CSE118 Summer 2023

Assignment 2

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

- Fragment Lifecycles
- `RecyclerView` and `RecyclerView.Adapter`
- Navigation Graphs
- Shared `ViewModel`
- Data Classes, Serialization & JSON Parsing

This assignment is worth 10% of your final grade.

Installation

If you not already done so, download and install Android Studio: https://developer.android.com/studio

Setup

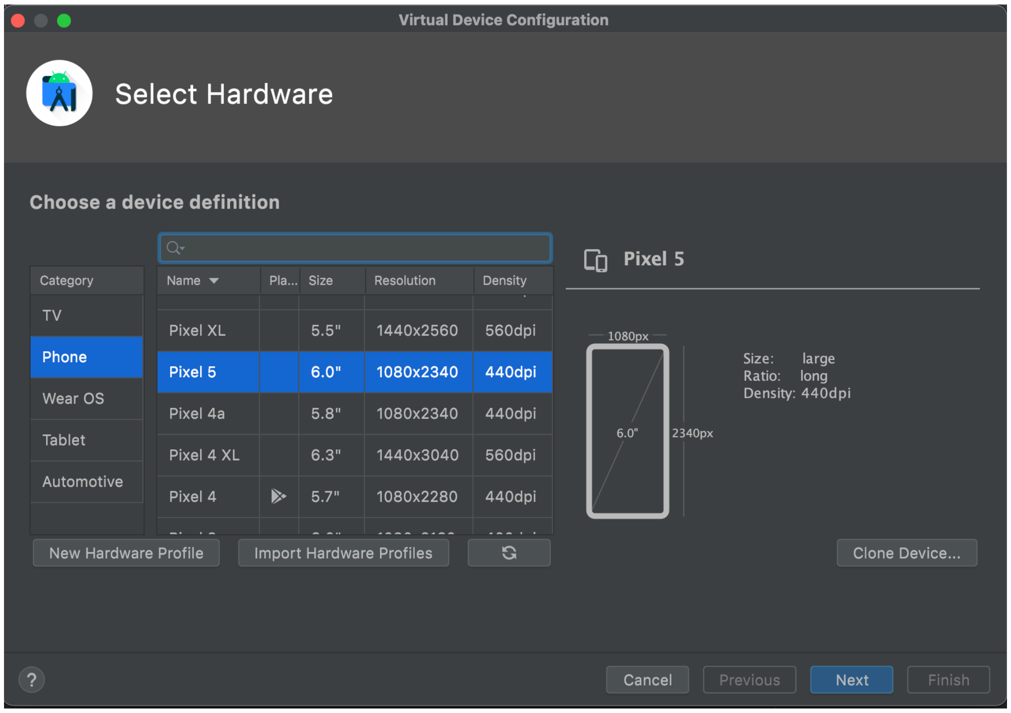
Download the starter code archive from Canvas and expand into an empty folder. I recommend creating a folder for the class and individual folders beneath that for each assignment.

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

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

Now create an emulator: **Tools → Device Manager**

Then click **Create Device** and select a Pixel 5 Phone:



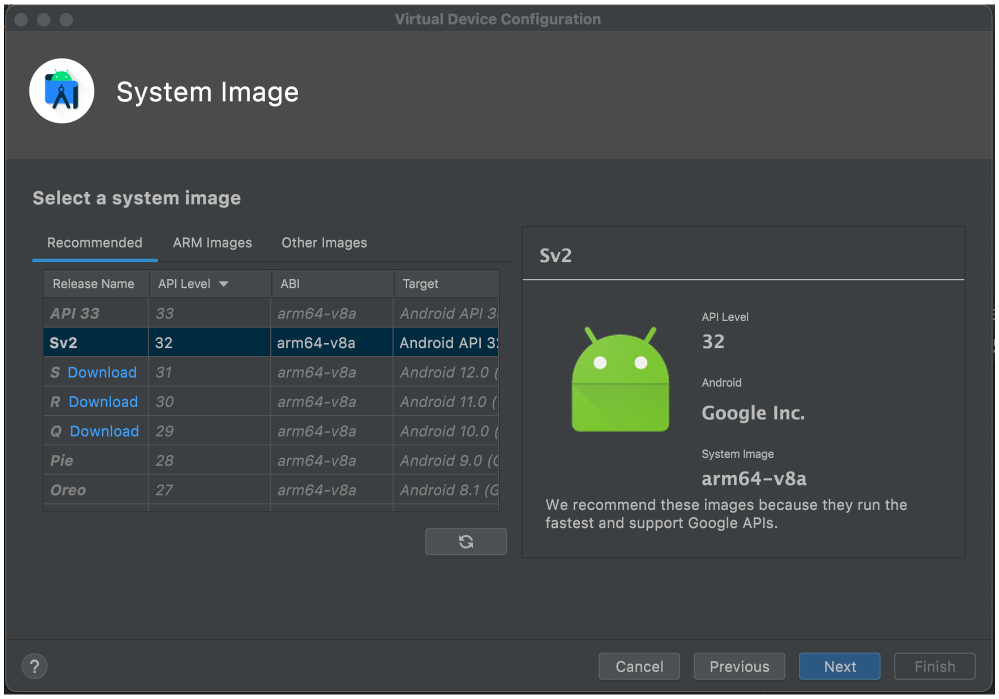
Name	Pla...	Size	Resolution	Density
Pixel XL		5.5"	1440x2560	560dpi
Pixel 5		6.0"	1080x2340	440dpi
Pixel 4a		5.8"	1080x2340	440dpi
Pixel 4 XL		6.3"	1440x3040	560dpi
Pixel 4	▶	5.7"	1080x2280	440dpi

