CSE118 Summer 2023

Assignment 3

In this assignment you will single-Activity, multi-Fragment, network connected Android App to reinforce your understanding of the following concepts:

- Accessing RESTful APIs using `HttpsURLConnection`
- Separation of responsibilities between Fragments
- Observing changes in `LiveData`
- `ViewModel` event handling
- Use of the `FloatingActionButton`
- Responding to swipe actions on `RecylerView`
- Creating Dialogs
- Using Snackbars

This assignment is worth 10% of your final grade.

Setup

Download the starter code archive from Canvas and expand into an empty folder. I recommend creating a folder for the class and individual folders beneath that for each assignment.

Start Android Studio and import the project: **File → New → Import Project**

Select the folder where you expanded the starter code and click **Open**.

The `cse118.com` RESTful API

Endpoints are at: **`https://cse118.com/api/v0`**

Method	Path	Description
POST	**`/login`**	Use your UCSC email address, and student number as password
GET	**`/workspace`**	Returns all workspaces available to the logged in user
GET	**`/workspace/{id}/channel`**	Returns all channels in the workspace with the supplied id
GET	**`/channel/{id}/message`**	Returns all messages in the channel with the supplied id
POST	**`/channel/{id}/message`**	Creates new message in the channel with the supplied id
DELETE	**`/message/{id}`**	Deletes the message with the supplied id

You can explore the API here: https://cse118.com/api/v2/api-docs/

For more details, consult the handouts and recording from the Android V lecture, ask questions in Section and Office Hours, and study the 06 Ball Clubs example.

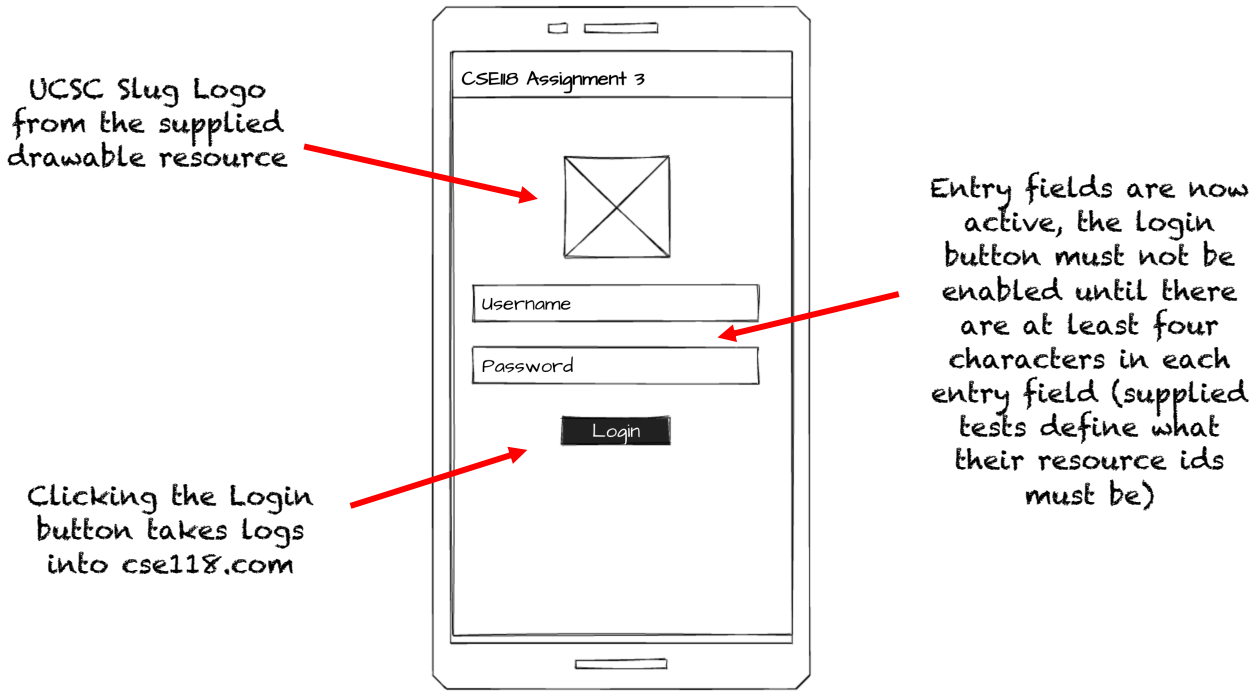
Do not use the `/ballclub` endpoints other than to run the Ball Clubs Example.

