PBD 2020

Android Mini App 3

Summary

In this assignment you will create a cooking recipe management Android app. The app allows its user to: (1) browse online recipes filtered by the main ingredient; (2) view details of each recipe; (3) mark recipes as “favorites”; (4) show the list and the details of the favorite recipes even when offline.

NOTE: Please read the instructions carefully before you begin programming!

Deadline: 23:59, May 24th, 2020.

Features and Architecture

To kickstart your project, please download the following repository
https://github.com/vpejovic/MiniApp3

Your app must have the following components and classes
[NOTE: packages are relative to si.uni_lj.fri.pbd.miniapp3, ie. “ui” means “si.uni_lj.fri.pbd.miniapp3.ui”]:

1. **SplashActivity** (of type AppCompatActivity; package ui), with layout file **activity_splash.xml**. The splash screen must cover the whole screen and should stay on for exactly 2 seconds. The screen must automatically advance to MainActivity.
2. **MainActivity** (of type AppCompatActivity; package ui), with layout file **activity_main.xml**. The layout must be of type CoordinatorLayout and must contain an AppBarLayout with a Toolbar for app title and a TabLayout for tabs. In addition the layout must contain a ViewPager2. In MainActivity the ViewPager2 should be connected with an Adapter (see SectionsPagerAdapter below).
    a. **SectionsPagerAdapter** (of type FragmentStateAdapter; package adapter) ensures that the right fragment is shown when tabs are switched. The correct title of the tab must also be set (in MainActivity).
3. **SearchFragment** (of type Fragment; package ui.search), with layout **fragment_search.xml** of type ConstraintLayout. This fragment allows a user to select the main ingredient for

which we search the recipes for. Then, it downloads the recipes with the given main ingredient from themealdb.com. The recipes are shown in a RecyclerView grid with pictures and recipe titles (see example below).

fragment_search.xml must contain the MaterialProgressBar that is shown while the app downloads the recipes (the external dependency me.zhanghai.android is already included in the Gradle file on Github). The layout must also contain a Spinner element that must be populated with the list of possible main ingredients via an Adapter (see SpinnerAdapter below). Below the Spinner, the Fragment must show a RecyclerView with the recipes. Put RecyclerView in a SwipeRefreshLayout (this will enable data refreshing when a user swipes the view down). You should also add the following:

   a. **SpinnerAdapter** (of type BaseAdapter; package adapter) ensures that a list of Strings (ingredients) is connected with a **spinner_item.xml** view (already provided, but feel free to modify it).
   b. **RecyclerViewAdapter** (of type RecycleView.Adapter; package adapter) ensures that each item in a list of RecipeSummary objects (see below) is connected with a **layout_grid_item.xml** (already provided, but feel free to modify it). The adapter should also keep track of whether it was instantiated from the SearchFragment or from the FavouritesFragment. Furthermore, the adapter must setOnClickListener for images - when a user clicks on an image of a recipe, it should be forwarded via an Intent to DetailsActivity (see below). The Intent should have the following Extra fields: (1) a recipe ID; (2) info on whether the RecyclerViewAdapter was instantiated from the SearchFragment or from the FavouritesFragment.

4. **FavoritesFragment** (of type Fragment; package ui.favorites), with layout **fragment_favorites.xml** of type ConstraintLayout. The layout contains a RecyclerView where recipes tagged as favorites are shown. These recipes must be pulled from a local Room database through a ViewModel and must be shown even if there is no connectivity.
5. **DetailsActivity** (of type AppCompatActivity; package ui) with **activity_details.xml** of type ScrollView. The layout must show the recipe picture, ingredients and their measures, and the preparation instructions for a selected recipe. In addition, the screen must show a button (star icon or similar is also acceptable) for adding the recipe to the favorites, if it is not in the favorites already. The same button must be used to remove the recipe from favorites, if it is already in the favorites.

   If DetailsActivity is started from SearchFragment it should fetch the recipe details from the remote server (lookup.php?i=RECIPE_ID) with RECIPE_ID forwarded from the Fragment. If it is started from FavoritesFragment it must fetch the recipe details from the local database and the ViewModel.

Remote connection in the app. The app uses a Rest API to communicate with themealdb.com. Use Retrofit2 external library to create a Rest client so that you can connect to the server. The classes you need for this are already (partly) provided:

1. **RestAPI** (package rest) defines API endpoints. The API call where you get a list of all the ingredients is already provided. You need to define two more:
    a. A call that gets all recipes that contain a certain main ingredient. It should point to “filter.php” and it takes a parameter “i”, a string representing the main ingredient (e.g. check https://www.themealdb.com/api/json/v1/1/filter.php?i=chicken_breast)
    b. A call that gets recipe details for a specific RECIPE_ID. It should point to “lookup.php” and it takes a parameter “i”, a string representing the recipe ID (e.g check https://www.themealdb.com/api/json/v1/1/lookup.php?i=52923)
2. **ServiceGenerator** (package rest) is a helper class that uses Retrofit2 and for a given Interface creates a Rest API client. You can call it with ServiceGenerator.createService(RestAPI.class); The class is almost fully implemented, you just need to add HttpLoggingInterceptor to obtain the JSON object sent by the server.

