DESIGN AND IMPLEMENTATION OF VIRTUALLY SHARED HOME SCREEN SYSTEM FOR ANDROID SMART PHONE

LEE JIA PING

UNIVERSITI TUNKU ABDUL RAHMAN

DESIGN AND IMPLEMENTATION OF VIRTUALLY SHARED HOME SCREEN SYSTEM FOR ANDROID SMART PHONE

LEE JIA PING

A project report submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering (Hons.) Electrical and Electronics Engineering

Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

September 2014

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature : _____

Name : _____

ID No. : _____

Date : _____

APPROVAL FOR SUBMISSION

I certify that this project report entitled **"DESIGN AND IMPLEMENTATION OF VIRTUALLY SHARED HOME SCREEN SYSTEM FOR ANDROID SMART PHONE"** was prepared by **LEE JIA PING** has met the required standard for submission in partial fulfillment of the requirements for the award of Bachelor of Engineering (Hons.) Electrical and Electronics Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature : _____

Supervisor : _____

Date : _____

DESIGN AND IMPLEMENTATION OF VIRTUALLY SHARED HOME SCREEN SYSTEM FOR ANDROID SMART PHONE

ABSTRACT

This project proposed a new idea to *virtually share home screens* among multiple smart phone systems in user level and the software architecture to implement it. A *home screen*, or application screen, is basically the initial screen we see when a smart phone or tablet is started, serves like a *home page* or *main menu* which typically displays a group of *apps icons* that can be clicked or tapped by the users to activate applications or use the internal functions. The proposed idea, called Virtually Shared Home Screen system (VSHS), allows the screens of the smart phones within a group of users to be combined into a single but bigger virtual home screen space which can be viewed and accessed by all smart phones within the group. By sharing the screens among multiple smart phones, remote *apps* and *files* on other smart phones within the group can be viewed and accessed by each user using his smart phone. As such, communication and file sharing can be integrated into one simple, direct, and fast interface for collaboration of work and file sharing among users within a group using existing *swipe user interface* supported in all smart phones, as compared to conventional approach requiring switching back and forth several times among several separate apps to communicate and transfer file. This project also defined the basic software architecture to implement VSHS, consisting of three independent modules, i.e. file transfer and consistency (FTC), screens and objects mapping (SOM), and seamless navigation interface (SNI). This project is the first exploration of this new idea by implementing the software architecture for the environment with two smart phones connected via Bluetooth.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF APPENDIXS

CHAPTER 1

INTRODUCTION

1.1 Background

Microsoft Remote Desktop is one of the many solutions which first implemented the idea to control a remote computer. Microsoft Remote Desktop is remote control software enabling users to access and control a remote work computer connected to internet from home. Once you log on the remote computer, display data and keyboard strokes transmit from the remote to the client computer, allowing you to view and work with the remote computer as if you were sitting directly in front of it. However, Remote Desktop allows only one connection. If someone attempts to use the remote computer, your remote connection will automatically terminate.

TeamViewer is another remote control software similar to Microsoft Remote Desktop but with significant improvement in the user interface, such as the support of live chat, contacts list featuring easy connection to remote computers previously connected or set up with just a single mouse-click, capability to wake up unattended computers that the user does not want to leave on all the time with Wake-on-LAN. In Microsoft Remote Desktop, file transfer service is not available at all. In TeamViewer, convenient file transferring to and from the remote computer is supported during TeamViewer session, but users have to literally choose the service in a separate option list.

A *home screen*, or application screen, is basically the initial screen we see when a smart phone or tablet is started, serves like a *home page* or *main menu* which typically displays a group of *apps icons* that can be clicked or tapped by the users to activate applications and internal functions. Allowing the user to directly treat the remote screen as his local screen is convenient, especially in a collaboration or group sharing environment.

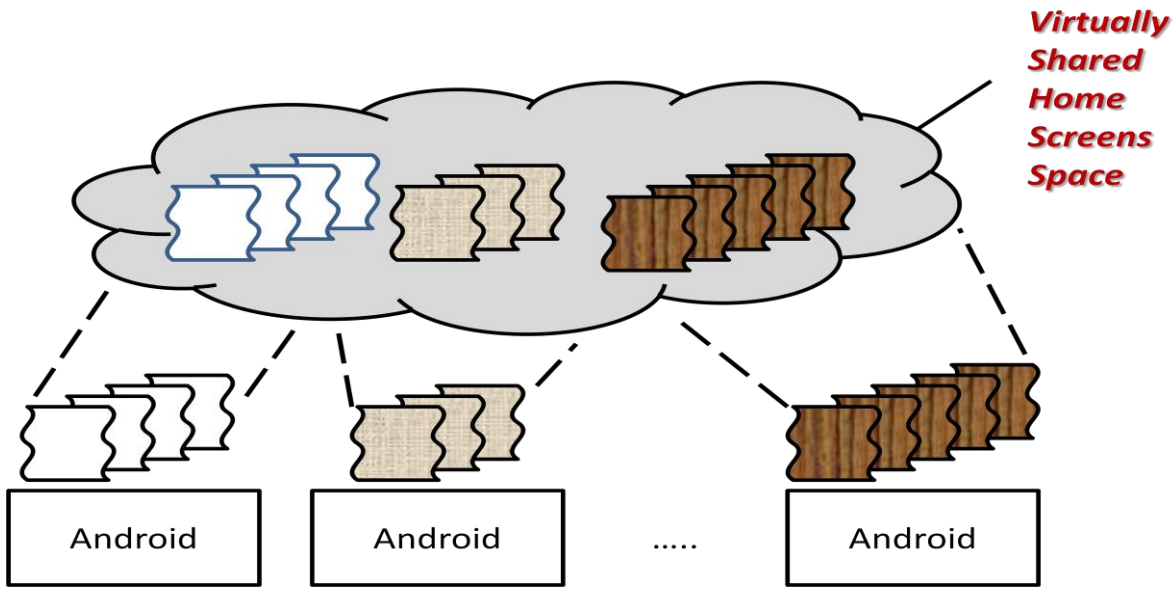

Figure 1.1 : Virtual Space is shared among all the Androids.

This project proposed a new system, called *Virtually Shared Home Screens* system (VSHS). After the connection with remote smart phones are set up, the home screens of remote smart phones will be combined into the local home screens to form a virtually shared home screens, as depicted in Figure 1.1. Whenever local user wants to receive a file from the remote device, he can simply swipe through the virtually shared home screens corresponding to the remote screen where the target file or apps is located, perform a long tab on the target file to select it, and drag the selected icon to the edge of the screen to swipe back to the destination in local home screen to initiate the file transfer seamlessly as if it is a local-to-local file transfer.

In addition, with VSHS, a local user not only can view the application screen of the remote device, but also has the access control to the remote device, given that permission is enabled during initialization phase. Screen sharing allows a local user

to play remote application and access (read/write) the remote file storage system. In this project, accessing the remote file storage system and seamless transferring files via Bluetooth will be the main focus to prove the concept.

The target operating system for developing VSHS is Android which currently dominates the smart phone market share. One of the reasons is that Android is an open-source platform with API readily available for download and modification. In contrast to proprietary software such as iOS and Windows where the software is under restrictive copyright, Android software cost is relatively cheaper and its source code does not contain the often proprietary device drives for the hardware components.

Next, Android uses Java language which is architecture neutral, portable, and dynamically adaptable. Applications are supported to be executing on different hardware architectures, operating systems and interoperating with multiple programming language interfaces. It generates byte codes, which is an architecture neutral intermediate format purposely designed to transport code efficiently to multiple hardware and software platforms. The same code can run on other platforms without recompile. Programs are therefore, the same on every platform and are compatible across hardware and software architectures. This makes programmer life easier as rewriting of programming code to run on other platform is tedious and time consuming.

1.2 Overview

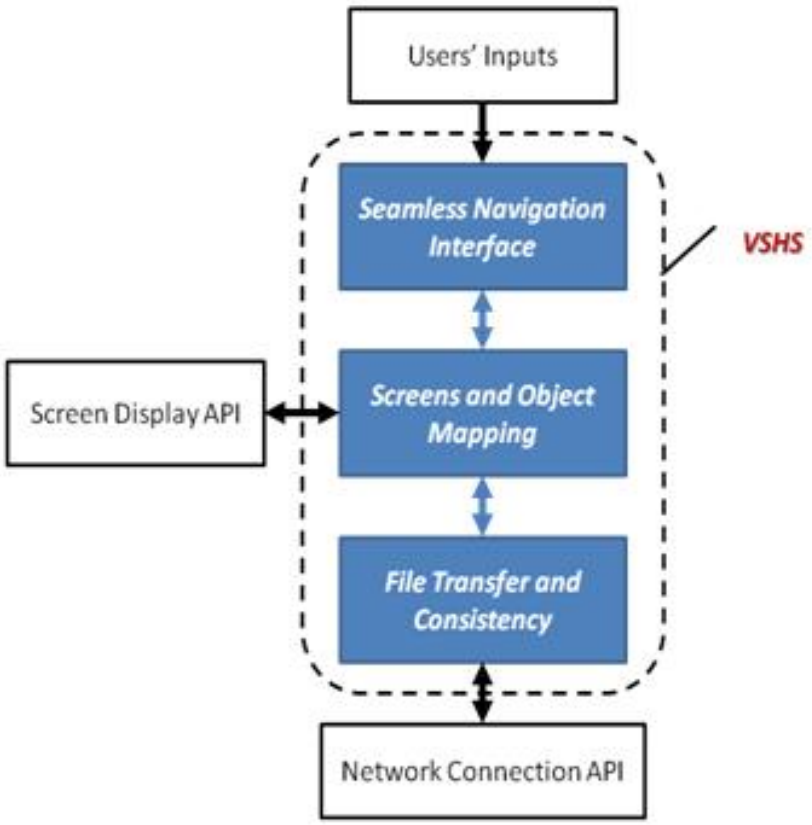

Figure 1.2 : Software Architecture for VSHS

The software architecture to implement VSHS, illustrated in Figure 1.2, can be layered into three independent modules, i.e. file transfer and consistency (FTC), screens and objects mapping (SOM), and seamless navigation interface (SNI).

The FTC module interacts with the underlying wireless communication application communication interface (API) provided by the operating system of the smart phone for both text communication and file transfer. In addition to file streaming, FTC also defines how the modification of remote file at local machine is written back to remote machine; as modification can happen at multiple machines simultaneously, a policy to synchronize and enforce the data consistency at the repository machine is required.

The SOM module in local machine identifies the remote sharable home screens and objects before acquiring them while the SOM module at the remote machine provides a menu to allow remote users to set up the read/write access permission. Each SOM module is also responsible to form the virtually shared home screens and maintain the mapping information of the remote objects in local memory so that when an object is selected for file transfer, it knows which machine to contact and what access permission to apply.

The SNI module intercepts the user inputs at the local machine to simulate the effect of screen swiping within the virtually shared home screens and to initiate the file transfer seamlessly as if it is a local-to-local file transfer upon request. When a shared object is tab for execution, the shared file will first be transferred from remote to local and then the SNI module will call the app associated with the file type to run in a separate thread to process the file, e.g. video player app for movie files, music player app for audio files, office app for text files, etc.

The VSHS appears like an ordinary app on the smart phone. To start the VSHS, local user will first ask the connection permission from the remote device in the Bluetooth list. In this study, after the remote user accepts the request, the local user can view the remote application screen and access to its remote file storage. Local user also can directly retrieve any file from the remote device as long as the file is readable or writeable.

Variety type of files can be transferred in this current implementation, including document file in word, excel, power point, text, or portable document format, music or video file and image file. The local user can treat the remote files as local files and no further request to the remote device is needed after the setup of the Bluetooth connection. The difference between the read-only and writeable files is any modification of the writeable file can be saved back to the remote device and replaced the older version of the file, whereas for non-writeable file, synchronization of the file is not allowed.

Furthermore, a more user-friendly feature is added into this application which is *copy*, *paste*, and *delete* functions. Local user can copy any readable and writeable

file from the remote device and paste the file to any directory in his local device or vice versa. Remote file deletion can be done only for writeable file in the remote device as to limit the control and accessibility of the local user to the remote device. In short, VSHS gives the ability to share files in both directions with a more attractive and user-friendly way while minimizing the steps for Bluetooth file transfer.

Due to security or privacy issue, remote users may not want to share all objects in their smart phone. One simple solution is to assign access permission to each object or restrict to sharing to happen in certain folders. Basically, before the remote user shares his file storage and data, he should configure the file access permission of the files to any of the three modes, i.e. not allowed to share, read-only sharing, and read-writeable sharing. All the remote files in read-only mode are modifiable in local machine but any modification from the local user will not be synchronized to the remote file.

Local user can retrieve all the files and treat them as local files as this application will create a copy of the file in the local device whenever the local user retrieve that particular file. Hence, he can modify the file and change the content in the file. The remote writeable file has an advantage of letting the local user to synchronize the modified file to the remote file. So, by giving the correct input, the modified file will be automatically sent back to the remote device and replaced the older version of the remote file. For example, lecturer can obtain the tutorials from his students and perform directly marking on the documents. After that, he can save the modified version of the tutorials with his marking back to his students' devices so they can view the latest version of the files.

VSHS also simplifies the conventional file transfer via Bluetooth in Android. For the conventional file transfer, the remote user has to make a file transfer request to the local user in prior to send the file. If the local user wants to obtain 10 files from the remote user, he has to ask the remote user to make 10 file transfer requests to the local device before sending those 10 files. However, this application only requires one request to the remote user. Before the local user can access to the remote file, he has to make a screen share request to the remote user. Once the

remote user accepts the request, the local user can view the remote application screen and access to all the read-only and writeable files in the remote device directly without making further file transfer request. Therefore, this application requires fewer steps from the user and it is more user-friendly as compared to conventional file transfer.

1.3 Problem Statement

This project is to implement the Virtually Shared Home Screen system (VSHS) to achieve a seamless file transfer via Bluetooth for app execution on Android-based smart phone systems. This project is to provide a solution to the three modules illustrated in Figure 1.2, i.e. file transfer and consistency (FTC), screens and objects mapping (SOM), and seamless navigation interface (SNI).

Few challenges arise when designing this system. First challenge is how the home screens and the remote file can be shared among the users. Second, how the remote file access permission can be configured. Third, how the file transferring can be stopped or terminated when the virtually shared home screen is formed among the users. Last challenge is once the application screen is shared, how the local machine can control the remote devices, respond incoming requests, or perform file transferring back and forth among remote devices.

There are many ways in smart phone for the users to send data wirelessly. The communication can be done through Bluetooth, WiFi, or mobile data. Bluetooth is used for a small area, about 100 meters square with limited number of connectors. WiFi can only be used when there is at least one device can access to Internet and that device has to create a hotspot so that other devices can connect to it. If there is a router that creates a hotspot, any device that can detect it is able to connect to the router. Its covered area is slightly larger than that of Bluetooth.

As WiFi allows user to access to the Internet, it is similar to mobile data, another way provides user accessibility to the Internet. The screen can be shared with

other users around the world. In this project, Bluetooth is chosen for the wireless communication in order to quickly test the idea. Therefore, the range of the users and the data transfer speed are limited.

This project develops an application to allow remote user to share his home screens and files with others. Determining the file access permission is important as some users might not want others to see some of the files or folders due to privacy. Hence, before the remote user accepts the screen sharing request, he has to configure the access permission of his files. Three modes of access permission are given which included not allowed mode, read-only mode and writeable mode. For the file in not allowed mode, the local user can only see the file name in the file list but he cannot access to the file. Both read-only and writeable files are able to be retrieved by the local user. However, the local user only can synchronize and replace the modified writeable files to the remote files.

The remote home screens can be shared after the Bluetooth connection is established. Ideally, local user can see all the remote application icons in his screen once shared. If both the local and remote devices have three application screens, then after the screen sharing is made, the local user can view all six application screens in the local device. All local application can be accessed. However, besides the remote file storage, all other remote applications are not accessible by the local user at this moment. To stop or terminate the file transfer, remote user can directly turn off the Bluetooth of his device, preventing further data transfer.

To make this project more interesting, copy, paste and delete buttons are added, allowing local user to share his files to the remote user, not only retrieving the remote files. Local user can copy his local file and paste it directly to the remote device, or vice versa. Besides, he can help remote user to delete the writeable file in the remote device if the file is no longer needed.

1.4 Aim and Objective

The main objective of the project is to create an Android app which is to share home screens and files among multiple smart phones within a group. The local device should be able to detect all the surround devices and connect to one of them as requested. Local user can directly retrieve the remote files without making any additional file transferring request.

Other objectives of this project are:

- To successfully share the application screen and files from the remote device via Bluetooth feature in the smart phone.
- To develop a configuration setting which allows remote user to choose the file access permission before sharing his application screen and files.
- To allow local user to view all the read-only and writeable files in the remote device, and synchronize all the modified writeable files to the remote device.
- To provide a more user-friendly access control of local user to remote device in performing activities such as copy and paste local file to the remote device, copy and paste remote file to the local device and delete remote writeable file.

1.5 Project Scope

This project is divided into first and second parts where the former should be completed at the end of the first semester and the latter is continued and finished in the second semester. The first chapter introduces the background of this project. The overview is written to describe the feature and function of the system and how it can simplify the conventional file transfer. Challenges are clarified in problem statement to give a basic idea on the developed system. A simple view on the aims and objectives provides the clear tasks and targets which should be accomplished by the end of these two semesters.

Next chapter brings out some existing ideas and researches that are similar with this project to widen the reader's view. Comparison is made between this project and the existing researches to briefly describe the strengths and weaknesses of each of them. This may improve the quality of the project as the useful features may be added to the system.

Third chapter gives a general explanation of methods used to complete the project. A basic idea is provided to show the way on designing the system. As the main focus is on the Android programming, there is no external peripheral used to incorporate with the Android smart phone. Basically, all the methods involve programming like controlling the Bluetooth adapter of the device, creating Bluetooth server socket to listen to the radio frequency signal, creating Bluetooth socket to start the communication and obtaining input stream or output stream to read or write the data that to be transmitted.

Fourth chapter describes the implementation, result and discussion for the project. This chapter will briefly describe the theory, work process and flow behind the methods used to develop the application, followed by the result in image format and discussion to show the completeness of the project.

Last chapter is the conclusion and recommendation for the project. This chapter will first compare the conventional file transfer with this developed application and conclude how this application can simplify the conventional file transfer and why this application is more user-friendly as compared to conventional file transfer. The weaknesses of this project are also written to show that this project has large improvement which may be carried on in the future.

CHAPTER 2

LITERATURE REVIEW

2.1 Overview

Various researches such as Group Digital Assistant, Escritoire, Spatial Interfaces, Low-latency Transmission Scheme for Interactive Screen Sharing and Multisurface Interaction in the WILD room have been carried out to promote the screen sharing system. Studies on the related works are important, not only to identify each strengths and weaknesses, but to improve the project developed in these two semesters.

2.2 Group Digital Assistant

In an earlier age, personal devices like smart phone and PDA are not suitable for cooperative by two or more persons because of the small screen. Users cannot view the contents at a glance simultaneously. So, GDA is designed to obtain a bigger screen by combining the screen of two PDAs. Unlike a laptop, PDA is more convenience and easy to carry with because of its small size and light weight.

Group Digital Assistant (GDA) is a concept of sharing and combining the screen of two Personal Digital Assistants (PDAs). As shown in Figure 2.1, the screen is shared at a sharing view mode, or combined at a combination mode. It also offers individual view mode just like a single PDA device. These view modes can be

changes flexibly. The radio communication is done by using wireless Local Area Network (LAN) and Bluetooth. This system is designed for users to have a better cooperative work such as brainstorming, categorizing of ideas, and cooperative writing.

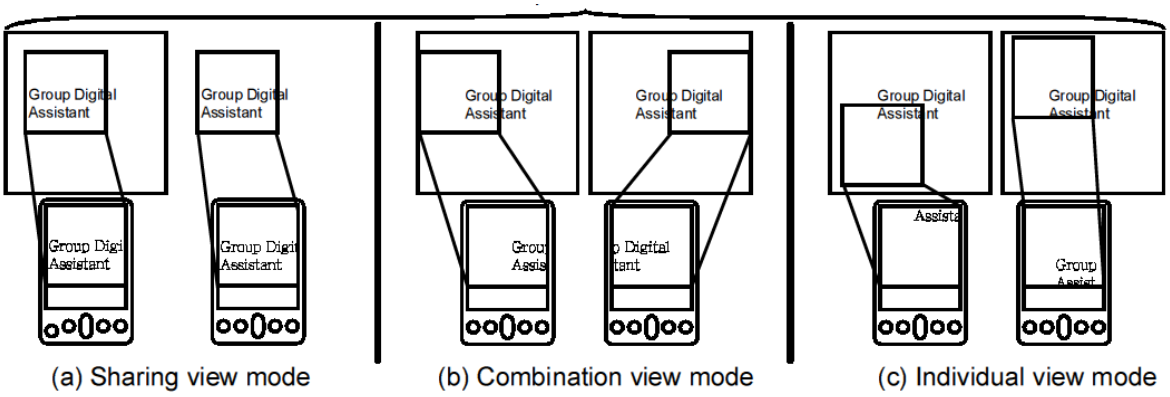

Figure 2.1: Three view modes are provided by GDA which are (a) sharing view mode, (b) combination view mode, and (c) individual view mode.

GDA can use two communication systems which are wireless LAN and Bluetooth. However, their protocols which are the set of rules for exchanging of data between two devices differ mutually. Hence, GDA middleware, a communication middleware is developed to handle both protocols by using the same API (Application Program Interface). It is computer software that provides functions and services to software applications beyond those functions available from the operating system, enabling two otherwise separate software components, processes, and applications to exchange information within one or multiple devices. This middleware, operating on Palm OS, consists of a program for Bluetooth library, a program for Net libraries, and common API.

In order to have screen sharing and screen combination, two mechanisms need to be fulfilled. First, all GDA units must have the same data of shared content. Second, screen share and screen combination are done by changing the display position of these same data. In screen share mode, the display positions of all GDA units in the large set of data are same and therefore, the display on each GDA unit is the same. Whereas in screen combination mode, the display positions are different

for each GDA unit, so the displays can be combined to provide a larger content. These mechanisms also make the change of a screen view mode flexible.

GDA uses the implementation of main screen and ‘offscreen’. Main screen is the actually display size of a GDA unit, which is 320 x 320 dots. While offscreen size is 640 x 640 dots and it is an imagination drawing area, which is a part of the memory of a GDA unit that used to manipulate the screen. No matter which view mode is used, the content of the offscreen is the same. Offscreen contains all the data display but a PDA unit only shows part of it because of its small screen size. Hence, GDA is designed to show all the data display of the offscreen to provide a glance view for the users and this can be done in screen combination mode. This implementation method of main screen and offscreen can deal simply with the screen share and screen combination features.

In GDA, three display modes are provided which are sharing view mode, combination view mode, and individual view mode. At sharing view mode, the contents showed on two GDA units are always the same. If one of the GDA units changes or updates its display, another GDA unit will automatically update its display corresponding to the GDA unit. The offscreen also will be updated as it is the source of the data display, containing all the data for manipulating and displaying.

At combination view mode, the screens of two GDA units are combined to form a large screen with more contents. Both GDA units are displaying different contents but the contents are related and are joined together when putting both GDA units together, one is next to another GDA. If one of the GDA units changes or updates its display, another GDA unit’s display is not influenced because there are displaying different content. However, their offscreens will be updated as to provide the latest information. The contents in offscreen must be the same in both the GDA units.

At individual view mode, the contents showed on two GDA units are independent but they are sharing the same offscreen. If one of the GDA units changes or updates its display, another GDA unit is not influenced as they are independent to each other. However, its offscreen will be updated so that when it

views to the related or same content, the updated and latest information can be displayed on that GDA.

In conclusion, this system is mainly used in cooperative work to improve the efficiency and also the interaction among the users. However, due to the high technology, large amount of smartphones with more advanced functions and powerful processor is manufactured and produced. Many types of smartphones in different brands, specifications, features, and a wide range of cost from few hundreds to two thousands dollar become affordable by most of the users are being sold in the market now. PDA or GDA is already eliminated and replaced by smartphone in the past few years. So this design knowledge should be implemented in smartphone because of its large market demand.

2.3 Escritoire

Escritoire is a personal projected display that employs two overlapping projectors to create a foveal display and the display is projected on a desk, as shown in Figure 2.2. One projector projects a large, horizontal with low-resolution periphery to fill the entire desk, while another projects a smaller and high-resolution fovea closed to users to accommodate their focus and to make a life-sized virtual document legible. The system can be used by only one person to do his personal task, or by remote participants to share the same visual display for their collaborative work.

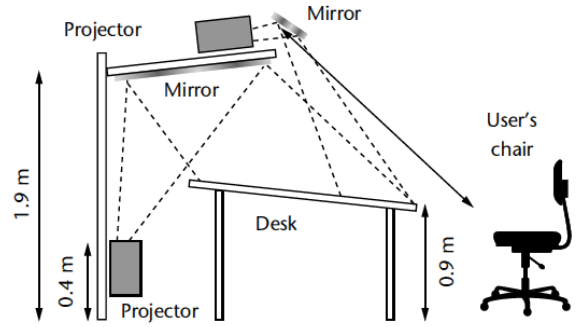

Figure 2.2: Two overlapping projectors are used to create a foveal display on a desk.

Typically, most of the projectors are mainly aligned orthogonally to a flat surface to provide an enlarged rectangular image. However, Escritoire uses two projectors where they are not aligned to the desk surface. It uses mirrors to reflect the projected images onto the desk as shown in Figure 2.2. Hence, the projected image will be distorted in such the oblique projection. To solve this problem, the image is wrapped before it is being displayed and this technique is known as projective transformation. Nowadays, many powerful 3D video cards can perform this projective transformation on images while two main application programming interfaces for developing these 3D video cards are DirectX and OpenGL.

This technique prepares the image in the texture memory of the video card then warped by texture mapping it onto a quadrilateral. Escritoire uses video cards that can warp a 1024 x 768 resolution image in 0.3 milliseconds. The amount of updating the texture is minimized as video card is designed to draw large numbers of polygons with static texture rather than to update the textures rapidly. Hence, the system performance is optimized to achieve 30 frames per second for the two-projector-display driven by a single dual-head video card. A dual-head video card is a video card that uses a single chipset to drive two display devices on one system at the same time.

Escritoire creates an environment, just like a user sitting in front of at a desk display, performing his work as he would be sitting in front of a normal desk with physical sheets of paper. In the older years or even now, most of the people are used to be worked in this paper, pen and desk environment. The difference is that this system is a fairly literal emulation of a normal desk where sheets of virtual paper are read, arranged and manipulated. There are no real papers on the desk as the all of them are digitized. Also, user can do sorting, searching, transmitting or passing of documents to another Escritoire, and even presenting an animated Graphical User Interface (GUI).

Escritoire is different from the conventional vertical wall display that often used in presentations or lectures. Its display is closer to horizontal and has different affordances compared to the frequently used large vertical screens. It acts like an

architect's drafting board so that the user can sit comfortably and peer over documents as if these virtual documents were papers on a desk.

To make it more user-friendly, two electronic pens are designed to provide bimanual input which requires two-handed operation over the entire desk, whereas only dominant hand is used when dealing with real papers. A thinner and more accurate pen uses digitizer based on electromagnetism to perform fine tasks such as writing while another chunky whiteboard pen with a lower accuracy uses ultrasonic wave to perform coarse tasks such as positioning sheets on desk display. The digitizer pen is used by dominant hand while the ultrasonic pen is used by non-dominant hand as shown in Figure 2.3.

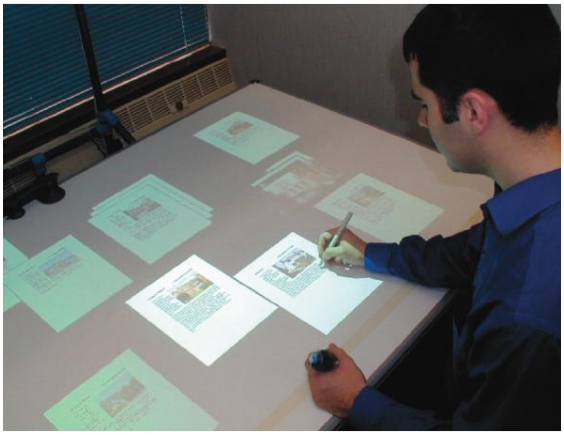

Figure 2.3: A new environment for writing and manipulating papers and documents.

Besides, user can annotate the virtual paper and save it for later viewing. Any changes can be made directly without printing the image to physical paper and then scan the revised version into the display again for later viewing. The Virtual Networking Computing (VNC) client enables applications such as Web browser to be viewed on the desk. VNC is a graphical desktop sharing system that allows user to remotely control another computer. It transmits the input event from one computer to Escritoire, to manipulate the content on Escritoire so that user can view applications such as Web browser on it. Furthermore, the notion of piles feature helps user to save space on the desk by grouping the related documents together in a pile, just like how a user deals with real papers. As user moves the digitizer pen over a pile, it will split

over to allow browsing. When the pen is lifted from the surface, the pile will revert back to its normal form and the light intensity displaying the pile will become lower.

Moreover, the employed client-server architecture links remote participants to have a shared display. Escritoire is given internet connection properties so that it can connect to the server and upload data for other participants to download and review. The server is a large database which contains sets of data while client is the user who accesses to the server, looking of the data.

The client program runs on a computer that cooperates with the projectors and pens. So, the computer will first identify and process the input from Escritoire before it sends the processed data to the server. A number of Escritoires, acting as clients, can be connected to the same server for collaborative work and sharing of information. The server will respond to events from clients and sends them updates whenever there is a change on one of the displays in the clients.

Furthermore, the implemented cursor has three options for the pens which are no cursor, crosshair and trace, and the cursor is duplicated on all the client desks. No cursor option is used for personal work as the pens are already the direct pointing devices on the desk. Crosshair is used to show the pen's position while the trace has a fading of 0.7 second to display an animated history of location. This can improve the gestural communication and interaction between participants as the cursor will draw attention and focus of each participant. This system also consists of audio, video and desk channels to assist the participants for conveying their messages rather than just the textual words.

Escritoire can minimize the obscuring problem that most of the projectors face. Obscuring is an issue for front projection as presenter can obscure the image by walking in front of the projector. But for Escritoire's display, the image is projected obliquely from the back of the desk as shown in Figure 2.2, so the user can lean considerably forward without obscuring the display. A private workspace like laptop is also provided where participants can keep items until they are needed by dragging documents from their laptops onto Escritoire.

This system has a disadvantage in large size and heavy weight. It only can be placed in office or conference room, but not as portable as mobile devices or laptops. Sometimes, real papers are better than these virtual papers provided in Escritoire as real papers can be brought to other users for further discussion or investigation. Some users are preferred in explaining to others face to face rather than just showing the words remotely. For example, most of the explanations in mathematical solutions or in architectural drawings are preferred in this way.

In conclusion, this system can reduce the amount of paper usage by digitizing the contents in the papers into documents in computer. It creates the similar pens, papers and desk environment that most people are familiar with. It is similar to desktop or laptop computers that store files and documents, but it provides pens for user to do annotations rather than using mouse or keyboard. Its remote collaboration feature makes it different, more advanced and applicable as user needs not send the document to another user in other places.

2.4 Spatial Interfaces

AlloSphere, a Multiuser Immersive Instrument, as shown in Figure 2.4, is a large scientific and artistic instrument for immersive human-centered visualization, sonification and interactive data manipulation. It serves as a laboratory room to

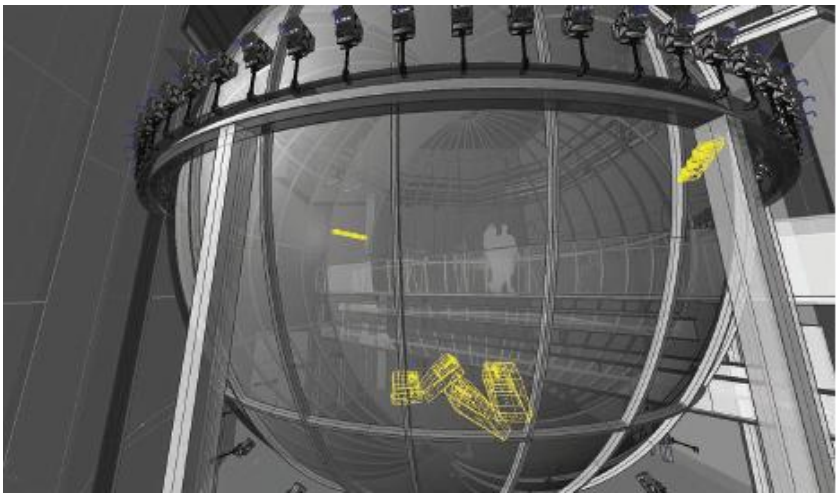

Figure 2.4: The outer view of AlloSphere.

facilitate multiuser parametric control of complex information at the same time and mimics a real-world environment with minimal artifacts. The goal is to support collaborative scientific data exploration and empower human perception and action, assist users to solve a very complex work.

Its design is very huge with a metal screen comprises two 10m diameter hemisphere and a connecting cylindrical section to form a three-storey capsule. Its internal volume could be more than 700 meter cube. There is a 2m wide bridge to support up to 30 users. Its length is same as diameter of the hemisphere. As the surrounding area is covered by screen, the secondary reflection issue becomes significant. To limit the secondary reflections and prevent users to see their images reflected by the screen, the super-black screen is used instead of just a normal screen that most of the display surfaces are employed.

Besides, sound-absorbing materials and special heating and air conditioning are installed to provide low noise and dry acoustics. The sound-absorbing materials will absorb the sound wave and convert it into heat and therefore, special heating and air conditioning is needed to remove the heat produced. Hence, users will feel comfortable when doing their experiment in AlloSphere.

There is 13-computer cluster to provide a total 41.5 million pixels surround stereoscopy graphical image display across 26 active stereo projectors. 55 independent loudspeakers are used to provide full-surround 3D spatialized audio. Moreover, tracking cameras are employed to allow surround interaction with the interfaces. AlloSphere also incorporates with wired, wireless, mobile, tablet and laptop for interaction and display.

AlloSphere supports full-surround 3D spatialized audio with controlled acoustics. Combination of visualization and sonification can overcome occlusion and cognitive-overload problem as humans can more readily to perceive and process audio in parallel. Audio can display better finer temporal structures over a wider range of frequencies than visual display. Spatializing audio in full-surround 3D is conducive to exploratory orientation and localization besides conveying distance,

relative motion and environmental properties. It is done by using third-order Ambisonics with common distance cues.

Ambisonics is a full-sphere surround sound technique as it covers sound sources above and below the listeners other than just the horizontal plane to the listeners. Third-order Ambisonics is used to increase the spatial resolution and enlarge the usable listening area as first-order Ambisonics provides a very low spatial resolution and comparably small usable listening area, and so second-order Ambisonics.

Figure 2.5: The seamless curved surround screen enables users to have a complete field of visualization.

Besides, AlloSphere's seamless curved surround screen enables nearly complete field of regard that avoids sharp distortion due to room corners as shown in Figure 2.5. It equally emphasizes visual and auditory surround display to leverage a combined audiovisual representation's benefits. Collaboration work is allowed in a shared surround-view environment and participants are either unencumbered or minimally encumbered with wireless stereo glasses.

Some spatial-interaction building block are shared which are multimodal surround display, shared and individual control and spatialized agents. Multimodal surround display includes surround stereoscopy, surround sound and surround interaction. The applications can run in either windowed mode (Figure 2.6) or full-surround mode (Figure 2.5). The full-surround mode can support surround-view

stereoscopic projection for multiple non-head-tracked users by using object-order rendering and image-order rendering.

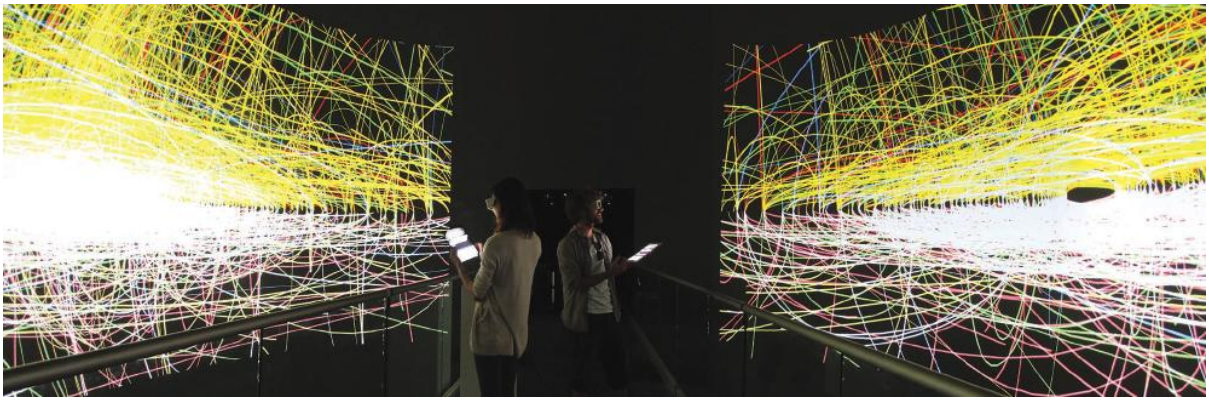

Figure 2.6: In the windowed mode, two quasi-rectangular display areas on both sides of the bridge are being used.

Rendering is the process of generating a digital image or raster graphics image file from a scene file, by means of computer programs. A scene file contains objects in a strictly defined language or data structure such as geometry viewpoint, texture, and lighting. Object-order rendering determines the effects of the pixels on the image plane for each data sample. Its algorithm loops through the data samples, projecting each sample onto the image plane. This algorithm iterates over the elements in the scene to be rendered, rather than the pixels in the image to be produced.

In contrast to object-order rendering, image-order rendering iterates over the pixels in the image to be produced, rather than the elements in the scene to be rendered. It determines the data samples for each pixel on the image plane. Ray casting is its algorithm that casts viewing rays through the volume. At discrete intervals along the ray, the three-dimensional function is reconstructed from the samples and the optical model is evaluated. This ray casting method can effectively avoid processing of occluded regions.

Furthermore, the open-source Device Server software configures spatial-navigation and interactive-control interfaces for AlloSphere applications to provide

high-level abstractions for networked interactivity. It also provides unified interface to low-level drivers for input sensing. This Device Server software provides network access to variety of hardware. Hardware access is managed in a process called a Device Server and this Device Server contains Devices, which is control objects belonging to different Device Classes with implement the hardware access. Control object can read and write device memory through I/O (input and output) port, memory mapped or device driver.

Researchers also can use their personal mobile devices as interfaces to AlloSphere content and thus, large group of interaction is enabled. Many applications also provide browser-based interfaces, extending support to laptops and requiring no software installation. The handheld-devices display textual data whereas shared display used for 3D interaction. The interactive magic lenses like tracked phones and tablets are employed to provide personalized annotated views as users are oriented toward the AlloSphere content. AlloSphere also imported a library that contains hand-based gesture recognition. As shown in Figure 2.7, user can have gestural control of the surround information.

Figure 2.7: A user employs hand gesture through wireless tracked gloves to navigate or to summon agents.

However, to support multiple types of display devices, the relations between data on personal displays and the shared display have to be clearly identified. It is

also important to distinguish each user's location and activity in the shared dataset. Thus, the tracking cameras should have a very high accuracy and reliability to separate each user even they are standing very close to each other.

Lastly, multiagent systems are used. It is a form of spatially distributed artificial intelligent to offer a computer-assisted means to distribute data exploration roles and address scenes that could be difficult to analyze and monitor unassisted. Agents are designed toward greater autonomy and intelligence, with more complex, individual, and stateful operations. Intelligence here, may include some methodic, functional, procedural or algorithmic search, find and processing approach.

Also, cooperative macrobehavior of agents is supported. Each agent can deal with other agents to simplify the workflow. These agents provide a spatial metaphor for information display to extend the arbitrary sounds, so that the sounds of agents behind a user's head or occluded by other visual elements can always be audible and localizable.

AlloSphere Research Group has developed many immersive applications, such as Graph Browser, Cloud Bridge, Allobrain, and Time of Doubles. Graph Browser and Cloud Bridge investigate the collaborative benefits of providing personal views and interaction in conjunction with the shared display.

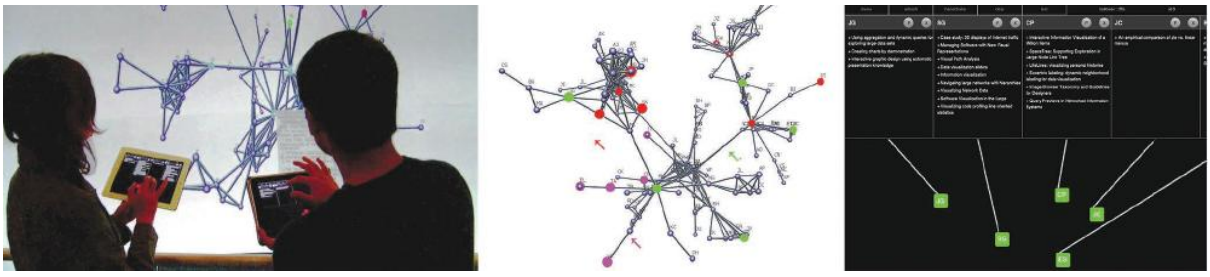

Figure 2.8: In Graph Browser, users are interacting with the shared image through their tablets (left); the shared display with different color to differentiate each user (middle); the textual data shown in tablet for user to communicate with the shared display (right).

In Graph Browser, the content types are separated by personal and shared display. Textual is displayed in personal display while structural is provided in shared display as shown in Figure 2.8. Thus, users can simultaneously explore the data through their tablets without cluttering the shared display. Each tablet only shows the specific textual data of the corresponding structural data pointed by user in the shared display.

Moreover, the text in personal display can be pushed to the shared display for sharing purpose if they think that data is important for investigating. As it supports multiple users, each user's activity has to be distinguished specifically. Hence, color is employed to provide an awareness of other users' activities, so users can focus on their own activities.

In Cloud Bridge, multiple users explore a dataset and display the textual results on their personal displays whereas the shared display presents a broad view of this dataset as a chronological spiral repeating over one-week periods. The textual results are visualized on the shared display as arcs from check-out to check-in points in the time spiral. It is different from Graph Browser as it displays more data sonically than visually. Color is also provided to create contextual awareness of each user's queries. By giving a unique amplitude envelope curve to each user, users can be identified in the shared auditory.

Allobrain is an agent-based computer-assisted data exploration application. It presents a virtual world consisting of isosurfaces of brain blood density. Each autonomous agent is an intelligent agent which is a software entity that carries out some set of operations on behalf of a user or another program with some degree of independence. It employs some knowledge or representation of the user's goals or desires and continuously emits a spatialized sound according to the agent trajectories, creating a complex, multi-path soundscape.

User can summon the agent into near-field view to report its finding. Although, only one user can navigate the dataset at a time, a maximum number of 12 users can simultaneously interact with the agents. Each agent's song is unique and is recognized by its pitch. It also incorporates a transient-rich pulse to support rapid and

accurate localization. The sonic amplitude will become higher as the agent comes into near-field view, providing user with a better observation.

Time of Doubles is the largest and most complex multiagent system developed in AlloSphere. Each agent is an evolving artificial-life organism that explores the fluid simulation, seeking particles to consume for survival, reproduction and to evolve superior search strategies, as shown in Figure 2.9. The agent acts to change part of the environment or of its status and influences what it sensed. A thousand agents are allowed to be active at a time by using just-in-time compilation where a program is compiled at run time. Each agent incorporates a distinct program combining input sensors, kinetic actions, memory and other operations. The program is generated at the agent's birth by genetic programming according to a mutated copy of the parent genome.

To sonified these agents, short pulse trains of narrow-band frequency implemented by asynchronous granular synthesis are used. The idea behind granular synthesis is the creation of complex soundscapes by combining thousands of grains of sound over the time. It is based on the production of a high density of small acoustic events called grains that are typically in the range of 10-60ms. The grain of sound is considered as a building block for the creation of more complex sonic events. While for asynchronous granular synthesis, grains are scattered randomly over a user-determined duration, with user-determined density and frequency content, depending on the waveform used in the grains.

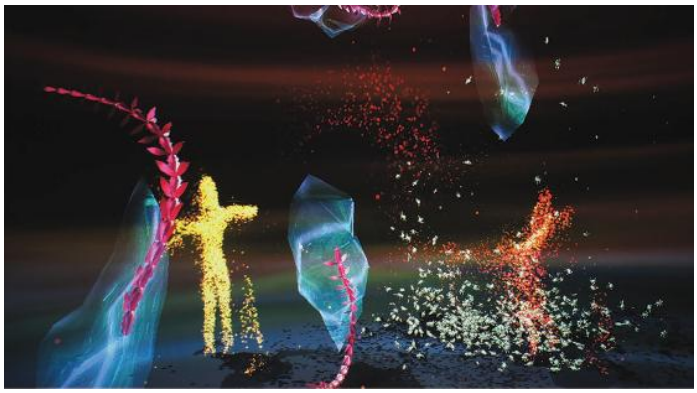

Figure 2.9: In Time of Doubles, agent seeks particles which are the representation of user's shape and movement to consume, identifying user's location and activity.

Granular pulse is easily to be localized while narrow-band width frequency allows users to identify many voices concurrently. Interaction employs an array of depth cameras. Dense cloud of food particles mirror each user's shape and movement. Three-dimensional optical flow analysis of the clouds adds forces to the fluid simulation. Optical flow is the pattern of apparent motion of objects, surfaces, and edges in a visual scene caused by the relative motion between a user and the scene. It can be used to estimate user's movement, shape, distance and locomotion. So, when agents are seeking for food particles, as shown in Figure 2.9, they can identify each user. Users can feed their doubles to growing populations of organisms and induce currents in the environment through their body movements.

However, this system has an issue of its very huge size, almost equal to a double-story building. Whenever users want to do their experiment, they have to access this building and it will become frustrated when the weather is bad. The investment and installation of this system will be very expensive as it could be difficult from just build a double-story building.

In conclusion, AlloSphere provides users with configurable interfaces, enabling them to adapt distinct responsibilities such as spatial navigation or agent control. This multi-display approach fosters collaborative action among users. It empowers human perception and action, helping them to solve complex data which is difficult to be done by a single computer.

2.5 Low-Latency Transmission Scheme for Interactive Screen Sharing

Screen sharing system allows users to share the desktop's application screen of the local computer to the remotely Internet-connected computers. The content of the local monitor will be shared and displayed on the other monitors that connected to it through Internet. Nevertheless, the remote computers can send back user inputs to control the local computer if they are given permission to do so. This screen sharing feature is useful for cooperative work, online education or sharing of information among users in different places.

In the past few years, the conventional screen sharing systems are mainly designed for the Local Area Network (LAN). However, when 3G, WiFi and ADSL come into account, which mean screen sharing can be done through these types of communication networks, the design of the systems should consider the limited bandwidth, poor latency and packet loss issues that are brought by the networks. Delay is the main issue for interactive screen sharing systems and most of the famous screen sharing systems are difficult to achieve a delay less than 100ms especially in the bandwidth-constrained error-prone network environments.

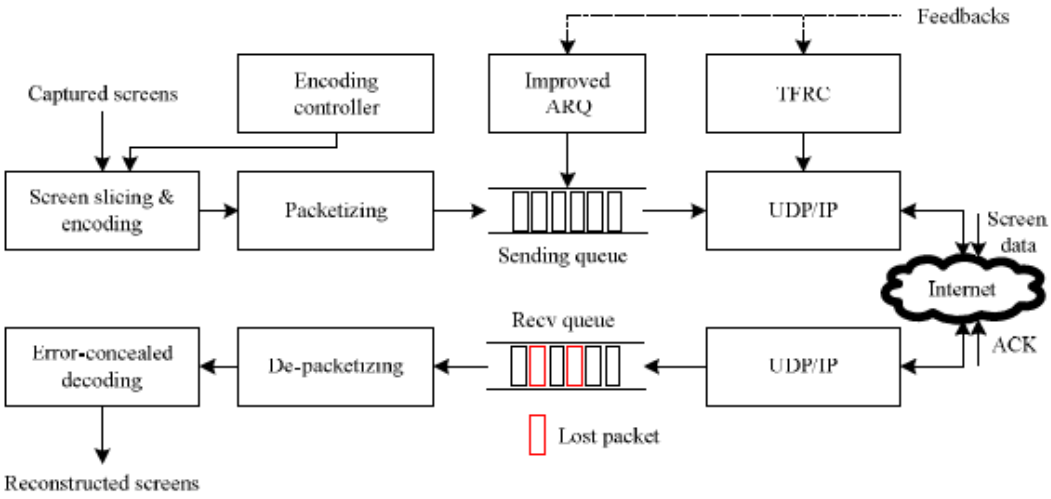

Figure 2.10: The low-latency transmission framework.

So, to improve the latency and reduce the delay when sharing the data through the network other than LAN, this paper proposes a UDP-based low-latency transmission scheme, as shown in Figure 2.10. This scheme can promote users' collaboration by providing interactive screen sharing feature. This paper first theoretically analyzes the difference in latency performance between Automatic Retransmission reQuest (ARQ) and hybrid Forward Error Correction (HFEC) for the User Datagram Protocol (UDP)-based screen sharing. Then, it proposes an improved ARQ scheme to reduce the transmission latency while considering the properties and characteristics of the main-stream screen codecs.