Requirements

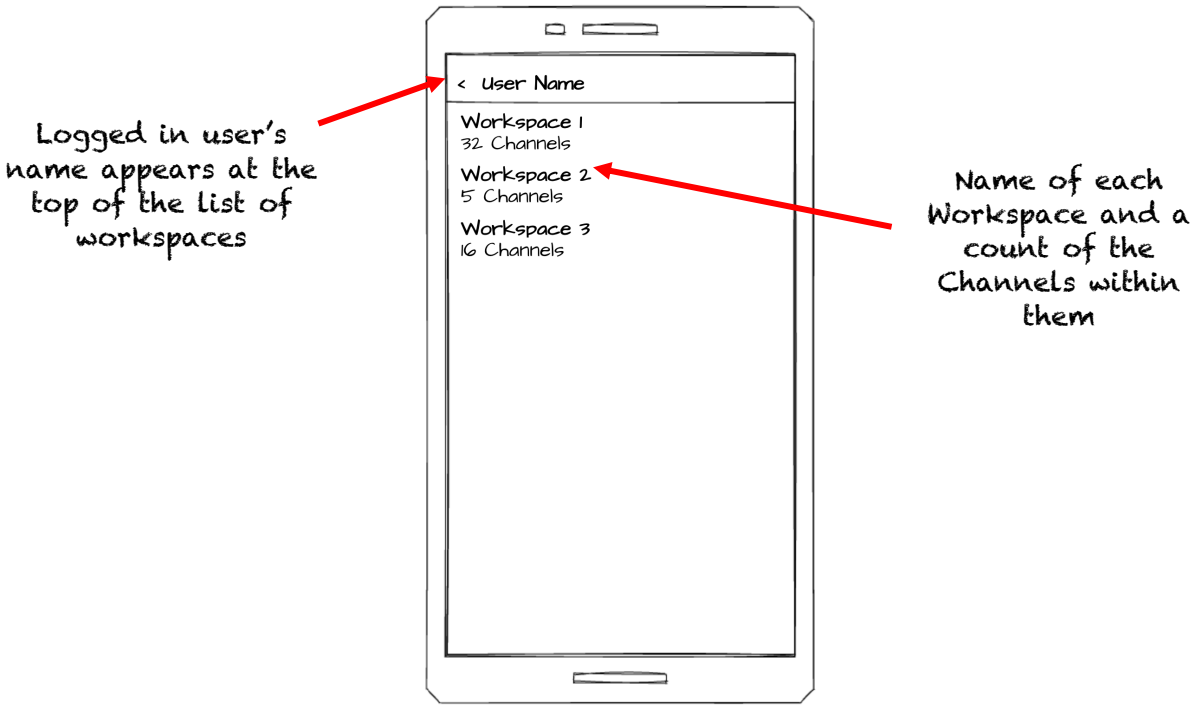
A client has given you a set of wireframes / mock-ups for a vaguely Slack-like messaging system they want you implement for Android. Your client has several requirements…

Basic: (instrumented tests are provided for this requirement)

