW_KEY":
            mouseControl.pressDownArrowKey();
            break;
        case "RIGHT_ARROW_KEY":
            mouseControl.pressRightArrowKey();
            break;
        case "UP_ARROW_KEY":
            mouseControl.pressUpArrowKey();
            break;
        case "F5_KEY":
            mouseControl.pressF5Key();
            break;
        case "FILE_DOWNLOAD_LIST_FILES":
            filePath = (String) MainScreen.objectInputStream.readObject();
            if (filePath.equals("/")) {
                filePath = fileAPI.getHomeDirectoryPath();
            }
            new SendFilesList().sendFilesList(fileAPI, filePath, MainScreen.objectOutputStream);
            break;
        case "FILE_DOWNLOAD_REQUEST":
            //filePath is complete path including file name
            filePath = (String) MainScreen.objectInputStream.readObject();
            new SendFile().sendFile(filePath, MainScreen.objectOutputStream);
            break;
        case "FILE_TRANSFER_REQUEST":
            fileName = (String) MainScreen.objectInputStream.readObject();
            //not in thread, blocking action
            new ReceiveFile().receiveFile(fileName);
            break;
        case "SHUTDOWN_PC":
            powerOff.shutdown();
            break;
        case "RESTART_PC":
            powerOff.restart();
            break;
        case "SLEEP_PC":
            powerOff.suspend();
            break;
        case "LOCK_PC":
```

```
                        powerOff.lock();
                        break;
                    case "PLAY_MUSIC":
                        fileName = (String) MainScreen.objectInputStream.readObject();
                        filePath = new FileAPI().getHomeDirectoryPath();
                        filePath = filePath + "/RemoteControlPC/" + fileName;
                        musicPlayer.playNewMedia(filePath);
                        break;
                    case "SLIDE_MUSIC":
                        slideDuration = (int) MainScreen.objectInputStream.readObject();
                        musicPlayer.slide(slideDuration);
                        break;
                    case "PAUSE_OR_RESUME_MUSIC":
                        musicPlayer.resumeOrPauseMedia();
                        break;
                    case "STOP_MUSIC":
                        musicPlayer.stopMusic();
                        break;
                    case "SET_VOLUME_MUSIC":
                        volume = (float) MainScreen.objectInputStream.readObject();
                        musicPlayer.setVolume(volume);
                        break;
                    case "SHOW_IMAGE":
                        fileName = (String) MainScreen.objectInputStream.readObject();
                        filePath = new FileAPI().getHomeDirectoryPath();
                        filePath = filePath + "/RemoteControlPC/" + fileName;
                        imageViewer.showImage(fileName, filePath);
                        break;
                    case "CLOSE_IMAGE_VIEWER":
                        //closing this close music player also
                        //imageViewer.closeImageViewer();
                        break;
                }
            } else {
                //remote connection closed
                connectionClosed();
                resetButton.setEnabled(true);
                connectionStatusLabel.setText("Disconnected");
                break;
            }
        } catch (Exception e) {
            e.printStackTrace();
            connectionClosed();
            resetButton.setEnabled(true);
            connectionStatusLabel.setText("Disconnected");
            break;
        }
    };
}
catch(Exception e) {
    e.printStackTrace();
}
}
```

7. Project Status

All the proposed functionalities has been successfully implemented and tested. Below is the Activity Time chart for Remote Control PC -

A. Project Finalised: 6th September
B. Project Imported in Netbeans and Eclipse: 17th October
C. Keyboard Fragment Added: 20th October
D. Touchpad Fragment Added: 20th October
E. File Transfer Added: 13th November
F. Power Off Fragment Added: 13th November
G. Presentation Fragment Added: 14th November
H. File Download Added: 18th November
I. Power Off Bug Fixed: 19th November
J. Image Viewer Added: 20th November
K. Music Control Added: 21th November
L. App successfully tested in multiple Android Devices: 24th November

8. Software and Tools used

8.1 Netbeans IDE

NetBeans is a software development platform written in Java. The NetBeans Platform allows applications to be developed from a set of modular software components called modules. Applications based on the NetBeans Platform, including the NetBeans integrated development environment (IDE), can be extended by third party developers.