UDP is one of the members of the Internet protocol suite which is the set of network protocols used for the Internet. It uses a simple transmission model with connectionless protocol. Datagram or packet can be transmitted to other computers without setting up special transmission channels or data paths. Hence, it is unreliable as there is no guarantee of delivery, ordering or duplicate protection of the packet and that is why ARQ or HFEC is needed. UDP is different from Transmission Control Protocol (TCP) as TCP is a connection-oriented protocol. TCP requires handshaking to set up end-to-end communications and allows user data to be sent in bi-directional over the connection. Hence, TCP is more reliable.

ARQ is an error-control method for data transmission that uses acknowledgements and timeouts in order to achieve reliable data transmission over an unreliable network, such as 3G, WiFi or ADSL. If the sender does not receive an acknowledgment before the timeout, it usually re-transmits the packet until acknowledgment is received or a predefined re-transmissions number is exceeded. The acknowledgement from the receiver to the sender is generated automatically when it receive the packet.

FEC is a technique used for controlling errors in data transmission over unreliable communication channels, like 3G, WiFi or ADSL. Redundancy such as parity is added to the packet for the receiver to detect a limited number of errors that may occur anywhere in the packet, and often to correct these errors without the need of retransmission. To receive the packets which cannot be recovered by FEC, it is combined with ARQ to have packet retransmission and thus, the hybrid FEC/ARQ (HFEC in short) is formed. So, the receiver has a higher possibility of getting the correct packet.

In UDP-based low-latency transmission scheme, the captured screens are chopped into slices based on the content and then encoded independently at the host side as shown in Figure 2.10. The generated bits are packetized before put into the sending queue. If the queue is full, the encoding controller will stop the screen encoder until there is an adequate space in the sending queue. The packet is then sent to client via the UDP channel. To prevent the congestion in the channel, TCP-

friendly rate control (TFRC), a congestion control mechanism is employed to estimate the available bandwidth and regulate the output bit rate.

Under this congestion control mechanism, the receiver will measure the congestion control information such as the loss event rate and acknowledge this information to the sender, so that the sender can use this feedback message to determine the round-trip time (RTT). Hence, the output bit rate can be adjusted to match the calculated transmit rate using TFRC's throughput equation as a function of loss event rate and RTT.

Besides, ARQ is used in the transmission scheme rather than FEC because the packet retransmission would not block the decoding of the successive screens. The packet retransmission and decoding of packet to reconstruct the screen can be done concurrently by using ARQ. ARQ also adapts to the network conditions and is more efficient than FEC in transmission as FEC needs to transmit redundant parity packets to the receiver to correct the error that might be occurred.

In the analysis of ARQ and HFEC in latency performance for screen sharing systems, few assumptions are made taking in the consideration of transmission delay and queuing delay. One of the assumptions is that the packet erasure is independent to each other. Second, one slice is packetized to single packet and each packet can be decoded independently. Third, the delay of each packet or slice can be counted separately and averaged to get the frame delay.

Transmission delay or packetization delay is the amount of time required to push all the packet's bits into the bandwidth. It is proportional to the packet's size and is independent to the transmission distance. While queuing delay is the time a packet waits in a queue until it can be executed. If the queue being filled is faster than it can be executed, the queuing delay will increase as some of the packets have to wait for execution.

The simulation result shows that ARQ has a lower latency than HFEC. This is because residual coding is not used in ARQ since its high complexity computation and thus packet loss will not result in error-drifting in decoding. In residual coding,

the samples in the prediction residual are assumed to be uncorrelated and therefore can be coded independently by determining the appropriate form for the probability density function (p.d.f.) of the distribution of residual values.

However, in FEC, predictive residual coding is used and hence the transmission error will result in error drifting in decoding. Error drifting is a situation where the mean of many separate measurements differs significantly from the actual value. This error is very difficult to be solved as their effects are only observable if they can be removed. Hence, this error will require forward transmission and retransmission and so, the delay is introduced.

So, this paper proposes the improved ARQ scheme that will not retransmit the data of the co-located blocks in previous frames if the data of one non-skip block in some frames is received and acknowledged. The data in the co-located blocks can be replaced by new data since it is outdated. Canceling the packet retransmission can improve the latency performance as the queuing delay in the sending queue, network traffic and congestion, and transmission delay are reduced.

In conclusion, the improved ARQ is proposed to be implemented in the network transmission to increase the latency performance so that the users can share their screens with a minimum delay, for a better interaction and cooperative work in screen share.

2.6 Multisurface Interaction in the WILD Room

The WILD (wall-sized interaction with large datasets) room, shown in Figure 2.11, serves as a working laboratory for exploring a more advanced interactive system by distributing interaction across diverse computing devices. It allows multiple users to create, share, and manipulate digital content easily. This multi-surface environment is featuring a wall-sized display, a motion tracking system, a multi-touch table, and various mobile devices. It is mainly used to assist the scientists for their collaborative

work on the analysis of large and complex datasets. Its main objective is to distribute interaction, not only content, across a variety of interactive surfaces.

Figure 2.11: The WILD platform. A wall-sized display (top left); 16-computer cluster (top right); motion tracking systems (bottom left); multitouch table (bottom right).

The multi-touch table employs Frustrated Total Internal Reflection (FTIR) which is a process where an evanescent wave will pass energy across the second into the third medium. Normally the evanescent wave transmits zero net energy. FTIR is occurred when the third medium has a higher refractive index than the second medium and it is placed within less than several wavelengths distance from the interface between the first medium and the second medium. Hence, the transmission coefficient for FTIR is highly sensitive to the distance between the third medium and the second medium. This effect is useful in modulating optical transmission and reflection with a large dynamic range.

Two complementary strategies for generating and testing ideas are employed. They are participatory design and controlled experiments. The former focuses on qualitative understanding and external validity which involves users throughout the design process. By implementing the Wizard of Oz technique, one of the most effective techniques in participatory design, scientists can experiment midair hand gesture and using external props to manipulate their data such as controlling the

orientations of all 64 normal and pathological brains displayed on the wall using a 3D physical model brain prop as shown in Figure 2.12.

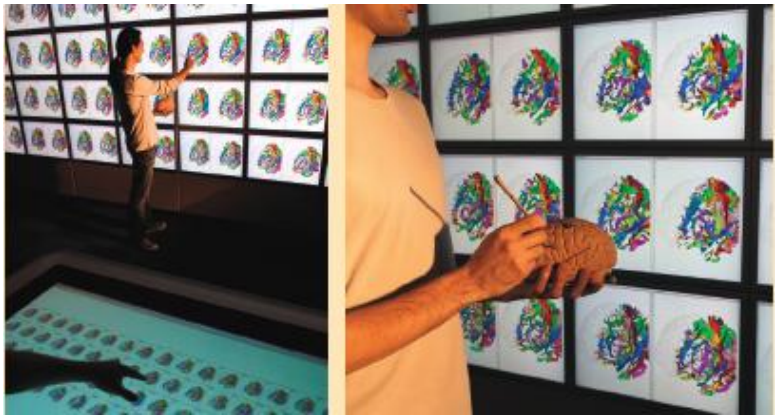

Figure 2.12: Wizard of Oz technique allows user to control the orientation of all 64 brain scans (left) using a 3D physical model brain prop (right).

A tablet also can be served as a mobile, physical interactive filter to focus on specific wavelengths in the Milky Way galaxy. There is also panning and zooming technique which allows users to have midair interaction, using the hands to point to the locus of zoom within an image on the wall display to control its expansion and contraction.

The controlled experiment focuses on quantitative evaluations and internal validity to evaluate about which factors could improve the performance, accuracy, and comfort. It is a scientific investigation in which both control group and experimental group are kept under same conditions apart from the factor under study. No treatment is given to the control group while the experimental group is changed according to the factor to determine the effect or influence of that factor. The goal is to understand the tradeoffs among each technique and help users and designers to decide which technique to be used under what circumstances.

To develop a multisurface application, few challenges have to be considered. First, applications are inherently distributed and environment is dynamic. These applications are distributed over all the computers involved in WILD room including the cluster for the wall display, computer for the interactive table and motion

tracking system, mobile devices and laptops. Second, input devices can be connected to communicate with the other surfaces and interactions can be done in parallel. Third, content comes from different sources, such as static document brought by users or live windows from legacy applications. These contents can be shifted among each surface flexibly by users.

After identified the challenges, the WILD system comes with a solution that uses an approach to separate user interaction, graphical rendering, and content sources. In distributed user interaction, multiple instruments can be used in parallel to mediate interaction between users and objects. The instruments are independent of the objects that users operate on. Users only need to know that the object is implemented with a given protocol, such as selecting, changing position, or setting a color. These instruments can be embodied in portable devices so that users can interact with objects located on other surfaces using their portable devices.

In distributed graphical rendering, users can organize their data and choose how to present or display the images or contents in multiple surfaces, either the same part or different parts of the canvas. Furthermore, an illusion of single, continuous surface with no tearing can be created on the tiled displays. Hence, to successfully develop the multisurface applications, jBricks and Shared Substance are established based on the replicated approach.

In distributed content sources, simple but effective solutions are employed so that users are able to bring up the content from multiple sources, integrating them into a unified environment, as if the various surfaces are extensions of their laptops. Users can e-mail a document to WILD or send it to a printer queue that WILD monitors to display the document on the wall. For webpage or live applications, users can fill out a simple Web form or use a bookmarklet to display them on the wall.

By distributing interaction, rendering and content, the development of multisurface applications is simplified while supporting flexible interaction as well as legacy content and applications. Users can interact with wall-sized images in real time with litter perceivable lag. In conclusion, the WILD room is designed to be an extreme environment that utilizes fully the properties and features of both hardware

and software, to support users such as scientists to share and understand complex datasets. They could bring up anything concurrently using their own interfaces such as their mobile devices. Remote collaboration among users is also allowed. It shows that multisurface interaction, not just content, can be distributed across devices.

CHAPTER 3

METHODOLOGY

3.1 Overview

The main focus of this project is to develop an Android application, known as Virtually Shared Home Screen system (VSHS) to share the other's screens remotely. The local device will retrieve remote application screen via Bluetooth and combine both local and remote application screens so that local user not only has the accessibility to his local applications, but he also able to access to the remote application screen and its applications. By applying the master and slave technology, the local device will be the slave and the remote device will be the master. The master has to configure its file access permission before allowing slave to connect to it.

After the Bluetooth connection is established and the remote screen data is transferred, the local user can directly view and obtain the remote file just like treating with his own files. Moreover, synchronization of files can be done which means local user can save the modified files back to the remote device so that remote user will have the latest version of the files. Furthermore, local user can copy any of his files and paste it to anywhere in the remote file directories. This feature allows sharing of local files to the remote user. Besides, he also can copy any read-only or writeable remote file and paste it to his local device. Hence, the file transfer is in bi-directional way. Remote file deletion also can be made to the remote writeable files as some files might be old and should be cleared.

3.2 Configuring the File Access Permission

Remote user can configure all his files before sharing his application screen and files. Three modes of file access permission are provided which are not allowed mode, read-only mode and writeable mode. Once done configuring the file access permission, the data will be saved in a text file. When the screen is to be shared, the configuration data will be read, as illustrated in Figure 3.1 to acknowledge the local device which files are in which modes.

```
try {
    //internal storage
    FileInputStream fIn = openFileInput("confimusic.txt");
    InputStreamReader isr = new InputStreamReader(fIn);
    int value1 = 0;

    char[] inputBuffer = new char[1];
    String s = "";

    int value = 0;
    value = isr.read();
    stringArray1 = new String[value];

    int charRead;
    while((charRead = isr.read(inputBuffer)) > 0) {
        //convert the chars to a string
        String readString = String.copyValueOf(inputBuffer, 0, charRead);

        if (readString.contains("\n")) {
            stringArray1[value1] = s;
            s = "";
            value1++;
        } else {
            s += readString;
        }
        inputBuffer = new char[1];
    }
    isr.close();
    fIn.close();
```

Figure 3.1 : Reading the Configuration Data

The files that are not allowed will be highlighted in red color in the local device to acknowledge the local user that these files cannot be opened. Read-only files are purple highlighted in the local device and these files can be opened, viewed and modified in local device only. Lastly, writeable files are highlighted in blue color in the local device and these files can be opened, viewed, modified and saved back to the remote device. The local device will have a copy of the opened read-only and

writeable files. The programming code that configures the colors of the file name for different file access permissions is shown in Figure 3.2.

```
for (int i = 0; i < MainActivity.numOfFolders; i++) {
    folderLabel[i] = new Names();
    folderLabel[i].folderNames = MainActivity.folderName[i];

    for(int x = 0; x < MainActivity.notAllowedSize; x++) {
        if (folderLabel[i].folderNames.equalsIgnoreCase(MainActivity.notAllowedFileName[x])) {
            folderLabel[i].color = "RED";
        } else {}
    }

    for(int y = 0; y < MainActivity.readOnlySize; y++) {
        if (folderLabel[i].folderNames.equalsIgnoreCase(MainActivity.readOnlyFileName[y])) {
            folderLabel[i].color = "PURPLE";
        } else {}
    }

    for(int z = 0; z < MainActivity.readWriteSize; z++) {
        if (folderLabel[i].folderNames.equalsIgnoreCase(MainActivity.readWriteFileName[z])) {
            folderLabel[i].color = "BLUE";
        } else {}
    }
}
```

Figure 3.2 : Configuring the Colors for different File Access Permissions.

3.3 Transmit the Screen Application Data via Bluetooth

To successfully share the screen data to another device, following subsections are needed. Missing one of them will eventually cause the Bluetooth connection unable to be setup, or failure in transmitting the screen data.

3.3.1 Bluetooth Adapter

To use the Bluetooth in the device, the device's Bluetooth adapter has to be obtained and turned on. This Bluetooth adapter contains the device's Bluetooth name and address so that all the surround Bluetooth devices can identify each other. Next step will be asking the Bluetooth adapter to scan or discover the surround devices. The corresponding programming code is illustrated in Figure 3.3.

```
BluetoothAdapter mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
TextView txt_1 = (TextView)findViewById(R.id.txt1);
txt_1.setText("My Device");
txt_1.setTextColor(Color.GREEN);

TextView txt_2 = (TextView)findViewById(R.id.txt2);
txt_2.setText("My device name : " + mBluetoothAdapter.getName() + "\n"
        + "Device address : " + mBluetoothAdapter.getAddress() + "\n");

mBluetoothAdapter.enable();
mBluetoothAdapter.startDiscovery();
```

Figure 3.3 : Handling the Bluetooth Adapter

```
Intent discoverableIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE);
startActivityForResult(discoverableIntent, REQUEST_DISCOVERABLE_CODE);
```

Figure 3.4 : Activity to set the Visibility of the Device

This activity or intent, as shown in Figure 3.4, has to be called so that the user can choose the visibility of his device. If the device is in discoverable mode, it is visible to other Bluetooth devices and can be detected by them. If the device is set to be invisible to other Bluetooth devices, it will not be discovered by other Bluetooth devices even though the device's Bluetooth is turned on. When this intent or activity is called, a dialog will appear as shown in Figure 3.5.

Figure 3.5 : Request Permission Dialog

3.3.2 Broadcast Receiver

For the device to detect the surround Bluetooth devices, the broadcast receiver software has to be used. This receiver is called every time when a remote Bluetooth device is discovered. Then this receiver will broadcast the remote Bluetooth device's name and address to acknowledge the local Bluetooth adapter so that the local Bluetooth adapter knows the presence of that remote Bluetooth device. If there is another Bluetooth device, this receiver will be called again and broadcast the corresponding remote Bluetooth device's name and address to the local Bluetooth adapter. Before using this broadcast receiver, an intent filter has to be implemented, as illustrated in Figure 3.6 to register the broadcast receiver. The corresponding programming code for the broadcast receiver is written in Figure 3.7.

```
// Register the BroadcastReceiver
IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
registerReceiver(mReceiver, filter); // Don't forget to unregister during onDestroy
```

Figure 3.6 : Intent Filter to register the Broadcast Receiver

```
final BroadcastReceiver mReceiver = new BroadcastReceiver() {
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();
        // When discovery finds a device
        if (BluetoothDevice.ACTION_FOUND.equals(action)) {
            // Get the BluetoothDevice object from the Intent
            BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
            // Add the name and address to an array adapter to show in a ListView
            if(arrayListBluetoothDevice.size() < 1) {
                // this checks if the size of BluetoothDevice is 0,then add the device to the array list.
                mArrayAdapter.add(device.getName()+"\n"+device.getAddress());
                arrayListBluetoothDevice.add(device);
                mArrayAdapter.notifyDataSetChanged();
            }
            else {
                // flag to indicate whether that particular device is already in the array list
                boolean flag = true;
                for(int i = 0; i < arrayListBluetoothDevice.size(); i++) {
                    if(device.getAddress().equals(arrayListBluetoothDevice.get(i).getAddress())) {
                        flag = false;
                    }
                }
                if(flag == true) {
                    mArrayAdapter.add(device.getName()+"\n"+device.getAddress());
                    arrayListBluetoothDevice.add(device);
                    mArrayAdapter.notifyDataSetChanged();
                }
            }
        }
    }
};
```

Figure 3.7 : Communicates with the Broadcast Receiver

3.3.3 Bluetooth Socket

Bluetooth socket is created to connect to the respect remote Bluetooth server socket, as shown in Figure 3.8 in prior to send data. The local Bluetooth socket will have a unique code to identify which remote Bluetooth server socket should be connected because the remote Bluetooth server socket is also holding the same code.

```
public RequestThread(BluetoothDevice device) {
    BluetoothSocket tmp = null;
    mmDevice = device;
    UUID MY_UUID = UUID.fromString("a60f35f0-b93a-11de-8a19-03002011c456");
    // Get a BluetoothSocket to connect with the given BluetoothDevice
    try {
    // MY_UUID is the app's UUID string, also used by the server code
        tmp = mmDevice.createInsecureRfcommSocketToServiceRecord(MY_UUID);
        mmSocket = tmp;
        // Cancel discovery because it will slow down the connection
        mBluetoothAdapter.cancelDiscovery();
        try {
            // Connect the device through the socket. This will block
            // until it succeeds or throws an exception
            mmSocket.connect();
```

Figure 3.8 : Creating Bluetooth Socket with a Unique Code

3.3.4 Output Stream Object

If the local Bluetooth socket is successfully connected to the respect remote Bluetooth server socket, the output stream object can be obtained. The device discovery process is terminated because this process will make the Bluetooth speed slower. This output stream is used to write the data that to be transmitted. After done writing the data, the output stream and Bluetooth socket are closed as shown in the programming code in Figure 3.9.

```
try {
    outStream = mmSocket.getOutputStream();
    try {
        outStream.write("request for screenshare".getBytes());
        try {
           outStream.flush();
           outStream.close();
           mmSocket.close();
           tmp.close();
```

Figure 3.9 : Obtaining Output Stream Object to write Data

3.3.5 Bluetooth Server Socket

This Bluetooth server socket is created when the Bluetooth is turned on. It holds a unique code and keeps listening to the remote Bluetooth socket which holding the same code. Hence, this Bluetooth server socket must be created before the remote Bluetooth socket is created. If any remote Bluetooth socket is successfully connected to it, the local Bluetooth socket can be obtained and the Bluetooth server socket will be closed. The corresponding programming code is explained in Figure 3.10.

```
try {
    btserver = mBluetoothAdapter.listenUsingInsecureRfcommWithServiceRecord(name, MY_UUID);
    try {
        btSocket = btserver.accept();
        btserver.close();
```

Figure 3.10 : Creating Bluetooth Server Socket to listen for Incoming Connection

If the remote Bluetooth socket is created earlier than the Bluetooth server socket, it will not be able to connect to that Bluetooth server socket because it is happening earlier. In the other word, the Bluetooth server socket will only listen to any remote Bluetooth socket that is created later than it. The order is illustrated in Figure 3.11.

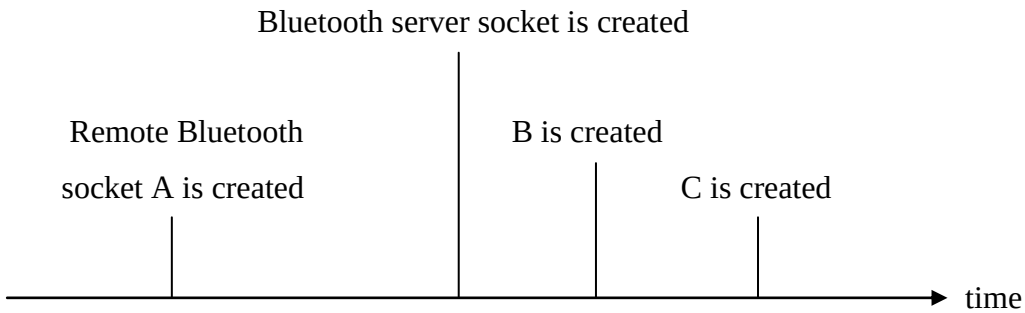

Figure 3.11 : The Sequence when Remote Bluetooth Socket is created

As remote Bluetooth socket A is created before the Bluetooth server socket, it will not be listened by the Bluetooth server socket. So A is unable to connect to the

Bluetooth server socket. For remote Bluetooth sockets B and C, as they are created after the Bluetooth server socket, they can be connected to it successfully.

3.3.6 Input Stream Object

After the local Bluetooth socket is obtained, the input stream object is retrieved and the data inside is read. The read data is then stored inside a buffer array before decoding. After data reading and storing are finished, the input stream object and the Bluetooth socket are closed as shown in the programming code in Figure 3.12. The Bluetooth server socket is created again to listen for further incoming data signal. These steps are kept repeated so that the local Bluetooth adapter is able to receive data forever if it is not turned off.

```
try {
    InputStream is = null;
    is = btSocket.getInputStream();
    int byteNo = 0;
    byte[] buffer = null;
    buffer = new byte[10000000];  // 10MB buffer size

    try {
        byteNo = is.read(buffer);
        try {
            if (byteNo != -1) {
            //ensure DATAMAXSIZE Byte is read.
                int byteNo1 = byteNo;
                int bufferSize1 = 10000000; // 10MB buffer size
                while(byteNo1 != bufferSize1) {
                    bufferSize1 = bufferSize1 - byteNo1;
                    byteNo1 = is.read(buffer, byteNo, bufferSize1);
                    if(byteNo1 == -1) {
                        break;
                    }
                    byteNo = byteNo + byteNo1;
                }
            }
            is.close();
            btSocket.close();
        }
```

Figure 3.12 : Obtaining Input Stream Object to read Data and store in Buffer

As shown in Figure 3.12, the buffer array size is limited to 10MB. If the data that to be transmitted is more than 10MB, the buffer will be overloaded and eventually causes the program to terminate. Therefore, to run the program smoothly, the shared data should be lesser than 10MB.

3.3.7 Work Flow of the Bluetooth Adapter

The work process of the server is shown in Figure 3.13.

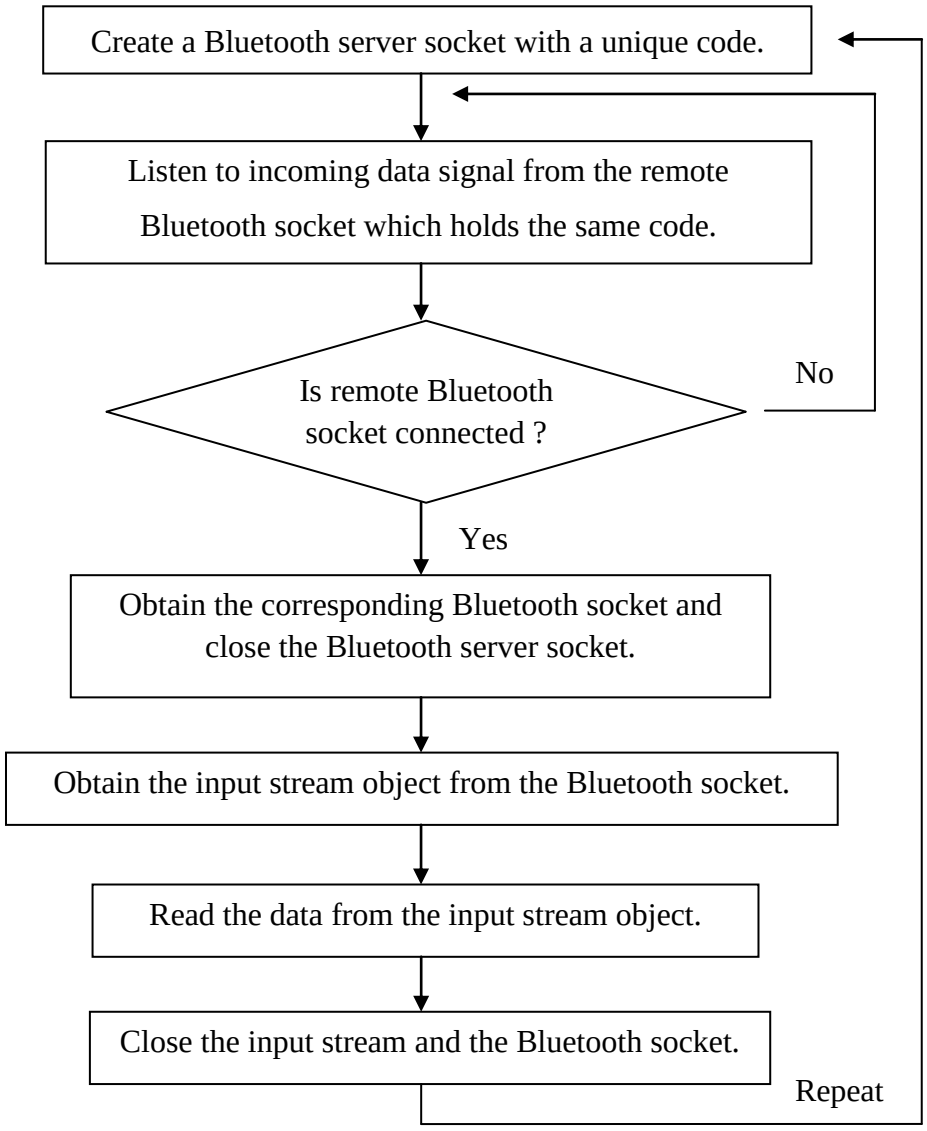

Figure 3.13 : Flow Chart of the Server

The work process of the client is illustrated in Figure 3.14.

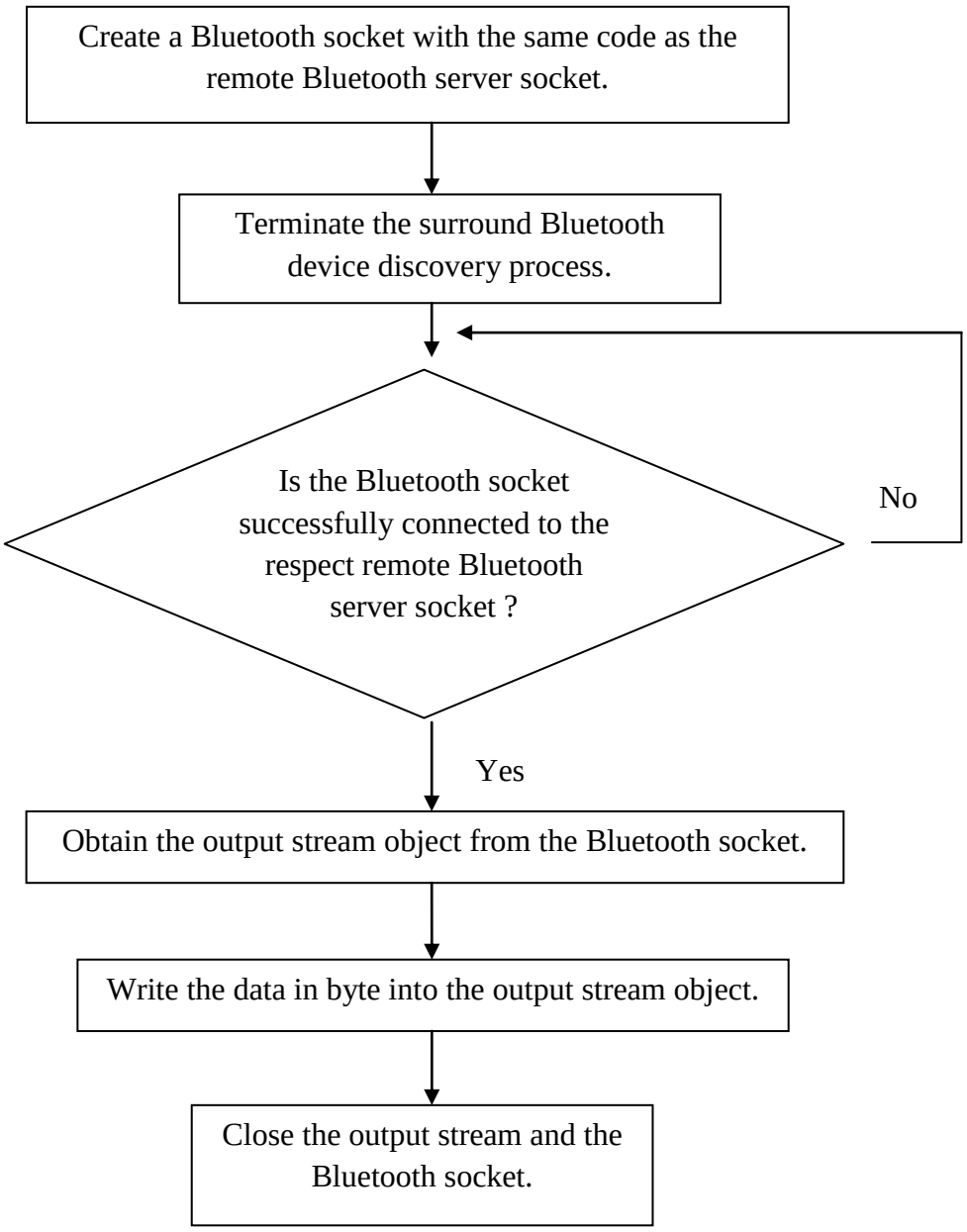

Figure 3.14 : Flow Chart of the Client

A Bluetooth Adapter can act as a server or a client depending on the command written to it. When it is asked to be a server, it will follow the work process shown in Figure 3.13. When it is asked to be a client, it will do what exactly shown in Figure 3.14.

3.3.8 Package Manager and Environment

The package manager is used to manage all the applications in the device. Hence, to obtain all the application data, the package manager has to be called by the remote device. From the package manager, each application name and icon can be retrieved. Then, all the application names and icons are put inside an array and transmitted to the local device. The local device will arrange all the application names and icons in a grid view, forming remote application screen. To call the package manager, following programming code in Figure 3.15 is needed.

```
pm = getPackageManager();
final Intent mainIntent = new Intent(Intent.ACTION_MAIN, null);
mainIntent.addCategory(Intent.CATEGORY_LAUNCHER);
List<ResolveInfo> pacsList = pm.queryIntentActivities(mainIntent, 0);
packs = new Pack[pacsList.size()];
for(int i=0 ; i < pacsList.size() ; i++){
    packs[i] = new Pack();
    packs[i].icon = pacsList.get(i).loadIcon(pm);
    packs[i].name = pacsList.get(i).activityInfo.packageName;
    packs[i].label = pacsList.get(i).loadLabel(pm).toString();
}
```

Figure 3.15 : Calling Package Manager to retrieve the Application Data

```
for(int i=0 ; i < pacsList.size() ; i++) {

    Bitmap bm = ((BitmapDrawable)packs[i].icon).getBitmap();
    ByteArrayOutputStream baos = new ByteArrayOutputStream();
    bm.compress(Bitmap.CompressFormat.PNG, 100, baos); //bm is the bitmap object
    byte[] imageByte = baos.toByteArray();
    outStream.write(imageByte);

    byte[] redundantImageByte = new byte[19950-imageByte.length];
    outStream.write(redundantImageByte);

    byte[] labelByte = packs[i].label.getBytes();
    outStream.write(labelByte);

    byte[] redundantLabelByte = new byte[50-labelByte.length];
    outStream.write(redundantLabelByte);
}
```

Figure 3.16 : Writing the Application Icon Data to the Output Stream Object

As shown in Figure 3.16, a For loop is used to write every application icon and label to the output stream object. These application icons and labels are converted into byte array before writing to the output stream object. A maximum

length of 19950 bytes is given to each icon and a maximum size of 50 bytes is given to each label. The redundant bytes are used to separate each icon and label.

Here, each application icon is assumed to have a size smaller than 19950 bytes. If the actual icon size is beyond the limit, only partial icon size up to 19950 bytes will be transmitted to the connected device, causing an incomplete icon appeared in the connected device.

The environment is a software that stores external storage directory. To obtain all the file directories, this environment software is called by the remote device, as shown in Figure 3.17. Then, all the file directories are managed and transmitted to the local device in a proper sequence order.

```
File sdCard = Environment.getExternalStorageDirectory();
File directory = new File(sdCard.getAbsolutePath());
File[] folder = directory.listFiles();
outStream.write(folder.length);

for (int i = 0; i < folder.length; i++) {
    outStream.write(folder[i].getName().getBytes());
    byte[] redundantFolderNameByte = new byte[250-folder[i].getName().getBytes().length];
    outStream.write(redundantFolderNameByte);
}
```

Figure 3.17 : Obtaining the File List and writing the File Name to the Output Stream Object

3.4 Opening Remote Read-Only or Writeable File

When the local user clicks a read-only or writeable file on the remote file list, the corresponding file name will be transmitted to the remote device. Then the remote device will start searching for that particular file in its file storage using the received file name. After that, the whole file data will be sent and made a copy inside the local device. Finally, the file will be opened automatically in the local device so that the local user does not need to search for that file in his device storage and open it manually. Hence, in the user's point of view, the process is same as he opens his local file.

The work flow of opening the remote file is illustrated in Figure 3.18..

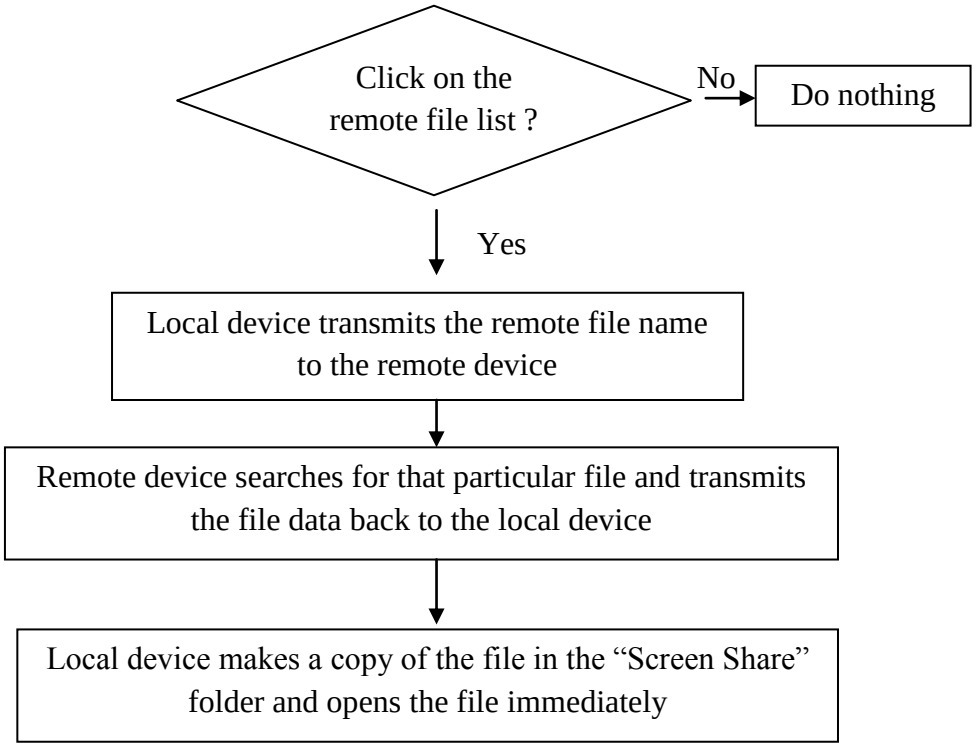

Figure 3.18 : Flow Chart when opening the Remote File

3.5 Remote File Synchronization

The local user has the ability to modify both the read-only and writeable files received from the remote device because the file is copied inside the local device. However, he only can synchronize the writeable file with the remote device so that both local and remote devices have the identical file and the remote user can read the modified file after the synchronization has been done.

A synchronization button is provided in the screen to let local user to synchronize all the modified writeable files. The local user also can choose not to synchronize the files as he might think the file has not been modified completely or correctly.

If the local user clicks the synchronization button, the local device will check for the modified files by looking at the last modified date of all the files retrieved from the remote device. Then all the modified files will be transmitted to the remote device and replace all the old version of the files. The synchronization button is located at the top right side of the screen as shown in Figure 3.19.

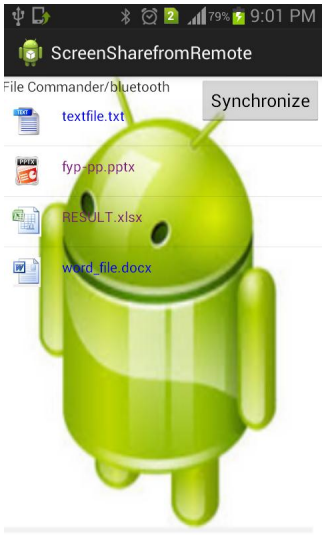

Figure 3.19 : Synchronize Button is located at the Top Right of the Screen

3.6 Copy, Paste and Delete of the File

The local user can copy his local file and paste to any of the remote directories or vice versa. This feature allows the file to be shared in bi-directional way. All the local files can be copied and pasted to the remote device but only read-only and writeable remote files can be copied and pasted to the local device. This setting assures the privacy of the remote user as the remote user can set the high privacy files to be not allowed files.

When copying local file, the local device will first read the file name and the whole file data. When local user pastes the file in any of the remote file directories, the local device will read the corresponding remote file directory and transmit the file name, file data and remote file directory to the remote device. Therefore, the remote

device can write the file with the same file name in the correct file directory. Lastly, the local device will refresh the file list shown in the screen.

When copying remote read-only or writeable file, the local device will first read the remote file name. When local user pastes the file in any of the local file directories, the local device will read the corresponding local file directory and send only the remote file name to the remote device. Then the remote device will search for that particular file using the received file name and transmit the whole file data to the local device. After receiving the whole file data from the remote device, the local device will write that file into the correct file directory by looking at the read local file directory. Lastly, the file list in the screen is refreshed.

For local file deletion, the local device will read the file name of the file that to be deleted and search for that particular file in its storage. After obtaining the file, the local device will delete the file and refresh the file list shown in the screen.

For remote file deletion, the local device will first read the remote file name of the file that to be deleted. Then, it will send the remote file name to the remote device to acknowledge the remote device which file should be deleted. After received the file name, the remote device will search and delete that particular file. Lastly, the local device will refresh the file list shown in the screen. The programming codes for reading and writing the file are illustrated in Figure 3.20 and Figure 3.21 respectively.

```
File fileDirectory3 = new File(sdCard.getAbsolutePath() + newPath);
byte[] filesByte = new byte[(int) fileDirectory3.length()];
FileInputStream fileInputStream;
try {
    fileInputStream = new FileInputStream(fileDirectory3);
    fileInputStream.read(filesByte);
    fileInputStream.close();
}
```

Figure 3.20 : Reading the File Data

```
File sdCard = Environment.getExternalStorageDirectory();
File directory = new File(sdCard.getAbsolutePath());
File fileDirectory1 = new File(directory + newPath, pastefile);
try {
    FileOutputStream fileOutputStream = new FileOutputStream(fileDirectory1);
    fileOutputStream.write(buffer, 1004, size + i*100 + j*100*100 + k*100*100*100);
    fileOutputStream.close();
}
```

Figure 3.21 : Writing Data to a File

CHAPTER 4

RESULT AND DISCUSSION

4.1 Overview

Basically, this project is all about the Android application. So the result only includes the screenshot pictures taken from the smartphone, followed by the discussion with the aid of Android programming in the Appendix.

4.2 Screen Sharing Application

When open the application, user will see the interface screen as shown in Figure 4.1.

Figure 4.1: Interface of Screen Sharing Application

There are three buttons named Bluetooth OFF, Configuration and Scan for Bluetooth devices. The Bluetooth button is used to turn on or turn off the Bluetooth of the device. When user clicks on that button, a Bluetooth permission request dialog, as shown in Figure 4.2 will appear, asking the user whether wants his device to be visible to other Bluetooth devices for 120 seconds. If the local Bluetooth is made visible, all the surround Bluetooth devices will be able to detect this local Bluetooth device. If it is made invisible, all the surround Bluetooth devices will not be able to discover it.

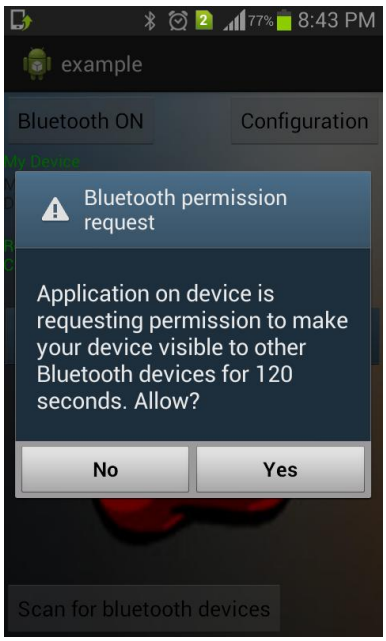

Figure 4.2 : Bluetooth Permission Request Dialog

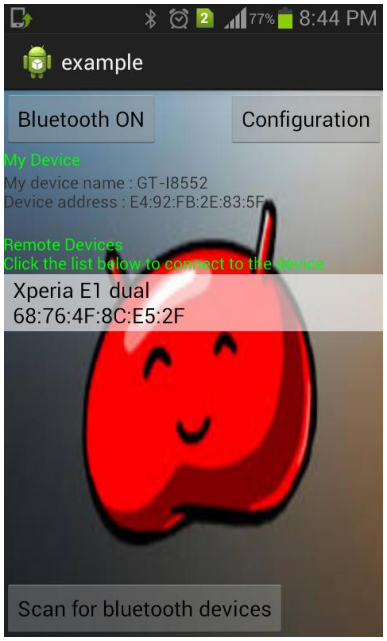

Figure 4.3 : Remote Bluetooth List

After the user made the local Bluetooth device visible, it will automatically scan for the surround Bluetooth devices. The successful detected remote Bluetooth devices' names and addresses will be shown in the list below the green text view named Remote Device, as shown in Figure 4.3. There is also another green text view named device name to indicate the local Bluetooth device's name and its address.

To construct the following layout as shown in Figure 4.3, a xml file located in res/layout in the Esclipe software is needed. In this project, the xml file is named as activity_main.xml and it can be found in the Appendix.

When the user clicks on the list to connect to a remote device to retrieve its application screen, a dialog will appear, as shown in Figure 4.4, asking the user whether wants his device to be the local device.

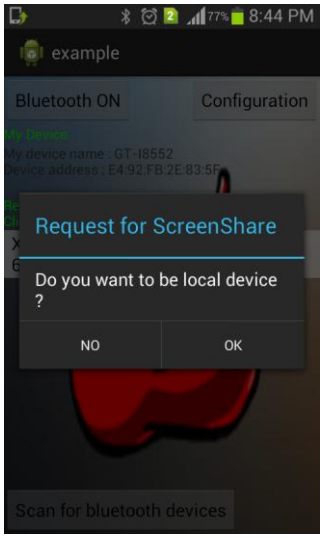

Figure 4.4 : A Local Request Dialog is appeared

If “OK” is clicked, a screen sharing request will be sent to that particular remote device and a dialog will appear in the remote device, as shown in Figure 4.5, asking the remote user whether wants to share his device’s application screen.

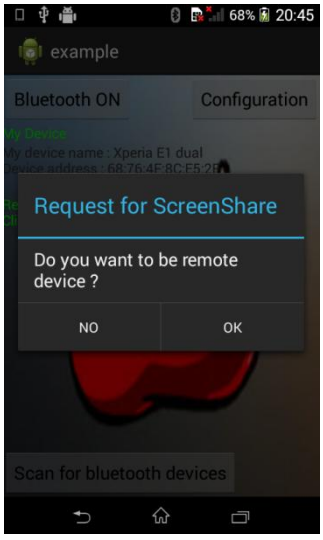

Figure 4.5 : A Remote Dialog is appeared in the Remote Device

After the remote application screen is shared, the local user can now view both his local and remote application screens. In this porject, both the local and remote devices used have three application screens. Therefore, the local user can view all the six application screens, as shown in Figure 4.6 and Figure 4.7 in his device after the remote user accepted the screen sharing request.

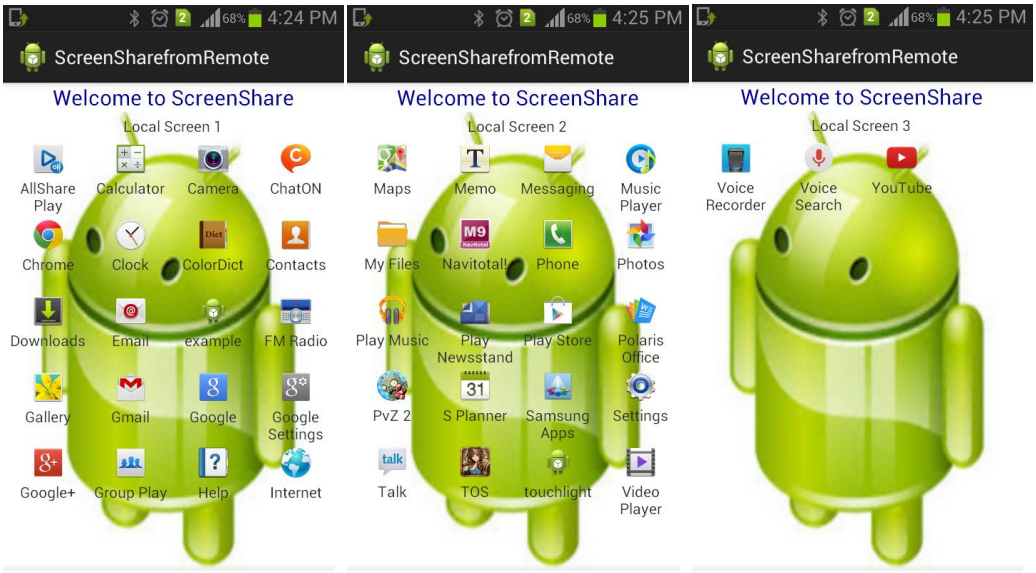

Figure 4.6 : Local Application Screens

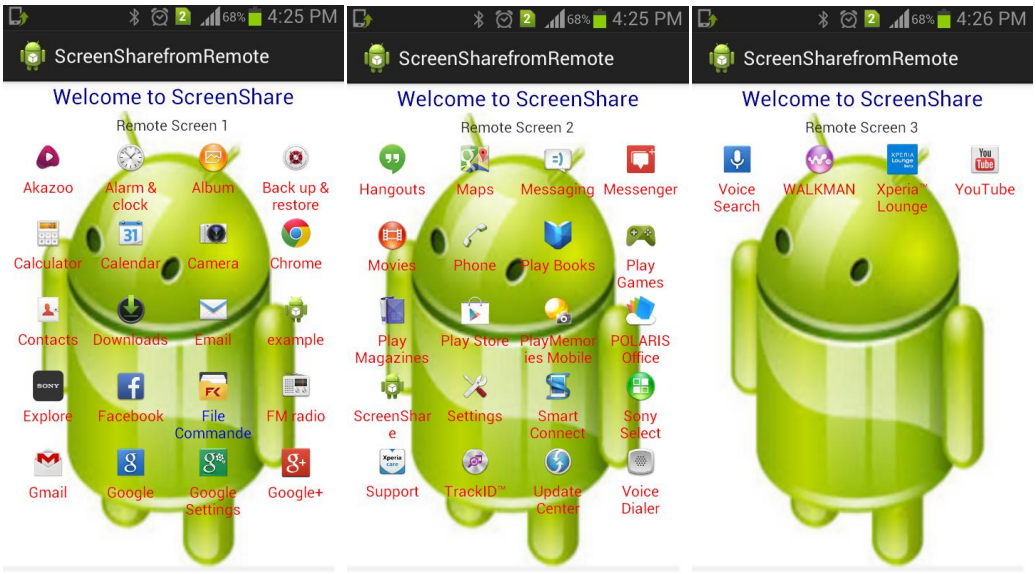

Figure 4.7 : Remote Application Screens

As usual, all the local applications can be accessed. However, only the file storage in the remote application screen can be opened. If local user clicks on the remote application other than the remote file storage, a dialog will appear, as shown in Figure 4.8, telling him that the clicked remote application is unable to be open. The main focus in this project is to complete the remote application screen with file transfer feature. Therefore, only remote file storage can be accessed. The accessibility for other remote applications, perharps, will be made in the future as it might be difficult and require more word.

