CS6018-Project2 Design Guide

Qi Liu

October 2019

1 Introduction

In Project 1, we created a user class and a local file to store user information and keep track of current user's activity. And there is no data storage performed in terms of fetching weather data from internet. In order to refactor our code to utilize MVVM architecture in conjunction with the Repository Design Pattern, multiple view models and repositories should be introduced. For user resource, a user view model with user related information is required in order to enable all activities and fragments to get access to user data without data transfer. And for user repository, a user database, user table and user DAO are also need to be created. For internet data resource, we create a weather view model which is used to provide weather data fetched from internet. And data repository is also applied to store the weather data in case that fetching data from network fails.

Plus, in our project, we transfer an integer from ChangedDesignedChoice Fragment back to Main Activity to decide which fragment we need to render when user clicks buttons. In order not to modify the basic structure we have, a Choice View Model is probably needed to replace integer transfer between fragments and Main Activity. But in the future the structure needs to be improved to render different activities by only clicking buttons without the help of an integer indicator.

2 User

2.1 User View Model

User View Model has MutableLiveData of user object and a user repository as member variables.

2.2 Room

A User table has id as the primary key, as well as other keys corresponding to user class's members. A User DAO has three operations: insert, get and

deleteAll. And eventually a User database is required with User DAO object as its member variable.

3 Weather Data

We could implement weather View Model and Room similarly to the example in lecture.

4 Workflow

In Main Activity, we choose which activity class to go by checking if user information exists in local file. We could modify it by removing local file storage and adding a User View Model object as its member variable. In Observer class, by checking if user object exists , we decide to create a new user or not. If user does not exist, we jump to NewUser Activity class, or display user information in HomeActivity if it's not.

In NewUser Activity, currently we pass an integer to check which fragment we should render and also pass a user object to each fragment. Since we are using User View Model, we can simply just call a method to direct user to each fragment one by one without checking which step we are currently at. In each fragment, we fetch user data from View model, which eventually routes the calling to Room to get user data, and we fill in properties of this user object in each step and insert this user back to database. Each fragment also needs to have a User Vie Model in order to get User data.

In Home Activity, currently we go to ChangeDesigned Fragment, get the integer and route back to Main Activity to render the fragment indicated by the integer transferred from Home Activity. Since we do not need to transfer integer anymore with the help of Choice View Model, Home Activity is not necessary. We add Change View Model in Main activity as another view model. If user exists, we fetch choice number from Choice View Model to tell which fragment needs to be rendered.