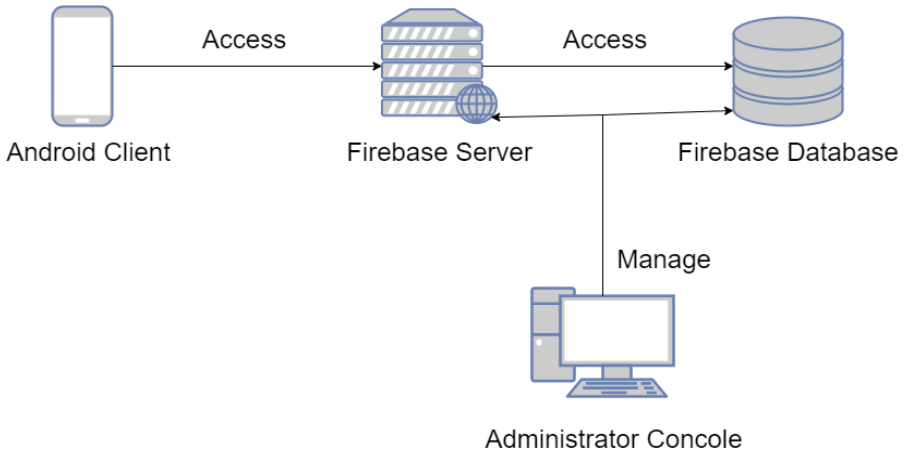
# Comp4342 Mobile Computing
# Project Report

| **Group Member Information Group 10** | |
|---|---|
| **Name** | **Student ID** |
| Wong Ting Ho | 21027973d |
| Yu Ying Ho | 21028689d |
| Li Pak Ho | 21018865d |
| Wong Fan Chun | 21027676d |
| **Project Title:**<br>Talksafe instant messaging system | |

# Table of content

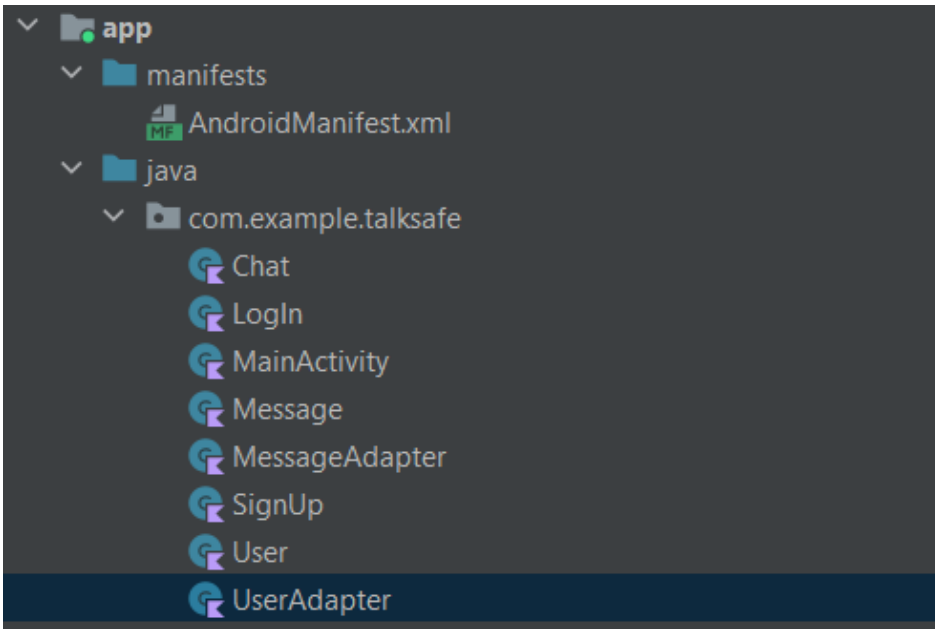
# System structure and components

## System structure:




This graph represents the system structure. The system consists of a client application which is used by the user, a firebase server which clients can connect to, a Firebase database which the server gets information from and a console which is used to manage both the server and database.

