ITIS/ITCS 5180 Mobile Application Development
In Class Assignment 13

Basic Instructions:

1. In every file submitted you MUST place the following comments:
   a. Assignment #.
   b. File Name.
   c. Full name of all students in your group.
2. Each group should submit only one assignment on behalf of all the other group members.
3. Your assignment will be graded for functional requirements and efficiency of your submitted solution. You will loose points if your code is not efficient, does unnecessary processing or blocks the UI thread.
4. Export your Android project and create a zip file which includes all the project folder and any required libraries.
5. Submission details:
   a. Compress the contents of your project folder. The file name is very important and should follow the following format: **Group#_InClass13.zip**
   b. Only one group member is required to submit on behalf of the whole group.
   c. You should submit the assignment through Canvas: Submit the zip file.
6. **The minimum SDK version should be set to 20, and the target SDK version should be set to 25.**
7. **Failure to follow the above instructions will result in point deductions.**

In Class Assignment 13 (100 points)

In this assignment you will develop "MessageMe." The App allows the user to send and receive messages from other users. The app uses Firebase to manage the user authentication and mailbox management.

Part A: Login (10 points)

This is the launcher screen of you app. The wireframe is shown in Figure 1(a). The requirements are as follows:

1. The user should provide their email and password. The provided credentials should be used to authenticate the user using Firebase Authentication. Clicking the "Login" button should submit the login information to the Firebase Authentication API to verify the user's credentials.
   a) If the user is successfully logged in then start the Chat Screen, and finish the Login Screen.
   b) If the user is not successfully logged in, then show a toast message indicating that the login was not successful.
2. Clicking the "New User?" button should start the Signup Screen Figure 1(b), and finish the login Screen.

Part B: SignUp (10 points)

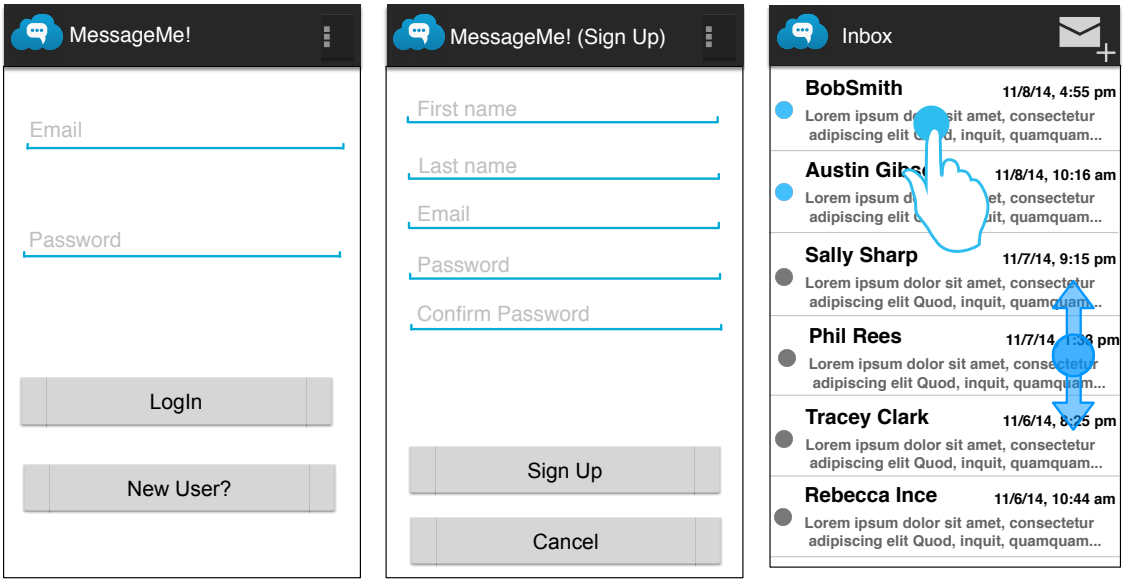
Create the Signup screen to match Figure 1(b), with the following requirements:

1. Clicking the "Cancel" button should finish the Signup Screen and start the Login Screen.
2. The user should provide their first name, last name, email, password and password confirmation. Preform the required validation(the given password and the repeated password must match). Clicking the "Sign Up" button should submit the user's information to Firebase Authentication API.
   a) If the signup is not successful display an error message indicating the error message received from the Firebase Authentication API.
   b) If the signup is successful, then start the Chat Screen and finish the Signup Screen.