Pixel 5

1080px
6.0"
2340px

Size: large
Ratio: long
Density: 440dpi

Click **Next** and download (if necessary) then select the Sv2 System Image:



Select a system image

Release Name	API Level	ABI	Target
API 33	33	arm64-v8a	Android API 3
Sv2	32	arm64-v8a	Android API 3
S Download	31	arm64-v8a	Android 12.0 (
R Download	30	arm64-v8a	Android 11.0 (
Q Download	29	arm64-v8a	Android 10.0 (
Pie	28	arm64-v8a	Android 9.0 ((
Oreo	27	arm64-v8a	Android 8.1 (G

Sv2

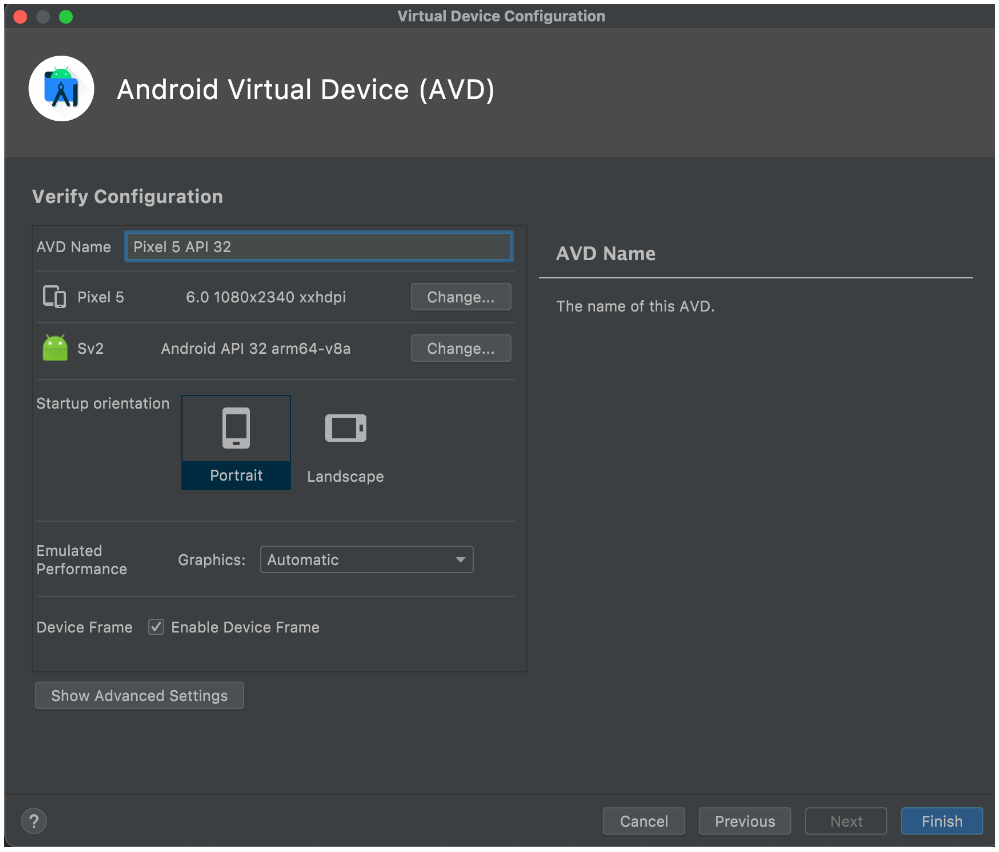
API Level
32

Android
Google Inc.