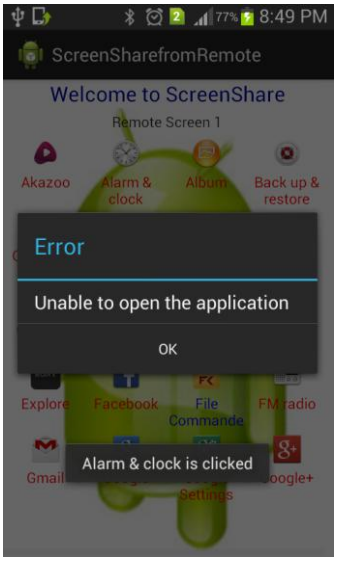

Figure 4.8 : A Dialog telling that the Remote Application cannot be opened.

4.3 Configuration of File Access Permission

It is remote user's responsibility to configure all his file before accpeting the screen sharing request. If he does not do it, all the file will be in read-only mode. In this project, the files that can be shared includes document, music, video and picture. The document file consists of word file, excel file, text file, pdf file and poweer point file. The music file can be mp3 or ogg file whereas the video file and the picture file only invovles mp4 and jpg file respectively.

One of the weaknesses in this project is a static buffer is used instead of dynamic buffer to hold the data received from the remote device. Its maximum size is 10Mb and therefore, any file larger than this size is unable to be shared.

When the user clicks on the Configuration Button in the interface screen, a new activity, as shown in Figure 4.9 will appear on top of the inteface activity.

Figure 4.9 : Four Directories to start the Configurations

Inside the music directory, all the music filles will be listed as shown in Figure 4.10(a), same for video, document and picture directories, as shown in Figure 4.10(b), 4.10(c) and 4.10(d) respectively.

Besides, performing a longpress on the list will open the corresponding file. Therefore, when the user is unsure about the content of the file, he can open and view it before configure its access permission.

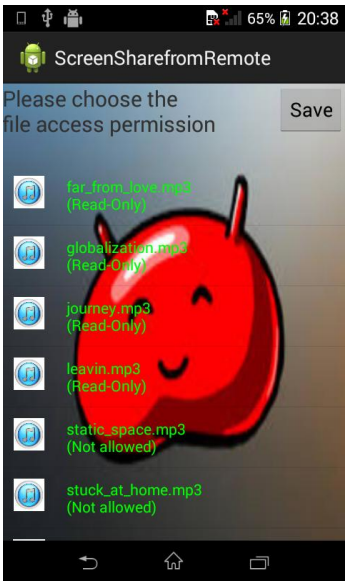

Figure 4.10(a) : Music File List

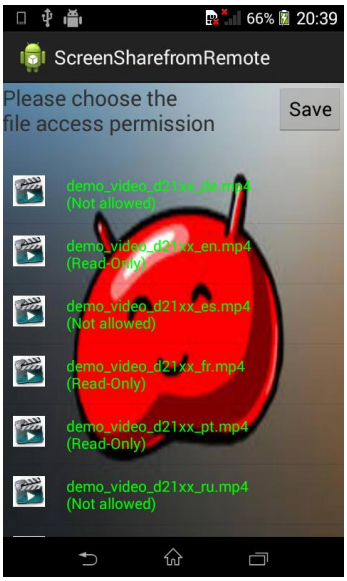

Figure 4.10(b) : Video File List

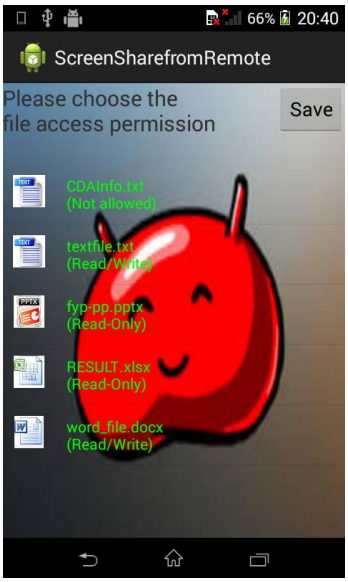

Figure 4.10(c) : Document File List

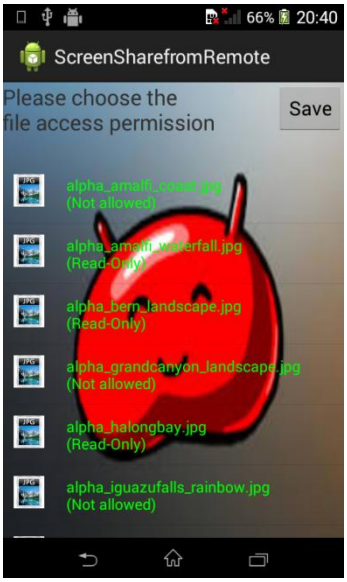

Figure 4.10(d) : Picture File List.

User can configure the file to be in not allowed mode, read-only mode or read/write mode. Files in not allowed mode will not be shared to other user connected to it. Other user can only view the file name but he cannot open the file. Both read-only and read/write file can be opened by the local user and the local user will have a copy of the file in his device once the file is openced. However, only remote read/write file allows local user to save back the modified version of that file to the remote device, where the sychronization of the file is made to both the local and remote devices.

4.4 Remote File List

Remote file list is a list of remote file names appeared virtually in the local device. As there are three different types of access permission configured by the remote user, it is necessary to differentiate which file is in which access permission mode in the local device so that the local user knows which file can be opened. Therefore, color is userd to highlight the file in the remote file list. For not allowed file, red color is used while purple color is used for read-only file and blue color is used for read/write file. Other files will be shown in black color in default. By highlighting each of the remote file, the local user can know what file can be opened or synchronized.

Once the remote read-only or read/write file is opened, the local device will have a copy of the file, making the local user has fullly control to the file. Whenever local user clicks on the files that have been opened previously, the local file will be opened instead of retrieving the remote file again.

When the local user open the remote file storage in the remote application screen, a dialog will appear on top of the remote file list, indicating that what color is used for what file access permission mode, as shown in Figure 4.11.

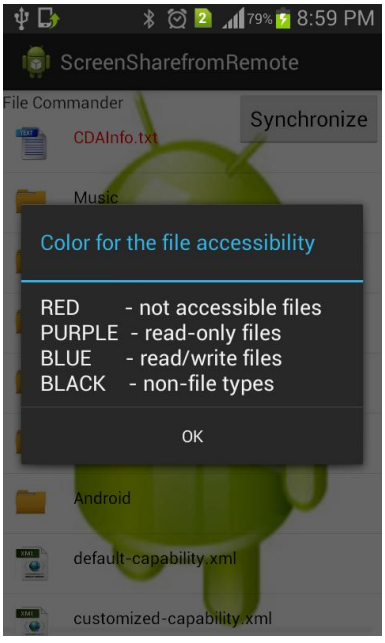

Figure 4.11 : A Dialog indicates the Colors for the File Accessibility

The local user can access to any remote directory just like accessing his own local file directory. However, there is a maximum directory level to be accessed because the shared file mainly can be found within the manimum level. In this project, the allowed directory level is set to five. If the user clicks on the list in the fifth directory, a message will appear, as shown in Figure 4.12, telling that the maximum limit is reached and beyond that limit is unaccessible.

As predicted, the remote file list will contain different colors showing that the file access modes are different, as shown in Figure 4.13(a) and 4.13(b). If user clicks on the not allowed file, a message will appear to inform him that the clicked file is not accessible.

Figure 4.12 : A Message shows the maximum Access Limit is reached

Figure 4.13(a) : Music Folder List with different Indicating Colors

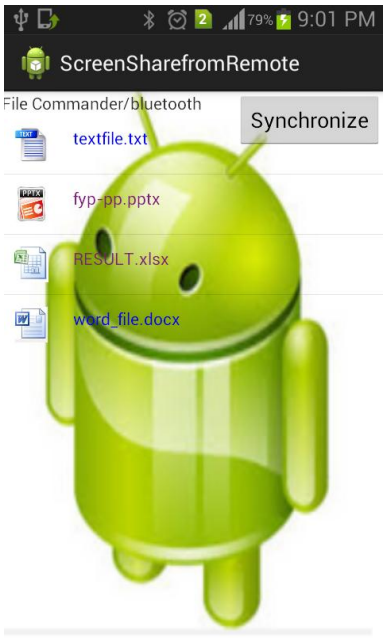

Figure 4.13(b) : Bluetooth Folder List with different Indicating Colors

The Synchronization Button is clicked whenever local user wants to save back the modified remote read/write file to the remote device. This button is created so that user can choose not to synchronize the file because the file might not be modified completely.

4.5 Copying and Pasting of File

Beside opening the remote file, local user also can copy his own file and paste it to the remote folder, or vice versa. This feature promotes a bi-directional way of file sharing. Local not only can retireve the remote file, but he also can share his own file to the remote user.

When local user wants to copy a file, for example textfile.txt, he has to input a longpress click on that particular file. Then, a window will appear consisting of three options, as shown in Figure 4.14. All the local files can be copied, but only remote read-only and read/write files can be copied by the local user.

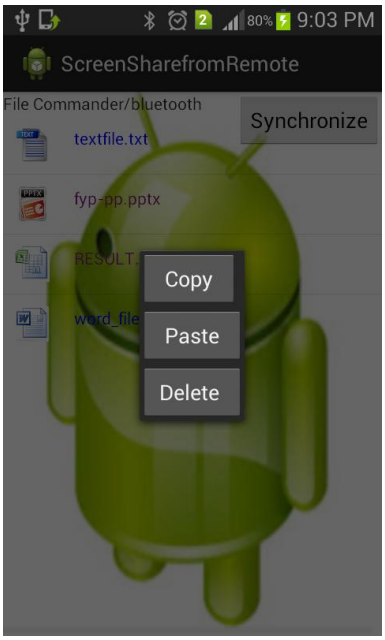

Figure 4.14 : A Window consisting Three Options is appeared

The Paste option is added in the window because some screens are fully filled by the file list and causing the user unable to input a longpress touch on the background. For that case, user only has to input a longpress click on any of the files in the list to paste the copied file into that directory.

The Delete option allows local user to delete a file in both local and remote devices. However, due to the remote user's authority, only remote read/write file can be deleted.

To paste a file, the user has to input a longpress touch on the background of the desired direcotry. A window consisting only a Paste option will appear, as shown in Figure 4.15. After pasting the file, the file list will be updated, as shown in Figure 4.16, with the pasted file. A message also appear to inform that the file has been pasted successfully. Here, the pasted file is textfile.txt.

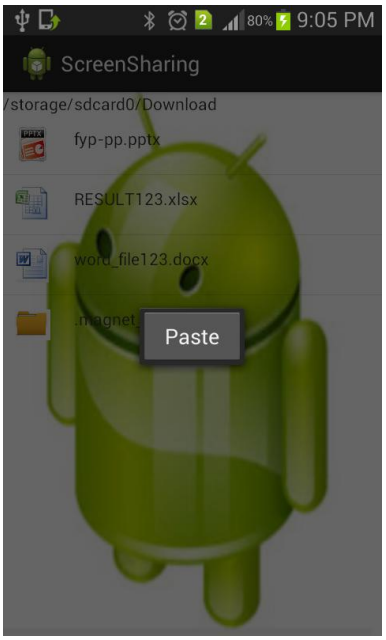

Figure 4.15 : A Paste Window

Figure 4.16 : Updated File List with Message

The work flow when copying local file to the remote device is illustrated in Figure 4.17(a).

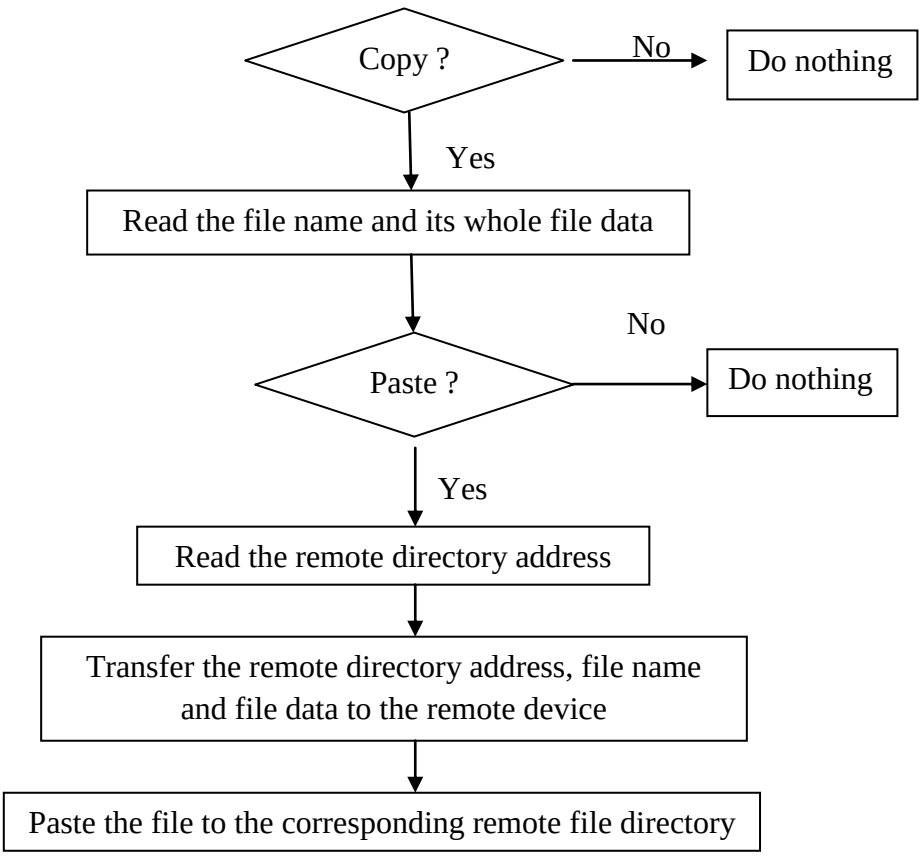

Figure 4.17(a) : Flow Chart when copying Local File to Remote Device

The work flow when copying remote file to the local device is illustrated in Figure 4.17(b).

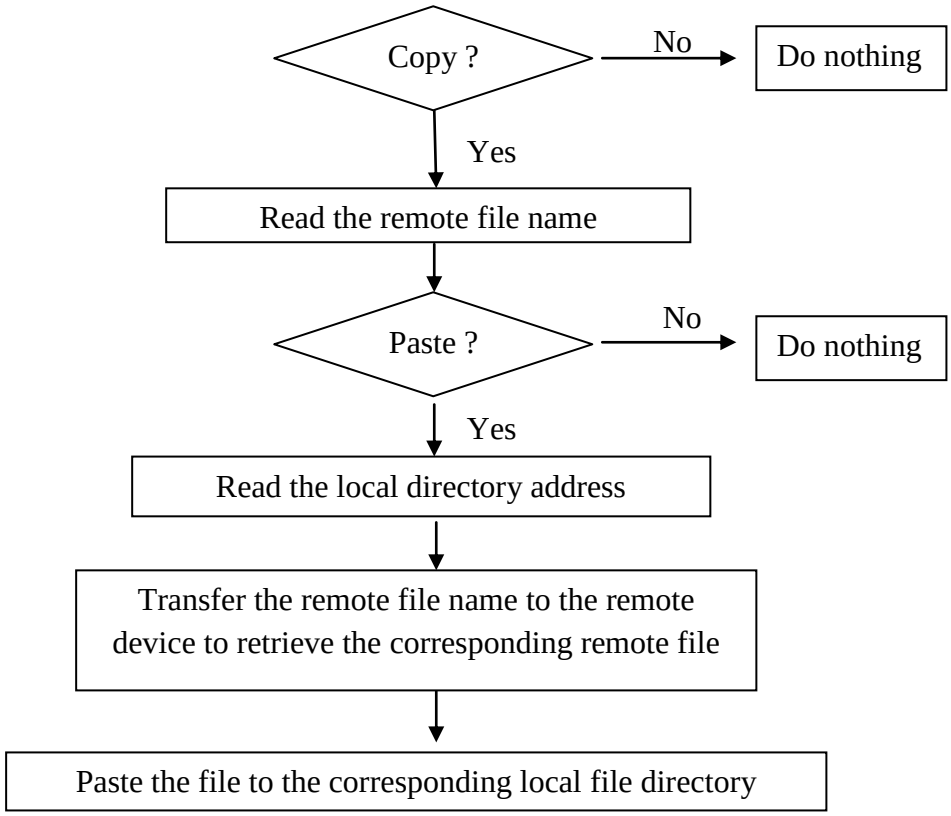

Figure 4.17(b) : Flow Chart when copying Remote File to Local Device

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.1 Conclusion

In conclusion, all the goals and objectives listed in Chapter 1 are achieved. As introduced in Chapter 1, the proposed project is Virtually Shared Home Screen system (VSHS) which is used to share the remote home screen and files among Android smart phones via Bluetooth. The remote screen is combined with the local screen to form a single screen with more content. Local user can open, modify and synchronize the remote file just like treating with his local file. VSHS also provides a seamless file transfer feature similar to the local-to-local file transfer.

However, due to the time constrain, remote application only can be viewed but cannot be accessed by the local user as the works handle it could be much more difficult. The application software such as media player or games that installed in remote device is completely “virtual” to the local device. Only the remote file storage can be accessed by the local user to handle the file transfer. This project is developed to prove the concept of sharing the remote screen and transferring of file in both directions.

Besides, the communication is only done by Bluetooth. Therefore, the data transfer speed is much slower than that of WiFi. The distance among users is also limited where the maximum range could be only 10 meters. Moreover, the number of users that can join the sharing group is limited to two persons. The third user has to wait the screen sharing session finishes before joining the group.

As the time is limited, only a certain files can be transferred among the smart phones which are document file in word, excel, power point, pdf and text format, music or video file, and image file in jpg format. As a static buffer is implemented instead of a dynamic buffer, the maximum size of the file is limited to 10 MB. Any file size larger than that cannot be shared.

Furthermore, an interesting and attractive drag-drop feature is excluded in this project due to its difficulty. This feature allows user to drag an item or a file from one screen and drop it into another screen, similar to move the item around the screens. When the item is moved to the left edge of the screen, the current screen will be swiped right and the screen left to the current screen will appear, or vice versa, until the end of the screens.

Although there are still many improvements where can be done to this VSHS, the current expected aims and objectives of this project are completed. The idea of virtually shared the remote home screen is proven and the seamless file transfer feature is developed. Regarding to the weaknesses stated above, they are proposed into future work where the time is not an issue.

5.2 Future Work and Recommendation

To fully complete this VSHS system, all the remote applications should be able to be accessed by the local user, not only the remote file storage. This enhances and extends the access control of the local user to the remote device.

The communication will include WiFi so that the sharing can be done around the world. WiFi can provide a relatively high data transfer speed and it allows unlimited users to join the communication group. Therefore, the virtual home screen is shared by all the users around the world, together share the application and files for their collaborative work.

Furthermore, all kinds of files can be transferred and the size is not limited to 10 MB anymore. On-streaming music or video can be viewed by users where they can listen to the music or watch the video while downloading the file.

The drag-drop feature also will be added into VSHS system so that user can move an item easily from one screen to another screen, making this system more interesting, attractive and user-friendly.

REFERENCES

Lee, W. M. (2012). *Beginning Android 4 Application Development*. Crosspoint Boulevard, Indianapolis: John Wiley & Sons, Inc.

Takashi, Y., Takahiro, N., & Jun, M. (2003). *Group Digital Assistant: Shared or Combined PDA Screen.* IEEE Transactions on Consumer Electronics, Vol. 49.

Mark, A., & Peter, R. (2005). *Escritoire: A Personal Projected Display.* IEEE Computer Society.

Graham, W., Tobias, H., JoAnn, K., Charles, R., & Matthew, W. (2013). Spatial Interfaces. *Spatial Interaction in a Multiuser Immersive Instrument.* IEEE Computer Society.

Pan, Z., Shen, H., Lu, Y., & Li, S. (2012). *A Low-Latency Transmission Scheme for Interactive Screen Sharing.* IEEE.

Michel, B., Huot, S., Mathieu, N., Wendy, M., Emmanuel, P., Romain, P., Julie, W., Olivier, C., Pillias, C., James, R. E., Tony, G., & Clemens, K. (2012).*Multisurface Interaction in the WILD Room.* IEEE Computer Society.

