## Source Code

**MainActivity.java**

```
package me.varunon9.remotecontrolpc;
import android.Manifest;
import android.annotation.TargetApi;
import android.content.pm.PackageManager;
import android.os.Build;
import android.os.Bundle;
import android.os.Handler;
import com.google.android.material.navigation.NavigationView;
import androidx.fragment.app.FragmentTransaction;
import androidx.core.view.GravityCompat;
import androidx.drawerlayout.widget.DrawerLayout;
import androidx.appcompat.app.ActionBarDrawerToggle;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.app.AppCompatDelegate;
import androidx.appcompat.widget.Toolbar;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Toast;
import androidx.fragment.app.Fragment;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;
import me.varunon9.remotecontrolpc.connect.ConnectFragment;
import me.varunon9.remotecontrolpc.help.HelpFragment;
import me.varunon9.remotecontrolpc.keyboard.KeyboardFragment;
import
me.varunon9.remotecontrolpc.livescreen.LiveScreenFragment;
import me.varunon9.remotecontrolpc.poweroff.PowerOffFragment;
import
me.varunon9.remotecontrolpc.presentation.PresentationFragment;
import me.varunon9.remotecontrolpc.server.Server;
import me.varunon9.remotecontrolpc.touchpad.TouchpadFragment;
public class MainActivity extends AppCompatActivity
        implements
NavigationView.OnNavigationItemSelectedListener {
    public static Socket clientSocket = null;
    public static ObjectInputStream objectInputStream = null;
    public static ObjectOutputStream objectOutputStream =
null;
    private static AppCompatActivity thisActivity;
    private boolean doubleBackToExitPressedOnce = false;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

AppCompatDelegate.setCompatVectorFromResourcesEnabled(true);
        setContentView(R.layout.activity_main);
        // selecting connect fragment by default
```

```
        replaceFragment(R.id.nav_connect);
        thisActivity = this;
        Toolbar toolbar = (Toolbar)
findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        /*FloatingActionButton fab = (FloatingActionButton)
findViewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own
action", Snackbar.LENGTH_LONG)
                        .setAction("Action", null).show();
            }
        });*/
        DrawerLayout drawer = (DrawerLayout)
findViewById(R.id.drawer_layout);
        ActionBarDrawerToggle toggle = new
ActionBarDrawerToggle(
                this, drawer, toolbar,
R.string.navigation_drawer_open,
R.string.navigation_drawer_close);
        drawer.setDrawerListener(toggle);
        toggle.syncState();
        NavigationView navigationView = (NavigationView)
findViewById(R.id.nav_view);

navigationView.setNavigationItemSelectedListener(this);
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            checkForPermission();
        }
    }
    @TargetApi(Build.VERSION_CODES.M)
    private void checkForPermission() {
        if
(thisActivity.checkSelfPermission(Manifest.permission.READ_EXT
ERNAL_STORAGE)
                != PackageManager.PERMISSION_GRANTED) {
            // Should we show an explanation?
            if
(thisActivity.shouldShowRequestPermissionRationale(Manifest.pe
rmission.READ_EXTERNAL_STORAGE)) {
                Toast.makeText(thisActivity, "Read Permission
is necessary to transfer", Toast.LENGTH_LONG).show();
            } else {
                thisActivity.requestPermissions(new
String[]{Manifest.permission.READ_EXTERNAL_STORAGE}, 1);
                //2 is integer constant for
WRITE_EXTERNAL_STORAGE permission, uses in
onRequestPermissionResult
            }
```

```
        }
    }
    @Override
    public void onBackPressed() {
        DrawerLayout drawer = (DrawerLayout)
findViewById(R.id.drawer_layout);
        if (drawer.isDrawerOpen(GravityCompat.START)) {
            drawer.closeDrawer(GravityCompat.START);
        } else {
            if (doubleBackToExitPressedOnce) {
                super.onBackPressed();
                return;
            }
            doubleBackToExitPressedOnce = true;
            Toast.makeText(this, "Please click BACK again to
exit", Toast.LENGTH_SHORT).show();
            new Handler().postDelayed(new Runnable() {
                @Override
                public void run() {
                    doubleBackToExitPressedOnce = false;
                }
            }, 2000);
        }
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar
if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar
will
        // automatically handle clicks on the Home/Up button,
so long
        // as you specify a parent activity in
AndroidManifest.xml.
        int id = item.getItemId();
        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }
        onNavigationItemSelected(item);
        return super.onOptionsItemSelected(item);
    }
    @SuppressWarnings("StatementWithEmptyBody")
    @Override
    public boolean onNavigationItemSelected(MenuItem item) {
        // Handle navigation view item clicks here.
```

```
        Fragment fragment = null;
        int id = item.getItemId();
        replaceFragment(id);
        DrawerLayout drawer = (DrawerLayout)
findViewById(R.id.drawer_layout);
        drawer.closeDrawer(GravityCompat.START);
        return true;
    }
    private void replaceFragment(int id) {
        Fragment fragment = null;
        if (id == R.id.nav_connect) {
            fragment = new ConnectFragment();
        } else if (id == R.id.nav_touchpad) {
            fragment = new TouchpadFragment();
        } else if (id == R.id.nav_keyboard) {
            fragment = new KeyboardFragment();
//        } else if (id == R.id.nav_file_transfer) {
//            fragment = new FileTransferFragment();
//        } else if (id == R.id.nav_file_download) {
//            fragment = new FileDownloadFragment();
//        } else if (id == R.id.nav_image_viewer) {
//            fragment = new ImageViewerFragment();
//        } else if (id == R.id.nav_media_player) {
//            fragment = new MediaPlayerFragment();
//        } else if (id == R.id.nav_presentation) {
//            fragment = new PresentationFragment();
        } else if (id == R.id.nav_power_off) {
            fragment = new PowerOffFragment();
//        } else if (id == R.id.action_help) {
//            fragment = new HelpFragment();
//        } else if (id == R.id.action_live_screen) {
//            fragment = new LiveScreenFragment();
        }
        if (fragment != null) {
            FragmentTransaction fragmentTransaction =
getSupportFragmentManager().beginTransaction();
            fragmentTransaction.replace(R.id.content_frame,
fragment);
            fragmentTransaction.commit();
        }
    }
    protected void onDestroy() {
        super.onDestroy();
        try {
            if (MainActivity.clientSocket != null) {
                MainActivity.clientSocket.close();
            }
            if (MainActivity.objectOutputStream != null) {
                MainActivity.objectOutputStream.close();
            }
            if (MainActivity.objectInputStream != null) {
```

```
                MainActivity.objectInputStream.close();
            }
        } catch(Exception e) {
            e.printStackTrace();
        }
        Server.closeServer();
    }
    //this method is called from fragments to send message to
server (Desktop)
    public static void sendMessageToServer(String message) {
        if (MainActivity.clientSocket != null) {
            new
SendMessageToServer().execute(String.valueOf(message),
"STRING");
            /*try {

MainActivity.objectOutputStream.writeObject(message);
                MainActivity.objectOutputStream.flush();
            } catch (Exception e) {
                e.printStackTrace();
                socketException();
            }*/
        }
    }
    public static void sendMessageToServer(int message) {
        if (MainActivity.clientSocket != null) {
            new
SendMessageToServer().execute(String.valueOf(message), "INT");
            /*try {

MainActivity.objectOutputStream.writeObject(message);
                MainActivity.objectOutputStream.flush();
            } catch (Exception e) {
                e.printStackTrace();
                socketException();
            }*/
        }
    }
    public static void socketException() {
        //Toast.makeText(thisActivity, "Connection Closed",
Toast.LENGTH_LONG).show();
        if (MainActivity.clientSocket != null) {
            try {
                MainActivity.clientSocket.close();
                MainActivity.objectOutputStream.close();
                MainActivity.clientSocket = null;
            } catch(Exception e2) {
                e2.printStackTrace();
            }
        }
    }
```

```
    public static void sendMessageToServer(float message) {
        if (MainActivity.clientSocket != null) {
            new
SendMessageToServer().execute(String.valueOf(message),
"FLOAT");
            /*try {

MainActivity.objectOutputStream.writeObject(message);
                MainActivity.objectOutputStream.flush();
            } catch (Exception e) {
                e.printStackTrace();
                socketException();
            }*/
        }
    }
    public static void sendMessageToServer(long message) {
        if (MainActivity.clientSocket != null) {
            new
SendMessageToServer().execute(String.valueOf(message),
"LONG");
            /*try {

MainActivity.objectOutputStream.writeObject(message);
                MainActivity.objectOutputStream.flush();
            } catch (Exception e) {
                e.printStackTrace();
                socketException();
            }*/
        }
    }
    @Override
    public void onRequestPermissionsResult(int requestCode,
                                           String
permissions[], int[] grantResults) {
        switch (requestCode) {
            case 2: {
                // If request is cancelled, the result arrays
are empty.