## System components:




The system consists of 8 kotlin classes as listed above. The functions of them are listed below:

### Login class:

The Login class is an authentication class that is used to verify users. The user must enter the correct email and password to get past this screen. As the Login class will authenticate the information with the Firebase server. The user can also click forget password to receive an email and reset the password.

### SignUp class:

The SignUp class is used to create the user account to the Talksafe server. The user must enter his name, email and password to create an account. The SignUp class will request the Firebase server to create an account for the User.

### User class:

The User class is an object that stores user information including email, name and UID. Other classes can use this class as an object to store user information.

### UserAdapter class:

The UserAdapter class is an adapter used to communicate between MainActivity class and Chat class. When the user clicks on one of the friends in the friend list. This class can also be used as an object to get the selected friend's name and UID from the recycler list in MainActivity class and send it to Chat class.

### MainActivity class:

The MainActivity class acts as the friend list for the application. This class allows the user to add any other users in the system as a friend as long as the user has the email of the friend. This class will access the Firebase database and retrieve user data based on the entered email address. After adding users to the friend list. the user can choose one of the friends to start a conversation. The MainActivity class also accesses the UserAdapter class to communicate with the Chat class. The user can also log out if he/she wants to switch to another account.

### Message class:

The User class is an object that stores user information including message, senderID and the boolean timed to determine whether the message is a timed. If the message is timed, the timeLimit variable will be used to store the time limit of the message.

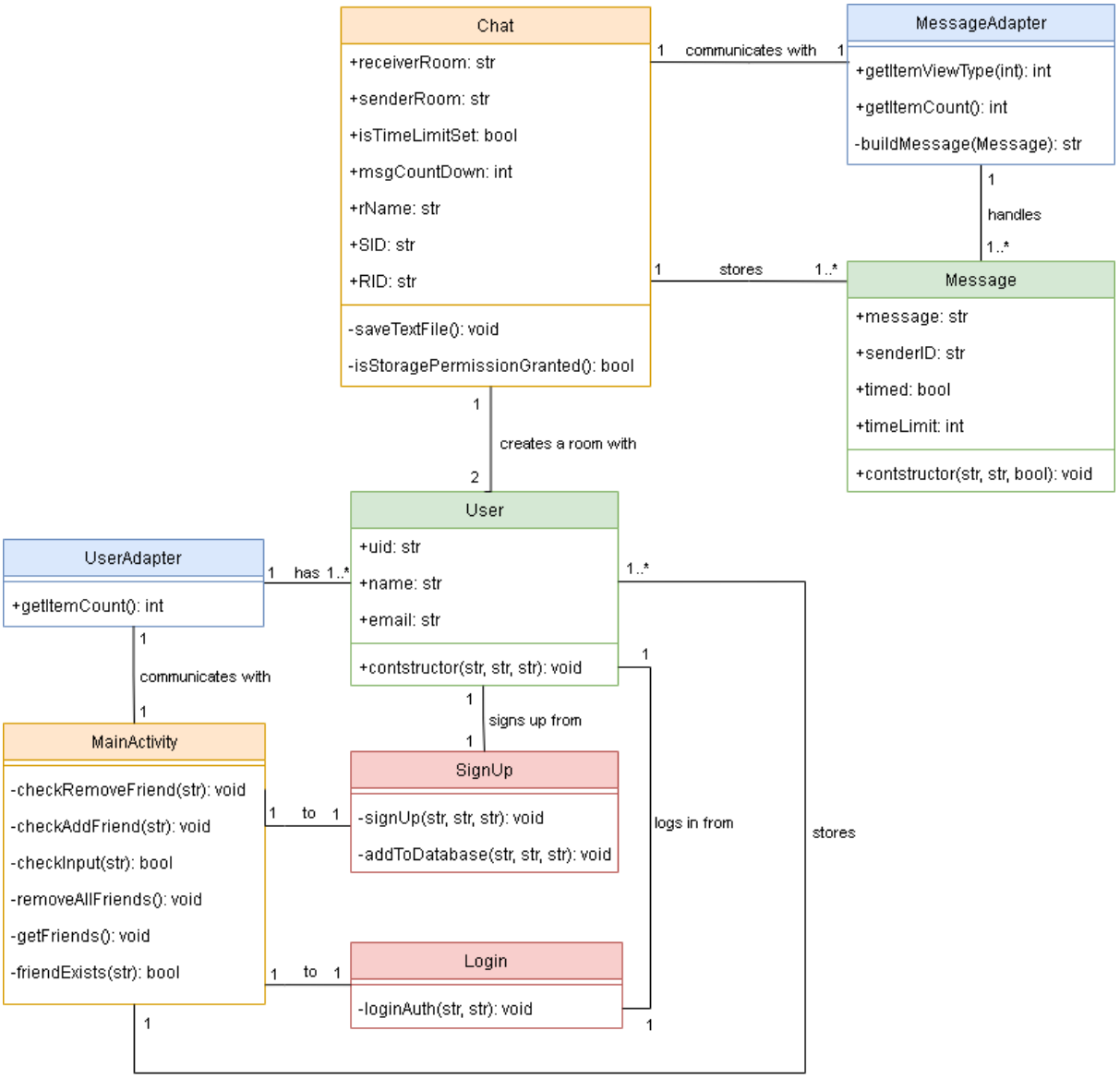
## MessageAdapter class:

