EFFICIENT MOBILE INFORMATION SHARING THROUGH FINGER SWIPE

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science.

By

Ping Hu

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

November 2016

Fargo, North Dakota

North Dakota State University
Graduate School

Title

EFFICIENT MOBILE INFORMATION SHARING THROUGH FINGER SWIPE

By

Ping Hu

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr Juan Li
Co-Chair

Dr Pratap, Kotala
Co-Chair

Dr Limin, Zhang

Approved:

January 9, 2017	Dr Brian M. Slator
Date	Department Chair

ABSTRACT

Smartphones have been pervasive in our daily life, and are now the dominant driver of social media. People post and access information on the social media through smartphones. In addition, users are likely to share information among a small group, such as taking a photo and sending the photo to a friend. However, there still lacks an efficient method to share information among smartphones in an ad-hoc manner. Mobile information sharing in general relies on some external application. This application is designed and implemented on the purpose of offering an alternative and more convenient tool for the mobile users' data exchange. This application especially focuses on users with little computer knowledge. Also it provides an efficient way to identify the users before making a group through physical finger swipe across all the screens, instead of distinguishing the actual identification from a virtual name.

ACKNOWLEDGEMENTS

I would like to express my gratitude to my adviser, Dr. Juan Li, for her continued support. Her advice, guidance and support have greatly helped me in making this work possible. I would also like to thank Dr. Pratap Kotala and Dr. Limin Zhang for their valuable time as my committee members. Besides, I would like to say thank you to Ms. Annette Sprague on my paper work.

I would like to thank my family, Jun, Juanfen, Guohua and Shawn, for their everlasting support in the success of my career. They are my source of my power and strength without a shadow of a doubt.

And I would like to thank for the help to my friend Zheng for sharing his valuable experience on the multi-platform communications.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION AND BACKGROUND

1.1. Problem Statement

Being highly portable, smartphones have been pervasive in our daily life, and are now the dominant driver of social media. People post and access information on the social media through smartphones. In addition, users are likely to share information among a small group, such as taking a photo and sending the photo to a friend. However, there still lacks an efficient method to share information among smart phones in an ad-hoc manner. Mobile information sharing in general relies on some external application. For example, the sender opens a mobile email application, completes the login process, writes the email, and attaches a photo. Then, the receiver repeats almost the same actions to download the photo. Due to the lack of a keyboard and a mouse, the above actions are especially tedious and error-prone on mobile devices.

The advance in the wireless networking makes it possible to automatically recognize mobile devices in a close range. Accordingly, a user can use a graphical user interface to set a virtual connection with another user by tapping the corresponding mobile device from a list that includes recognized mobile devices. However, skimming a list on the small screen of a mobile device and selecting one object from the list are inefficient and error prone. First, selecting an object on a mobile device by tapping through a finger is not as efficient and precise as the selection on a desktop through a mouse due to the fat finger error. Second, each device may have a nick name so that a user has to map the nick name to a physical mobile device. This thinking effort increases the cognitive load and consequently slows down the selection process.

This project develops a natural mobile interaction, which makes mobile information sharing as efficient and intuitive as dragging and dropping in desktop computers. Our approach is featured with a novel and intuitive pairing process that sets up a virtual connection among multiple devices to form a group in an ad hoc manner. In other words, a user can join or leave a group without the knowledge of network addresses of other members. This project implements an intuitive finger gesture to complete the paring process. Specifically speaking, a user starts the pairing process by touching his/her finger on the screen of a mobile device, swiping the finger across boundaries and finally stopping the swipe on the target device. Consequently, the source and target devices set up a logic connection for information sharing. A finger swipe gesture is more efficient than a tapping gesture on a mobile device. Furthermore, our approach supports multiple devices to form a group by simply swiping the figure across the screens of multiple devices during the paring process. Since multiple groups may be formed at the same time, an efficient ad-hoc algorithm was developed to differentiate groups. Specifically speaking, the slope of the finger swipe right before exiting the current mobile device is compared with that right after entering a new mobile device. This heuristics is designed based on the assumption that a user has a smooth finger swipe so that the moving direction of a finger swipe can be used to differentiate groups. In addition, our approach compares the time of exiting and entering a device to supplement the slope comparison to avoid conflicts in the pairing process.

The pairing process virtually connects two or multiple devices together, which forms the foundation of information sharing. Without losing generality, an application was developed to transfer information from a master device (i.e., the device initiating) to slave devices. This paper also discusses the application of the proposed pairing technique to different domains, such as smart travel or brainstorming.

In summary, the contributions of this work are summarized as follows.

- **An intuitive finger gesture**. Our approach is featured with an intuitive finger gesture. The gesture of swiping a finger across the touch screens of multiple devices mimics the stitching action that has the semantics of integrating objects together in the real world. Therefore, the semantics of the finger swipe gesture is consistent with gesture's effect. This gesture is intuitive, natural and efficient.

- **Conflict avoidance**. Since multiple groups may start the pairing process simultaneously. An efficient heuristic algorithm was developed by analyzing the time and slope of a finger swipe to differentiate mobile devices in different groups.
- **A mobile information sharing application**. The above concept was implemented in a client-server architecture, where the client, deployed on an Android device, is developed with Java in the Android Studio and the server, deployed on a Windows desktop, with C# in the Visual Studio. A set of implementation-related issues were addressed in the development, such as cross-language and cross-platform communication, thread safety, Android interface design and etc.

The remainder of the paper is organized as follows.

1.2. Previous Work

One of central themes in the information sharing is to efficiently pair two or multiple devices in an ad hoc manner without knowing the network address of other devices. The diversity of sensors installed on mobile devices provides a rich space to develop various gestures to set up a virtual connection between devices. Those gestures were different with the underlying sensing technique and applied to different interaction contexts.

- **Pen-based gesture**. The stitching approach proposed a pen-based gesture which dynamically forged a connection between two devices with the gesture of continuously moving a pen from one device to another one [Hin04].

- **Radio based sensing**. ConnecTables [Tan01] used the radio frequency transponder technology to realize a flexible coupling of displays when they were moving close to each other. This approach applies to desktops to form a larger workspace for col-located tasks [Tan01].
- **Accelerometer-based sensing**. Accelerometer is a standard sensor built in a mobile device, and is useful to detect hand's movements. Accordingly, accelerometer data are used to detect a user's gesture for connecting two devices, such as bumping two tablets to set up a connection [Hin03]. Since desktop computers in general do not have an accelerometer sensor, the connection between a smartphone and a large display device involves multiple sensing techniques, such as the combination of accelerometer and touch screen. PhoneTouch implemented a gesture of bumping a mobile device with a tabletop to connect the mobile device and tabletop together. This approach synchronized a touch event detected on an interactive surface with a bump event detected by an accelerometer sensor on the mobile device [Sch10]. Similarly, Hutama et al. [Hut11] combined accelerometer with multi-touch screens by correlating the angle of two touch points with accelerometer data to set up a connection between two devices.
- **Acoustic sensing**. Point&Connect [Pen09] implemented an intuitive gesture of pointing a mobile device to the intended target for setting up a connection between the source and target devices. This approach derives the target by calculating the maximum distance change based on the acoustic signals.
- **NFC Technique**. NFC is an emerging technique that allows two devices to establish communication in a close distance. Hardy and Rukzio [Har08] implemented a grid of

NFC tags on a display. Accordingly, a user's selection is detected based on the tag touched by a user through an NFC enabled mobile device.

- **Camera-based sensing**. Camera has been a standard hardware component built in the smartphone. Therefore, visual recognition was commonly used to select an object from a distant display. The SpotCodes system realized a point & shot interaction pattern [Mad04]. A distant display presents a set of visual markers. Then, a user points his/her phone at a visual tag on the display to select the corresponding object [Mad04]. Since a built-in camera can only detect distant objects, some approaches used an external camera to recognize objects and accordingly set up a connection between two devices. In BlueTable [Wil07], a user placed a mobile device on top of an interactive surface. Then, a vision-based handshaking procedure was determined by detecting near-infrared light blinking from the mobile device. The handshaking procedure set up a connection between the mobile device and the interactive surface.
- **Tapping based gesture on touch screens**. Touch screen is the dominant input method on mobile devices. Some approaches synchronized the tapping activities on touch screens to connect corresponding devices, such as synchronizing button pressing and releasing on two devices [Rek03], or pressing a plug-button on a source device followed by pressing a socket button on a target device [Iwa03].

In summary, various intuitive gestures have been designed to pair multiple devices together. However, the majority of the previous work focused on pairing two devices while our approach is applicable to multiple devices. Though different sensing techniques were used to detect a user's gesture, our approach uses the touch screen as the sensing technique, which has the benefit of robustness and reliability, Different from other touch-screen based gestures

[Rek03, Iwa03], our approach proposed a finger swipe gesture which is more efficient on mobile devices than the tapping gesture.

Table 1-1. Summary of Gestures for Setting Up a Connection between Devices

	Sensing technology	Gesture
[Hin04]	Pen	Continuously move a pen from one device to another one
[Tan01]	Radio frequency transponder technology	Move close to each other
[Hin03]	Accelerometer	Bump two mobile devices
[Sch10]	Accelerometer and touch screen	Bump a mobile device to the touch screen of a large display device
[Hut11]	Accelerometer and touch screen	Touch a mobile device on top of a multi-touch screen
[Pen09]	Speaker and microphone	Point a mobile device to the intended target
[Har08]	NFC communication technique	Touch an NFC tag on a display
[Mad04]	Camera built in a smartphone	Point and select
[Wil07]	Camera	Place a mobile device on top of an interactive surface and detect blinking from the mobile device
[Rek03]	Touch screen	Press and release buttons simultaneously on two devices
[Iwa03]	Touch screen	Press a plug-button on a source device followed by pressing a socket button on a target device

Once devices are connected, they can start exchanging information. Various gestures were proposed to transfer data between paired devices, such as the Scoop-and-Spread gesture [Aya00], the pen gesture of pick-and-drop [Rek97], the pouring gesture [Sey13], or throw and tilt gesture [Dac09]. Furthermore, a mobile device is commonly used as a remote controller to interact with a distant display, such as sweep and point & shoot [Bal05], touch projector [Bor10], SnapAndGrab for content sharing [Mau08], and remote operations based on display registration [Pea09]. Our research focuses on the information sharing among mobile devices, but the proposed method can be easily extended to various devices, including both large and small devices. We implemented a common tap gesture to share information among paired mobile devices, but previous gestures can be integrated with finger swipe in different application scenarios in the future.

2. PROJECT DESCRIPTION/SYSTEM OVERVIEW

2.1. Function Specification

2.1.1. Applications Introduction

The system will start-up automatically with the ‘login screen’. This screen shows clickable objects that give access to all the services, such as connect the server, disconnect the server, send user name, and make a group. And then app goes to capture the finger movement across all the cell phone screens. Next screen to show the list of all the group members. The master cell phone turns different screen to display the images meanwhile the slave cell phones keep the pervious screen for the group member list. The last screen always allows the master cell phone to share specific image to another specific slave cell phone. Alternative function is to add a new member to the existing group. The new user starts from ‘login screen’ and go to ‘make a group’ screen according to the instructions. And the existing team leader stays at the screen showing images. The adding new member into an existing group can be done with the finger movement starting from leader screen to that of the new cell phone.

2.1.1.1. Image Share Application

The applications as defined in last chapter are described in more details in the form of flowcharts.

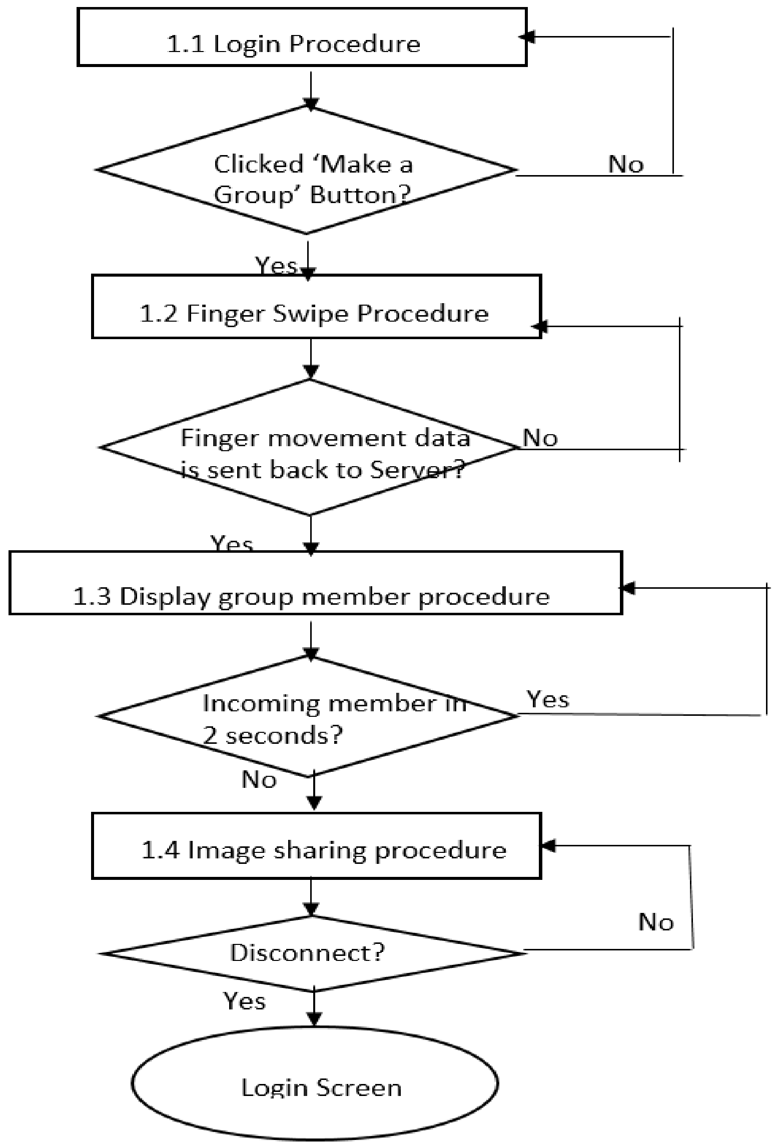

1.1 Login Procedure
Clicked 'Make a Group' Button?
No
Yes
1.2 Finger Swipe Procedure
Finger movement data is sent back to Server?
No
Yes
1.3 Display group member procedure
Incoming member in 2 seconds?
Yes
No
1.4 Image sharing procedure
Disconnect?
No
Yes
Login Screen

Figure 2-1. Flowchart of Image Sharing Application

2.1.1.1.1. Login procedure

Main aspects:

- Click the structured 'CONNECT' button.
- Type username in text field.
- Click the structured 'SEND' button.
- Click the structured 'MAKE A GROUP' button.