System Image
arm64-v8a

We recommend these images because they run the fastest and support Google APIs.

Give the emulator (AVD) a name of “Pixel 5 API 32” and click **Finish**:



Verify Configuration

AVD Name: Pixel 5 API 32

Pixel 5 6.0 1080x2340 xxhdpi

Sv2 Android API 32 arm64-v8a

Startup orientation: Portrait

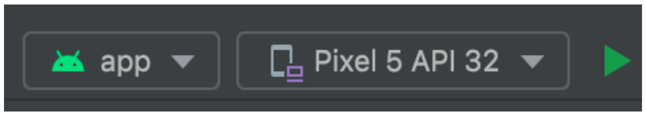
Emulated Performance Graphics: Automatic

Device Frame ✓ Enable Device Frame

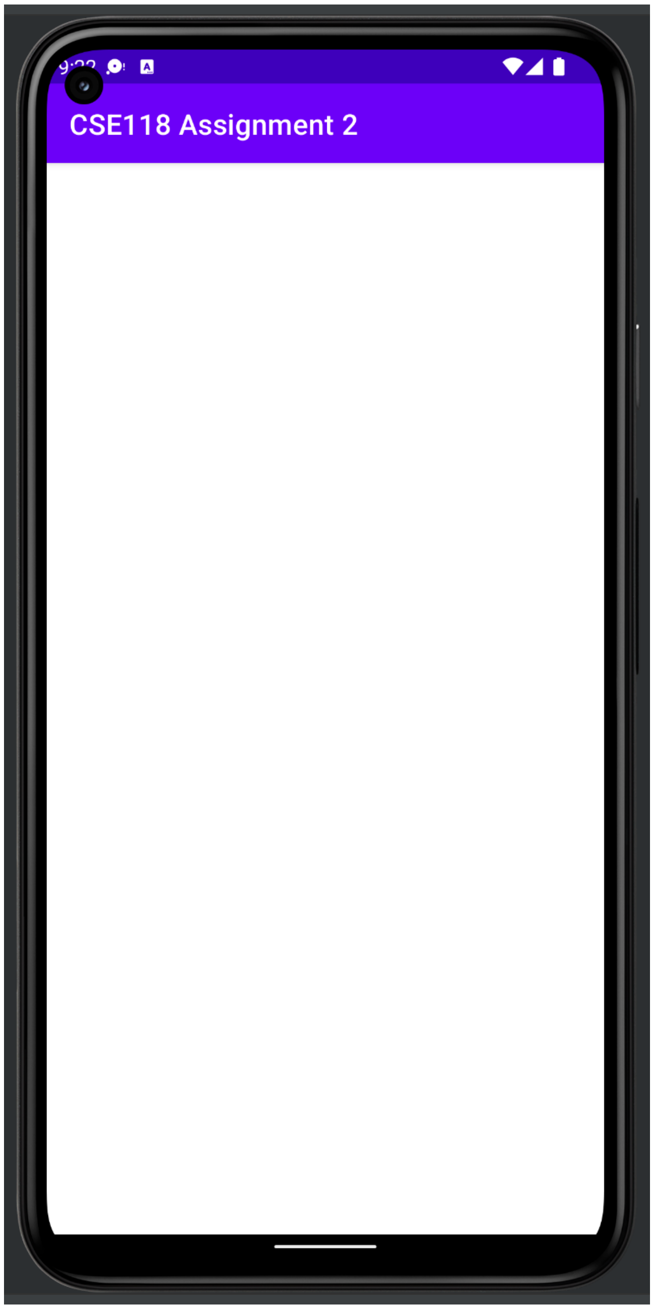
AVD Name

The name of this AVD.

