MINI-PROJECT II
(2020-2021)

MID TERM REPORT

Screen Honcho: App

Group Members:

1) Ayush Singh Chauhan **(181500180)**
2) Rachit Rai **(181500524)**
3) Rohit Gupta **(181500590)**
4) Kshitij Gupta **(181500336)**
5) Ayush Gupta **(181500173)**

Under the Supervision of**: Mr. Vinay Agarwal**

Technical Trainer

Department of Computer Science Engineering & Applications

CONTENTS

1. Introduction	**1**
2. Objective	**2**
3. Motivation	**3**
4. Technology Used	**4**
5. Requirements	**5**
6. Modules	**6**
7. Progress till date and remaining	**8**
8 . Implementation details	**9**
9. Future Scope	**26**
10. References	**27**

INTRODUCTION

This project is aimed at developing an app to use your Android phone to control your Laptop. This application will be very initiative and would have features like Control Left Click, Right Click, Mouse Scroll, Type text Transfer files from phone to laptop, all the basic media and computer control. For this project we'll need Android Studio for our app implementation, Kotlin Language for backend, XML also used MVVM for application frontend and architectural layout. User can also perform features like Download files from laptop to phone,

Use their laptop as speaker to play music files of phone, and See images of phone on laptop Control presentation on laptop via phone ;Suspend, Restart or Shut down your laptop using phone. Fetch your laptop screen to Android (only single click supported) Browse Android files on Desktop (View and Download)

It enables you to transfer files across different platforms, mirror and remote control mobile devices, receive and reply to messages on the computer.

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 .

Screen Honcho 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 Kolton/JAVA technology in this project.

OBJECTIVE

There exist several situations where we want to wirelessly and comfortably operate or access 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.

There are many devices in the market to reduce the work required but can only do on or two things and people end up owning multiple devices , some overlapping the functions of others. This can be solved by making the use of one device that almost every human on the planet owns, i.e. mobile phones. Our App makes the use of mobile device to let the user access their computer and perform required functions.

Our app aims 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. Control presentation on laptop via phone
G. 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).

MOTIVATION

Think of those lazy weekends when you just don't want to move a muscle; or those chilling winter nights when you are comfortably enjoying a movie on your couch, and you wished you didn't have to leave your comfort zone to change the volume or skip tracks.

So, you may think, "Can I use my Android phone as a mouse?" Controlling devices with your mind through a brain-computer interface is not yet commercially viable. Nevertheless, we have Android apps that can work as a PC remote control. 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 .

Android apps that can control your other devices via local Wi-Fi, Bluetooth, or from anywhere via the internet come in handy for remote administration.

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 .

TECHNOLOGIES USED

1. **XML** stands for Extensible Mark-up Language, which gives us a clue to what it does. A mark-up language is slightly different from a programming language. Whereas a programming language (C#, C++, Java, Kotlin, Python, BASIC) will allow you to define behaviours, interactions, and conditions; a mark-up language is used more to describe data, and in this case, layouts. Programming languages create dynamic interactions, whereas mark-up languages generally handle things like static user interfaces.

2. **KOTLIN** is a cross-platform, statically typed, general-purpose programming language with type inference. Kotlin is designed to interoperate fully with Java, and the JVM version of Katlin's standard library depends on the Java Class Library, but type inference allows its syntax to be more concise Although, you can use both **Kotlin** and **Java** to develop native android apps, **Google announced in 2019** to make Kotlin the preferred way of developing android applications. If you were to start learning android development today, Kotlin is our language of choice.

3. **HTML**: Hypertext Mark-up Language is the standard mark-up language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets and scripting languages such as JavaScript.

For user interfaces. To make a simple initiative UI for pc also which will be require for functioning of some of the app module.

5. 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.

5. **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.