- App results in a new screen to ask user to swipe the finger across all the screens in a linear line.

Besides these services, a clickable button 'DISCONNECT' will also be available for disconnection.

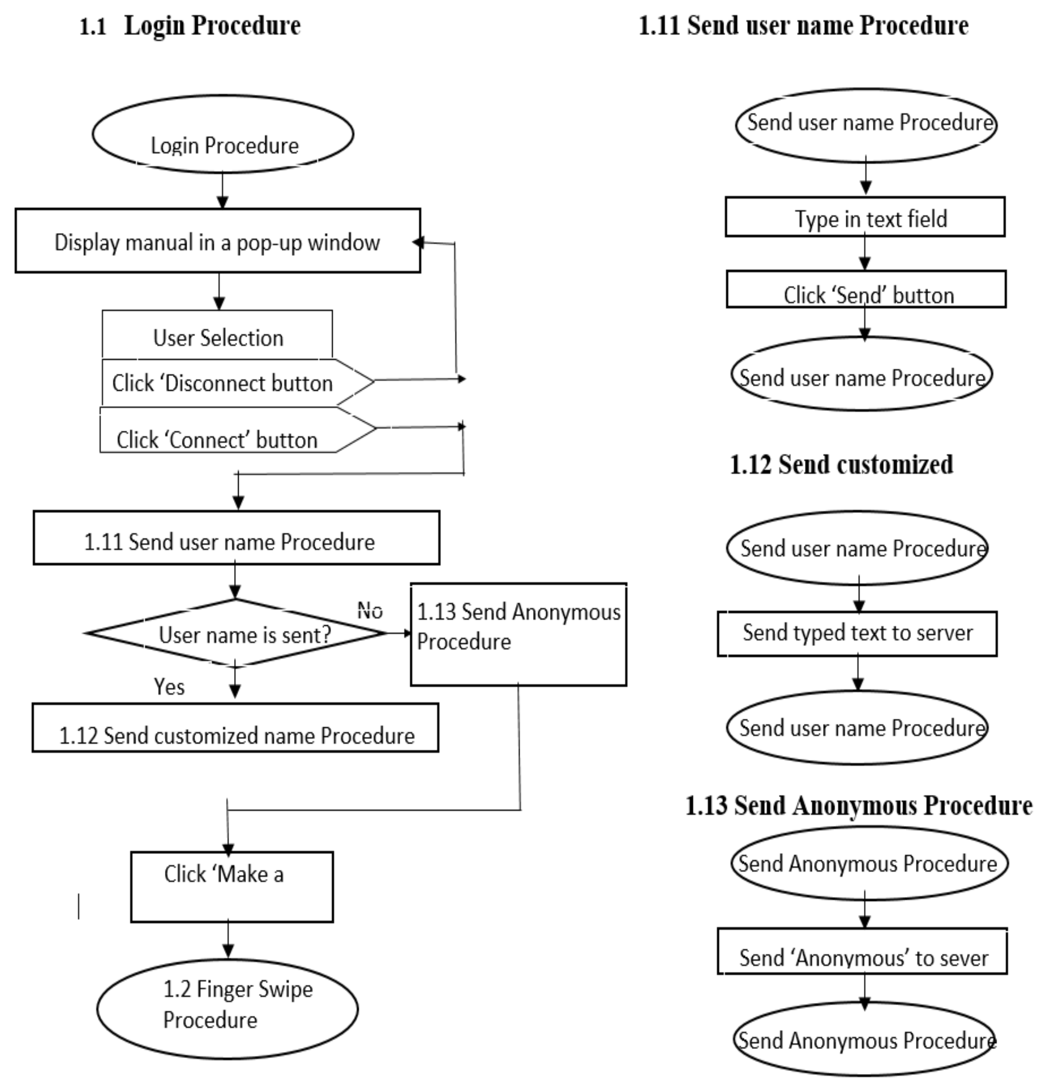


Figure 2-2. Flowchart of Login Procedure

2.1.1.1.2. Finger swipe procedure

Main aspects:

- App results in a new screen to ask user to swipe the finger across all the screens in a linear line.

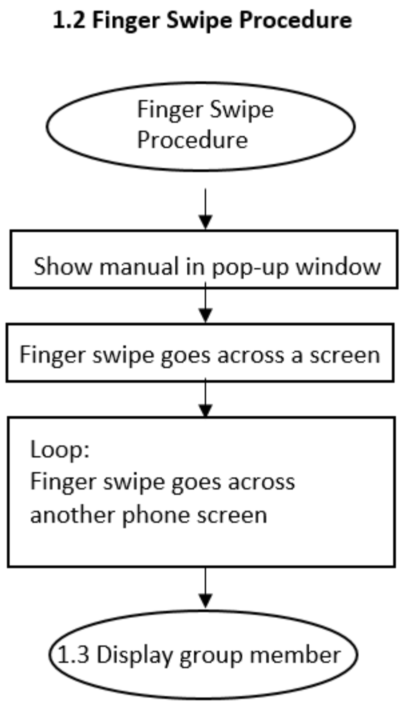

1.2 Finger Swipe Procedure
Finger Swipe Procedure
Show manual in pop-up window
Finger swipe goes across a screen
Loop:
Finger swipe goes across another phone screen
1.3 Display group member

Figure 2-3. Flowchart of Finger Swipe Procedure

2.1.1.1.3. Display group members procedure

Main aspects:

- App results in a new screen to allow user to go through all the member name list. If more name list are available than can fit into one screen, user can apply finger scroll to indicate additional screens.
- Alternative action: If the group member’s name is too long to display within one line. The user can click the name and display the full long name in a popped-up window toast.

- The system will block other functionalities if the cell phone is not the leader of this team. And there will be a pop-up toast to let the slave cell phone user wait for the action of the team leader.

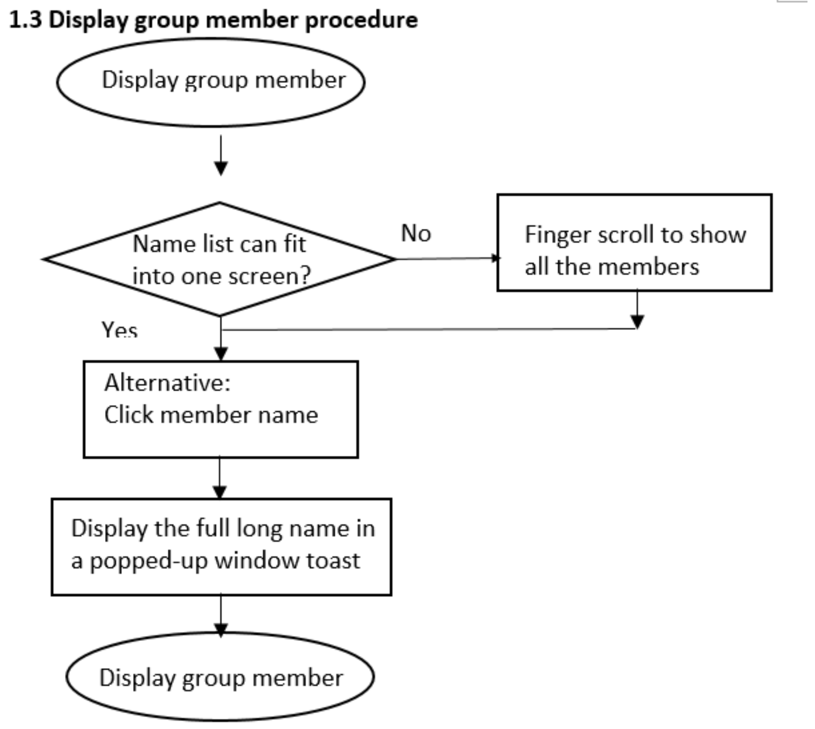

1.3 Display group member procedure
Display group member
Name list can fit into one screen?
No
Finger scroll to show all the members
Yes
Alternative: Click member name
Display the full long name in a popped-up window toast
Display group member

Figure 2-4. Flowchart of Display Group Member Procedure

2.1.1.1.4. Image sharing procedure

Main aspects:

- Alternative action for team leader:
    - App results in a new screen to display all the images in a predefined folder in a grid view. The user (team leader) to go through all images. If more grid view images are available than can fit into one screen, user can apply finger scroll to indicate additional screens.
    - Loop:

1. Click a specific image that the leader wants to share with another client. In a pop-up window, all the client names are shown.
2. Click a specific name to identify to whom the leader is going to share that image. And system sends the image to the targeted client, then turns back to the image view screen.

- Alternative action for non-team leader:
  - Loop:
    1. No action should take. System will display the image sent from the team leader in a single image way.

Besides these services, a clickable button will also be available for disconnection.

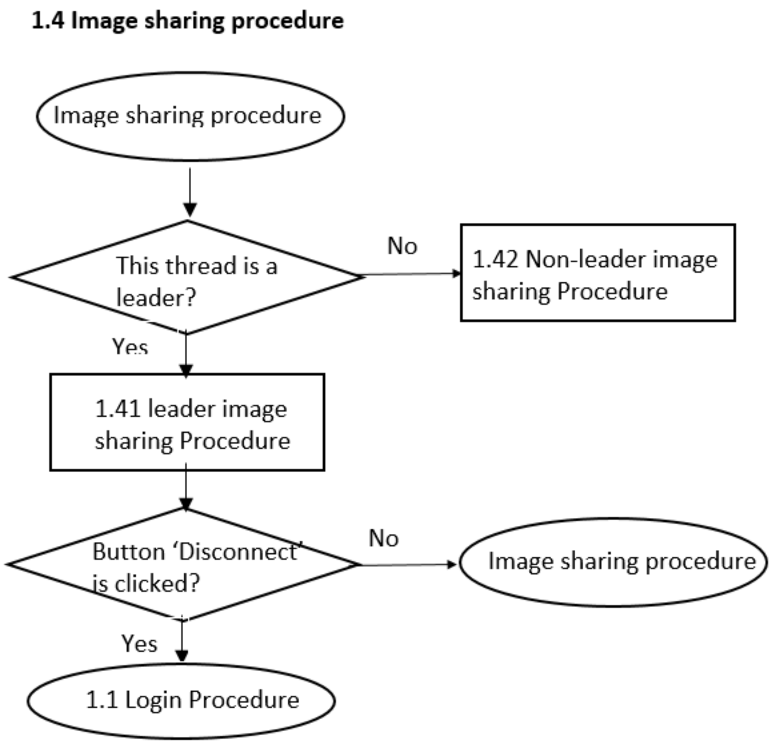

1.4 Image sharing procedure
Image sharing procedure
This thread is a leader?
No
1.42 Non-leader image sharing Procedure
Yes
1.41 leader image sharing Procedure
Button 'Disconnect' is clicked?
No
Image sharing procedure
Yes
1.1 Login Procedure

Figure 2-5. Flowchart of Image Sharing Procedure Overview

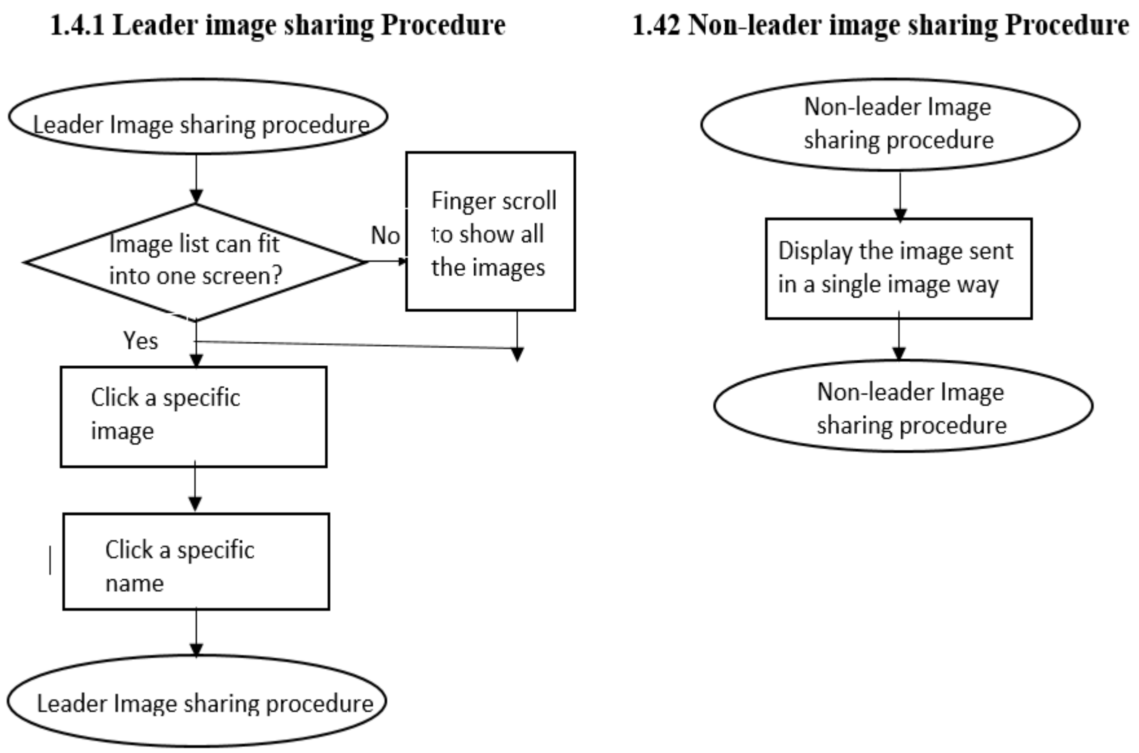

1.4.1 Leader image sharing Procedure
1.42 Non-leader image sharing Procedure
Leader Image sharing procedure
Image list can fit into one screen?
No
Finger scroll to show all the images
Yes
Click a specific image
Click a specific name
Leader Image sharing procedure
Non-leader Image sharing procedure
Display the image sent in a single image way
Non-leader Image sharing procedure