                if (grantResults.length > 0
                        && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                    Toast.makeText(this, "Click again to
download", Toast.LENGTH_SHORT).show();
                } else {
                    Toast.makeText(this, "Failed to download",
Toast.LENGTH_SHORT).show();
                }
                return;
            }
            case 1: {
```

```
                // If request is cancelled, the result arrays
are empty.
                if (grantResults.length > 0
                        && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                    //Toast.makeText(this, "Click again to
download", Toast.LENGTH_SHORT).show();
                } else {
                    Toast.makeText(this, "File Transfer will
not work", Toast.LENGTH_SHORT).show();
                }
                return;
            }
        }
    }
}
```

**Utility.java**

```
package me.varunon9.remotecontrolpc;
import android.content.ContentUris;
import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.net.Uri;
import android.os.ParcelFileDescriptor;
import java.io.FileDescriptor;
import java.text.SimpleDateFormat;
import java.util.Calendar;
public class Utility {
     public String getDate(String date, String dateFormat) {
          long milliSeconds = Long.parseLong(date);
          // Create a DateFormatter object for displaying date
in specified format.
         SimpleDateFormat formatter = new
SimpleDateFormat(dateFormat);
         // Create a calendar object that will convert the
date and time value in milliseconds to date.
          Calendar calendar = Calendar.getInstance();
          calendar.setTimeInMillis(milliSeconds);
          return formatter.format(calendar.getTime());
     }
     public String getDate(long milliSeconds, String
dateFormat) {
          // Create a DateFormatter object for displaying date
in specified format.
         SimpleDateFormat formatter = new
SimpleDateFormat(dateFormat);
         // Create a calendar object that will convert the
date and time value in milliseconds to date.
          Calendar calendar = Calendar.getInstance();
          calendar.setTimeInMillis(milliSeconds);
```

```
          return formatter.format(calendar.getTime());
     }
     /*http://stackoverflow.com/questions/22859350/android-
listview-out-of-memory-exception*/
     public Bitmap decodeImageFile(String path) {
          try {
            // Decode image size
            BitmapFactory.Options o = new
BitmapFactory.Options();
            o.inJustDecodeBounds = true;
            BitmapFactory.decodeFile(path, o);
            // The new size we want to scale to
            final int REQUIRED_SIZE = 60;
            // Find the correct scale value. It should be the
power of 2.
            int scale = 1;
            while (o.outWidth / scale / 2 >= REQUIRED_SIZE &&
o.outHeight / scale / 2 >= REQUIRED_SIZE)
                scale *= 2;
            // Decode with inSampleSize
            BitmapFactory.Options o2 = new
BitmapFactory.Options();
            o2.inSampleSize = scale;
            return BitmapFactory.decodeFile(path, o2);
        } catch (Throwable e) {
            e.printStackTrace();
        }
        return null;
     }
     public String getDuration(int duration) {
          String time = "";
          duration /= 1000;//in seconds
          int minutes = duration / 60;
          duration %= 60;
          if (minutes > 0) {
               time += minutes + " mins ";
          }
          time += duration + " secs";
          return time;
     }
     public Bitmap getAlbumart(int albumId, Context context) {
         Bitmap bm = null;
         try {
             final Uri sArtworkUri =
Uri.parse("content://media/external/audio/albumart");
             Uri uri = ContentUris.withAppendedId(sArtworkUri,
albumId);
             ParcelFileDescriptor pfd =
context.getContentResolver().openFileDescriptor(uri, "r");
            if (pfd != null) {
                 FileDescriptor fd = pfd.getFileDescriptor();
```

```
                bm = BitmapFactory.decodeFileDescriptor(fd);
            }
        } catch (Exception e) {
        }
        return bm;
    }
    public String getSize(int size) {
        size /= 1024;
        return size + "KB";
    }
    public String getSize(long size) {
        size /= 1024;
        return size + "KB";
    }
}
```

**Server.java**

```
package me.varunon9.remotecontrolpc.server;
/**
 * Created by varun on 30/7/17.
 */
import android.app.Activity;
import android.widget.Toast;
import java.io.InputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.OutputStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.ArrayList;
//import file.AvatarFile;
//import me.varunon9.remotecontrolpc.filetransfer.FilesList;
//import
me.varunon9.remotecontrolpc.filetransfer.SendFilesList;
/**
 * This class will create a ServerSocket and connect to PC
 * It is mandatory to download or browse Android files on
Desktop
 */
public class Server {
    private static ServerSocket serverSocket;
    private static Socket clientSocket;
    private static InputStream inputStream;
    private static OutputStream outputStream;
    private static ObjectInputStream objectInputStream;
    private static ObjectOutputStream objectOutputStream;
    private static Activity activity;
```

```
    public Server(Activity activity) {
        this.activity = activity;
    }
    public void startServer(int port) {
        try {
            serverSocket = new ServerSocket(port);
        } catch(Exception e) {
            activity.runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    Toast.makeText(activity, "Unable to start
server", Toast.LENGTH_LONG).show();
                }
            });
            e.printStackTrace();
            return;
        }
        try {
            clientSocket = serverSocket.accept();
            inputStream = clientSocket.getInputStream();
            outputStream = clientSocket.getOutputStream();
            objectInputStream = new
ObjectInputStream(inputStream);
            objectOutputStream = new
ObjectOutputStream(outputStream);
            String filePath;
            while (true) {
                String message = (String)
objectInputStream.readObject();
                if (message == null) {
                    // connection closed
                    closeServer();
                    break;
                }
                switch (message) {
                    case "FILE_DOWNLOAD":
                        filePath = (String)
objectInputStream.readObject();
                        new
TransferFileToPC().transferFile(filePath, objectOutputStream);
                        break;
                    default: ;
                }
            }
        } catch(Exception e) {
            e.printStackTrace();
        }
    }
    public static void closeServer() {
        try {
            if (serverSocket != null) {
```

```
                serverSocket.close();
            }
            if (clientSocket != null) {
                clientSocket.close();
            }
            if (inputStream != null) {
                inputStream.close();
            }
            if (outputStream != null) {
                outputStream.close();
            }
            if (objectOutputStream != null) {
                objectOutputStream.close();
            }
            if (objectInputStream != null) {
                objectInputStream.close();
            }
        } catch(Exception e) {
            System.out.println(e);
        }
    }
    public static void sendMessageToServer(long message) {
        if (clientSocket != null) {
            try {
                objectOutputStream.writeObject(message);
                objectOutputStream.flush();
            } catch (Exception e) {
                e.printStackTrace();
                socketException();
            }
        }
    }
    private static void socketException() {
        Toast.makeText(activity, "Connection Closed",
Toast.LENGTH_LONG).show();
        if (clientSocket != null) {
            try {
                clientSocket.close();
                objectOutputStream.close();
                clientSocket = null;
            } catch(Exception e2) {
                e2.printStackTrace();
            }
        }
    }
}
```

**SendMessageToserver.java**

```
package me.varunon9.remotecontrolpc;
import android.os.AsyncTask;
```

```
/**
 * Created by varun on 28/9/17.