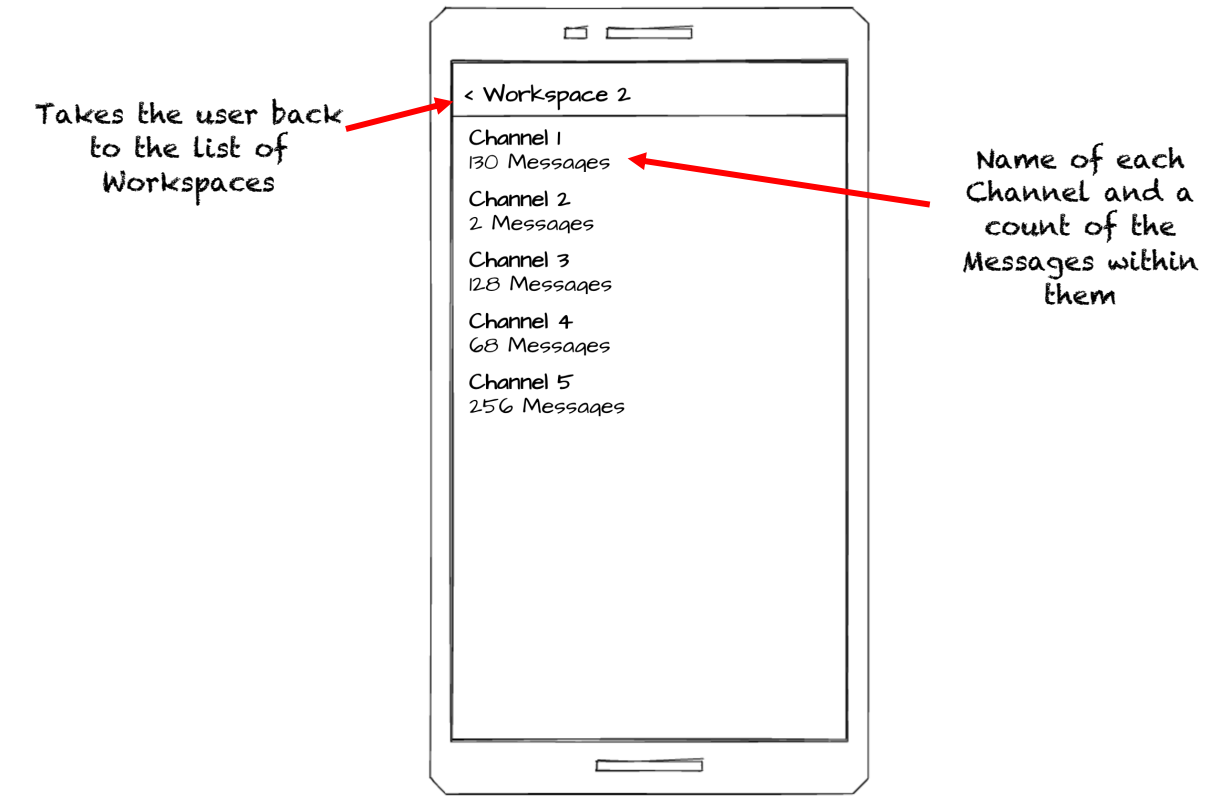
Show a functional login screen, before listing workspaces.