APPENDIX A

MainActivity.java

```
package com.example.example;

import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.util.ArrayList;
import java.util.List;
import java.util.UUID;

import android.net.Uri;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.Environment;
import android.os.Handler;
import android.app.Activity;
import android.app.AlertDialog;
import android.app.AlertDialog.Builder;
import android.app.Dialog;
import android.app.ProgressDialog;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.DialogInterface;
import android.content.IntentFilter;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;
import android.bluetooth.*;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.content.pm.ResolveInfo;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Color;
import android.graphics.drawable.BitmapDrawable;
import android.graphics.drawable.Drawable;
import android.widget.Button;

public class MainActivity extends Activity {
   static int abc = 0;
   public static final int REQUEST_DISCOVERABLE_CODE = 42;
   static BluetoothAdapter mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
   static BluetoothDevice requestDevice;
   static int progress = 0;
   int progressStatus = 0;
   Handler handler = new Handler();
   static UUID MY_UUID = UUID.fromString("a60f35f0-b93a-11de-8a19-03002011c456");

   ArrayAdapter<String> mArrayAdapter;
```

```
ArrayList<BluetoothDevice> arrayListBluetoothDevice = new ArrayList<BluetoothDevice>();
class Pack{
   Drawable icon;
   String name;
   String label;
}
Pack[] packs;
PackageManager pm;

static String[] sb;
static Bitmap[] bitmap;

static String[] folderName;
static String[] subFolderName;
static Integer[] numOfSubFolder;
static String[] subSubFolderName;
static Integer[] numOfSubSubFolder;
static String[] subSubFolder1Name;
static Integer[] numOfSubSubFolder1;
static String[] subSubFolder2Name;
static Integer[] numOfSubSubFolder2;

static int numOfApps;
static int numOfFolders;
int numOfSubFolders;
int numOfSubSubFolders;
int numOfSubSubFolders1;
int numOfSubSubFolders2;

static int totalfiles;
int totalsubfiles;
int totalsubSubfiles;
int totalsubSubfiles1;

static int notAllowedSize;
static String[] notAllowedFileName;
static int readOnlySize;
static String[] readOnlyFileName;
static int readWriteSize;
static String[] readWriteFileName;

static File file[];
static int num;
File fileDirectory;
File filepath;
static long date[];

File fileDirectory1;

static String BDaddress;

static int done;

@Override
public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);

   ListView listView = (ListView)findViewById(R.id.listViewRemoteDevice);
```

```
mArrayAdapter = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1);
listView.setAdapter(mArrayAdapter);
listView.setBackgroundColor(-1677721601);
listView.setOnItemClickListener(new ListItemClicked());

final ProgressBar progressBar = (ProgressBar)findViewById(R.id.progressbar);
progressBar.setVisibility(View.INVISIBLE);

TextView txt_1 = (TextView)findViewById(R.id.txt1);
txt_1.setText("My Device");
txt_1.setTextColor(Color.GREEN);

TextView txt_2 = (TextView)findViewById(R.id.txt2);
txt_2.setText("My device name : " + mBluetoothAdapter.getName() + "\n"
      + "Device address : " + mBluetoothAdapter.getAddress() + "\n");

TextView txt_3 = (TextView)findViewById(R.id.txt3);
txt_3.setText("Remote Devices\n" + "Click the list below to connect to the device");
txt_3.setTextColor(Color.GREEN);

pm = getPackageManager();

set_packs();

IntentFilter filter1 = new IntentFilter();
filter1.addAction(Intent.ACTION_PACKAGE_ADDED);
filter1.addAction(Intent.ACTION_PACKAGE_REMOVED);
filter1.addAction(Intent.ACTION_PACKAGE_CHANGED);
filter1.addDataScheme("package");
registerReceiver(new PacReceiver(), filter1);

final BroadcastReceiver mReceiver = new BroadcastReceiver() {
   public void onReceive(Context context, Intent intent) {
      String action = intent.getAction();
      if (BluetoothDevice.ACTION_FOUND.equals(action)) {
         BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
         if(arrayListBluetoothDevice.size() < 1) {
            mArrayAdapter.add(device.getName()+"\n"+device.getAddress());
            arrayListBluetoothDevice.add(device);
            mArrayAdapter.notifyDataSetChanged();
         }
         else{
            boolean flag = true;
            for(int i = 0; i < arrayListBluetoothDevice.size(); i++){
               if(device.getAddress().equals(arrayListBluetoothDevice.get(i).getAddress())){
                  flag = false;
               }
            }
            if(flag == true){
               mArrayAdapter.add(device.getName()+"\n"+device.getAddress());
               arrayListBluetoothDevice.add(device);
               mArrayAdapter.notifyDataSetChanged();
            }
         }
      }
   }
};

final Button btnBT = (Button)findViewById(R.id.btn_0);
btnBT.setText("Bluetooth OFF");
```

```
mBluetoothAdapter.disable();
btnBT.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View v) {
      if (!mBluetoothAdapter.isEnabled()) {
         btnBT.setText("Bluetooth ON");

         mArrayAdapter.clear();
         arrayListBluetoothDevice.clear();

         mBluetoothAdapter.enable();

         final ProgressDialog dialog = ProgressDialog.show(
               MainActivity.this, "Turning On Bluetooth", "Please wait...", true);
            new Thread(new Runnable(){
               public void run(){
                  try {
                     Thread.sleep(2000);
                     dialog.dismiss();

                     Intent discoverableIntent = new
                              Intent(BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE);
                     startActivityForResult(discoverableIntent,
                              REQUEST_DISCOVERABLE_CODE);

                     mBluetoothAdapter.startDiscovery();
                     // Register the BroadcastReceiver
                     IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
                     registerReceiver(mReceiver, filter);
                     new DoCountingClientTask().execute();
                  } catch (InterruptedException e) {
                     e.printStackTrace();
                  }
               }
            }).start();
      }
      else {
         btnBT.setText("Bluetooth OFF");
         mBluetoothAdapter.cancelDiscovery();
         progressBar.setVisibility(View.INVISIBLE);

         final ProgressDialog dialog = ProgressDialog.show(
               MainActivity.this, "Turning Off Bluetooth", "Please wait...", true);
            new Thread(new Runnable(){
               public void run(){
                  try {
                     Thread.sleep(2000);
                     mBluetoothAdapter.disable();
                     dialog.dismiss();
                  } catch (InterruptedException e) {
                     e.printStackTrace();
                  }
               }
            }).start();
            mArrayAdapter.clear();
            arrayListBluetoothDevice.clear();
      }
   }
});
```

```
Button btnConfi = (Button)findViewById(R.id.btn_configuration);
btnConfi.setText("Configuration");
btnConfi.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View v) {
      // TODO Auto-generated method stub
      startActivity(new Intent("com.example.example.configuration"));
   }
});

Button btnScan = (Button)findViewById(R.id.btn_1);
btnScan.setText("Scan for bluetooth devices");
btnScan.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View v) {
      if (!mBluetoothAdapter.isEnabled()) {
         Builder builder = new Builder(MainActivity.this, AlertDialog.THEME_HOLO_DARK);
         builder.setTitle("Error");
         builder.setCancelable(true);
         builder.setMessage("Please Turn On Bluetooth");
         builder.setNeutralButton("OK", new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int which) {
               dialog.cancel();
            }
         });
         builder.show();
      }
      else {
      progress = 0;
      progressStatus = 0;

      mArrayAdapter.clear();
      arrayListBluetoothDevice.clear();
      progressBar.setVisibility(View.VISIBLE);

      mBluetoothAdapter.cancelDiscovery();

      mBluetoothAdapter.startDiscovery();

      // Register the BroadcastReceiver
      IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
      registerReceiver(mReceiver, filter); // Don't forget to unregister during onDestroy

      new Thread(new Runnable(){
         public void run(){
            //do some work here
            while(progressStatus < 12) {
               progressStatus = doSomeWork();

               //update the progress bar
               handler.post(new Runnable(){
                  public void run(){
                     progressBar.setProgress(progressStatus);
                  }
               });
            }

            //hide the progress bar
            handler.post(new Runnable(){
               public void run(){
```

```
                    // 0-Visible; 4-Invisible; 8-Gone
                    progressBar.setVisibility(View.INVISIBLE);
                    mBluetoothAdapter.cancelDiscovery();
                }
            });
        }

        //do some long running work here
        private int doSomeWork(){
            try {
                //simulate doing some work
                Thread.sleep(1000);
            }catch (InterruptedException e){
                e.printStackTrace();
            }
            return ++progress;
        }
    }).start();
  }
    }
  });
}

@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
  if (requestCode == REQUEST_DISCOVERABLE_CODE) {
    if (resultCode > 100) {
      showDialog(0);
    } else {
      showDialog(1);
    }
  } else {}
}

@Override
protected Dialog onCreateDialog(int id) {
  switch (id) {
  case 0:
    Builder builder = new Builder(this, AlertDialog.THEME_HOLO_DARK);
    builder.setCancelable(true);
    builder.setMessage("Bluetooth is now visible to other devices");
    builder.setNeutralButton("OK", new DialogInterface.OnClickListener() {
      public void onClick(DialogInterface dialog, int which) {
        mBluetoothAdapter.startDiscovery();
        dialog.cancel();
      }
    });
    return builder.create();

  case 1:
    Builder builder1 = new Builder(this, AlertDialog.THEME_HOLO_DARK);
    builder1.setCancelable(true);
    builder1.setMessage("Bluetooth is invisible to other devices");
    builder1.setNeutralButton("OK", new DialogInterface.OnClickListener() {
      public void onClick(DialogInterface dialog, int which) {
        dialog.cancel();
      }
    });
    return builder1.create();
  }
```

```
    return null;
}

class ListItemClicked implements OnItemClickListener {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        // TODO Auto-generated method stub
        requestDevice = arrayListBluetoothDevice.get(position);

        Builder builder = new Builder(MainActivity.this, AlertDialog.THEME_HOLO_DARK);
        builder.setTitle("Request for ScreenShare");
        builder.setCancelable(true);
        builder.setMessage("Do you want to be local device ?");
        builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int which) {
                final RequestThread requestThread = new RequestThread(requestDevice);
                requestThread.run();
                dialog.cancel();
            }
        });

        builder.setNegativeButton("NO", new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int which) {
                dialog.cancel();
            }
        });
        builder.show();
    }
}

private class RequestThread extends Thread {
    private BluetoothSocket mmSocket = null;
    private BluetoothDevice mmDevice = null;
    private OutputStream outStream = null;

    public RequestThread(BluetoothDevice device) {
        BluetoothSocket tmp = null;
        mmDevice = device;
        // Get a BluetoothSocket to connect with the given BluetoothDevice
        try {
        // MY_UUID is the app's UUID string, also used by the server code
            tmp = mmDevice.createInsecureRfcommSocketToServiceRecord(MY_UUID);
            mmSocket = tmp;

            // Cancel discovery because it will slow down the connection
            mBluetoothAdapter.cancelDiscovery();

            try {
                // Connect the device through the socket. This will block
                // until it succeeds or throws an exception
                mmSocket.connect();
                try {
                    outStream = mmSocket.getOutputStream();
                    try {
                        outStream.write("request for screenshare".getBytes());
                        Toast.makeText(getBaseContext(), "ScreenShare request made",
                                Toast.LENGTH_SHORT).show();

                        try {
                          outStream.flush();
```

```
                    outStream.close();
                    mmSocket.close();
                    tmp.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
         } catch (IOException e) {
            e.printStackTrace();
         }
      } catch (IOException e) {
         e.printStackTrace();
      }
   }  catch (IOException connectException) {}
  }
}

public class DoCountingClientTask extends AsyncTask<Void, Void, String> {
   String name = "bluetoothComm";
   BluetoothServerSocket btserver = null;
   BluetoothSocket serverSocket = null;
   String requestStr = "";
   String pathDirectory;
   @Override
   protected String doInBackground(Void... params) {
      try {
         btserver = mBluetoothAdapter.listenUsingInsecureRfcommWithServiceRecord(name,
                  MY_UUID);
         try {
            serverSocket = btserver.accept();
            btserver.close();
            try {
               InputStream is = null;
               BDaddress = serverSocket.getRemoteDevice().getAddress();
               is = serverSocket.getInputStream();
               int byteNo = 0;
               byte[] buffer = null;
               buffer = new byte[10000000];

               try {
                  byteNo = is.read(buffer);
                  try {
                  if (byteNo != -1) {
                  //ensure DATAMAXSIZE Byte is read.
                     int byteNo1 = byteNo;
                     int bufferSize1 = 10000000;
                     while(byteNo1 != bufferSize1) {
                        bufferSize1 = bufferSize1 - byteNo1;
                        byteNo1 = is.read(buffer, byteNo, bufferSize1);
                        if(byteNo1 == -1) {
                           break;
                        }
                        byteNo = byteNo + byteNo1;
                     }
                  }
                  } catch (IOException e) {}
                  requestStr = "";
                  requestStr = requestStr + new String(buffer, 0, 45);
```

```
    pathDirectory = "";
    pathDirectory = pathDirectory + new String(buffer, 50, 440);

  if (requestStr.contains("sending screen data")) {
    sb = null;
    bitmap = null;

    folderName = null;
    subFolderName = null;
    numOfSubFolder = null;
    subSubFolderName = null;
    numOfSubSubFolder = null;
    subSubFolder1Name = null;
    numOfSubSubFolder1 = null;
    subSubFolder2Name = null;
    numOfSubSubFolder2 = null;

    numOfApps = 0;
    numOfFolders = 0;
    numOfSubFolders = 0;
    numOfSubSubFolders = 0;
    numOfSubSubFolders1 = 0;
    numOfSubSubFolders2 = 0;

    totalfiles = 0;
    totalsubfiles = 0;
    totalsubSubfiles = 0;
    totalsubSubfiles1 = 0;

    notAllowedSize = 0;
    notAllowedFileName = null;
    readOnlySize = 0;
    readOnlyFileName = null;
    readWriteSize = 0;
    readWriteFileName = null;

    numOfApps = buffer[500];
    numOfFolders = buffer[501];

    bitmap = new Bitmap[numOfApps];
    sb = new String[numOfApps];
    for (int i = 0; i < numOfApps; i ++) {
      bitmap[i] = BitmapFactory.decodeByteArray(buffer, 502 + 20000*i, 19950);
      String result = "";
      result = result + new String(buffer, 502 + 19950 + 20000*i, 25);
      sb[i] = result;
    }

    if (numOfFolders > 0) {
      folderName = new String[numOfFolders];
      for (int i = 0; i < numOfFolders; i ++) {
        String result = "";
        result = result + new String(buffer, 502 + 20000*numOfApps + 250*i, 240);
        folderName[i] = result;
      }

      int temp = 0;

      numOfSubFolder = new Integer[numOfFolders];
      for (int i = 0; i < numOfFolders; i ++) {
```

```
    int numOfBytesforSubFolders = 0;

    numOfBytesforSubFolders = buffer[502 + 20000*numOfApps +
                                    250*numOfFolders + i + temp];
    numOfSubFolders = 0;
    for (int j = 0; j < numOfBytesforSubFolders; j++) {
                    numOfSubFolders = numOfSubFolders + buffer[502 +
                    20000*numOfApps + 250*numOfFolders + i + temp + j + 1];
    }
    temp = temp + numOfBytesforSubFolders;

    numOfSubFolder[i] = numOfSubFolders;

    if (numOfSubFolders == -1) {
       numOfSubFolders = 0;
    } else {}
    totalfiles = totalfiles + numOfSubFolders;

}

subFolderName = new String[totalfiles];
for (int i = 0; i < totalfiles; i ++) {
    String result = "";
    result = result + new String(buffer, 502 + 20000*numOfApps +
            250*numOfFolders + 1*numOfFolders + temp + 250*i, 240);
    subFolderName[i] = result;
}

if (totalfiles > 0) {
    int temp1 = 0;

    numOfSubSubFolder = new Integer[totalfiles];
    for (int i = 0; i < totalfiles; i ++) {
       int numOfBytesforSubSubFolders = 0;

       numOfBytesforSubSubFolders = buffer[502 + 20000*numOfApps +
                                        250*numOfFolders + 1*numOfFolders +
                                        temp + 250*totalfiles + i + temp1];
       numOfSubSubFolders = 0;
       for (int j = 0; j < numOfBytesforSubSubFolders; j++) {
                        numOfSubSubFolders = numOfSubSubFolders + buffer[502 +
                                                20000*numOfApps +
                                                250*numOfFolders +
                                                1*numOfFolders + temp +
                                                250*totalfiles + i + temp1 + j + 1];
       }
       temp1 = temp1 + numOfBytesforSubSubFolders;

       numOfSubSubFolder[i] = numOfSubSubFolders;

       if (numOfSubSubFolders == -1) {
          numOfSubSubFolders = 0;
       } else {}
       totalsubfiles = totalsubfiles + numOfSubSubFolders;

    }

    subSubFolderName = new String[totalsubfiles];
    for (int i = 0; i < totalsubfiles; i ++) {
       String result = "";
```

```
                result = result + new String(buffer, 502 + 20000*numOfApps +
                        250*numOfFolders + 1*numOfFolders + temp +
                        250*totalfiles + 1*totalfiles + temp1 + 250*i, 240);
    subSubFolderName[i] = result;
}

if (totalsubfiles > 0) {
    int temp2 = 0;

    numOfSubSubFolder1 = new Integer[totalsubfiles];
    for (int i = 0; i < totalsubfiles; i ++) {
        int numOfBytesforSubSubFolders1 = 0;

        numOfBytesforSubSubFolders1 = buffer[502 + 20000*numOfApps +
                                        250*numOfFolders + 1*numOfFolders +
                                        temp + 250*totalfiles + 1*totalfiles +
                                        temp1 + 250*totalsubfiles + i + temp2];
        numOfSubSubFolders1 = 0;
        for (int j = 0; j < numOfBytesforSubSubFolders1; j++) {
                numOfSubSubFolders1 = numOfSubSubFolders1 + buffer[502 +
                                        20000*numOfApps +
                                        250*numOfFolders + 1*numOfFolders +
                                        temp + 250*totalfiles + 1*totalfiles +
                                        temp1 + 250*totalsubfiles + i + temp2 + j
                                        + 1];
        }
        temp2 = temp2 + numOfBytesforSubSubFolders1;

        numOfSubSubFolder1[i] = numOfSubSubFolders1;

        if (numOfSubSubFolders1 == -1) {
            numOfSubSubFolders1 = 0;
        } else {}
        totalsubSubfiles = totalsubSubfiles + numOfSubSubFolders1;

    }

    subSubFolder1Name = new String[totalsubSubfiles];
    for (int i = 0; i < totalsubSubfiles; i ++) {
        String result = "";
       result = result + new String(buffer, 502 + 20000*numOfApps +
                250*numOfFolders + 1*numOfFolders + temp + 250*totalfiles +
                1*totalfiles + temp1 + 250*totalsubfiles + 1*totalsubfiles + temp2 +
                250*i, 240);
        subSubFolder1Name[i] = result;
    }

    if (totalsubSubfiles > 0) {
        int temp3 = 0;

        numOfSubSubFolder2 = new Integer[totalsubSubfiles];
        for (int i = 0; i < totalsubSubfiles; i ++) {
            int numOfBytesforSubSubFolders2 = 0;

           numOfBytesforSubSubFolders2 = buffer[502 + 20000*numOfApps +
                                            250*numOfFolders + 1*numOfFolders
                                            + temp + 250*totalfiles + 1*totalfiles +
                                            temp1 + 250*totalsubfiles +
                                            1*totalsubfiles + temp2 +
                                           250*totalsubSubfiles + i + temp3];
```

```
    numOfSubSubFolders2 = 0;
    for (int j = 0; j < numOfBytesforSubSubFolders2; j++) {
            numOfSubSubFolders2 = numOfSubSubFolders2 + buffer[502 +
                                    20000*numOfApps +
                                    250*numOfFolders + 1*numOfFolders +
                                    temp + 250*totalfiles + 1*totalfiles +
                                    temp1 + 250*totalsubfiles +
                                    1*totalsubfiles + temp2 +
                                    250*totalsubSubfiles + i + temp3 + j + 1];
    }
    temp3 = temp3 + numOfBytesforSubSubFolders2;

    numOfSubSubFolder2[i] = numOfSubSubFolders2;

    if (numOfSubSubFolders2 == -1) {
       numOfSubSubFolders2 = 0;
    } else {}
    totalsubSubfiles1 = totalsubSubfiles1 + numOfSubSubFolders2;

}

subSubFolder2Name = new String[totalsubSubfiles1];
for (int i = 0; i < totalsubSubfiles1; i ++) {
    String result = "";
    result = result + new String(buffer, 502 + 20000*numOfApps +
            250*numOfFolders  + 1*numOfFolders + temp + 250*totalfiles +
            1*totalfiles + temp1 + 250*totalsubfiles + 1*totalsubfiles + temp2
            + 250*totalsubSubfiles + 1*totalsubSubfiles + temp3 + 250*i,
            240);
    subSubFolder2Name[i] = result;
}

notAllowedSize = buffer[502 + 20000*numOfApps + 250*numOfFolders
                    + 1*numOfFolders + temp + 250*totalfiles + 1*totalfiles
                    + temp1 + 250*totalsubfiles + 1*totalsubfiles + temp2 +
                    250*totalsubSubfiles + 1*totalsubSubfiles + temp3 +
                    250*totalsubSubfiles1];

notAllowedFileName = new String[notAllowedSize];
for (int i = 0; i < notAllowedSize; i++) {
    String result = "";
    result = result + new String(buffer, 502 + 20000*numOfApps +
            250*numOfFolders + 1*numOfFolders + temp + 250*totalfiles +
            1*totalfiles + temp1 + 250*totalsubfiles + 1*totalsubfiles + temp2
            + 250*totalsubSubfiles + 1*totalsubSubfiles + temp3 +
            250*totalsubSubfiles1 + 1 + 250*i, 240);
    notAllowedFileName[i] = result;
}

readOnlySize = buffer[502 + 20000*numOfApps + 250*numOfFolders +
                1*numOfFolders + temp + 250*totalfiles + 1*totalfiles +
                temp1 + 250*totalsubfiles + 1*totalsubfiles + temp2 +
                250*totalsubSubfiles + 1*totalsubSubfiles + temp3 +
                250*totalsubSubfiles1 + 1 + 250*notAllowedSize];

readOnlyFileName = new String[readOnlySize];
for (int i = 0; i < readOnlySize; i++) {
    String result = "";
    result = result + new String(buffer, 502 + 20000*numOfApps +
            250*numOfFolders + 1*numOfFolders + temp + 250*totalfiles +
```

```
                        1*totalfiles + temp1 +  250*totalsubfiles + 1*totalsubfiles + temp2 +
                        250*totalsubSubfiles + 1*totalsubSubfiles + temp3 +
                        250*totalsubSubfiles1 + 1 + 250*notAllowedSize + 1 + 250*i, 240);
                    readOnlyFileName[i] = result;
                }

                readWriteSize = buffer[502 + 20000*numOfApps + 250*numOfFolders +
                                1*numOfFolders + temp + 250*totalfiles + 1*totalfiles +
                                temp1 + 250*totalsubfiles +   1*totalsubfiles + temp2 +
                                250*totalsubSubfiles + 1*totalsubSubfiles + temp3 +
                                250*totalsubSubfiles1 + 1 + 250*notAllowedSize + 1 +
                                250*readOnlySize];

                readWriteFileName = new String[readWriteSize];
                for (int i = 0; i < readWriteSize; i++) {
                    String result = "";
                    result = result + new String(buffer, 502 + 20000*numOfApps +
                            250*numOfFolders + 1*numOfFolders + temp + 250*totalfiles +
                            1*totalfiles + temp1 +  250*totalsubfiles + 1*totalsubfiles +
                            temp2 + 250*totalsubSubfiles + 1*totalsubSubfiles + temp3 +
                            250*totalsubSubfiles1 + 1 +  250*notAllowedSize + 1 +
                            250*readOnlySize + 1 + 250*i, 240);
                    readWriteFileName[i] = result;
                }
                file = new File[readOnlySize + readWriteSize];
                date = new long[readOnlySize + readWriteSize];
                num = 0;
            } else{}
        } else{}
    } else {}
} else {}

} else if (requestStr.contains("here is the file")) {
    File sdCard = Environment.getExternalStorageDirectory();
    File folderName = new File(sdCard.getAbsolutePath() + "/ScreenShare");
    folderName.mkdirs();

    int fileSize = 0;
    fileSize = buffer[500];
    String fileNames = "";
    fileNames = fileNames + new String(buffer, 501, fileSize);

    int size = 0;
    int i = 0;
    int j = 0;
    int k = 0;

    size = buffer[1000];
    i = buffer[1001];
    j = buffer[1002];
    k = buffer[1003];

    file[num] = new File(folderName, fileNames);
    FileOutputStream fileOutputStream = new FileOutputStream(file[num]);
    fileOutputStream.write(buffer, 1004, size + i*100 + j*100*100 + k*100*100*100);
    fileOutputStream.close();
    file[num].setWritable(true);
    date[num] = file[num].lastModified();
    num++;
```

```
} else if (requestStr.contains("copied file is ready")) {
   int fileSize = 0;
   fileSize = buffer[500];
   String fileNames = "";
   fileNames = fileNames + new String(buffer, 501, fileSize);

   int size = 0;
   int i = 0;
   int j = 0;
   int k = 0;

   size = buffer[1000];
   i = buffer[1001];
   j = buffer[1002];
   k = buffer[1003];

   File fileName = new
                  File(Environment.getExternalStorageDirectory().getAbsolutePath()
                  + nextpage.listClick1 + nextpage.listClick2 + nextpage.listClick3 +
                  nextpage.listClick4, fileNames);
   FileOutputStream fileOutputStream = new FileOutputStream(fileName);
   fileOutputStream.write(buffer, 1004, size + i*100 + j*100*100 + k*100*100*100);
   fileOutputStream.close();

} else if (requestStr.contains("file saved back")) {
   int sizes = 0;
   int iss = 0;
   int js = 0;
   int ks = 0;
   int filesSize = 0;
   int fileLength = 0;
   fileLength = buffer[500];

   for (int x = 0; x < fileLength; x++) {
      int fileSize = 0;
      fileSize = buffer[501 + x*(500 + 4 + sizes + iss*100 + js*100*100 +
               ks*100*100*100)];
      String path = "";
      path = path + new String(buffer, 502 + x*(500 + 4 + sizes + iss*100 + js*100*100
            + ks*100*100*100), fileSize);

      File sdCard = Environment.getExternalStorageDirectory();
      File directory = new File(sdCard.getAbsolutePath());
      File[] folder = directory.listFiles();

      String newPath = "";
      for (int a = 0; a < folder.length; a++) {
         if (path.contains(folder[a].getName()) && folder[a].isFile()) {
            newPath = "/" + folder[a].getName();
         }
         else {
            File[] subFolder = folder[a].listFiles();
            if (subFolder != null) {
               if(subFolder.length > 0) {
                  for (int b = 0; b < subFolder.length; b++) {

            if (path.contains(subFolder[b].getName()) && subFolder[b].isFile())  {
               newPath = "/" + folder[a].getName() + "/" + subFolder[b].getName();
            }
            else {
```

```
            File[] subSubFolder = subFolder[b].listFiles();
            if (subSubFolder != null) {
               if(subSubFolder.length > 0) {
                  for (int c = 0; c < subSubFolder.length; c++) {

         if (path.contains(subSubFolder[c].getName()) && subSubFolder[c].isFile()) {
            newPath = "/" + folder[a].getName() + "/" + subFolder[b].getName() + "/"
                     + subSubFolder[c].getName();
          }
          else {
            File[] subSubFolder1 = subSubFolder[c].listFiles();
            if (subSubFolder1 != null) {
               if(subSubFolder1.length > 0) {
                  for (int d = 0; d < subSubFolder1.length; d++) {

          if (path.contains(subSubFolder1[d].getName()) &&
                              subSubFolder1[d].isFile()) {
            newPath = "/" + folder[a].getName() + "/" + subFolder[b].getName() + "/"
                     + subSubFolder[c].getName() + "/" +
                     subSubFolder1[d].getName();
          }
          else {
            File[] subSubFolder2 = subSubFolder1[d].listFiles();
               if (subSubFolder2 != null) {
                  if(subSubFolder2.length > 0) {
                     for (int e = 0; e < subSubFolder2.length; e++) {

         if (path.contains(subSubFolder2[e].getName()) &&
                              subSubFolder2[e].isFile()) {
            newPath = "/" + folder[a].getName() + "/" + subFolder[b].getName() + "/"
                     + subSubFolder[c].getName() + "/" +
                     subSubFolder1[d].getName() + "/" +
                     subSubFolder2[e].getName();
         } else {}
                     }
                  } else {}
               } else {}
         }
                  }
               } else {}
            } else {}
         }
                  }
               } else {}
            } else {}

         }
                  }
               } else {}
            } else {}
         }
      }
   }

   int size = 0;
   int i = 0;
   int j = 0;
   int k = 0;

   size = buffer[1001 + x*(500 + 4 + sizes + iss*100 + js*100*100 +
          ks*100*100*100)];
```

```
        i = buffer[1002 + x*(500 + 4 + sizes + iss*100 + js*100*100 + ks*100*100*100)];
        j = buffer[1003 + x*(500 + 4 + sizes + iss*100 + js*100*100 + ks*100*100*100)];
        k = buffer[1004 + x*(500 + 4 + sizes + iss*100 + js*100*100 +
          ks*100*100*100)];

        fileDirectory = null;
        fileDirectory = new File(sdCard.getAbsolutePath() + newPath);
        FileOutputStream fileOutputStream = new FileOutputStream(fileDirectory);
        fileOutputStream.write(buffer, 1005 + x*(500 + 4 + sizes + iss*100 + js*100*100
                        + ks*100*100*100), size + i*100 + j*100*100 +
                        k*100*100*100);
        fileOutputStream.close();
        filesSize = filesSize + fileSize;
        sizes = sizes + size;
        iss = iss + i;
        js = js + j;
        ks = ks + k;
      }
    } else if (requestStr.contains("upload file to remote")) {
      File sdCard = Environment.getExternalStorageDirectory();
      File folderName = new File(sdCard.getAbsolutePath() + "/ScreenShare");
      folderName.mkdirs();

      int fileSize = 0;
      fileSize = buffer[500];
      String fileNames = "";
      fileNames = fileNames + new String(buffer, 501, fileSize);

      int size = 0;
      int i = 0;
      int j = 0;
      int k = 0;

      size = buffer[1000];
      i = buffer[1001];
      j = buffer[1002];
      k = buffer[1003];

      filepath = new File(folderName, fileNames);
      FileOutputStream fileOutputStream = new FileOutputStream(filepath);
      fileOutputStream.write(buffer, 1004, size + i*100 + j*100*100 + k*100*100*100);
      fileOutputStream.close();

    } else if (requestStr.contains("paste file to remote")) {
      int fileSize0 = 0;
      int fileSize1 = 0;
      fileSize0 = buffer[500];
      fileSize1 = buffer[501];
      String path = "";
      path = path + new String(buffer, 502, fileSize0 + 100*fileSize1);

      int fileSize2 = 0;
      fileSize2 = buffer[502 + fileSize0 + 100*fileSize1];
      String pastefile = "";
      pastefile = pastefile + new String(buffer, 502 + fileSize0 + 100*fileSize1 + 1,
                fileSize2);

      File sdCard = Environment.getExternalStorageDirectory();
      File directory = new File(sdCard.getAbsolutePath());
      File[] folder = directory.listFiles();
```

```
String newPath = "";
for (int a = 0; a < folder.length; a++) {
        if (path.contains(folder[a].getName())) {}
        else {
           File[] subFolder = folder[a].listFiles();
           if (subFolder != null) {
              if(subFolder.length > 0) {
                 for (int b = 0; b < subFolder.length; b++) {

           if (path.contains(subFolder[b].getName()))  {
              newPath = "/" + folder[a].getName();
           }
           else {
              File[] subSubFolder = subFolder[b].listFiles();
              if (subSubFolder != null) {
                 if(subSubFolder.length > 0) {
                    for (int c = 0; c < subSubFolder.length; c++) {

           if (path.contains(subSubFolder[c].getName())) {
              newPath = "/" + folder[a].getName() + "/" + subFolder[b].getName();
           }
           else {
              File[] subSubFolder1 = subSubFolder[c].listFiles();
              if (subSubFolder1 != null) {
                 if(subSubFolder1.length > 0) {
                    for (int d = 0; d < subSubFolder1.length; d++) {

           if (path.contains(subSubFolder1[d].getName())) {
              newPath = "/" + folder[a].getName() + "/" + subFolder[b].getName() + "/"
                        + subSubFolder[c].getName();
           }
           else {
              File[] subSubFolder2 = subSubFolder1[d].listFiles();
                 if (subSubFolder2 != null) {
                    if(subSubFolder2.length > 0) {
                       for (int e = 0; e < subSubFolder2.length; e++) {

          if (path.contains(subSubFolder2[e].getName())) {
             newPath = "/" + folder[a].getName() + "/" + subFolder[b].getName() + "/"
                       + subSubFolder[c].getName() + "/" +
                       subSubFolder1[d].getName();
          } else {}
                    }
                 } else {}
              } else {}
          }
                 }
              } else {}
           } else {}
          }
                  }
              } else {}
           } else {}

          }
                 }
              } else {}
          } else {}
        }
```

```
        }

        int size = 0;
        int i = 0;
        int j = 0;
        int k = 0;

        size = buffer[1000];
        i = buffer[1001];
        j = buffer[1002];
        k = buffer[1003];

        fileDirectory1 = new File(directory + newPath, pastefile);
        FileOutputStream fileOutputStream = new FileOutputStream(fileDirectory1);
        fileOutputStream.write(buffer, 1004, size + i*100 + j*100*100 +
                               k*100*100*100);
        fileOutputStream.close();

    } else if (requestStr.contains("paste file to empty folder in remote")) {
       int fileSize0 = 0;
       int fileSize1 = 0;
       fileSize0 = buffer[500];
       fileSize1 = buffer[501];
       String path = "";
       path = path + new String(buffer, 502, fileSize0 + 100*fileSize1);

       int fileSize2 = 0;
       fileSize2 = buffer[502 + fileSize0 + 100*fileSize1];
       String pastefile = "";
       pastefile = pastefile + new String(buffer, 502 + fileSize0 + 100*fileSize1 + 1,
                   fileSize2);

       File sdCard = Environment.getExternalStorageDirectory();
       File directory = new File(sdCard.getAbsolutePath());
       File[] folder = directory.listFiles();

       String newPath = "";
       for (int a = 0; a < folder.length; a++) {
            if (path.contains(folder[a].getName())) {
               newPath = "/" + folder[a].getName();
            }
            else {
               File[] subFolder = folder[a].listFiles();
               if (subFolder != null) {
                  if(subFolder.length > 0) {
                     for (int b = 0; b < subFolder.length; b++) {

               if (path.contains(subFolder[b].getName()))  {
                  newPath = "/" + folder[a].getName() + "/" + subFolder[b].getName();
               }
               else {
                  File[] subSubFolder = subFolder[b].listFiles();
                  if (subSubFolder != null) {
                     if(subSubFolder.length > 0) {
                        for (int c = 0; c < subSubFolder.length; c++) {

               if (path.contains(subSubFolder[c].getName())) {
                  newPath = "/" + folder[a].getName() + "/" + subFolder[b].getName() + "/"
                            + subSubFolder[c].getName();
               }
```

```
                else {
                    File[] subSubFolder1 = subSubFolder[c].listFiles();
                    if (subSubFolder1 != null) {
                        if(subSubFolder1.length > 0) {
                            for (int d = 0; d < subSubFolder1.length; d++) {

                if (path.contains(subSubFolder1[d].getName())) {
                    newPath = "/" + folder[a].getName() + "/" + subFolder[b].getName() + "/"
                            + subSubFolder[c].getName() + "/" +
                            subSubFolder1[d].getName();
                }
                else {
                    File[] subSubFolder2 = subSubFolder1[d].listFiles();
                        if (subSubFolder2 != null) {
                            if(subSubFolder2.length > 0) {
                                for (int e = 0; e < subSubFolder2.length; e++) {

               if (path.contains(subSubFolder2[e].getName())) {
                   newPath = "/" + folder[a].getName() + "/" + subFolder[b].getName() + "/"
                            + subSubFolder[c].getName() + "/" +
                            subSubFolder1[d].getName() + "/" +
                            subSubFolder2[e].getName();
                } else {}
                            }
                        } else {}
                    } else {}
                }
                        }
                    } else {}
                } else {}
                }
                        }
                    } else {}
                } else {}

                }
                        }
                    } else {}
                } else {}
            }
        }

        int size = 0;
        int i = 0;
        int j = 0;
        int k = 0;

        size = buffer[1000];
        i = buffer[1001];
        j = buffer[1002];
        k = buffer[1003];

        fileDirectory1 = new File(directory + newPath, pastefile);
        FileOutputStream fileOutputStream = new FileOutputStream(fileDirectory1);
        fileOutputStream.write(buffer, 1004, size + i*100 + j*100*100 +
                                k*100*100*100);
        fileOutputStream.close();

} else if (requestStr.contains("copy and paste file in remote")) {
    int fileSize0 = 0;
```

```
int fileSize1 = 0;
fileSize0 = buffer[500];
fileSize1 = buffer[501];
String pathCopy = "";
pathCopy = pathCopy + new String(buffer, 502, fileSize0 + 100*fileSize1);

File sdCard = Environment.getExternalStorageDirectory();
File directory = new File(sdCard.getAbsolutePath());
File[] folder = directory.listFiles();

String newPath = "";
for (int a = 0; a < folder.length; a++) {
   if (pathCopy.contains(folder[a].getName())) {
      newPath = "/" + folder[a].getName();
   }
   else {
      File[] subFolder = folder[a].listFiles();
      if (subFolder != null) {
         if(subFolder.length > 0) {
            for (int b = 0; b < subFolder.length; b++) {

   if (pathCopy.contains(subFolder[b].getName()) && subFolder[b].isFile())  {
      newPath = "/" + folder[a].getName() + "/" + subFolder[b].getName();
   }
   else {
      File[] subSubFolder = subFolder[b].listFiles();
      if (subSubFolder != null) {
         if(subSubFolder.length > 0) {
            for (int c = 0; c < subSubFolder.length; c++) {

 if (pathCopy.contains(subSubFolder[c].getName()) && subSubFolder[c].isFile()) {
      newPath = "/" + folder[a].getName() + "/" + subFolder[b].getName() + "/" +
          subSubFolder[c].getName();
   }
   else {
      File[] subSubFolder1 = subSubFolder[c].listFiles();
      if (subSubFolder1 != null) {
         if(subSubFolder1.length > 0) {
            for (int d = 0; d < subSubFolder1.length; d++) {

   if (pathCopy.contains(subSubFolder1[d].getName()) &&
                              subSubFolder1[d].isFile()) {
      newPath = "/" + folder[a].getName() + "/" + subFolder[b].getName() + "/" +
                subSubFolder[c].getName() + "/" + subSubFolder1[d].getName();
   }
   else {
      File[] subSubFolder2 = subSubFolder1[d].listFiles();
      if (subSubFolder2 != null) {
         if(subSubFolder2.length > 0) {
            for (int e = 0; e < subSubFolder2.length; e++) {

   if (pathCopy.contains(subSubFolder2[e].getName()) &&
                              subSubFolder2[e].isFile()) {
      newPath = "/" + folder[a].getName() + "/" + subFolder[b].getName() + "/" +
                 subSubFolder[c].getName() + "/" + subSubFolder1[d].getName() +
                 "/" + subSubFolder2[e].getName();
   } else {}
            }
            } else {}
         } else {}
```

```
    }
            }
            } else {}
        } else {}
    }
            }
            } else {}
        } else {}
    }
            }
            } else {}
        } else {}
    }
}
File fileDirectory3 = new File(sdCard.getAbsolutePath() + newPath);
byte[] filesByte = new byte[(int) fileDirectory3.length()];
FileInputStream fileInputStream;
try {
    fileInputStream = new FileInputStream(fileDirectory3);
    fileInputStream.read(filesByte);
    fileInputStream.close();
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

int fileSize2 = 0;
int fileSize3 = 0;
fileSize2 = buffer[502 + fileSize0 + 100*fileSize1];
fileSize3 = buffer[503 + fileSize0 + 100*fileSize1];
String pathPaste = "";
pathPaste = pathPaste + new String(buffer, 504 + fileSize0 + 100*fileSize1,
            fileSize2 + 100*fileSize3);

newPath = "";
for (int a = 0; a < folder.length; a++) {
    if (pathPaste.contains(folder[a].getName())) {}
    else {
        File[] subFolder = folder[a].listFiles();
        if (subFolder != null) {
            if(subFolder.length > 0) {
                for (int b = 0; b < subFolder.length; b++) {

    if (pathPaste.contains(subFolder[b].getName()))  {
        newPath = "/" + folder[a].getName();
    }
    else {
        File[] subSubFolder = subFolder[b].listFiles();
        if (subSubFolder != null) {
            if(subSubFolder.length > 0) {
                for (int c = 0; c < subSubFolder.length; c++) {

    if (pathPaste.contains(subSubFolder[c].getName())) {
        newPath = "/" + folder[a].getName() + "/" + subFolder[b].getName();
    }
    else {
        File[] subSubFolder1 = subSubFolder[c].listFiles();
        if (subSubFolder1 != null) {
            if(subSubFolder1.length > 0) {
                for (int d = 0; d < subSubFolder1.length; d++) {
```

```
            if (pathPaste.contains(subSubFolder1[d].getName())) {
               newPath = "/" + folder[a].getName() + "/" + subFolder[b].getName() + "/" +
                        subSubFolder[c].getName();
            }
            else {
               File[] subSubFolder2 = subSubFolder1[d].listFiles();
               if (subSubFolder2 != null) {
                  if(subSubFolder2.length > 0) {
                     for (int e = 0; e < subSubFolder2.length; e++) {

            if (pathPaste.contains(subSubFolder2[e].getName())) {
               newPath = "/" + folder[a].getName() + "/" + subFolder[b].getName() + "/" +
                         subSubFolder[c].getName() + "/" + subSubFolder1[d].getName();
            } else {}
                     }
                  } else {}
               } else {}
            }
                     }
                  } else {}
               } else {}
            }
                     }
                  } else {}
               } else {}
            }
                     }
                  } else {}
               } else {}
            }
         }

         File fileDirectory4 = new File(sdCard.getAbsolutePath() + newPath,
                                 fileDirectory3.getName());
         FileOutputStream fileOutputStream;
         try {
            fileOutputStream = new FileOutputStream(fileDirectory4);
            fileOutputStream.write(filesByte);
            fileOutputStream.close();
         } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
         }
      } else if (requestStr.contains("copy file in remote and paste in remote")) {
         int fileSize0 = 0;
         int fileSize1 = 0;
         fileSize0 = buffer[500];
         fileSize1 = buffer[501];
         String pathCopy = "";
         pathCopy = pathCopy + new String(buffer, 502, fileSize0 + 100*fileSize1);

         File sdCard = Environment.getExternalStorageDirectory();
         File directory = new File(sdCard.getAbsolutePath());
         File[] folder = directory.listFiles();

         String newPath = "";
         for (int a = 0; a < folder.length; a++) {
            if (pathCopy.contains(folder[a].getName())) {
               newPath = "/" + folder[a].getName();
```

```
    }
    else {
       File[] subFolder = folder[a].listFiles();
       if (subFolder != null) {
          if(subFolder.length > 0) {
             for (int b = 0; b < subFolder.length; b++) {

    if (pathCopy.contains(subFolder[b].getName()) && subFolder[b].isFile())  {
       newPath = "/" + folder[a].getName() + "/" + subFolder[b].getName();
    }
    else {
       File[] subSubFolder = subFolder[b].listFiles();
       if (subSubFolder != null) {
          if(subSubFolder.length > 0) {
             for (int c = 0; c < subSubFolder.length; c++) {

   if (pathCopy.contains(subSubFolder[c].getName()) && subSubFolder[c].isFile()) {
       newPath = "/" + folder[a].getName() + "/" + subFolder[b].getName() + "/" +
                 subSubFolder[c].getName();
    }
    else {
       File[] subSubFolder1 = subSubFolder[c].listFiles();
       if (subSubFolder1 != null) {
          if(subSubFolder1.length > 0) {
             for (int d = 0; d < subSubFolder1.length; d++) {

    if (pathCopy.contains(subSubFolder1[d].getName()) &&
                              subSubFolder1[d].isFile()) {
       newPath = "/" + folder[a].getName() + "/" + subFolder[b].getName() + "/" +
                 subSubFolder[c].getName() + "/" + subSubFolder1[d].getName();
    }
    else {
       File[] subSubFolder2 = subSubFolder1[d].listFiles();
       if (subSubFolder2 != null) {
          if(subSubFolder2.length > 0) {
             for (int e = 0; e < subSubFolder2.length; e++) {

    if (pathCopy.contains(subSubFolder2[e].getName()) &&
                               subSubFolder2[e].isFile()) {
       newPath = "/" + folder[a].getName() + "/" + subFolder[b].getName() + "/" +
                 subSubFolder[c].getName() + "/" + subSubFolder1[d].getName() + "/"
                 + subSubFolder2[e].getName();
    } else {}
             }
             } else {}
          } else {}
    }
             }
             } else {}
          } else {}
    }
             }
             } else {}
          } else {}
    }
             }
             } else {}
          } else {}
    }
}
```

```
File fileDirectory3 = new File(sdCard.getAbsolutePath() + newPath);
byte[] filesByte = new byte[(int) fileDirectory3.length()];
FileInputStream fileInputStream;
try {
   fileInputStream = new FileInputStream(fileDirectory3);
   fileInputStream.read(filesByte);
   fileInputStream.close();
} catch (IOException e) {
   // TODO Auto-generated catch block
   e.printStackTrace();
}

int fileSize2 = 0;
int fileSize3 = 0;
fileSize2 = buffer[502 + fileSize0 + 100*fileSize1];
fileSize3 = buffer[503 + fileSize0 + 100*fileSize1];
String pathPaste = "";
pathPaste = pathPaste + new String(buffer, 504 + fileSize0 + 100*fileSize1,
            fileSize2 + 100*fileSize3);

newPath = "";
for (int a = 0; a < folder.length; a++) {
   if (pathPaste.contains(folder[a].getName())) {
      newPath = "/" + folder[a].getName();
   }
   else {
      File[] subFolder = folder[a].listFiles();
      if (subFolder != null) {
         if(subFolder.length > 0) {
            for (int b = 0; b < subFolder.length; b++) {

   if (pathPaste.contains(subFolder[b].getName()))  {
      newPath = "/" + folder[a].getName() + "/" + subFolder[b].getName();
   }
   else {
      File[] subSubFolder = subFolder[b].listFiles();
      if (subSubFolder != null) {
         if(subSubFolder.length > 0) {
            for (int c = 0; c < subSubFolder.length; c++) {

   if (pathPaste.contains(subSubFolder[c].getName())) {
      newPath = "/" + folder[a].getName() + "/" + subFolder[b].getName() + "/" +
                subSubFolder[c].getName();
   }
   else {
      File[] subSubFolder1 = subSubFolder[c].listFiles();
      if (subSubFolder1 != null) {
         if(subSubFolder1.length > 0) {
            for (int d = 0; d < subSubFolder1.length; d++) {

   if (pathPaste.contains(subSubFolder1[d].getName())) {
      newPath = "/" + folder[a].getName() + "/" + subFolder[b].getName() + "/" +
                subSubFolder[c].getName() + "/" + subSubFolder1[d].getName();
   }
   else {
      File[] subSubFolder2 = subSubFolder1[d].listFiles();
      if (subSubFolder2 != null) {
         if(subSubFolder2.length > 0) {
            for (int e = 0; e < subSubFolder2.length; e++) {
```

```
            if (pathPaste.contains(subSubFolder2[e].getName())) {
               newPath = "/" + folder[a].getName() + "/" + subFolder[b].getName() + "/" +
                        subSubFolder[c].getName() + "/" + subSubFolder1[d].getName() + "/"
                        + subSubFolder2[e].getName();
            } else {}
                     }
                  } else {}
               } else {}
            }
                     }
                  } else {}
               } else {}
            }
                     }
                  } else {}
               } else {}
            }
                     }
                  } else {}
               } else {}
            }
         }

         File fileDirectory4 = new File(sdCard.getAbsolutePath() + newPath,
                              fileDirectory3.getName());
         FileOutputStream fileOutputStream;
         try {
            fileOutputStream = new FileOutputStream(fileDirectory4);
            fileOutputStream.write(filesByte);
            fileOutputStream.close();
         } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
         }
      } else if (requestStr.contains("delete file to remote")) {
         int fileSize0 = 0;
         int fileSize1 = 0;
         fileSize0 = buffer[500];
         fileSize1 = buffer[501];
         String path = "";
         path = path + new String(buffer, 502, fileSize0 + 100*fileSize1);

         File sdCard = Environment.getExternalStorageDirectory();
         File directory = new File(sdCard.getAbsolutePath());
         File[] folder = directory.listFiles();

         String newPath = "";
         for (int a = 0; a < folder.length; a++) {
            if (path.contains(folder[a].getName())) {
               newPath = "/" + folder[a].getName();
            }
            else {
               File[] subFolder = folder[a].listFiles();
               if (subFolder != null) {
                  if(subFolder.length > 0) {
                     for (int b = 0; b < subFolder.length; b++) {

            if (path.contains(subFolder[b].getName()) && subFolder[b].isFile())  {
               newPath = "/" + folder[a].getName() + "/" + subFolder[b].getName();
            }
```

```
                else {
                    File[] subSubFolder = subFolder[b].listFiles();
                    if (subSubFolder != null) {
                        if(subSubFolder.length > 0) {
                            for (int c = 0; c < subSubFolder.length; c++) {

                if (path.contains(subSubFolder[c].getName()) && subSubFolder[c].isFile()) {
                    newPath = "/" + folder[a].getName() + "/" + subFolder[b].getName() + "/" +
                                subSubFolder[c].getName();
                }
                else {
                    File[] subSubFolder1 = subSubFolder[c].listFiles();
                    if (subSubFolder1 != null) {
                        if(subSubFolder1.length > 0) {
                            for (int d = 0; d < subSubFolder1.length; d++) {

                if (path.contains(subSubFolder1[d].getName()) && subSubFolder1[d].isFile()) {
                    newPath = "/" + folder[a].getName() + "/" + subFolder[b].getName() + "/" +
                                subSubFolder[c].getName() + "/" + subSubFolder1[d].getName();
                }
                else {
                    File[] subSubFolder2 = subSubFolder1[d].listFiles();
                    if (subSubFolder2 != null) {
                        if(subSubFolder2.length > 0) {
                            for (int e = 0; e < subSubFolder2.length; e++) {

                if (path.contains(subSubFolder2[e].getName()) && subSubFolder2[e].isFile()) {
                    newPath = "/" + folder[a].getName() + "/" + subFolder[b].getName() + "/" +
                                subSubFolder[c].getName() + "/" + subSubFolder1[d].getName() + "/"
                                + subSubFolder2[e].getName();
                } else {}
                            }
                            } else {}
                        } else {}
                }
                            }
                            } else {}
                        } else {}
                }
                            }
                            } else {}
                        } else {}
                }
                            }
                            } else {}
                        } else {}
                    }
                }
                File fileDirectory2 = new File(sdCard.getAbsolutePath() + newPath);
                fileDirectory2.delete();
            } else {}
                is.close();
                serverSocket.close();
            }catch (IOException e) {}
                is.close();
                serverSocket.close();
        }catch (IOException e) {
                serverSocket.close();
        }
    } catch (IOException e) {
```

```
            btserver.close();
        }
    } catch (IOException e) {}
return requestStr;
}

@Override
protected void onPostExecute(String requestCode) {
    if (requestCode.contains("request for screenshare")) {
        new DoCountingClientTask().execute();
        Builder builder = new Builder(MainActivity.this, AlertDialog.THEME_HOLO_DARK);
        builder.setTitle("Request for ScreenShare");
        builder.setCancelable(true);
        builder.setMessage("Do you want to be remote device ?");
        builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int which) {
                final SharingThread sharingThread = new SharingThread(BDaddress);
                sharingThread.run();
                dialog.cancel();
            }
        });
        builder.setNegativeButton("NO", new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int which) {
                dialog.cancel();
            }
        });
        builder.show();
    }

    else if (requestCode.contains("sending screen data")) {
        new DoCountingClientTask().execute();
        startActivity(new Intent("com.example.example.nextpage"));
    }

    else if (requestCode.contains("request for file transfer")) {
        new DoCountingClientTask().execute();
        final FileTransferThread fileTransferThread = new
                        FileTransferThread(serverSocket.getRemoteDevice(), pathDirectory);
        fileTransferThread.run();
    }

    else if (requestCode.contains("here is the file")) {
        new DoCountingClientTask().execute();
        Toast.makeText(getBaseContext(), "Received", Toast.LENGTH_SHORT).show();

        Uri uri = Uri.fromFile(file[num-1]);
        Intent intent = new Intent(Intent.ACTION_VIEW);
        if (file[num-1].toString().contains(".docx")) {
            // Word document
            intent.setDataAndType(uri, "application/msword");
        } else if(file[num-1].toString().contains(".pdf")) {
            // PDF file
            intent.setDataAndType(uri, "application/pdf");
        } else if(file[num-1].toString().contains(".pptx")) {
            // Powerpoint file
            intent.setDataAndType(uri, "application/vnd.ms-powerpoint");
        } else if(file[num-1].toString().contains(".mp3") || file[num-1].toString().contains(".ogg")) {
            // WAV audio file
            intent.setDataAndType(uri, "audio/x-wav");
        } else if(file[num-1].toString().contains(".mp4")) {
```

```
        // Video files
        intent.setDataAndType(uri, "video/*");
    } else if(file[num-1].toString().contains(".jpg")) {
        // JPG file
        intent.setDataAndType(uri, "image/jpeg");
    } else if(file[num-1].toString().contains(".xlsx")) {
        // Excel file
        intent.setDataAndType(uri, "application/vnd.ms-excel");
    } else if(file[num-1].toString().contains(".txt")) {
        // Text file
        intent.setDataAndType(uri, "text/plain");
    } else {}
    startActivity(intent);
}

else if (requestCode.contains("paste file to remote") ||
            requestCode.contains("paste file to empty folder in remote")) {
    new DoCountingClientTask().execute();
    Toast.makeText(getBaseContext(), "Paste to remote success",
                            Toast.LENGTH_SHORT).show();
    Builder builder = new Builder(MainActivity.this, AlertDialog.THEME_HOLO_DARK);
    builder.setTitle("Open the pasted file ?");
    builder.setCancelable(true);
    builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int which) {
            dialog.cancel();

            Uri uri = Uri.fromFile(fileDirectory1);
            Intent intent = new Intent(Intent.ACTION_VIEW);
            if (fileDirectory1.toString().contains(".docx")) {
                // Word document
                intent.setDataAndType(uri, "application/msword");
            } else if(fileDirectory1.toString().contains(".pdf")) {
                // PDF file
                intent.setDataAndType(uri, "application/pdf");
            } else if(fileDirectory1.toString().contains(".pptx")) {
                // Powerpoint file
                intent.setDataAndType(uri, "application/vnd.ms-powerpoint");
            } else if(fileDirectory1.toString().contains(".mp3") ||
                        fileDirectory1.toString().contains(".ogg")) {
                // WAV audio file
                intent.setDataAndType(uri, "audio/x-wav");
            } else if(fileDirectory1.toString().contains(".mp4")) {
                // Video files
                intent.setDataAndType(uri, "video/*");
            } else if(fileDirectory1.toString().contains(".jpg")) {
                // JPG file
                intent.setDataAndType(uri, "image/jpeg");
            } else if(fileDirectory1.toString().contains(".xlsx")) {
                // Excel file
                intent.setDataAndType(uri, "application/vnd.ms-excel");
            } else if(fileDirectory1.toString().contains(".txt")) {
                // Text file
                intent.setDataAndType(uri, "text/plain");
            } else {}
            startActivity(intent);
        }
    });
    builder.setNegativeButton("NO", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int which) {
```

```
            dialog.cancel();
        }
    });
    builder.show();
}

else if (requestCode.contains("delete file to remote")) {
    new DoCountingClientTask().execute();
    Toast.makeText(getBaseContext(), "Delete file success in remote",
                    Toast.LENGTH_SHORT).show();
}

else if (requestCode.contains("copy file from remote to local")) {
    new DoCountingClientTask().execute();
    Toast.makeText(getBaseContext(), "Request for remote file",
                    Toast.LENGTH_SHORT).show();
    final FileCopyThread fileCopyThread = new
                    FileCopyThread(serverSocket.getRemoteDevice(), pathDirectory);
    fileCopyThread.run();
}

else if (requestCode.contains("copied file is ready")) {
    new DoCountingClientTask().execute();
    Toast.makeText(getBaseContext(), "Done pasting to local",
                    Toast.LENGTH_SHORT).show();
    done = 1;
    }

else if (requestCode.contains("copy and paste file in remote") ||
          requestCode.contains("copy file in remote and paste in remote")) {
    new DoCountingClientTask().execute();
    Toast.makeText(getBaseContext(), "Copy and paste success in remote",
                    Toast.LENGTH_SHORT).show();
}

else if (requestCode.contains("file saved back")) {
    new DoCountingClientTask().execute();
    Toast.makeText(getBaseContext(), "Saved back", Toast.LENGTH_SHORT).show();
}

else if (requestCode.contains("upload file to remote")) {
    new DoCountingClientTask().execute();
    Toast.makeText(getBaseContext(), "file uploaded from local",
                    Toast.LENGTH_SHORT).show();
    Builder builder = new Builder(MainActivity.this, AlertDialog.THEME_HOLO_DARK);
    builder.setTitle("Open the uploaded file ?");
    builder.setCancelable(true);
    builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int which) {
            dialog.cancel();

            Uri uri = Uri.fromFile(filepath);
            Intent intent = new Intent(Intent.ACTION_VIEW);
            if (filepath.toString().contains(".docx")) {
                // Word document
                intent.setDataAndType(uri, "application/msword");
            } else if(filepath.toString().contains(".pdf")) {
                // PDF file
                intent.setDataAndType(uri, "application/pdf");
            } else if(filepath.toString().contains(".xlsx")) {
```

```
                    // Excel file
                    intent.setDataAndType(uri, "application/vnd.ms-excel");
                } else if(filepath.toString().contains(".pptx")) {
                    // Powerpoint file
                    intent.setDataAndType(uri, "application/vnd.ms-powerpoint");
                } else if(filepath.toString().contains(".mp3") || filepath.toString().contains(".ogg")) {
                    // WAV audio file
                    intent.setDataAndType(uri, "audio/x-wav");
                } else if(filepath.toString().contains(".mp4")) {
                    // Video files
                    intent.setDataAndType(uri, "video/*");
                } else if(filepath.toString().contains(".jpg")) {
                    // JPG file
                    intent.setDataAndType(uri, "image/jpeg");
                } else if(filepath.toString().contains(".txt")) {
                    // Text file
                    intent.setDataAndType(uri, "text/plain");
                } else {}
                startActivity(intent);
            }
        });
        builder.setNegativeButton("NO", new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int which) {
                dialog.cancel();
            }
        });
        builder.show();
    }

    else if (requestCode == "") {}
    else {
        Builder builder = new Builder(MainActivity.this, AlertDialog.THEME_HOLO_DARK);
        builder.setTitle("ERROR");
        builder.setCancelable(true);
        builder.setMessage("Please relaunch the application");
        builder.setNeutralButton("OK", new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int which) {
                new DoCountingClientTask().execute();
                dialog.cancel();
            }
        });
        builder.show();
    }
  }
  @Override
  protected void onPreExecute() {}

  @Override
  protected void onProgressUpdate(Void... values) {}
}
private class FileTransferThread extends Thread {
  private BluetoothSocket mmSocket = null;
  private BluetoothDevice mmDevice = null;
  private OutputStream outStream = null;

  public FileTransferThread(BluetoothDevice device, String path) {
    // Use a temporary object that is later assigned to mmSocket,
    // because mmSocket is final
    BluetoothSocket tmp = null;
    mmDevice = device;
```

```
// Get a BluetoothSocket to connect with the given BluetoothDevice
try {
// MY_UUID is the app's UUID string, also used by the server code
   tmp = mmDevice.createInsecureRfcommSocketToServiceRecord(MY_UUID);
   mmSocket = tmp;

   // Cancel discovery because it will slow down the connection
   mBluetoothAdapter.cancelDiscovery();

   try {
      // Connect the device through the socket. This will block
      // until it succeeds or throws an exception
      mmSocket.connect();

      try {
         outStream = mmSocket.getOutputStream();

         try {
            outStream.write("here is the file".getBytes());
            byte[] redundantRequestByte = new byte[500-"here is the file".length()];
            outStream.write(redundantRequestByte);

            String newPath = "";

            File sdCard = Environment.getExternalStorageDirectory();
            File directory = new File(sdCard.getAbsolutePath());
            File[] folder = directory.listFiles();

            for (int i = 0; i < folder.length; i++) {
               if (path.contains(folder[i].getName()) && folder[i].isFile()) {
                  newPath = "/" + folder[i].getName();
               }
               else {
                  File[] subFolder = folder[i].listFiles();
                  if (subFolder != null) {
                     if(subFolder.length > 0) {
                        for (int j = 0; j < subFolder.length; j++) {

                  if (path.contains(subFolder[j].getName()) && subFolder[j].isFile())  {
                     newPath = "/" + folder[i].getName() + "/" + subFolder[j].getName();
                  }
                  else {
                     File[] subSubFolder = subFolder[j].listFiles();
                     if (subSubFolder != null) {
                        if(subSubFolder.length > 0) {
                           for (int k = 0; k < subSubFolder.length; k++) {

                  if (path.contains(subSubFolder[k].getName()) && subSubFolder[k].isFile()) {
                     newPath = "/" + folder[i].getName() + "/" + subFolder[j].getName() + "/" +
                           subSubFolder[k].getName();
                  }
                  else {
                     File[] subSubFolder1 = subSubFolder[k].listFiles();
                     if (subSubFolder1 != null) {
                        if(subSubFolder1.length > 0) {
                           for (int l = 0; l < subSubFolder1.length; l++) {

                 if (path.contains(subSubFolder1[l].getName()) && subSubFolder1[l].isFile()) {
                     newPath = "/" + folder[i].getName() + "/" + subFolder[j].getName() + "/" +
                                 subSubFolder[k].getName() + "/" + subSubFolder1[l].getName();
```

```
        }
        else {
            File[] subSubFolder2 = subSubFolder1[l].listFiles();
                if (subSubFolder2 != null) {
                    if(subSubFolder2.length > 0) {
                        for (int m = 0; m < subSubFolder2.length; m++) {

    if (path.contains(subSubFolder2[m].getName()) && subSubFolder2[m].isFile()) {
          newPath = "/" + folder[i].getName() + "/" + subFolder[j].getName() + "/" +
                        subSubFolder[k].getName() + "/" + subSubFolder1[l].getName() +
                        "/" + subSubFolder2[m].getName();
       } else {}
                    }
                } else {}
            } else {}
       }
                }
            } else {}
          } else {}
       }
                }
            } else {}
           } else {}

       }
                }
           } else {}
       } else {}
     }
}

FileInputStream fileInputStream = null;
File sdCard1 = Environment.getExternalStorageDirectory();
File fileName = new File(sdCard1.getAbsolutePath() + newPath);
byte[] fileByte = new byte[(int) fileName.length()];

outStream.write(fileName.getName().length());
outStream.write(fileName.getName().getBytes());
byte[] redundantfileNameByte = new byte[499-fileName.getName().length()];
outStream.write(redundantfileNameByte);

try {
   fileInputStream = new FileInputStream(fileName);
   fileInputStream.read(fileByte);
   fileInputStream.close();
   Toast.makeText(getBaseContext(), "read success",
                        Toast.LENGTH_SHORT).show();
} catch(Exception e) {
   Toast.makeText(getBaseContext(), "fail read", Toast.LENGTH_SHORT).show();
}

int size = (int) fileName.length();
int i = 0;

while (size > 100) {
   size = size - 100;
   i++;
}

int j = 0;
```

```
                while (i > 100) {
                    i = i - 100;
                    j++;
                }

                int k = 0;
                while (j > 100) {
                    j = j - 100;
                    k++;
                }

                outStream.write(size);
                outStream.write(i);
                outStream.write(j);
                outStream.write(k);

                outStream.write(fileByte);
                Toast.makeText(getBaseContext(), "file transferred",
                                Toast.LENGTH_SHORT).show();
                try {
                    outStream.flush();
                    outStream.close();
                    mmSocket.close();
                    tmp.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
              } catch (IOException e) {
                 e.printStackTrace();
              }
            } catch (IOException e) {
               e.printStackTrace();
            }
          } catch (IOException e) {
             e.printStackTrace();
          }
        }  catch (IOException connectException) {}
    }
}

private class FileCopyThread extends Thread {
    private BluetoothSocket mmSocket = null;
    private BluetoothDevice mmDevice = null;
    private OutputStream outStream = null;

    public FileCopyThread(BluetoothDevice device, String path) {
        // Use a temporary object that is later assigned to mmSocket,
        // because mmSocket is final
        BluetoothSocket tmp = null;
        mmDevice = device;
        // Get a BluetoothSocket to connect with the given BluetoothDevice
        try {
        // MY_UUID is the app's UUID string, also used by the server code
            tmp = mmDevice.createInsecureRfcommSocketToServiceRecord(MY_UUID);
            mmSocket = tmp;

            // Cancel discovery because it will slow down the connection
            mBluetoothAdapter.cancelDiscovery();

            try {
```

```
// Connect the device through the socket. This will block
// until it succeeds or throws an exception
mmSocket.connect();

try {
   outStream = mmSocket.getOutputStream();

   try {
      outStream.write("copied file is ready".getBytes());
      byte[] redundantRequestByte = new byte[500-"copied file is ready".length()];
      outStream.write(redundantRequestByte);

      String newPath = "";

      File sdCard = Environment.getExternalStorageDirectory();
      File directory = new File(sdCard.getAbsolutePath());
      File[] folder = directory.listFiles();

      for (int i = 0; i < folder.length; i++) {
         if (path.contains(folder[i].getName()) && folder[i].isFile()) {
            newPath = "/" + folder[i].getName();
         }
         else {
            File[] subFolder = folder[i].listFiles();
            if (subFolder != null) {
               if(subFolder.length > 0) {
                  for (int j = 0; j < subFolder.length; j++) {

            if (path.contains(subFolder[j].getName()) && subFolder[j].isFile())  {
               newPath = "/" + folder[i].getName() + "/" + subFolder[j].getName();
            }
            else {
               File[] subSubFolder = subFolder[j].listFiles();
               if (subSubFolder != null) {
                  if(subSubFolder.length > 0) {
                     for (int k = 0; k < subSubFolder.length; k++) {

            if (path.contains(subSubFolder[k].getName()) && subSubFolder[k].isFile()) {
               newPath = "/" + folder[i].getName() + "/" + subFolder[j].getName() + "/" +
                     subSubFolder[k].getName();
            }
            else {
               File[] subSubFolder1 = subSubFolder[k].listFiles();
               if (subSubFolder1 != null) {
                  if(subSubFolder1.length > 0) {
                     for (int l = 0; l < subSubFolder1.length; l++) {

           if (path.contains(subSubFolder1[l].getName()) && subSubFolder1[l].isFile()) {
               newPath = "/" + folder[i].getName() + "/" + subFolder[j].getName() + "/" +
                           subSubFolder[k].getName() + "/" + subSubFolder1[l].getName();
            }
            else {
               File[] subSubFolder2 = subSubFolder1[l].listFiles();
                  if (subSubFolder2 != null) {
                     if(subSubFolder2.length > 0) {
                        for (int m = 0; m < subSubFolder2.length; m++) {

          if (path.contains(subSubFolder2[m].getName()) && subSubFolder2[m].isFile()) {
                newPath = "/" + folder[i].getName() + "/" + subFolder[j].getName() + "/" +
                           subSubFolder[k].getName() + "/" + subSubFolder1[l].getName() +
```

```
                "/" + subSubFolder2[m].getName();
        } else {}
                    }
                } else {}
            } else {}
        }
                }
            } else {}
          } else {}
        }
                }
            } else {}
         } else {}

        }
                }
            } else {}
        } else {}
    }
}

FileInputStream fileInputStream = null;
File sdCard1 = Environment.getExternalStorageDirectory();
File fileName = new File(sdCard1.getAbsolutePath() + newPath);
byte[] fileByte = new byte[(int) fileName.length()];

Toast.makeText(getBaseContext(), fileName.getName(),
                        Toast.LENGTH_SHORT).show();
outStream.write(fileName.getName().length());
outStream.write(fileName.getName().getBytes());
byte[] redundantfileNameByte = new byte[499-fileName.getName().length()];
outStream.write(redundantfileNameByte);

try {
   fileInputStream = new FileInputStream(fileName);
   fileInputStream.read(fileByte);
   fileInputStream.close();
} catch(Exception e) {}

int size = (int) fileName.length();
int i = 0;

while (size > 100) {
   size = size - 100;
   i++;
}

int j = 0;
while (i > 100) {
   i = i - 100;
   j++;
}

int k = 0;
while (j > 100) {
   j = j - 100;
   k++;
}

outStream.write(size);
```

```
                    outStream.write(i);
                    outStream.write(j);
                    outStream.write(k);

                    outStream.write(fileByte);
                    Toast.makeText(getBaseContext(), "file transferred",
                                   Toast.LENGTH_SHORT).show();
                    try {
                       outStream.flush();
                       outStream.close();
                       mmSocket.close();
                       tmp.close();
                    } catch (IOException e) {
                       e.printStackTrace();
                    }
                 } catch (IOException e) {
                    e.printStackTrace();
                 }
              } catch (IOException e) {
                 e.printStackTrace();
              }
           } catch (IOException e) {
              e.printStackTrace();
           }
        }  catch (IOException connectException) {}
     }
  }

  private class SharingThread extends Thread {
     private BluetoothSocket mmSocket = null;
     private BluetoothDevice mmDevice = null;
     private OutputStream outStream = null;

     public SharingThread (String address) {
        // Use a temporary object that is later assigned to mmSocket,
        // because mmSocket is final
        BluetoothSocket tmp = null;
        mmDevice = mBluetoothAdapter.getRemoteDevice(address);
        // Get a BluetoothSocket to connect with the given BluetoothDevice
        try {
        // MY_UUID is the app's UUID string, also used by the server code
           tmp = mmDevice.createInsecureRfcommSocketToServiceRecord(MY_UUID);
        } catch (IOException e) {
           Builder builder = new Builder(MainActivity.this, AlertDialog.THEME_HOLO_DARK);
           builder.setTitle("Unable to create BluetoothSocket");
           builder.setCancelable(true);
           builder.setMessage("Please Relaunch the application");
           builder.setNeutralButton("OK", new DialogInterface.OnClickListener() {
              public void onClick(DialogInterface dialog, int which) {
                 dialog.cancel();
              }
           });
           builder.show();

           e.printStackTrace();
        }

        mmSocket = tmp;
        mBluetoothAdapter.cancelDiscovery();
```

```
try {
   mmSocket.connect();

   try {
      outStream = mmSocket.getOutputStream();

      try {
         outStream.write("sending screen data".getBytes());
         byte[] redundantRequestByte = new byte[500-"sending screen data".length()];
         outStream.write(redundantRequestByte);

         final Intent mainIntent = new Intent(Intent.ACTION_MAIN, null);
         mainIntent.addCategory(Intent.CATEGORY_LAUNCHER);
         List<ResolveInfo> pacsList = pm.queryIntentActivities(mainIntent, 0);
         outStream.write(pacsList.size());

         File sdCard = Environment.getExternalStorageDirectory();
         File directory = new File(sdCard.getAbsolutePath());
         File[] folder = directory.listFiles();
         outStream.write(folder.length);

         for(int i=0 ; i < pacsList.size() ; i++) {

            Bitmap bm = ((BitmapDrawable)packs[i].icon).getBitmap();
            ByteArrayOutputStream baos = new ByteArrayOutputStream();
            bm.compress(Bitmap.CompressFormat.PNG, 100, baos); //bm is the bitmap object
            byte[] imageByte = baos.toByteArray();
            outStream.write(imageByte);

            byte[] redundantImageByte = new byte[19950-imageByte.length];
            outStream.write(redundantImageByte);

            byte[] labelByte = packs[i].label.getBytes();
            outStream.write(labelByte);

            byte[] redundantLabelByte = new byte[50-labelByte.length];
            outStream.write(redundantLabelByte);
         }

         for (int i = 0; i < folder.length; i++) {
            outStream.write(folder[i].getName().getBytes());
            byte[] redundantFolderNameByte = new byte[250-
                                                folder[i].getName().getBytes().length];
            outStream.write(redundantFolderNameByte);
         }

         for (int i = 0; i < folder.length; i++) {
            File[] subFolder = folder[i].listFiles();
            if (subFolder != null) {
               if(subFolder.length > 0) {
                  int num = subFolder.length;
                  int subFolderBytes = 1;
                  while (num > 100) {
                     num = num -100;
                     subFolderBytes = subFolderBytes + 1;
                  }
                  outStream.write(subFolderBytes);

                  num = subFolder.length;
                  while (num > 100) {
```

```
                num = num - 100;
                outStream.write(100);
            }
            outStream.write(num);
        } else {
            outStream.write(1);
            outStream.write(0);
        }
    } else {
        outStream.write(1);
        outStream.write(-1);
    }
}

for (int i = 0; i < folder.length; i++) {
    File[] subFolder = folder[i].listFiles();
    if (subFolder != null) {
        if(subFolder.length > 0) {
            for (int j = 0; j < subFolder.length; j++) {
                outStream.write(subFolder[j].getName().getBytes());
                byte[] redundantSubFolderNameByte = new byte[250-
                        subFolder[j].getName().getBytes().length];
                outStream.write(redundantSubFolderNameByte);
            }
        } else {}
    } else {}
}

for (int i = 0; i < folder.length; i++) {
    File[] subFolder = folder[i].listFiles();
    if (subFolder != null) {
        if(subFolder.length > 0) {
            for (int j = 0; j < subFolder.length; j++) {
                File[] subSubFolder = subFolder[j].listFiles();
                if (subSubFolder != null) {
                    if(subSubFolder.length > 0) {
                        int num = subSubFolder.length;
                        int subSubFolderBytes = 1;
                        while (num > 100) {
                            num = num - 100;
                            subSubFolderBytes = subSubFolderBytes + 1;
                        }
                        outStream.write(subSubFolderBytes);
                        num = subSubFolder.length;
                        while (num > 100) {
                            num = num - 100;
                            outStream.write(100);
                        }
                        outStream.write(num);
                    } else {
                        outStream.write(1);
                        outStream.write(0);
                    }
                } else {
                    outStream.write(1);
                    outStream.write(-1);
                }
            }
        } else {}
    } else {}
```

```
}

for (int i = 0; i < folder.length; i++) {
   File[] subFolder = folder[i].listFiles();
      if (subFolder != null) {
         if(subFolder.length > 0) {
            for (int j = 0; j < subFolder.length; j++) {
               File[] subSubFolder = subFolder[j].listFiles();
               if (subSubFolder != null) {
                  if(subSubFolder.length > 0) {
                     for (int k = 0; k < subSubFolder.length; k++) {
                        outStream.write(subSubFolder[k].getName().getBytes());
                        byte[] redundantSubSubFolderNameByte
                           = new byte[250-subSubFolder[k].getName().getBytes().length];
                        outStream.write(redundantSubSubFolderNameByte);
                     }
                  } else {}
               } else {}
            }
         } else {}
      } else {}
}

for (int i = 0; i < folder.length; i++) {
   File[] subFolder = folder[i].listFiles();
   if (subFolder != null) {
      if(subFolder.length > 0) {
         for (int j = 0; j < subFolder.length; j++) {
            File[] subSubFolder = subFolder[j].listFiles();
            if (subSubFolder != null) {
               if(subSubFolder.length > 0) {
                  for (int k = 0; k < subSubFolder.length; k++) {
                     File[] subSubFolder1 = subSubFolder[k].listFiles();
                     if (subSubFolder1 != null) {
                        if(subSubFolder1.length > 0) {
                           int num = subSubFolder1.length;
                           int subSubFolder1Bytes = 1;
                           while (num > 100) {
                              num = num -100;
                              subSubFolder1Bytes = subSubFolder1Bytes + 1;
                           }
                           outStream.write(subSubFolder1Bytes);

                           num = subSubFolder1.length;
                           while (num > 100) {
                              num = num - 100;
                              outStream.write(100);
                           }
                           outStream.write(num);
                        } else {
                           outStream.write(1);
                           outStream.write(0);
                        }
                     } else {
                        outStream.write(1);
                        outStream.write(-1);
                     }
                  }
               } else {}
            } else {}
```

```
            }
        } else {}
    } else {}
}

for (int i = 0; i < folder.length; i++) {
    File[] subFolder = folder[i].listFiles();
    if (subFolder != null) {
        if(subFolder.length > 0) {
            for (int j = 0; j < subFolder.length; j++) {
                File[] subSubFolder = subFolder[j].listFiles();
                if (subSubFolder != null) {
                    if(subSubFolder.length > 0) {
                        for (int k = 0; k < subSubFolder.length; k++) {
                            File[] subSubFolder1 = subSubFolder[k].listFiles();
                            if (subSubFolder1 != null) {
                                if(subSubFolder1.length > 0) {
                                    for (int l = 0; l < subSubFolder1.length; l++) {
                                        outStream.write(subSubFolder1[l].getName().getBytes());
                                        byte[] redundantSubSubFolder1NameByte
                                          = new byte[250-
                                            subSubFolder1[l].getName().getBytes().length];
                                        outStream.write(redundantSubSubFolder1NameByte);
                                    }
                                } else {}
                            } else {}
                        }
                    } else {}
                } else {}
            }
        } else {}
    } else {}
}

for (int i = 0; i < folder.length; i++) {
    File[] subFolder = folder[i].listFiles();
    if (subFolder != null) {
        if(subFolder.length > 0) {
            for (int j = 0; j < subFolder.length; j++) {
                File[] subSubFolder = subFolder[j].listFiles();
                if (subSubFolder != null) {
                    if(subSubFolder.length > 0) {
                        for (int k = 0; k < subSubFolder.length; k++) {
                            File[] subSubFolder1 = subSubFolder[k].listFiles();
                            if (subSubFolder1 != null) {
                                if(subSubFolder1.length > 0) {
                                    for (int l = 0; l < subSubFolder1.length; l++) {
                                        File[] subSubFolder2 = subSubFolder1[l].listFiles();
                                        if (subSubFolder2 != null) {
                                            if(subSubFolder2.length > 0) {
                                                int num = subSubFolder2.length;
                                                int subSubFolder2Bytes = 1;
                                                while (num > 100) {
                                                    num = num -100;
                                                    subSubFolder2Bytes = subSubFolder2Bytes + 1;
                                                }
                                                outStream.write(subSubFolder2Bytes);

                                                num = subSubFolder2.length;
                                                while (num > 100) {
```

```
                                    num = num - 100;
                                    outStream.write(100);
                                }
                                outStream.write(num);
                            } else {
                                outStream.write(1);
                                outStream.write(0);
                            }
                        } else {
                            outStream.write(1);
                            outStream.write(-1);
                        }
                    }
                } else {}
            } else {}
        }
    } else {}
} else {}
}

for (int i = 0; i < folder.length; i++) {
    File[] subFolder = folder[i].listFiles();
    if (subFolder != null) {
        if(subFolder.length > 0) {
            for (int j = 0; j < subFolder.length; j++) {
                File[] subSubFolder = subFolder[j].listFiles();
                if (subSubFolder != null) {
                    if(subSubFolder.length > 0) {
                        for (int k = 0; k < subSubFolder.length; k++) {
                            File[] subSubFolder1 = subSubFolder[k].listFiles();
                            if (subSubFolder1 != null) {
                                if(subSubFolder1.length > 0) {
                                    for (int l = 0; l < subSubFolder1.length; l++) {
                                        File[] subSubFolder2 = subSubFolder1[l].listFiles();
                                        if (subSubFolder2 != null) {
                                            if(subSubFolder2.length > 0) {

                                for (int m = 0; m < subSubFolder2.length; m++) {
                                    outStream.write(subSubFolder2[m].getName().getBytes());
                                    byte[] redundantSubSubFolder2NameByte
                                                = new byte[250-
                                                 subSubFolder2[m].getName().getBytes().length];
                                    outStream.write(redundantSubSubFolder2NameByte);
                                }
                                            } else {}
                                        } else {}
                                    }
                                } else {}
                            }
                        } else {}
                    } else {}
                }
            } else {}
        } else {}
    } else {}
}
```

```
String[] stringArray, stringArray1, stringArray2, stringArray3;
try {
  //internal storage
  FileInputStream fIn = openFileInput("confimusic.txt");
  InputStreamReader isr = new InputStreamReader(fIn);
  int value1 = 0;

  char[] inputBuffer = new char[1];
  String s = "";

  int value = 0;
  value = isr.read();
  stringArray1 = new String[value];

  int charRead;
  while((charRead = isr.read(inputBuffer)) > 0) {
     //convert the chars to a string
     String readString = String.copyValueOf(inputBuffer, 0, charRead);
     if (readString.contains("\n")) {
        stringArray1[value1] = s;
        s = "";
        value1++;
     } else {
        s += readString;
     }
     inputBuffer = new char[1];
  }
  isr.close();
  fIn.close();

  fIn = openFileInput("confivideo.txt");
  isr = new InputStreamReader(fIn);

  inputBuffer = new char[1];
  s = "";

  value = 0;
  value = isr.read();
  stringArray2 = new String[value + stringArray1.length];

  for (int a = 0; a < stringArray1.length; a++) {
     stringArray2[a] = stringArray1[a];
  }

  while((charRead = isr.read(inputBuffer)) > 0) {
     //convert the chars to a string
     String readString = String.copyValueOf(inputBuffer, 0, charRead);
     if (readString.contains("\n")) {
        stringArray2[value1] = s;
        s = "";
        value1++;
     } else {
        s += readString;
     }
     inputBuffer = new char[1];
  }
  isr.close();
  fIn.close();

  fIn = openFileInput("confidocument.txt");
```

```
isr = new InputStreamReader(fIn);

inputBuffer = new char[1];
s = "";

value = 0;
value = isr.read();
stringArray3 = new String[value + stringArray2.length];

for (int a = 0; a < stringArray2.length; a++) {
   stringArray3[a] = stringArray2[a];
}

while((charRead = isr.read(inputBuffer)) > 0) {
   //convert the chars to a string
   String readString = String.copyValueOf(inputBuffer, 0, charRead);
   if (readString.contains("\n")) {
      stringArray3[value1] = s;
      s = "";
      value1++;
   } else {
      s += readString;
   }
   inputBuffer = new char[1];
}
isr.close();
fIn.close();

fIn = openFileInput("confipicture.txt");
isr = new InputStreamReader(fIn);

inputBuffer = new char[1];
s = "";

value = 0;
value = isr.read();
stringArray = new String[value + stringArray3.length];

for (int a = 0; a < stringArray3.length; a++) {
   stringArray[a] = stringArray3[a];
}

while((charRead = isr.read(inputBuffer)) > 0) {
   //convert the chars to a string
   String readString = String.copyValueOf(inputBuffer, 0, charRead);
   if (readString.contains("\n")) {
      stringArray[value1] = s;
      s = "";
      value1++;
   } else {
      s += readString;
   }
   inputBuffer = new char[1];
}
isr.close();
fIn.close();

int notAllowed = 0;
int readOnly = 0;
int readWrite = 0;
```

```
for (int a = 0; a < stringArray.length; a++) {
   if (stringArray[a].contains("Not allowed")) {
      notAllowed++;
   }
   else if (stringArray[a].contains("Read-Only")) {
      readOnly++;
   }
   else if (stringArray[a].contains("Read/Write")) {
      readWrite++;
   } else {}
}

String[] notAllowedFile = new String[notAllowed];
String[] readOnlyFile = new String[readOnly];
String[] readWriteFile = new String[readWrite];

int notAllowedLength = 0;
int readOnlyLength = 0;
int readWriteLength = 0;

File[] folders = directory.listFiles();
for (int i = 0; i < folders.length; i++) {
   if (folders[i].getName().contains(".txt") || folders[i].getName().contains(".docx")
      || folders[i].getName().contains(".xlsx") || folders[i].getName().contains(".pdf")
      || folders[i].getName().contains(".pptx") || folders[i].getName().contains(".mp3")
      || folders[i].getName().contains(".mp4") || folders[i].getName().contains(".jpg")
      || folders[i].getName().contains(".ogg")) {
      for (int a = 0; a < stringArray.length; a++) {
         if (stringArray[a].contains(folders[i].getName()) &&
             stringArray[a+1].contains("Read-Only")) {
            readOnlyFile[readOnlyLength] = folders[i].getName();
            readOnlyLength++;
            a++;
         }
         else if (stringArray[a].contains(folders[i].getName()) &&
                  stringArray[a+1].contains("Read/Write")) {
            readWriteFile[readWriteLength] = folders[i].getName();
            readWriteLength++;
            a++;
         }
         else if (stringArray[a].contains(folders[i].getName()) &&
                  stringArray[a+1].contains("Not allowed")) {
            notAllowedFile[notAllowedLength] = folders[i].getName();
            notAllowedLength++;
            a++;
         } else {}
      }
   }
   File[] folderDirectory = folders[i].listFiles();
   if (folderDirectory != null) {
      if (folderDirectory.length > 0) {

for (int j = 0; j < folderDirectory.length; j++) {
   if (folderDirectory[j].getName().contains(".txt") ||
         folderDirectory[j].getName().contains(".docx")
      || folderDirectory[j].getName().contains(".xlsx") ||
         folderDirectory[j].getName().contains(".pdf")
      || folderDirectory[j].getName().contains(".pptx") ||
         folderDirectory[j].getName().contains(".mp3")
      || folderDirectory[j].getName().contains(".mp4") ||
```

```
            folderDirectory[j].getName().contains(".jpg")
         || folderDirectory[j].getName().contains(".ogg")) {
         for (int a = 0; a < stringArray.length; a++) {
            if (stringArray[a].contains(folderDirectory[j].getName()) &&
                stringArray[a+1].contains("Read-Only")) {
               readOnlyFile[readOnlyLength] = folderDirectory[j].getName();
               readOnlyLength++;
               a++;
            }
            else if (stringArray[a].contains(folderDirectory[j].getName()) &&
                     stringArray[a+1].contains("Read/Write")) {
               readWriteFile[readWriteLength] = folderDirectory[j].getName();
               readWriteLength++;
               a++;
            }
            else if (stringArray[a].contains(folderDirectory[j].getName()) &&
                     stringArray[a+1].contains("Not allowed")) {
               notAllowedFile[notAllowedLength] = folderDirectory[j].getName();
               notAllowedLength++;
               a++;
            } else {}
         }
      }
      File[] subFolderDirectory = folderDirectory[j].listFiles();
      if (subFolderDirectory != null) {
         if (subFolderDirectory.length > 0) {

   for (int k = 0; k < subFolderDirectory.length; k++) {
      if (subFolderDirectory[k].getName().contains(".txt") ||
            subFolderDirectory[k].getName().contains(".docx")
         || subFolderDirectory[k].getName().contains(".xlsx") ||
            subFolderDirectory[k].getName().contains(".pdf")
         || subFolderDirectory[k].getName().contains(".pptx") ||
            subFolderDirectory[k].getName().contains(".mp3")
         || subFolderDirectory[k].getName().contains(".mp4") ||
            subFolderDirectory[k].getName().contains(".jpg")
         || subFolderDirectory[k].getName().contains(".ogg")) {
         for (int a = 0; a < stringArray.length; a++) {
            if (stringArray[a].contains(subFolderDirectory[k].getName()) &&
                stringArray[a+1].contains("Read-Only")) {
               readOnlyFile[readOnlyLength] = subFolderDirectory[k].getName();
               readOnlyLength++;
               a++;
            }
            else if (stringArray[a].contains(subFolderDirectory[k].getName()) &&
                     stringArray[a+1].contains("Read/Write")) {
               readWriteFile[readWriteLength] = subFolderDirectory[k].getName();
               readWriteLength++;
               a++;
            }
            else if (stringArray[a].contains(subFolderDirectory[k].getName()) &&
                     stringArray[a+1].contains("Not allowed")) {
               notAllowedFile[notAllowedLength] = subFolderDirectory[k].getName();
               notAllowedLength++;
               a++;
            } else {}
         }
      }
   File[] subSubFolderDirectory = subFolderDirectory[k].listFiles();
   if (subSubFolderDirectory != null) {
```

```
    if (subSubFolderDirectory.length > 0) {

for (int l = 0; l < subSubFolderDirectory.length; l++) {
    if (subSubFolderDirectory[l].getName().contains(".txt") ||
          subSubFolderDirectory[l].getName().contains(".docx")
       || subSubFolderDirectory[l].getName().contains(".xlsx") ||
          subSubFolderDirectory[l].getName().contains(".pdf")
       || subSubFolderDirectory[l].getName().contains(".pptx") ||
          subSubFolderDirectory[l].getName().contains(".mp3")
       || subSubFolderDirectory[l].getName().contains(".mp4") ||
          subSubFolderDirectory[l].getName().contains(".jpg")
       || subSubFolderDirectory[l].getName().contains(".ogg")) {
       for (int a = 0; a < stringArray.length; a++) {
          if (stringArray[a].contains(subSubFolderDirectory[l].getName()) &&
              stringArray[a+1].contains("Read-Only")) {
             readOnlyFile[readOnlyLength] = subSubFolderDirectory[l].getName();
             readOnlyLength++;
             a++;
          }
          else if (stringArray[a].contains(subSubFolderDirectory[l].getName()) &&
                   stringArray[a+1].contains("Read/Write")) {
             readWriteFile[readWriteLength] = subSubFolderDirectory[l].getName();
             readWriteLength++;
             a++;
          }
          else if (stringArray[a].contains(subSubFolderDirectory[l].getName()) &&
                   stringArray[a+1].contains("Not allowed")) {
             notAllowedFile[notAllowedLength] =
                                                  subSubFolderDirectory[l].getName();
             notAllowedLength++;
             a++;
          } else {}
       }
    }
    File[] subSubFolder1Directory = subSubFolderDirectory[l].listFiles();
    if (subSubFolder1Directory != null) {
       if (subSubFolder1Directory.length > 0) {

for (int m = 0; m < subSubFolder1Directory.length; m++) {
    if (subSubFolder1Directory[m].getName().contains(".txt") ||
          subSubFolder1Directory[m].getName().contains(".docx")
       || subSubFolder1Directory[m].getName().contains(".xlsx") ||
          subSubFolder1Directory[m].getName().contains(".pdf")
       || subSubFolder1Directory[m].getName().contains(".pptx") ||
          subSubFolder1Directory[m].getName().contains(".mp3")
       || subSubFolder1Directory[m].getName().contains(".mp4") ||
          subSubFolder1Directory[m].getName().contains(".jpg")
       || subSubFolder1Directory[m].getName().contains(".ogg")) {
       for (int a = 0; a < stringArray.length; a++) {
          if (stringArray[a].contains(subSubFolder1Directory[m].getName() &&
              stringArray[a+1].contains("Read-Only")) {
             readOnlyFile[readOnlyLength] = subSubFolder1Directory[m].getName();
             readOnlyLength++;
             a++;
             }
             else if (stringArray[a].contains(subSubFolder1Directory[m].getName())
                      && stringArray[a+1].contains("Read/Write")) {
                readWriteFile[readWriteLength] =
                                                  subSubFolder1Directory[m].getName();
                readWriteLength++;
```

```
                        a++;
                    }
                    else if (stringArray[a].contains(subSubFolder1Directory[m].getName())
                        && stringArray[a+1].contains("Not allowed")) {
                        notAllowedFile[notAllowedLength] =
                                                    subSubFolder1Directory[m].getName();
                        notAllowedLength++;
                        a++;
                    } else {}
                }
            }
        }
            } else{}
        } else{}
        }
            } else{}
        } else{}
        }
            } else{}
        } else{}
        }
            } else {}
        } else{}
        }

        outStream.write(notAllowed);
        for (int i = 0; i < notAllowed; i++) {
            outStream.write(notAllowedFile[i].getBytes());
            byte[] redundantNAByte = new byte[250-notAllowedFile[i].getBytes().length];
            outStream.write(redundantNAByte);
        }

        outStream.write(readOnly);
        for (int i = 0; i < readOnly; i++) {
            outStream.write(readOnlyFile[i].getBytes());
            byte[] redundantROByte = new byte[250-readOnlyFile[i].getBytes().length];
            outStream.write(redundantROByte);
        }

        outStream.write(readWrite);
        for (int i = 0; i < readWrite; i++) {
            outStream.write(readWriteFile[i].getBytes());
            byte[] redundantRWByte = new byte[250-readWriteFile[i].getBytes().length];
            outStream.write(redundantRWByte);
        }
      } catch (IOException ios) {
            ios.printStackTrace();
      }
      Toast.makeText(getBaseContext(), "Sharing has completed!",
                        Toast.LENGTH_SHORT).show();

      try {
         outStream.flush();
         outStream.close();
         mmSocket.close();
         tmp.close();
      } catch (IOException e) {}
    } catch (IOException e) {}
  } catch (IOException e) {}
} catch (IOException e) {}
```

```
        }
    }

    public void set_packs() {
        final Intent mainIntent = new Intent(Intent.ACTION_MAIN, null);
        mainIntent.addCategory(Intent.CATEGORY_LAUNCHER);
        List<ResolveInfo> pacsList = pm.queryIntentActivities(mainIntent, 0);
        packs = new Pack[pacsList.size()];
        for(int i=0 ; i < pacsList.size() ; i++){
            packs[i]= new Pack();
            packs[i].icon=pacsList.get(i).loadIcon(pm);
            packs[i].name=pacsList.get(i).activityInfo.packageName;
            packs[i].label=pacsList.get(i).loadLabel(pm).toString();
        }
    }

    public class PacReceiver extends BroadcastReceiver{
        @Override
        public void onReceive(Context context, Intent intent) {
            set_packs();
        }
    }
}
```

