CS 52—Summer 2023
Mobile Systems and Applications
Assignment Five—Application Architecture II

In this assignment, you will build on the previous assignment in two ways:

1. You will use `ViewModel` as a backing store for your activities.
2. You will replace the use of `ListView` with the more versatile and efficient `RecyclerView`. You can think of this as taking off the training wheels in your handling of lists of data in the UI.

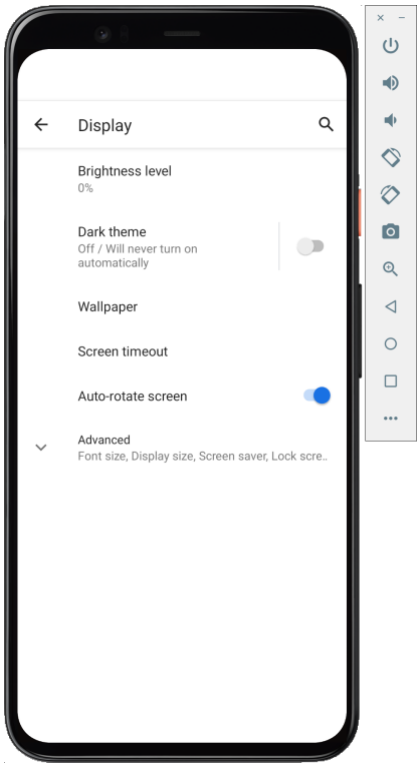
You should follow these guidelines for your implementation.

First, you are given the implementation of the `ChatViewModel` for the `ChatServer` activity. Complete this activity to only access the database indirectly, through the view model. Also complete the implementations of the view models for the other activities, and incorporate the use of these view models into the activities.

Second, in your main activity where you displayed the messages received in a `ListView`, use a `RecyclerView` instead. You are given an adapter `MessageAdapter` that uses the `Recycler.Adapter` class to connect the list of messages resulting from a query to the recycler view. The `MessageAdapter` class uses the layout resource `R.layout.message` as the layout resource for each row in your recycler view. In the adapter class, you are given a `ViewHolder` nested class that specifies the layout for each row in the list of messages, and you should complete the adapter to specify how the text views in this layout are initialized with data from a message (sender and message text).

The activities for viewing peer information also use recycler views (to show the list of peers, or the list of messages from a peer). You should use the generic class `TextAdapter`, which supports a list of objects that are displayed just by calling their `toString()` operations. The interesting wrinkle is how to support clicking on a peer in the list of peers, since `RecyclerView` does not support `OnItemClickListener`. See the `TextAdapter` class for an approach that does this, by registering an `OnClickListener` object on each row of the `RecyclerView` (and complete the code).

Since the point of using `ViewModel` is to cache the result of a database query, and re-use it if the device orientation changes and the activity is destroyed and recreated, you need to enable dynamic orientation changes in the device. You can do this in the display settings of the device:



In Assignment 7, you will see how to provide different user interfaces for portrait and landscape orientation of the device.

Submitting Your Assignment

Once you have your code working, please follow these instructions for submitting your assignment:

1. Create a zip archive file, named after you, containing a directory with your name. E.g. if your name is Humphrey Bogart, then name the directory Humphrey_Bogart.
2. In that directory you should provide the Android Studio projects for your apps.

In addition, record short mpeg videos of your apps working. Make sure that your name appears at the beginning of the videos. For example, put your name in the title of the app. *Do not provide private information such as your email or cwid in the video*. The videos should demonstrate running the app from Android Studio, where the code including the viewmodel definitions are clearly visible, exiting the app, then rerunning the app and demonstrating that the saved state from the previous run (messages for the chat app) are still present when the app is restarted. Make sure that you send messages from several peers to the chat server. Also, show by logging (using the logcat window) that a reorientation event causes the activity to be destroyed and recreated, but the view model is not destroyed until you leave the activity. You only need to show this for the main activity.

Your solution should be uploaded via the Canvas classroom. Your solution should consist of a zip archive with one folder, identified by your name. Within that folder, you should have two Android projects, for the apps you have built. You should also provide

videos demonstrating the working of your assignments, the schema generated as a Json file by Room, as well as a completed rubric.