The MessageAdapter class is an adapter that handles all messages. The adapter will distinguish between received messages and sent messages using the senderID in the Message class. The adapter will then apply different layouts to the received messages and sent messages. Also, if a message is timed, the adapter will add a “[]” mark in front of the message to indicate the message as a timed message.

## Chat class:

The chat class is the class that handles all messages sent and received by the user. This class will connect to the Firebase Database and retrieve information about messages from there. This class will get receiver information form the UserAdapter class. This class also handles the timer for timed messages received by the user, the timer will start when the user receives a timed message. If the user receives more than one timed message, the timer will focus on the first sent timed message first. After finishing the countdown, the timer will move on to the next message. The user can also export a txt file which consists of all currently available messages to the android phone storage regardless of type.

# Class diagram and descriptions




User objects can be created via SignUp, logged into MainActivity via Login, and is stored in both UserAdapter and MainActivity. MainActivity class manages all functionalities of users such as adding and removing friends as well as error checking.

The Chat class handles all messaging requests in the application. It creates a chat room for 2 user objects to communicate with each other by messages. When a user sends a message, a Message object is created and handled via MessageAdapter for displaying the message in the chat room. The Chat class also communicates with MessageAdapter for handling requests such as exporting messages to a text file.

# Database description

## Firebase

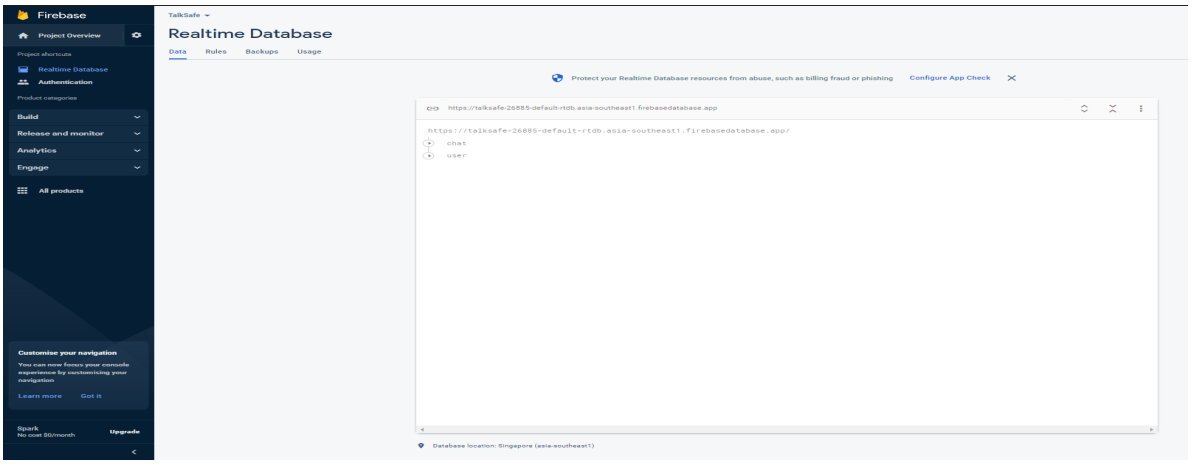



Figure 3.1, Overview of Firebase Realtime Database system

In Figure 3.1, it shows that we are using the Realtime Database of Firebase service as our database. According to the official website of Firebase, this Database is a cloud-hosted database and data is stored as JSON and synchronized in realtime to every connected client. This database will be concerned about two data objects, which store User information and Chat information.

## User Info


Figure 3.2, the basic architecture of user object

After user signing up the account, database will store the user's information immediately and random generate UID value as the unique key, which will be unique and not null.

