CMPS 312 Mobile Application Development

Lab 5 – Displaying Lists

Objective

In this lab, you will practice

- Building UI with Compose
- Manage state in composable functions
- Display lists using LazyColumn

Part A - Tip Calculator App

In this part, we will collaborate and create the following application that allows users to calculate the amount of tip they need to give after a service. The application should allow the user

- to enter the total bill amount,
- to select one of three service quality options [OK, Good, and Amaizing].
- Depending on the service quality they selected, the application should calculate and display the total tip.
- The application should also allow the users to round up the tip. See Fig 1.

1. Create a new project place it under the **`Lab5-TipCalculatorApp`** folder on your GitHub repo.
2. Name the project "TipCalculator."
3. Use `cmps312.tipcalculator` as the package.
4. Select *Empty Compose Activity* as the project template.
5. Try to create the tip app that is described above, which has a similar UI as Fig. 1

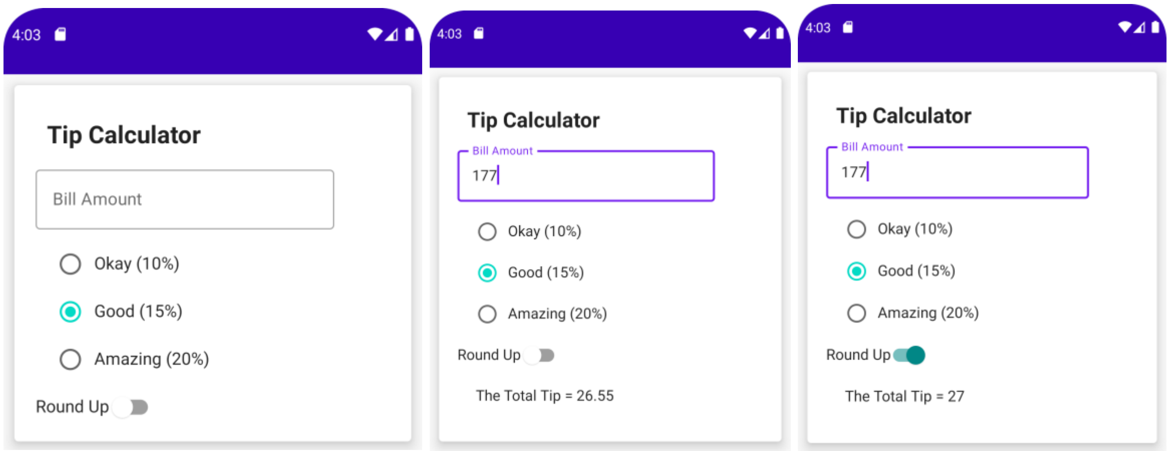
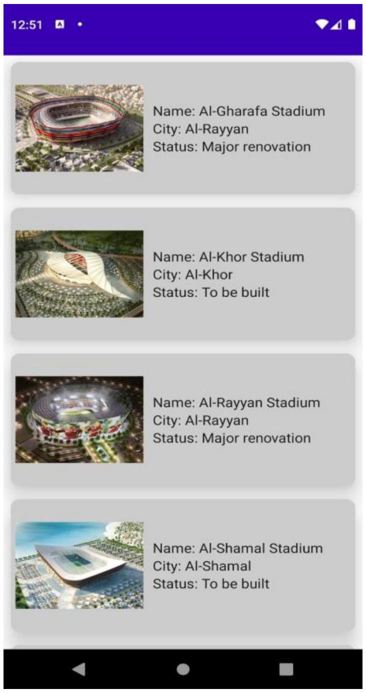


Figure 1. Tip Calculator App

Part B – Stadium App

In Part B, you will build the Stadiums App shown in Figure 2 by reading the stadiums details from a JSON file then displaying them in a LazyColumn.