 */
public class SendMessageToServer extends AsyncTask<String,
Void, Void> {
    @Override
    protected Void doInBackground(String... params) {
        String message = params[0];
        String code = params[1];
        // message may be int, float, long or string
        int intMessage;
        float floatMessage;
        long longMessage;
        System.out.println(message + ", " + code);
        if (code.equals("STRING")) {
            try {

MainActivity.objectOutputStream.writeObject(message);
                MainActivity.objectOutputStream.flush();
            } catch(Exception e) {
                e.printStackTrace();
                MainActivity.socketException();
            }
        } else if (code.equals("INT")) {
            try {
                intMessage = Integer.parseInt(message);

MainActivity.objectOutputStream.writeObject(intMessage);
                MainActivity.objectOutputStream.flush();
            } catch(Exception e) {
                e.printStackTrace();
                MainActivity.socketException();
            }
        } else if (code.equals("FLOAT")) {
            try {
                floatMessage = Float.parseFloat(message);

MainActivity.objectOutputStream.writeObject(floatMessage);
                MainActivity.objectOutputStream.flush();
            } catch(Exception e) {
                e.printStackTrace();
                MainActivity.socketException();
            }
        } else if (code.equals("LONG")) {
            try {
                longMessage = Long.parseLong(message);

MainActivity.objectOutputStream.writeObject(longMessage);
                MainActivity.objectOutputStream.flush();
            } catch(Exception e) {
                e.printStackTrace();
```

```
                MainActivity.socketException();
            }
        }
        return null;
    }
}
```

**KeyboardFragment.java**

```
package me.varunon9.remotecontrolpc.keyboard;
import android.os.Bundle;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;
import android.text.Editable;
import android.text.TextWatcher;
import android.view.LayoutInflater;
import android.view.MotionEvent;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;
import me.varunon9.remotecontrolpc.MainActivity;
import me.varunon9.remotecontrolpc.R;
/**
 * A simple {@link Fragment} subclass.
 */
public class KeyboardFragment extends Fragment implements
View.OnTouchListener, View.OnClickListener, TextWatcher {
    private EditText typeHereEditText;
    private Button ctrlButton, altButton, shiftButton,
enterButton, tabButton, escButton, printScrButton,
backspaceButton;
    private Button deleteButton, clearTextButton;
    private Button nButton, tButton, wButton, rButton,
fButton, zButton;
    private Button cButton, xButton, vButton, aButton,
oButton, sButton;
    private Button ctrlAltTButton, ctrlShiftZButton,
altF4Button;
    private String previousText = "";
    public KeyboardFragment() {
        // Required empty public constructor
    }
    @Override
    public View onCreateView(LayoutInflater inflater,
ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View rootView =
inflater.inflate(R.layout.fragment_keyboard, container,
false);
        initialization(rootView);
```

```
        return rootView;
    }
    @Override
    public void onViewCreated(View view, @Nullable Bundle
savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);

getActivity().setTitle(getResources().getString(R.string.keybo
ard));
    }
    private void initialization(View rootView) {
        typeHereEditText = (EditText)
rootView.findViewById(me.varunon9.remotecontrolpc.R.id.typeHer
eEditText);
        ctrlButton = (Button)
rootView.findViewById(me.varunon9.remotecontrolpc.R.id.ctrlBut
ton);
        altButton = (Button)
rootView.findViewById(me.varunon9.remotecontrolpc.R.id.altButt
on);
        shiftButton = (Button)
rootView.findViewById(me.varunon9.remotecontrolpc.R.id.shiftBu
tton);
        enterButton = (Button)
rootView.findViewById(me.varunon9.remotecontrolpc.R.id.enterBu
tton);
        tabButton = (Button)
rootView.findViewById(me.varunon9.remotecontrolpc.R.id.tabButt
on);
        escButton = (Button)
rootView.findViewById(me.varunon9.remotecontrolpc.R.id.escButt
on);
        printScrButton = (Button)
rootView.findViewById(me.varunon9.remotecontrolpc.R.id.printSc
rButton);
        backspaceButton = (Button)
rootView.findViewById(me.varunon9.remotecontrolpc.R.id.backspa
ceButton);
        deleteButton = (Button)
rootView.findViewById(me.varunon9.remotecontrolpc.R.id.deleteB
utton);
        clearTextButton = (Button)
rootView.findViewById(me.varunon9.remotecontrolpc.R.id.clearTe
xtButton);
        nButton = (Button)
rootView.findViewById(me.varunon9.remotecontrolpc.R.id.nButton
);
        tButton = (Button)
rootView.findViewById(me.varunon9.remotecontrolpc.R.id.tButton
);
```

```
        wButton = (Button)
rootView.findViewById(me.varunon9.remotecontrolpc.R.id.wButton
);
        rButton = (Button)
rootView.findViewById(me.varunon9.remotecontrolpc.R.id.rButton
);
        fButton = (Button)
rootView.findViewById(me.varunon9.remotecontrolpc.R.id.fButton
);
        zButton = (Button)
rootView.findViewById(me.varunon9.remotecontrolpc.R.id.zButton
);
        cButton = (Button)
rootView.findViewById(me.varunon9.remotecontrolpc.R.id.cButton
);
        xButton = (Button)
rootView.findViewById(me.varunon9.remotecontrolpc.R.id.xButton
);
        vButton = (Button)
rootView.findViewById(me.varunon9.remotecontrolpc.R.id.vButton
);
        aButton = (Button)
rootView.findViewById(me.varunon9.remotecontrolpc.R.id.aButton
);
        oButton = (Button)
rootView.findViewById(me.varunon9.remotecontrolpc.R.id.oButton
);
        sButton = (Button)
rootView.findViewById(me.varunon9.remotecontrolpc.R.id.sButton
);
        ctrlAltTButton = (Button)
rootView.findViewById(me.varunon9.remotecontrolpc.R.id.ctrlAlt
TButton);
        ctrlShiftZButton = (Button)
rootView.findViewById(me.varunon9.remotecontrolpc.R.id.ctrlShi
ftZButton);
        altF4Button = (Button)
rootView.findViewById(me.varunon9.remotecontrolpc.R.id.altF4Bu
tton);
        ctrlButton.setOnTouchListener(this);
        altButton.setOnTouchListener(this);
        shiftButton.setOnTouchListener(this);
        backspaceButton.setOnClickListener(this);
        enterButton.setOnClickListener(this);
        tabButton.setOnClickListener(this);
        escButton.setOnClickListener(this);
        printScrButton.setOnClickListener(this);
        deleteButton.setOnClickListener(this);
        clearTextButton.setOnClickListener(this);
        nButton.setOnClickListener(this);
        tButton.setOnClickListener(this);
```

```
        wButton.setOnClickListener(this);
        rButton.setOnClickListener(this);
        fButton.setOnClickListener(this);
        zButton.setOnClickListener(this);