To enable the TA to test your app, you should create all the accounts listed in Table 1.

UserName	First Name	Last Name	Password
agibson@g.com	Austin	Gibson	test111
ssharp@g.com	Sally	Sharp	test222
preese@g.com	Phill	Reese	test333
tclark@g.com	Tracy	Clark	test444
rinc@g.com	Rebbeca	Ince	test555

Table 1, Users accounts to be created.



(a) Login Activity (b) SignUp Activity (c) Inbox Activity

Figure 1, Wireframe for Login and SignUp Activities

Part B: Loading and Parsing Messages (Inbox Activity) (40 Points)

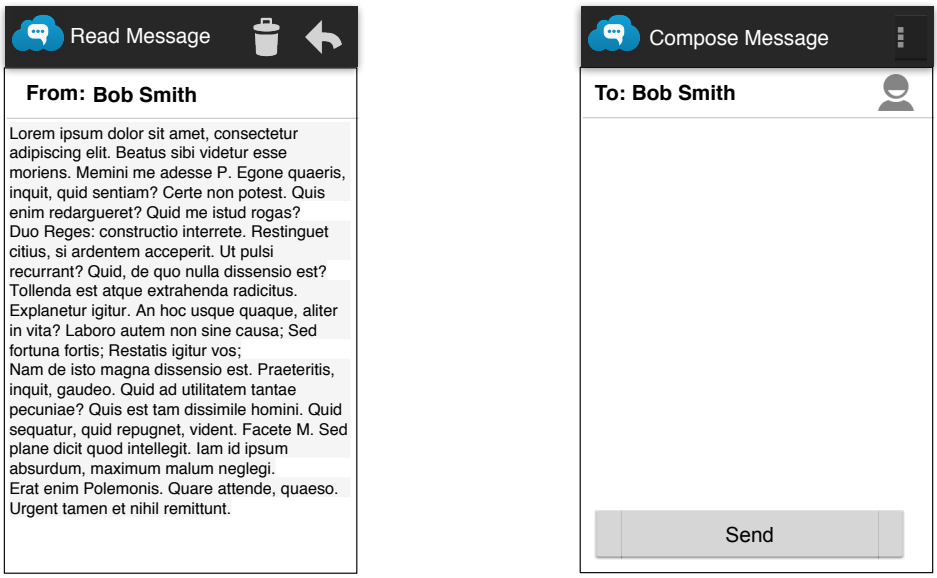
The Inbox activity should display the messages sent to the currently logged in user. The requirements are as follows:

1. In Firebase design the required Realtime database to manage the users' inbox. For each message store the message data, which includes the sender, receiver, message, and isRead (Boolean), created at and any other data required for managing the message.
2. Retrieve all the messages sent to the currently logged in user. The message row items should be displayed as shown in Figure 1(c). The items should be displayed in descending order by the message sent date (most recent at the top and older messages at the bottom).
3. The message row items should display the sender's name, time/date sent, and a short preview of the message, which is two lines with maximum 45 characters in length, so format the message to fit these constraints. A blue circle beside the message row item indicates the message has not been read, and a grey circle indicates the message has been read. Use the provided image files provided.
4. Clicking a message row item should start the ReadMsgActivity, which provides a more detailed view of the selected message item.
5. The ActionBar should show a compose icon: Clicking the compose icon should start the ComposeMsgActivity, which allows the current user to compose a new message.
6. The Inbox Activity should be refreshed when ever the activity is resumed to display the up to date message inbox.