In Figure 3.2, it shows the basic architecture of the user object that is stored in the database. For the account with name "Silver", it will be the default user information that appears in the database after the user creates this account; For the account with name "Patrick", it will have the history of talk with other users, so there is a "friends" object to store the target. In terms of the password of each account, it also will be encrypted and stored in the database to validate their account. The user can request to reset his password within the application if it is forgotten. But the passwords are not allowed to be read by everyone.

## Chat Info

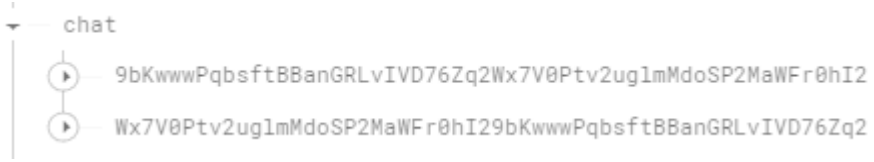



Figure 3.3.1, two chat object created by two user

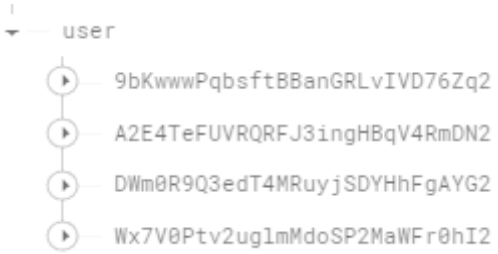



Figure 3.3.2, the current user list

In the chat object, two subclasses will be generated once the people start talking with others who have never talked before. For the subclass ID, it will be following the pattern that combines two user's uid, and order by receiver uid then sender uid. For example, the top subclass ID in the chat will be 9bKwww……Fr0hI2 in Figure 3.3.1, which is combines the first user(9bK……Zq2) and the last user(Wx7…… hI2) in Figure 3.3.2.


Figure 3.4, the detail information under the subclass in chat object

In Figure 3.4, it shows the elements that appear under the subclass in chat object, and there are three default elements – message, senderID, timed and one extend element – timeLimit, which is depend on timed element is true or false. For the description of each elements, message will store the message sent by user; senderID will store the uid of the sender; timed will store the boolean of the message is/isn't consist timer factor; timeLimit will store the second of message that will be disappear after readed by the receiver.


Figure 3.5, Comparison with two subclasses with same users

In Figure 3.5, it shows the comparison with two subclasses with the same users. In view of the case that the message has only one sender in the chat room, there are two messages sended by the user(9bK……Zq2), the difference between two messages is whether they include the timer element or not. On the receiver side, the receiver can read all the messages in the app instantly, but the message that has a timer will be automatically deleted by the application after he/she is reading the message. In the beginning, the realtime database will store all the messages unless the message already time out after the receiver is reading then the database will delete the record of that message. Therefore, there is only one message that is stored in the database from the receiver side.