        cButton.setOnClickListener(this);
        xButton.setOnClickListener(this);
        vButton.setOnClickListener(this);
        aButton.setOnClickListener(this);
        oButton.setOnClickListener(this);
        sButton.setOnClickListener(this);
        ctrlAltTButton.setOnClickListener(this);
        ctrlShiftZButton.setOnClickListener(this);
        altF4Button.setOnClickListener(this);
        typeHereEditText.addTextChangedListener(this);
    }
    @Override
    public boolean onTouch(View v, MotionEvent event) {
        String action = "KEY_PRESS";
        if (event.getAction() == MotionEvent.ACTION_DOWN) {
            action = "KEY_PRESS";
        } else if (event.getAction() == MotionEvent.ACTION_UP)
{
            action = "KEY_RELEASE";
        }
        int keyCode = 17;//dummy initialization
        switch (v.getId()) {
            case me.varunon9.remotecontrolpc.R.id.ctrlButton:
                keyCode = 17;
                break;
            case me.varunon9.remotecontrolpc.R.id.altButton:
                keyCode = 18;
                break;
            case me.varunon9.remotecontrolpc.R.id.shiftButton:
                keyCode = 16;
                break;
        }
        sendKeyCodeToServer(action, keyCode);
        return false;
    }
    @Override
    public void onClick(View v) {
        int id = v.getId();
        if (id ==
me.varunon9.remotecontrolpc.R.id.clearTextButton) {
            typeHereEditText.setText("");
        } else if (id ==
me.varunon9.remotecontrolpc.R.id.ctrlAltTButton || id ==
me.varunon9.remotecontrolpc.R.id.ctrlShiftZButton || id ==
me.varunon9.remotecontrolpc.R.id.altF4Button) {
            String message = "CTRL_SHIFT_Z";
            switch (id) {
```

```
                case
me.varunon9.remotecontrolpc.R.id.ctrlAltTButton:
                    message = "CTRL_ALT_T";
                    break;
                case
me.varunon9.remotecontrolpc.R.id.ctrlShiftZButton:
                    message = "CTRL_SHIFT_Z";
                    break;
                case
me.varunon9.remotecontrolpc.R.id.altF4Button:
                    message = "ALT_F4";
                    break;
            }
            MainActivity.sendMessageToServer(message);
        } else {
            int keyCode = 17;//dummy initialization
            String action = "TYPE_KEY";
            switch (id) {
                case
me.varunon9.remotecontrolpc.R.id.enterButton:
                    keyCode = 10;
                    break;
                case
me.varunon9.remotecontrolpc.R.id.tabButton:
                    keyCode = 9;
                    break;
                case
me.varunon9.remotecontrolpc.R.id.escButton:
                    keyCode = 27;
                    break;
                case
me.varunon9.remotecontrolpc.R.id.printScrButton:
                    keyCode = 154;
                    break;
                case
me.varunon9.remotecontrolpc.R.id.deleteButton:
                    keyCode = 127;
                    break;
                case me.varunon9.remotecontrolpc.R.id.nButton:
                    keyCode = 78;
                    break;
                case me.varunon9.remotecontrolpc.R.id.tButton:
                    keyCode = 84;
                    break;
                case me.varunon9.remotecontrolpc.R.id.wButton:
                    keyCode = 87;
                    break;
                case me.varunon9.remotecontrolpc.R.id.rButton:
                    keyCode = 82;
                    break;
                case me.varunon9.remotecontrolpc.R.id.fButton:
```

```
                    keyCode = 70;
                    break;
                case me.varunon9.remotecontrolpc.R.id.zButton:
                    keyCode = 90;
                    break;
                case me.varunon9.remotecontrolpc.R.id.cButton:
                    keyCode = 67;
                    break;
                case me.varunon9.remotecontrolpc.R.id.xButton:
                    keyCode = 88;
                    break;
                case me.varunon9.remotecontrolpc.R.id.vButton:
                    keyCode = 86;
                    break;
                case me.varunon9.remotecontrolpc.R.id.aButton:
                    keyCode = 65;
                    break;
                case me.varunon9.remotecontrolpc.R.id.oButton:
                    keyCode = 79;
                    break;
                case me.varunon9.remotecontrolpc.R.id.sButton:
                    keyCode = 83;
                    break;
                case
me.varunon9.remotecontrolpc.R.id.backspaceButton:
                    keyCode = 8;
                    break;
            }
            sendKeyCodeToServer(action, keyCode);
        }
    }
    private void sendKeyCodeToServer(String action, int
keyCode) {
        MainActivity.sendMessageToServer(action);
        MainActivity.sendMessageToServer(keyCode);
    }
    @Override
    public void beforeTextChanged(CharSequence s, int start,
int count,
                                  int after) {
    }
    @Override
    public void onTextChanged(CharSequence s, int start, int
before, int count) {
        char ch = newCharacter(s, previousText);
        if (ch == 0) {
            return;
        }
        MainActivity.sendMessageToServer("TYPE_CHARACTER");

MainActivity.sendMessageToServer(Character.toString(ch));
```

```
        previousText = s.toString();
    }
    @Override
    public void afterTextChanged(Editable s) {
    }
    private char newCharacter(CharSequence currentText,
CharSequence previousText) {
        char ch = 0;
        int currentTextLength = currentText.length();
        int previousTextLength = previousText.length();
        int difference = currentTextLength -
previousTextLength;
        if (currentTextLength > previousTextLength) {
            if (1 == difference) {
                ch = currentText.charAt(previousTextLength);
            }
        } else if (currentTextLength < previousTextLength) {
            if (-1 == difference) {
                ch = '\b';
            } else {
                ch = ' ';
            }
        }
        return ch;
    }
}
/**
 * ctrl: 17
 * alt: 18
 * shift: 16
 * enter: 10
 * tab: 9
 * esc: 27
 * prntScr: 154
 * backspace: 524
 * delete: 127
 * backspace: 8
 */
/**
 * N: 78
 * T: 84
 * W: 87
 * R: 82
 * F: 70
 * Z: 90
 * C: 67
 * X: 88
 * V: 86
 * A: 65
 * O: 79
 * S: 83
```

```
 */
```

**TouchpadFragment.java**

```
package me.varunon9.remotecontrolpc.touchpad;
import android.os.Bundle;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;
import android.view.LayoutInflater;
import android.view.MotionEvent;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.TextView;
import me.varunon9.remotecontrolpc.MainActivity;
import me.varunon9.remotecontrolpc.R;
/**
 * A simple {@link Fragment} subclass.