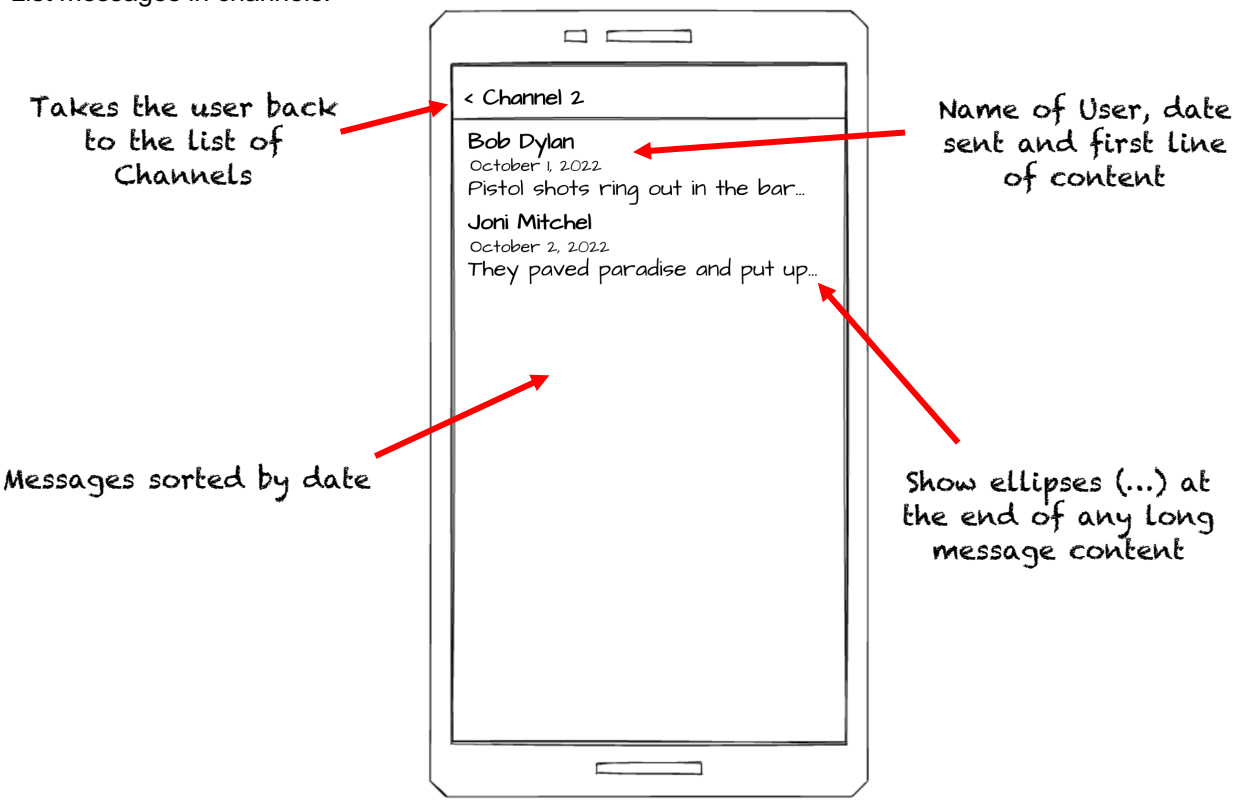
List Workspaces returned by cse118.com.



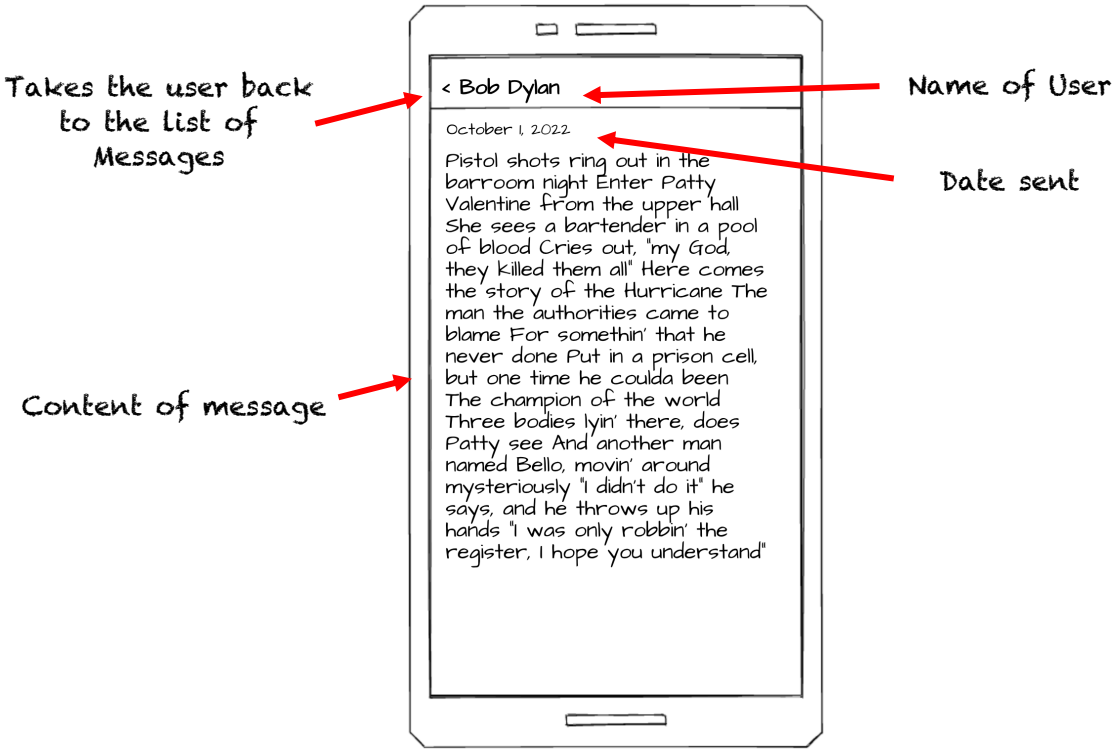
When a Workspace is selected, show a list of channels in that Workspace.