Name: Al-Gharafa Stadium
City: Al-Rayyan
Status: Major renovation

Name: Al-Khor Stadium
City: Al-Khor
Status: To be built

Name: Al-Rayyan Stadium
City: Al-Rayyan
Status: Major renovation

Name: Al-Shamal Stadium
City: Al-Shamal
Status: To be built

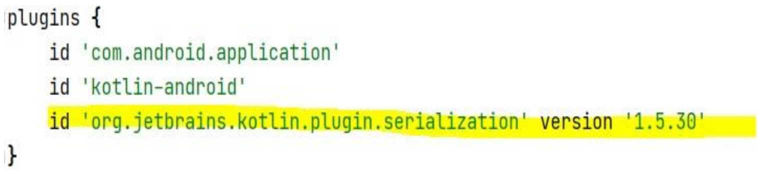
Figure 2. Stadiums App

1. Sync Lab GitHub repo to get the resources needed for this project.
2. Create a new project place it under the `Lab5-StadiumApp` folder on your GitHub repo. Name the project "Qatar 2022 Stadiums". Use `cmps312.stadiumapp` as the package. Select *Empty Compose Activity* as the project template.
   Note all Kotlin files related to the UI related should be added under `cmps312.stadiumapp.UI` package and the ones related to the app logic should be placed under `cmps312.stadiumapp.model` package.
3. Add the following dependency to the **module** build.gradle file.

```
//For Kotlin Serialization
implementation 'org.jetbrains.kotlinx:kotlinx-serialization-json:1.2.2'
```

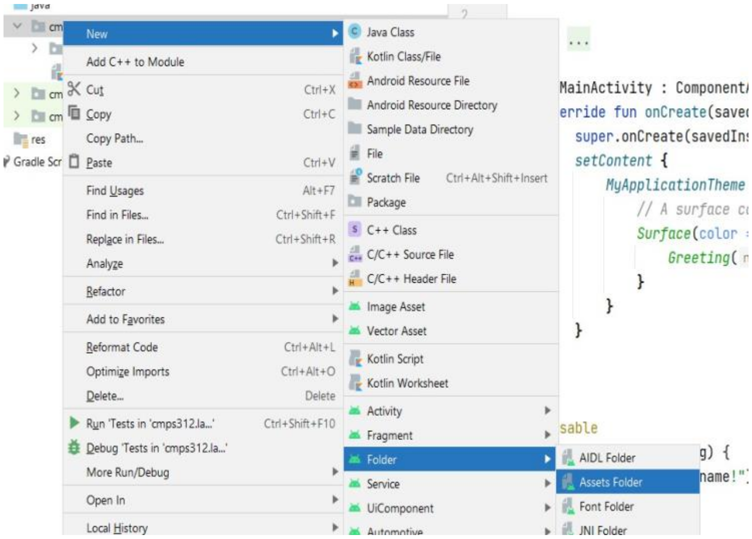
4. Also add the following plug-in:

```
id 'org.jetbrains.kotlin.plugin.serialization' version '1.5.30'
```



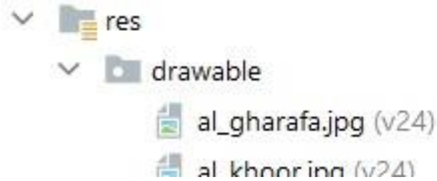
```
plugins {
    id 'com.android.application'
    id 'kotlin-android'
    id 'org.jetbrains.kotlin.plugin.serialization' version '1.5.30'
}
```

5. Create an **assets** folder and add the ***stadium.json*** (available under Lab5 folder)



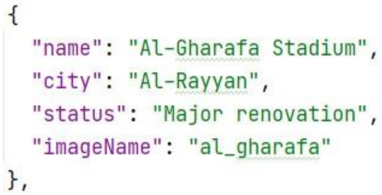
```
...
MainActivity : Component/
erride fun onCreate(save
  super.onCreate(savedIns
  setContent {
      MyApplicationTheme
          // A surface c
          Surface(color =
              Greeting( n
          }
      }
  }
sable
g) {
name!"
```

6. Copy all the stadium images from ***Lab5/images*** folder and paste them to **res>drawable** folder



```
res
  drawable
    al_gharafa.jpg (v24)
    al_khoor.jpg (v24)
```

7. Create a data class called **Stadium** (in a Kotlin file named Stadium). Derive Stadium properties from the JSON object shown below. Make sure you annotate the class with `@Serializable`.