Data management in the application must be done through a Room database and Data Transfer Objects (DTOs).

1. The following classes provide access to the database.
    a. **Database** (package database) - a helper class for database instantiation and access. Almost fully complete - you must add a data access object DAO reference.
    b. **RecipeDetails** (package database.entity) - an entity (table) in the database. Completed already.
    c. **RecipeDao** (package database.dao) - an interface defining access methods (queries) on the database. The class has one access method defined, you should add others as needed.
2. Data downloaded from a remote API
    a. Calling the above URLs you should have noticed that the data is downloaded from a remote API as a JSON string. The string structure is different depending on the exact endpoint you called. The data is converted to Java objects called Data Transfer Objects (DTOs). We have provided two such objects for you - **IngredientsDTO** (package models.dto), a high-level object that contains a list of **IngredientDTO** (package models.dto) objects defining individual ingredients. The data you get when you call https://www.themealdb.com/api/json/v1/1/list.php?i=list is converted to IngredientsDTO (and consequently, to a list of IngredientDTOs) when you call getAllIngeredients function of the Rest API client. You must create other DTOs to convert the JSON data to Java objects:

    i. **RecipeDetailsDTO** (package models.dto) - contains a detailed recipe,
    ii. **RecipesByIdDTO** (package models.dto) - a higher-level object created when https://www.themealdb.com/api/json/v1/1/lookup.php?i=RECIPE_ID is called,
    iii. **RecipeSummaryDTO** (package models.dto) - contains a summary of recipes,
    iv. **RecipesByIngredientDTO** (package models.dto) - a higher-level object created when https://www.themealdb.com/api/json/v1/1/filter.php?i=INGREDIENT is called.

b. Since in DetailsActivity we show recipe details for recipes that come either from the server or from the local database, we can save some code by introducing an intermediate class that we will use with for holding the data that comes from either of the two sources. In DetailsActivity we will mostly work with an instance of this class, rather than with RecipeDetails or RecipeDetailsDTO. This class is called **RecipeDetailsIM** (package models) and is provided for you.

c. Similarly, we want to use the same class (RecyclerViewAdapter) in SearchFragment and in FavoritesFragment. Since the adapter shows summaries of recipes, we introduced **RecipeSummaryIM** (package models) class to hold recipe summary data that comes either from the local database (via RecipeDetails class) or from the server. The class is fully completed.

d. **Mapper** (package models) is a helper class that lets you convert to/from RecipeDetailsIM and RecipeSummaryIM. The class is fully completed for the purpose of this project, you just need to uncomment the code.

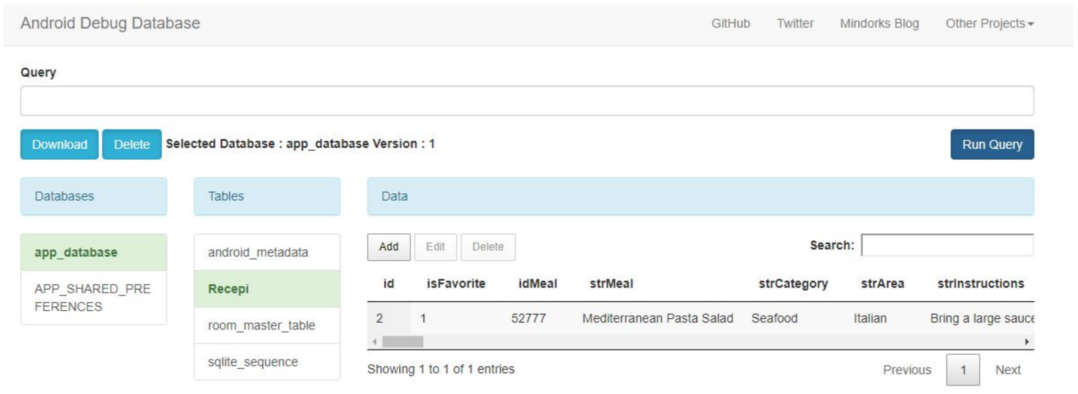
Requirements

1. Start from the code provided at https://github.com/vpejovic/MiniApp3
2. The app must target API version 28;
3. Use Android X (do not use the support libraries);
4. Use only external libraries provided in the gradle file you have downloaded and what the Android SDK and Android X offer you.