List messages in channels:

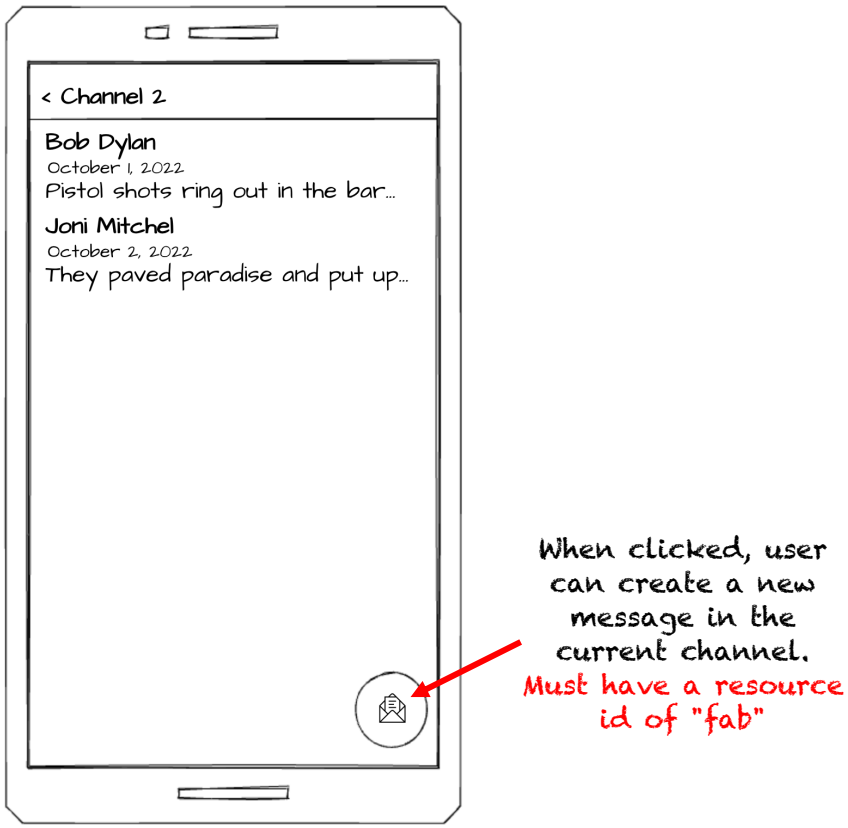


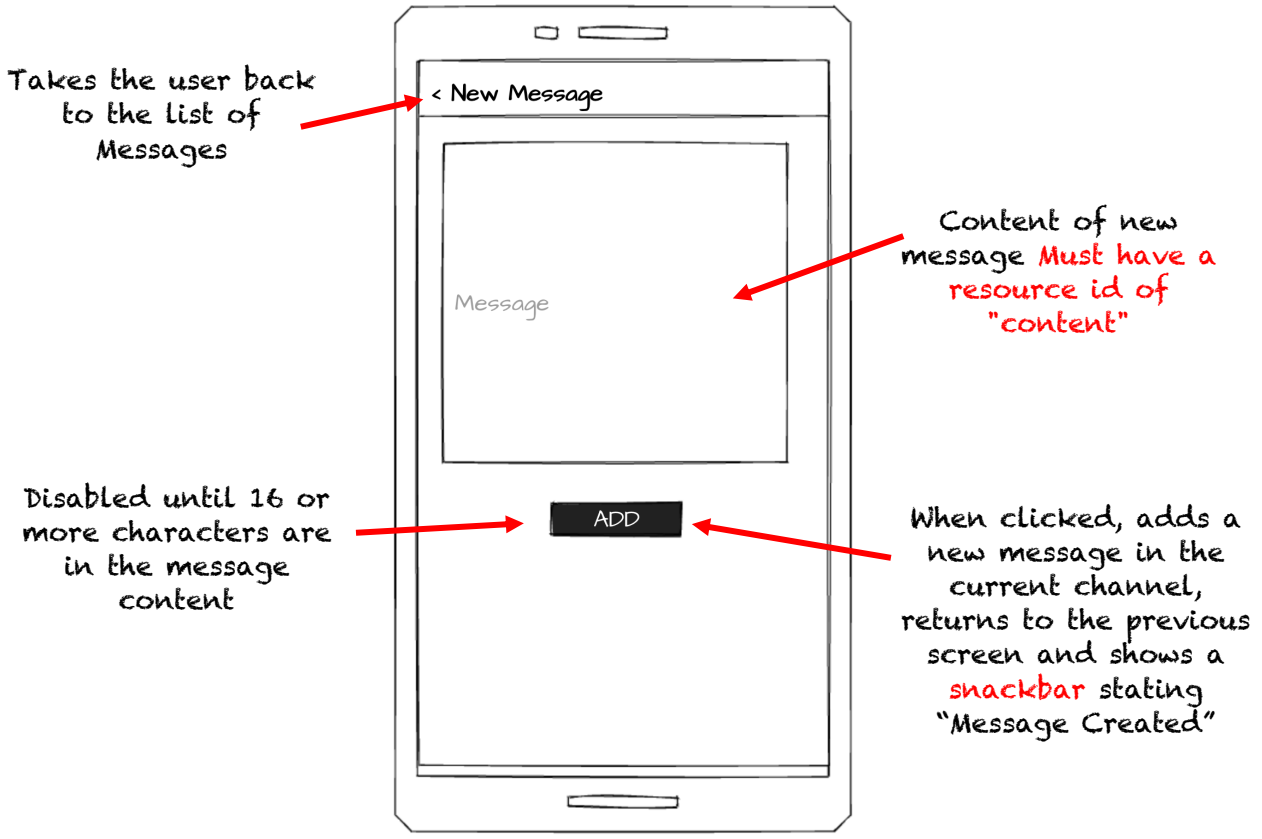
When a message is selected, show its contents:



Advanced:

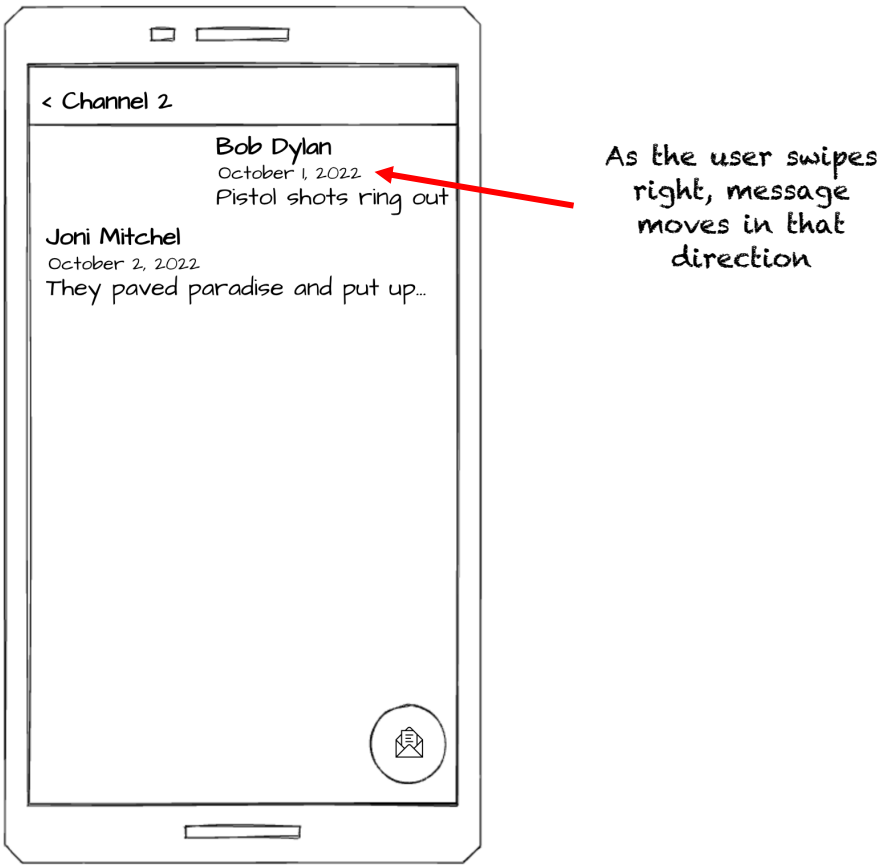
On the list of messages in a channel, show a `FloatingActionButton` that when clicked allows the user to create a new message in the current channel.