Figure 2-6. Flowchart of Image Sharing Procedure Details

2.1.1.2. Add a new member application

Main aspects:

- Alternative action for team leader:
  - User’s finger swipe starts from the image view screen and go across both leader screen and the new member finger swipe screen.
- Alternative action for new member:
  - Login Procedure
  - Finger swipe Procedure
  - Display Group members Procedure

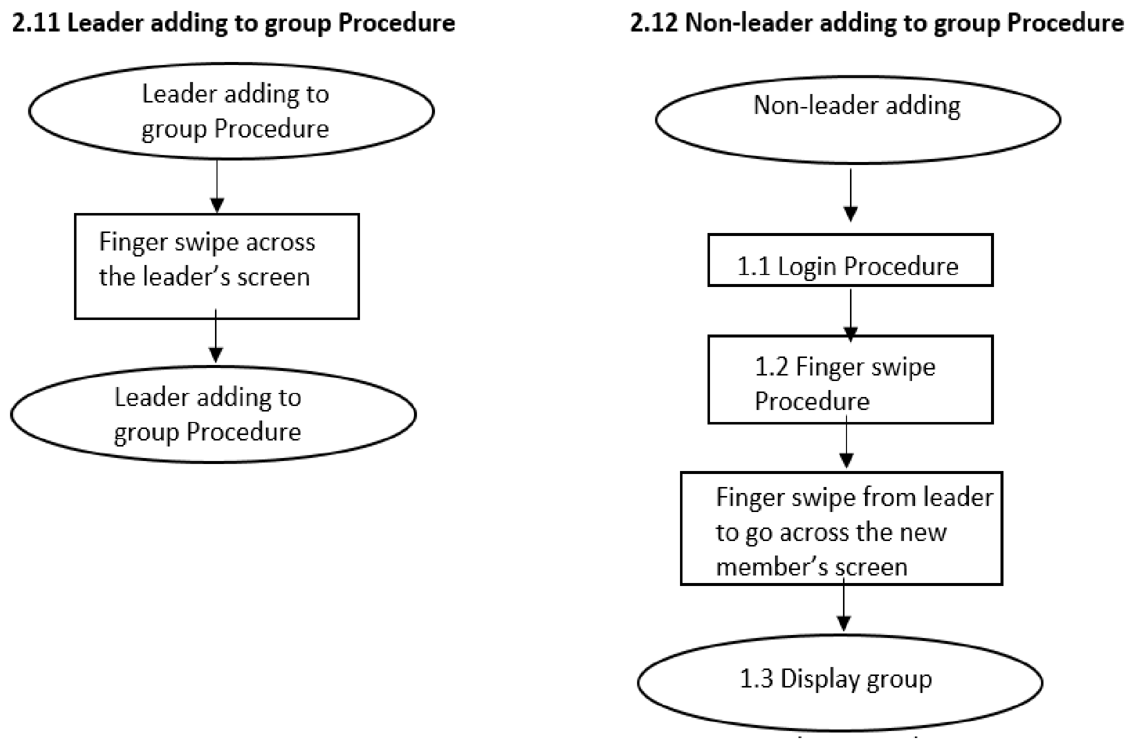
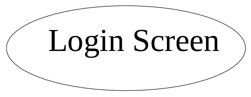

2.11 Leader adding to group Procedure
Leader adding to group Procedure
Finger swipe across the leader's screen
Leader adding to group Procedure
2.12 Non-leader adding to group Procedure
Non-leader adding
1.1 Login Procedure
1.2 Finger swipe Procedure
Finger swipe from leader to go across the new member's screen
1.3 Display group

Figure 2-7. Flowchart of Adding a New Member

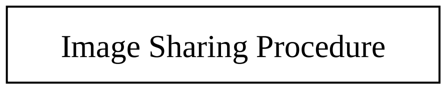
2.1.1.3. Notations Used

2.1.1.3.1. Oval shape (or circle):



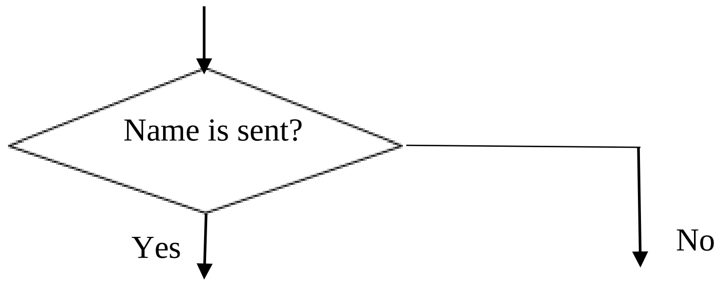
This Symbol is a start or end point of a process or a sub-process.

2.1.1.3.2. Rectangle:



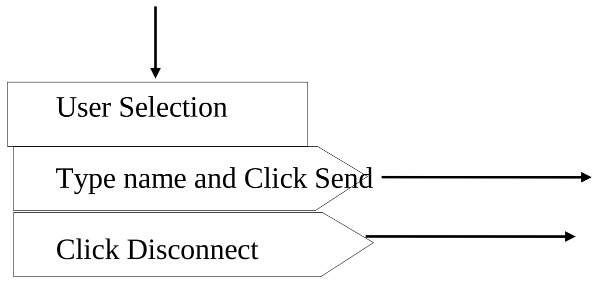
This symbol indicates a process or an application. The number in the rectangle shows the position and level of this block in the whole application. If anyone wants to review the details of this block, he or she can go through its lower level documentation. Such as the lower level documentation for 1.4 is documented in 1.4.1, 1.4.2 or so on.

2.1.1.3.3. Decision point:



This symbol separates the outcomes of the problem in the parallelogram based on the answers of the question. In the simple If-Else statement, the process can move on properly and logically.

2.1.1.3.4. Switch case



This symbol is a little bit complicated than the decision point. For the same problem, there probably are more than 2 reconstructed answers. Based on the specific answer, the system will move on in only one way based on the user's selection.

2.1.1.4. Paring Algorithm

My systems applies a client-server architecture. The server determines which smartphones are paired and implemented with C# in the visual studio. Since users may try to form different groups at the same time, the server calculate and time and the slope of finger movements right before exiting a smartphone and that right after entering the next smartphone. Only the time and slope of two smartphones that are varied within a threshold are considered to be in a group.

<u>Pseudo code</u>

```
Public void addIntoGroup (User newuser){
Mutual-exclusion;
Get latest <List<List<EndPoint>>>groups
    If (groups is not null) {
        Assign int new GroupID and float dif as MaxValue
        Foreach (each group in groups){
            If (the difference of new DOWN time and last UP time is within 1 second
and the
            new DOWN time is later than the last UP time) {
                If(slope difference of new DOWN time and last UP time is within
0.2 and
                less than current dif){
                    NewGroupID = the groupID of this current group;
                    Update dif }
                      }
                  }
        If (newGroupID is not MaxValue){
                  Set the new user's EndPoint into group    }
        Else generate a new group and set the new user's EndPoint as the first one of this
group   }
 else   create groups and assign the new user's EndPoint as the first element
}
```

Note:

The object <List<List<EndPoint>>>groups should be an object of a singleton class.

Two ways to implement:

1. Set groups as a static variable; once the data will be locked once it is read or written
2. Create a singleton class; generate an object from that class; once the data will be locked once it is read or written

2.1.2. Functions Performed

2.1.2.1. Share image Application

2.1.2.1.1. Login Procedure

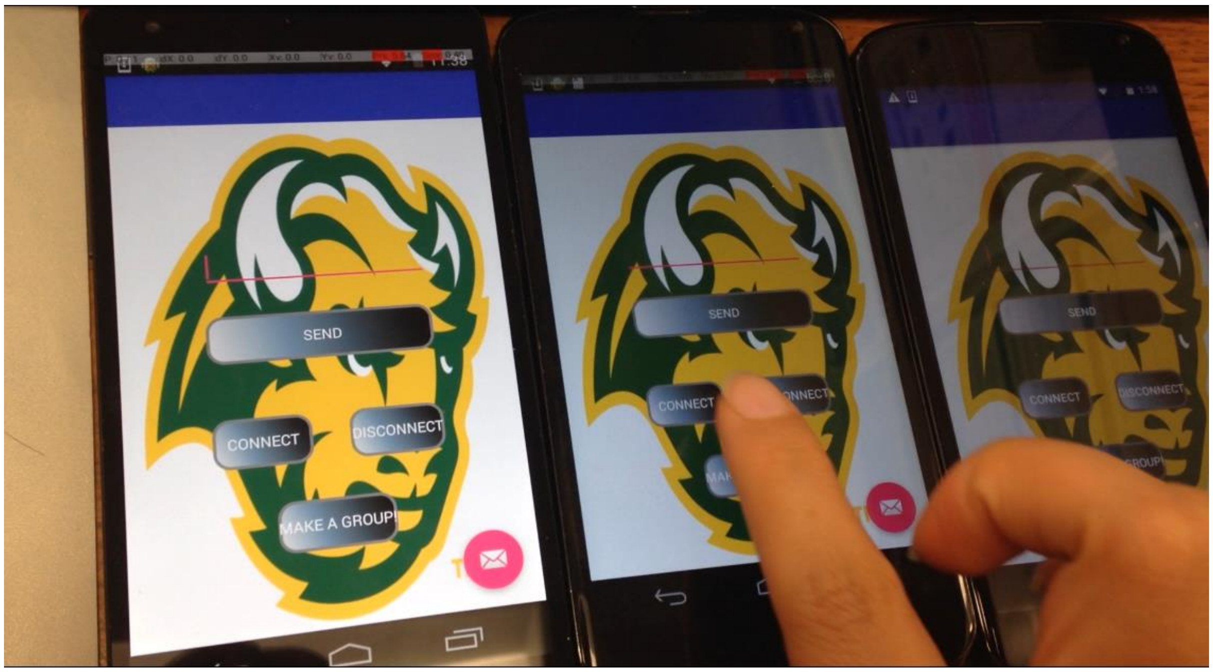


Figure 2-8. Click ‘CONNECT’ Button

User clicks the pre-structured ‘CONNECT’ button to connect the individual cell phone to server.

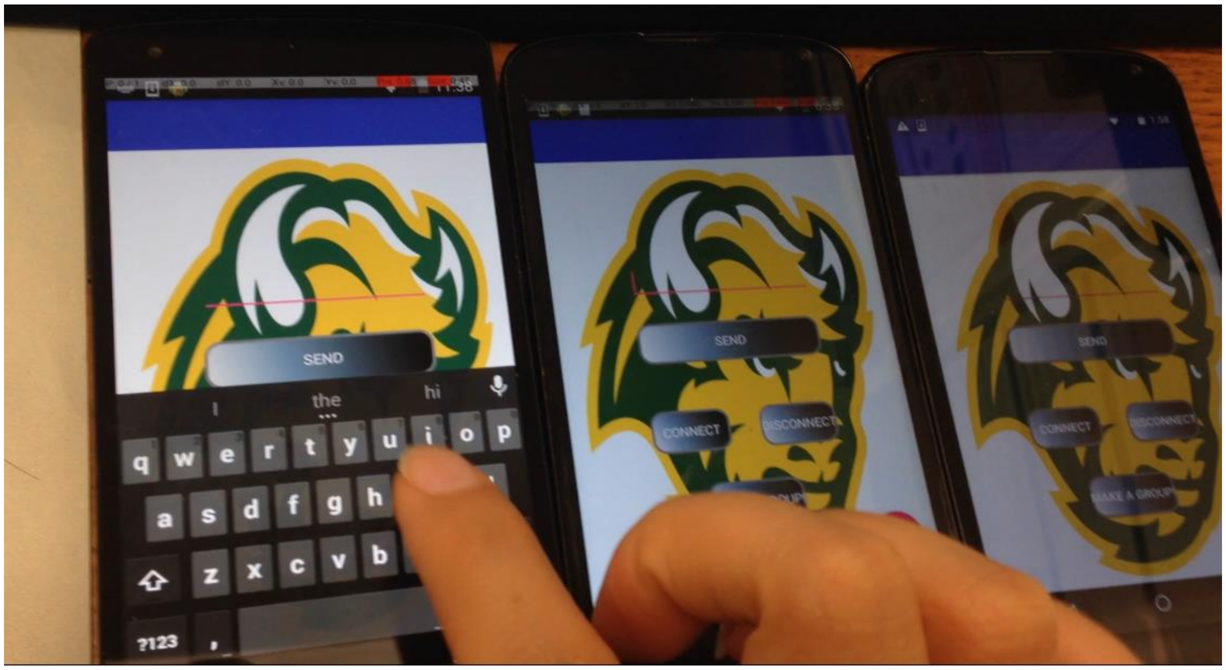


Figure 2-9. Type Name and Click ‘SEND’ Button

User types his or her name and click send button to send the user name to server. And then click the pre-constructed ‘MAKE A GROUP’ Button.