The NetBeans IDE is primarily intended for development in Java, but also supports other languages, in particular PHP, C/C++ and HTML5.

NetBeans is cross-platform and runs on Microsoft Windows, Mac OS X, Linux, Solaris and other platforms supporting a compatible JVM.

The NetBeans Team actively support the product and seek feature suggestions from the wider community. Every release is preceded by a time for Community testing and feedback.

NetBeans began in 1996 as Xelfi (word play on Delphi),[7][8] a Java IDE student project under the guidance of the Faculty of Mathematics and Physics at Charles University in Prague. In 1997, Roman Staněk formed a company around the project and produced commercial versions of the NetBeans IDE until it was bought by Sun Microsystems in 1999. Sun open-sourced the NetBeans IDE in June of the following year. Since then, the NetBeans community has continued to grow.[9] In 2010, Sun (and thus NetBeans) was acquired by Oracle Corporation.

Under Oracle, NetBeans competed with JDeveloper, a freeware IDE that has historically been a product of the company. In September 2016, Oracle submitted a proposal to donate the NetBeans project to the Apache Software Foundation, stating that it was "opening up the NetBeans governance model to give NetBeans constituents a greater voice in the project's direction and future success through the upcoming release of Java 9 and NetBeans 9 and beyond". The move was endorsed by Java creator James Gosling.

NetBeans IDE 6.0 introduced support for developing IDE modules and rich client applications based on the NetBeans platform, a Java Swing GUI builder (formerly known as "Project Matisse"), improved CVS support, WebLogic 9 and JBoss 4 support, and many editor enhancements. NetBeans 6 is available in official repositories of major Linux distributions.

NetBeans IDE 6.5, released in November 2008, extended the existing Java EE features (including Java Persistence support, EJB 3 and JAX-WS). Additionally, the NetBeans Enterprise Pack supports the development of Java EE 5 enterprise applications, including SOA visual design tools, XML schema tools, web services orchestration (for BPEL), and UML modeling. The NetBeans IDE Bundle for C/C++ supports C/C++ and FORTRAN development.

NetBeans IDE 6.8 is the first IDE to provide complete support of Java EE 6 and the GlassFish Enterprise Server v3. Developers hosting their open-source projects on kenai.com additionally benefit from instant messaging and issue tracking integration and navigation right in the IDE, support for web application development with PHP 5.3 and the Symfony framework, and improved code completion, layouts, hints and navigation in JavaFX projects.

NetBeans IDE 6.9, released in June 2010, added support for OSGi, Spring Framework 3.0, Java EE dependency injection (JSR-299), Zend Framework for PHP, and easier code navigation (such as "Is Overridden/Implemented" annotations), formatting, hints, and refactoring across several languages.

NetBeans IDE 7.0 was released in April 2011. On August 1, 2011, the NetBeans Team released NetBeans IDE 7.0.1, which has full support for the official release of the Java SE 7 platform.

NetBeans IDE 7.3 was released in February 2013 which added support for HTML5 and web technologies.

NetBeans IDE 7.4 was released on October 15, 2013.

NetBeans IDE 8.0 was released on March 18, 2014.

NetBeans IDE 8.1 was released on November 4, 2015.

NetBeans IDE 8.2 was released on October 3, 2016.

NetBeans has a roadmap document for release plans

8.2 Eclipse IDE

Eclipse is an integrated development environment (IDE) used in computer programming, and is the most widely used Java IDE.[3] It contains a base workspace and an extensible plug-in system for customizing the environment. Eclipse is written mostly in Java and its primary use is for developing Java applications, but it may also be used to develop applications in other programming languages through the use of plugins, including: Ada, ABAP, C, C++, COBOL, D, Fortran, Haskell, JavaScript, Julia,[4] Lasso, Lua, NATURAL, Perl, PHP, Prolog, Python, R, Ruby (including Ruby on Rails framework), Rust, Scala, Clojure, Groovy, Scheme, and Erlang. It can also be used to develop documents with LaTeX (through the use of the TeXlipse plugin) and packages for the software Mathematica. Development environments include the Eclipse Java development tools (JDT) for Java and Scala, Eclipse CDT for C/C++ and Eclipse PDT for PHP, among others.