configuration.java

```
package com.example.example;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.animation.AnimationUtils;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.AdapterView.OnItemClickListener;

public class configuration extends Activity {
   ListView lstView;
   ArrayAdapter<String> mArrayAdapter;

   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.configuration);

      lstView = (ListView)findViewById(R.id.lstViewnum1);
      lstView.setAnimation(AnimationUtils.loadAnimation(this, android.R.anim.slide_in_left));

      TextView txtView = (TextView)findViewById(R.id.txtViewnum1);
      txtView.setText("Please choose the file access permission\n");
      txtView.setTextSize(20);

      mArrayAdapter = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1);
      lstView.setAdapter(mArrayAdapter);
      mArrayAdapter.clear();
      mArrayAdapter.add("Music");
      mArrayAdapter.add("Video");
      mArrayAdapter.add("Document");
      mArrayAdapter.add("Picture");
      lstView.setOnItemClickListener(new ListItemClick());
   }

   class ListItemClick implements OnItemClickListener {
      @Override
      public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
         MainActivity.abc = 0;
         if (mArrayAdapter.getItem(position).equalsIgnoreCase("Music")) {
            startActivity(new Intent("com.example.example.music"));
         }
         else if (mArrayAdapter.getItem(position).equalsIgnoreCase("Video")) {
            startActivity(new Intent("com.example.example.video"));
         }
         else if (mArrayAdapter.getItem(position).equalsIgnoreCase("Document")) {
            startActivity(new Intent("com.example.example.document"));
         }
         else if (mArrayAdapter.getItem(position).equalsIgnoreCase("Picture")) {
            startActivity(new Intent("com.example.example.picture"));
         } else {}
      }
   }
}
```

document.java

```
package com.example.example;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.util.ArrayList;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.animation.AnimationUtils;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.AdapterView.OnItemLongClickListener;
import android.widget.ArrayAdapter;
import android.widget.BaseAdapter;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

public class document extends Activity {
   ListView lstView;
   Person person;
   ArrayList<Person> persons = new ArrayList<Person>();
   String[] strArray;
   ArrayAdapter<String> listArray;

   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.listaccessiblefile);

      lstView = (ListView)findViewById(R.id.lst_view);
      lstView.setAnimation(AnimationUtils.loadAnimation(this, android.R.anim.slide_in_left));

      listArray = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1);
      listArray.clear();

      try {
         //internal storage
         FileInputStream fIn = openFileInput("confidocument.txt");

         InputStreamReader isr = new InputStreamReader(fIn);
         int value1 = 0;

         //the content is read in blocks of 100 characters into a buffer(character array)
```

```
    char[] inputBuffer = new char[1];
    String s = "";

    int value = 0;
    value = isr.read();
    strArray = new String[value];

    int charRead;
    while((charRead = isr.read(inputBuffer)) > 0) {
        //convert the chars to a string
        String readString = String.copyValueOf(inputBuffer, 0, charRead);
        if (readString.contains("\n")) {
            strArray[value1] = s;
            s = "";
            value1++;
        } else {
            s += readString;
        }
        inputBuffer = new char[1];
    }

    Toast.makeText(getBaseContext(), "File loaded successfully",
                        Toast.LENGTH_SHORT).show();
    MainActivity.abc = 1;
} catch (IOException ios) {
    ios.printStackTrace();
}
TextView txtView = (TextView)findViewById(R.id.txt_view);
txtView.setText("Please choose the \n" + "file access permission\n");
txtView.setTextSize(20);

Button btnSave = (Button)findViewById(R.id.btn_save);
btnSave.setText("Save");
btnSave.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        MainActivity.abc = 1;
        int x = 0;
        int y = 0;
        int z = 0;
        String[] a = new String[x];
        String[] b = new String[y];
        String[] c = new String[z];
        for (int i = 0; i < lstView.getCount(); i++) {
            if (persons.get(i).getAge() == "Not allowed") {
                a = new String[1+x];
                x++;
            }
            else if (persons.get(i).getAge() == "Read-Only") {
                b = new String[1+y];
                y++;
            }
            else {
                c = new String[1+z];
                z++;
            }
        }
        x = 0;
        y = 0;
```

```
        z = 0;
        try {
            FileOutputStream fOut = openFileOutput("confidocument.txt", MODE_PRIVATE);

            //convert character stream into a byte stream
            OutputStreamWriter osw = new OutputStreamWriter(fOut);
            osw.write(2*(a.length + b.length + c.length));

            for (int i = 0; i < lstView.getCount(); i++) {
                if (persons.get(i).getAge() == "Not allowed") {
                    a[x] = persons.get(i).getName();
                    osw.write(a[x] + "\n");
                    osw.write("Not allowed" + "\n");
                    x++;
                }
                else if (persons.get(i).getAge() == "Read-Only") {
                    b[y] = persons.get(i).getName();
                    osw.write(b[y] + "\n");
                    osw.write("Read-Only" + "\n");
                    y++;
                }
                else {
                    c[z] = persons.get(i).getName();
                    osw.write(c[z] + "\n");
                    osw.write("Read/Write" + "\n");
                    z++;
                }
            }
            //to ensure all the bytes are written to the file
            osw.flush();
            //close the file
            osw.close();
            //display file saved message
            Toast.makeText(getBaseContext(), "File saved successfully",
                                    Toast.LENGTH_SHORT).show();
        } catch (IOException ioe) {
            ioe.printStackTrace();
        }
        finish();
    }
});

lstView.setAdapter(new MyAdapter(this, persons));
lstView.setOnItemClickListener(new ListItemClick());
lstView.setOnItemLongClickListener(new ListItemLongClick());

File sdCard = Environment.getExternalStorageDirectory();
File directory = new File(sdCard.getAbsolutePath());
File[] folders = directory.listFiles();

persons.clear();

for (int i = 0; i < folders.length; i++) {
    if (folders[i].getName().contains(".txt") || folders[i].getName().contains(".docx")
        || folders[i].getName().contains(".xlsx") || folders[i].getName().contains(".pdf")
        || folders[i].getName().contains(".pptx")) {
        person = new Person();
        person.setName(folders[i].getName());
        listArray.add(folders[i].getAbsolutePath());
        if (MainActivity.abc == 1) {
```

```
            for (int a = 0; a < strArray.length; a++) {
               if (strArray[a].contains(folders[i].getName()) &&
                    strArray[a+1].contains("Read-Only")) {
                  person.setAge("Read-Only");
                  a++;
               }
               else if (strArray[a].contains(folders[i].getName()) &&
                           strArray[a+1].contains("Read/Write")) {
                  person.setAge("Read/Write");
                  a++;
               }
               else if (strArray[a].contains(folders[i].getName()) &&
                           strArray[a+1].contains("Not allowed")) {
                  person.setAge("Not allowed");
                  a++;
               } else {}
            }
         }
         else {
            person.setAge("Read-Only");
         }
         persons.add(person);
      }
      File[] folderDirectory = folders[i].listFiles();
      if (folderDirectory != null) {
         if (folderDirectory.length > 0) {

   for (int j = 0; j < folderDirectory.length; j++) {
      if (folderDirectory[j].getName().contains(".txt") ||
          folderDirectory[j].getName().contains(".docx")
         || folderDirectory[j].getName().contains(".xlsx") ||
            folderDirectory[j].getName().contains(".pdf")
         || folderDirectory[j].getName().contains(".pptx")) {
         person = new Person();
         person.setName(folderDirectory[j].getName());
         listArray.add(folderDirectory[j].getAbsolutePath());
         if (MainActivity.abc == 1) {
            for (int a = 0; a < strArray.length; a++) {
               if (strArray[a].contains(folderDirectory[j].getName()) &&
                    strArray[a+1].contains("Read-Only")) {
                  person.setAge("Read-Only");
                  a++;
               }
               else if (strArray[a].contains(folderDirectory[j].getName()) &&
                           strArray[a+1].contains("Read/Write")) {
                  person.setAge("Read/Write");
                  a++;
               }
               else if (strArray[a].contains(folderDirectory[j].getName()) &&
                           strArray[a+1].contains("Not allowed")) {
                  person.setAge("Not allowed");
                  a++;
               } else {}
            }
         }
         else {
            person.setAge("Read-Only");
         }
         persons.add(person);
      }
```

```
    File[] subFolderDirectory = folderDirectory[j].listFiles();
    if (subFolderDirectory != null) {
        if (subFolderDirectory.length > 0) {

for (int k = 0; k < subFolderDirectory.length; k++) {
    if (subFolderDirectory[k].getName().contains(".txt") ||
         subFolderDirectory[k].getName().contains(".docx")
       || subFolderDirectory[k].getName().contains(".xlsx") ||
          subFolderDirectory[k].getName().contains(".pdf")
       || subFolderDirectory[k].getName().contains(".pptx")) {
        person = new Person();
        person.setName(subFolderDirectory[k].getName());
        listArray.add(subFolderDirectory[k].getAbsolutePath());
        if (MainActivity.abc == 1) {
            for (int a = 0; a < strArray.length; a++) {
                if (strArray[a].contains(subFolderDirectory[k].getName()) &&
                     strArray[a+1].contains("Read-Only")) {
                    person.setAge("Read-Only");
                    a++;
                }
                else if (strArray[a].contains(subFolderDirectory[k].getName()) &&
                            strArray[a+1].contains("Read/Write")) {
                    person.setAge("Read/Write");
                    a++;
                }
                else if (strArray[a].contains(subFolderDirectory[k].getName()) &&
                            strArray[a+1].contains("Not allowed")) {
                    person.setAge("Not allowed");
                    a++;
                } else {}
            }
        }
        else {
            person.setAge("Read-Only");
        }
        persons.add(person);
    }
File[] subSubFolderDirectory = subFolderDirectory[k].listFiles();
    if (subSubFolderDirectory != null) {
        if (subSubFolderDirectory.length > 0) {

for (int l = 0; l < subSubFolderDirectory.length; l++) {
    if (subSubFolderDirectory[l].getName().contains(".txt") ||
         subSubFolderDirectory[l].getName().contains(".docx")
       || subSubFolderDirectory[l].getName().contains(".xlsx") ||
          subSubFolderDirectory[l].getName().contains(".pdf")
       || subSubFolderDirectory[l].getName().contains(".pptx")) {
        person = new Person();
        person.setName(subSubFolderDirectory[l].getName());
        listArray.add(subSubFolderDirectory[l].getAbsolutePath());
        if (MainActivity.abc == 1) {
            for (int a = 0; a < strArray.length; a++) {
                if (strArray[a].contains(subSubFolderDirectory[l].getName()) &&
                     strArray[a+1].contains("Read-Only")) {
                    person.setAge("Read-Only");
                    a++;
                }
                else if (strArray[a].contains(subSubFolderDirectory[l].getName()) &&
                            strArray[a+1].contains("Read/Write")) {
                    person.setAge("Read/Write");
```

```
                    a++;
                }
                else if (strArray[a].contains(subSubFolderDirectory[l].getName()) &&
                        strArray[a+1].contains("Not allowed")) {
                    person.setAge("Not allowed");
                    a++;
                } else {}
            }
        }
        else {
            person.setAge("Read-Only");
        }
        persons.add(person);
    }
  }
        } else{}
    } else{}
  }
        } else{}
    } else{}
  }
        } else {}
    } else{}
  }
}

class Person {
    String name;
    String accessibility;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getAge() {
        return accessibility;
    }

    public void setAge(String age) {
        this.accessibility = age;
    }

}

class MyAdapter extends BaseAdapter {
    private Context context;
    private ArrayList<Person> persons;

    public MyAdapter(Context context, ArrayList<Person> persons) {
        this.context = context;
        this.persons = persons;
    }

    @Override
    public int getCount() {
        return persons.size();
```

```
}

@Override
public Object getItem(int position) {
   return persons.get(position);
}

@Override
public long getItemId(int position) {
   return 0;
}

class ViewHolder {
   TextView text;
   ImageView icon;
}

@Override
public View getView(int position, View convertView, ViewGroup parent) {
   ViewHolder viewHolder;
   LayoutInflater li = (LayoutInflater)
                        context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);

   if (convertView==null){
      convertView = li.inflate(R.layout.main3, null);

      viewHolder = new ViewHolder();
      viewHolder.text= (TextView)convertView.findViewById(R.id.txtView);
      viewHolder.icon= (ImageView)convertView.findViewById(R.id.imgView);

      convertView.setTag(viewHolder);
   }
   else {
      viewHolder = (ViewHolder) convertView.getTag();
   }

   viewHolder.text.setText(persons.get(position).getName() + "\n" + "(" +
                                    persons.get(position).getAge() + ")");
   viewHolder.text.setTextColor(Color.GREEN);

   if (persons.get(position).getName().contains(".txt")) {
      viewHolder.icon.setImageResource(R.drawable.txt);
   }

   else if (persons.get(position).getName().contains(".docx")) {
      viewHolder.icon.setImageResource(R.drawable.docx);
   }

   else if (persons.get(position).getName().contains(".xlsx")) {
      viewHolder.icon.setImageResource(R.drawable.xlsx);
   }

   else if (persons.get(position).getName().contains(".pptx")) {
      viewHolder.icon.setImageResource(R.drawable.pptx);
   }

   else if (persons.get(position).getName().contains(".pdf")) {
      viewHolder.icon.setImageResource(R.drawable.pdf);
   }
```

```
            else {
                viewHolder.icon.setImageResource(R.drawable.folder);
            }
            return convertView;
        }
    }

    class ListItemClick implements OnItemClickListener {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
            if (persons.get(position).getAge() == "Not allowed") {
                persons.get(position).setAge("Read-Only");
                lstView.setAdapter(new MyAdapter(document.this, persons));
            }
            else if (persons.get(position).getAge() == "Read-Only") {
                persons.get(position).setAge("Read/Write");
                lstView.setAdapter(new MyAdapter(document.this, persons));
            }
            else {
                persons.get(position).setAge("Not allowed");
                lstView.setAdapter(new MyAdapter(document.this, persons));
            }
        }
    }
    class ListItemLongClick implements OnItemLongClickListener {
        @Override
        public boolean onItemLongClick(AdapterView<?> parent, View view, int position, long id) {
            File file = new File(listArray.getItem(position));
            Uri url = Uri.fromFile(file);
            Intent intent = new Intent(Intent.ACTION_VIEW);
            // Check what kind of file you are trying to open, by comparing the url with extensions.
            // When the if condition is matched, plugin sets the correct intent (mime) type,
            // so Android knew what application to use to open the file
            if (file.toString().contains(".doc") || file.toString().contains(".docx")) {
                // Word document
                intent.setDataAndType(url, "application/msword");
            } else if(file.toString().contains(".pdf")) {
                // PDF file
                intent.setDataAndType(url, "application/pdf");
            } else if(file.toString().contains(".ppt") || file.toString().contains(".pptx")) {
                // Powerpoint file
                intent.setDataAndType(url, "application/vnd.ms-powerpoint");
            } else if(file.toString().contains(".xls") || file.toString().contains(".xlsx")) {
                // Excel file
                intent.setDataAndType(url, "application/vnd.ms-excel");
            } else if(file.toString().contains(".txt")) {
                // Text file
                intent.setDataAndType(url, "text/plain");
            } else {
                //if you want you can also define the intent type for any other file
                //additionally use else clause below, to manage other unknown extensions
                //in this case, Android will show all applications installed on the device
                //so you can choose which application to use
                intent.setDataAndType(url, "*/*");
            }
            startActivity(intent);
            return false;
        }
    }
}
```

music.java

```
package com.example.example;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.util.ArrayList;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.animation.AnimationUtils;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.AdapterView.OnItemLongClickListener;
import android.widget.ArrayAdapter;
import android.widget.BaseAdapter;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

public class music extends Activity {
   ListView lstView;
   Person person;
   ArrayList<Person> persons = new ArrayList<Person>();
   String[] strArray;
   ArrayAdapter<String> listArray;

   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.listaccessiblefile);

      lstView = (ListView)findViewById(R.id.lst_view);
      lstView.setAnimation(AnimationUtils.loadAnimation(this, android.R.anim.slide_in_left));

      listArray = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1);
      listArray.clear();

      try {
         //internal storage
         FileInputStream fIn = openFileInput("confimusic.txt");

         InputStreamReader isr = new InputStreamReader(fIn);
         int value1 = 0;

         //the content is read in blocks of 100 characters into a buffer(character array)
```

```
    char[] inputBuffer = new char[1];
    String s = "";

    int value = 0;
    value = isr.read();
    strArray = new String[value];

    int charRead;
    while((charRead = isr.read(inputBuffer)) > 0) {
        //convert the chars to a string
        String readString = String.copyValueOf(inputBuffer, 0, charRead);
        if (readString.contains("\n")) {
            strArray[value1] = s;
            s = "";
            value1++;
        } else {
            s += readString;
        }
        inputBuffer = new char[1];
    }

    Toast.makeText(getBaseContext(), "File loaded successfully",
                            Toast.LENGTH_SHORT).show();
    MainActivity.abc = 1;
} catch (IOException ios) {
    ios.printStackTrace();
}
TextView txtView = (TextView)findViewById(R.id.txt_view);
txtView.setText("Please choose the \n" + "file access permission\n");
txtView.setTextSize(20);

Button btnSave = (Button)findViewById(R.id.btn_save);
btnSave.setText("Save");
btnSave.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        MainActivity.abc = 1;
        int x = 0;
        int y = 0;
        int z = 0;
        String[] a = new String[x];
        String[] b = new String[y];
        String[] c = new String[z];
        for (int i = 0; i < lstView.getCount(); i++) {
            if (persons.get(i).getAge() == "Not allowed") {
                a = new String[1+x];
                x++;
            }
            else if (persons.get(i).getAge() == "Read-Only") {
                b = new String[1+y];
                y++;
            }
            else {
                c = new String[1+z];
                z++;
            }
        }
        x = 0;
        y = 0;
```

```
        z = 0;
        try {
            FileOutputStream fOut = openFileOutput("confimusic.txt", MODE_PRIVATE);

            //convert character stream into a byte stream
            OutputStreamWriter osw = new OutputStreamWriter(fOut);
            osw.write(2*(a.length + b.length + c.length));

            for (int i = 0; i < lstView.getCount(); i++) {
                if (persons.get(i).getAge() == "Not allowed") {
                    a[x] = persons.get(i).getName();
                    osw.write(a[x] + "\n");
                    osw.write("Not allowed" + "\n");
                    x++;
                }
                else if (persons.get(i).getAge() == "Read-Only") {
                    b[y] = persons.get(i).getName();
                    osw.write(b[y] + "\n");
                    osw.write("Read-Only" + "\n");
                    y++;
                }
                else {
                    c[z] = persons.get(i).getName();
                    osw.write(c[z] + "\n");
                    osw.write("Read/Write" + "\n");
                    z++;
                }
            }
            //to ensure all the bytes are written to the file
            osw.flush();
            //close the file
            osw.close();
            //display file saved message
            Toast.makeText(getBaseContext(), "File saved successfully",
                                    Toast.LENGTH_SHORT).show();
        } catch (IOException ioe) {
            ioe.printStackTrace();
        }
        finish();
    }
});

lstView.setAdapter(new MyAdapter(this, persons));
lstView.setOnItemClickListener(new ListItemClick());
lstView.setOnItemLongClickListener(new ListItemLongClick());

File sdCard = Environment.getExternalStorageDirectory();
File directory = new File(sdCard.getAbsolutePath());
File[] folders = directory.listFiles();

persons.clear();

for (int i = 0; i < folders.length; i++) {
    if (folders[i].getName().contains(".mp3") || folders[i].getName().contains(".ogg")) {
        person = new Person();
        person.setName(folders[i].getName());
        listArray.add(folders[i].getAbsolutePath());
        if (MainActivity.abc == 1) {
            for (int a = 0; a < strArray.length; a++) {
                if (strArray[a].contains(folders[i].getName()) &&
```

```
                    strArray[a+1].contains("Read-Only")) {
                person.setAge("Read-Only");
                a++;
            }
            else if (strArray[a].contains(folders[i].getName()) &&
                    strArray[a+1].contains("Read/Write")) {
                person.setAge("Read/Write");
                a++;
            }
            else if (strArray[a].contains(folders[i].getName()) &&
                    strArray[a+1].contains("Not allowed")) {
                person.setAge("Not allowed");
                a++;
            } else {}
        }
    }
    else {
        person.setAge("Read-Only");
    }
    persons.add(person);
}
File[] folderDirectory = folders[i].listFiles();
if (folderDirectory != null) {
    if (folderDirectory.length > 0) {

for (int j = 0; j < folderDirectory.length; j++) {
    if (folderDirectory[j].getName().contains(".mp3") ||
        folderDirectory[j].getName().contains(".ogg")) {
        person = new Person();
        person.setName(folderDirectory[j].getName());
        listArray.add(folderDirectory[j].getAbsolutePath());
        if (MainActivity.abc == 1) {
            for (int a = 0; a < strArray.length; a++) {
                if (strArray[a].contains(folderDirectory[j].getName()) &&
                    strArray[a+1].contains("Read-Only")) {
                    person.setAge("Read-Only");
                    a++;
                }
                else if (strArray[a].contains(folderDirectory[j].getName()) &&
                        strArray[a+1].contains("Read/Write"))
                {
                    person.setAge("Read/Write");
                    a++;
                }
                else if (strArray[a].contains(folderDirectory[j].getName()) &&
                        strArray[a+1].contains("Not allowed"))
                {
                    person.setAge("Not allowed");
                    a++;
                } else {}
            }
        }
        else {
            person.setAge("Read-Only");
        }
        persons.add(person);
    }
    File[] subFolderDirectory = folderDirectory[j].listFiles();
    if (subFolderDirectory != null) {
        if (subFolderDirectory.length > 0) {
```

```
for (int k = 0; k < subFolderDirectory.length; k++) {
   if (subFolderDirectory[k].getName().contains(".mp3") ||
       subFolderDirectory[k].getName().contains(".ogg")) {
      person = new Person();
      person.setName(subFolderDirectory[k].getName());
      listArray.add(subFolderDirectory[k].getAbsolutePath());
      if (MainActivity.abc == 1) {
         for (int a = 0; a < strArray.length; a++) {
            if (strArray[a].contains(subFolderDirectory[k].getName()) &&
                strArray[a+1].contains("Read-Only")) {
               person.setAge("Read-Only");
               a++;
            }
            else if (strArray[a].contains(subFolderDirectory[k].getName()) &&
                     strArray[a+1].contains("Read/Write")) {
               person.setAge("Read/Write");
               a++;
            }
            else if (strArray[a].contains(subFolderDirectory[k].getName()) &&
                     strArray[a+1].contains("Not allowed")) {
               person.setAge("Not allowed");
               a++;
            } else {}
         }
      }
      else {
         person.setAge("Read-Only");
      }
      persons.add(person);
   }
File[] subSubFolderDirectory = subFolderDirectory[k].listFiles();
   if (subSubFolderDirectory != null) {
      if (subSubFolderDirectory.length > 0) {

for (int l = 0; l < subSubFolderDirectory.length; l++) {
   if (subSubFolderDirectory[l].getName().contains(".mp3") ||
       subSubFolderDirectory[l].getName().contains(".ogg")) {
      person = new Person();
      person.setName(subSubFolderDirectory[l].getName());
      listArray.add(subSubFolderDirectory[l].getAbsolutePath());
      if (MainActivity.abc == 1) {
         for (int a = 0; a < strArray.length; a++) {
            if (strArray[a].contains(subSubFolderDirectory[l].getName()) &&
                strArray[a+1].contains("Read-Only"))
            {
               person.setAge("Read-Only");
               a++;
            }
            else if (strArray[a].contains(subSubFolderDirectory[l].getName()) &&
                     strArray[a+1].contains("Read/Write")) {
               person.setAge("Read/Write");
               a++;
            }
            else if (strArray[a].contains(subSubFolderDirectory[l].getName()) &&
                     strArray[a+1].contains("Not allowed")) {
               person.setAge("Not allowed");
               a++;
            } else {}
         }
```

```
            }
            else {
               person.setAge("Read-Only");
            }
            persons.add(person);
         }
         File[] subSubFolder1Directory = subSubFolderDirectory[l].listFiles();
         if (subSubFolder1Directory != null) {
            if (subSubFolder1Directory.length > 0) {

      for (int m = 0; m < subSubFolder1Directory.length; m++) {
         if (subSubFolder1Directory[m].getName().contains(".mp3") ||
             subSubFolder1Directory[m].getName().contains(".ogg"))
         {
            person = new Person();
            person.setName(subSubFolder1Directory[m].getName());
            listArray.add(subSubFolder1Directory[m].getAbsolutePath());
            if (MainActivity.abc == 1) {
               for (int a = 0; a < strArray.length; a++) {
                  if (strArray[a].contains(subSubFolder1Directory[m].getName()) &&
                      strArray[a+1].contains("Read-Only"))
                  {
                     person.setAge("Read-Only");
                     a++;
                  }
                  else if (strArray[a].contains(subSubFolder1Directory[m].getName()) &&
                           strArray[a+1].contains("Read/Write")) {
                     person.setAge("Read/Write");
                     a++;
                  }
                  else if (strArray[a].contains(subSubFolder1Directory[m].getName()) &&
                           strArray[a+1].contains("Not allowed")) {
                     person.setAge("Not allowed");
                     a++;
                  } else {}
               }
            }
            else {
               person.setAge("Read-Only");
            }
            persons.add(person);
         }
      }
            } else{}
         } else{}
      }
            } else{}
         } else{}
      }
            } else{}
         } else{}
      }
            } else {}
         } else{}
      }
   }

   class Person {
      String name;
      String accessibility;
```

```
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getAge() {
        return accessibility;
    }

    public void setAge(String age) {
        this.accessibility = age;
    }

}

class MyAdapter extends BaseAdapter {

    private Context context;
    private ArrayList<Person> persons;

    public MyAdapter(Context context, ArrayList<Person> persons) {
        this.context = context;
        this.persons = persons;
    }

    @Override
    public int getCount() {
        return persons.size();
    }

    @Override
    public Object getItem(int position) {
        return persons.get(position);
    }

    @Override
    public long getItemId(int position) {
        return 0;
    }

    class ViewHolder {
        TextView text;
        ImageView icon;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        ViewHolder viewHolder;
        LayoutInflater li = (LayoutInflater)
                            context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);

        if (convertView==null){
            convertView = li.inflate(R.layout.main3, null);

            viewHolder = new ViewHolder();
            viewHolder.text= (TextView)convertView.findViewById(R.id.txtView);
```

```
            viewHolder.icon= (ImageView)convertView.findViewById(R.id.imgView);

            convertView.setTag(viewHolder);
        }
        else {
            viewHolder = (ViewHolder) convertView.getTag();
        }

        viewHolder.text.setText(persons.get(position).getName() + "\n" + "(" +
                                    persons.get(position).getAge() + ")");
        viewHolder.text.setTextColor(Color.GREEN);

        if (persons.get(position).getName().contains(".ogg")) {
            viewHolder.icon.setImageResource(R.drawable.ogg);
        }

        else if (persons.get(position).getName().contains(".mp3")) {
            viewHolder.icon.setImageResource(R.drawable.music);
        }

        else {
            viewHolder.icon.setImageResource(R.drawable.folder);
        }

        return convertView;
    }
}

class ListItemClick implements OnItemClickListener {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        if (persons.get(position).getAge() == "Not allowed") {
            persons.get(position).setAge("Read-Only");
            lstView.setAdapter(new MyAdapter(music.this, persons));
        }
        else if (persons.get(position).getAge() == "Read-Only") {
            persons.get(position).setAge("Read/Write");
            lstView.setAdapter(new MyAdapter(music.this, persons));
        }
        else {
            persons.get(position).setAge("Not allowed");
            lstView.setAdapter(new MyAdapter(music.this, persons));
        }
    }
}

class ListItemLongClick implements OnItemLongClickListener {
    @Override
    public boolean onItemLongClick(AdapterView<?> parent, View view, int position, long id) {
        File file = new File(listArray.getItem(position));
        Uri url = Uri.fromFile(file);
        Intent intent = new Intent(Intent.ACTION_VIEW);
        // Check what kind of file you are trying to open, by comparing the url with extensions.
        // When the if condition is matched, plugin sets the correct intent (mime) type,
        // so Android knew what application to use to open the file

        if(file.toString().contains(".wav") || file.toString().contains(".mp3") ||
             file.toString().contains(".ogg")) {
            // WAV audio file
            intent.setDataAndType(url, "audio/x-wav");
```

```
            } else {
                //if you want you can also define the intent type for any other file

                //additionally use else clause below, to manage other unknown extensions
                //in this case, Android will show all applications installed on the device
                //so you can choose which application to use
                intent.setDataAndType(url, "*/*");
            }
            startActivity(intent);
            return false;
        }
    }
}
```