# Programming languages and tools used

## Programming languages:

Kotlin is the main language used in the project. As Kotlin is designed to be used alongside Java, some Java code is also included in the project for optimization purposes.

## Tools used:

### Android Studio:

Android studio is the official development environment used to develop Android applications. This application will provide developers with APIs that allow them to build an Android app with ease.

### Firebase:

Firebase is a hosting service which allows the use of a real time database for server side applications. This service is free of charge and provides good data security. As Firebase is provided by Google, it provides a Base64 hash function for account passwords as follow:

```
hash_config {
  algorithm: SCRYPT,
  base64_signer_key:
Bz2JePtfQyZJADHvYZHwg6T1Tx1dLWCax0hzkHKSzufR0Z6GEWjJc5qhMjSk11Xz4uKa95rxHIMBZggIpJHyog==,
  base64_salt_separator: Bw==,
  rounds: 8,
  mem_cost: 14,
}
```

This hash function can be modified by users for enhanced security, but currently the function provides 8 rounds which is already enough for our application.

# Testing strategies and results

As the system involves both back end and front end. We have decided to use test cases to test out different functions.

## Test case “SignUp”:

**Starts at Home Screen**

1. Press the “SIGN UP” button, user sent to SignUp Screen

**In SignUp Screen**

1. Enter Username
2. Enter Email
3. Enter Password
4. Press “SIGN UP” button
5. SignUp successful, user sent to Friend list Screen
6. Test case completed

Exceptions:

1. Either username or email or password is empty -> Sign up fail with empty field
2. Enter wrong email structure -> Sign up fail with invalid email address
3. Enter registered email address -> Sign up fail with email address already used
4. Enter password with less than 6 characters -> Sign up fail with less password length
5. Re-entered password does not match with password

## Test case “Login”:

**Starts at Home Screen**

1. Enter email
2. Enter password
3. Press “LOGIN” button
4. Login successful, user sent to Friend List Screen
5. Test case completed

Exceptions:

1. Either email or password is wrong or empty -> login fail

## Test case “Forget Password”:

**Starts at Home Screen**

1. Enter the email address
2. Press “FORGET PASSWORD” button
3. Test case completed

Exceptions:

1. Email address is empty or no match with database record -> forget password action fail

## Test case “Add friend”:

**Starts at Home Screen:**

1. Do the test case “Login” with successful, user sent to Friend List Screen

**In Friend List Screen**

1. Enter Email of friend
2. Press “ADD” button
3. Add the user to friend list
4. List friend list
5. show message "$username added to your friend list"
6. Test case completed

Exceptions:

Press “ADD” button with blank input of email -> show message "Please enter user email"

Press “ADD” button with enter not signed up email -> show message "No user found with this email "

Press “ADD” button with enter added email -> show message "$username is already in your friend list"

## Test case “Remove friend”:

**Starts at Home Screen:**

1. Do the test case “Login” with successful, user sent to Friend List Screen

**In Friend List Screen**

1. Enter Email of friend
2. Press  “REMOVE” button
3. Remove the user to friend list
4. List friend list
5. Test case completed

Exceptions:

1. Press  “REMOVE” button with blank input of email -> show message "This user is not in your friend list"
2. Press  “REMOVE” button with enter not signed up email -> show message "This user is not in your friend list"

## Test case “Remove all friends”:

**Starts at Home Screen:**

1. Do the test case “Login” with successful, user sent to Friend List Screen

**In Friend List Screen**

1. Press “MENU” button
2. Press “Remove All Friends” button
3. Remove all user to friend list
4. List friend list
5. Test case completed

## Test case “Send chat”:

**Starts at Home Screen:**

1. Do the test case “Login” with successful, user sent to Friend List Screen
2. Choosing chat target in the Friend List Screen, user sent to Chat Room Screen