 */
public class TouchpadFragment extends Fragment {
    private Button leftClickButton, rightClickButton;
    private TextView touchPadTextView;
    private int initX, initY, disX, disY;
    boolean mouseMoved = false, moultiTouch = false;
    public TouchpadFragment() {
        // Required empty public constructor
    }
    @Override
    public View onCreateView(LayoutInflater inflater,
ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View rootView =
inflater.inflate(R.layout.fragment_touchpad, container,
false);
        leftClickButton = (Button)
rootView.findViewById(R.id.leftClickButton);
        rightClickButton = (Button)
rootView.findViewById(R.id.rightClickButton);
        touchPadTextView = (TextView)
rootView.findViewById(R.id.touchPadTextView);
        leftClickButton.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                simulateLeftClick();
            }
        });
        rightClickButton.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
```

```
                simulateRightClick();
            }
        });
        touchPadTextView.setOnTouchListener(new
View.OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent event)
{
                if (MainActivity.clientSocket != null) {
                    switch(event.getAction() &
MotionEvent.ACTION_MASK){
                        case MotionEvent.ACTION_DOWN:
                            //save X and Y positions when user
touches the TextView
                            initX = (int) event.getX();
                            initY = (int) event.getY();
                            mouseMoved = false;
                            break;
                        case MotionEvent.ACTION_MOVE:
                            if(moultiTouch == false) {
                                disX = (int) event.getX()-
initX; //Mouse movement in x direction
                                disY = (int) event.getY()-
initY; //Mouse movement in y direction
                                /*set init to new position so
that continuous mouse movement
                                is captured*/
                                initX = (int) event.getX();
                                initY = (int) event.getY();
                                if (disX != 0 || disY != 0) {

MainActivity.sendMessageToServer("MOUSE_MOVE");
                                    //send mouse movement to
server

MainActivity.sendMessageToServer(disX);

MainActivity.sendMessageToServer(disY);
                                    mouseMoved=true;
                                }
                            }
                            else {
                                disY = (int) event.getY()-
initY; //Mouse movement in y direction
                                disY = (int) disY / 2;//to
scroll by less amount
                                initY = (int) event.getY();
                                if(disY != 0) {

MainActivity.sendMessageToServer("MOUSE_WHEEL");
```

```
MainActivity.sendMessageToServer(disY);
                                    mouseMoved=true;
                                }
                            }
                            break;
                        case MotionEvent.ACTION_CANCEL:
                        case MotionEvent.ACTION_UP:
                            //consider a tap only if user did
not move mouse after ACTION_DOWN
                            if(!mouseMoved){

MainActivity.sendMessageToServer("LEFT_CLICK");
                            }
                            break;
                        case MotionEvent.ACTION_POINTER_DOWN:
                            initY = (int) event.getY();
                            mouseMoved = false;
                            moultiTouch = true;
                            break;
                        case MotionEvent.ACTION_POINTER_UP:
                            if(!mouseMoved) {

MainActivity.sendMessageToServer("LEFT_CLICK");
                            }
                            moultiTouch = false;
                            break;
                    }
                }
                return true;
            }
        });
        return rootView;
    }
    @Override
    public void onViewCreated(View view, @Nullable Bundle
savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);

getActivity().setTitle(getResources().getString(R.string.touch
pad));
    }
    private void simulateLeftClick() {
        String message = "LEFT_CLICK";
        MainActivity.sendMessageToServer(message);
    }
    private void simulateRightClick() {
        String message = "RIGHT_CLICK";
        MainActivity.sendMessageToServer(message);
    }
}
```

## Server code Files:

### ClientToAndroid.java

```
package remotecontrolpc;
import file.AvatarFile;
import java.io.InputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.OutputStream;
import java.net.InetAddress;
import java.net.InetSocketAddress;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.SocketAddress;
import java.util.ArrayList;
import javafx.concurrent.Service;
import javafx.concurrent.Task;
import static
remotecontrolpc.MainScreenController.mainScreenController;
/**
 *
 * @author varun
 */
public class ClientToAndroid {

    private static ServerSocket serverSocket;
    private static Socket clientSocket;
    private static InputStream inputStream;
    private static OutputStream outputStream;
    public static ObjectInputStream objectInputStream;
    private static ObjectOutputStream objectOutputStream;

    public void connect(InetAddress inetAddress, int port) {
        new Service<Void>() {
            @Override
            protected Task<Void> createTask() {
                return new Task<Void>() {
                    @Override
                    protected Void call() throws Exception {
                        Thread.sleep(3000);
                        connectToAndroid(inetAddress, port);
                        return null;
                    }

                };
            }

        }.start();
```

```
    }

    private void connectToAndroid(InetAddress inetAddress, int
port) {
            try {
                SocketAddress socketAddress
                        = new InetSocketAddress(inetAddress,
port);
                clientSocket = new Socket();

                // 3s timeout
                clientSocket.connect(socketAddress, 3000);
                inputStream = clientSocket.getInputStream();
                outputStream = clientSocket.getOutputStream();
                objectOutputStream = new
ObjectOutputStream(outputStream);
                objectInputStream = new
ObjectInputStream(inputStream);

                // Request Android to fetch files list for
root directory
                fetchDirectory("/");
            } catch(Exception e) {
                e.printStackTrace();
            }
    }
    public static void closeConnectionToAndroid() {
        try {
            if (serverSocket != null) {
                serverSocket.close();
            }
            if (clientSocket != null) {
                clientSocket.close();
            }
            if (inputStream != null) {
                inputStream.close();
            }
            if (outputStream != null) {
                outputStream.close();
            }
            if (objectOutputStream != null) {
                objectOutputStream.close();
            }
            if (objectInputStream != null) {
                objectInputStream.close();
            }
        } catch (Exception e) {
            System.out.println(e);
        }
    }
```

```
    public static void sendMessageToAndroid(String message) {
        if (clientSocket != null) {
            try {
                objectOutputStream.writeObject(message);
                objectOutputStream.flush();
            } catch(Exception e) {
                e.printStackTrace();
            }
        }
    }

    public static void fetchDirectory(String path) {
        try {
            sendMessageToAndroid("FILE_DOWNLOAD_LIST_FILES");
            sendMessageToAndroid(path);
            ArrayList<AvatarFile> filesInFolder
                    = (ArrayList<AvatarFile>)
objectInputStream.readObject();
            if (filesInFolder == null ||
filesInFolder.isEmpty()) {
            } else {
                mainScreenController.showFiles(filesInFolder);
                mainScreenController.displayPath(path);
            }
        } catch(Exception e) {
            e.printStackTrace();
        }
    }
}
```

**MainScreenController.java**

```
package remotecontrolpc;
import file.AvatarFile;
import ipaddress.GetFreePort;
import ipaddress.GetMyIpAddress;
import java.io.File;
import java.io.InputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.OutputStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.URL;
import java.util.ArrayList;
import java.util.ResourceBundle;
import java.util.Stack;
import javafx.application.Platform;
import javafx.concurrent.Service;
import javafx.concurrent.Task;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
```

```