2.1.2.1.2 Finger Swipe Procedure

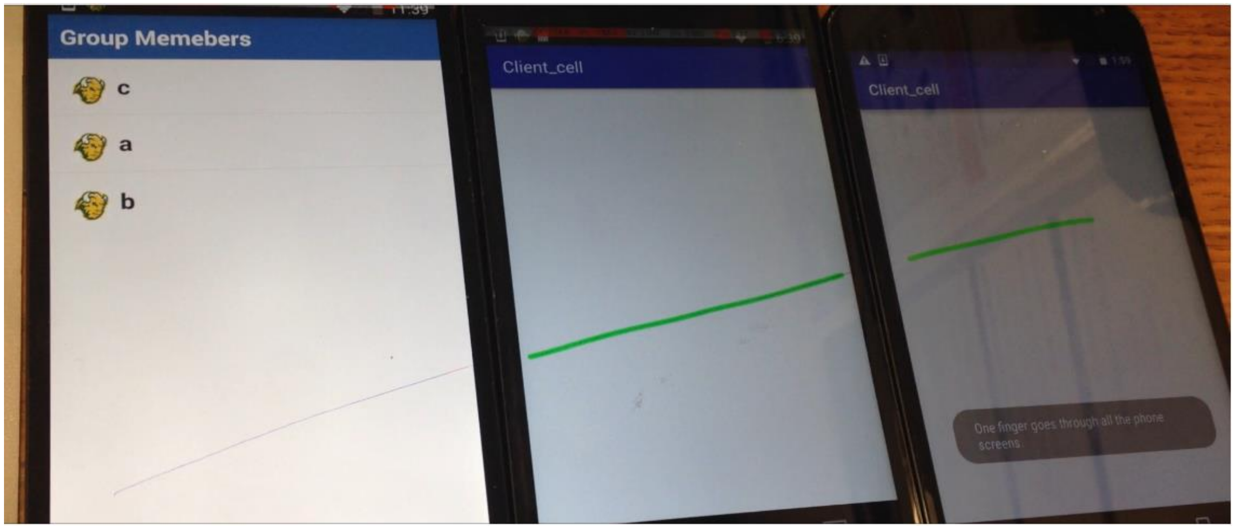


Figure 2-10. Finger Swipe

User swipes the finger across all the screens in a linear line. And the system sends all the data of each screen to server for pairing. Once the server is done, the result will be sent to every group member.

The data collected from every client are the time of the user put the finger on the screen, the time of the user move the finger away from the screen, the slope of the first two points just after the user touch down the screen and the slope of the last two points before the user's finger moves away from the screen.

After manual tests, to help the individual client to join in to the proper group, the minimum threshold for the slope is 0.2 and that for the time difference is 1 second.

2.1.2.1.3 Display group member procedure

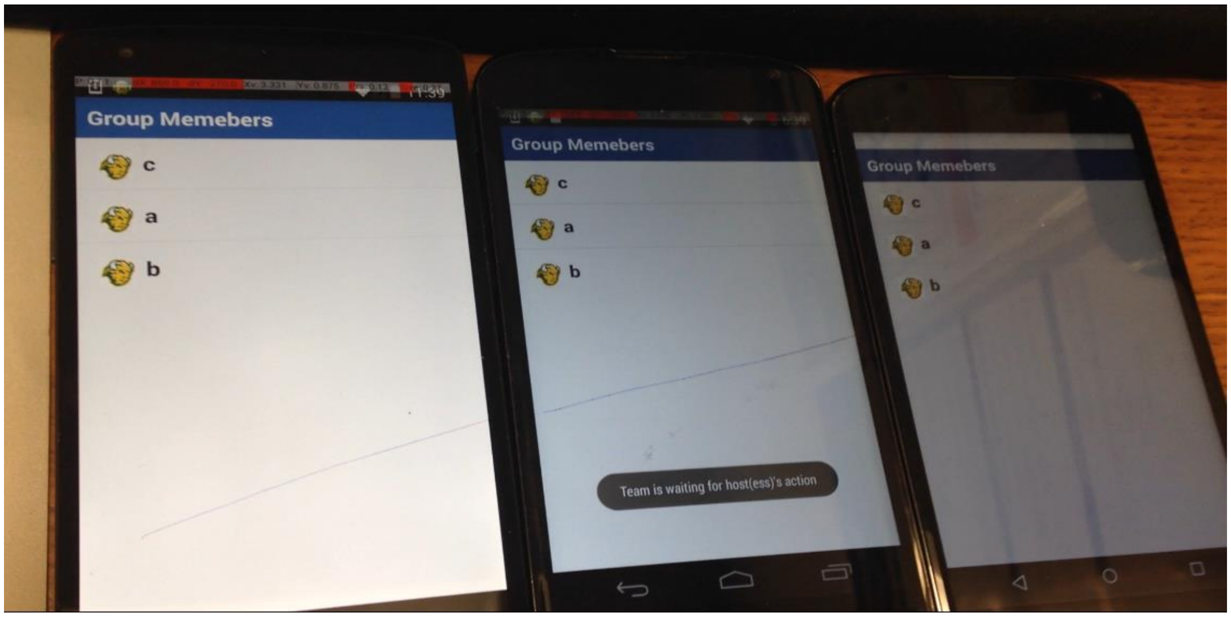


Figure 2-11. Client Displays All the Group Members

Each client keeps listen to the server. If there is any updates, the list view will be updated, that is, the new name list will be displayed. If there are no updates after 2 seconds, each client concludes the group is formed. And the system turns into next Activity.

2.1.2.1.4. Image sharing procedure - Leader

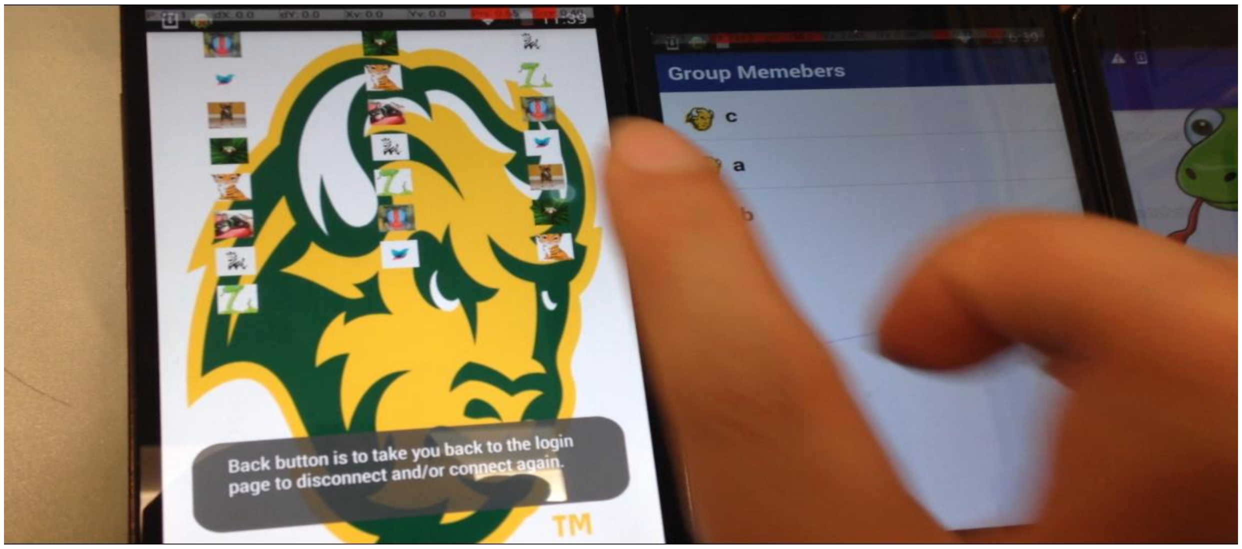

Group Memebers
c
a
b
Back button is to take you back to the login page to disconnect and/or connect again.

Figure 2-12. Leader Displays all the Images in a Predefined Folder

Leader is the Android phone whose screen has been touched first. For the leader of this group, it displays all the images in a predefined folder in a grid view.

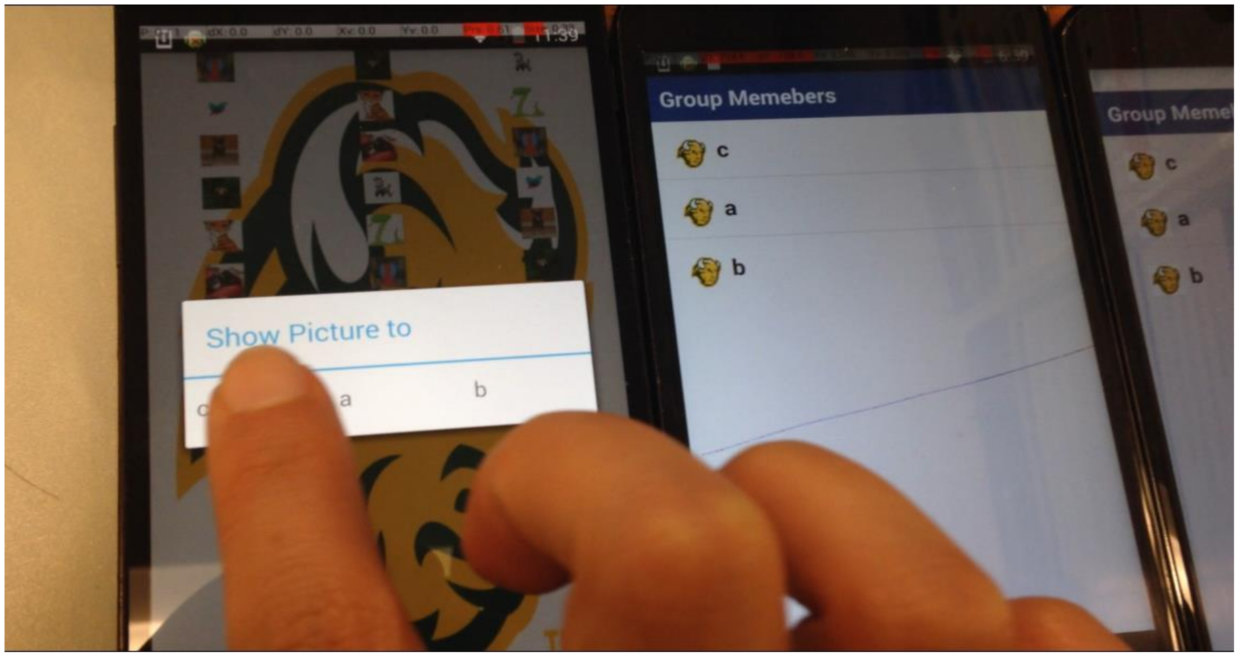


Figure 2-13. Send a Selected Image to a Specific Client

Once the user clicked a specific image and system shows a pop-up window for the user to choose the targeted client. Then the system will send the given image to the given client.

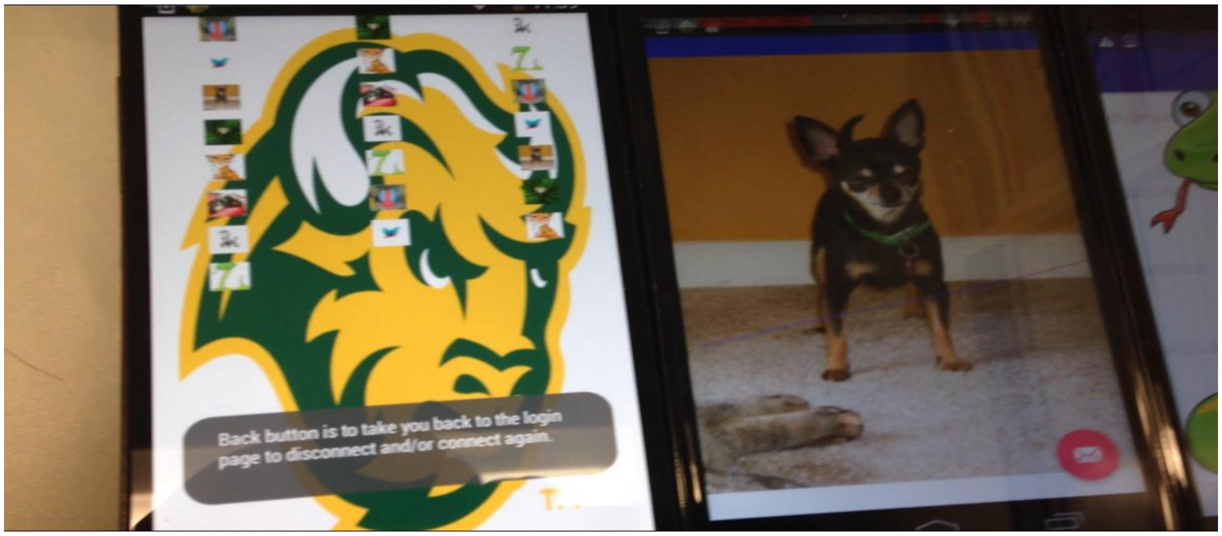

Back button is to take you back to the login page to disconnect and/or connect again.

Figure 2-14. Send a Different Image

For the non-leader side, the system always shows the group member list, and there is a pop-up toast tells the user to wait for the group leader's action.