**In Chat Room Screen:**

1. Enter message’s string
2. Press “SUBMIT” button
3. Add message data
4. Relist chat list
5. Test case completed

Exceptions:

1. Press “SUBMIT” button with blank input -> show message "Please type a message before sending it"

## Test case “Send timed chat”:

1. Enter message’s string
2. Enter time
3. Press “SUBMIT” button
4. Relist chat list
5. Remove message when time is up
6. Test case completed

Exceptions:

1. Press “SUBMIT” button with blank input -> show message "Please type a message before sending it"

## Test case “Remove all messages”:

**Starts at Home Screen:**

1. Do the test case “Login” with successful, user sent to Friend List Screen
2. Choosing chat target in the Friend List Screen, user sent to Chat Room Screen

**In Chat Room Screen:**

1. Press “MENU” button
2. Press “Remove chat” button
3. Delete chat data
4. Go back to Friend list
5. Test case completed

## Test case “Export chat”:

**Starts at Home Screen:**

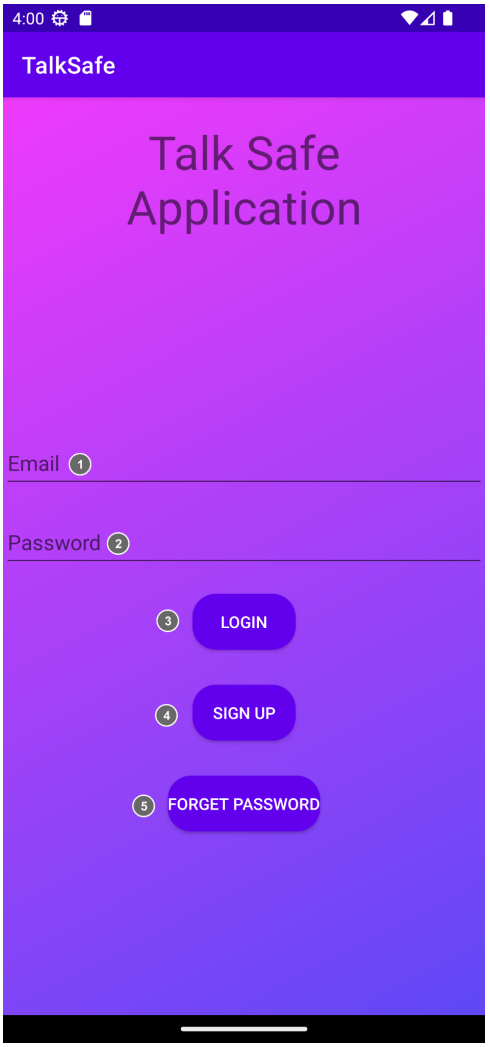
1. Do the test case “Login” with successful, user sent to Friend List Screen
2. Choosing chat target in the Friend List Screen, user sent to Chat Room Screen

**In Chat Room Screen:**

1. Press “MENU” button
2. Press “Export chat” button
3. Generate .txt file and write chat data
4. Save the file in "TalkSafe/messages"
5. show message "Information saved to SD card"
6. Test case completed