Select the new emulator and click the green run button:



The following App should appear:



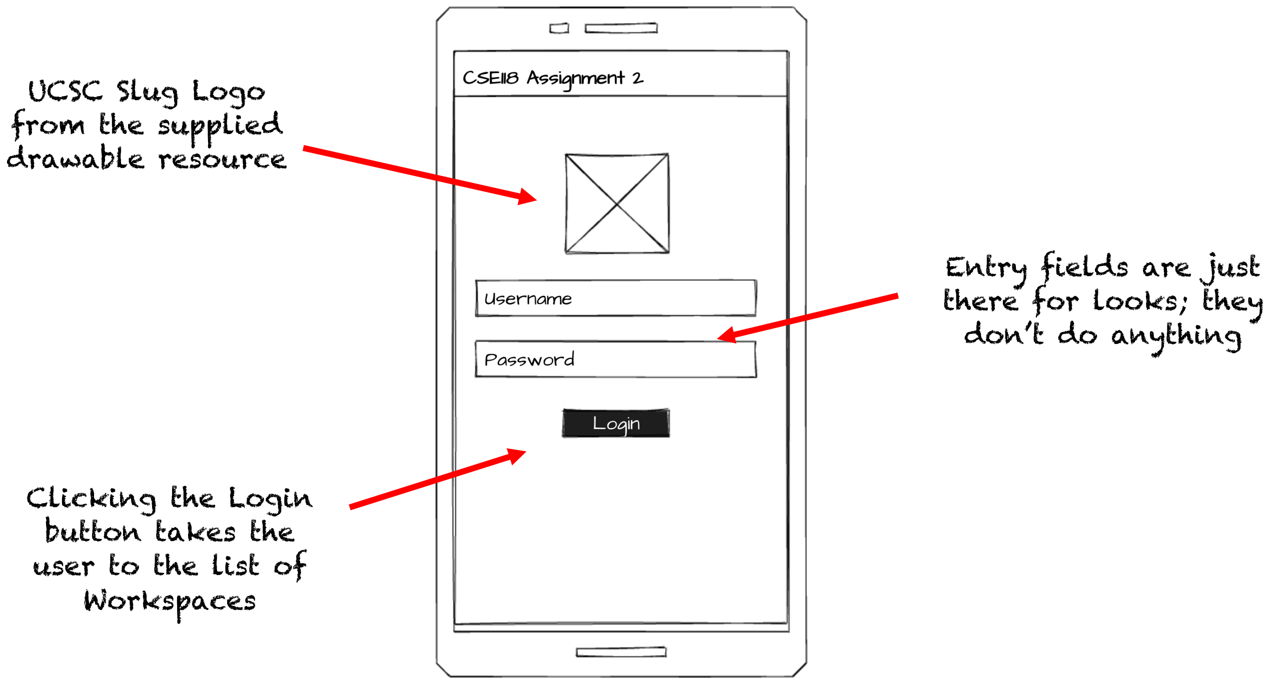
You are now ready to start developing your solution.

Requirements

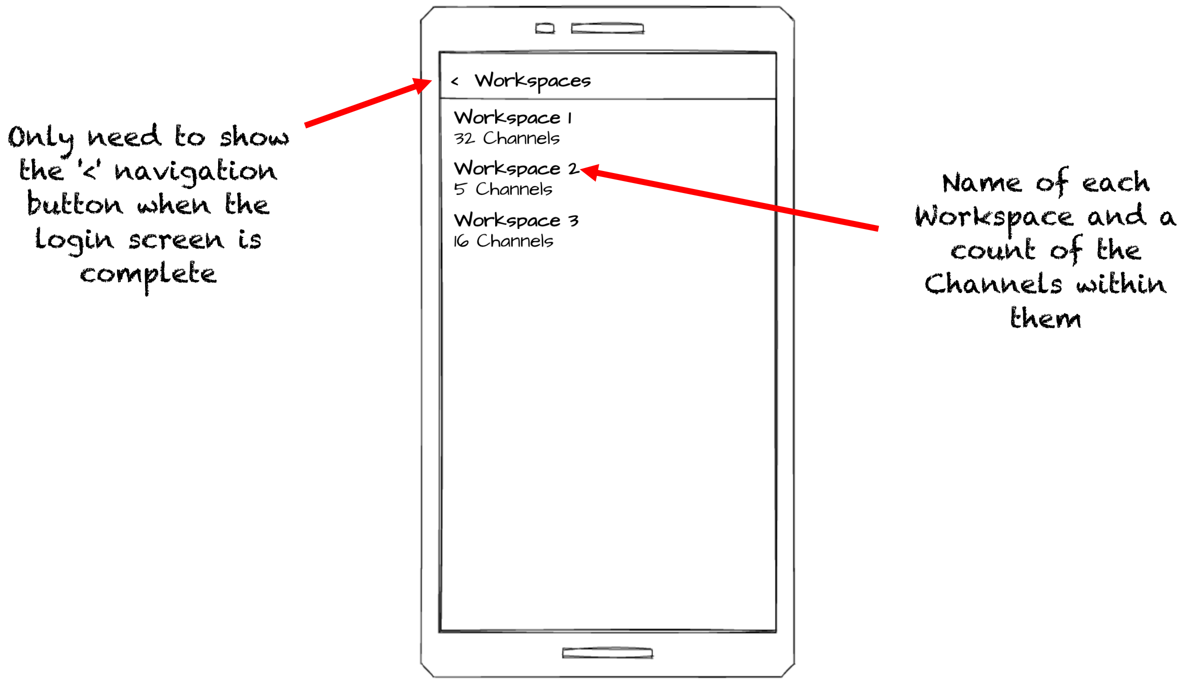
A client has given you a set of wireframes / mock-ups for a vaguely Slack-like messaging system they want you implement for Android. Your client has several requirements…