video.java

```
package com.example.example;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.util.ArrayList;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.animation.AnimationUtils;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.AdapterView.OnItemLongClickListener;
import android.widget.ArrayAdapter;
import android.widget.BaseAdapter;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

public class video extends Activity {
   ListView lstView;
   Person person;
   ArrayList<Person> persons = new ArrayList<Person>();
   String[] strArray;
   ArrayAdapter<String> listArray;

   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.listaccessiblefile);

      lstView = (ListView)findViewById(R.id.lst_view);
      lstView.setAnimation(AnimationUtils.loadAnimation(this, android.R.anim.slide_in_left));

      listArray = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1);
      listArray.clear();

      try {
         //internal storage
         FileInputStream fIn = openFileInput("confivideo.txt");

         InputStreamReader isr = new InputStreamReader(fIn);
         int value1 = 0;

         //the content is read in blocks of 100 characters into a buffer(character array)
```

```
    char[] inputBuffer = new char[1];
    String s = "";

    int value = 0;
    value = isr.read();
    strArray = new String[value];

    int charRead;
    while((charRead = isr.read(inputBuffer)) > 0) {
        //convert the chars to a string
        String readString = String.copyValueOf(inputBuffer, 0, charRead);
        if (readString.contains("\n")) {
            strArray[value1] = s;
            s = "";
            value1++;
        } else {
            s += readString;
        }
        inputBuffer = new char[1];
    }

    Toast.makeText(getBaseContext(), "File loaded successfully",
                        Toast.LENGTH_SHORT).show();
    MainActivity.abc = 1;
} catch (IOException ios) {
    ios.printStackTrace();
}
TextView txtView = (TextView)findViewById(R.id.txt_view);
txtView.setText("Please choose the \n" + "file access permission\n");
txtView.setTextSize(20);

Button btnSave = (Button)findViewById(R.id.btn_save);
btnSave.setText("Save");
btnSave.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        MainActivity.abc = 1;
        int x = 0;
        int y = 0;
        int z = 0;
        String[] a = new String[x];
        String[] b = new String[y];
        String[] c = new String[z];
        for (int i = 0; i < lstView.getCount(); i++) {
            if (persons.get(i).getAge() == "Not allowed") {
                a = new String[1+x];
                x++;
            }
            else if (persons.get(i).getAge() == "Read-Only") {
                b = new String[1+y];
                y++;
            }
            else {
                c = new String[1+z];
                z++;
            }
        }
        x = 0;
        y = 0;
```

```
        z = 0;
        try {
            //the file is readable by other applications (internal storage)
            FileOutputStream fOut = openFileOutput("confivideo.txt", MODE_PRIVATE);

            //convert character stream into a byte stream
            OutputStreamWriter osw = new OutputStreamWriter(fOut);
            osw.write(2*(a.length + b.length + c.length));

            for (int i = 0; i < lstView.getCount(); i++) {
                if (persons.get(i).getAge() == "Not allowed") {
                    a[x] = persons.get(i).getName();
                    osw.write(a[x] + "\n");
                    osw.write("Not allowed" + "\n");
                    x++;
                }
                else if (persons.get(i).getAge() == "Read-Only") {
                    b[y] = persons.get(i).getName();
                    osw.write(b[y] + "\n");
                    osw.write("Read-Only" + "\n");
                    y++;
                }
                else {
                    c[z] = persons.get(i).getName();
                    osw.write(c[z] + "\n");
                    osw.write("Read/Write" + "\n");
                    z++;
                }
            }
            //to ensure all the bytes are written to the file
            osw.flush();
            //close the file
            osw.close();
            //display file saved message
            Toast.makeText(getBaseContext(), "File saved successfully",
                                    Toast.LENGTH_SHORT).show();
        } catch (IOException ioe) {
            ioe.printStackTrace();
        }
        finish();
    }
});

lstView.setAdapter(new MyAdapter(this, persons));
lstView.setOnItemClickListener(new ListItemClick());
lstView.setOnItemLongClickListener(new ListItemLongClick());

File sdCard = Environment.getExternalStorageDirectory();
File directory = new File(sdCard.getAbsolutePath());
File[] folders = directory.listFiles();

persons.clear();

for (int i = 0; i < folders.length; i++) {
    if (folders[i].getName().contains(".mp4")) {
        person = new Person();
        person.setName(folders[i].getName());
        listArray.add(folders[i].getAbsolutePath());
        if (MainActivity.abc == 1) {
            for (int a = 0; a < strArray.length; a++) {
```

```
                if (strArray[a].contains(folders[i].getName()) &&
                    strArray[a+1].contains("Read-Only")) {
                   person.setAge("Read-Only");
                   a++;
                }
                else if (strArray[a].contains(folders[i].getName()) &&
                         strArray[a+1].contains("Read/Write")) {
                   person.setAge("Read/Write");
                   a++;
                }
                else if (strArray[a].contains(folders[i].getName()) &&
                         strArray[a+1].contains("Not allowed")) {
                   person.setAge("Not allowed");
                   a++;
                } else {}
             }
          }
          else {
             person.setAge("Read-Only");
          }
          persons.add(person);
       }
       File[] folderDirectory = folders[i].listFiles();
       if (folderDirectory != null) {
          if (folderDirectory.length > 0) {

    for (int j = 0; j < folderDirectory.length; j++) {
       if (folderDirectory[j].getName().contains(".mp4")) {
          person = new Person();
          person.setName(folderDirectory[j].getName());
          listArray.add(folderDirectory[j].getAbsolutePath());
          if (MainActivity.abc == 1) {
             for (int a = 0; a < strArray.length; a++) {
                if (strArray[a].contains(folderDirectory[j].getName()) &&
                    strArray[a+1].contains("Read-Only")) {
                   person.setAge("Read-Only");
                   a++;
                }
                else if (strArray[a].contains(folderDirectory[j].getName()) &&
                         strArray[a+1].contains("Read/Write"))
                {
                   person.setAge("Read/Write");
                   a++;
                }
                else if (strArray[a].contains(folderDirectory[j].getName()) &&
                         strArray[a+1].contains("Not allowed"))
                {
                   person.setAge("Not allowed");
                   a++;
                } else {}
             }
          }
          else {
             person.setAge("Read-Only");
          }
          persons.add(person);
       }
       File[] subFolderDirectory = folderDirectory[j].listFiles();
       if (subFolderDirectory != null) {
          if (subFolderDirectory.length > 0) {
```

```
for (int k = 0; k < subFolderDirectory.length; k++) {
   if (subFolderDirectory[k].getName().contains(".mp4")) {
      person = new Person();
      person.setName(subFolderDirectory[k].getName());
      listArray.add(subFolderDirectory[k].getAbsolutePath());
      if (MainActivity.abc == 1) {
         for (int a = 0; a < strArray.length; a++) {
            if (strArray[a].contains(subFolderDirectory[k].getName()) &&
                strArray[a+1].contains("Read-Only")) {
               person.setAge("Read-Only");
               a++;
            }
            else if (strArray[a].contains(subFolderDirectory[k].getName()) &&
                     strArray[a+1].contains("Read/Write")) {
               person.setAge("Read/Write");
               a++;
            }
            else if (strArray[a].contains(subFolderDirectory[k].getName()) &&
                     strArray[a+1].contains("Not allowed")) {
               person.setAge("Not allowed");
               a++;
            } else {}
         }
      }
      else {
         person.setAge("Read-Only");
      }
      persons.add(person);
   }
File[] subSubFolderDirectory = subFolderDirectory[k].listFiles();
   if (subSubFolderDirectory != null) {
      if (subSubFolderDirectory.length > 0) {

for (int l = 0; l < subSubFolderDirectory.length; l++) {
   if (subSubFolderDirectory[l].getName().contains(".mp4")) {
      person = new Person();
      person.setName(subSubFolderDirectory[l].getName());
      listArray.add(subSubFolderDirectory[l].getAbsolutePath());
      if (MainActivity.abc == 1) {
         for (int a = 0; a < strArray.length; a++) {
            if (strArray[a].contains(subSubFolderDirectory[l].getName()) &&
                strArray[a+1].contains("Read-Only")) {
               person.setAge("Read-Only");
               a++;
            }
            else if (strArray[a].contains(subSubFolderDirectory[l].getName()) &&
                     strArray[a+1].contains("Read/Write")) {
               person.setAge("Read/Write");
               a++;
            }
            else if (strArray[a].contains(subSubFolderDirectory[l].getName()) &&
                     strArray[a+1].contains("Not allowed")) {
               person.setAge("Not allowed");
               a++;
            } else {}
         }
      }
      else {
         person.setAge("Read-Only");
```

```
            }
            persons.add(person);
        }
        File[] subSubFolder1Directory = subSubFolderDirectory[l].listFiles();
        if (subSubFolder1Directory != null) {
            if (subSubFolder1Directory.length > 0) {

    for (int m = 0; m < subSubFolder1Directory.length; m++) {
        if (subSubFolder1Directory[m].getName().contains(".mp4")) {
            person = new Person();
            person.setName(subSubFolder1Directory[m].getName());
            listArray.add(subSubFolder1Directory[m].getAbsolutePath());
            if (MainActivity.abc == 1) {
                for (int a = 0; a < strArray.length; a++) {
                    if (strArray[a].contains(subSubFolder1Directory[m].getName()) &&
                         strArray[a+1].contains("Read-Only")) {
                        person.setAge("Read-Only");
                        a++;
                    }
                    else if (strArray[a].contains(subSubFolder1Directory[m].getName()) &&
                            strArray[a+1].contains("Read/Write")) {
                        person.setAge("Read/Write");
                        a++;
                    }
                    else if (strArray[a].contains(subSubFolder1Directory[m].getName()) &&
                            strArray[a+1].contains("Not allowed")) {
                        person.setAge("Not allowed");
                        a++;
                    } else {}
                }
            }
            else {
                person.setAge("Read-Only");
            }
            persons.add(person);
        }
    }
            } else{}
        } else{}
    }
            } else{}
        } else{}
    }
            } else{}
        } else{}
    }
            } else {}
        } else{}
    }
}

class Person {
    String name;
    String accessibility;

    public String getName() {
        return name;
    }

    public void setName(String name) {
```

```
        this.name = name;
    }

    public String getAge() {
        return accessibility;
    }

    public void setAge(String age) {
        this.accessibility = age;
    }

}

class MyAdapter extends BaseAdapter {
    private Context context;
    private ArrayList<Person> persons;

    public MyAdapter(Context context, ArrayList<Person> persons) {
        this.context = context;
        this.persons = persons;
    }

    @Override
    public int getCount() {
        return persons.size();
    }

    @Override
    public Object getItem(int position) {
        return persons.get(position);
    }

    @Override
    public long getItemId(int position) {
        return 0;
    }

    class ViewHolder {
        TextView text;
        ImageView icon;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        ViewHolder viewHolder;
        LayoutInflater li = (LayoutInflater)
                            context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);

        if (convertView==null){
            convertView = li.inflate(R.layout.main3, null);

            viewHolder = new ViewHolder();
            viewHolder.text= (TextView)convertView.findViewById(R.id.txtView);
            viewHolder.icon= (ImageView)convertView.findViewById(R.id.imgView);

            convertView.setTag(viewHolder);
        }
        else {
            viewHolder = (ViewHolder) convertView.getTag();
        }
```

```
            viewHolder.text.setText(persons.get(position).getName() + "\n" + "(" +
                                        persons.get(position).getAge() + ")");
            viewHolder.text.setTextColor(Color.GREEN);

            if (persons.get(position).getName().contains(".mp4")) {
               viewHolder.icon.setImageResource(R.drawable.video);
            }
            else {
               viewHolder.icon.setImageResource(R.drawable.folder);
            }
            return convertView;
         }
      }

      class ListItemClick implements OnItemClickListener {
         @Override
         public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
            if (persons.get(position).getAge() == "Not allowed") {
               persons.get(position).setAge("Read-Only");
               lstView.setAdapter(new MyAdapter(video.this, persons));
            }
            else if (persons.get(position).getAge() == "Read-Only") {
               persons.get(position).setAge("Read/Write");
               lstView.setAdapter(new MyAdapter(video.this, persons));
            }
            else {
               persons.get(position).setAge("Not allowed");
               lstView.setAdapter(new MyAdapter(video.this, persons));
            }
         }
      }

      class ListItemLongClick implements OnItemLongClickListener {
         @Override
         public boolean onItemLongClick(AdapterView<?> parent, View view, int position, long id) {
            File file = new File(listArray.getItem(position));
            Uri url = Uri.fromFile(file);
            Intent intent = new Intent(Intent.ACTION_VIEW);
            // Check what kind of file you are trying to open, by comparing the url with extensions.
            // When the if condition is matched, plugin sets the correct intent (mime) type,
            // so Android knew what application to use to open the file
            if (file.toString().contains(".3gp") || file.toString().contains(".mpg") ||
                 file.toString().contains(".mpeg")
              || file.toString().contains(".mpe") || file.toString().contains(".mp4") ||
                 file.toString().contains(".avi")) {
               // Video files
               intent.setDataAndType(url, "video/*");
            } else {
               //if you want you can also define the intent type for any other file
               //additionally use else clause below, to manage other unknown extensions
               //in this case, Android will show all applications installed on the device
               //so you can choose which application to use
               intent.setDataAndType(url, "*/*");
            }
            startActivity(intent);
            return false;
         }
      }
   }
```

picture.java

```
package com.example.example;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.util.ArrayList;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.animation.AnimationUtils;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.AdapterView.OnItemLongClickListener;
import android.widget.ArrayAdapter;
import android.widget.BaseAdapter;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

public class picture extends Activity {
   ListView lstView;
   Person person;
   ArrayList<Person> persons = new ArrayList<Person>();
   String[] strArray;
   ArrayAdapter<String> listArray;

   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.listaccessiblefile);

      lstView = (ListView)findViewById(R.id.lst_view);
      lstView.setAnimation(AnimationUtils.loadAnimation(this, android.R.anim.slide_in_left));

      listArray = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1);
      listArray.clear();

      try {
         //internal storage
         FileInputStream fIn = openFileInput("confipicture.txt");

         InputStreamReader isr = new InputStreamReader(fIn);
         int value1 = 0;

         //the content is read in blocks of 100 characters into a buffer(character array)
```

```
    char[] inputBuffer = new char[1];
    String s = "";

    int value = 0;
    value = isr.read();
    strArray = new String[value];

    int charRead;
    while((charRead = isr.read(inputBuffer)) > 0) {
        //convert the chars to a string
        String readString = String.copyValueOf(inputBuffer, 0, charRead);
        if (readString.contains("\n")) {
            strArray[value1] = s;
            s = "";
            value1++;
        } else {
            s += readString;
        }
        inputBuffer = new char[1];
    }

    Toast.makeText(getBaseContext(), "File loaded successfully",
                            Toast.LENGTH_SHORT).show();
    MainActivity.abc = 1;
} catch (IOException ios) {
    ios.printStackTrace();
}
TextView txtView = (TextView)findViewById(R.id.txt_view);
txtView.setText("Please choose the \n" + "file access permission\n");
txtView.setTextSize(20);

Button btnSave = (Button)findViewById(R.id.btn_save);
btnSave.setText("Save");
btnSave.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        MainActivity.abc = 1;
        int x = 0;
        int y = 0;
        int z = 0;
        String[] a = new String[x];
        String[] b = new String[y];
        String[] c = new String[z];
        for (int i = 0; i < lstView.getCount(); i++) {
            if (persons.get(i).getAge() == "Not allowed") {
                a = new String[1+x];
                x++;
            }
            else if (persons.get(i).getAge() == "Read-Only") {
                b = new String[1+y];
                y++;
            }
            else {
                c = new String[1+z];
                z++;
            }
        }
        x = 0;
        y = 0;
```

```
        z = 0;
        try {
            //the file is readable by other applications (internal storage)
            FileOutputStream fOut = openFileOutput("confipicture.txt", MODE_PRIVATE);

            //convert character stream into a byte stream
            OutputStreamWriter osw = new OutputStreamWriter(fOut);
            osw.write(2*(a.length + b.length + c.length));

            for (int i = 0; i < lstView.getCount(); i++) {
                if (persons.get(i).getAge() == "Not allowed") {
                    a[x] = persons.get(i).getName();
                    osw.write(a[x] + "\n");
                    osw.write("Not allowed" + "\n");
                    x++;
                }
                else if (persons.get(i).getAge() == "Read-Only") {
                    b[y] = persons.get(i).getName();
                    osw.write(b[y] + "\n");
                    osw.write("Read-Only" + "\n");
                    y++;
                }
                else {
                    c[z] = persons.get(i).getName();
                    osw.write(c[z] + "\n");
                    osw.write("Read/Write" + "\n");
                    z++;
                }
            }
            //to ensure all the bytes are written to the file
            osw.flush();
            //close the file
            osw.close();
            //display file saved message
            Toast.makeText(getBaseContext(), "File saved successfully",
                                    Toast.LENGTH_SHORT).show();
        } catch (IOException ioe) {
            ioe.printStackTrace();
        }
        finish();
    }
});

lstView.setAdapter(new MyAdapter(this, persons));
lstView.setOnItemClickListener(new ListItemClick());
lstView.setOnItemLongClickListener(new ListItemLongClick());

File sdCard = Environment.getExternalStorageDirectory();
File directory = new File(sdCard.getAbsolutePath());
File[] folders = directory.listFiles();

persons.clear();

for (int i = 0; i < folders.length; i++) {
    if (folders[i].getName().contains(".jpg")) {
        person = new Person();
        person.setName(folders[i].getName());
        listArray.add(folders[i].getAbsolutePath());
        if (MainActivity.abc == 1) {
            for (int a = 0; a < strArray.length; a++) {
```

```
                if (strArray[a].contains(folders[i].getName()) &&
                    strArray[a+1].contains("Read-Only")) {
                   person.setAge("Read-Only");
                   a++;
                }
                else if (strArray[a].contains(folders[i].getName()) &&
                          strArray[a+1].contains("Read/Write")) {
                   person.setAge("Read/Write");
                   a++;
                }
                else if (strArray[a].contains(folders[i].getName()) &&
                          strArray[a+1].contains("Not allowed")) {
                   person.setAge("Not allowed");
                   a++;
                } else {}
             }
          }
          else {
             person.setAge("Read-Only");
          }
          persons.add(person);
       }
       File[] folderDirectory = folders[i].listFiles();
       if (folderDirectory != null) {
          if (folderDirectory.length > 0) {

    for (int j = 0; j < folderDirectory.length; j++) {
       if (folderDirectory[j].getName().contains(".jpg")) {
          person = new Person();
          person.setName(folderDirectory[j].getName());
          listArray.add(folderDirectory[j].getAbsolutePath());
          if (MainActivity.abc == 1) {
             for (int a = 0; a < strArray.length; a++) {
                if (strArray[a].contains(folderDirectory[j].getName()) &&
                    strArray[a+1].contains("Read-Only")) {
                   person.setAge("Read-Only");
                   a++;
                }
                else if (strArray[a].contains(folderDirectory[j].getName()) &&
                          strArray[a+1].contains("Read/Write"))
                {
                   person.setAge("Read/Write");
                   a++;
                }
                else if (strArray[a].contains(folderDirectory[j].getName()) &&
                          strArray[a+1].contains("Not allowed"))
                {
                   person.setAge("Not allowed");
                   a++;
                } else {}
             }
          }
          else {
             person.setAge("Read-Only");
          }
          persons.add(person);
       }
       File[] subFolderDirectory = folderDirectory[j].listFiles();
       if (subFolderDirectory != null) {
          if (subFolderDirectory.length > 0) {
```

```
for (int k = 0; k < subFolderDirectory.length; k++) {
   if (subFolderDirectory[k].getName().contains(".jpg")) {
      person = new Person();
      person.setName(subFolderDirectory[k].getName());
      listArray.add(subFolderDirectory[k].getAbsolutePath());
      if (MainActivity.abc == 1) {
         for (int a = 0; a < strArray.length; a++) {
            if (strArray[a].contains(subFolderDirectory[k].getName()) &&
                strArray[a+1].contains("Read-Only")) {
               person.setAge("Read-Only");
               a++;
            }
            else if (strArray[a].contains(subFolderDirectory[k].getName()) &&
                     strArray[a+1].contains("Read/Write")) {
               person.setAge("Read/Write");
               a++;
            }
            else if (strArray[a].contains(subFolderDirectory[k].getName()) &&
                     strArray[a+1].contains("Not allowed")) {
               person.setAge("Not allowed");
               a++;
            } else {}
         }
      }
      else {
         person.setAge("Read-Only");
      }
      persons.add(person);
   }
File[] subSubFolderDirectory = subFolderDirectory[k].listFiles();
   if (subSubFolderDirectory != null) {
      if (subSubFolderDirectory.length > 0) {

for (int l = 0; l < subSubFolderDirectory.length; l++) {
   if (subSubFolderDirectory[l].getName().contains(".jpg")) {
      person = new Person();
      person.setName(subSubFolderDirectory[l].getName());
      listArray.add(subSubFolderDirectory[l].getAbsolutePath());
      if (MainActivity.abc == 1) {
         for (int a = 0; a < strArray.length; a++) {
            if (strArray[a].contains(subSubFolderDirectory[l].getName()) &&
                strArray[a+1].contains("Read-Only"))
            {
               person.setAge("Read-Only");
               a++;
            }
            else if (strArray[a].contains(subSubFolderDirectory[l].getName()) &&
                     strArray[a+1].contains("Read/Write")) {
               person.setAge("Read/Write");
               a++;
            }
            else if (strArray[a].contains(subSubFolderDirectory[l].getName()) &&
                     strArray[a+1].contains("Not allowed")) {
               person.setAge("Not allowed");
               a++;
            } else {}
         }
      }
      else {
```

```
                person.setAge("Read-Only");
            }
            persons.add(person);
        }
        File[] subSubFolder1Directory = subSubFolderDirectory[l].listFiles();
        if (subSubFolder1Directory != null) {
            if (subSubFolder1Directory.length > 0) {

    for (int m = 0; m < subSubFolder1Directory.length; m++) {
        if (subSubFolder1Directory[m].getName().contains(".jpg")) {
            person = new Person();
            person.setName(subSubFolder1Directory[m].getName());
            listArray.add(subSubFolder1Directory[m].getAbsolutePath());
            if (MainActivity.abc == 1) {
                for (int a = 0; a < strArray.length; a++) {
                    if (strArray[a].contains(subSubFolder1Directory[m].getName()) &&
                         strArray[a+1].contains("Read-Only")) {
                        person.setAge("Read-Only");
                        a++;
                    }
                    else if (strArray[a].contains(subSubFolder1Directory[m].getName()) &&
                            strArray[a+1].contains("Read/Write")) {
                        person.setAge("Read/Write");
                        a++;
                    }
                    else if (strArray[a].contains(subSubFolder1Directory[m].getName()) &&
                            strArray[a+1].contains("Not allowed")) {
                        person.setAge("Not allowed");
                        a++;
                    } else {}
                }
            }
            else {
                person.setAge("Read-Only");
            }
            persons.add(person);
        }
    }
            } else{}
        } else{}
    }
            } else{}
        } else{}
    }
            } else{}
        } else{}
    }
            } else {}
        } else{}
    }
}

class Person {
    String name;
    String accessibility;

    public String getName() {
        return name;
    }
```

```
    public void setName(String name) {
        this.name = name;
    }

    public String getAge() {
        return accessibility;
    }

    public void setAge(String age) {
        this.accessibility = age;
    }

}

class MyAdapter extends BaseAdapter {

    private Context context;
    private ArrayList<Person> persons;

    public MyAdapter(Context context, ArrayList<Person> persons) {
        this.context = context;
        this.persons = persons;
    }

    @Override
    public int getCount() {
        return persons.size();
    }

    @Override
    public Object getItem(int position) {
        return persons.get(position);
    }

    @Override
    public long getItemId(int position) {
        return 0;
    }

    class ViewHolder {
        TextView text;
        ImageView icon;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        ViewHolder viewHolder;
        LayoutInflater li = (LayoutInflater)
                            context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);

        if (convertView==null){
            convertView = li.inflate(R.layout.main3, null);

            viewHolder = new ViewHolder();
            viewHolder.text= (TextView)convertView.findViewById(R.id.txtView);
            viewHolder.icon= (ImageView)convertView.findViewById(R.id.imgView);
            convertView.setTag(viewHolder);
        }
        else {
            viewHolder = (ViewHolder) convertView.getTag();
```

```
            }

            viewHolder.text.setText(persons.get(position).getName() + "\n" + "(" +
                                        persons.get(position).getAge() + ")");
            viewHolder.text.setTextColor(Color.GREEN);

            if (persons.get(position).getName().contains(".jpg")) {
               viewHolder.icon.setImageResource(R.drawable.jpg);
            }
            else {
               viewHolder.icon.setImageResource(R.drawable.folder);
            }
            return convertView;
         }
      }

      class ListItemClick implements OnItemClickListener {
         @Override
         public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
            if (persons.get(position).getAge() == "Not allowed") {
               persons.get(position).setAge("Read-Only");
               lstView.setAdapter(new MyAdapter(picture.this, persons));
            }
            else if (persons.get(position).getAge() == "Read-Only") {
               persons.get(position).setAge("Read/Write");
               lstView.setAdapter(new MyAdapter(picture.this, persons));
            }
            else {
               persons.get(position).setAge("Not allowed");
               lstView.setAdapter(new MyAdapter(picture.this, persons));
            }
         }
      }

      class ListItemLongClick implements OnItemLongClickListener {
         @Override
         public boolean onItemLongClick(AdapterView<?> parent, View view, int position, long id) {
            File file = new File(listArray.getItem(position));
            Uri url = Uri.fromFile(file);
            Intent intent = new Intent(Intent.ACTION_VIEW);
            // Check what kind of file you are trying to open, by comparing the url with extensions.
            // When the if condition is matched, plugin sets the correct intent (mime) type,
            // so Android knew what application to use to open the file
            if (file.toString().contains(".jpg") || file.toString().contains(".jpeg") ||
                 file.toString().contains(".png")) {
               // JPG file
               intent.setDataAndType(url, "image/jpeg");
            } else {
               //if you want you can also define the intent type for any other file

               //additionally use else clause below, to manage other unknown extensions
               //in this case, Android will show all applications installed on the device
               //so you can choose which application to use
               intent.setDataAndType(url, "*/*");
            }
            startActivity(intent);
            return false;
         }
      }
   }
```

copy.java

```
package com.example.example;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

import android.app.Activity;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Intent;
import android.os.Bundle;
import android.os.Environment;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

public class copy extends Activity {
   int RESULT_PASTEFROMREMOTE = 12;
   int RESULT_DRAG = 22;
   static int dragFromLocal;

   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.copy);
      Button btnCopy = (Button)findViewById(R.id.button1);
      btnCopy.setText("  Copy  ");
      btnCopy.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            if (lists.folderLabels[lists.pos].folderNames.contains(".pdf")
                  || lists.folderLabels[lists.pos].folderNames.contains(".txt")
                  || lists.folderLabels[lists.pos].folderNames.contains(".docx")
                  || lists.folderLabels[lists.pos].folderNames.contains(".xlsx")
                  || lists.folderLabels[lists.pos].folderNames.contains(".pptx")
                  || lists.folderLabels[lists.pos].folderNames.contains(".mp3")
                  || lists.folderLabels[lists.pos].folderNames.contains(".mp4")
                  || lists.folderLabels[lists.pos].folderNames.contains(".ogg")
                  || lists.folderLabels[lists.pos].folderNames.contains(".jpg")) {
               Toast.makeText(getBaseContext(), "copy", Toast.LENGTH_SHORT).show();
               nextpage.name = lists.folderLabels[lists.pos].folderNames;
               File fileName = new File(Environment.getExternalStorageDirectory().getAbsolutePath()
                           + nextpage.listClick1 + nextpage.listClick2 + nextpage.listClick3 +
                           nextpage.listClick4 + "/" + lists.folderLabels[lists.pos].folderNames);
               nextpage.fileByte = new byte[(int) fileName.length()];
               FileInputStream fileInputStream;
               try {
                  fileInputStream = new FileInputStream(fileName);
                  fileInputStream.read(nextpage.fileByte);
                  fileInputStream.close();
               } catch (IOException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
               }
            } else {
               Toast.makeText(getBaseContext(), "Unable to copy", Toast.LENGTH_SHORT).show();
```

```
        }
        finish();
    }
});

Button btnPaste = (Button)findViewById(R.id.button2);
btnPaste.setText("  Paste  ");
btnPaste.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (nextpage.name == null) {
            Toast.makeText(getBaseContext(), "No file is copied", Toast.LENGTH_SHORT).show();
        }
        else if (nextpage.name != null && nextpage.fileByte == null) {
            Toast.makeText(getBaseContext(), "paste file from remote to local",
                                Toast.LENGTH_SHORT).show();
            final CopyThread copyThread = new CopyThread(MainActivity.BDaddress,
                                                nextpage.name);
            copyThread.run();
            Intent i = new Intent();
            setResult(RESULT_PASTEFROMREMOTE, i);
        }
        else {
            Toast.makeText(getBaseContext(), "paste own file in local",
                                Toast.LENGTH_SHORT).show();
            File fileName = new File(Environment.getExternalStorageDirectory().getAbsolutePath()
                            + nextpage.listClick1 + nextpage.listClick2 + nextpage.listClick3 +
                             nextpage.listClick4 + "/" + nextpage.name);
            Toast.makeText(getBaseContext(), String.valueOf(nextpage.fileByte.length),
                                Toast.LENGTH_SHORT).show();
            FileOutputStream fileOutputStream;
            try {
                fileOutputStream = new FileOutputStream(fileName);
                fileOutputStream.write(nextpage.fileByte);
                fileOutputStream.close();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            Intent i = new Intent();
            setResult(RESULT_OK, i);
        }
        finish();
    }
});

Button btnDelete = (Button)findViewById(R.id.button3);
btnDelete.setText(" Delete  ");
btnDelete.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (lists.folderLabels[lists.pos].folderNames.contains(".pdf")
                || lists.folderLabels[lists.pos].folderNames.contains(".txt")
                || lists.folderLabels[lists.pos].folderNames.contains(".docx")
                || lists.folderLabels[lists.pos].folderNames.contains(".xlsx")
                || lists.folderLabels[lists.pos].folderNames.contains(".pptx")
                || lists.folderLabels[lists.pos].folderNames.contains(".mp3")
                || lists.folderLabels[lists.pos].folderNames.contains(".mp4")
                || lists.folderLabels[lists.pos].folderNames.contains(".ogg")
                || lists.folderLabels[lists.pos].folderNames.contains(".jpg")) {
```

```
            Toast.makeText(getBaseContext(), "delete", Toast.LENGTH_SHORT).show();
            File fileName = new File(Environment.getExternalStorageDirectory().getAbsolutePath()
                        + nextpage.listClick1 + nextpage.listClick2 + nextpage.listClick3 +
                        nextpage.listClick4 + "/" + lists.folderLabels[lists.pos].folderNames);
            fileName.delete();
            Intent i = new Intent();
            setResult(RESULT_OK, i);
          } else {
            Toast.makeText(getBaseContext(), "Unable to delete", Toast.LENGTH_SHORT).show();
          }
          finish();
        }
      });
}

private class CopyThread extends Thread {
    private BluetoothSocket mmSocket = null;
    private BluetoothDevice mmDevice = null;
    private OutputStream outStream = null;

    public CopyThread(String deviceAddress, String nameFile) {
        // Use a temporary object that is later assigned to mmSocket,
        // because mmSocket is final
        BluetoothSocket tmp = null;
        mmDevice = MainActivity.mBluetoothAdapter.getRemoteDevice(deviceAddress);
        // Get a BluetoothSocket to connect with the given BluetoothDevice
        try {
        // MY_UUID is the app's UUID string, also used by the server code
            tmp =
                mmDevice.createInsecureRfcommSocketToServiceRecord(MainActivity.MY_UUID);
            mmSocket = tmp;

            // Cancel discovery because it will slow down the connection
            MainActivity.mBluetoothAdapter.cancelDiscovery();

            try {
                // Connect the device through the socket. This will block
                // until it succeeds or throws an exception
                mmSocket.connect();

                try {
                    outStream = mmSocket.getOutputStream();

                    try {
                        outStream.write("copy file from remote to local".getBytes());
                        Toast.makeText(getBaseContext(), "File is copied from remote!",
                                        Toast.LENGTH_SHORT).show();

                        byte[] redundantRequestByte = new byte[50-
                                                "copy file from remote to local".length()];
                        outStream.write(redundantRequestByte);

                        outStream.write(nameFile.getBytes());

                        try {
                          outStream.flush();
                          outStream.close();
                          mmSocket.close();
                          tmp.close();
                        } catch (IOException e) {
```

```
                        e.printStackTrace();
                    }
                } catch (IOException e) {
                    e.printStackTrace();
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    } catch (IOException connectException) {}
  }
 }
}
```

copy1.java

```
package com.example.example;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

import android.app.Activity;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Intent;
import android.os.Bundle;
import android.os.Environment;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

public class copy1 extends Activity {
   int RESULT_PASTEFROMREMOTE = 12;

   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.paste);

      Button btnPaste = (Button)findViewById(R.id.button1);
      btnPaste.setText("  Paste  ");
      btnPaste.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            if (nextpage.name == null) {
               Toast.makeText(getBaseContext(), "No file is copied", Toast.LENGTH_SHORT).show();
            }
            else if (nextpage.name != null && nextpage.fileByte == null) {
               Toast.makeText(getBaseContext(), "paste file from remote to local",
                                 Toast.LENGTH_SHORT).show();
               final CopyThread copyThread = new CopyThread(MainActivity.BDaddress,
                                                   nextpage.name);
               copyThread.run();
               Intent i = new Intent();
               setResult(RESULT_PASTEFROMREMOTE, i);
            }
            else {
               Toast.makeText(getBaseContext(), "paste own file in local",
                                 Toast.LENGTH_SHORT).show();
               File fileName = new File(Environment.getExternalStorageDirectory().getAbsolutePath()
                              + nextpage.listClick1 + nextpage.listClick2 + nextpage.listClick3 +
                               nextpage.listClick4 + "/" + nextpage.name);
               FileOutputStream fileOutputStream;
               try {
                  fileOutputStream = new FileOutputStream(fileName);
                  fileOutputStream.write(nextpage.fileByte);
                  fileOutputStream.close();
               } catch (IOException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
               }
               Intent i = new Intent();
```

```
            setResult(RESULT_OK, i);
        }
        finish();
    }
  });

}

private class CopyThread extends Thread {
  private BluetoothSocket mmSocket = null;
  private BluetoothDevice mmDevice = null;
  private OutputStream outStream = null;

  public CopyThread(String deviceAddress, String nameFile) {
    // Use a temporary object that is later assigned to mmSocket,
    // because mmSocket is final
    BluetoothSocket tmp = null;
    mmDevice = MainActivity.mBluetoothAdapter.getRemoteDevice(deviceAddress);
    // Get a BluetoothSocket to connect with the given BluetoothDevice
    try {
    // MY_UUID is the app's UUID string, also used by the server code
      tmp =
          mmDevice.createInsecureRfcommSocketToServiceRecord(MainActivity.MY_UUID);
      mmSocket = tmp;

      // Cancel discovery because it will slow down the connection
      MainActivity.mBluetoothAdapter.cancelDiscovery();

      try {
        // Connect the device through the socket. This will block
        // until it succeeds or throws an exception
        mmSocket.connect();

        try {
          outStream = mmSocket.getOutputStream();

          try {
            outStream.write("copy file from remote to local".getBytes());
            Toast.makeText(getBaseContext(), "File is copied from remote!",
                          Toast.LENGTH_SHORT).show();

            byte[] redundantRequestByte = new byte[50-
                                        "copy file from remote to local".length()];
            outStream.write(redundantRequestByte);

            outStream.write(nameFile.getBytes());

            try {
              outStream.flush();
              outStream.close();
              mmSocket.close();
              tmp.close();
            } catch (IOException e) {
              e.printStackTrace();
            }
          } catch (IOException e) {
            e.printStackTrace();
          }
        } catch (IOException e) {
          e.printStackTrace();
```

```
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }  catch (IOException connectException) {}
    }
  }
}
```

paste.java

```
package com.example.example;

import java.io.IOException;
import java.io.OutputStream;

import android.app.Activity;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

public class paste extends Activity {
   int RESULT_PASTE = 15;
   int RESULT_DELETE = 25;
   int RESULT_DRAGFROMREMOTE = 14;
   static String dragFile;
   static int dragFromRemote;

   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.copy);

      Button btnCopy = (Button)findViewById(R.id.button1);
      btnCopy.setText("  Copy  ");
      btnCopy.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            int code = 0;
            if (listfiles.folderLabel[listfiles.posi].folderNames.contains(".pdf")
                  || listfiles.folderLabel[listfiles.posi].folderNames.contains(".txt")
                  || listfiles.folderLabel[listfiles.posi].folderNames.contains(".docx")
                  || listfiles.folderLabel[listfiles.posi].folderNames.contains(".xlsx")
                  || listfiles.folderLabel[listfiles.posi].folderNames.contains(".pptx")
                  || listfiles.folderLabel[listfiles.posi].folderNames.contains(".mp3")
                  || listfiles.folderLabel[listfiles.posi].folderNames.contains(".mp4")
                  || listfiles.folderLabel[listfiles.posi].folderNames.contains(".ogg")
                  || listfiles.folderLabel[listfiles.posi].folderNames.contains(".jpg")) {
            for (int i = 0; i < MainActivity.notAllowedSize; i++) {
            if (listfiles.folderLabel[listfiles.posi].folderNames.equalsIgnoreCase(
                MainActivity.notAllowedFileName[i])) {
                     Toast.makeText(getBaseContext(), "Not accessible",
                                       Toast.LENGTH_SHORT).show();
                     code = 10;
            } else {}
            }
               if (code == 0) {
                  Toast.makeText(getBaseContext(), "copy", Toast.LENGTH_SHORT).show();
                  nextpage.name = listfiles.folderLabel[listfiles.posi].folderNames;
                  nextpage.fileByte = null;
               } else {}
            } else {
               Toast.makeText(getBaseContext(), "Unable to copy", Toast.LENGTH_SHORT).show();
            }
            finish();
```

```
    }
});

Button btnPaste = (Button)findViewById(R.id.button2);
btnPaste.setText("  Paste  ");
btnPaste.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (nextpage.name == null) {
            Toast.makeText(getBaseContext(), "No file is copied", Toast.LENGTH_SHORT).show();
        }
        else {
            Toast.makeText(getBaseContext(), "paste", Toast.LENGTH_SHORT).show();
            final PasteThread pasteThread = new PasteThread(MainActivity.BDaddress,
                                                listfiles.folderLabel[0].folderNames);
            pasteThread.run();
            Intent i = new Intent();
            setResult(RESULT_PASTE, i);
        }
        finish();
    }
});

Button btnDelete = (Button)findViewById(R.id.button3);
btnDelete.setText(" Delete  ");
btnDelete.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        int code = 0;
        if (listfiles.folderLabel[listfiles.posi].folderNames.contains(".pdf")
                || listfiles.folderLabel[listfiles.posi].folderNames.contains(".txt")
                || listfiles.folderLabel[listfiles.posi].folderNames.contains(".docx")
                || listfiles.folderLabel[listfiles.posi].folderNames.contains(".xlsx")
                || listfiles.folderLabel[listfiles.posi].folderNames.contains(".pptx")
                || listfiles.folderLabel[listfiles.posi].folderNames.contains(".mp3")
                || listfiles.folderLabel[listfiles.posi].folderNames.contains(".mp4")
                || listfiles.folderLabel[listfiles.posi].folderNames.contains(".ogg")
                || listfiles.folderLabel[listfiles.posi].folderNames.contains(".jpg")) {
        for (int i = 0; i < MainActivity.notAllowedSize; i++) {
        if (listfiles.folderLabel[listfiles.posi].folderNames.equalsIgnoreCase(
            MainActivity.notAllowedFileName[i])) {
                Toast.makeText(getBaseContext(), "Not accessible",
                                    Toast.LENGTH_SHORT).show();
                code = 10;
        } else {}
        }
            for (int i = 0; i < MainActivity.readOnlySize; i++) {
         if (listfiles.folderLabel[listfiles.posi].folderNames.equalsIgnoreCase(
             MainActivity.readOnlyFileName[i])) {
                Toast.makeText(getBaseContext(), "Not accessible",
                                    Toast.LENGTH_SHORT).show();
                code = 20;
         } else {}
            }
            if (code == 0) {
                Toast.makeText(getBaseContext(), "delete", Toast.LENGTH_SHORT).show();
                final DeleteThread deleteThread = new DeleteThread(MainActivity.BDaddress,
                                                    listfiles.folderLabel[listfiles.posi].folderNames);
                deleteThread.run();
                Intent i = new Intent();
```

```
                setResult(RESULT_DELETE, i);
            } else {}
        } else {
            Toast.makeText(getBaseContext(), "Unable to delete", Toast.LENGTH_SHORT).show();
        }
        finish();
    }
  });
}

private class PasteThread extends Thread {
  private BluetoothSocket mmSocket = null;
  private BluetoothDevice mmDevice = null;
  private OutputStream outStream = null;

  public PasteThread(String deviceAddress, String nameFile) {
    // Use a temporary object that is later assigned to mmSocket,
    // because mmSocket is final
    BluetoothSocket tmp = null;
    mmDevice = MainActivity.mBluetoothAdapter.getRemoteDevice(deviceAddress);
    // Get a BluetoothSocket to connect with the given BluetoothDevice
    try {
    // MY_UUID is the app's UUID string, also used by the server code
      tmp =
          mmDevice.createInsecureRfcommSocketToServiceRecord(MainActivity.MY_UUID);
      mmSocket = tmp;

      // Cancel discovery because it will slow down the connection
      MainActivity.mBluetoothAdapter.cancelDiscovery();

      try {
        // Connect the device through the socket. This will block
        // until it succeeds or throws an exception
        mmSocket.connect();

        try {
          outStream = mmSocket.getOutputStream();

          try {
            if (nextpage.fileByte == null) {
              outStream.write("copy and paste file in remote".getBytes());
              Toast.makeText(getBaseContext(), "File is copied and pasted in remote!",
                          Toast.LENGTH_SHORT).show();

              byte[] redundantRequestByte = new byte[500-
                                                  "copy and paste file in remote".length()];
              outStream.write(redundantRequestByte);

              int size = nextpage.name.length();
              int i = 0;
              while (size > 100) {
                size = size - 100;
                i++;
              }

              outStream.write(size);
              outStream.write(i);
              outStream.write(nextpage.name.getBytes());

              size = nameFile.length();
```

```
    i = 0;
    while (size > 100) {
        size = size - 100;
        i++;
    }

    outStream.write(size);
    outStream.write(i);
    outStream.write(nameFile.getBytes());
}
else {
    outStream.write("paste file to remote".getBytes());
    Toast.makeText(getBaseContext(), "File is pasted to remote!",
                        Toast.LENGTH_SHORT).show();

    byte[] redundantRequestByte = new byte[500-"paste file to remote".length()];
    outStream.write(redundantRequestByte);

    int size = nameFile.length();
    int i = 0;

    while (size > 100) {
        size = size - 100;
        i++;
    }

    outStream.write(size);
    outStream.write(i);

    outStream.write(nameFile.getBytes());

    outStream.write(nextpage.name.length());
    outStream.write(nextpage.name.getBytes());

    byte[] redundantfileNameByte = new byte[497-nameFile.length()-
                                            nextpage.name.length()];
    outStream.write(redundantfileNameByte);

    size = nextpage.fileByte.length;
    i = 0;

    while (size > 100) {
        size = size - 100;
        i++;
    }

    int j = 0;
    while (i > 100) {
        i = i - 100;
        j++;
    }

    int k = 0;
    while (j > 100) {
        j = j - 100;
        k++;
    }

    outStream.write(size);
    outStream.write(i);
```

```
                    outStream.write(j);
                    outStream.write(k);

                    outStream.write(nextpage.fileByte);
                  }

                  try {
                    outStream.flush();
                    outStream.close();
                    mmSocket.close();
                    tmp.close();
                  } catch (IOException e) {
                    e.printStackTrace();
                  }
                } catch (IOException e) {
                  e.printStackTrace();
                }
              } catch (IOException e) {
                e.printStackTrace();
              }
            } catch (IOException e) {
              e.printStackTrace();
            }
          }  catch (IOException connectException) {}
        }
      }

      private class DeleteThread extends Thread {
        private BluetoothSocket mmSocket = null;
        private BluetoothDevice mmDevice = null;
        private OutputStream outStream = null;

        public DeleteThread(String deviceAddress, String nameFile) {
          // Use a temporary object that is later assigned to mmSocket,
          // because mmSocket is final
          BluetoothSocket tmp = null;
          mmDevice = MainActivity.mBluetoothAdapter.getRemoteDevice(deviceAddress);
          // Get a BluetoothSocket to connect with the given BluetoothDevice
          try {
          // MY_UUID is the app's UUID string, also used by the server code
            tmp =
                mmDevice.createInsecureRfcommSocketToServiceRecord(MainActivity.MY_UUID);
            mmSocket = tmp;

            // Cancel discovery because it will slow down the connection
            MainActivity.mBluetoothAdapter.cancelDiscovery();

            try {
              // Connect the device through the socket. This will block
              // until it succeeds or throws an exception
              mmSocket.connect();

              try {
                  outStream = mmSocket.getOutputStream();

                  try {
                    outStream.write("delete file to remote".getBytes());

                    byte[] redundantRequestByte = new byte[500-"delete file to remote".length()];
                    outStream.write(redundantRequestByte);
```

```
                        int size = nameFile.length();
                        int i = 0;

                        while (size > 100) {
                            size = size - 100;
                            i++;
                        }

                        outStream.write(size);
                        outStream.write(i);
                        outStream.write(nameFile.getBytes());

                        try {
                          outStream.flush();
                          outStream.close();
                          mmSocket.close();
                          tmp.close();
                        } catch (IOException e) {
                          e.printStackTrace();
                        }
                      } catch (IOException e) {
                         e.printStackTrace();
                      }
                    } catch (IOException e) {
                      e.printStackTrace();
                    }
                  } catch (IOException e) {
                     e.printStackTrace();
                  }
                }  catch (IOException connectException) {}
            }
        }
    }
```

paste1.java

```
package com.example.example;

import java.io.IOException;
import java.io.OutputStream;

import android.app.Activity;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

public class paste1 extends Activity {
   int RESULT_PASTE = 15;
   int RESULT_DELETE = 25;

   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.paste);

      Button btnPaste = (Button)findViewById(R.id.button1);
      btnPaste.setText("  Paste  ");
      btnPaste.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            if (nextpage.name == null) {
               Toast.makeText(getBaseContext(), "No file is copied", Toast.LENGTH_SHORT).show();
            }
            else {
               if (listfiles.folderLabel.length == 0) {
                  final PasteThread pasteThread = new PasteThread(MainActivity.BDaddress,
                                                   listfiles.listClicked);
                  pasteThread.run();
               }
               else {
                  final PasteThread pasteThread = new PasteThread(MainActivity.BDaddress,
                                                   listfiles.folderLabel[0].folderNames);
                  pasteThread.run();
               }
               Intent i = new Intent();
               setResult(RESULT_PASTE, i);
            }
            finish();
         }
      });
   }

   private class PasteThread extends Thread {
      private BluetoothSocket mmSocket = null;
      private BluetoothDevice mmDevice = null;
      private OutputStream outStream = null;

      public PasteThread(String deviceAddress, String nameFile) {
         // Use a temporary object that is later assigned to mmSocket,
         // because mmSocket is final
```

```
BluetoothSocket tmp = null;
mmDevice = MainActivity.mBluetoothAdapter.getRemoteDevice(deviceAddress);
// Get a BluetoothSocket to connect with the given BluetoothDevice
try {
// MY_UUID is the app's UUID string, also used by the server code
   tmp =
         mmDevice.createInsecureRfcommSocketToServiceRecord(MainActivity.MY_UUID);
   mmSocket = tmp;

   // Cancel discovery because it will slow down the connection
   MainActivity.mBluetoothAdapter.cancelDiscovery();

   try {
      // Connect the device through the socket. This will block
      // until it succeeds or throws an exception
      mmSocket.connect();

      try {
          outStream = mmSocket.getOutputStream();

          try {
             if (nextpage.fileByte == null && listfiles.folderLabel.length != 0) {
                outStream.write("copy and paste file in remote".getBytes());
                Toast.makeText(getBaseContext(), "File is copied and pasted in remote!",
                        Toast.LENGTH_SHORT).show();

                byte[] redundantRequestByte = new byte[500-
                                                "copy and paste file in remote".length()];
                outStream.write(redundantRequestByte);

                int size = nextpage.name.length();
                int i = 0;
                while (size > 100) {
                   size = size - 100;
                   i++;
                }
                outStream.write(size);
                outStream.write(i);
                outStream.write(nextpage.name.getBytes());

                size = nameFile.length();
                i = 0;
                while (size > 100) {
                   size = size - 100;
                   i++;
                }
                outStream.write(size);
                outStream.write(i);
                outStream.write(nameFile.getBytes());
             }
             else if (nextpage.fileByte == null && listfiles.folderLabel.length == 0) {
                outStream.write("copy file in remote and paste in remote".getBytes());
                Toast.makeText(getBaseContext(), "File is copied and pasted in remote!",
                                Toast.LENGTH_SHORT).show();

                byte[] redundantRequestByte = new byte[500-
                                                "copy file in remote and paste in remote".length()];
                outStream.write(redundantRequestByte);

                int size = nextpage.name.length();
```

```
    int i = 0;
    while (size > 100) {
       size = size - 100;
       i++;
    }

    outStream.write(size);
    outStream.write(i);
    outStream.write(nextpage.name.getBytes());

    size = nameFile.length();
    i = 0;
    while (size > 100) {
       size = size - 100;
       i++;
    }

    outStream.write(size);
    outStream.write(i);
    outStream.write(nameFile.getBytes());
 }
 else if (nextpage.fileByte != null && listfiles.folderLabel.length != 0){
    outStream.write("paste file to remote".getBytes());
    Toast.makeText(getBaseContext(), "File is pasted to remote!",
                          Toast.LENGTH_SHORT).show();

    byte[] redundantRequestByte = new byte[500-"paste file to remote".length()];
    outStream.write(redundantRequestByte);

    int size = nameFile.length();
    int i = 0;

    while (size > 100) {
       size = size - 100;
       i++;
    }

    outStream.write(size);
    outStream.write(i);

    outStream.write(nameFile.getBytes());

    outStream.write(nextpage.name.length());
    outStream.write(nextpage.name.getBytes());

    byte[] redundantfileNameByte = new byte[497-nameFile.length()-
                                            nextpage.name.length()];
    outStream.write(redundantfileNameByte);

    size = nextpage.fileByte.length;
    i = 0;

    while (size > 100) {
       size = size - 100;
       i++;
    }

    int j = 0;
    while (i > 100) {
       i = i - 100;
```

```
        j++;
    }

    int k = 0;
    while (j > 100) {
        j = j - 100;
        k++;
    }
    outStream.write(size);
    outStream.write(i);
    outStream.write(j);
    outStream.write(k);

    outStream.write(nextpage.fileByte);
}

else if (nextpage.fileByte != null && listfiles.folderLabel.length == 0){
    outStream.write("paste file to empty folder in remote".getBytes());
    Toast.makeText(getBaseContext(), "File is pasted to remote!",
                        Toast.LENGTH_SHORT).show();

    byte[] redundantRequestByte = new byte[500-
                                        "paste file to empty folder in remote".length()];
    outStream.write(redundantRequestByte);

    int size = nameFile.length();
    int i = 0;

    while (size > 100) {
        size = size - 100;
        i++;
    }
    outStream.write(size);
    outStream.write(i);

    outStream.write(nameFile.getBytes());

    outStream.write(nextpage.name.length());
    outStream.write(nextpage.name.getBytes());

    byte[] redundantfileNameByte = new byte[497-nameFile.length()-
                                        nextpage.name.length()];
    outStream.write(redundantfileNameByte);

    size = nextpage.fileByte.length;
    i = 0;

    while (size > 100) {
        size = size - 100;
        i++;
    }

    int j = 0;
    while (i > 100) {
        i = i - 100;
        j++;
    }

    int k = 0;
    while (j > 100) {
```

```
                        j = j - 100;
                        k++;
                      }

                      outStream.write(size);
                      outStream.write(i);
                      outStream.write(j);
                      outStream.write(k);

                      outStream.write(nextpage.fileByte);
                    } else {}

                    try {
                      outStream.flush();
                      outStream.close();
                      mmSocket.close();
                      tmp.close();
                    } catch (IOException e) {
                      e.printStackTrace();
                    }
                  } catch (IOException e) {
                    e.printStackTrace();
                  }
                } catch (IOException e) {
                  e.printStackTrace();
                }
              } catch (IOException e) {
                e.printStackTrace();
              }
            } catch (IOException connectException) {}
          }
        }
      }
```