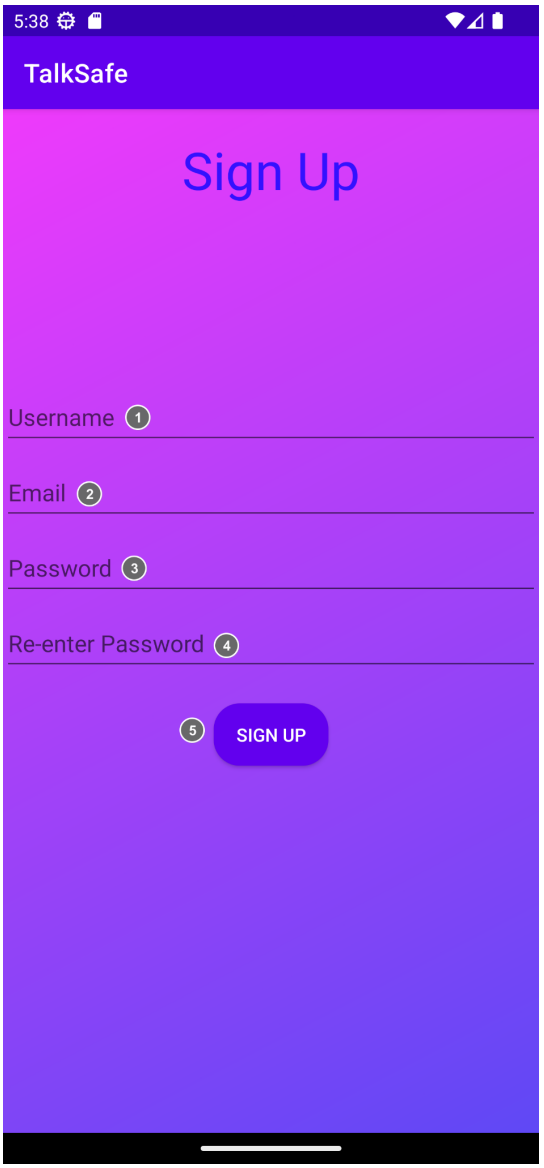
# User Manual

## Home Screen



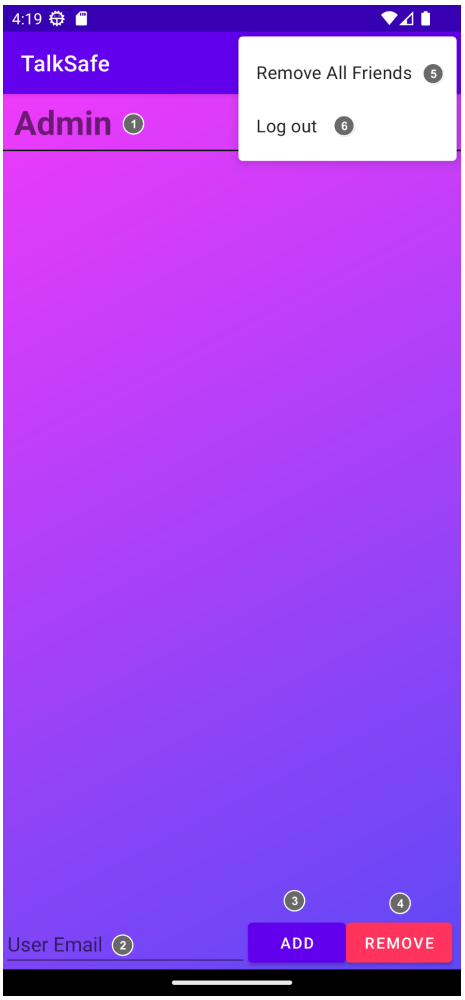

1. Email field: user can enter email here
2. Password field: user can enter password here
3. Login button: user can press this button to log in, both the email field and the password field must not be empty. After logging in, the user will be taken to the Friend list screen
4. Sign Up button: user can press this button to sign up for a new account, this will take the user to the Sign up screen
5. Forget Password button: user can press this button to receive a email to reset password, the email field must not be empty