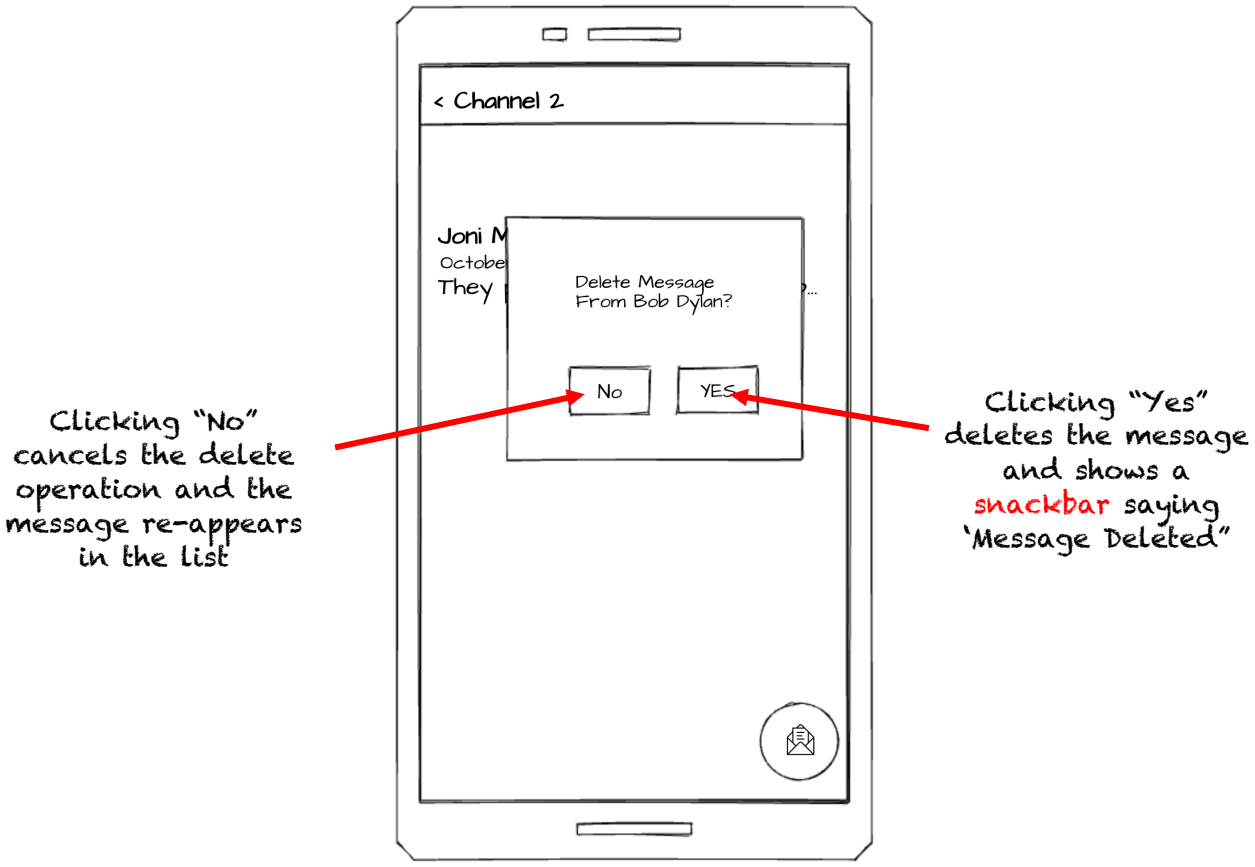


Stretch:

Delete a message when the user swipes right on it:



When the user completes the swipe, show a confirmation dialog:



Extreme:

Write Automated tests for Advanced and Stretch Requirements that pass when executed against the model Solution.

To achieve this the only View IDs you should rely on are `R.id.recyclerview`, `Ri.id.fab` and `R.id.content` other views should be found by the text they display.

What steps should you take to tackle this?

Whilst the order in which you put your code together is entirely up to you, a plausible initial development schedule might include the following steps:

1. Define Serializable Data Classes for:

    - Member
        - Name
        - Email
        - Role (user or admin)
        - Access Token (returned by cse118.com when you log in)
    - Workspace
        - ID
        - Name
        - Channels (how many channels are in the workspace)
    - Channel
        - ID
        - Name
        - Messages (how many messages are in the channel)
    - Message
        - ID
        - Poster (name of the person who posted it)
        - Date
        - Content

3. Create a ViewModel to store the logged in member and workspaces.

2. Create a Login Fragment and handle a login request by connecting to cse118.com.

4. On successful login, show a list of workspaces.

5. Create a Channel Fragment and handle the user selecting a workspace by showing the channels within it after making a call to cse118.com to find them

6. Create a Message Fragment and handle the user selecting a channel by showing the messages within it after making a call to cse118.com to find them.

7. Create a Message Detail Fragment and handle the user selecting a message by showing full content.

