CS 522—Spring 2021
Mobile Systems and Applications
Assignment Five—Application Architecture

In this assignment, you will redo Assignment Three, where you saved messages and chat peers in a database. In this assignment, you will use the Room ORM to interface with the SQLite database, and you will use `LiveData` to make your application reactive to updates in the database. This includes querying the database asynchronously, and updating the user interface using the observer pattern when the query is completed. We will also replace `ListView` for displaying a list of items with `RecyclerView`, which is more efficient in its memory usage. For now, we will **not** use a `ViewModel` to persist the UI state between rotation events.

You should follow these guidelines for your implementation.

First, annotate the `Message` and `Peer` entity classes that you defined in the previous assignment, to specify that these are to be stored in a Room-managed database. You should specify that the `senderId` field in a `Message` object is a foreign-key reference to the `Peer` record for the sender. We will continue to store sender names redundantly in a message object. This doesn't cause problems because peers are not able to change their names.

Second, define a subpackage called `databases`. In this class, define the database adapter class `ChatDatabase`, as well as the DAO classes `MessageDao` and `PeerDao`. These classes are largely complete, you just have to add annotations for Room. The `PeerDao` class shows how you can implement an "upsert" operation in Room. **You should not change the signatures of the DAO operations.** The bulk query operations always return `LiveData` observable results, but we will allow the insert operations to be performed on the main thread **for this assignment only**.

Third, in your main activity where you displayed the messages received in a `ListView`, use a `RecyclerView` instead, and define an adapter `MessageAdapter` to use 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 the adapter must specify how the text views in this layout are initialized with data from a message (sender and message text).

As in the previous assignment, provide another UI (accessible using the options menu from the action bar) for displaying a list of the peers. This UI just lists peers by name in a recycler view. Rather than passing the list of peers as a parcelable list from the main activity, you should in this assignment load the list of peers from the database (instantiating the `Peer` DAO in the activity for viewing chat peers).

If the user selects one of these peers, then provide in a third UI information about that peer (their currently known IP address, and the last time that a message was received from that user). Also display a list of messages just from this peer. You should pass the peer entity to this subactivity.

Both of these latter activities 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`. It is easy to do this in a bad way. See the `TextAdapter` class for an approach that avoids bad practices (and complete the code).

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 database definition with annotations is 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.

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.