Basic:

Show a non-functional login screen, before listing workspaces.

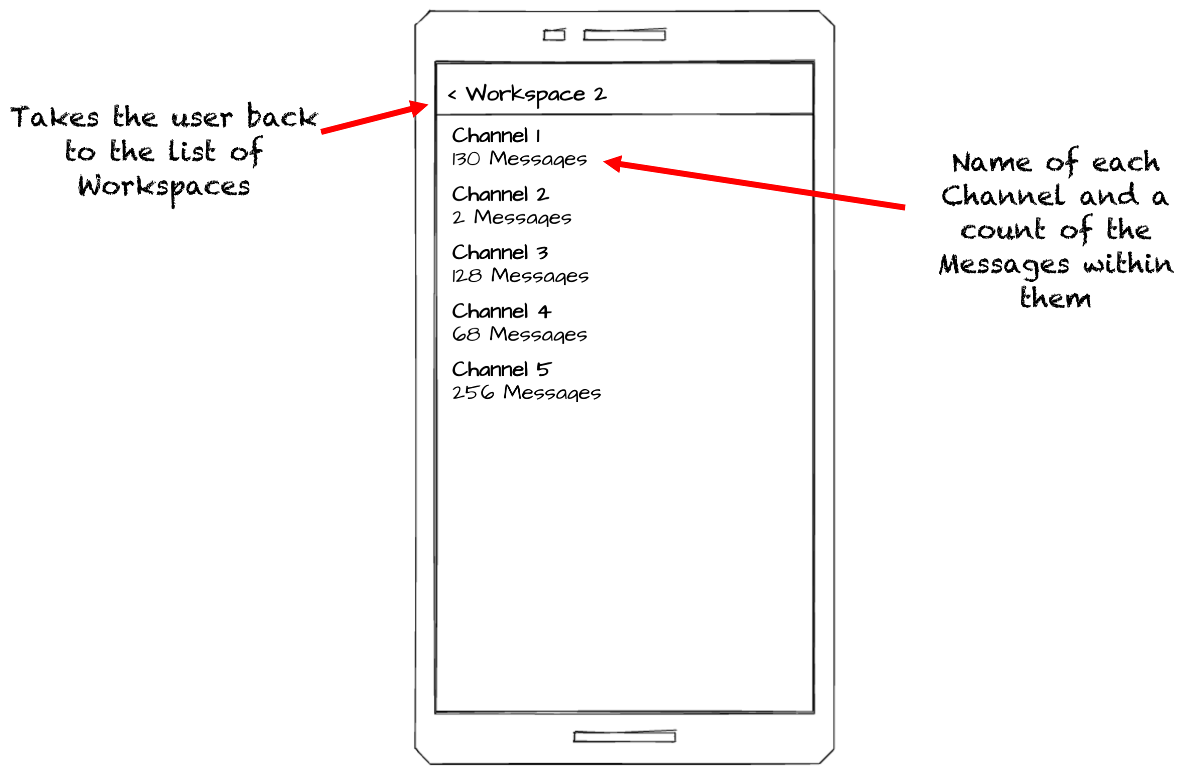

UCSC Slug Logo from the supplied drawable resource
CSE118 Assignment 2
Username
Password
Login
Entry fields are just there for looks; they don't do anything
Clicking the Login button takes the user to the list of Workspaces

List Workspaces found in the supplied JSON.

