**Task :- User Details App   
( With Pagination and Room Database)**

**Explanation →**  
1. **Presentation Layer:**

* Used Traditional UI (Recycler View).
* Screen 1 is Splash screen displays for 3 seconds.
* Screen 2 is Login screen added for the app, two fields (mobile number, password) are with validations.
* Screen 3 is Dashboard Screen, where the main list (required list) is displayed in recycler view which has a simple layout in activity only. (To keep code short and simple as the problem statement is very short, I didn’t use fragments or tabs or viewpager.

**2.** **App Responsiveness:**

* To make sure that UI is responsive on different screen sizes and orientations, I used libraries for DP and SP:

com.intuit.sdp:sdp-android

com.intuit.ssp:ssp-android

**3. App Structure:**

Packages used are as follows: -

* **adapter** contains adapter class for recycler view.
* **api** package contains all data related to Api’s.
* **net** package contains two files, interface for Api and retrofit client instance class, to build http client and logging events.
* **repository** package contains repositories for Api’s and room databases.
* **server response** has the model classes for the responses received from Server. (It is nothing but a simple class having variables and getter setter methods for it as like in JAVA.)
* **db** package contains classes related to storing app data.
* AppDataBase is an abstract class to create instance of Room DB
* Session Manager is a class which handles Shared Preference variables and functions.
* UserDAO defines all queries.
* **model** Package contains a model class for user details, variables available in List.
* **presentation** is nothing but views which contain activities and fragments.
* Login Details: any 10-digit mobile number, password any except null(empty)  
  (to store a session of login Shared pref. Used, which is having in Session manager)
* Dashboard Screen: display data from Api. (Here I inserted data in the table Named Users and updating data (To test updating used dummy data).  But not retrieved it from DB (code is commented for data retrieval) as it is not mentioned in the problem statement, directly displaying Api response. For pagination rather than using the library I used nested scroll views.
* **utility** is nothing but as package that contains validation functions, messages to display, textview, edit text classes to set same font to all over the app, URLs etc.
* **view model** is having to view model classes, one for Api and other to retrieve data from room Database.

**4. UI Design:**

* Glide library used for image loading.
* Purple theme is used all over the app.
* Fonts are stored in the Asset folder. Used font: Inter.
* Data Binding is Used.

**5. API**:

* Library Used Retrofit

**6. Test Case:**

* I used Mockito, Junit Runner libraries for Unit tests. 1 test case written for inserting items to DB. I had not worked on Unit tests. But I tried to work on it and write one.