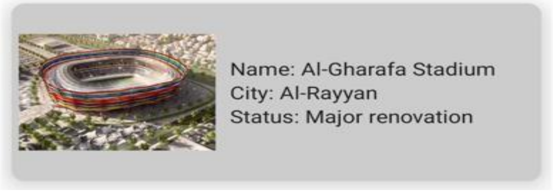
```
{
  "name": "Al-Gharafa Stadium",
  "city": "Al-Rayyan",
  "status": "Major renovation",
  "imageName": "al_gharafa"
},
```

8. Add **`StadiumRepository`** object. Add **`getStadiums()`** function to read the **stadiums.json** file a list of stadiums. Stadiums retrieved from the JSON file should be cached in the `StadiumRepository` object to avoid repetitive reads.

1. Test your `StadiumRepository.`**`getStadiums()`** in the MainActivity and display all the stadiums on the Logcat using Log.d(…).

2. Create a **`StadiumCard`** Kotlin file and build **`StadiumCard`** composable to display a stadium, as shown in Figure 3.

Tip: add appropriate modifiers and properties such as elevation, shape to achieve the desired design

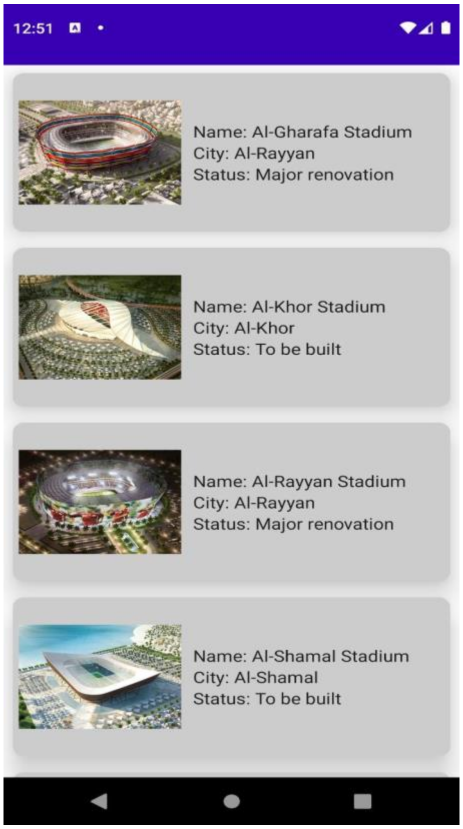


Name: Al-Gharafa Stadium
City: Al-Rayyan
Status: Major renovation

Figure 3. Stadium Card Composable

3. Add **`StadiumCardPreview`** composable function to test the **`StadiumCard`** composable using AlGharafa Stadium details as shown in Figure 3.
4. Create **`StadiumScreen`** Kotlin file and build **`StadiumScreen`** composable to display the stadium returned by `StadiumRepository.`**`getStadiums()`** using a `LazyColumn`.
5. Add **`StadiumScreenPreview`** composable to test the **`StadiumScreen`** composable as shown in Figure 4.
6. Load the **`StadiumScreen`** in the MainActivity, then test the whole app as shown in Figure 4.
7. Experiment with changing `LazyColumn` to `LazyRow` and retest your app.

Remember to test as you go and push your work to the GitHub repository once completed.



Name: Al-Gharafa Stadium
City: Al-Rayyan
Status: Major renovation

Name: Al-Khor Stadium
City: Al-Khor
Status: To be built

Name: Al-Rayyan Stadium
City: Al-Rayyan
Status: Major renovation

Name: Al-Shamal Stadium
City: Al-Shamal
Status: To be built

Figure 4. Stadiums List