Implementation details

- All errors must be handled gracefully

    - If no recipes exist for the selected main ingredient, a user friendly message should be shown instead of the grid with pictures (e.g. “sorry, no recipes for this ingredient exist”)
    - If there are connectivity issues, the app should state that it could not connect to the server.
    - Any other errors that may arise should show an informative message to a user and not crash the app.
- Favorite recipes must be shown in the Favorite recipes tab even if there is no Internet connection.
- Content should be scrollable, if it does not fit a single screen.
- When a user swipes down in SearchFragment, the data should be pulled again from the API. To save bandwidth, this should not be done if a swipe comes within five seconds since the last data download.
- The interface you see in the example screenshots is very rudimentary, you are encouraged to improve it.
- For database debugging the following library has already been added to the Gradle file https://github.com/amitshekhariitbhu/Android-Debug-Database . Sync the project and run app. In the Logcat search for “D/DebugDB: Open http://192.168.1.6:8080 in your browser”. You can also write this code somewhere inside the OnCreate method “Log.d(TAG, "Database Address: " + DebugDB.getAddressLog());”. Now copy the provided address into the browser. Now you can check what you have in the database stored. Below is a screenshot example.



Selected Database : app_database Version : 1

id	isFavorite	idMeal	strMeal	strCategory	strArea	strInstructions
2	1	52777	Mediterranean Pasta Salad	Seafood	Italian	Bring a large sauce

Showing 1 to 1 of 1 entries

Important grading notes

- You must submit your code in a repository titled **PBD2020-MA-3** in your Bitbucket account. The repository must be private and the user “pbdfrita” (pbdfrita@gmail.com) must be added as a read-only member;
- Your project title must be: **MiniApp3**
- Your classes must be named as stated in the text above. You are free to add your helper classes if needed.
- Your project package must be named: **si.uni_lj.fri.pbd.miniapp3**
- **Your app must compile and run solely from the provided Bitbucket code;**
- Your code must be fully anonymous - your email, name, or bitbucket user name must not be shown anywhere in the code/comments;
- Your comments, variable names, etc., must be in English
- The bonus task can bring you up to 3 additional points (see below)

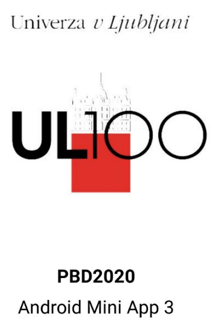
Bonus tasks

There is one bonus task for this lab:

- Background processing. By default, database querying is done on the foreground thread (see “allowMainThreadQueries” in Database class). In this bonus task, you should ensure that the code runs on a background thread.