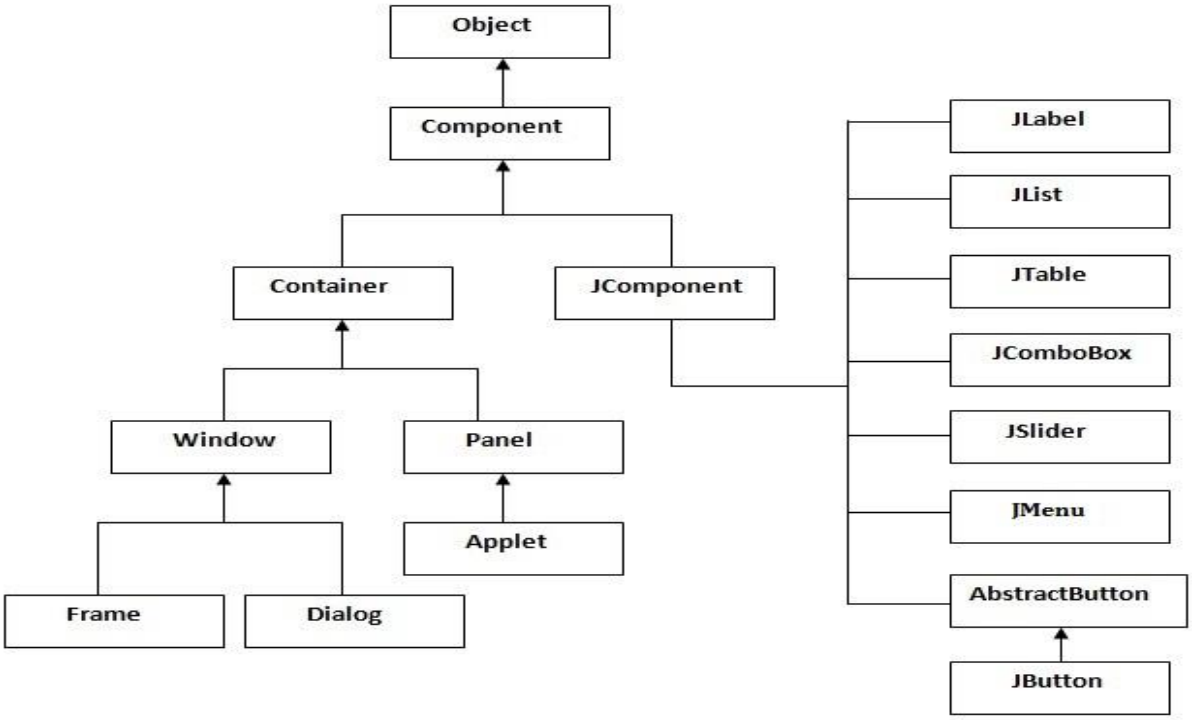
Swing is slowly being replaced by JavaFX

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.



REQUIREMENTS

➢ DEVELOPMENT HARDWARE USED:

Development Hardware that we have used is listed below:

- I5 7th/ i7 7th /9th GEN HQ KABI-LAKE/ KABI-LAKE / Coffee Lake Family
- RAM – 8/12GB
- 1/2TB HARDRIVE + 128GB/256GB SSD

➢ SOFTWARE USED:

- Microsoft Windows 7/8/10 or Linux
- Balsamiq / Figma
- Android Studio/Canary

MODULES

There are 5 modules in the App.

➢ DEVICE CONNECTION

- Bluetooth
- Wi-Fi
- Bluetooth HID Profile

➢ DEVICE CONTROL

- Device Control as mouse
- Use Keyboard functionality
- Type Text

➢ FILE TRANSFER

- Send file from Pc to device
- Send file from device to pc
- Device here refers to android phone.
- Download files from laptop to phone

➢ MEDIA CONTROL

- Use your laptop as speaker to play music files of phone.
- See images of phone on laptop.
- Control presentation on laptop via phone.

➢ ADDITIONAL CONTROLS

- Suspend, Restart or Shut down your laptop using phone.
- Fetch your laptop screen to Android (only single click supported).
- Browse Android files on Desktop (View and Download).