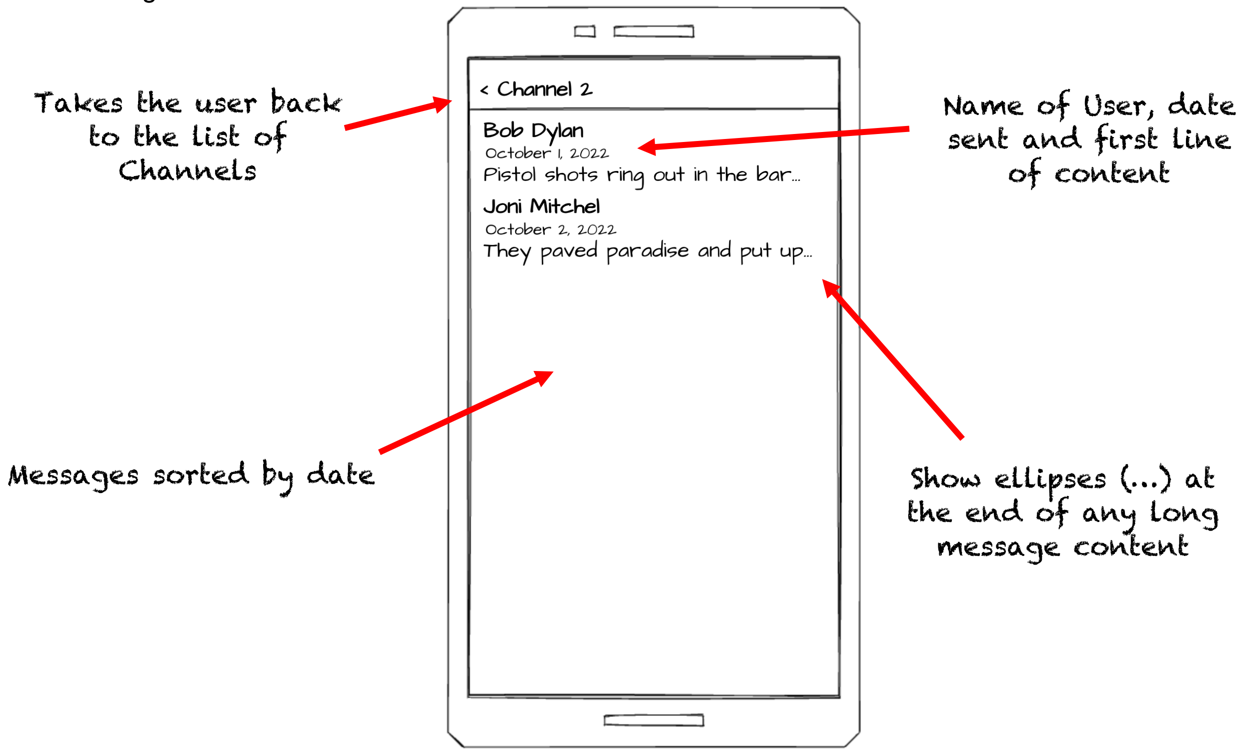

< Workspaces
Workspace 1
32 Channels
Workspace 2
5 Channels
Workspace 3
16 Channels
Only need to show the '<' navigation button when the login screen is complete
Name of each Workspace and a count of the Channels within them

Advanced:

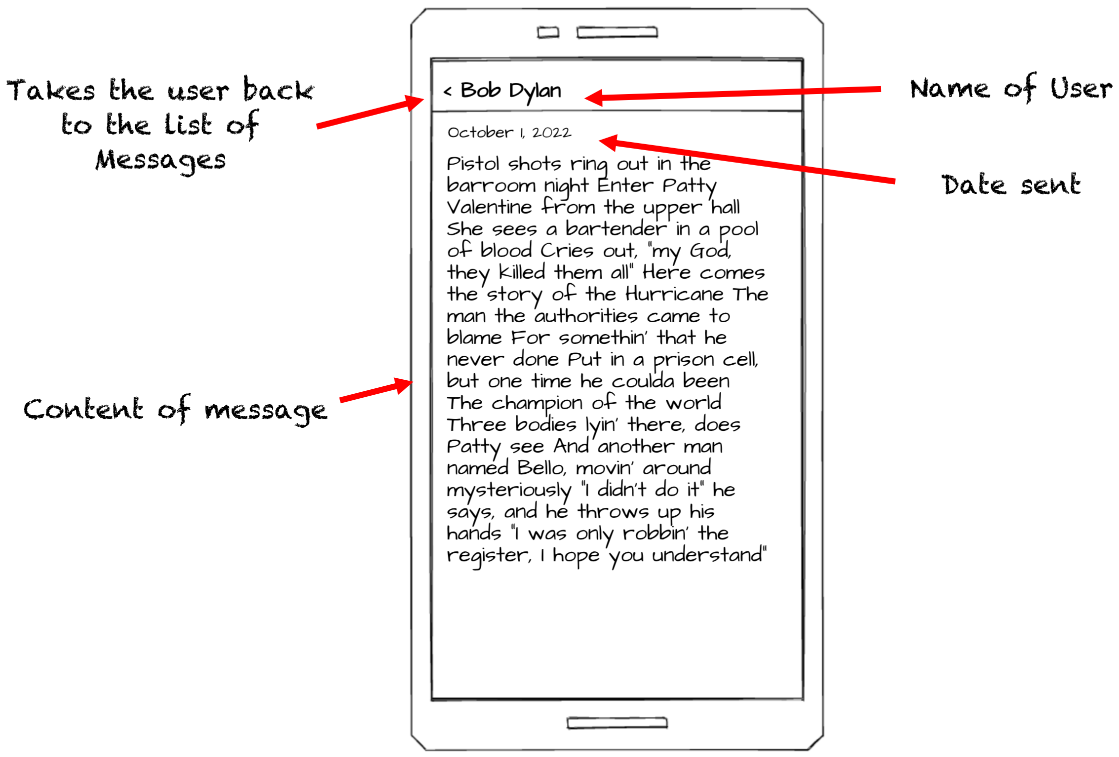
Workspace list is as shown in the Basic Requirement. When a Workspace is selected, show a list of channels in that Workspace.