2.1.2.2. Add a new member application

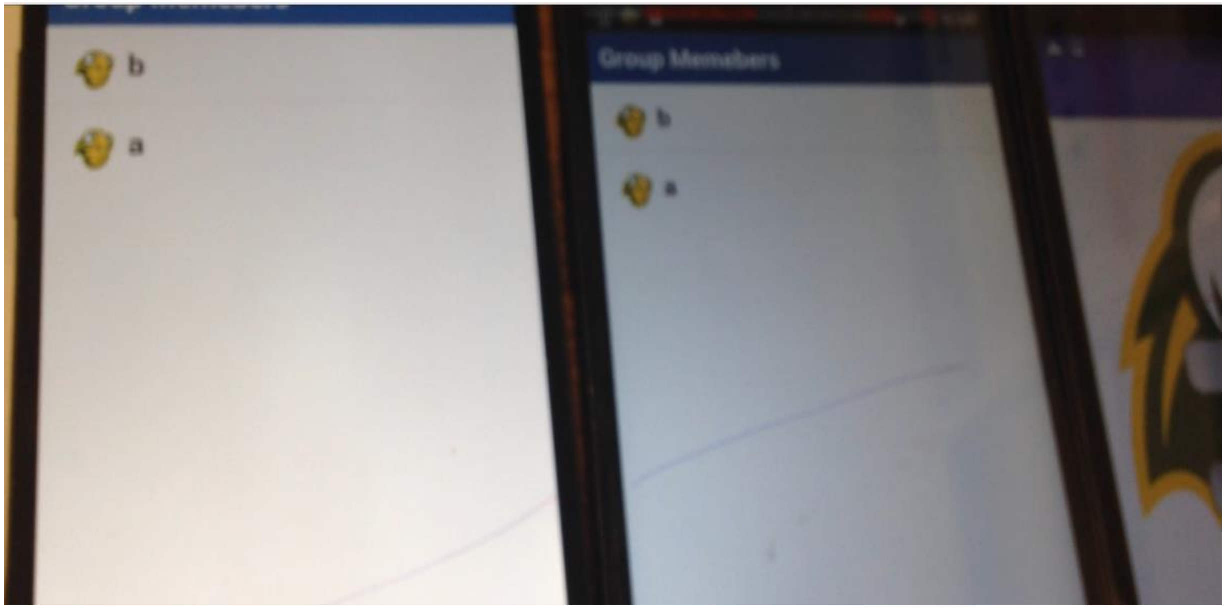


Figure 2-15. Two Members in an Existing Group

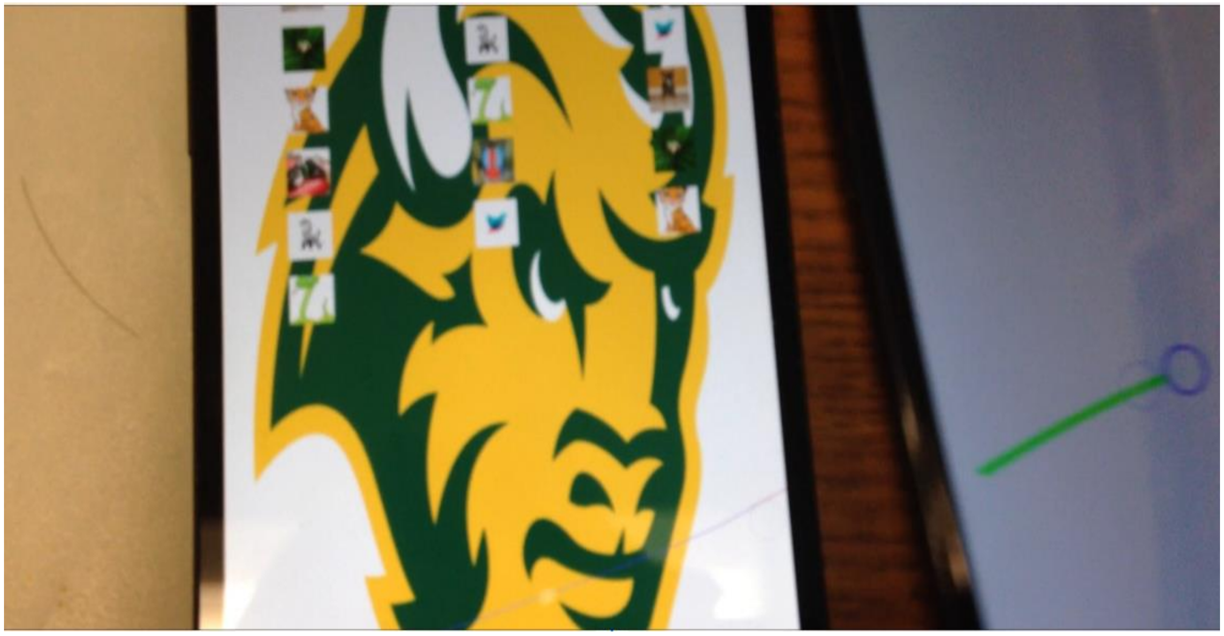


Figure 2-16. A New Member Joins an Existing Group

At the very beginning, there are only 2 members in the group. Once the new client wants to join in the existing group, user can swipe the finger from the leader’s screen to the new comer’s screen. User swipes the finger across all the screens in a linear line.

And the system sends all the data of each screen to server for pairing.

Once the server is done, the result will be sent to every group member.

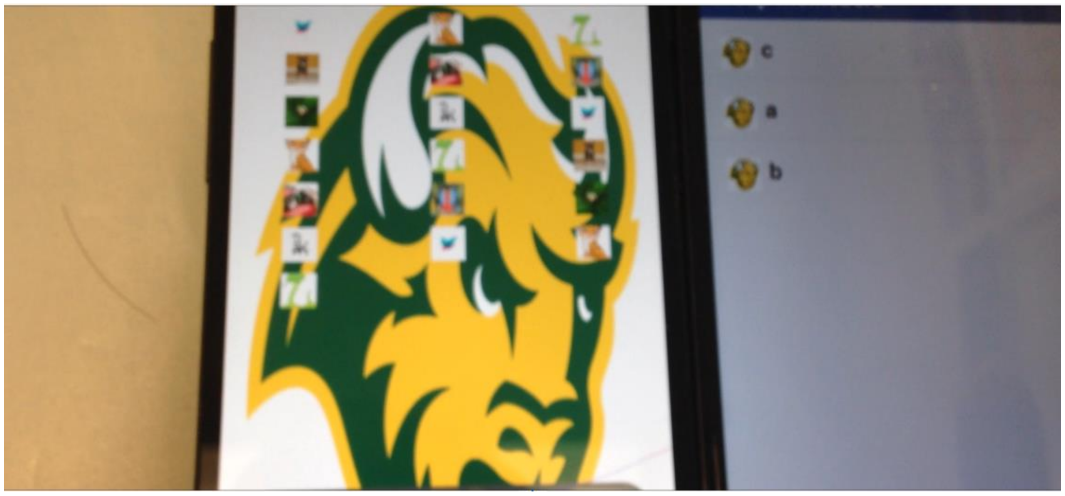


Figure 2-17. A New Member

The new member is added into the group, and the new member's screen display the name list. And also all the client's group member list is updated.

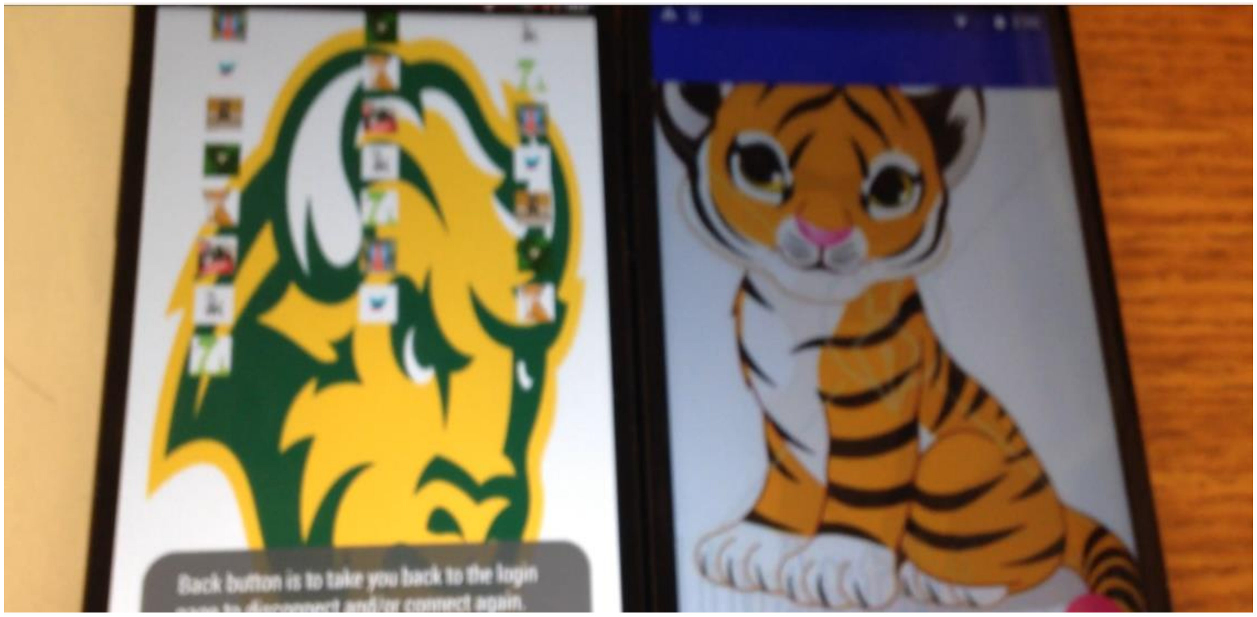

Back button is to take you back to the login page to disconnect and/or connect again.

Figure 2-18. Send an Image to a New Client

2.1.3. Limitations and Restrictions

The results for the threshold are mainly based on tests and adjusted manually, which is hard to prove its accuracy. I designed a Decision tree-based SVM algorithm using WEKA to implement Machine Learning to build up the module to make pairing more accurate and efficient. In the future work, the analysis on the features will be taken to get better module for classification.

2.1.4. User Interface Design

In Android, based on the requirements, lots of User Interface Design patterns are used, such as Android button, LinearLayout, RelativeLayout, EditText, ScrollView, Toast, Dialog, ListView and Image Grid View.

Android button, EditText, LinearLayout and RelativeLayout are all basic layouts. LinearLayout arranges its children in a single direction, it can be vertical or horizontal. RelativeLayout is a view group that displays child views in relative positions. It is a most common types of layouts in android. Android EditText is used to get inputs from user.

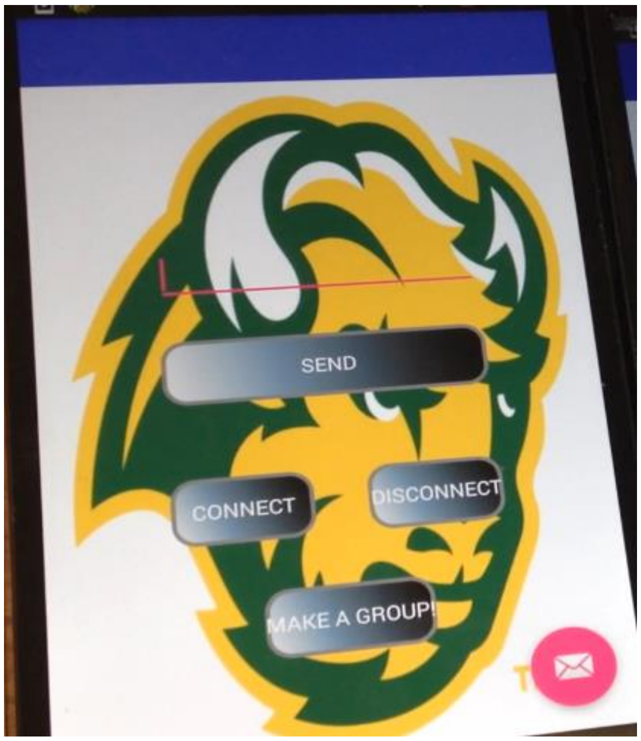


Figure 2-19. Home Screen

ListView is a different kind of layout rather than LinearLayout, RelativeLayout, etc. It is designed to view larger than its actual size. It takes care of its own vertical scrolling and forces to display all the items in the long list in a customized or default style.

Android toast displays simple feedback about an operation in a small popup. Toast is displayed in front of the activity and automatically disappear after a timeout. Normally, android toast is displayed at the bottom of screen, you can also change display position.

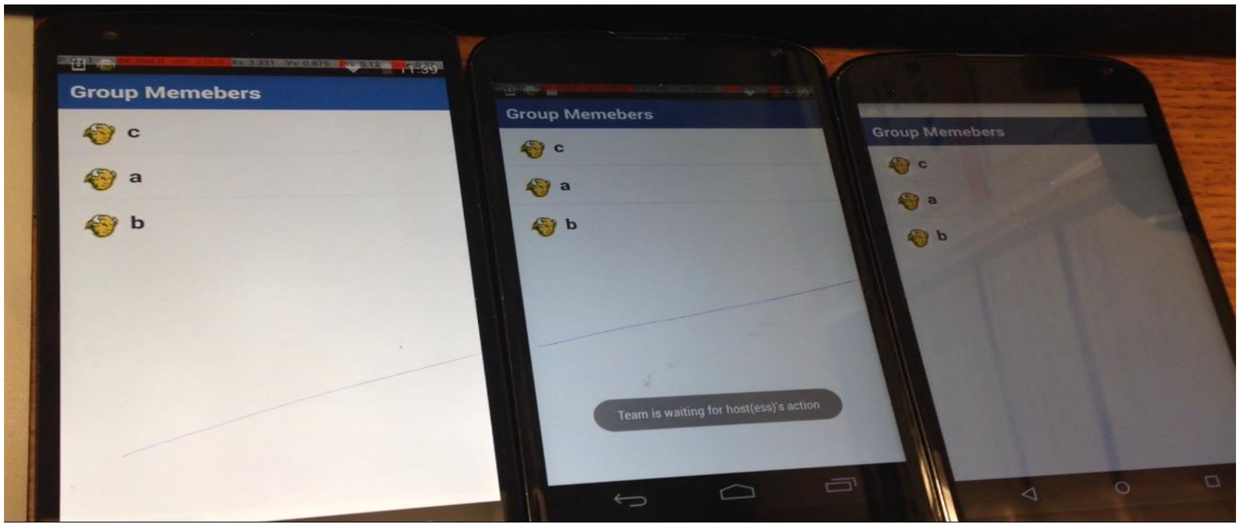


Figure 2-20. ListView and Toast

Android Image View has many features. Image GridView allows users to display image in a grid view. Android Image Single View shows the image full of the whole screen with a customized volume and column sizes.

Dialog is a window that provides information, some decision or asking additional information from the users. Most of the dialog has a title, message and some decision condition. User is required to take a certain action before the system proceeds.