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Parent;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.TilePane;
import layout.FileOrFolderController;
/**
 *
 * @author varun
 */
public class MainScreenController implements Initializable {

    public static MainScreenController mainScreenController;
    public static ServerSocket serverSocket = null;
    public static Socket clientSocket = null;
    public static InputStream inputStream = null;
    public static OutputStream outputStream = null;
    public static ObjectOutputStream objectOutputStream =
null;
    public static ObjectInputStream objectInputStream = null;
    private Stack<String> pathStack;

    @FXML
    private TilePane tilePane;
    @FXML
    private BorderPane borderPane;
    @FXML
    private Label ipAddressLabel;
    @FXML
    private Label portNumberLabel;
    @FXML
    private Label connectionStatusLabel;
    @FXML
    private Button resetButton;
    @FXML
    private Label messageLabel;
    @FXML
    private Button backButton;
    @FXML
    private Button rootButton;
    @FXML
    Label pathLabel;

    public MainScreenController() {
        pathStack = new Stack<>();
    }
```

```
    public MainScreenController getMainScreenController() {
        return MainScreenController.mainScreenController;
    }

    public void setMainScreenController(MainScreenController
mainScreenController) {
        MainScreenController.mainScreenController =
mainScreenController;
    }

    @FXML
    private void resetConnection(ActionEvent event) {
        try {
            if (serverSocket != null) {
                serverSocket.close();
            }
            if (clientSocket != null) {
                clientSocket.close();
            }
            if (inputStream != null) {
                inputStream.close();
            }
            if (outputStream != null) {
                outputStream.close();
            }
            if (objectOutputStream != null) {
                objectOutputStream.close();
            }
            if (objectInputStream != null) {
                objectInputStream.close();
            }
        } catch(Exception e) {
            e.printStackTrace();
        }
        ClientToAndroid.closeConnectionToAndroid();
        setConnectionDetails();
    }

    @FXML
    private void fetchRootDirectory() {
        ClientToAndroid.fetchDirectory("/");
    }

    @FXML
    private void previousLocation() {
        String currentPath = pathStack.peek();
        if (currentPath == null || currentPath.equals("/")) {
            Platform.runLater(() -> {
                backButton.setDisable(true);
            });
            return;
```

```
        }
        pathStack.pop();
        currentPath = pathStack.peek();
        ClientToAndroid.fetchDirectory(currentPath);
    }

    @Override
    public void initialize(URL url, ResourceBundle rb) {
        setConnectionDetails();
    }

    private void setConnectionDetails() {
        String ipAddresses[] = new
GetMyIpAddress().ipAddress();
        String connectionStatus = "Not Connected";
        int port = new GetFreePort().getFreePort();
        String ipAddress = ipAddresses[0];
        if (ipAddresses[1] != null) {
            ipAddress = ipAddress + " | " + ipAddresses[1];
        }
        ipAddressLabel.setText(ipAddress);
        portNumberLabel.setText(Integer.toString(port));
        connectionStatusLabel.setText(connectionStatus);
        if (ipAddresses[0].equals("127.0.0.1")) {
            showMessage("Connect your PC to Android phone
hotspot or" +
                    " connect both devices to a local
network.");
        } else {
            try {
                serverSocket = new ServerSocket(port);
                startServer(port);
            } catch(Exception e) {
                showMessage("Error in initializing server");
                e.printStackTrace();
            }
        }
    }


    private void startServer(int port) throws Exception {
        new Service<Void>() {
            @Override
            protected Task<Void> createTask() {
                return new Task<Void>() {
                    @Override
                    protected Void call() throws Exception {
                        new Server().connect(resetButton,
connectionStatusLabel,
                                messageLabel, port);
                        return null;
```

```
                    }

                };
            }

        }.start();
    }

    public void showMessage(String message) {
        Platform.runLater(() -> {
            messageLabel.setText(message);
        });
    }

    public void displayPath(String path) {
        pathStack.push(path);
        pathLabel.setDisable(false);
        Platform.runLater(() -> {
            pathLabel.setText(path);
        });
    }

    public void showImage(String name, String path) {
        showMessage(name);
        Image image = new Image(new
File(path).toURI().toString());
        ImageView imageView = new ImageView(image);
        imageView.setPreserveRatio(true);
        Platform.runLater(() -> {

imageView.fitWidthProperty().bind(tilePane.widthProperty());

imageView.fitHeightProperty().bind(tilePane.heightProperty());
            tilePane.getChildren().clear();
            tilePane.getChildren().add(imageView);
        });
    }

    public void closeImageViewer() {
        Platform.runLater(() -> {
            tilePane.getChildren().clear();
        });
    }

    public void showFiles(ArrayList<AvatarFile> filesInFolder)
{
        Platform.runLater(() -> {
            tilePane.getChildren().clear();
            for (AvatarFile file : filesInFolder) {
                FXMLLoader fxmlLoader = new FXMLLoader(
```

```
getClass().getResource("/layout/FileOrFolder.fxml")
                );
                Parent root;
                try {
                    root = (Parent) fxmlLoader.load();
                } catch(Exception e) {
                    e.printStackTrace();
                    return;
                }
                FileOrFolderController fileOrFolderController
=
                        (FileOrFolderController)
fxmlLoader.getController();
                String fileType = file.getType();
                Image icon = null;
                switch(fileType) {
                    case "folder":
                        icon = new Image(

getClass().getResourceAsStream("/resources/folder.png")
                        );
                        break;
                    case "file":
                        icon = new Image(

getClass().getResourceAsStream("/resources/file.png")
                        );
                        break;
                    case "image":
                        icon = new Image(

getClass().getResourceAsStream("/resources/image.png")
                        );
                        break;
                    case "mp3":
                        icon = new Image(

getClass().getResourceAsStream("/resources/music.png")
                        );
                        break;
                    case "pdf":
                        icon = new Image(

getClass().getResourceAsStream("/resources/pdf.png")
                        );
                        break;
                    default: ;
                }
                fileOrFolderController.setIcon(icon);
```

```
fileOrFolderController.setHeading(file.getHeading());

fileOrFolderController.setSubHeading(file.getSubheading());

fileOrFolderController.setPath(file.getPath());

fileOrFolderController.setFileType(file.getType());
                tilePane.getChildren().add(root);
            }
        });
    }

}
```

**RemoteControlPC.java**

```
/*
 * To change this license header, choose License Headers in
Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package remotecontrolpc;
import javafx.application.Application;
import javafx.application.Platform;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;
/**
 *
 * @author varun
 */
public class RemoteControlPC extends Application {

    @Override
    public void start(Stage stage) throws Exception {
        //Parent root =
FXMLLoader.load(getClass().getResource("MainScreen.fxml"));
        FXMLLoader fxmlLoader = new FXMLLoader(
                getClass().getResource("MainScreen.fxml")
        );
        Parent root = (Parent) fxmlLoader.load();
        MainScreenController mainScreenController =
                (MainScreenController)
fxmlLoader.getController();

mainScreenController.setMainScreenController(mainScreenControl
ler);
```

```
        Scene scene = new Scene(root);

        stage.setScene(scene);
        stage.setTitle("RemoteControlPC");
        stage.setOnCloseRequest(e -> {
            Platform.exit();
            System.exit(0);
        });
        stage.show();
    }
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        launch(args);
    }

}
```

**Server.java**

```
package remotecontrolpc;
import filesharing.Screenshot;
import image.ImageViewer;
import java.awt.Dimension;
import java.awt.MouseInfo;
import java.awt.Point;