List messages in channels:


Takes the user back to the list of Channels
< Channel 2
Bob Dylan
October 1, 2022
Pistol shots ring out in the bar...
Joni Mitchel
October 2, 2022
They paved paradise and put up...
Name of User, date sent and first line of content
Messages sorted by date
Show ellipses (...) at the end of any long message content

When a message is selected, show its contents:


Takes the user back to the list of Messages
< Bob Dylan
Name of User
October 1, 2022
Date sent
Content of message
Pistol shots ring out in the barroom night Enter Patty Valentine from the upper hall She sees a bartender in a pool of blood Cries out, "my God, they killed them all" Here comes the story of the Hurricane The man the authorities came to blame For somethin' that he never done Put in a prison cell, but one time he coulda been The champion of the world Three bodies lyin' there, does Patty see And another man named Bello, movin' around mysteriously "I didn't do it" he says, and he throws up his hands "I was only robbin' the register, I hope you understand"

What steps should I 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:

- Workspace
  - Name
  - Array of Channels
- Channel
  - Name
  - Array of Messages
- Message
  - User
  - Date
  - Content
- User
  - Name
  - Email

2. Read in the supplied JSON and make sure it can be parsed to an array of Kotlin Workspace objects. If you App keeps crashing, put a try/catch block around this code and log the exception.

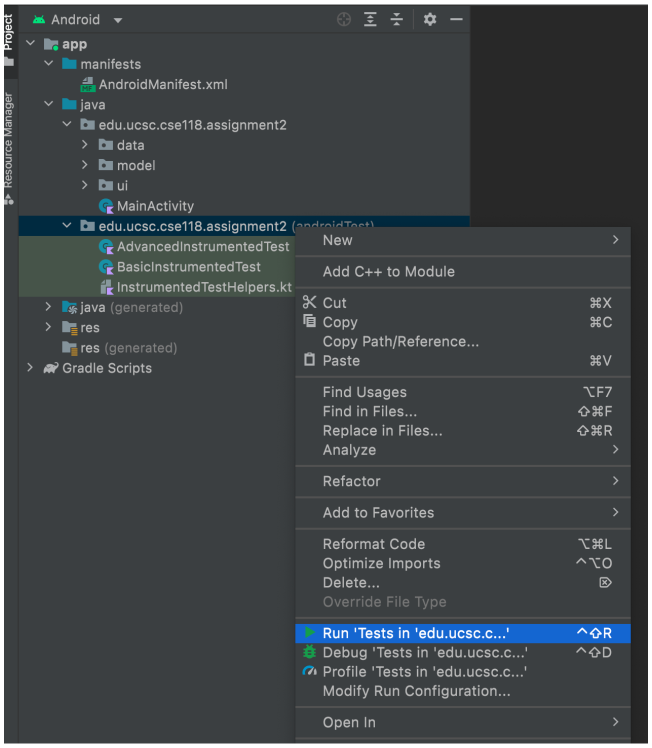
3. Create a ViewModel to store the workspaces.

4. Add a RecyclerView to the main activity and give it a RecycleView.Adapter backed by your ViewModel. You'll need a card layout or similar for the RecyclerView items.