SortApps.java

```
package com.example.example;

public class SortApps {
   public void exchange_sort(nextpage.Pac1[] pacs1){
       int i, j;
       nextpage.Pac1 temp;
       for ( i = 0;  i < pacs1.length - 1;  i++ )
       {
          for ( j = i + 1;  j < pacs1.length;  j++ )
          {
                if ( pacs1 [ i ].label.compareToIgnoreCase( pacs1 [ j ].label ) > 0 )
                 {                                    // ascending sort
                           temp = pacs1 [ i ];
                           pacs1 [ i ] = pacs1 [ j ];    // swapping
                           pacs1 [ j ] = temp;

                 }
            }
        }
   }

   public void exchange_sort(nextpage.Pac[] pacs){
       int i, j;
       nextpage.Pac temp;
      for ( i = 0;  i < pacs.length - 1;  i++ )
      {
         for ( j = i + 1;  j < pacs.length;  j++ )
         {
               if ( pacs [ i ].label.compareToIgnoreCase( pacs [ j ].label ) > 0 )
                {                                    // ascending sort
                          temp = pacs [ i ];
                          pacs [ i ] = pacs [ j ];    // swapping
                          pacs [ j ] = temp;

                }
           }
       }
   }
}
```

nextpage.java

```
package com.example.example;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.util.List;

import android.app.Activity;
import android.app.AlertDialog;
import android.app.AlertDialog.Builder;
import android.app.Dialog;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.BroadcastReceiver;
import android.content.ComponentName;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.pm.PackageManager;
import android.content.pm.ResolveInfo;
import android.graphics.Bitmap;
import android.graphics.Color;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import android.os.Environment;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnTouchListener;
import android.widget.AdapterView;
import android.widget.GridView;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.RelativeLayout;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.ViewFlipper;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.RelativeLayout.LayoutParams;

public class nextpage extends Activity {
    /** Called when the activity is first created. */

   static byte[] fileByte;
   static String name;
   static int order1, order2, order3, order4;
   static String listClick1, listClick2;
   static String listClick3, listClick4;
   static int orders1, orders2, orders3, orders4;
   static int pages = 0;
   static String title1, title2, title3, title4;
   static File[] folder, subFolder, subSubFolder, subSubFolder1, subSubFolder2;

   static int page = 0;
```

```
static int previousNum = 0;
static int previousNum1 = 0;
static int previousNum2 = 0;
static int previousNum3 = 0;

DrawerAdapter drawerAdapterObject;
DrawerAdapter1 drawerAdapterObject1;

static String directory;

class Pac{
   Drawable icon;
   String name;
   String packageName;
   String label;
}
Pac[] pacs, pacs0;

class Pac1{
   Bitmap icon;
   String name;
   int color;
   String label;
}

Pac1[] pacs1, pacs2;
PackageManager pm;

private LinearLayout linear;
private TextView txtView;
private GridView gridView;

static ViewFlipper viewFlipper;
private float lastX;
int i, j;

RelativeLayout homeView;
TextView screenText;
LinearLayout ll;

@Override
public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.screen);

   homeView = (RelativeLayout)findViewById(R.id.home);
   screenText = (TextView)findViewById(R.id.screentext);
   screenText.setText("Welcome to ScreenShare");

   viewFlipper = (ViewFlipper) findViewById(R.id.view_flipper);

   pm = getPackageManager();
   set_pacs();
   set_pacs1();

   IntentFilter filter1 = new IntentFilter();
   filter1.addAction(Intent.ACTION_PACKAGE_ADDED);
   filter1.addAction(Intent.ACTION_PACKAGE_REMOVED);
   filter1.addAction(Intent.ACTION_PACKAGE_CHANGED);
   filter1.addDataScheme("package");
```

```
  registerReceiver(new PacReceiver(), filter1);
}

public class PacReceiver extends BroadcastReceiver{
   @Override
   public void onReceive(Context context, Intent intent) {
      set_pacs();
   }
}

public void set_pacs() {
   final Intent mainIntent = new Intent(Intent.ACTION_MAIN, null);
   mainIntent.addCategory(Intent.CATEGORY_LAUNCHER);
   List<ResolveInfo> pacsList = pm.queryIntentActivities(mainIntent, 0);
   pacs = new Pac[pacsList.size()];
   for(int i=0 ; i < pacsList.size() ; i++){
      pacs[i]= new Pac();
      pacs[i].icon=pacsList.get(i).loadIcon(pm);
      pacs[i].packageName=pacsList.get(i).activityInfo.packageName;
      pacs[i].name=pacsList.get(i).activityInfo.name;
      pacs[i].label=pacsList.get(i).loadLabel(pm).toString();
   }
   new SortApps().exchange_sort(pacs);

   int pacsSize = pacsList.size();
     i = 0;
     while (pacsSize > 0) {
        if (pacsSize > 20) {
           pacs0 = new Pac[20];
           for (int I=0; I<20; I++) {
              pacs0[I] = pacs[I + i*20];
           }
        }
        else {
           pacs0 = new Pac[pacsSize];
           for (int I=0; I<pacsSize; I++) {
              pacs0[I] = pacs[I + i*20];
           }
        }
        linear = new LinearLayout(this);
        linear.setGravity(Gravity.CENTER_HORIZONTAL);
        linear.setOrientation(LinearLayout.VERTICAL);
        linear.setBackgroundResource(R.drawable.android);
        linear.setLayoutParams(new
              LinearLayout.LayoutParams(LinearLayout.LayoutParams.FILL_PARENT,
              LinearLayout.LayoutParams.FILL_PARENT));

        txtView = new TextView(this);
        txtView.setText("Local Screen " + String.valueOf(i+1));
        LinearLayout.LayoutParams params = new
                    LinearLayout.LayoutParams(LinearLayout.LayoutParams.WRAP_CONTENT,
                    LinearLayout.LayoutParams.WRAP_CONTENT);
        params.setMargins(0, 5, 0, 5);

        gridView = new GridView(this);
        LinearLayout.LayoutParams paramsGV = new
                    LinearLayout.LayoutParams(LinearLayout.LayoutParams.MATCH_PARENT,
                    LinearLayout.LayoutParams.MATCH_PARENT);
        paramsGV.setMargins(10, 0, 10, 10);
        gridView.setNumColumns(4);
```

```
            gridView.setGravity(Gravity.CENTER);
            gridView.setHorizontalSpacing(7);

            linear.addView(txtView, params);
            linear.addView(gridView, paramsGV);
            viewFlipper.addView(linear);

            drawerAdapterObject1 = new DrawerAdapter1(this, pacs0);
            gridView.setAdapter(drawerAdapterObject1);
            gridView.setOnItemClickListener(new DrawerItemClickListener());
            gridView.setOnTouchListener(new GridTouch());
            pacsSize = pacsSize - 20;
            i++;
        }
    }

    public void set_pacs1() {
        pacs1 = new Pac1[MainActivity.numOfApps];
        for(int j=0 ; j < MainActivity.numOfApps ; j++){
            pacs1[j] = new Pac1();
            pacs1[j].icon = MainActivity.bitmap[j];
            pacs1[j].label = MainActivity.sb[j];
            if (pacs1[j].label.contains("File")) {
                pacs1[j].color = Color.BLUE;
            }
            else {
                pacs1[j].color = Color.RED;
            }
        }
        new SortApps().exchange_sort(pacs1);

        int pacsSize1 = MainActivity.numOfApps;
          j = 0;
          while (pacsSize1 > 0) {
            if (pacsSize1 > 20) {
                pacs2 = new Pac1[20];
                for (int I=0; I<20; I++) {
                    pacs2[I] = pacs1[I + j*20];
                }
            }
            else {
                pacs2 = new Pac1[pacsSize1];
                for (int I=0; I<pacsSize1; I++) {
                    pacs2[I] = pacs1[I + j*20];
                }
            }
            linear = new LinearLayout(this);
            linear.setGravity(Gravity.CENTER_HORIZONTAL);
            linear.setOrientation(LinearLayout.VERTICAL);
            linear.setBackgroundResource(R.drawable.android);
            linear.setLayoutParams(new
                    LinearLayout.LayoutParams(LinearLayout.LayoutParams.FILL_PARENT,
                    LinearLayout.LayoutParams.FILL_PARENT));

            txtView = new TextView(this);
            txtView.setText("Remote Screen " + String.valueOf(j+1));
            LinearLayout.LayoutParams params = new
                        LinearLayout.LayoutParams(LinearLayout.LayoutParams.WRAP_CONTENT,
                        LinearLayout.LayoutParams.WRAP_CONTENT);
            params.setMargins(0, 5, 0, 5);
```

```
            gridView = new GridView(this);
            LinearLayout.LayoutParams paramsGV = new
                        LinearLayout.LayoutParams(LinearLayout.LayoutParams.MATCH_PARENT,
                        LinearLayout.LayoutParams.MATCH_PARENT);
            paramsGV.setMargins(10, 0, 10, 10);
            gridView.setNumColumns(4);
            gridView.setGravity(Gravity.CENTER);
            gridView.setHorizontalSpacing(7);

            linear.addView(txtView, params);
            linear.addView(gridView, paramsGV);
            viewFlipper.addView(linear);

            drawerAdapterObject = new DrawerAdapter(this, pacs2);
            gridView.setAdapter(drawerAdapterObject);
            gridView.setOnItemClickListener(new DrawerItemClicked());
            gridView.setOnTouchListener(new GridTouch());
            pacsSize1 = pacsSize1 - 20;
            j++;
        }
    }

    class DrawerItemClicked implements OnItemClickListener {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        // TODO Auto-generated method stub
            Toast.makeText(getBaseContext(), pacs1[position + 20*(viewFlipper.getDisplayedChild() –
                                i)].label + " is clicked", Toast.LENGTH_SHORT).show();
            if (pacs1[position + 20*(viewFlipper.getDisplayedChild() - i)].label.contains("File")) {
                directory = pacs1[position + 20*(viewFlipper.getDisplayedChild() - i)].label;
                page = -1;
                startActivity(new Intent("com.example.example.listfiles"));
            }
            else {
                  showDialog(0);
            }
        }
    }

    @Override
    protected Dialog onCreateDialog(int id) {
       switch (id) {
       case 0:
          Builder builder = new Builder(this, AlertDialog.THEME_HOLO_DARK);
          builder.setTitle("Error");
          builder.setCancelable(true);
          builder.setMessage("Unable to open the application");
          builder.setNeutralButton("OK", new DialogInterface.OnClickListener() {
             public void onClick(DialogInterface dialog, int which) {
                dialog.cancel();
             }
          });
          return builder.create();
       }
       return null;
    }

    class GridTouch implements OnTouchListener {
       public boolean onTouch(View v, MotionEvent me) {
```

```
        switch (me.getAction()) {
        // when user first touches the screen to swap
            case MotionEvent.ACTION_DOWN: {
                lastX = me.getX();
                break;
            }
            case MotionEvent.ACTION_UP: {
                float currentX = me.getX();
                // if left to right swipe on screen
                if (lastX - currentX < -100) {
                    // If no more View/Child to flip
                    if (viewFlipper.getDisplayedChild() == 0)
                        break;
                    // set the required Animation type to ViewFlipper
                    // The Next screen will come in form Left and current Screen will go OUT from Right
                    viewFlipper.setInAnimation(nextpage.this, R.anim.in_from_left);
                    viewFlipper.setOutAnimation(nextpage.this, R.anim.out_to_right);
                    // Show the next Screen
                    homeView.removeView(ll);
                    viewFlipper.showPrevious();
                }

                // if right to left swipe on screen
                if (lastX - currentX > 100) {
                    if (viewFlipper.getDisplayedChild() == i+j-1)
                        break;
                    // set the required Animation type to ViewFlipper
                    // The Next screen will come in form Right and current Screen will go OUT from Left
                    viewFlipper.setInAnimation(nextpage.this, R.anim.in_from_right);
                    viewFlipper.setOutAnimation(nextpage.this, R.anim.out_to_left);
                    // Show The Previous Screen
                    homeView.removeView(ll);
                    viewFlipper.showNext();
                }
                break;
            }
        }
        return false;
    }
}

@Override
public boolean onTouchEvent(MotionEvent touchevent) {
    switch (touchevent.getAction()) {
    // when user first touches the screen to swap
        case MotionEvent.ACTION_DOWN: {
            lastX = touchevent.getX();
            break;
        }
        case MotionEvent.ACTION_UP: {
            float currentX = touchevent.getX();
            // if left to right swipe on screen
            if (lastX < currentX) {
                // If no more View/Child to flip
                if (viewFlipper.getDisplayedChild() == 0)
                    break;
                // set the required Animation type to ViewFlipper
                // The Next screen will come in form Left and current Screen will go OUT from Right
                viewFlipper.setInAnimation(this, R.anim.in_from_left);
                viewFlipper.setOutAnimation(this, R.anim.out_to_right);
```

```
                // Show the next Screen
                viewFlipper.showPrevious();
            }

            // if right to left swipe on screen
            if (lastX > currentX) {
                if (viewFlipper.getDisplayedChild() == i+j-1)
                    break;
                // set the required Animation type to ViewFlipper
                // The Next screen will come in form Right and current Screen will go OUT from Left
                viewFlipper.setInAnimation(this, R.anim.in_from_right);
                viewFlipper.setOutAnimation(this, R.anim.out_to_left);
                // Show The Previous Screen
                viewFlipper.showNext();
            }
            break;
        }
    }
    return false;
}

public void onBackPressed() {
    Builder builder = new Builder(this, AlertDialog.THEME_HOLO_DARK);
    builder.setTitle("Exit ScreenShare ?");
    builder.setCancelable(true);
    builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int which) {
            dialog.cancel();
            finish();
        }
    });
    builder.setNegativeButton("NO", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int which) {
            dialog.cancel();
        }
    });
    builder.show();
}

public class DrawerItemClickListener implements OnItemClickListener {
    @Override
    public void onItemClick(AdapterView<?> arg0, View arg1, int pos, long arg3) {
        Toast.makeText(getBaseContext(), pacs[pos + 20*(viewFlipper.getDisplayedChild())].label
                + " is clicked", Toast.LENGTH_SHORT).show();
        if (!pacs[pos + 20*(viewFlipper.getDisplayedChild())].label.contains("File")) {
            Intent launchIntent = new Intent(Intent.ACTION_MAIN);
            launchIntent.addCategory(Intent.CATEGORY_LAUNCHER);
            ComponentName cp = new ComponentName(pacs[pos +
                                    20*(viewFlipper.getDisplayedChild())].packageName, pacs[pos +
                                    20*(viewFlipper.getDisplayedChild())].name);
            launchIntent.setComponent(cp);
            startActivity(launchIntent);
        }
        else {
            nextpage.pages = 0;
            startActivity(new Intent("com.example.example.lists"));
        }
    }
}
}
```

DrawerClickListener.java

```
package com.example.example;

import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;

public class DrawerClickListener implements OnItemClickListener {
   Context mContext;
   nextpage.Pac[] pacsForAdapter;

   public DrawerClickListener(Context c, nextpage.Pac[] pacs0){
      mContext=c;
      pacsForAdapter=pacs0;
   }

   @Override
   public void onItemClick(AdapterView<?> arg0, View arg1, int pos, long arg3) {
      if (!pacsForAdapter[pos].label.contains("File")) {
         Intent launchIntent = new Intent(Intent.ACTION_MAIN);
         launchIntent.addCategory(Intent.CATEGORY_LAUNCHER);
         ComponentName cp = new ComponentName(pacsForAdapter[pos].packageName
               , pacsForAdapter[pos].name);
         launchIntent.setComponent(cp);

         mContext.startActivity(launchIntent);
      }
      else {
         nextpage.pages = 0;
         mContext.startActivity(new Intent("com.example.example.lists"));
      }
   }
}
```

DrawerAdapter.java

```
package com.example.example;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.ImageView;
import android.widget.TextView;

public class DrawerAdapter extends BaseAdapter{
   Context mContext;
   nextpage.Pac1[] pacsForAdapter1;
   public DrawerAdapter (Context c, nextpage.Pac1 pacs1[]){
      mContext = c;
      pacsForAdapter1 = pacs1;
   }
   @Override
   public int getCount() {
      // TODO Auto-generated method stub
      return pacsForAdapter1.length;
   }
   @Override
   public Object getItem(int arg0) {
      // TODO Auto-generated method stub
      return null;
   }
   @Override
   public long getItemId(int arg0) {
      // TODO Auto-generated method stub
      return 0;
   }
   static class ViewHolder{
      TextView text;
      ImageView icon;
   }
   @Override
   public View getView(int pos, View convertView, ViewGroup arg2) {
      // TODO Auto-generated method stub
      ViewHolder viewHolder;
      LayoutInflater li = (LayoutInflater)
                           mContext.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
      if (convertView==null){
         convertView = li.inflate(R.layout.drawer_item, null);
         viewHolder = new ViewHolder();
         viewHolder.text= (TextView)convertView.findViewById(R.id.icon_text);
         viewHolder.icon= (ImageView)convertView.findViewById(R.id.icon_image);
         convertView.setTag(viewHolder);
      }
      else
         viewHolder = (ViewHolder) convertView.getTag();

      viewHolder.text.setText(pacsForAdapter1[pos].label + "\n");
      viewHolder.text.setTextColor(pacsForAdapter1[pos].color);
      viewHolder.icon.setImageBitmap(pacsForAdapter1[pos].icon);
      return convertView;
   }
}
```

DrawerAdapter1.java

```
package com.example.example;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.ImageView;
import android.widget.TextView;

public class DrawerAdapter1 extends BaseAdapter{
   Context mContext;
   nextpage.Pac[] pacsForAdapter;
   public DrawerAdapter1 (Context c, nextpage.Pac pacs[]){
      mContext = c;
      pacsForAdapter = pacs;
   }
   @Override
   public int getCount() {
      // TODO Auto-generated method stub
      return pacsForAdapter.length;
   }
   @Override
   public Object getItem(int arg0) {
      // TODO Auto-generated method stub
      return null;
   }
   @Override
   public long getItemId(int arg0) {
      // TODO Auto-generated method stub
      return 0;
   }
   static class ViewHolder{
      TextView text;
      ImageView icon;
   }
   @Override
   public View getView(int pos, View convertView, ViewGroup arg2) {
      // TODO Auto-generated method stub
      ViewHolder viewHolder;
      LayoutInflater li = (LayoutInflater)
                           mContext.getSystemService(Context.LAYOUT_INFLATER_SERVICE);

      if (convertView==null){
         convertView = li.inflate(R.layout.drawer_item, null);
         viewHolder = new ViewHolder();
         viewHolder.text= (TextView)convertView.findViewById(R.id.icon_text);
         viewHolder.icon= (ImageView)convertView.findViewById(R.id.icon_image);
         convertView.setTag(viewHolder);
      }
      else
         viewHolder = (ViewHolder) convertView.getTag();

      viewHolder.text.setText(pacsForAdapter[pos].label + "\n");
      viewHolder.icon.setImageDrawable(pacsForAdapter[pos].icon);
      return convertView;
   }
}
```

lists.java

```
package com.example.example;

import java.io.File;

import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.view.GestureDetector.OnGestureListener;
import android.view.GestureDetector;
import android.view.MotionEvent;
import android.view.View;
import android.view.animation.AnimationUtils;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemLongClickListener;
import android.widget.ArrayAdapter;
import android.widget.TextView;
import android.widget.ListView;
import android.widget.Toast;
import android.widget.AdapterView.OnItemClickListener;

public class lists extends Activity implements OnGestureListener {
    int RESULT_PASTEFROMREMOTE = 12;
    int RESULT_DRAG = 22;
    static int drag;

    TextView textBox;
    static ListView lstView1;
    static ArrayAdapter<String> mArrayPath;
    static int pos;
    String listClick;

    listfiles3 listIcon;
    class FileName{
        String folderNames;
    }
    static FileName[] folderLabels;

    private GestureDetector gestureScanner;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.list);

        gestureScanner = new GestureDetector(this);

        textBox = (TextView)findViewById(R.id.txtnum1);

        lstView1 = (ListView)findViewById(R.id.lstnum1);
        lstView1.setAnimation(AnimationUtils.loadAnimation(this, android.R.anim.slide_in_left));

        mArrayPath = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1);
        mArrayPath.clear();

        if (nextpage.pages == 0) {
```

```
    File sdCard = Environment.getExternalStorageDirectory();
    File directory = new File(sdCard.getAbsolutePath());
    nextpage.folder = directory.listFiles();
    folderLabels = new FileName[nextpage.folder.length];
    for (int i = 0; i < nextpage.folder.length; i++) {
       folderLabels[i] = new FileName();
       folderLabels[i].folderNames = nextpage.folder[i].getName();
    }
    mArrayPath.add(nextpage.folder[0].getParent());
    textBox.setText(nextpage.folder[0].getParent());
    listIcon = new listfiles3(this, folderLabels);
    lstView1.setAdapter(listIcon);
    lstView1.setOnItemClickListener(new ListItemClicked());
    lstView1.setOnItemLongClickListener(new ListItemLongClicked());
}

else if (nextpage.pages == 1) {
    nextpage.subFolder = nextpage.folder[nextpage.orders1].listFiles();
    mArrayPath.add(Environment.getExternalStorageDirectory().getAbsolutePath() +
                        nextpage.listClick1);
    textBox.setText(Environment.getExternalStorageDirectory().getAbsolutePath() +
                        nextpage.listClick1);
    folderLabels = new FileName[nextpage.subFolder.length];
    for (int i = 0; i < nextpage.subFolder.length; i++) {
       folderLabels[i] = new FileName();
       folderLabels[i].folderNames = nextpage.subFolder[i].getName();
    }
    listIcon = new listfiles3(this, folderLabels);
    lstView1.setAdapter(listIcon);
    lstView1.setOnItemClickListener(new ListItemClicked());
    lstView1.setOnItemLongClickListener(new ListItemLongClicked());
}

else if (nextpage.pages == 2) {
    nextpage.subSubFolder = nextpage.subFolder[nextpage.orders2].listFiles();
    mArrayPath.add(Environment.getExternalStorageDirectory().getAbsolutePath() +
                        nextpage.listClick1 + nextpage.listClick2);
    textBox.setText(Environment.getExternalStorageDirectory().getAbsolutePath() +
                        nextpage.listClick1 + nextpage.listClick2);
    folderLabels = new FileName[nextpage.subSubFolder.length];
    for (int i = 0; i < nextpage.subSubFolder.length; i++) {
       folderLabels[i] = new FileName();
       folderLabels[i].folderNames = nextpage.subSubFolder[i].getName();
    }
    listIcon = new listfiles3(this, folderLabels);
    lstView1.setAdapter(listIcon);
    lstView1.setOnItemClickListener(new ListItemClicked());
    lstView1.setOnItemLongClickListener(new ListItemLongClicked());
}

else if (nextpage.pages == 3) {
    nextpage.subSubFolder1 = nextpage.subSubFolder[nextpage.orders3].listFiles();
    mArrayPath.add(Environment.getExternalStorageDirectory().getAbsolutePath()
                        + nextpage.listClick1 + nextpage.listClick2 + nextpage.listClick3);
    textBox.setText(Environment.getExternalStorageDirectory().getAbsolutePath()
                       + nextpage.listClick1 + nextpage.listClick2 + nextpage.listClick3);
    folderLabels = new FileName[nextpage.subSubFolder1.length];
    for (int i = 0; i < nextpage.subSubFolder1.length; i++) {
       folderLabels[i] = new FileName();
       folderLabels[i].folderNames = nextpage.subSubFolder1[i].getName();
```

```
        }
        listIcon = new listfiles3(this, folderLabels);
        lstView1.setAdapter(listIcon);
        lstView1.setOnItemClickListener(new ListItemClicked());
        lstView1.setOnItemLongClickListener(new ListItemLongClicked());
    }

    else if (nextpage.pages == 4) {
        nextpage.subSubFolder2 = nextpage.subSubFolder1[nextpage.orders4].listFiles();
        mArrayPath.add(Environment.getExternalStorageDirectory().getAbsolutePath()
                + nextpage.listClick1 + nextpage.listClick2 + nextpage.listClick3 + nextpage.listClick4);
        textBox.setText(Environment.getExternalStorageDirectory().getAbsolutePath()
                + nextpage.listClick1 + nextpage.listClick2 + nextpage.listClick3 + nextpage.listClick4);
        folderLabels = new FileName[nextpage.subSubFolder2.length];
        for (int i = 0; i < nextpage.subSubFolder2.length; i++) {
            folderLabels[i] = new FileName();
            folderLabels[i].folderNames = nextpage.subSubFolder2[i].getName();
        }
        listIcon = new listfiles3(this, folderLabels);
        lstView1.setAdapter(listIcon);
        lstView1.setOnItemClickListener(new ListItemClicked());
        lstView1.setOnItemLongClickListener(new ListItemLongClicked());
    } else {}
}

class ListItemClicked implements OnItemClickListener {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        // TODO Auto-generated method stub
        listClick = folderLabels[position].folderNames;

        if (nextpage.pages == 0) {
            nextpage.listClick1 = "/" + folderLabels[position].folderNames;
            nextpage.listClick2 = "";
            nextpage.listClick3 = "";
            nextpage.listClick4 = "";
            nextpage.orders1 = position;
            nextpage.subFolder = null;
            nextpage.subFolder = nextpage.folder[nextpage.orders1].listFiles();
            if (nextpage.subFolder != null) {
                nextpage.pages++;
                finish();
                startActivity(new Intent("com.example.example.lists"));
            }
            else {
                Toast.makeText(getBaseContext(), "The file is not accessible.",
                                Toast.LENGTH_SHORT).show();
            }
        }
        else if (nextpage.pages == 1) {
            nextpage.listClick2 = "/" + folderLabels[position].folderNames;
            nextpage.listClick3 = "";
            nextpage.listClick4 = "";
            nextpage.orders2 = position;
            nextpage.subSubFolder = nextpage.subFolder[nextpage.orders2].listFiles();
            if (nextpage.subSubFolder != null) {
                nextpage.pages++;
                finish();
                startActivity(new Intent("com.example.example.lists"));
            }
```

```
    else {
        Toast.makeText(getBaseContext(), "The file is not accessible.",
                                Toast.LENGTH_SHORT).show();
    }
}
else if (nextpage.pages == 2) {
    nextpage.listClick3 = "/" + folderLabels[position].folderNames;
    nextpage.listClick4 = "";
    nextpage.orders3 = position;
    nextpage.subSubFolder1 = nextpage.subSubFolder[nextpage.orders3].listFiles();
    if (nextpage.subSubFolder1 != null) {
            nextpage.pages++;
            finish();
            startActivity(new Intent("com.example.example.lists"));
    }
    else {
        Toast.makeText(getBaseContext(), "The file is not accessible.",
                                Toast.LENGTH_SHORT).show();
    }
}
else if (nextpage.pages == 3) {
    nextpage.listClick4 = "/" + folderLabels[position].folderNames;
    nextpage.orders4 = position;
    nextpage.subSubFolder2 = nextpage.subSubFolder1[nextpage.orders4].listFiles();
    if (nextpage.subSubFolder2 != null) {
            nextpage.pages++;
            finish();
            startActivity(new Intent("com.example.example.lists"));
    }
    else {
        Toast.makeText(getBaseContext(), "The file is not accessible.",
                                Toast.LENGTH_SHORT).show();
    }
}
else {
    Toast.makeText(getBaseContext(), "The file is not accessible.(upto 5 level)",
                            Toast.LENGTH_SHORT).show();
}
if (listClick.contains(".doc") || listClick.contains(".docx")
        || listClick.contains(".pdf") || listClick.contains(".ppt")
        || listClick.contains(".pptx") || listClick.contains(".xls")
        || listClick.contains(".xlsx") || listClick.contains(".zip")
        || listClick.contains(".rar") || listClick.contains(".rtf")
        || listClick.contains(".wav") || listClick.contains(".mp3")
        || listClick.contains(".gif") || listClick.contains(".jpg")
        || listClick.contains(".jpeg") || listClick.contains(".png")
        || listClick.contains(".txt") || listClick.contains(".3gp")
        || listClick.contains(".mpg") || listClick.contains(".mpeg")
        || listClick.contains(".mpe") || listClick.contains(".mp4")
        || listClick.contains(".avi") || listClick.contains(".ogg")) {
    File file = new File(Environment.getExternalStorageDirectory().getAbsolutePath()
                + nextpage.listClick1 + nextpage.listClick2 + nextpage.listClick3 +
                 nextpage.listClick4);
    Uri url = Uri.fromFile(file);
    Intent intent = new Intent(Intent.ACTION_VIEW);
    // Check what kind of file you are trying to open, by comparing the url with extensions.
    // When the if condition is matched, plugin sets the correct intent (mime) type,
    // so Android knew what application to use to open the file
    if (file.toString().contains(".doc") || file.toString().contains(".docx")) {
        // Word document
```

```
            intent.setDataAndType(url, "application/msword");
         } else if(file.toString().contains(".pdf")) {
            // PDF file
            intent.setDataAndType(url, "application/pdf");
         } else if(file.toString().contains(".ppt") || file.toString().contains(".pptx")) {
            // Powerpoint file
            intent.setDataAndType(url, "application/vnd.ms-powerpoint");
         } else if(file.toString().contains(".xls") || file.toString().contains(".xlsx")) {
            // Excel file
            intent.setDataAndType(url, "application/vnd.ms-excel");
         } else if(file.toString().contains(".zip") || file.toString().contains(".rar")) {
            // WAV audio file
            intent.setDataAndType(url, "application/x-wav");
         } else if(file.toString().contains(".rtf")) {
            // RTF file
            intent.setDataAndType(url, "application/rtf");
         } else if(file.toString().contains(".wav") || file.toString().contains(".mp3") ||
               file.toString().contains(".ogg")) {
            // WAV audio file
            intent.setDataAndType(url, "audio/x-wav");
         } else if(file.toString().contains(".gif")) {
            // GIF file
            intent.setDataAndType(url, "image/gif");
         } else if(file.toString().contains(".jpg") || file.toString().contains(".jpeg") ||
               file.toString().contains(".png")) {
            // JPG file
            intent.setDataAndType(url, "image/jpeg");
         } else if(file.toString().contains(".txt")) {
            // Text file
            intent.setDataAndType(url, "text/plain");
         } else if(file.toString().contains(".3gp") || file.toString().contains(".mpg") ||
               file.toString().contains(".mpeg")
              || file.toString().contains(".mpe") || file.toString().contains(".mp4") ||
               file.toString().contains(".avi")) {
            // Video files
            intent.setDataAndType(url, "video/*");
         } else {
            //if you want you can also define the intent type for any other file
            //additionally use else clause below, to manage other unknown extensions
            //in this case, Android will show all applications installed on the device
            //so you can choose which application to use
            intent.setDataAndType(url, "*/*");
         }
         startActivity(intent);
      } else {}
   }
}

class ListItemLongClicked implements OnItemLongClickListener {
   @Override
   public boolean onItemLongClick(AdapterView<?> parent, View view, int position, long id) {
      pos = position;
      startActivityForResult(new Intent("com.example.example.copy"), 100);
      return false;
   }
}

public void onActivityResult(int requestCode, int resultCode, Intent data) {
   if (requestCode == 100) {
      drag = 0;
```

```
        if (resultCode == RESULT_OK) {
            final ProgressDialog dialog = ProgressDialog.show(
                    lists.this, "Refreshing", "Please wait...", true);
                new Thread(new Runnable(){
                    public void run(){
                        try {
                            Thread.sleep(5000);
                            finish();
                            startActivity(new Intent("com.example.example.lists"));
                            dialog.dismiss();
                        } catch (InterruptedException e) {
                            e.printStackTrace();
                        }
                    }
                }).start();
        }
        else if (resultCode == RESULT_PASTEFROMREMOTE) {
            MainActivity.done = 0;
            final ProgressDialog dialog = ProgressDialog.show(
                    lists.this, "Refreshing", "Please wait...", true);
                new Thread(new Runnable(){
                    public void run(){
                        try {
                            while (MainActivity.done == 0) {
                                Thread.sleep(1000);
                            }
                            finish();
                            startActivity(new Intent("com.example.example.lists"));
                            dialog.dismiss();
                        } catch (InterruptedException e) {
                            e.printStackTrace();
                        }
                    }
                }).start();
        }
        else if (resultCode == RESULT_DRAG) {
            drag = 1;
            finish();
            startActivity(new Intent("com.example.example.nextpage"));
        }
    }
}

@Override
public void onBackPressed() {
    --nextpage.pages;
    finish();
    if (nextpage.pages > -1) {
        startActivity(new Intent("com.example.example.lists"));
    }
}

@Override
public boolean onTouchEvent(MotionEvent me) {
    return gestureScanner.onTouchEvent(me);
}
@Override
public boolean onDown(MotionEvent e) {
    return true;
}
```

```
    @Override
    public void onLongPress(MotionEvent e) {
        startActivityForResult(new Intent("com.example.example.copy1"), 100);
    }
    @Override
    public boolean onScroll(MotionEvent e1, MotionEvent e2, float distanceX, float distanceY) {
        return true;
    }
    @Override
    public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX, float velocityY) {
        return true;
    }
    @Override
    public void onShowPress(MotionEvent e) {}
    @Override
    public boolean onSingleTapUp(MotionEvent e) {
        return true;
    }
}
```