import java.awt.Toolkit;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import filesharing.FileAPI;
import filesharing.ReceiveFile;
import filesharing.SendFile;
import filesharing.SendFilesList;
import java.net.InetAddress;
import javafx.application.Platform;
import mousekeyboardcontrol.MouseKeyboardControl;
import poweroff.PowerOff;
import music.MusicPlayer;
/**
 *
 * @author varun
 */
public class Server {

    private Label messageLabel;

    public void connect(Button resetButton, Label
connectionStatusLabel,
            Label messageLabel, int port) {
```

```
        this.messageLabel = messageLabel;
        MouseKeyboardControl mouseControl = new
MouseKeyboardControl();
        Dimension screenSize =
Toolkit.getDefaultToolkit().getScreenSize();
        int screenWidth = (int) screenSize.getWidth();
        int screenHeight = (int) screenSize.getHeight();
        try {
            MainScreenController.clientSocket =

MainScreenController.serverSocket.accept();
            Platform.runLater(() -> {
                resetButton.setDisable(true);
            });
            InetAddress remoteInetAddress =

MainScreenController.clientSocket.getInetAddress();
            String connectedMessage = "Connected to: " +
remoteInetAddress;
            Platform.runLater(() -> {

connectionStatusLabel.setText(connectedMessage);
            });
            showMessage(connectedMessage);

            // connecting another socket to app (Peer to Peer)
            new ClientToAndroid().connect(remoteInetAddress,
port);
            MainScreenController.inputStream =

MainScreenController.clientSocket.getInputStream();
            MainScreenController.outputStream =

MainScreenController.clientSocket.getOutputStream();
            MainScreenController.objectOutputStream =
                    new
ObjectOutputStream(MainScreenController.outputStream);
            MainScreenController.objectInputStream =
                    new
ObjectInputStream(MainScreenController.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)
MainScreenController.objectInputStream.readObject();
                    int keyCode;
                    if (message != null) {
                        switch (message) {
                            case "LEFT_CLICK":
                                mouseControl.leftClick();
                                break;
                            case "RIGHT_CLICK":
                                mouseControl.rightClick();
                                break;
                            case "DOUBLE_CLICK":
                                mouseControl.doubleClick();
                                break;
                            case "MOUSE_WHEEL":
                                int scrollAmount =
                                        (int)
MainScreenController.objectInputStream.readObject();

mouseControl.mouseWheel(scrollAmount);
                                break;
                            case "MOUSE_MOVE":
                                int x = (int)
MainScreenController.objectInputStream.readObject();
                                int y = (int)
MainScreenController.objectInputStream.readObject();
                                Point point =
MouseInfo.getPointerInfo().getLocation();
                                // Get current mouse position
                                float nowx = point.x;
                                float nowy = point.y;
                                mouseControl.mouseMove((int)
(nowx + x), (int) (nowy + y));
                                break;
                            case "MOUSE_MOVE_LIVE":
                                // need to adjust coordinates
                                float xCord = (float)
MainScreenController.objectInputStream.readObject();
                                float yCord = (float)
MainScreenController.objectInputStream.readObject();
                                xCord = xCord * screenWidth;
                                yCord = yCord * screenHeight;
                                mouseControl.mouseMove((int)
xCord, (int) yCord);
                                break;
                            case "KEY_PRESS":
                                keyCode = (int)
MainScreenController.objectInputStream.readObject();

mouseControl.keyPress(keyCode);
                                break;
```

```
                                case "KEY_RELEASE":
                                    keyCode = (int)
MainScreenController.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":
                                    //handle
StringIndexOutOfBoundsException here when pressing soft enter
key
                                    char ch = ((String)
MainScreenController.objectInputStream.readObject()).charAt(0)
;

mouseControl.typeCharacter(ch);
                                    break;
                                case "TYPE_KEY":
                                    keyCode = (int)
MainScreenController.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)
MainScreenController.objectInputStream.readObject();
                                if (filePath.equals("/")) {
                                    filePath =
fileAPI.getHomeDirectoryPath();
                                }
                                new
SendFilesList().sendFilesList(
                                        fileAPI, filePath,
MainScreenController.objectOutputStream
                                );
                                break;
                            case "FILE_DOWNLOAD_REQUEST":
                                //filePath is complete path
including file name
                                filePath = (String)
MainScreenController.objectInputStream.readObject();
                                new
SendFile().sendFile(filePath,
MainScreenController.objectOutputStream);
                                break;
                            case "FILE_TRANSFER_REQUEST":
                                fileName = (String)
MainScreenController.objectInputStream.readObject();
                                long fileSize = (long)
MainScreenController.objectInputStream.readObject();
                                //not in thread, blocking
action
                                new ReceiveFile().receiveFile(
                                        fileName, fileSize,
MainScreenController.objectInputStream
                                );
                                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)
MainScreenController.objectInputStream.readObject();
                                filePath = new
FileAPI().getHomeDirectoryPath();
```

```
                                filePath = filePath +
"/RemoteControlPC/" + fileName;
                                try {

musicPlayer.playNewMedia(filePath);
                                    showMessage("Playing: " +
fileName);
                                } catch(Exception e) {
                                    showMessage("Unsupported
Media: " + fileName);
                                }
                                break;
                            case "SLIDE_MUSIC":
                                slideDuration = (int)
MainScreenController.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)
MainScreenController.objectInputStream.readObject();
                                musicPlayer.setVolume(volume);
                                break;
                            case "SHOW_IMAGE":
                                fileName = (String)
MainScreenController.objectInputStream.readObject();
                                filePath = new
FileAPI().getHomeDirectoryPath();
                                filePath = filePath +
"/RemoteControlPC/" + fileName;

imageViewer.showImage(fileName, filePath);
                                break;
                            case "CLOSE_IMAGE_VIEWER":

imageViewer.closeImageViewer();
                                break;
                            case "SCREENSHOT_REQUEST":
                                new
Screenshot().sendScreenshot(

MainScreenController.objectOutputStream
                                );
                                break;
```

```
                        }
                    } else {
                        //remote connection closed
                        Platform.runLater(() -> {
                            resetButton.setDisable(false);

connectionStatusLabel.setText("Disconnected");
                        });
                        connectionClosed();
                        break;
                    }
                } catch (Exception e) {
                    e.printStackTrace();
                    connectionClosed();

ClientToAndroid.closeConnectionToAndroid();
                    Platform.runLater(() -> {
                        resetButton.setDisable(false);

connectionStatusLabel.setText("Disconnected");
                    });
                    break;
                }
            };
        } catch(Exception e) {
            e.printStackTrace();
        }
    }