## Sign up Screen



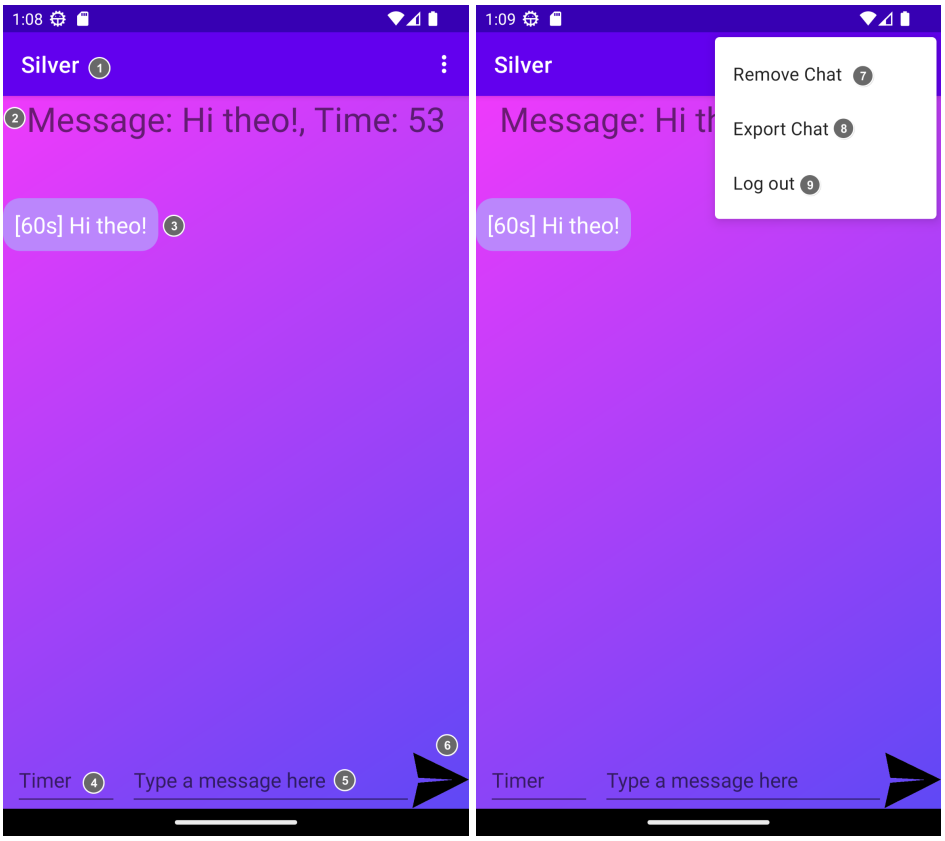

1. Username field: user can enter username here
2. Email field: user can enter email here, the email must be valid
3. Password field: user can enter password here
4. Re-enter password field: user must re-enter the password here to confirm password
5. Sign Up button: user can press this button to sign up for a new account, all fields above must not be empty. After successful account creation, the user will be taken to the Friend list Screen

## Friend list Screen




1. Friend list: user can choose a friend to talk to after adding him/her
2. User email field: user can enter friend's email here to add him/her as a friend
3. Add button: user can press this button to add a friend, the User email field must not be empty
4. Remove button: user can press this button to remove a friend, the User email field must not be empty
5. Remove all friends button: user can press this button to remove all friends from the friend list
6. Logout button: user can press this button to log out. After logging out, user will be sent to the Home Screen

## Chat room Screen




1. Receiver's name: this is the name of the receiver
2. Time remaining message: this message notify the user the remaining time of the current timed message
3. Timer field: user can enter a value between 1 to 60 in seconds to send a timed message, the receiver will have the corresponding seconds to read this message.
4. Message: this is a message in the chat. There are 2 kinds of message: timed and normal. This message is a timed message indicated by the "[60s]" mark in front of the message
5. Message field: user can enter a message and send it to the receiver
6. Send button: user can press this button to send either timed or normal message to the receiver depending on whether the timer field is empty. The message field must not be empty
7. Remove chat button: user can press this button to remove the entire chat room including all messages regardless of type. User will be sent to the Friend list Screen after.
8. Remove chat button: user can press this button to export all current messages regardless of type to a .txt file, the file will be saved in the local storage of the phone.
9. Logout button: user can press this button to log out. After logging out, user will be send to the Home Screen

# Roles and contributions

| Name | Roles | Contribution |
|---|---|---|
| Wong Ting Ho | Project version control, Backend developer, Application logic designer | 25% |
| Yu Ying Ho | Backend developer, Application logic designer | 25% |
| Li Pak Ho | Layout designer, Tester | 25% |
| Wong Fan Chun | Backend Developer, Tester | 25% |