Once you have these steps complete, you've satisfied the Basic Requirement and can move on to Advanced.

How much code will you need to write?

A model solution that satisfies all requirements has approximately 1100 lines of Kotlin and 500 lines of XML.

Grading scheme

The following aspects will be assessed:

1. (100%) **Does it work?**

    a. Basic (60%)
    b. Advanced (15%)
    c. Stretch (15%)
    d. Extreme (10%)

2. (-100%) **Did you give credit where credit is due?**

    a. Your submission is found to contain code segments copied from on-line resources or created by code generation tools and you failed to give clear and unambiguous credit to the original author(s) in your source code You will also be subject to the university academic misconduct procedure as stated in the class academic integrity policy. (-100%).

    b. Your submission is determined to be a copy of a past or present student's submission. (-100%)

    c. Your submission is found to contain code segments copied from on-line resources that you did give a clear an unambiguous credit to in your source code, but the copied code constitutes too significant a percentage of your submission:

        - < 25% copied code No deduction
        - 25% to 50% copied code (-50%)
        - > 50% copied code (-100%)

What to submit

On the console (PowerShell on Windows), navigate to the folder you extracted the starter code into and run the appropriate command to create the submission archive:

Windows:

```
$ gradlew.bat clean
$ Compress-Archive -Path . -DestinationPath Assignment3.Submission.zip
```

Linux:

```
$ ./gradlew clean
$ zip -r Assignment3.Submission.zip *
```

Mac:

```
$ ./gradlew clean
$ zip -x "*.DS_Store" -r Assignment3.Submission.zip *
```

**** UPLOAD** `Assignment3.Submission.zip` **TO THE CANVAS ASSIGNMENT AND SUBMIT ****