The initial codebase originated from IBM VisualAge.[5] The Eclipse software development kit (SDK), which includes the Java development tools, is meant for Java developers. Users can extend its abilities by installing plug-ins written for the Eclipse Platform, such as development toolkits for other programming languages, and can write and contribute their own plug-in modules.

Released under the terms of the Eclipse Public License, Eclipse SDK is free and open-source software, although it is incompatible with the GNU General Public License.[6] It was one of the first IDEs to run under GNU Classpath and it runs without problems under IcedTea.

Eclipse was inspired by the Smalltalk-based VisualAge family of integrated development environment (IDE) products.[5] Although fairly successful, a major drawback of the VisualAge products was that developed code was not in a component model; instead, all code for a project was held in a compressed lump (somewhat like a zip file but in a proprietary format called .dat); individual classes could not be easily accessed, certainly not outside the tool. A team primarily at the IBM Cary NC lab developed the new product as a Java-based replacement.[7] In November 2001, a consortium was formed with a board of stewards to further the development of Eclipse as open-source software. It is estimated that IBM had already invested close to $40 million by that time.[8] The original members were Borland, IBM, Merant, QNX Software Systems, Rational Software, Red Hat, SuSE, TogetherSoft and WebGain.[9] The number of stewards increased to over 80 by the end of 2003. In January 2004, the Eclipse Foundation was created.

Eclipse 3.0 (released on 21 June 2004) selected the OSGi Service Platform specifications as the runtime architecture.

The Association for Computing Machinery recognized Eclipse with the 2011 ACM Software Systems Award on 26 April 2012.

The Eclipse Public License (EPL) is the fundamental license under which Eclipse projects are released. Some projects require dual licensing, for which the Eclipse Distribution License (EDL) is available, although use of this license must be applied for and is considered on a case-by-case basis.

Eclipse was originally released under the Common Public License, but was later re-licensed under the Eclipse Public License. The Free Software Foundation has said that both licenses are free software licenses, but are incompatible with the GNU General Public License (GPL).

8.3 Core Java

Java is a set of computer software and specifications developed by Sun Microsystems, which was later acquired by the Oracle Corporation, that provides a system for developing application software and deploying it in a cross-platform computing environment. Java is used in a wide variety of computing platforms from embedded devices and mobile phones to enterprise servers and supercomputers. While they are less common than standalone Java applications, Java applets run in secure, sandboxed environments to provide many features of native applications and can be embedded in HTML pages.

Writing in the Java programming language is the primary way to produce code that will be deployed as byte code in a Java Virtual Machine (JVM); byte code compilers are also available for other languages, including Ada, JavaScript, Python, and Ruby. In addition, several languages have been designed to run natively on the JVM, including Scala, Clojure and Apache Groovy. Java syntax borrows heavily from C and C++, but object-oriented features are modeled after Smalltalk and Objective-C. Java eschews certain low-level constructs such as pointers and has a very simple memory model where every object is allocated on the heap and all variables of object types are references. Memory management is handled through integrated automatic garbage collection performed by the JVM.

On November 13, 2006, Sun Microsystems made the bulk of its implementation of Java available under the GNU General Public License (GPL).

The latest version is Java 8, the only supported (with e.g. security updates) version as of 2016. Oracle (and others) has announced that using older versions (than Java 8) of their JVM implementation presents serious risks due to unresolved security issues.

8.4 JavaFX

JavaFX is a software platform for creating and delivering desktop applications, as well as rich internet applications (RIAs) that can run across a wide variety of devices. JavaFX is intended to replace Swing as the standard GUI library for Java SE, but both will be included for the foreseeable future. JavaFX has support for desktop computers and web browsers on Microsoft Windows, Linux, and Mac OS X.

Before version 2.0 of JavaFX, developers used a statically typed, declarative language called JavaFX Script to build JavaFX applications. Because JavaFX Script was compiled to Java bytecode, programmers could also use Java code instead. JavaFX applications could run on any desktop that could run Java SE, on any browser that could run Java EE, or on any mobile phone that could run Java ME.

