CS 522—Spring 2019
Mobile Systems and Applications
Assignment Four—Content Providers

Set your minimum SDK version to Lollipop (Android 5.1, API 22) for your submission for this assignment. Your app theme should extend `android:Theme.Material.Light.DarkActionBar`We will be testing your app on a virtual Google Nexus 5X running API 22, so you should ensure that your app works on that platform, although you may develop on another device, e.g., your personal telephone.

In this assignment, you will extend the chat server app from the previous assignment. You are required to replace the use of a SQLite database from the previous assignment with a content provider. Define a single content provider with two tables, visible to the app, and distinguished by their URIs that have the same authority but different content paths. One table stores the messages that have been received, while the second table stores information about the peers from whom we have received messages. You should define a contract and a provider class for this content provider, as well as entity classes for messages and peers, using the same package structure as in the previous assignment.

You should follow these guidelines for your solution:

1. Define two contract classes, `MessageContract and PeerContract`, for the content provider, `ChatProvider`. Place the former two classes in the `contracts` subpackage of your app, and the latter in the `providers` subpackage of your app. This is a practice that you will be expected to follow for all assignments for the remainder of the course. The contracts class should define content URIs and content paths, content types, the column names for the cursor and content values, and operations for retrieving columns from a cursor and inserting them into a content values table. The code you are provided with defines a base class `BaseContract` that defines some operations common to all contract classes.
2. Define an entity class, `Message`, for messages entities stored in the database, and an entity class, `Peer`, for chat peer entities stored in the database. You should have done most of this for the previous assignment. This should define entity fields, an implementation of the `Parcelable` interface, a constructor for initializing an entity from a cursor, and an operation `writeToProvider` for initializing a `ContentValues` object (for insertion into a provider) with the fields of the entity.
3. Use cursor loaders and the loader manager to query the provider. `ChatServer` activity uses the loader manager to query the content provider for a list of all messages. `ViewPeersActivity` uses the loader manager to query the content provider for a list of all chat peers. When it launches `ViewPeerActivity` to view the details for a chat peer, it should pass a peer entity object (as a `Parcelable` extra) to the subactivity; the latter will still need to use a loader manager to query the content provider for a list of all messages, filtered for a particular sender. *Define all callbacks for the loader manager as methods in your activities, not in separate callback objects.*
4. `ChatServer` is responsible for inserting a record for a message, and upserting a record for the sender. For this project, it is okay if those operations are performed on the main thread, using the content resolver.

Note that your apps should never use the `startManagingCursor` or `managedQuery` operations, or the constructor for `SimpleCursorAdapter` that takes a cursor as its argument. However it is all right in general to use the `SimpleCursorAdapter` class, using the second constructor that provides a flag that indicates that the query should not be managed by the activity. Just do not use the first, deprecated constructor that causes queries to be managed on the UI thread.

Your solution should consist of two apps, `Chat-Client-Oneway` and `Chat-Server-Oneway`. The client app is unchanged from previous assignments. The chat server should follow the guidelines described above above: Use cursor loaders and the loader manager for querying all messages received so far, and for querying all peers from whom we have received messages. It is okay to "upsert" peer and message information into the database, when a message is retrieved, on the main thread. We will fix this in the next assignment, when we define asynchronous content resolvers.

Submitting Your Assignment

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

1. You should submit two Android Studio projects: chat client and chat server.
2. 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.
3. In that directory you should provide the Android Studio projects for your apps.
4. You should also provide APK files for your compiled projects.
5. Also include in the directory a completed rubric for your submission.

In addition, record short mpeg or Quicktime videos of demonstrations of your assignment working. Make sure that your name appears at the beginning of each video. 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*. Be careful of any "free" apps that you download to do the recording, there are known cases of such apps containing Trojan horses, including key loggers.

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 Studio projects, for the apps you have built. You should also provide videos demonstrating the working of your assignment. **Make sure that your video shows the server app being launched from the Android Studio project based on content providers and loaders.**