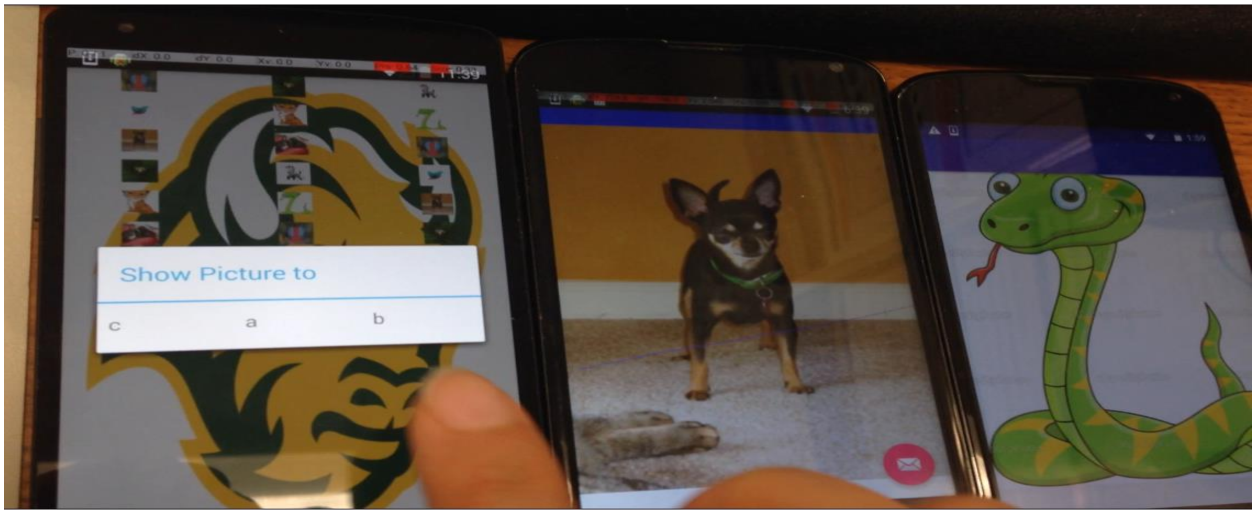


Figure 2-21. Image GridView and Dialog

2.2. Design Specification

2.2.1. Introduction

This is the software design specification for “Sharing information among smartphones through finger swipe” application. The SDS breaks down the project from architecture level till component level to describe in detail what the purpose of each component is and how it will be implemented. The SDS will also be used as a tool for verification and validation of the final delivery.

2.2.2. System Architecture Description

The system combines client-server architecture and MVC design patterns.

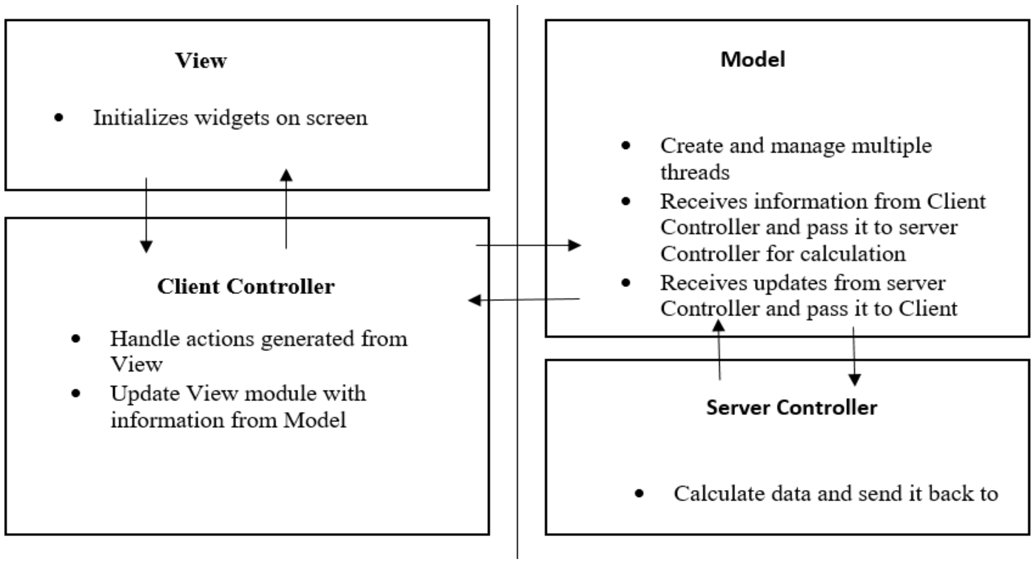
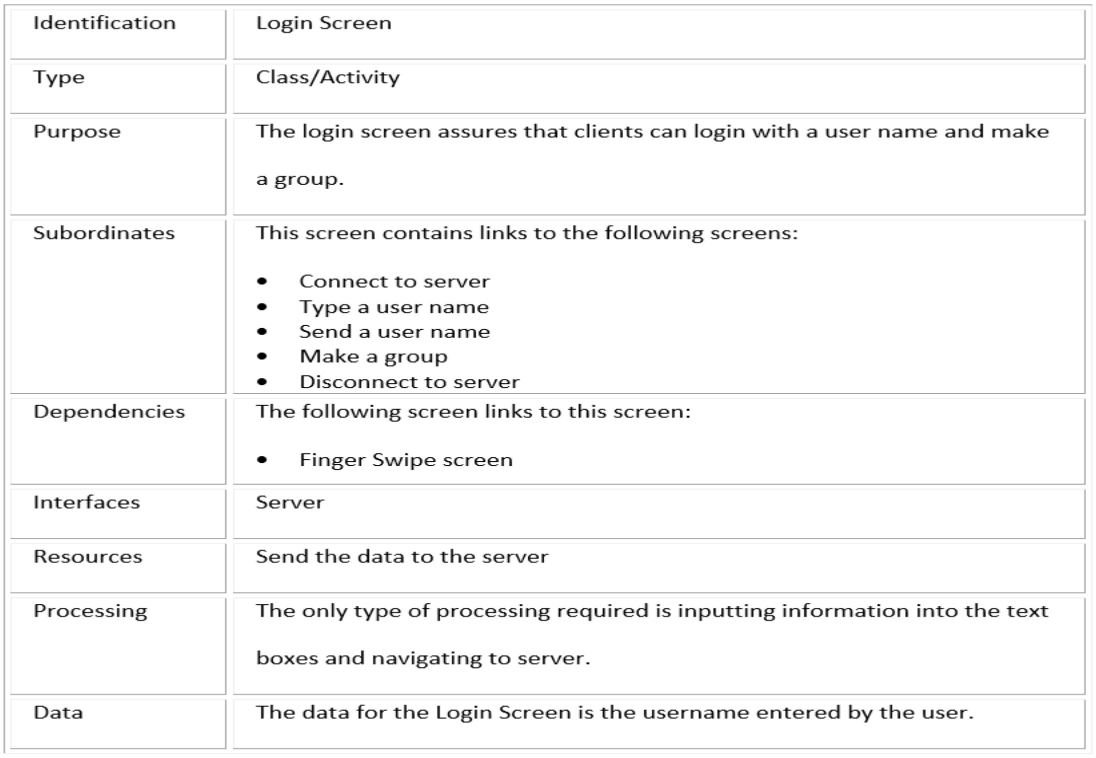

View
Initializes widgets on screen
Model
Create and manage multiple threads
Receives information from Client Controller and pass it to server Controller for calculation
Receives updates from server Controller and pass it to Client
Client Controller
Handle actions generated from View
Update View module with information from Model
Server Controller
Calculate data and send it back to

Figure 2-22. Overview of Architecture

2.2.3. Detailed Description

Component design is based on every screen in detail.

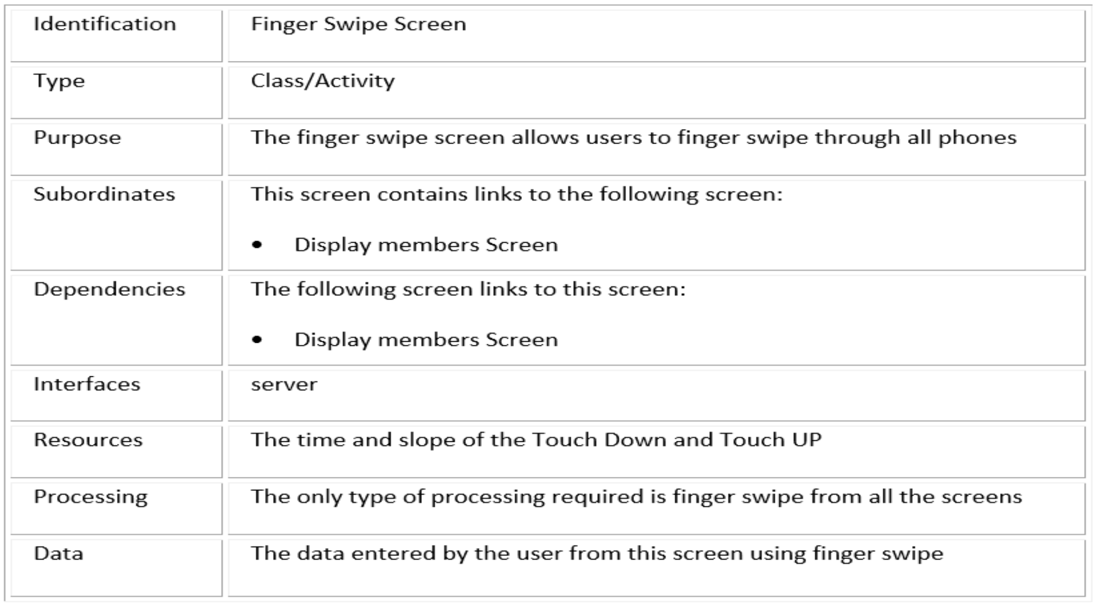
2.2.3.1. Description of Login Screen



Identification	Login Screen
Type	Class/Activity
Purpose	The login screen assures that clients can login with a user name and make a group.
Subordinates	This screen contains links to the following screens: • Connect to server • Type a user name • Send a user name • Make a group • Disconnect to server
Dependencies	The following screen links to this screen: • Finger Swipe screen
Interfaces	Server
Resources	Send the data to the server
Processing	The only type of processing required is inputting information into the text boxes and navigating to server.
Data	The data for the Login Screen is the username entered by the user.

Figure 2-23. Login Screen

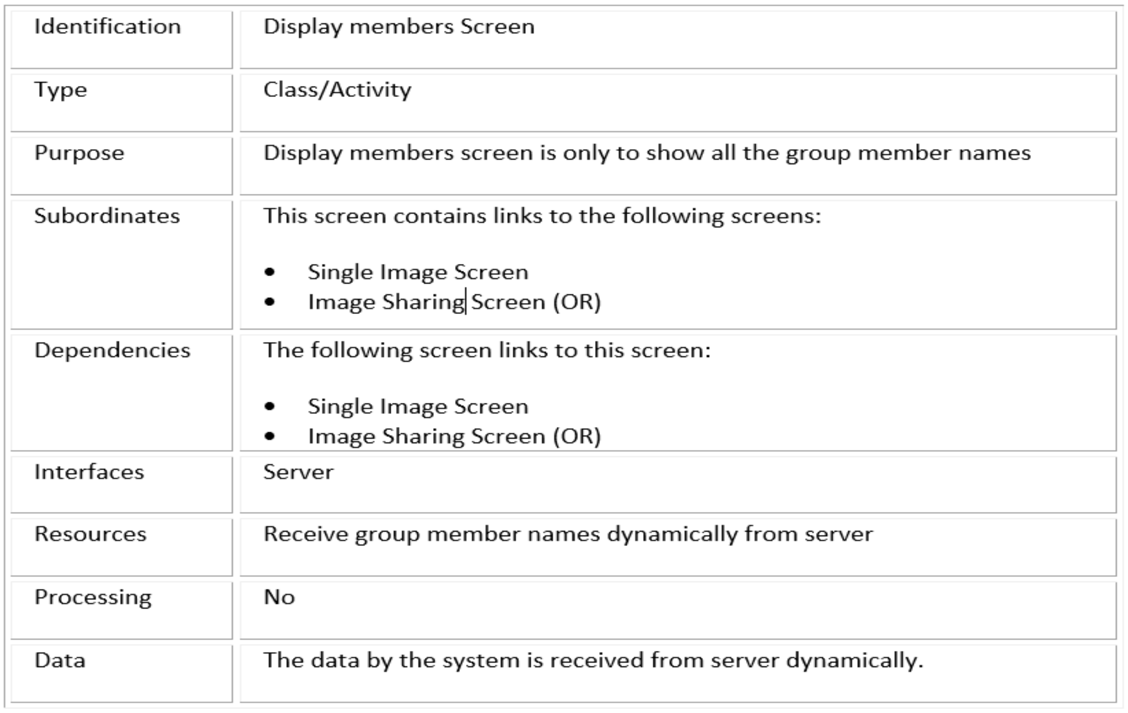
2.2.3.2. Description of Finger Swipe Screen



Identification	Finger Swipe Screen
Type	Class/Activity
Purpose	The finger swipe screen allows users to finger swipe through all phones
Subordinates	This screen contains links to the following screen: • Display members Screen
Dependencies	The following screen links to this screen: • Display members Screen
Interfaces	server
Resources	The time and slope of the Touch Down and Touch UP
Processing	The only type of processing required is finger swipe from all the screens
Data	The data entered by the user from this screen using finger swipe

Figure 2-24. Finger Swipe Screen

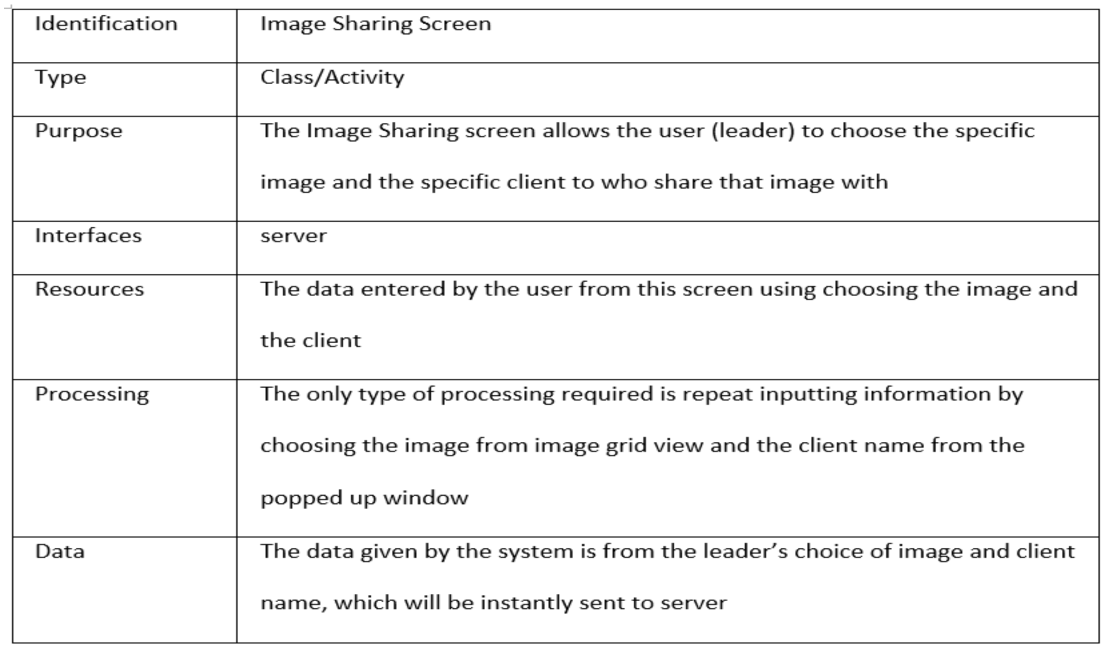
2.2.3.3. Description of Display members Screen



Identification	Display members Screen
Type	Class/Activity
Purpose	Display members screen is only to show all the group member names
Subordinates	This screen contains links to the following screens: • Single Image Screen • Image Sharing Screen (OR)
Dependencies	The following screen links to this screen: • Single Image Screen • Image Sharing Screen (OR)
Interfaces	Server
Resources	Receive group member names dynamically from server
Processing	No
Data	The data by the system is received from server dynamically.