JavaFX 2.0 and later is implemented as a native Java library, and applications using JavaFX are written in native Java code. JavaFX Script has been scrapped by Oracle, but development is being continued in the Visage project. JavaFX 2.x does not support the Solaris operating system or mobile phones; however, Oracle plans to integrate JavaFX to Java SE Embedded 8, and Java FX for ARM processors is in developer preview phase.

On desktops, JavaFX supports Windows XP, Windows Vista, Windows 7, Windows 8, Windows 10, Mac OS X and Linux operating systems. Beginning with JavaFX 1.2, Oracle has released beta versions for OpenSolaris. On mobile, JavaFX Mobile 1.x is capable of running on multiple mobile operating systems, including Symbian OS, Windows Mobile, and proprietary real-time operating systems.

JavaFX 1.1 was based on the concept of a "common profile" that is intended to span across all devices supported by JavaFX. This approach makes it possible for developers to use a common programming model while building an application targeted for both desktop and mobile devices and to share much of the code, graphics assets and content between desktop and mobile versions.

To address the need for tuning applications on a specific class of devices, the JavaFX 1.1 platform includes API that are desktop or mobile-specific. For example JavaFX Desktop profile includes Swing and advanced visual effects.

8.5 Java Swing

Swing is a GUI widget toolkit for Java. It is part of Oracle's Java Foundation Classes (JFC) – an API for providing a graphical user interface (GUI) for Java programs.

Swing was developed to provide a more sophisticated set of GUI components than the earlier Abstract Window Toolkit (AWT). Swing provides a native look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check boxes and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables, and lists.

Unlike AWT components, Swing components are not implemented by platform-specific code. Instead, they are written entirely in Java and therefore are platform-independent. The term "lightweight" is used to describe such an element.

Swing is slowly being replaced by JavaFX

The Internet Foundation Classes (IFC) were a graphics library for Java originally developed by Netscape Communications Corporation and first released on December 16, 1996. On April 2, 1997, Sun Microsystems and Netscape Communications Corporation announced their intention to incorporate IFC with other technologies to form the Java Foundation Classes. The "Java Foundation Classes" were later renamed "Swing."