Example

Splash screen



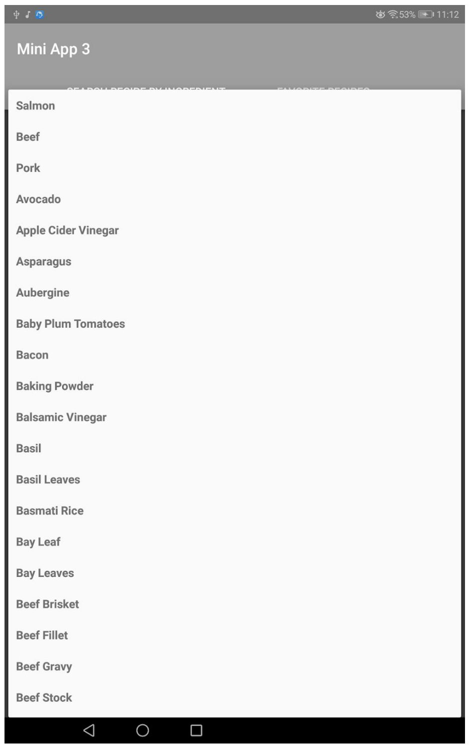
Univerza v Ljubljani

UL100

PBD2020

Android Mini App 3

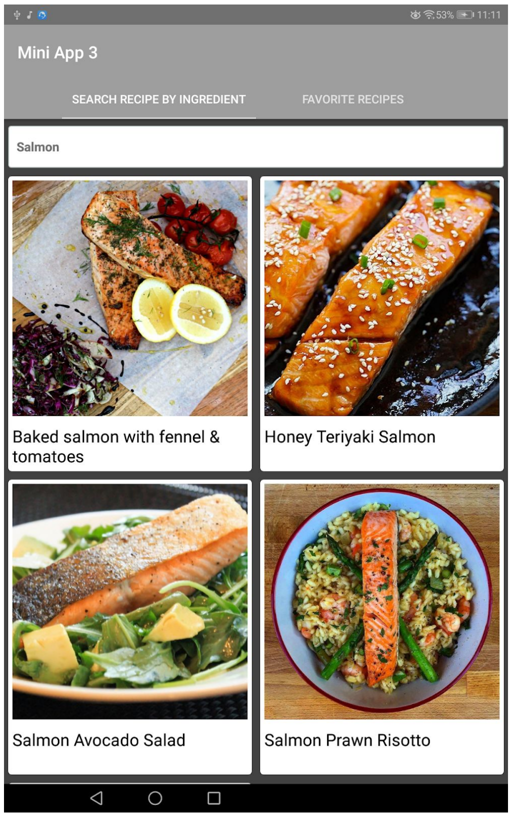
Spinner selection in the SearchFragment



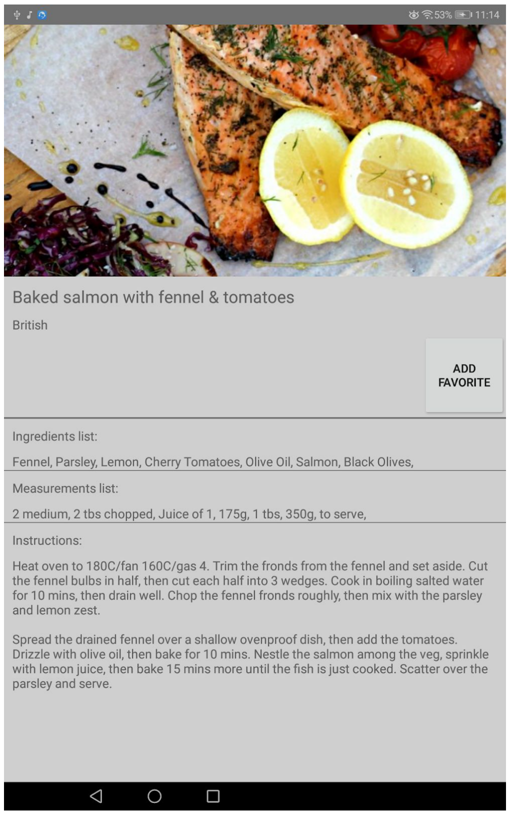
Mini App 3

Salmon
Beef
Pork
Avocado
Apple Cider Vinegar
Asparagus
Aubergine
Baby Plum Tomatoes
Bacon
Baking Powder
Balsamic Vinegar
Basil
Basil Leaves
Basmati Rice
Bay Leaf
Bay Leaves
Beef Brisket
Beef Fillet
Beef Gravy
Beef Stock

SearchFragment with a selection made



DetailsActivity



Baked salmon with fennel & tomatoes

British

ADD FAVORITE

Ingredients list:

Fennel, Parsley, Lemon, Cherry Tomatoes, Olive Oil, Salmon, Black Olives,

Measurements list:

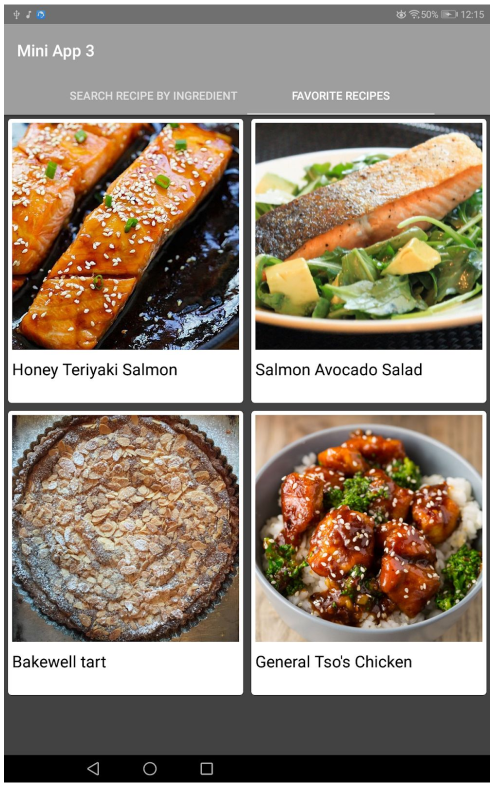
2 medium, 2 tbs chopped, Juice of 1, 175g, 1 tbs, 350g, to serve,

Instructions:

Heat oven to 180C/fan 160C/gas 4. Trim the fronds from the fennel and set aside. Cut the fennel bulbs in half, then cut each half into 3 wedges. Cook in boiling salted water for 10 mins, then drain well. Chop the fennel fronds roughly, then mix with the parsley and lemon zest.

Spread the drained fennel over a shallow ovenproof dish, then add the tomatoes. Drizzle with olive oil, then bake for 10 mins. Nestle the salmon among the veg, sprinkle with lemon juice, then bake 15 mins more until the fish is just cooked. Scatter over the parsley and serve.

FavoritesFragment


50% 12:15
Mini App 3
SEARCH RECIPE BY INGREDIENT
FAVORITE RECIPES
Honey Teriyaki Salmon
Salmon Avocado Salad
Bakewell tart
General Tso's Chicken