Figure 2-25. Display Members Screen

2.2.3.4. Description of Image Sharing Screen



Identification	Image Sharing Screen
Type	Class/Activity
Purpose	The Image Sharing screen allows the user (leader) to choose the specific image and the specific client to who share that image with
Interfaces	server
Resources	The data entered by the user from this screen using choosing the image and the client
Processing	The only type of processing required is repeat inputting information by choosing the image from image grid view and the client name from the popped up window
Data	The data given by the system is from the leader’s choice of image and client name, which will be instantly sent to server

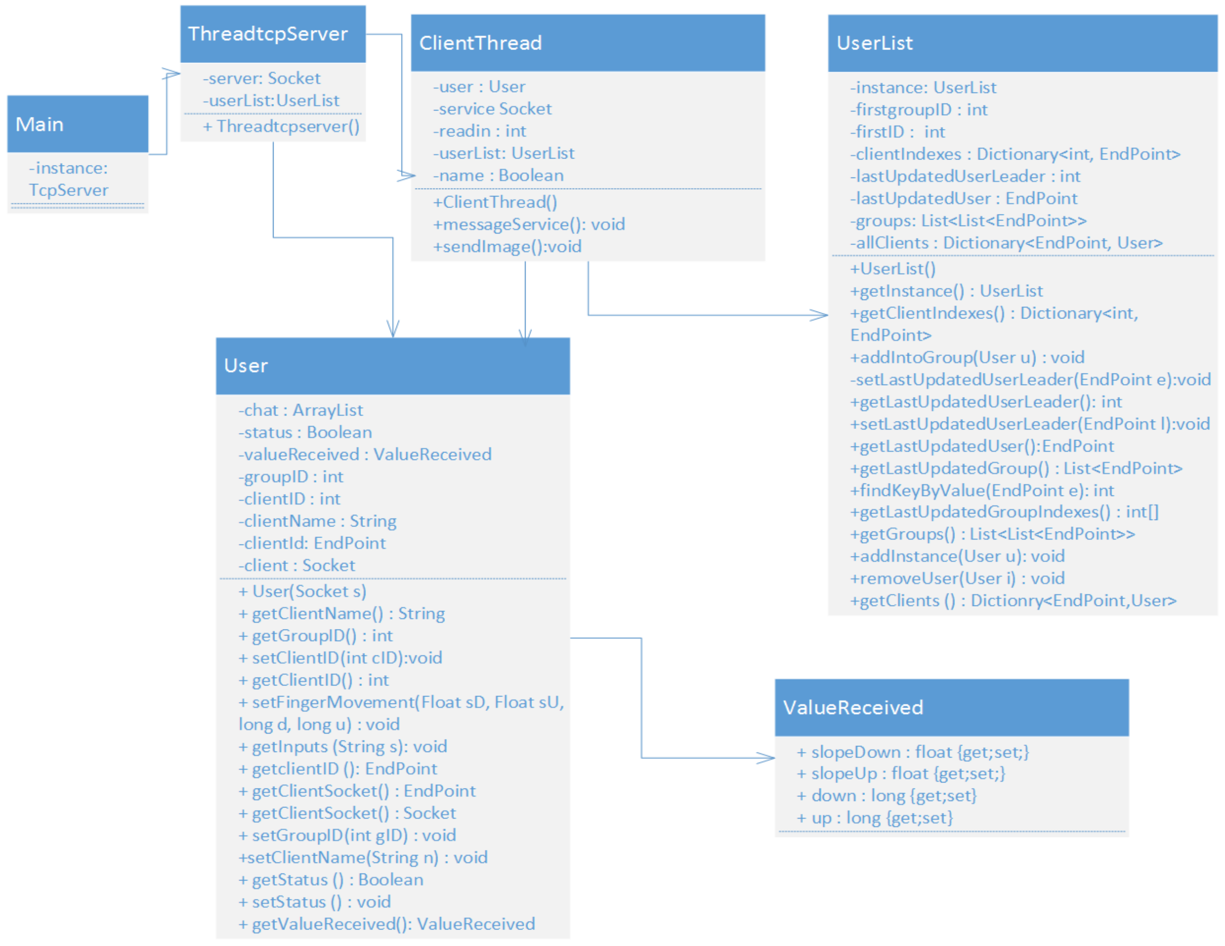
Figure 2-26. Image Sharing Screen

2.2.3.5. Description of Single Image Screen

Identification	Single Image Screen
Type	Class/Activity
Purpose	The single image screen is just for display the image shared from the leader
Interfaces	server
Resources	The data received from the server
Processing	No
Data	The data given by the system is from the server

Figure 2-27. Single Image Screen

2.2.4. Class Diagram Design



Main
-instance: TcpServer
ThreadtcpServer
-server: Socket
-userList:UserList
+ Threadtcpserver()
ClientThread
-user : User
-service Socket
-readin : int
-userList: UserList
-name : Boolean
+ClientThread()
+messageService(): void
+sendImage():void
UserList
-instance: UserList
-firstgroupID : int
-firstID : int
-clientIndexes : Dictionary<int, EndPoint>
-lastUpdatedUserLeader : int
-lastUpdatedUser : EndPoint
-groups: List<List<EndPoint>>
-allClients : Dictionary<EndPoint, User>
+UserList()
+getInstance() : UserList
+getClientIndexes() : Dictionary<int, EndPoint>
+addIntoGroup(User u) : void
-setLastUpdatedUserLeader(EndPoint e):void
+getLastUpdatedUserLeader(): int
+setLastUpdatedUserLeader(EndPoint l):void
+getLastUpdatedUser():EndPoint
+getLastUpdatedGroup() : List<EndPoint>
+findKeyByValue(EndPoint e): int
+getLastUpdatedGroupIndexes() : int[]
+getGroups() : List<List<EndPoint>>
+addInstance(User u): void
+removeUser(User i) : void
+getClients () : Dictionry<EndPoint,User>
User
-chat : ArrayList
-status : Boolean
-valueReceived : ValueReceived
-groupID : int
-clientID : int
-clientName : String
-clientId: EndPoint
-client : Socket
+ User(Socket s)
+ getClientName() : String
+ getGroupID() : int
+ setClientID(int cID):void
+ getClientID() : int
+ setFingerMovement(Float sD, Float sU, long d, long u) : void
+ getInputs (String s): void
+ getclientID (): EndPoint
+ getClientSocket() : EndPoint
+ getClientSocket() : Socket
+ setGroupID(int gID) : void
+setClientName(String n) : void
+ getStatus () : Boolean
+ setStatus () : void
+ getValueReceived(): ValueReceived
ValueReceived
+ slopeDown : float {get;set;}
+ slopeUp : float {get;set;}
+ down : long {get;set}
+ up : long {get;set}

Figure 2-28. Design of C# Server

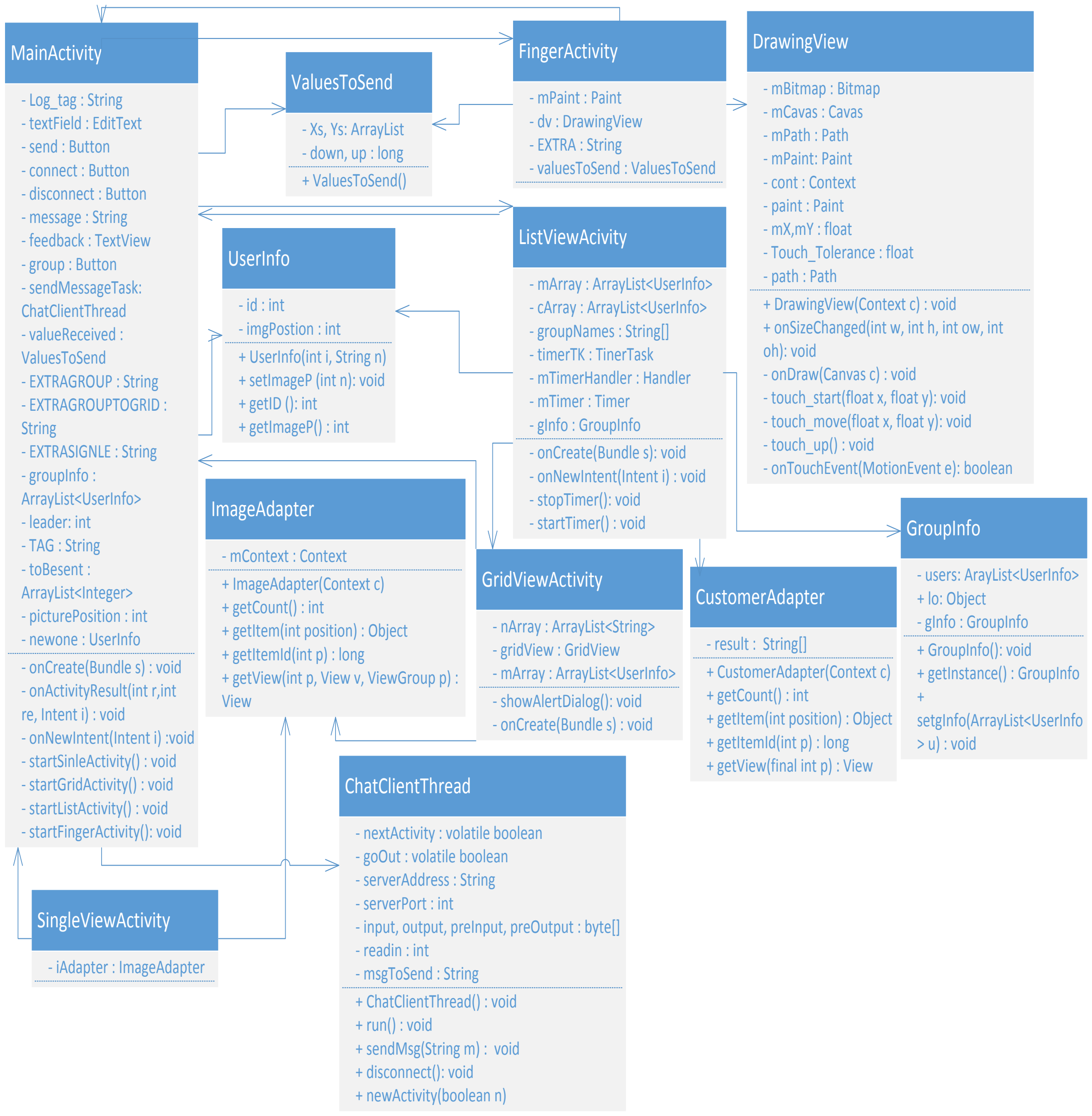

MainActivity
- Log_tag : String
- textField : EditText
- send : Button
- connect : Button
- disconnect : Button
- message : String
- feedback : TextView
- group : Button
- sendMessageTask: ChatClientThread
- valueReceived : ValuesToSend
- EXTRAGROP : String
- EXTRAGROUPTOGRID : String
- EXTRASIGNLE : String
- groupInfo : ArrayList<UserInfo>
- leader: int
- TAG : String
- toBesent : ArrayList<Integer>
- picturePosition : int
- newone : UserInfo
- onCreate(Bundle s) : void
- onActivityResult(int r,int re, Intent i) : void
- onNewIntent(Intent i) :void
- startSinleActivity() : void
- startGridActivity() : void
- startListActivity() : void
- startFingerActivity(): void
ValuesToSend
- Xs, Ys: ArrayList
- down, up : long
+ ValuesToSend()
FingerActivity
- mPaint : Paint
- dv : DrawingView
- EXTRA : String
- valuesToSend : ValuesToSend
DrawingView
- mBitmap : Bitmap
- mCavas : Cavas
- mPath : Path
- mPaint: Paint
- cont : Context
- paint : Paint
- mX,mY : float
- Touch_Tolerance : float
- path : Path
+ DrawingView(Context c) : void
+ onSizeChanged(int w, int h, int ow, int oh): void
- onDraw(Canvas c) : void
- touch_start(float x, float y): void
- touch_move(float x, float y): void
- touch_up() : void
- onTouchEvent(MotionEvent e): boolean
UserInfo
- id : int
- imgPostion : int
+ UserInfo(int i, String n)
+ setImageP (int n): void
+ getID (): int
+ getImageP() : int
ListViewAcivity
- mArray : ArrayList<UserInfo>
- cArray : ArrayList<UserInfo>
- groupNames : String[]
- timerTK : TinerTask
- mTimerHandler : Handler
- mTimer : Timer
- gInfo : GroupInfo
- onCreate(Bundle s): void
- onNewIntent(Intent i) : void
- stopTimer(): void
- startTimer() : void
ImageAdapter
- mContext : Context
+ ImageAdapter(Context c)
+ getCount() : int
+ getItem(int position) : Object
+ getItemId(int p) : long
+ getView(int p, View v, ViewGroup p) : View
GridViewActivity
- nArray : ArrayList<String>
- gridView : GridView
- mArray : ArrayList<UserInfo>
- showAlertDialog(): void
- onCreate(Bundle s) : void
CustomerAdapter
- result : String[]
+ CustomerAdapter(Context c)
+ getCount() : int
+ getItem(int position) : Object
+ getItemId(int p) : long
+ getView(final int p) : View
GroupInfo
- users: ArrayList<UserInfo>
+ lo: Object
- gInfo : GroupInfo
+ GroupInfo(): void
+ getInstance() : GroupInfo
+ setgInfo(ArrayList<UserInfo> u) : void
ChatClientThread
- nextActivity : volatile boolean
- goOut : volatile boolean
- serverAddress : String
- serverPort : int
- input, output, preInput, preOutput : byte[]
- readin : int
- msgToSend : String
+ ChatClientThread() : void
+ run() : void
+ sendMsg(String m) : void
+ disconnect(): void
+ newActivity(boolean n)
SingleViewActivity
- iAdapter : ImageAdapter