Swing introduced a mechanism that allowed the look and feel of every component in an application to be altered without making substantial changes to the application code. The introduction of support for a pluggable look and feel allows Swing components to emulate the appearance of native components while still retaining the benefits of platform independence. Originally distributed as a separately downloadable library, Swing has been included as part of the Java Standard Edition since release 1.2 The Swing classes and components are contained in the javax.swing package hierarchy.

Swing is a platform-independent, Model-View-Controller GUI framework for Java, which follows a single-threaded programming model. Additionally, this framework provides a layer of abstraction between the code structure and graphic presentation of a Swing-based GUI.

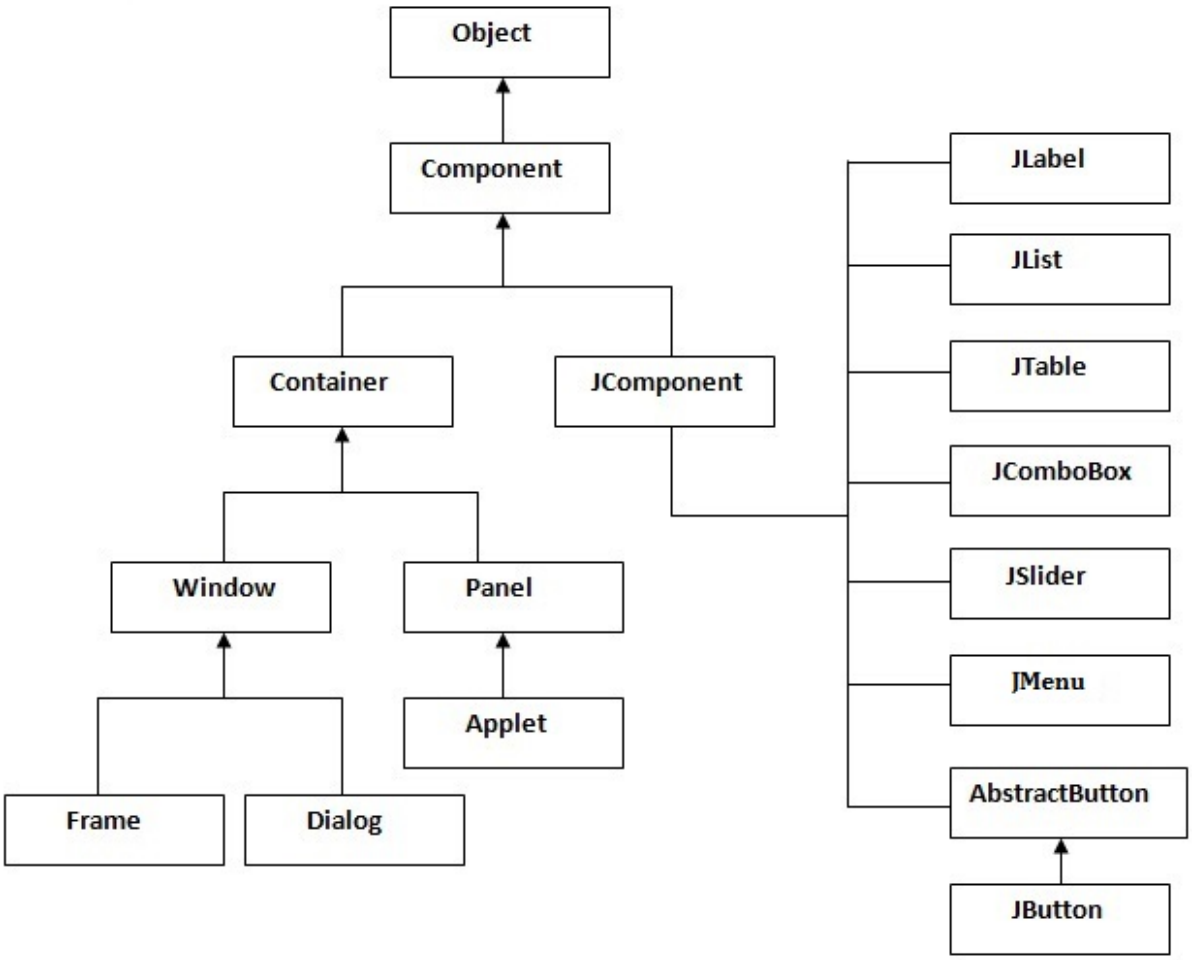

Fig 14: Hierarchy of Java Swing class

8.6 Android

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

The unveiling of the Android platform on 5 November 2007 was announced with the founding of the Open Handset Alliance, a consortium of 34 hardware, software and telecom companies devoted to advancing open standards for mobile devices. When released in 2008, most of the Android platform will be made available under the Apache free-software and open-source license.

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

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

Google has unveiled at least three prototypes for Android, at the Mobile World Congress on February 12, 2008. One prototype at the ARM booth displayed several basic Google applications. A 'd-pad' control zooming of items in the dock with a relatively quick response.

9. Conclusion and Future Work

This report describes how to turn smart devices, more specifically smartphones, into computer remote controllers. The system presented above can be widely used in classrooms and meeting/conference rooms for presentation and interactive discussion. Currently we are exploring approaches of using smart devices as controllers or operators for other devices. We are also exploring possibilities of developing app for desktop which will control mobile functionalities and apps e.g calling, texting from desktop app.

References

[1] https://en.wikipedia.org/wiki/Android_(operating_system)

[2] http://www.cs.rutgers.edu/~pxk/rutgers/notes/sockets/

[3] Y. Yang and L. Li, "Turn Smartphones into Computer Remote Controllers " International Journal of Computer Theory and Engineering, Vol. 4, No. 4, Page No 561 - 564

[4] https://docs.oracle.com/javase/7/docs/api/java/awt/Robot.html

[5] https://developer.android.com/guide/index.html

[6]https://blog.idrsolutions.com/2015/04/javafx-mp3-music-player-embedding-sound-in-your-application/

[7]http://stackoverflow.com/questions/12715321/java-networking-explain-inputstream-and-outputstream-in-socket

[8] https://docs.oracle.com/javase/tutorial/uiswing/