    private void connectionClosed() {
        try {
            MainScreenController.objectInputStream.close();
            MainScreenController.clientSocket.close();
            MainScreenController.serverSocket.close();
            MainScreenController.inputStream.close();
            MainScreenController.outputStream.close();
            MainScreenController.objectOutputStream.close();
        }
        catch(Exception e) {
            e.printStackTrace();
        }
    }

    private void showMessage(String message) {
        Platform.runLater(() -> {
            messageLabel.setText(message);
        });
    }
}
/*
* Codes:
```

```
* 1. LEFT_CLICK
* 2. RIGHT_CLICK
* 3. MOUSE_WHEEL
* 4. MOUSE_MOVE
* 5. KEY_PRESS
* 6. KEY_RELEASE
* 7. CTRL_ALT_T
* 8. CTRL_SHIFT_Z
* 9. ALT_F4
* 10. TYPE_CHARACTER
* 11. TYPE_KEY
*/
```

**MouseKeyboardControl.java**

```
/*
 * To change this license header, choose License Headers in
Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package mousekeyboardcontrol;
import java.awt.Robot;
import java.awt.event.InputEvent;
import java.awt.event.KeyEvent;
import javax.swing.JOptionPane;
import static java.awt.event.KeyEvent.*;
/**
 *
 * @author varun
 */
public class MouseKeyboardControl {
    Robot robot;
    public MouseKeyboardControl() {
        try {
            robot = new Robot();
        }
        catch(Exception e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(null, "Error
Occured!");
        }
    }
    public void leftClick() {
        robot.mousePress(InputEvent.BUTTON1_MASK);
        robot.mouseRelease(InputEvent.BUTTON1_MASK);
    }
    public void doubleClick() {
        leftClick();
        robot.delay(500);
        leftClick();
    }
```

```
public void rightClick() {
    robot.mousePress(InputEvent.BUTTON3_MASK);
    robot.mouseRelease(InputEvent.BUTTON3_MASK);
}
public void mouseMove(int x, int y) {
    robot.mouseMove(x, y);
}
public void mouseWheel(int wheelAmount) {
    robot.mouseWheel(wheelAmount);
}
public void keyPress(int keyCode) {
    robot.keyPress(keyCode);
}
public void keyRelease(int keyCode) {
    robot.keyRelease(keyCode);
}
public void ctrlAltT() {
    robot.keyPress(KeyEvent.VK_CONTROL);
    robot.keyPress(KeyEvent.VK_ALT);
    robot.keyPress(KeyEvent.VK_T);
    robot.delay(10);
    robot.keyRelease(KeyEvent.VK_T);
    robot.keyRelease(KeyEvent.VK_ALT);
    robot.keyRelease(KeyEvent.VK_CONTROL);
}
public void ctrlAltL() {
    robot.keyPress(KeyEvent.VK_CONTROL);
    robot.keyPress(KeyEvent.VK_ALT);
    robot.keyPress(KeyEvent.VK_L);
    robot.delay(10);
    robot.keyRelease(KeyEvent.VK_L);
    robot.keyRelease(KeyEvent.VK_ALT);
    robot.keyRelease(KeyEvent.VK_CONTROL);
}
public void ctrlShiftZ() {
    robot.keyPress(KeyEvent.VK_CONTROL);
    robot.keyPress(KeyEvent.VK_SHIFT);
    robot.keyPress(KeyEvent.VK_Z);
    robot.delay(10);
    robot.keyRelease(KeyEvent.VK_Z);
    robot.keyRelease(KeyEvent.VK_SHIFT);
    robot.keyRelease(KeyEvent.VK_CONTROL);
}
public void altF4() {
    robot.keyPress(KeyEvent.VK_ALT);
    robot.keyPress(KeyEvent.VK_F4);
    robot.delay(10);
    robot.keyRelease(KeyEvent.VK_F4);
    robot.keyRelease(KeyEvent.VK_ALT);
}
public void doType(int... keyCodes) {
```

```
        int length = keyCodes.length;
        for (int i = 0; i < length; i++) {
            robot.keyPress(keyCodes[i]);
        }
        robot.delay(10);
        for (int i = length - 1; i >= 0; i--) {
            robot.keyRelease(keyCodes[i]);
        }
    }
    public void typeCharacter(char character) {
        switch (character) {
        case 'a': doType(VK_A); break;
        case 'b': doType(VK_B); break;
        case 'c': doType(VK_C); break;
        case 'd': doType(VK_D); break;
        case 'e': doType(VK_E); break;
        case 'f': doType(VK_F); break;
        case 'g': doType(VK_G); break;
        case 'h': doType(VK_H); break;
        case 'i': doType(VK_I); break;
        case 'j': doType(VK_J); break;
        case 'k': doType(VK_K); break;
        case 'l': doType(VK_L); break;
        case 'm': doType(VK_M); break;
        case 'n': doType(VK_N); break;
        case 'o': doType(VK_O); break;
        case 'p': doType(VK_P); break;
        case 'q': doType(VK_Q); break;
        case 'r': doType(VK_R); break;
        case 's': doType(VK_S); break;
        case 't': doType(VK_T); break;
        case 'u': doType(VK_U); break;
        case 'v': doType(VK_V); break;
        case 'w': doType(VK_W); break;
        case 'x': doType(VK_X); break;
        case 'y': doType(VK_Y); break;
        case 'z': doType(VK_Z); break;
        case 'A': doType(VK_SHIFT, VK_A); break;
        case 'B': doType(VK_SHIFT, VK_B); break;
        case 'C': doType(VK_SHIFT, VK_C); break;
        case 'D': doType(VK_SHIFT, VK_D); break;
        case 'E': doType(VK_SHIFT, VK_E); break;
        case 'F': doType(VK_SHIFT, VK_F); break;
        case 'G': doType(VK_SHIFT, VK_G); break;
        case 'H': doType(VK_SHIFT, VK_H); break;
        case 'I': doType(VK_SHIFT, VK_I); break;
        case 'J': doType(VK_SHIFT, VK_J); break;
        case 'K': doType(VK_SHIFT, VK_K); break;
        case 'L': doType(VK_SHIFT, VK_L); break;
        case 'M': doType(VK_SHIFT, VK_M); break;
        case 'N': doType(VK_SHIFT, VK_N); break;
```

```
            case 'O': doType(VK_SHIFT, VK_O); break;
            case 'P': doType(VK_SHIFT, VK_P); break;
            case 'Q': doType(VK_SHIFT, VK_Q); break;
            case 'R': doType(VK_SHIFT, VK_R); break;
            case 'S': doType(VK_SHIFT, VK_S); break;
            case 'T': doType(VK_SHIFT, VK_T); break;
            case 'U': doType(VK_SHIFT, VK_U); break;
            case 'V': doType(VK_SHIFT, VK_V); break;
            case 'W': doType(VK_SHIFT, VK_W); break;
            case 'X': doType(VK_SHIFT, VK_X); break;
            case 'Y': doType(VK_SHIFT, VK_Y); break;
            case 'Z': doType(VK_SHIFT, VK_Z); break;
            case '`': doType(VK_BACK_QUOTE); break;
            case '0': doType(VK_0); break;
            case '1': doType(VK_1); break;
            case '2': doType(VK_2); break;
            case '3': doType(VK_3); break;
            case '4': doType(VK_4); break;
            case '5': doType(VK_5); break;
            case '6': doType(VK_6); break;
            case '7': doType(VK_7); break;
            case '8': doType(VK_8); break;
            case '9': doType(VK_9); break;
            case '-': doType(VK_MINUS); break;
            case '=': doType(VK_EQUALS); break;
            case '~': doType(VK_BACK_QUOTE); break;
            case '!': doType(VK_SHIFT, VK_EXCLAMATION_MARK);
    break;
            case '@': doType(VK_SHIFT, VK_AT); break;
            case '#': doType(VK_SHIFT, VK_NUMBER_SIGN); break;
            case '$': doType(VK_SHIFT, VK_DOLLAR); break;
            case '%': doType(VK_SHIFT, VK_5); break;
            case '^': doType(VK_SHIFT, VK_CIRCUMFLEX); break;
            case '&': doType(VK_SHIFT, VK_AMPERSAND); break;
            case '*': doType(VK_SHIFT, VK_ASTERISK); break;
            case '(': doType(VK_LEFT_PARENTHESIS); break;
            case ')': doType(VK_RIGHT_PARENTHESIS); break;
            case '_': doType(VK_SHIFT, VK_UNDERSCORE); break;
            case '+': doType(VK_SHIFT, VK_PLUS); break;
            case '\t': doType(VK_TAB); break;
            case '\n': doType(VK_ENTER); break;
            case '[': doType(VK_OPEN_BRACKET); break;
            case ']': doType(VK_CLOSE_BRACKET); break;
            case '\\': doType(VK_BACK_SLASH); break;
            case '{': doType(VK_SHIFT, VK_OPEN_BRACKET); break;
            case '}': doType(VK_SHIFT, VK_CLOSE_BRACKET); break;
            case '|': doType(VK_SHIFT, VK_BACK_SLASH); break;
            case ';': doType(VK_SEMICOLON); break;
            case ':': doType(VK_SHIFT, VK_COLON); break;
            case '\'': doType(VK_QUOTE); break;
            case '"': doType(VK_SHIFT, VK_QUOTEDBL); break;
```

```
            case ',': doType(VK_COMMA); break;
            case '<': doType(VK_SHIFT, VK_COMMA); break;
            case '.': doType(VK_PERIOD); break;
            case '>': doType(VK_SHIFT, VK_PERIOD); break;
            case '/': doType(VK_SLASH); break;
            case '?': doType(VK_SHIFT, VK_SLASH); break;
            case ' ': doType(VK_SPACE); break;
            case '\b': doType(VK_BACK_SPACE); break;
            default:
                //throw new IllegalArgumentException("Cannot type
character " + character);
            }
        }
        public void typeCharacter(int keyCode) {
            robot.keyPress(keyCode);
            robot.delay(10);
            robot.keyRelease(keyCode);
        }
        public void pressLeftArrowKey() {
            typeCharacter(VK_LEFT);
        }
        public void pressDownArrowKey() {
            typeCharacter(VK_DOWN);
        }
        public void pressRightArrowKey() {
            typeCharacter(VK_RIGHT);
        }
        public void pressUpArrowKey() {
            typeCharacter(VK_UP);
        }
        public void pressF5Key() {
            typeCharacter(VK_F5);
        }
}
```