Figure 2-29. Design of Android Studio

2.2.5. System Data Flow Diagrams

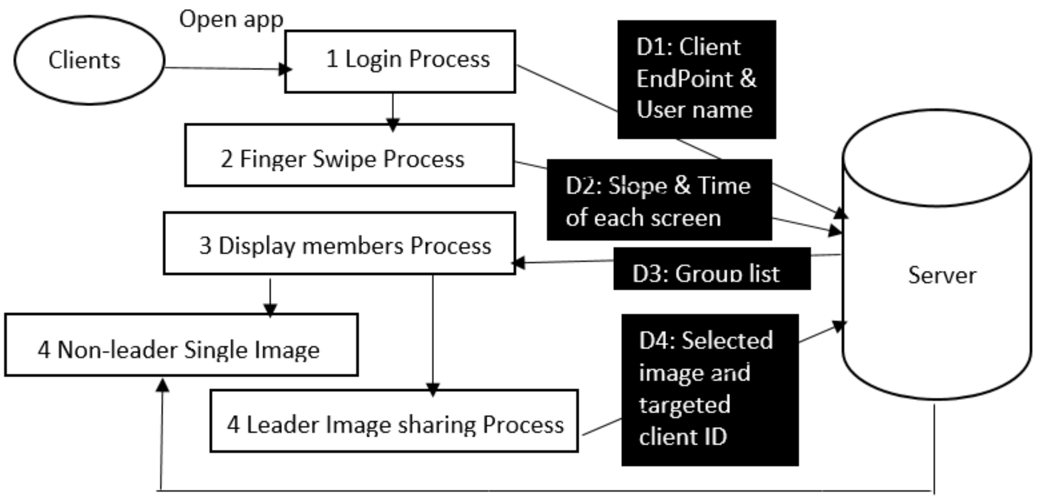

Open app
Clients
1 Login Process
2 Finger Swipe Process
3 Display members Process
4 Non-leader Single Image
4 Leader Image sharing Process
D1: Client EndPoint & User name
D2: Slope & Time of each screen
D3: Group list
D4: Selected image and targeted client ID
Server

Figure 2-30. DFD Overview

2.2.6. Equipment Configuration

Windows 7

Android Studio 1.5 SDK 23

C# visual studio 2010

Server obtains IP address from NDSU campus Ethernet; client visit server IP address via TCP/IP (wifi)

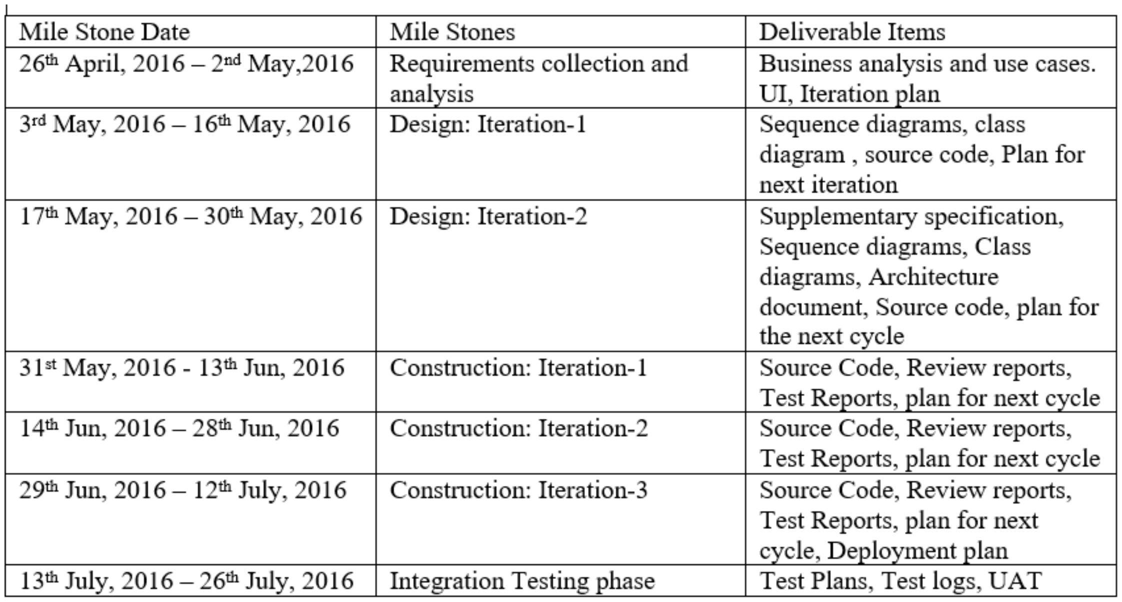
2.2.7. Implementation Languages

C# and Android Studio.

Since this project is a part of a large project with C# as an interface and the client should be android cellphone, I have to work on a multiple platform environment and handle both Java-based language and C#.

2.3. Implementation

2.3.1. Project management & Deliverable Items



Mile Stone Date	Mile Stones	Deliverable Items
26th April, 2016 – 2nd May,2016	Requirements collection and analysis	Business analysis and use cases. UI, Iteration plan
3rd May, 2016 – 16th May, 2016	Design: Iteration-1	Sequence diagrams, class diagram , source code, Plan for next iteration
17th May, 2016 – 30th May, 2016	Design: Iteration-2	Supplementary specification, Sequence diagrams, Class diagrams, Architecture document, Source code, plan for the next cycle
31st May, 2016 - 13th Jun, 2016	Construction: Iteration-1	Source Code, Review reports, Test Reports, plan for next cycle
14th Jun, 2016 – 28th Jun, 2016	Construction: Iteration-2	Source Code, Review reports, Test Reports, plan for next cycle
29th Jun, 2016 – 12th July, 2016	Construction: Iteration-3	Source Code, Review reports, Test Reports, plan for next cycle, Deployment plan
13th July, 2016 – 26th July, 2016	Integration Testing phase	Test Plans, Test logs, UAT

Figure 2-31. Project Timeline

3. EVALUATIONS

3.1. Testing Data

Testing is executed based on the test cases. And test cases are created or added based on user cases listed in 2.1.1 Applications Introduction. For all the test cases, mapping to one or more user cases is very important.

Table 3-1. Login Screen Testing Data

Project Name: Efficient mobile information sharing with finger swipe

Test Tile: Login 1.1   Module: Login Screen

TC#	Test Case	Mapping Design Doc	Test Execution Steps	Input Value	Expected Result	Actual Result	Test Result(P/F)	Comment
1	Login - Positive							
1.1	with a user name	1.12	a. start installed APP					
			b. Click 'CONNECT' Button					
			c. Type name in TextField	Tom				
			d. Click 'SEND' button		New Client : Tom	New Client : Tom	P	
1.2	without a user name	1.13	a. start installed APP					
			b. Click 'CONNECT' Button					
			c. Click 'SEND' button		New Client : anonymous	New Client : anonymous	P	

(Continued)

Table 3-1. Login Screen Testing Data (continued)

TC#	Test Case	Mapping Design Doc	Test Execution Steps	Input Value	Expected Result	Actual Result	Test Result(P/F)	Comment
1.3	'DISCONNECTION' before 'CONNECTION'	1.11	a. start installed APP					
			b. Click 'DISCONNECT' Button					
			c. Click 'CONNECTION' Button					
			d. Type name in TextField	Tom				
			e. Click 'SEND' button		New Client: Tom	New Client: Tom	P	
2	Login - Negative	1.11						
2.1	without connection		a. start installed APP					
			b. Type name in TextField	Tom				
			c. Click 'SEND' button		-	-	P	- means nothing represents on the server side
2.2	'DISCONNECT'	1.11	a. start installed APP					
			b. Click 'DISCONNECT' Button					
			c. Type name in TextField	Tom				
			d. Click 'SEND' button		-	-	P	

(Continued)

Table 3-1. Login Screen Testing Data (continued)

TC#	Test Case	Mapping Design Doc	Test Execution Steps	Input Value	Expected Result	Actual Result	Test Result(P/F)	Comment
2.3	'DISCONNECTION' after 'CONNECTION'	1.11	a. start installed APP					- means nothing represents on the server side
			b. Click 'CONNECT' Button					
			c. Click 'DISCONNECT' Button					
			d. Type name in TextField	Tom				
			e. Click 'SEND' button		-	-	P	

The rest test cases are created and tested according to the same rules.

3.2. Results and Discussion

All the test cases are passed. But the efficiency of manual testing is poor, automation testing is the solution to achieve high efficiency results.

4. CONCLUSION

Though smartphones have been commonly used in our daily life, it is not user friendly to share information among smartphones. My work allows users to share information among smartphones through an intuitive gesture of finger swipe, without customizing network configurations or acquiring knowledge of Bluetooth or pairing the virtual identification with physical cell phone.

The project shows the application of sharing information, but this technique can be applied to other applications, such as brainstorming, augmented reality, cooperative shopping, or active learning.

Brainstorming. In large collaborative tasks, our application allows team members share their ideas with other team members. The collected ideas can be displayed on each member's mobile device. Then, a group member can vote for those ideas through finger gestures (such as right finger swipe for yes and left for no).

Augmented reality. For example, when users are travelling a foreign country, our application can provide augmented reality to introduce landmarks by connecting multiple devices together. Specifically speaking, after connecting two mobile devices together, one device can take a live streaming of a landmark while the other one provides corresponding introductions in the language that the users are familiar.

Cooperative shopping. If we have to purchase a lot of products, we will separate the shopping list and distribute them to different people. Our application improves the shopping efficiency by dividing shopping list to different members in a group.
Active learning. An instructor can assign different topics to different students based on their backgrounds or progresses.

5. REFERENCES

[Aya00] Y. Ayatsuka, N. Matsushita, and J. Rekimoto, “HyperPalette: a Hybrid Computing Environment for Small Computing Devices”, Proc. CHI '00 Extended Abstracts on Human Factors in Computing Systems, pp.133-134, 2000.

[Bor10] S. Boring, D. Baur, A. Butz, S. Gustafson, and P. Baudisch, “Touch Projector: Mobile Interaction Through Video”, Proc. CHI’10, 2010.

[Dac09] R. Dachselt and R. Buchholz, “Natural throw and tilt interaction between mobile phones and distant displays”, CHI Ext. Abstracts, 2009.

[Har08] R. Hardy and E. Rukzio, “Touch & Interact: Touch-based Interaction of Mobile Phones with Displays”, Proc. MobileHCI’08, pp.245-254, 2008.

[Hin03] K. Hinckley, “Synchronous Gestures for Multiple Persons and Computers”, Proc. UITS’03, pp.149-158, 2003.

[Hin04] K. Hinckley, G. Ramos, F. Guimbretiere, P. Baudisch, and M. Smith, “Stitching: Pen Gestures that Span multiple Displays”, Proc. AVI’04, 2004.

[Hut11] W. Hutama, P. Song, C. Fu and W. Goh, “Distinguishing Multiple Smart-Phone Interactions on a Multi-touch Wall Display using Tilt Correlation”, Proc. CHI, 2011.

[Iwa03] Y. Iwasaki, N. Kawaguchi, and Y. Inagaki, “Touch-and-Connect: A Connection Request Framework for Ad-hoc Networks and the Pervasive Computing Environment”, Proc. PerCom’03, 2003.

[Mad04] A. Madhavapeddy, D. Scott, R. Sharp, and E. Upton, “Using Camera-Phones to Enhance Human-Computer Interaction”, Proc. UbiComp’04, 2004.

[Mau08] A.J. Maunder, G. Marsden and R. Harper, “SnapAndGrab – Accessing and sharing contextual multi-media content using Bluetooth enabled cameraphones and large situated displays”, Proc. CHI, 2008.

[Pea09] N. Pears, D.G. Jackson, and P. Oliver, “Smart phone interactions with registered displays", IEEE Perv. Comp., 8:14–21, 2009.

[Pen09] C. Y. Peng, G. B. Shen, Y. G. Zhang, and S. W. Lu, “Point&Connect: Interaction-based Device Pairing for Mobile Phone Users”, Proc. the 7th international conference on Mobile systems, applications, and services, pp.137-150, 2009.

[Rek97] J. Rekimoto, “Pick-and-Drop: A Direct Manipulation Technique for Multiple Computer Environments”, Proc. UIST’97, pp.31-39, 1997.

[Rek03] J. Rekimoto, Y. Ayatsuka, and M. Kohno, “SyncTap: An Interaction Technique for Mobile Networking”, Proc. 5th International Symposium on Mobile HCI, LNCS 2795, pp.104-115, 2003.

[Sch10] D. Schmidt, F. Chehimi, E. Rukzio and H. Gellersen, “PhoneTouch: A Technique for Direct Phone Interaction on Surfaces”, Proc. UIST, pp.13-16, 2010.

[Sey13] T. Seyed, M. C. Sousa, F. Maurer, and A. Tang, “SkyHunter: A Multi-Surface Environment for Supporting Oil and Gas Exploration”, Proc. ITS’13, pp.15-22, 2013.

[Tan01] P. Tandler, T. Prante, C. Muller-Tomfelde, N. Streitz, and R. Steinmetz, “ConnecTables: Dynamic Coupling of Displays for the Flexible Creation of Shared Workspaces”, Proc. UIST’01, pp.11-20, 2001.

[Wil07] A. Wilson and R. Sarin, “BlueTable: Connecting Wireless Mobile Devices on Interactive Surfaces Using Vision-Based Handshaking”, Proc. Graphics Interface 2007, pp.119-125, 2007.