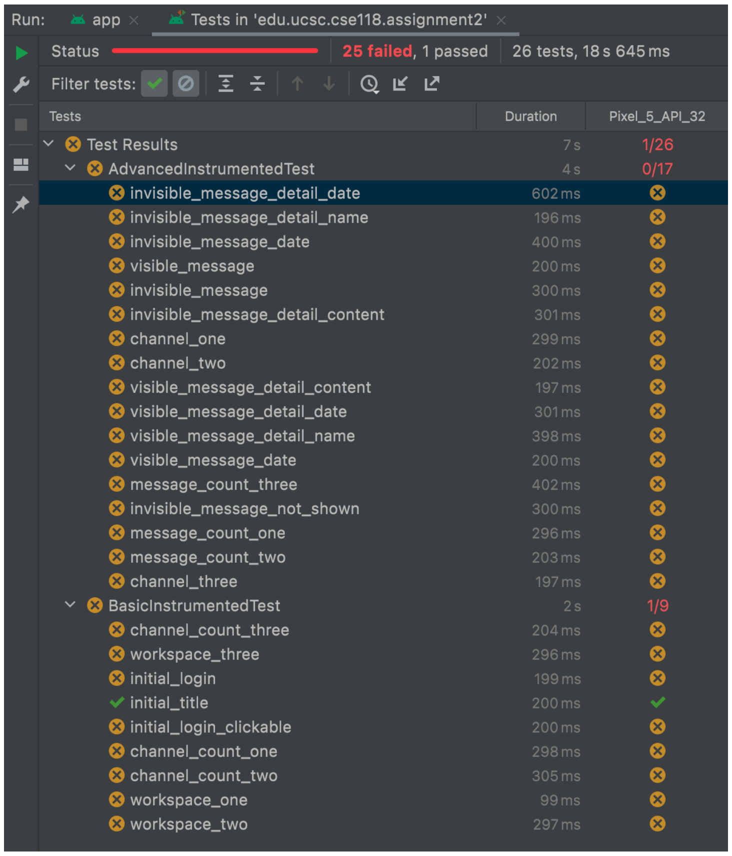
Once you have these steps complete, you've satisfied the Basic Requirement (see above) and can move on to creating Fragments for the login page and each of the lists (Workspace, Channel, Message) and linking them together in a Navigation Map.

Running the tests

Activate the context menu (i.e right click) on the `androidTest` code folder and select **Run**:



All tests will fail to compile until you have one Fragment with a `RecyclerView`, once you have that you will see a result like this:



```
Status    25 failed, 1 passed    26 tests, 18 s 645 ms

Tests                                      Duration   Pixel_5_API_32
Test Results                                   7 s    1/26
  AdvancedInstrumentedTest                     4 s    0/17
    invisible_message_detail_date            602 ms
    invisible_message_detail_name            196 ms
    invisible_message_date                   400 ms
    visible_message                          200 ms
    invisible_message                        300 ms
    invisible_message_detail_content         301 ms
    channel_one                              299 ms
    channel_two                              202 ms
    visible_message_detail_content           197 ms
    visible_message_detail_date              301 ms
    visible_message_detail_name              398 ms
    visible_message_date                     200 ms
    message_count_three                      402 ms
    invisible_message_not_shown              300 ms
    message_count_one                        296 ms
    message_count_two                        203 ms
    channel_three                            197 ms
  BasicInstrumentedTest                        2 s    1/9
    channel_count_three                      204 ms
    workspace_three                          296 ms
    initial_login                            199 ms
  ✓ initial_title                            200 ms    ✓
    initial_login_clickable                  200 ms
    channel_count_one                        298 ms
    channel_count_two                        305 ms
    workspace_one                             99 ms
    workspace_two                            297 ms
```

How much code will I need to write?

A model solution that satisfies all requirements has approximately 600 lines of Kotlin and 400 lines of XML.

Grading scheme

The following aspects will be assessed:

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

a. Basic	(50%)
b. Advanced	(50%)

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:

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

What to submit

Clean the project:

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:
```
      $ Compress-Archive -Path src -DestinationPath Assignment2.Submission.zip
```
Linux:
```
      $ zip -r Assignment2.Submission.zip src
```
Mac:
```
      $ zip -x "*.DS_Store" -r Assignment2.Submission.zip src
```

**** UPLOAD** `Assignment2.Submission.zip` **TO THE CANVAS ASSIGNMENT AND SUBMIT ****