PROGRESS TILL DATE AND REMAINING WORK

Progress Till Date:

The most time taking and important is the decision on what project to do . That being said most of the initial work completed till date includes frontend of most of the pages on the app . This includes pages that show device connection status , device control i.e. device control as mouse , keyboard , type text , page for file transfer , media control etc.

Remaining Work:

The work left is mostly of backend. Adding functionalities to the pages is also remaining . After adding functionalities and finishing the design of frontend on pages, combining all the pages will done.

IMPLEMENTATION DETAILS

- Brief Description of the project

Remote Control PC consists of following characteristics-

A. It has been developed in Java
B. It use TCP Socket (Via Wi-Fi) 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:

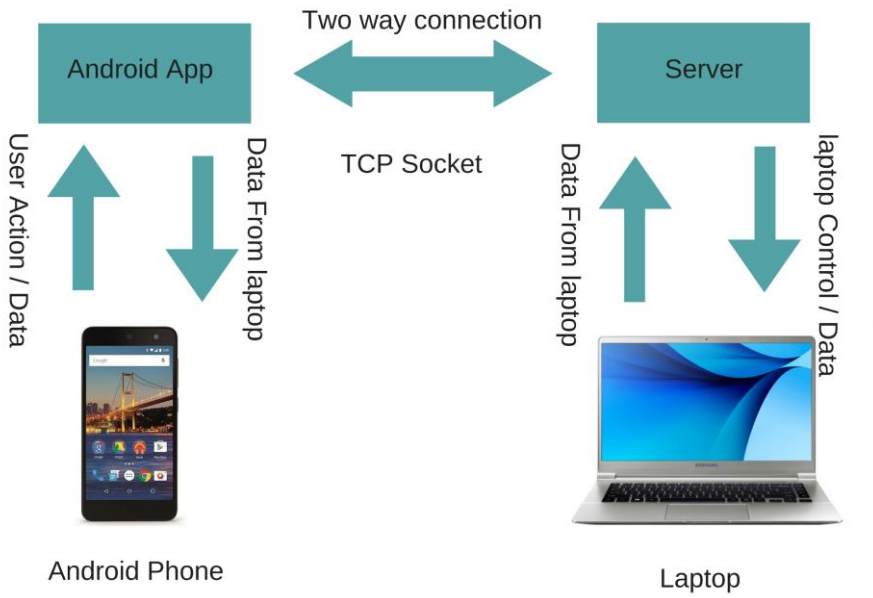


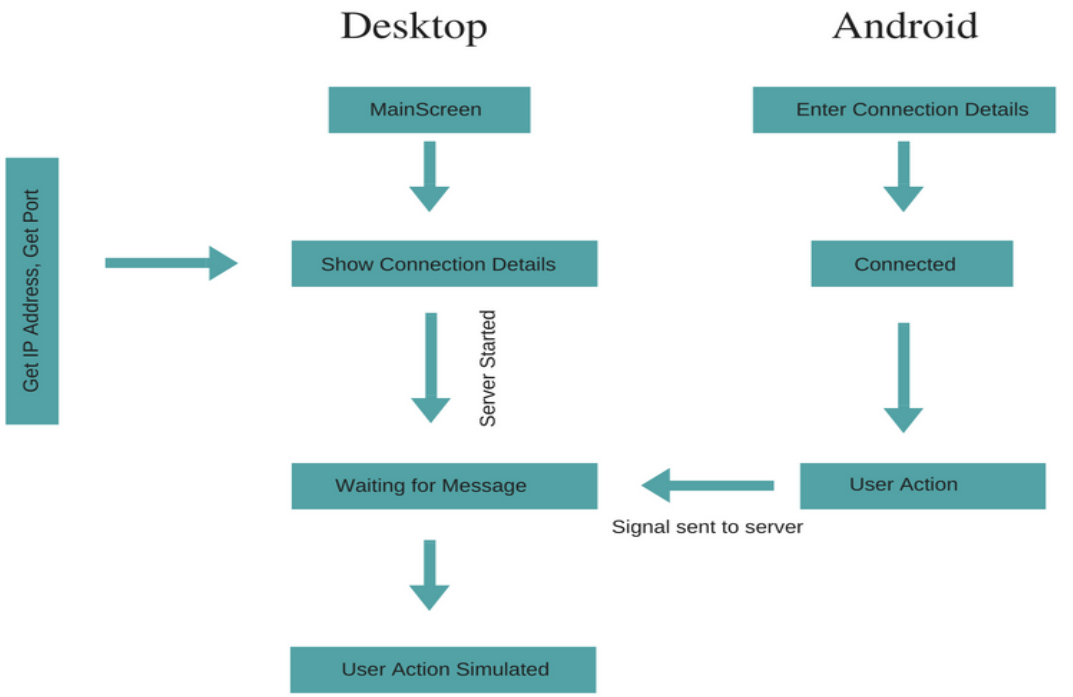
Fig 1: Working Model of Remote Control PC