listfiles.java

```
package com.example.example;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.OutputStream;

import android.net.Uri;
import android.os.Bundle;
import android.app.Activity;
import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.app.AlertDialog.Builder;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.DialogInterface;
import android.content.Intent;
import android.view.GestureDetector;
import android.view.GestureDetector.OnGestureListener;
import android.view.MotionEvent;
import android.view.animation.AnimationUtils;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemLongClickListener;
import android.widget.Button;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.AdapterView.OnItemClickListener;

public class listfiles extends Activity implements OnGestureListener {
    int RESULT_PASTE = 15;
    int RESULT_DELETE = 25;
    int RESULT_DRAGFROMREMOTE = 14;

    TextView txtView;
    ListView lstView;
    static String listClicked;
    int listNo;
    listfiles2 listIcon;
    class Names{
        String folderNames;
        String color;
    }
    static Names[] folderLabel;
    File[] filetitle;

    static int posi;

    private GestureDetector gestureScanner;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main2);

        gestureScanner = new GestureDetector(this);
```

```
txtView = (TextView)findViewById(R.id.txtview);
lstView = (ListView)findViewById(R.id.lstview);
lstView.setAnimation(AnimationUtils.loadAnimation(this, android.R.anim.slide_in_left));

Button btnSynchronize = (Button)findViewById(R.id.btn_synchronize);
btnSynchronize.setText("Synchronize");
btnSynchronize.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View v) {
      int codes = 0;
      int sum = 0;
      if (MainActivity.file[0] != null) {
         for (int i = 0; i < MainActivity.readWriteSize; i ++) {
            for (int j = 0; j < MainActivity.file.length; j++) {
               if (MainActivity.file[j] == null) {}
               else if (MainActivity.readWriteFileName[i].contains(MainActivity.file[j].getName())
                     && MainActivity.date[j] != MainActivity.file[j].lastModified()) {
                  sum++;
               } else {}
            }
         }
         filetitle = new File[sum];
         sum = 0;
         for (int i = 0; i < MainActivity.readWriteSize; i ++) {
            for (int j = 0; j < MainActivity.file.length; j++) {
               if (MainActivity.file[j] == null) {}
               else if (MainActivity.readWriteFileName[i].contains(MainActivity.file[j].getName())
                     && MainActivity.date[j] != MainActivity.file[j].lastModified()) {
                  filetitle[sum] = MainActivity.file[j];
                  sum++;
                  codes = 10;
               } else {}
            }
         }
         if (codes == 10) {
            final SavedBackThread savedBackThread = new
                                       SavedBackThread(MainActivity.BDaddress, filetitle);
            savedBackThread.run();
            MainActivity.num = 0;
            MainActivity.file = new File[MainActivity.readOnlySize +
                                 MainActivity.readWriteSize];;
         } else {}
      }
      else {
         Builder builder = new Builder(listfiles.this, AlertDialog.THEME_HOLO_DARK);
         builder.setTitle("Error");
         builder.setCancelable(true);
         builder.setMessage("No read/write file is modified");
         builder.setNeutralButton("OK", new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int which) {
               dialog.cancel();
            }
         });
         builder.show();
      }
   }
});

if (nextpage.page == -1) {
```

```
    nextpage.page = 0;
    Builder builder = new Builder(listfiles.this, AlertDialog.THEME_HOLO_DARK);
    builder.setTitle("Color for the file accessibility");
    builder.setCancelable(true);
    builder.setMessage("RED        - not accessible files \n"
                + "PURPLE  - read-only files \n"
                + "BLUE      - read/write files \n"
                + "BLACK    - non-file types");
    builder.setNeutralButton("OK", new DialogInterface.OnClickListener() {
       public void onClick(DialogInterface dialog, int which) {
          dialog.cancel();
       }
    });
    builder.show();
}

if (nextpage.page == 0) {
    txtView.setText(nextpage.directory);
    listNo = MainActivity.numOfFolders;
    folderLabel = new Names[listNo];
    for (int i = 0; i < MainActivity.numOfFolders; i++) {
       folderLabel[i] = new Names();
       folderLabel[i].folderNames = MainActivity.folderName[i];
       for(int x = 0; x < MainActivity.notAllowedSize; x++) {
          if (folderLabel[i].folderNames.equalsIgnoreCase(MainActivity.notAllowedFileName[x]))
          {
             folderLabel[i].color = "RED";
          } else {}
       }

       for(int y = 0; y < MainActivity.readOnlySize; y++) {
          if (folderLabel[i].folderNames.equalsIgnoreCase(MainActivity.readOnlyFileName[y])) {
             folderLabel[i].color = "PURPLE";
          } else {}
       }

       for(int z = 0; z < MainActivity.readWriteSize; z++) {
          if (folderLabel[i].folderNames.equalsIgnoreCase(MainActivity.readWriteFileName[z])) {
             folderLabel[i].color = "BLUE";
          } else {}
       }
    }
    listIcon = new listfiles2(this, folderLabel);
    lstView.setAdapter(listIcon);
    lstView.setOnItemClickListener(new ListItemClicked());
    lstView.setOnItemLongClickListener(new ListItemLongClicked());
}

else if (nextpage.page == 1) {
    txtView.setText(nextpage.title1);
    listNo = MainActivity.numOfSubFolder[nextpage.order1];
    folderLabel = new Names[listNo];
    nextpage.previousNum = 0;
    for (int i = 0; i < nextpage.order1; i++) {
       if (MainActivity.numOfSubFolder[i] == -1) {
          nextpage.previousNum = nextpage.previousNum + 0;
       }
       else {
          nextpage.previousNum = nextpage.previousNum + MainActivity.numOfSubFolder[i];
       }
```

```
    }
    for (int j = 0; j < listNo; j++) {
        folderLabel[j] = new Names();
        folderLabel[j].folderNames = MainActivity.subFolderName[nextpage.previousNum + j];
        for(int x = 0; x < MainActivity.notAllowedSize; x++) {
            if (folderLabel[j].folderNames.equalsIgnoreCase(MainActivity.notAllowedFileName[x]))
            {
                folderLabel[j].color = "RED";
            } else {}
        }

        for(int y = 0; y < MainActivity.readOnlySize; y++) {
            if (folderLabel[j].folderNames.equalsIgnoreCase(MainActivity.readOnlyFileName[y])) {
                folderLabel[j].color = "PURPLE";
            } else {}
        }

        for(int z = 0; z < MainActivity.readWriteSize; z++) {
            if (folderLabel[j].folderNames.equalsIgnoreCase(MainActivity.readWriteFileName[z])) {
                folderLabel[j].color = "BLUE";
            } else {}
        }
    }
    listIcon = new listfiles2(this, folderLabel);
    lstView.setAdapter(listIcon);
    lstView.setOnItemClickListener(new ListItemClicked());
    lstView.setOnItemLongClickListener(new ListItemLongClicked());
}

else if (nextpage.page == 2) {
    txtView.setText(nextpage.title2);
    listNo = MainActivity.numOfSubSubFolder[nextpage.previousNum + nextpage.order2];
    folderLabel = new Names[listNo];
    nextpage.previousNum1 = 0;
    for (int i = 0; i < (nextpage.previousNum + nextpage.order2); i++) {
        if (MainActivity.numOfSubSubFolder[i] == -1) {
            nextpage.previousNum1 = nextpage.previousNum1 + 0;
        }
        else {
            nextpage.previousNum1 = nextpage.previousNum1 +
                                    MainActivity.numOfSubSubFolder[i];
        }
    }
    for (int j = 0; j < listNo; j++) {
        folderLabel[j] = new Names();
        folderLabel[j].folderNames = MainActivity.subSubFolderName[nextpage.previousNum1 +
                                     j];
        for(int x = 0; x < MainActivity.notAllowedSize; x++) {
            if (folderLabel[j].folderNames.equalsIgnoreCase(MainActivity.notAllowedFileName[x]))
            {
                folderLabel[j].color = "RED";
            } else {}
        }

        for(int y = 0; y < MainActivity.readOnlySize; y++) {
            if (folderLabel[j].folderNames.equalsIgnoreCase(MainActivity.readOnlyFileName[y])) {
                folderLabel[j].color = "PURPLE";
            } else {}
        }
```

```
        for(int z = 0; z < MainActivity.readWriteSize; z++) {
            if (folderLabel[j].folderNames.equalsIgnoreCase(MainActivity.readWriteFileName[z])) {
                folderLabel[j].color = "BLUE";
            } else {}
        }
    }
    listIcon = new listfiles2(this, folderLabel);
    lstView.setAdapter(listIcon);
    lstView.setOnItemClickListener(new ListItemClicked());
    lstView.setOnItemLongClickListener(new ListItemLongClicked());
}

else if (nextpage.page == 3) {
    txtView.setText(nextpage.title3);
    listNo = MainActivity.numOfSubSubFolder1[nextpage.previousNum1 + nextpage.order3];
    folderLabel = new Names[listNo];
    nextpage.previousNum2 = 0;
    for (int i = 0; i < (nextpage.previousNum1 + nextpage.order3); i++) {
        if (MainActivity.numOfSubSubFolder1[i] == -1) {
            nextpage.previousNum2 = nextpage.previousNum2 + 0;
        }
        else {
            nextpage.previousNum2 = nextpage.previousNum2 +
                                    MainActivity.numOfSubSubFolder1[i];
        }
    }
    for (int j = 0; j < listNo; j++) {
        folderLabel[j] = new Names();
        folderLabel[j].folderNames = MainActivity.subSubFolder1Name[nextpage.previousNum2
                                     + j];
        for(int x = 0; x < MainActivity.notAllowedSize; x++) {
            if (folderLabel[j].folderNames.equalsIgnoreCase(MainActivity.notAllowedFileName[x]))
            {
                folderLabel[j].color = "RED";
            } else {}
        }

        for(int y = 0; y < MainActivity.readOnlySize; y++) {
            if (folderLabel[j].folderNames.equalsIgnoreCase(MainActivity.readOnlyFileName[y])) {
                folderLabel[j].color = "PURPLE";
            } else {}
        }

        for(int z = 0; z < MainActivity.readWriteSize; z++) {
            if (folderLabel[j].folderNames.equalsIgnoreCase(MainActivity.readWriteFileName[z])) {
                folderLabel[j].color = "BLUE";
            } else {}
        }
    }
    listIcon = new listfiles2(this, folderLabel);
    lstView.setAdapter(listIcon);
    lstView.setOnItemClickListener(new ListItemClicked());
    lstView.setOnItemLongClickListener(new ListItemLongClicked());
}

else if (nextpage.page == 4) {
    txtView.setText(nextpage.title4);
    listNo = MainActivity.numOfSubSubFolder2[nextpage.previousNum2 + nextpage.order4];
    folderLabel = new Names[listNo];
    nextpage.previousNum3 = 0;
```

```
        for (int i = 0; i < (nextpage.previousNum2 + nextpage.order4); i++) {
            if (MainActivity.numOfSubSubFolder2[i] == -1) {
                nextpage.previousNum3 = nextpage.previousNum3 + 0;
            }
            else {
                nextpage.previousNum3 = nextpage.previousNum3 +
                                        MainActivity.numOfSubSubFolder2[i];
            }
        }
        for (int j = 0; j < listNo; j++) {
            folderLabel[j] = new Names();
            folderLabel[j].folderNames = MainActivity.subSubFolder2Name[nextpage.previousNum3
                                         + j];
            for(int x = 0; x < MainActivity.notAllowedSize; x++) {
                if (folderLabel[j].folderNames.equalsIgnoreCase(MainActivity.notAllowedFileName[x]))
                {
                    folderLabel[j].color = "RED";
                } else {}
            }

            for(int y = 0; y < MainActivity.readOnlySize; y++) {
                if (folderLabel[j].folderNames.equalsIgnoreCase(MainActivity.readOnlyFileName[y])) {
                    folderLabel[j].color = "PURPLE";
                } else {}
            }

            for(int z = 0; z < MainActivity.readWriteSize; z++) {
                if (folderLabel[j].folderNames.equalsIgnoreCase(MainActivity.readWriteFileName[z])) {
                    folderLabel[j].color = "BLUE";
                } else {}
            }
        }
        listIcon = new listfiles2(this, folderLabel);
        lstView.setAdapter(listIcon);
        lstView.setOnItemClickListener(new ListItemClicked());
        lstView.setOnItemLongClickListener(new ListItemLongClicked());
    } else {}

class ListItemClicked implements OnItemClickListener {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        // TODO Auto-generated method stub
        listClicked = null;

        if (nextpage.page == 0) {
            listClicked = folderLabel[position].folderNames;
            if (MainActivity.numOfSubFolder[position] >= 0) {
                nextpage.page++;
                nextpage.order1 = position;
                nextpage.title1 = nextpage.directory + "/" + listClicked;
                finish();
                startActivity(new Intent("com.example.example.listfiles"));
            }
            else if (listClicked.contains(".pdf") || listClicked.contains(".txt")
                    || listClicked.contains(".docx") || listClicked.contains(".xlsx")
                    || listClicked.contains(".mp3") || listClicked.contains(".mp4")
                    || listClicked.contains(".pptx") || listClicked.contains(".jpg")
                    || listClicked.contains(".ogg")) {
                int code = 0;
                int j = 0;
```

```
   for (int i = 0; i < MainActivity.notAllowedSize; i++) {
      if (listClicked.equalsIgnoreCase(MainActivity.notAllowedFileName[i])) {
         Toast.makeText(getBaseContext(), "Not accessible",
                              Toast.LENGTH_SHORT).show();
         code = 10;
      } else {}
   }
   if (code == 0) {
      for (int i = 0 ; i < MainActivity.file.length; i++) {
         if (MainActivity.file[i] == null) {}
         else if (listClicked.contains(MainActivity.file[i].getName())) {
            code = 20;
            j = i;
         } else {}
      }
   } else {}

   if (code == 20) {
      Uri uri = Uri.fromFile(MainActivity.file[j]);
      Intent intent = new Intent(Intent.ACTION_VIEW);
      if (MainActivity.file[j].toString().contains(".docx")) {
         // Word document
         intent.setDataAndType(uri, "application/msword");
      } else if(MainActivity.file[j].toString().contains(".pdf")) {
         // PDF file
         intent.setDataAndType(uri, "application/pdf");
      } else if(MainActivity.file[j].toString().contains(".pptx")) {
         // Powerpoint file
         intent.setDataAndType(uri, "application/vnd.ms-powerpoint");
      } else if(MainActivity.file[j].toString().contains(".xlsx")) {
         // Excel file
         intent.setDataAndType(uri, "application/vnd.ms-excel");
      } else if(MainActivity.file[j].toString().contains(".mp3") ||
            MainActivity.file[j].toString().contains(".ogg")) {
         // WAV audio file
         intent.setDataAndType(uri, "audio/x-wav");
      } else if(MainActivity.file[j].toString().contains(".mp4")) {
         // Video files
         intent.setDataAndType(uri, "video/*");
      } else if(MainActivity.file[j].toString().contains(".jpg")) {
         // JPG file
         intent.setDataAndType(uri, "image/jpeg");
      } else if(MainActivity.file[j].toString().contains(".txt")) {
         // Text file
         intent.setDataAndType(uri, "text/plain");
      } else {}
      startActivity(intent);
   } else if (code == 10) {}
   else {
      final RequestThread requestThread = new RequestThread(MainActivity.requestDevice,
                                          listClicked);
      requestThread.run();
   }
}
else if (listClicked.contains(".xml") || listClicked.contains(".m4a") ||
         listClicked.contains(".png")) {
   Toast.makeText(getBaseContext(), "Cannot access", Toast.LENGTH_SHORT).show();
}
else {
```

```
                Toast.makeText(getBaseContext(), "The folder is empty",
                                    Toast.LENGTH_SHORT).show();
        }
    }

    else if (nextpage.page == 1) {
        listClicked = folderLabel[position].folderNames;
        if (MainActivity.numOfSubSubFolder[nextpage.previousNum + position] >= 0) {
            nextpage.page++;
            nextpage.order2 = position;
            nextpage.title2 = nextpage.title1 + "/" + listClicked;
            finish();
            startActivity(new Intent("com.example.example.listfiles"));
        }
        else if (listClicked.contains(".pdf") || listClicked.contains(".txt")
                || listClicked.contains(".docx") || listClicked.contains(".xlsx")
                || listClicked.contains(".mp3") || listClicked.contains(".mp4")
                || listClicked.contains(".pptx") || listClicked.contains(".jpg")
                || listClicked.contains(".ogg")) {
            int code = 0;
            int j = 0;
            for (int i = 0; i < MainActivity.notAllowedSize; i++) {
                if (listClicked.equalsIgnoreCase(MainActivity.notAllowedFileName[i])) {
                    Toast.makeText(getBaseContext(), "Not accessible",
                                        Toast.LENGTH_SHORT).show();
                    code = 10;
                } else {}
            }
            if (code == 0) {
                for (int i = 0 ; i < MainActivity.file.length; i++) {
                    if (MainActivity.file[i] == null) {}
                    else if (listClicked.contains(MainActivity.file[i].getName())) {
                        code = 20;
                        j = i;
                    } else {}
                }
            } else {}

            if (code == 20) {
                Uri uri = Uri.fromFile(MainActivity.file[j]);
                Intent intent = new Intent(Intent.ACTION_VIEW);
                if (MainActivity.file[j].toString().contains(".docx")) {
                    // Word document
                    intent.setDataAndType(uri, "application/msword");
                } else if(MainActivity.file[j].toString().contains(".pdf")) {
                    // PDF file
                    intent.setDataAndType(uri, "application/pdf");
                } else if(MainActivity.file[j].toString().contains(".pptx")) {
                    // Powerpoint file
                    intent.setDataAndType(uri, "application/vnd.ms-powerpoint");
                } else if(MainActivity.file[j].toString().contains(".xlsx")) {
                    // Excel file
                    intent.setDataAndType(uri, "application/vnd.ms-excel");
                } else if(MainActivity.file[j].toString().contains(".mp3") ||
                        MainActivity.file[j].toString().contains(".ogg")) {
                    // WAV audio file
                    intent.setDataAndType(uri, "audio/x-wav");
                } else if(MainActivity.file[j].toString().contains(".jpg")) {
                    // JPG file
                    intent.setDataAndType(uri, "image/jpeg");
```

```
            } else if(MainActivity.file[j].toString().contains(".mp4")) {
                // Video files
                intent.setDataAndType(uri, "video/*");
            } else if(MainActivity.file[j].toString().contains(".txt")) {
                // Text file
                intent.setDataAndType(uri, "text/plain");
            } else {}
            startActivity(intent);
        } else if (code == 10) {}
        else {
            final RequestThread requestThread = new RequestThread(MainActivity.requestDevice,
                                                    listClicked);
            requestThread.run();
        }
    }
    else if (listClicked.contains(".xml") || listClicked.contains(".m4a") ||
             listClicked.contains(".png")) {
        Toast.makeText(getBaseContext(), "Cannot access", Toast.LENGTH_SHORT).show();
    }
    else {
        Toast.makeText(getBaseContext(), "The folder is empty",
                       Toast.LENGTH_SHORT).show();
    }
}

else if (nextpage.page == 2) {
    listClicked = folderLabel[position].folderNames;
    if (MainActivity.numOfSubSubFolder1[nextpage.previousNum1 + position] >= 0) {
        nextpage.page++;
        nextpage.order3 = position;
        nextpage.title3 = nextpage.title2 + "/" + listClicked;
        finish();
        startActivity(new Intent("com.example.example.listfiles"));
    }
    else if (listClicked.contains(".pdf") || listClicked.contains(".txt")
          || listClicked.contains(".docx") || listClicked.contains(".xlsx")
          || listClicked.contains(".mp3") || listClicked.contains(".mp4")
          || listClicked.contains(".pptx") || listClicked.contains(".jpg")
          || listClicked.contains(".ogg")) {
        int code = 0;
        int j = 0;
        for (int i = 0; i < MainActivity.notAllowedSize; i++) {
            if (listClicked.equalsIgnoreCase(MainActivity.notAllowedFileName[i])) {
                Toast.makeText(getBaseContext(), "Not accessible",
                               Toast.LENGTH_SHORT).show();
                code = 10;
            } else {}
        }
        if (code == 0) {
            for (int i = 0 ; i < MainActivity.file.length; i++) {
                if (MainActivity.file[i] == null) {}
                else if (listClicked.contains(MainActivity.file[i].getName())) {
                    code = 20;
                    j = i;
                } else {}
            }
        } else {}

        if (code == 20) {
            Uri uri = Uri.fromFile(MainActivity.file[j]);
```

```
            Intent intent = new Intent(Intent.ACTION_VIEW);
            if (MainActivity.file[j].toString().contains(".docx")) {
               // Word document
               intent.setDataAndType(uri, "application/msword");
            } else if(MainActivity.file[j].toString().contains(".pdf")) {
               // PDF file
               intent.setDataAndType(uri, "application/pdf");
            } else if(MainActivity.file[j].toString().contains(".pptx")) {
               // Powerpoint file
               intent.setDataAndType(uri, "application/vnd.ms-powerpoint");
            } else if(MainActivity.file[j].toString().contains(".xlsx")) {
               // Excel file
               intent.setDataAndType(uri, "application/vnd.ms-excel");
            } else if(MainActivity.file[j].toString().contains(".mp4")) {
               // Video files
               intent.setDataAndType(uri, "video/*");
            } else if(MainActivity.file[j].toString().contains(".mp3") ||
                  MainActivity.file[j].toString().contains(".ogg")) {
               // WAV audio file
               intent.setDataAndType(uri, "audio/x-wav");
            } else if(MainActivity.file[j].toString().contains(".jpg")) {
               // JPG file
               intent.setDataAndType(uri, "image/jpeg");
            } else if(MainActivity.file[j].toString().contains(".txt")) {
               // Text file
               intent.setDataAndType(uri, "text/plain");
            } else {}
            startActivity(intent);
         } else if (code == 10) {}
         else {
            final RequestThread requestThread = new RequestThread(MainActivity.requestDevice,
                                                   listClicked);
            requestThread.run();
         }
      }
      else if (listClicked.contains(".xml") || listClicked.contains(".m4a") ||
               listClicked.contains(".png")) {
         Toast.makeText(getBaseContext(), "Cannot access", Toast.LENGTH_SHORT).show();
      }
      else {
         Toast.makeText(getBaseContext(), "The folder is empty",
                              Toast.LENGTH_SHORT).show();
      }
   }

   else if (nextpage.page == 3) {
      listClicked = folderLabel[position].folderNames;
      if (MainActivity.numOfSubSubFolder2[nextpage.previousNum2 + position] >= 0) {
         nextpage.page++;
         nextpage.order4 = position;
         nextpage.title4 = nextpage.title3 + "/" + listClicked;
         finish();
         startActivity(new Intent("com.example.example.listfiles"));
      }
      else if (listClicked.contains(".pdf") || listClicked.contains(".txt")
            || listClicked.contains(".docx") || listClicked.contains(".xlsx")
            || listClicked.contains(".mp3") || listClicked.contains(".mp4")
            || listClicked.contains(".pptx") || listClicked.contains(".jpg")
            || listClicked.contains(".ogg")) {
         int code = 0;
```

```
    int j = 0;
    for (int i = 0; i < MainActivity.notAllowedSize; i++) {
        if (listClicked.equalsIgnoreCase(MainActivity.notAllowedFileName[i])) {
            Toast.makeText(getBaseContext(), "Not accessible",
                                    Toast.LENGTH_SHORT).show();
            code = 10;
        } else {}
    }
    if (code == 0) {
        for (int i = 0 ; i < MainActivity.file.length; i++) {
            if (MainActivity.file[i] == null) {}
            else if (listClicked.contains(MainActivity.file[i].getName())) {
                code = 20;
                j = i;
            } else {}
        }
    } else {}

    if (code == 20) {
        Uri uri = Uri.fromFile(MainActivity.file[j]);
        Intent intent = new Intent(Intent.ACTION_VIEW);
        if (MainActivity.file[j].toString().contains(".docx")) {
            // Word document
            intent.setDataAndType(uri, "application/msword");
        } else if(MainActivity.file[j].toString().contains(".pdf")) {
            // PDF file
            intent.setDataAndType(uri, "application/pdf");
        } else if(MainActivity.file[j].toString().contains(".pptx")) {
            // Powerpoint file
            intent.setDataAndType(uri, "application/vnd.ms-powerpoint");
        } else if(MainActivity.file[j].toString().contains(".xlsx")) {
            // Excel file
            intent.setDataAndType(uri, "application/vnd.ms-excel");
        } else if(MainActivity.file[j].toString().contains(".mp4")) {
            // Video files
            intent.setDataAndType(uri, "video/*");
        } else if(MainActivity.file[j].toString().contains(".mp3") ||
                MainActivity.file[j].toString().contains(".ogg")) {
            // WAV audio file
            intent.setDataAndType(uri, "audio/x-wav");
        } else if(MainActivity.file[j].toString().contains(".jpg")) {
            // JPG file
            intent.setDataAndType(uri, "image/jpeg");
        } else if(MainActivity.file[j].toString().contains(".txt")) {
            // Text file
            intent.setDataAndType(uri, "text/plain");
        } else {}
        startActivity(intent);
    } else if (code == 10) {}
    else {
        final RequestThread requestThread = new RequestThread(MainActivity.requestDevice,
                                                listClicked);
        requestThread.run();
    }
}
else if (listClicked.contains(".xml") || listClicked.contains(".m4a") ||
            listClicked.contains(".png")) {
    Toast.makeText(getBaseContext(), "Cannot access", Toast.LENGTH_SHORT).show();
}
else {
```

```
            Toast.makeText(getBaseContext(), "The folder is empty",
                                Toast.LENGTH_SHORT).show();
         }
      } else if (nextpage.page == 4) {
         Toast.makeText(getBaseContext(), "Cannot access (upto 5 levels)",
                             Toast.LENGTH_SHORT).show();
      } else {
         Toast.makeText(getBaseContext(), "ERROR", Toast.LENGTH_SHORT).show();
      }
   }
}

class ListItemLongClicked implements OnItemLongClickListener {
   @Override
   public boolean onItemLongClick(AdapterView<?> parent, View view, int position, long id) {
      posi = position;
      startActivityForResult(new Intent("com.example.example.paste"), 70);
      return false;
   }
}

public void onActivityResult(int requestCode, int resultCode, Intent data) {
   if (requestCode == 70) {
      lists.drag = 0;
      if (resultCode == RESULT_PASTE) {
         int position = 0;
         String[] temp;
         Integer[] tempIn;
         if (folderLabel.length == 0) {
            for (int i = 0; i < MainActivity.folderName.length; i++) {
               position = 0;
               int code = 0;
               if (MainActivity.folderName[i].equalsIgnoreCase(listClicked)) {
                  MainActivity.numOfSubFolder[i]++;
                  temp = MainActivity.subFolderName;
                  tempIn = MainActivity.numOfSubSubFolder;

                  MainActivity.subFolderName = new String[temp.length + 1];
                  MainActivity.numOfSubSubFolder = new Integer[temp.length + 1];
                  for (int k = 0; k < i; k++) {
                     if (MainActivity.numOfSubFolder[k] == -1) {
                        position = position + 0;
                     }
                     else {
                        position = position + MainActivity.numOfSubFolder[k];
                     }
                  }
                  for (int j = 0; j < MainActivity.subFolderName.length; j++) {
                     if (j == position) {
                        MainActivity.subFolderName[j] = nextpage.name;
                        MainActivity.numOfSubSubFolder[j] = -1;
                        code = 10;
                     }
                     else if (code == 10) {
                        MainActivity.subFolderName[j] = temp[j-1];
                        MainActivity.numOfSubSubFolder[j] = tempIn[j-1];
                     }
                     else {
                        MainActivity.subFolderName[j] = temp[j];
                        MainActivity.numOfSubSubFolder[j] = tempIn[j];
```

```
            }
        }
        break;
    } else {}
}
for (int i = 0; i < MainActivity.subFolderName.length; i++) {
    position = 0;
    int code = 0;
    if (MainActivity.subFolderName[i].equalsIgnoreCase(listClicked)) {
        MainActivity.numOfSubSubFolder[i]++;
        temp = MainActivity.subSubFolderName;
        tempIn = MainActivity.numOfSubSubFolder1;

        MainActivity.subSubFolderName = new String[temp.length + 1];
        MainActivity.numOfSubSubFolder1 = new Integer[temp.length + 1];
        for (int k = 0; k < i; k++) {
            if (MainActivity.numOfSubSubFolder[k] == -1) {
                position = position + 0;
            }
            else {
                position = position + MainActivity.numOfSubSubFolder[k];
            }
        }
        for (int j = 0; j < MainActivity.subSubFolderName.length; j++) {
            if (j == position) {
                MainActivity.subSubFolderName[j] = nextpage.name;
                MainActivity.numOfSubSubFolder1[j] = -1;
                code = 10;
            }
            else if (code == 10) {
                MainActivity.subSubFolderName[j] = temp[j-1];
                MainActivity.numOfSubSubFolder1[j] = tempIn[j-1];
            }
            else {
                MainActivity.subSubFolderName[j] = temp[j];
                MainActivity.numOfSubSubFolder1[j] = tempIn[j];
            }
        }
        break;
    } else {}
}
for (int i = 0; i < MainActivity.subSubFolderName.length; i++) {
    position = 0;
    int code = 0;
    if (MainActivity.subSubFolderName[i].equalsIgnoreCase(listClicked)) {
        MainActivity.numOfSubSubFolder1[i]++;
        temp = MainActivity.subSubFolder1Name;
        tempIn = MainActivity.numOfSubSubFolder2;

        MainActivity.subSubFolder1Name = new String[temp.length + 1];
        MainActivity.numOfSubSubFolder2 = new Integer[temp.length + 1];
        for (int k = 0; k < i; k++) {
            if (MainActivity.numOfSubSubFolder1[k] == -1) {
                position = position + 0;
            }
            else {
                position = position + MainActivity.numOfSubSubFolder1[k];
            }
        }
        for (int j = 0; j < MainActivity.subSubFolder1Name.length; j++) {
```

```
                if (j == position) {
                    MainActivity.subSubFolder1Name[j] = nextpage.name;
                    MainActivity.numOfSubSubFolder2[j] = -1;
                    code = 10;
                }
                else if (code == 10) {
                    MainActivity.subSubFolder1Name[j] = temp[j-1];
                    MainActivity.numOfSubSubFolder2[j] = tempIn[j-1];
                }
                else {
                    MainActivity.subSubFolder1Name[j] = temp[j];
                    MainActivity.numOfSubSubFolder2[j] = tempIn[j];
                }
            }
            break;
        } else {}
    }
    for (int i = 0; i < MainActivity.subSubFolder1Name.length; i++) {
        position = 0;
        int code = 0;
        if (MainActivity.subSubFolder1Name[i].equalsIgnoreCase(listClicked)) {
            MainActivity.numOfSubSubFolder2[i]++;
            temp = MainActivity.subSubFolder2Name;

            MainActivity.subSubFolder2Name = new String[temp.length + 1];
            for (int k = 0; k < i; k++) {
                if (MainActivity.numOfSubSubFolder2[k] == -1) {
                    position = position + 0;
                }
                else {
                    position = position + MainActivity.numOfSubSubFolder2[k];
                }
            }
            for (int j = 0; j < MainActivity.subSubFolder2Name.length; j++) {
                if (j == position) {
                    MainActivity.subSubFolder2Name[j] = nextpage.name;
                    code = 10;
                }
                else if (code == 10) {
                    MainActivity.subSubFolder2Name[j] = temp[j-1];
                }
                else {
                    MainActivity.subSubFolder2Name[j] = temp[j];
                }
            }
            break;
        } else {}
    }
}
else {
    for (int i = 0; i < MainActivity.folderName.length; i++) {
        if (MainActivity.folderName[i].equalsIgnoreCase(folderLabel[0].folderNames)) {
            MainActivity.numOfFolders++;
            temp = MainActivity.folderName;
            tempIn = MainActivity.numOfSubFolder;

            MainActivity.folderName = new String[MainActivity.numOfFolders];
            MainActivity.numOfSubFolder = new Integer[MainActivity.numOfFolders];
            for (int j = 0; j < MainActivity.folderName.length-1; j++) {
                MainActivity.folderName[j] = temp[j];
```

```
            MainActivity.numOfSubFolder[j] = tempIn[j];
        }
        MainActivity.folderName[MainActivity.folderName.length - 1] = nextpage.name;
        MainActivity.numOfSubFolder[MainActivity.numOfSubFolder.length - 1] = -1;
        break;
    } else {}
}
for (int i = 0; i < MainActivity.subFolderName.length; i++) {
    if (MainActivity.subFolderName[i].equalsIgnoreCase(folderLabel[0].folderNames)) {
        position = i;
        int code = 0;
        for (int j = 0; j < MainActivity.numOfSubFolder.length; j++) {
            if (MainActivity.numOfSubFolder[j] == -1) {
                position = position - 0;
            }
            else {
                position = position - MainActivity.numOfSubFolder[j];
            }
            if (position < 0) {
                MainActivity.numOfSubFolder[j]++;
                temp = MainActivity.subFolderName;
                tempIn = MainActivity.numOfSubSubFolder;

                MainActivity.subFolderName = new String[temp.length + 1];
                MainActivity.numOfSubSubFolder = new Integer[temp.length + 1];
                for (int k = 0; k < MainActivity.subFolderName.length; k++) {
                    if (k == i) {
                        MainActivity.subFolderName[k] = nextpage.name;
                        MainActivity.numOfSubSubFolder[k] = -1;
                        code = 10;
                    }
                    else if (code == 10) {
                        MainActivity.subFolderName[k] = temp[k-1];
                        MainActivity.numOfSubSubFolder[k] = tempIn[k-1];
                    }
                    else {
                        MainActivity.subFolderName[k] = temp[k];
                        MainActivity.numOfSubSubFolder[k] = tempIn[k];
                    }
                }
                break;
            } else {}
        }
        break;
    } else {}
}
for (int i = 0; i < MainActivity.subSubFolderName.length; i++) {
    if (MainActivity.subSubFolderName[i].equalsIgnoreCase(folderLabel[0].folderNames))
    {
        position = i;
        int code = 0;
        for (int j = 0; j < MainActivity.numOfSubSubFolder.length; j++) {
            if (MainActivity.numOfSubSubFolder[j] == -1) {
                position = position - 0;
            }
            else {
                position = position - MainActivity.numOfSubSubFolder[j];
            }
            if (position < 0) {
                MainActivity.numOfSubSubFolder[j]++;
```

```
                temp = MainActivity.subSubFolderName;
                tempIn = MainActivity.numOfSubSubFolder1;

                MainActivity.subSubFolderName = new String[temp.length + 1];
                MainActivity.numOfSubSubFolder1 = new Integer[temp.length + 1];
                for (int k = 0; k < MainActivity.subSubFolderName.length; k++) {
                   if (k == i) {
                      MainActivity.subSubFolderName[k] = nextpage.name;
                      MainActivity.numOfSubSubFolder1[k] = -1;
                      code = 10;
                   }
                   else if (code == 10) {
                      MainActivity.subSubFolderName[k] = temp[k-1];
                      MainActivity.numOfSubSubFolder1[k] = tempIn[k-1];
                   }
                   else {
                      MainActivity.subSubFolderName[k] = temp[k];
                      MainActivity.numOfSubSubFolder1[k] = tempIn[k];
                   }
                }
                break;
             } else {}
          }
          break;
       } else {}
    }
    for (int i = 0; i < MainActivity.subSubFolder1Name.length; i++) {
       if (MainActivity.subSubFolder1Name[i].equalsIgnoreCase(
            folderLabel[0].folderNames)) {
          position = i;
          int code = 0;
          for (int j = 0; j < MainActivity.numOfSubSubFolder1.length; j++) {
             if (MainActivity.numOfSubSubFolder1[j] == -1) {
                position = position - 0;
             }
             else {
                position = position - MainActivity.numOfSubSubFolder1[j];
             }
             if (position < 0) {
                MainActivity.numOfSubSubFolder1[j]++;
                temp = MainActivity.subSubFolder1Name;
                tempIn = MainActivity.numOfSubSubFolder2;

                MainActivity.subSubFolder1Name = new String[temp.length + 1];
                MainActivity.numOfSubSubFolder2 = new Integer[temp.length + 1];
                for (int k = 0; k < MainActivity.subSubFolder1Name.length; k++) {
                   if (k == i) {
                      MainActivity.subSubFolder1Name[k] = nextpage.name;
                      MainActivity.numOfSubSubFolder2[k] = -1;
                      code = 10;
                   }
                   else if (code == 10) {
                      MainActivity.subSubFolder1Name[k] = temp[k-1];
                      MainActivity.numOfSubSubFolder2[k] = tempIn[k-1];
                   }
                   else {
                      MainActivity.subSubFolder1Name[k] = temp[k];
                      MainActivity.numOfSubSubFolder2[k] = tempIn[k];
                   }
                }
```

```
                    break;
                } else {}
            }
            break;
        } else {}
    }
    for (int i = 0; i < MainActivity.subSubFolder2Name.length; i++) {
        if (MainActivity.subSubFolder2Name[i].equalsIgnoreCase(
             folderLabel[0].folderNames)) {
            position = i;
            int code = 0;
            for (int j = 0; j < MainActivity.numOfSubSubFolder2.length; j++) {
                if (MainActivity.numOfSubSubFolder2[j] == -1) {
                    position = position - 0;
                }
                else {
                    position = position - MainActivity.numOfSubSubFolder2[j];
                }
                if (position < 0) {
                    MainActivity.numOfSubSubFolder2[j]++;
                    temp = MainActivity.subSubFolder2Name;

                    MainActivity.subSubFolder2Name = new String[temp.length + 1];
                    for (int k = 0; k < MainActivity.subSubFolder2Name.length; k++) {
                        if (k == i) {
                            MainActivity.subSubFolder2Name[k] = nextpage.name;
                            code = 10;
                        }
                        else if (code == 10) {
                            MainActivity.subSubFolder2Name[k] = temp[k-1];
                        }
                        else {
                            MainActivity.subSubFolder2Name[k] = temp[k];
                        }
                    }
                    break;
                } else {}
            }
            break;
        } else {}
    }
}

File[] tempFile;
long[] tempDate;

tempFile = MainActivity.file;
tempDate = MainActivity.date;

MainActivity.file = new File[tempFile.length + 1];
MainActivity.date = new long[tempDate.length + 1];

for (int i = 0; i < tempFile.length; i++) {
    MainActivity.file[i] = tempFile[i];
    MainActivity.date[i] = tempDate[i];
}

String[] tempRWfile;
tempRWfile = MainActivity.readWriteFileName;
MainActivity.readWriteFileName = new String[tempRWfile.length + 1];
```

```
    MainActivity.readWriteSize++;

    for (int i = 0; i < tempRWfile.length; i++) {
        MainActivity.readWriteFileName[i] = tempRWfile[i];
    }
    MainActivity.readWriteFileName[tempRWfile.length] = nextpage.name;

    final ProgressDialog dialog = ProgressDialog.show(
            listfiles.this, "Refreshing", "Please wait...", true);
        new Thread(new Runnable(){
            public void run(){
                try {
                    Thread.sleep(2000);
                    finish();
                    startActivity(new Intent("com.example.example.listfiles"));
                    dialog.dismiss();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }).start();
}
else if (resultCode == RESULT_DELETE) {
    int position = 0;
    int code = 0;
    String[] temp;
    Integer[] tempIn;
    for (int i = 0; i < MainActivity.folderName.length; i++) {
        if (MainActivity.folderName[i].equalsIgnoreCase(folderLabel[posi].folderNames)) {
            MainActivity.numOfFolders--;
            temp = MainActivity.folderName;
            tempIn = MainActivity.numOfSubFolder;

            MainActivity.folderName = new String[MainActivity.numOfFolders];
            MainActivity.numOfSubFolder = new Integer[MainActivity.numOfFolders];
            for (int j = 0; j < MainActivity.folderName.length + 1; j++) {
                if (j == i) {
                    code = 10;
                }
                else if (code == 10) {
                    MainActivity.folderName[j-1] = temp[j];
                    MainActivity.numOfSubFolder[j-1] = tempIn[j];
                }
                else {
                    MainActivity.folderName[j] = temp[j];
                    MainActivity.numOfSubFolder[j] = tempIn[j];
                }
            }
            break;
        } else {}
    }
    for (int i = 0; i < MainActivity.subFolderName.length; i++) {
        if (MainActivity.subFolderName[i].equalsIgnoreCase(folderLabel[posi].folderNames)) {
            position = i;
            code = 0;
            for (int j = 0; j < MainActivity.numOfSubFolder.length; j++) {
                if (MainActivity.numOfSubFolder[j] == -1) {
                    position = position - 0;
                }
                else {
```

```
                position = position - MainActivity.numOfSubFolder[j];
            }
            if (position < 0) {
                MainActivity.numOfSubFolder[j]--;
                temp = MainActivity.subFolderName;
                tempIn = MainActivity.numOfSubSubFolder;

                MainActivity.subFolderName = new String[temp.length - 1];
                MainActivity.numOfSubSubFolder = new Integer[temp.length - 1];
                for (int k = 0; k < MainActivity.subFolderName.length + 1; k++) {
                    if (k == i) {
                        code = 10;
                    }
                    else if (code == 10) {
                        MainActivity.subFolderName[k-1] = temp[k];
                        MainActivity.numOfSubSubFolder[k-1] = tempIn[k];
                    }
                    else {
                        MainActivity.subFolderName[k] = temp[k];
                        MainActivity.numOfSubSubFolder[k] = tempIn[k];
                    }
                }
                break;
            } else {}
        }
        break;
    } else {}
}
for (int i = 0; i < MainActivity.subSubFolderName.length; i++) {
    if (MainActivity.subSubFolderName[i].equalsIgnoreCase(
         folderLabel[posi].folderNames)) {
        position = i;
        code = 0;
        for (int j = 0; j < MainActivity.numOfSubSubFolder.length; j++) {
            if (MainActivity.numOfSubSubFolder[j] == -1) {
                position = position - 0;
            }
            else {
                position = position - MainActivity.numOfSubSubFolder[j];
            }
            if (position < 0) {
                MainActivity.numOfSubSubFolder[j]--;
                temp = MainActivity.subSubFolderName;
                tempIn = MainActivity.numOfSubSubFolder1;

                MainActivity.subSubFolderName = new String[temp.length - 1];
                MainActivity.numOfSubSubFolder1 = new Integer[temp.length - 1];
                for (int k = 0; k < MainActivity.subSubFolderName.length + 1; k++) {
                    if (k == i) {
                        code = 10;
                    }
                    else if (code == 10) {
                        MainActivity.subSubFolderName[k-1] = temp[k];
                        MainActivity.numOfSubSubFolder1[k-1] = tempIn[k];
                    }
                    else {
                        MainActivity.subSubFolderName[k] = temp[k];
                        MainActivity.numOfSubSubFolder1[k] = tempIn[k];
                    }
                }
```

```
                break;
            } else {}
        }
        break;
    } else {}
}
for (int i = 0; i < MainActivity.subSubFolder1Name.length; i++) {
    if (MainActivity.subSubFolder1Name[i].equalsIgnoreCase(
         folderLabel[posi].folderNames)) {
        position = i;
        code = 0;
        for (int j = 0; j < MainActivity.numOfSubSubFolder1.length; j++) {
            if (MainActivity.numOfSubSubFolder1[j] == -1) {
                position = position - 0;
            }
            else {
                position = position - MainActivity.numOfSubSubFolder1[j];
            }
            if (position < 0) {
                MainActivity.numOfSubSubFolder1[j]--;
                temp = MainActivity.subSubFolder1Name;
                tempIn = MainActivity.numOfSubSubFolder2;

                MainActivity.subSubFolder1Name = new String[temp.length - 1];
                MainActivity.numOfSubSubFolder2 = new Integer[temp.length - 1];
                for (int k = 0; k < MainActivity.subSubFolder1Name.length + 1; k++) {
                    if (k == i) {
                        code = 10;
                    }
                    else if (code == 10) {
                        MainActivity.subSubFolder1Name[k-1] = temp[k];
                        MainActivity.numOfSubSubFolder2[k-1] = tempIn[k];
                    }
                    else {
                        MainActivity.subSubFolder1Name[k] = temp[k];
                        MainActivity.numOfSubSubFolder2[k] = tempIn[k];
                    }
                }
                break;
            } else {}
        }
        break;
    } else {}
}
for (int i = 0; i < MainActivity.subSubFolder2Name.length; i++) {
    if (MainActivity.subSubFolder2Name[i].equalsIgnoreCase(
         folderLabel[posi].folderNames)) {
        position = i;
        code = 0;
        for (int j = 0; j < MainActivity.numOfSubSubFolder2.length; j++) {
            if (MainActivity.numOfSubSubFolder2[j] == -1) {
                position = position - 0;
            }
            else {
                position = position - MainActivity.numOfSubSubFolder2[j];
            }
            if (position < 0) {
                MainActivity.numOfSubSubFolder2[j]--;
                temp = MainActivity.subSubFolder2Name;
```

```
                    MainActivity.subSubFolder2Name = new String[temp.length - 1];
                    for (int k = 0; k < MainActivity.subSubFolder2Name.length + 1; k++) {
                        if (k == i) {
                            code = 10;
                        }
                        else if (code == 10) {
                            MainActivity.subSubFolder2Name[k-1] = temp[k];
                        }
                        else {
                            MainActivity.subSubFolder2Name[k] = temp[k];
                        }
                    }
                    break;
                } else {}
            }
            break;
        } else {}
    }
    final ProgressDialog dialog = ProgressDialog.show(
            listfiles.this, "Refreshing", "Please wait...", true);
        new Thread(new Runnable(){
            public void run(){
                try {
                    Thread.sleep(2000);
                    finish();
                    startActivity(new Intent("com.example.example.listfiles"));
                    dialog.dismiss();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }).start();
    }
    else if (resultCode == RESULT_DRAGFROMREMOTE) {
        lists.drag = 1;
        finish();
        startActivity(new Intent("com.example.example.nextpage"));
    } else {}
  } else {}
}

public void onBackPressed() {
  --nextpage.page;
  finish();
  if (nextpage.page > -1) {
    startActivity(new Intent("com.example.example.listfiles"));
  }
}

private class RequestThread extends Thread {
  private BluetoothSocket mmSocket = null;
  private BluetoothDevice mmDevice = null;
  private OutputStream outStream = null;

  public RequestThread(BluetoothDevice device, String path) {
    // Use a temporary object that is later assigned to mmSocket,
    // because mmSocket is final
    BluetoothSocket tmp = null;
    mmDevice = device;
    // Get a BluetoothSocket to connect with the given BluetoothDevice
```

```
        try {
        // MY_UUID is the app's UUID string, also used by the server code
           tmp =
                 mmDevice.createInsecureRfcommSocketToServiceRecord(MainActivity.MY_UUID);
           mmSocket = tmp;

           // Cancel discovery because it will slow down the connection
           MainActivity.mBluetoothAdapter.cancelDiscovery();

           try {
              // Connect the device through the socket. This will block
              // until it succeeds or throws an exception
              mmSocket.connect();

              try {
                  outStream = mmSocket.getOutputStream();

                  try {
                     outStream.write("request for file transfer".getBytes());
                     Toast.makeText(getBaseContext(), "File transfer request made!",
                                    Toast.LENGTH_SHORT).show();

                     byte[] redundantRequestByte = new byte[50-"request for file transfer".length()];
                     outStream.write(redundantRequestByte);

                     outStream.write(path.getBytes());

                     try {
                       outStream.flush();
                       outStream.close();
                       mmSocket.close();
                       tmp.close();
                     } catch (IOException e) {
                       e.printStackTrace();
                     }
                  } catch (IOException e) {
                     e.printStackTrace();
                  }
               } catch (IOException e) {
                  e.printStackTrace();
              }
           } catch (IOException e) {
              e.printStackTrace();
           }
        }  catch (IOException connectException) {}
     }
  }

  private class SavedBackThread extends Thread {
     private BluetoothSocket mmSocket = null;
     private BluetoothDevice mmDevice = null;
     private OutputStream outStream = null;

     public SavedBackThread(String deviceAddress, File[] nameFile) {
        // Use a temporary object that is later assigned to mmSocket,
        // because mmSocket is final
        BluetoothSocket tmp = null;
        mmDevice = MainActivity.mBluetoothAdapter.getRemoteDevice(deviceAddress);
        // Get a BluetoothSocket to connect with the given BluetoothDevice
        try {
```

```
// MY_UUID is the app's UUID string, also used by the server code
   tmp =
         mmDevice.createInsecureRfcommSocketToServiceRecord(MainActivity.MY_UUID);
   mmSocket = tmp;

   // Cancel discovery because it will slow down the connection
   MainActivity.mBluetoothAdapter.cancelDiscovery();

   try {
      // Connect the device through the socket. This will block
      // until it succeeds or throws an exception
      mmSocket.connect();

      try {
          outStream = mmSocket.getOutputStream();

          try {
             outStream.write("file saved back".getBytes());
             Toast.makeText(getBaseContext(), "File is saved back to remote!",
                                  Toast.LENGTH_SHORT).show();

             byte[] redundantRequestByte = new byte[500-"file saved back".length()];
             outStream.write(redundantRequestByte);

             outStream.write(nameFile.length);
             for (int x = 0; x < nameFile.length; x++) {
                byte[] fileByte = new byte[(int) nameFile[x].length()];
                try {
                   FileInputStream fileInputStream = new FileInputStream(nameFile[x]);
                   fileInputStream.read(fileByte);
                   fileInputStream.close();
                } catch (FileNotFoundException e) {
                   // TODO Auto-generated catch block
                   e.printStackTrace();
                }

                outStream.write(nameFile[x].getName().length());
                outStream.write(nameFile[x].getName().getBytes());
                byte[] redundantfileNameByte = new byte[499-nameFile[x].getName().length()];
                outStream.write(redundantfileNameByte);

                int size = (int) nameFile[x].length();
                int i = 0;

                while (size > 100) {
                   size = size - 100;
                   i++;
                }
                int j = 0;
                while (i > 100) {
                   i = i - 100;
                   j++;
                }

                int k = 0;
                while (j > 100) {
                   j = j - 100;
                   k++;
                }
```

```
                        outStream.write(size);
                        outStream.write(i);
                        outStream.write(j);
                        outStream.write(k);

                        outStream.write(fileByte);
                    }

                    try {
                      outStream.flush();
                      outStream.close();
                      mmSocket.close();
                      tmp.close();
                    } catch (IOException e) {
                      e.printStackTrace();
                    }
                  } catch (IOException e) {
                     e.printStackTrace();
                  }
                } catch (IOException e) {
                  e.printStackTrace();
                }
              } catch (IOException e) {
                 e.printStackTrace();
              }
            }  catch (IOException connectException) {}
        }
    }

    @Override
    public boolean onTouchEvent(MotionEvent me) {
        return gestureScanner.onTouchEvent(me);
    }
    @Override
    public boolean onDown(MotionEvent e) {
        return true;
    }
    @Override
    public void onLongPress(MotionEvent e) {
        startActivityForResult(new Intent("com.example.example.paste1"), 70);
    }
    @Override
    public boolean onScroll(MotionEvent e1, MotionEvent e2, float distanceX, float distanceY) {
        return true;
    }
    @Override
    public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX, float velocityY) {
        return true;
    }
    @Override
    public void onShowPress(MotionEvent e) {}
    @Override
    public boolean onSingleTapUp(MotionEvent e) {
        return true;
    }
}
```

listfiles2.java

```
package com.example.example;

import android.content.Context;
import android.graphics.Color;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.ImageView;
import android.widget.TextView;

public class listfiles2 extends BaseAdapter {
   Context mContext;
   listfiles.Names[] mArray;
   public listfiles2 (Context c, listfiles.Names mArraylist[]){
      mContext = c;
      mArray = mArraylist;
   }

   @Override
   public int getCount() {
      // TODO Auto-generated method stub
      return mArray.length;
   }

   @Override
   public Object getItem(int arg0) {
      // TODO Auto-generated method stub
      return null;
   }

   @Override
   public long getItemId(int arg0) {
      // TODO Auto-generated method stub
      return 0;
   }

   static class ViewHolder{
      TextView text;
      ImageView icon;
   }
   @Override
   public View getView(int pos, View convertView, ViewGroup arg2) {
      // TODO Auto-generated method stub
      ViewHolder viewHolder;
      LayoutInflater li = (LayoutInflater)
                           mContext.getSystemService(Context.LAYOUT_INFLATER_SERVICE);

      if (convertView==null){
         convertView = li.inflate(R.layout.main3, null);

         viewHolder = new ViewHolder();
         viewHolder.text= (TextView)convertView.findViewById(R.id.txtView);
         viewHolder.icon= (ImageView)convertView.findViewById(R.id.imgView);

         convertView.setTag(viewHolder);
      }
      else
```

```
    viewHolder = (ViewHolder) convertView.getTag();

viewHolder.text.setText(mArray[pos].folderNames);

if (mArray[pos].color == "RED") {
    viewHolder.text.setTextColor(Color.RED);
}

else if (mArray[pos].color == "PURPLE") {
    viewHolder.text.setTextColor(-8578178);
}

else if (mArray[pos].color == "BLUE") {
    viewHolder.text.setTextColor(Color.BLUE);
}

else {
    viewHolder.text.setTextColor(Color.BLACK);
}

if (mArray[pos].folderNames.contains(".txt")) {
    viewHolder.icon.setImageResource(R.drawable.txt);
}

else if (mArray[pos].folderNames.contains(".docx")) {
    viewHolder.icon.setImageResource(R.drawable.docx);
}

else if (mArray[pos].folderNames.contains(".xlsx")) {
    viewHolder.icon.setImageResource(R.drawable.xlsx);
}

else if (mArray[pos].folderNames.contains(".pdf")) {
    viewHolder.icon.setImageResource(R.drawable.pdf);
}

else if (mArray[pos].folderNames.contains(".mp3")) {
    viewHolder.icon.setImageResource(R.drawable.music);
}

else if (mArray[pos].folderNames.contains(".pptx")) {
    viewHolder.icon.setImageResource(R.drawable.pptx);
}

else if (mArray[pos].folderNames.contains(".mp4")) {
    viewHolder.icon.setImageResource(R.drawable.video);
}

else if (mArray[pos].folderNames.contains(".xml")) {
    viewHolder.icon.setImageResource(R.drawable.xml);
}

else if (mArray[pos].folderNames.contains(".jpg")) {
    viewHolder.icon.setImageResource(R.drawable.jpg);
}

else if (mArray[pos].folderNames.contains(".ogg")) {
    viewHolder.icon.setImageResource(R.drawable.ogg);
}
```

```
        else if (mArray[pos].folderNames.contains(".m4a")) {
            viewHolder.icon.setImageResource(R.drawable.m4a);
        }

        else if (mArray[pos].folderNames.contains(".png")) {
            viewHolder.icon.setImageResource(R.drawable.png);
        }

        else {
            viewHolder.icon.setImageResource(R.drawable.folder);
        }

        return convertView;
    }
}
```

listfiles3.java

```
package com.example.example;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.ImageView;
import android.widget.TextView;

public class listfiles3 extends BaseAdapter {
   Context mContext;
   lists.FileName[] mArray;

   public listfiles3 (Context c, lists.FileName mArraylist[]){
      mContext = c;
      mArray = mArraylist;
   }

   @Override
   public int getCount() {
      // TODO Auto-generated method stub
      return mArray.length;
   }

   @Override
   public Object getItem(int arg0) {
      // TODO Auto-generated method stub
      return null;
   }

   @Override
   public long getItemId(int arg0) {
      // TODO Auto-generated method stub
      return 0;
   }

   static class ViewHolder{
      TextView text;
      ImageView icon;
   }
   @Override
   public View getView(int pos, View convertView, ViewGroup arg2) {
      // TODO Auto-generated method stub
      ViewHolder viewHolder;
      LayoutInflater li = (LayoutInflater)
                           mContext.getSystemService(Context.LAYOUT_INFLATER_SERVICE);

      if (convertView==null){
         convertView = li.inflate(R.layout.main3, null);

         viewHolder = new ViewHolder();
         viewHolder.text= (TextView)convertView.findViewById(R.id.txtView);
         viewHolder.icon= (ImageView)convertView.findViewById(R.id.imgView);

         convertView.setTag(viewHolder);
      }
      else
```

```
            viewHolder = (ViewHolder) convertView.getTag();

        viewHolder.text.setText(mArray[pos].folderNames);

        if (mArray[pos].folderNames.contains(".txt")) {
            viewHolder.icon.setImageResource(R.drawable.txt);
        }

        else if (mArray[pos].folderNames.contains(".docx")) {
            viewHolder.icon.setImageResource(R.drawable.docx);
        }

        else if (mArray[pos].folderNames.contains(".xlsx")) {
            viewHolder.icon.setImageResource(R.drawable.xlsx);
        }

        else if (mArray[pos].folderNames.contains(".pptx")) {
            viewHolder.icon.setImageResource(R.drawable.pptx);
        }

        else if (mArray[pos].folderNames.contains(".pdf")) {
            viewHolder.icon.setImageResource(R.drawable.pdf);
        }

        else if (mArray[pos].folderNames.contains(".mp3")) {
            viewHolder.icon.setImageResource(R.drawable.music);
        }

        else if (mArray[pos].folderNames.contains(".mp4")) {
            viewHolder.icon.setImageResource(R.drawable.video);
        }

        else if (mArray[pos].folderNames.contains(".xml")) {
            viewHolder.icon.setImageResource(R.drawable.xml);
        }

        else if (mArray[pos].folderNames.contains(".jpg")) {
            viewHolder.icon.setImageResource(R.drawable.jpg);
        }

        else if (mArray[pos].folderNames.contains(".ogg")) {
            viewHolder.icon.setImageResource(R.drawable.ogg);
        }

        else if (mArray[pos].folderNames.contains(".m4a")) {
            viewHolder.icon.setImageResource(R.drawable.m4a);
        }

        else if (mArray[pos].folderNames.contains(".png")) {
            viewHolder.icon.setImageResource(R.drawable.png);
        }

        else {
            viewHolder.icon.setImageResource(R.drawable.folder);
        }

        return convertView;
    }
}
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android”
   package="com.example.example"
   android:versionCode="1"
   android:versionName="1.0" >

   <uses-permission android:name="android.permission.BLUETOOTH" />
   <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />

   <uses-permission android:name="android.permission.WRITE_INTERNAL_STORAGE" />
   <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

   <uses-sdk
      android:minSdkVersion="14"
      android:targetSdkVersion="19" />

   <application
      android:allowBackup="true"
      android:icon="@drawable/ic_launcher"
      android:label="@string/app_name"
      android:theme="@style/AppTheme" >
      <activity
         android:name="com.example.example.MainActivity"
         android:label="@string/app_name" >
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />

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

      <activity
         android:name="com.example.example.nextpage"
         android:label="ScreenSharefromRemote" >
         <intent-filter>
            <action android:name="com.example.example.nextpage" />

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

      <activity
         android:name="com.example.example.listfiles"
         android:label="ScreenSharefromRemote" >
         <intent-filter>
            <action android:name="com.example.example.listfiles" />

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

      <activity
         android:name="com.example.example.synchronize"
         android:theme="@android:style/Theme.Dialog"
         android:label="ScreenSharing" >
         <intent-filter>
            <action android:name="com.example.example.synchronize" />
```

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

<activity
    android:name="com.example.example.lists"
    android:label="ScreenSharing" >
    <intent-filter>
        <action android:name="com.example.example.lists" />

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

<activity
    android:name="com.example.example.copy"
    android:theme="@android:style/Theme.Holo.Dialog.NoActionBar">
    <intent-filter>
        <action android:name="com.example.example.copy" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>

<activity
    android:name="com.example.example.paste"
    android:theme="@android:style/Theme.Holo.Dialog.NoActionBar">
    <intent-filter>
        <action android:name="com.example.example.paste" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>

<activity
    android:name="com.example.example.paste1"
    android:theme="@android:style/Theme.Holo.Dialog.NoActionBar">
    <intent-filter>
        <action android:name="com.example.example.paste1" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>

<activity
    android:name="com.example.example.copy1"
    android:theme="@android:style/Theme.Holo.Dialog.NoActionBar">
    <intent-filter>
        <action android:name="com.example.example.copy1" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>

<activity
    android:name="com.example.example.music"
    android:label="ScreenSharefromRemote" >
    <intent-filter>
        <action android:name="com.example.example.music" />

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

```
        <activity
            android:name="com.example.example.video"
            android:label="ScreenSharefromRemote" >
            <intent-filter>
                <action android:name="com.example.example.video" />

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

        <activity
            android:name="com.example.example.picture"
            android:label="ScreenSharefromRemote" >
            <intent-filter>
                <action android:name="com.example.example.picture" />

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

        <activity
            android:name="com.example.example.document"
            android:label="ScreenSharefromRemote" >
            <intent-filter>
                <action android:name="com.example.example.document" />

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

        <activity
            android:name="com.example.example.configuration"
            android:label="ScreenSharefromRemote" >
            <intent-filter>
                <action android:name="com.example.example.configuration" />

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

    </application>

</manifest>
```

/res/anim/out_to_right.xml

```
<set xmlns:android="http://schemas.android.com/apk/res/android"
   android:shareInterpolator="false">
     <translate android:fromXDelta="0%" android:toXDelta="100%"
       android:fromYDelta="0%" android:toYDelta="0%"
       android:duration="1000"/>
</set>
```

/res/anim/out_to_left.xml

```
<set xmlns:android="http://schemas.android.com/apk/res/android"
   android:shareInterpolator="false">
     <translate android:fromXDelta="0%" android:toXDelta="-100%"
       android:fromYDelta="0%" android:toYDelta="0%"
       android:duration="1000"/>
</set>
```

/res/anim/in_from_right.xml

```
<set xmlns:android="http://schemas.android.com/apk/res/android"
   android:shareInterpolator="false">
   <translate
      android:fromXDelta="100%" android:toXDelta="0%"
        android:fromYDelta="0%" android:toYDelta="0%"
        android:duration="1000" />
</set>
```

/res/anim/in_from_left.xml

```
<set xmlns:android="http://schemas.android.com/apk/res/android"
   android:shareInterpolator="false">
   <translate
      android:fromXDelta="-100%" android:toXDelta="0%"
        android:fromYDelta="0%" android:toYDelta="0%"
        android:duration="1000" />
</set>
```

/res/layout/activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:background="@drawable/jellybean"
   android:orientation="vertical" >

  <Button
      android:id="@+id/btn_0"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentTop="true"
      android:layout_alignParentLeft="true"/>

  <TextView
      android:id="@+id/txt1"
      android:layout_below="@+id/btn_0"
      android:layout_alignLeft="@+id/btn_0"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"/>

  <TextView
      android:id="@+id/txt2"
      android:layout_below="@+id/txt1"
      android:layout_alignLeft="@+id/txt1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"/>

  <TextView
      android:id="@+id/txt3"
      android:layout_below="@+id/txt2"
      android:layout_alignLeft="@+id/txt2"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"/>

  <ProgressBar
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_below="@+id/txt2"
      android:layout_alignRight="@+id/txt2"
      style="@android:style/Widget.ProgressBar.Small"
      android:id="@+id/progressbar" />

  <ListView
      android:id="@+id/listViewRemoteDevice"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:layout_below="@+id/txt3"/>

   <Button
      android:id="@+id/btn_1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentBottom="true"/>

   <Button
      android:id="@+id/btn_configuration"
      android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_alignParentRight="true"/>

</RelativeLayout>
```

/res/layout/screen.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:id="@+id/home"
    android:orientation="vertical">

    <TextView
        android:id="@+id/screentext"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:background="#FFFFFFFF"
        android:textColor="#000099"
        android:textSize="20sp"
        android:layout_alignParentTop="true"/>

    <ViewFlipper
        android:id="@+id/view_flipper"
        android:layout_below="@+id/screentext"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent">
    </ViewFlipper>
</RelativeLayout>
```

/res/layout/configuration.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/android"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/txtViewnum1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <ListView
        android:id="@+id/lstViewnum1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />

</LinearLayout>
```

/res/layout/listaccessiblefile.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:background="@drawable/jellybean"
   android:orientation="vertical" >

   <TextView
      android:id="@+id/txt_view"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentTop="true"
      android:layout_alignParentLeft="true" />

   <Button
      android:id="@+id/btn_save"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentTop="true"
      android:layout_alignParentRight="true" />

   <ListView
      android:id="@+id/lst_view"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:layout_below="@+id/txt_view" />

</RelativeLayout>
```

/res/layout/copy.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical" >

   <Button
     android:id="@+id/button1"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content" />

   <Button
     android:id="@+id/button2"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content" />

   <Button
     android:id="@+id/button3"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content" />

</LinearLayout>
```

/res/layout/paste.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical" >

   <Button
      android:id="@+id/button1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" />

</LinearLayout>
```

/res/layout/list.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:background="@drawable/android"
   android:orientation="vertical" >

   <TextView
      android:id="@+id/txtnum1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" />

   <ListView
      android:id="@+id/lstnum1"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content" />

</LinearLayout>
```

/res/layout/main2.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:background="@drawable/android"
   android:orientation="vertical" >

   <TextView
      android:id="@+id/txtview"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentTop="true"
      android:layout_alignParentLeft="true"
      android:layout_marginRight="100dp" />

   <Button
      android:id="@+id/btn_synchronize"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentTop="true"
      android:layout_alignParentRight="true" />

   <ListView
      android:id="@+id/lstview"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:layout_below="@+id/txtview"
      android:layout_alignLeft="@+id/txtview"/>

</RelativeLayout>
```

/res/layout/main3.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="horizontal" >

   <ImageView
   android:id="@+id/imgView"
   android:layout_width="30dp"
   android:layout_height="30dp"
   android:layout_margin="10dp"/>

   <TextView
   android:id="@+id/txtView"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_margin="10dp"/>

</LinearLayout>
```

/res/layout/drawer_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:gravity="center_horizontal">

    <ImageView
        android:id="@+id/icon_image"
        android:layout_width="35dp"
        android:layout_height="35dp"
        android:padding="3dp"/>

    <TextView
        android:id="@+id/icon_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:maxLines="2"
        android:gravity="center_horizontal"/>
</LinearLayout>
```