Project 2: MVVM Architecture and the Repository Pattern

This is a group project with specific requirements. There will be no project presentation for this one. You will change roles (Design, Team, Test) for this from Project 1, and notify me via Canvas of your new roles. Submit the new class and test diagrams, and also an updated design sketch, together with your code. Only the new team leads must submit code.

Your task for Project 2 is to refactor your code from Project 1 to utilize the Model-View-ViewModel (MVVM) architecture in conjunction with the Repository Design Pattern. This will require the use of ViewModel and LiveData.

1. You will re-organize your code to use the **Repository design pattern**. In other words, your user data and any internet data must be fed into a **repository class**. The repository class must in turn be managed by a **view model class** extending ViewModel or AndroidViewModel, which creates, writes to, and reads from the repository. **You are welcome to experiment with a design that includes multiple ViewModels and repositories to enforce separation of concerns.**

2. You have, in general, 2 sources of data for your projects. The first is user data, and the second is data from the internet. Modify your repository (or repositories) to do the job of fetching data from the internet using whatever utility classes you have; the repository (or whatever class feeds the data to the repository) must use AsyncTask to achieve this. Remember, the ViewModel holds all repository instances, giving automatic protection against configuration changes, but onSaveInstanceState will still be required to handle the application stopping.

3. You will reorganize your code so that all Activities and Fragments talk to the ViewModel, and not to each other. This means no more data-passing from Fragments to Activities and Activities to Fragments. All data lives in the ViewModel, and all data is fetched from the ViewModel. See here for a simple example: https://developer.android.com/topic/libraries/architecture/viewmodel#sharing

4. You will eliminate all loaders (a la AsyncTaskLoader) from your code, since you are now strategically using ViewModel and LiveData.

5. Add a Room database to backup user data from the repository. Decide if Room is needed for the internet access part of your app; if you are fetching some form of continually refreshed data, implement a Room database for this also.

6. Do any user-interface and architecture polishing needed to accommodate these new changes.