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.

- Architecture Decision Record

An Architectural Decision (AD) is a software design choice that addresses a functional or non-functional requirement that is architecturally significant. An Architecturally Significant Requirement (ASR) is a requirement that has a measurable effect on a software system's architecture and quality. An Architectural Decision Record (ADR) captures a single AD, such as often done when writing personal notes or meeting minutes; the collection of ADRs created and maintained in a project constitute its decision log. All these are within the topic of Architectural Knowledge Management (AKM).

- Activity Diagram



Are graphical representations of workflows of stepwise activities and actions [1] with support for choice, iteration and concurrency? In the Unified Modelling Language, activity diagrams are intended to model both computational and organizational processes (i.e., workflows), as well as the data flows intersecting with the related activities. Although activity diagrams primarily show the overall flow of control, they can

Decision:

- It shows the judgment points and alternate flows.
- Additional changes allow the diagram to better support continuous behaviours and continuous data flows.

Consequences:

- It makes building the model easier concerning the process flow of the model.
- It will help in building the class diagram as it is communicating with different classes of the model.

Socket Streams

- It does not show the crucial information flow. 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 Output Stream from it, and write to the Output Stream (we output some data). To read data from the socket, we get its Input Stream, 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 Input Stream to read from the network, and its Output Stream to write

Desktop Interface & Functionality Implementation Snippet

FXML For layout

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.*?>
<?import javafx.scene.image.*?>
<?import java.lang.*?>
<?import java.net.*?>

<?import java.util.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>


<VBox fx:id="vBox" alignment="CENTER" maxHeight="-Infinity" maxWidth="-Infinity"
minHeight="-Infinity" minWidth="-Infinity" onMouseClicked="#showDetails"
prefHeight="80.0" prefWidth="84.0" xmlns="http://javafx.com/javafx/8"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="layout.FileOrFolderController">
   <children>

      <ImageView id="iconImageView" fx:id="iconImageView" fitHeight="41.0"
fitWidth="45.0" pickOnBounds="true" preserveRatio="true">
         <VBox.margin>
            <Insets />
         </VBox.margin>
         <image>
```

```
            <Image url="@../resources/folder.png" />
          </image></ImageView>
       <Label fx:id="headingLabel" alignment="CENTER" prefHeight="36.0"
prefWidth="84.0" text="folder" wrapText="true" />