Part C: Read Message Activity (20 Points)

The ReadMsgActivity shows details of the selected message item. The user can view the message, reply to sender, or delete the message. The requirements are as follows:

1. The activity's wireframe is shown in Figure 2(a). The UI should display the message sender's name, and the complete message text.
2. The message status should be updated to read status and should the updated status should be stored on Firebase.
3. Pressing the back key should return to the Inbox Activity.
4. The ActionBar should contain two icons: a trash, and a reply arrow.
   - 4.1. Clicking the trash icon should delete the message from the user's inbox and return to the Inbox Activity after the message is successfully deleted. The Inbox Activity should be refreshed to reflect the up to date messages stored in the user's inbox.
   - 4.2. Clicking the reply arrow should start the **ComposeMsgActivity.**



(a) Read Msg Activity

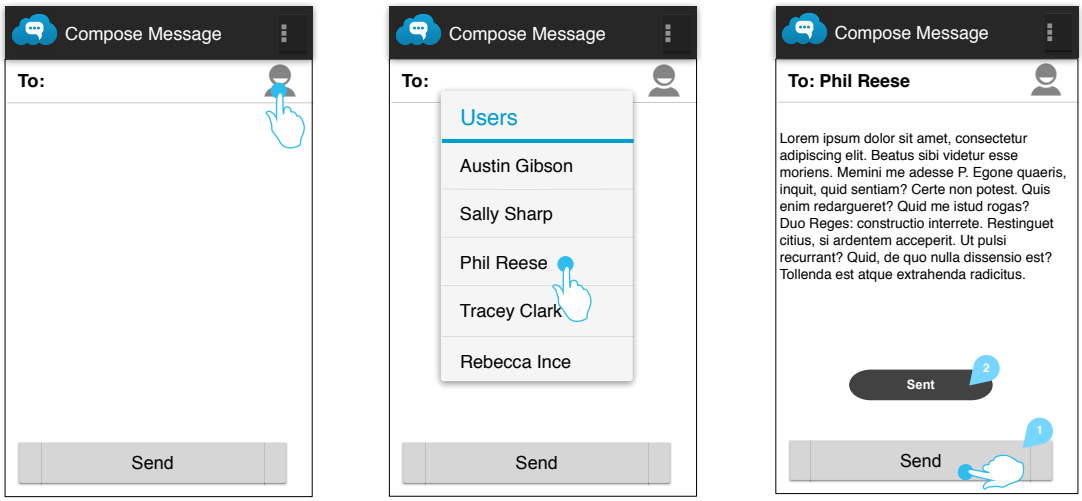
(b) Compose Msg Activity entered via clicking the reply button in the Read Msg Activity

Figure 2, Wireframe for Inbox Activity

Part D: Compose Message Activity (20 Points)

The ComposeMsgActivity allows the user to either compose a new message or reply to a message item selected from the Inbox. The requirements are as follows:

1. If the Compose Message Activity is started by Read Message Activity, then the “To:” TextView should be pre-initialized with the name of original message sender, to which the user would like to reply. The user should not be allowed to change the pre-initialized name. See Figure 2(b).
2. If the Compose Message Activity is started by the Inbox activity by selecting the composed icon, then the “To: ” row should be empty, see Figure 3(a).
   a) Clicking the “Contact” icon should display an alert dialog containing a list of all the full names of the users from Firebase. Selecting a name should initialize the “To: ” TextView field. See Figures 3(b)&(c).
3. After allowing the user to type in the desired message, pressing the “Send” button should upload the message to Firebase. After the message is successfully sent, a Toast message should be displayed indicating the message has been successfully sent, finish the ComposeMsgActivity and return to the Inbox Activity.



(a) Selecting the users icon (b) Users loaded from Firebase (c) Sending the message

Figure 3, Wireframe for Compose Msg Activity