    </children>
</VBox>
```

Android Interface & Functionality Implementation Snippet

How to connect

- Start hotspot on phone and connect your laptop via Wi-Fi or connect your laptop and phone with same Wi-Fi
- Open the desktop app on your laptop
- Open Android app and enter connection details provided by desktop app to connect

How Open PC App

- Open Screen-Honcho-JavaFXML file in your ide.(must have jdk 1.8 support and FXML support)
- Go to project Structure and select create jar option
- Open terminal or command Prompt Enter cd /Address of that project jar file
- Run command java –jar Screen-Honcho-JavaFXML.jar

App Dependencies

- User must have JRE 8 installed
- Only Android mobiles are supported
- Working Wi-Fi/hotspot is necessary

Limitations

- Media Player and Image Viewer work only after transferring complete file.
- There is no live streaming.
- There is some delay depending on file size.
- Shutdown, Restart and Suspend features do not work on Linux due to security issues.

Implementation Details

The complete project is divided into two parts

- ScreenHoncho-MiniProject: Android part of the project which has been developed in Android Studio.
- Screen-Honcho-JavaFxml: Desktop part of the project which has been developed in Jetbeans.

Packages and Classes structure

Desktop Part-

Java Packages	Java Classes
image	ImageViewer

music	MusicPlayer
screenhoncho.desktop	MainScreen Utility
screenhoncho.desktop.filesharing	FileAPI ReceiveFile SendFile SendFilesList
screenhoncho.desktop.ipaddress	GetFreePort GetMyIpAddress
screenhoncho.desktop.mousekeyboardcontrol	MouseKeyboardControl
screenhoncho.desktop.poweroff	PowerOff
screenhoncho.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

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);
```

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();
```

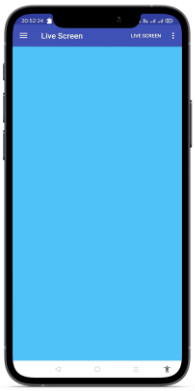
```
                    float nowx = point.x;
                    float nowy = point.y;
                    mouseControl.mouseMove((int) (nowx + x), (int) (nowy + y));
                    break;
                case "KEY_PRESS":
                    keyCode = (int) MainScreen.objectInputStream.readObject();
                    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();
    }
```

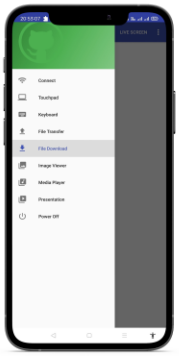
LIVE SCREEN



}

This Activity will let you interact with Live Laptop screen and from here user can And from here you can navigate and perform mouse like Operation from this Android Application. Offers interactive multi-touch and gesture options for easy control.

NAVIGATION MENU OR HAMBURGER MENU

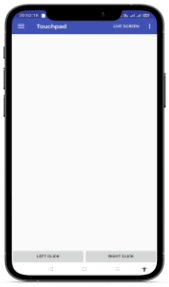


The hamburger menu in a apps is that typically opens up into a side menu or navigation drawer. From here you can navigate to other more important functional Activities like Touchpad, Presentation, Media Player, and Power off etc.

CONNECTION

From this activity we can perform the functionality of connecting the android Phone With The Computer or Laptop, Connecting your laptop with your phone via hotspot (or any local network) This is the activity to perform that functionality.

TOUCHPAD



As the name suggest Touchpad this activity has functionality exactly like laptop trackpad. From this activity you can perform functionalities that trackpad can perform.

POWER OFF

This activity has features like switching off laptop or putting it on sleep mode or functionalities like lock the pc or putting it on aeroplane mode.

KEYBOARD

This activity is also self-explanatory in itself it acts as keyboard of laptop or pc and you can perform all the functionalities of the actual physical keyword

PRESENTATION

Wireless presentation controller, allows user to operate his/her computer remotely for PowerPoint presentation through Bluetooth connection.

FUTURE SCOPE

The future scopes for this project are many. We are looking to add more functionalities. We hope to add a gamming console like functionality i.e. a person will be able to use their mobile devices as a gamming console.

We are also 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://developer.android.com/studio
2. https://stackoverflow.com/
3. https://www.w3schools.com/
4. http://www.cs.rutgers.edu/~pxk/rutgers/notes/sockets/
5. https://en.wikipedia.org/wiki/Android_(operating_system)
6. https://docs.oracle.com/javase/7/docs/api/java/awt/Robot.html
7. https://developer.android.com/guide/index.html
8. https://blog.idrsolutions.com/2015/04/javafx-mp3-music-player-embedding-sound-in-your-application/
9. http://stackoverflow.com/questions/12715321/java-networking-explain-inputstream-and-outputstream-in-socket
10. 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
11. https://docs.oracle.com/javase/tutorial/uiswing/