CMPS 312 Mobile Application Development

Lab 5 – Compose Layouts and Lists

Objective

In this lab, you will practice

- Learn how to analyze and build apps with Compose
- Display lists using Lazy Column , Row and Horizontal Grid
- Interact with lists such as Search, Sort.
- Add Navigation UI components such as TopAppBar , BottomBar using Material Composobles Scafold

Overview

- **Part A -** Baisc Layout in Compose CodeLab
- **Part B -** Implement Tip Calculator App
- **Part C -** Implement Student Management App.
- **Part D** –List Interaction.

Part A – Jetpack Compose Layouts

Complete Basic layouts in Compose codelab and practice how to use modifiers, standard layout components like Column and LazyRow, alignments and arrangements, Material composables like Scaffold and Bottom Navigation, slot APIs, and build layouts for different screen configurations.

Part B - Tip Calculator App

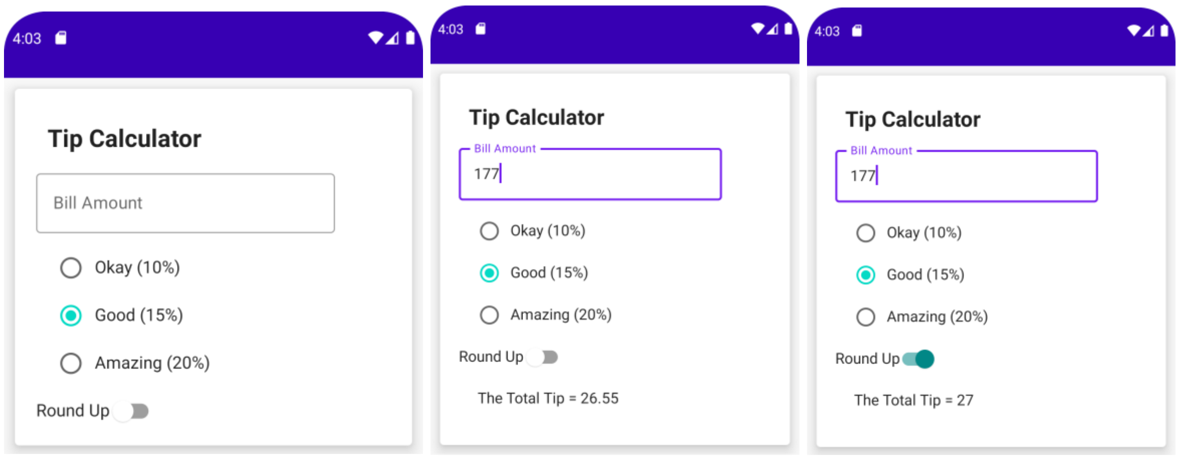
In this part, we will collaborate and create the following application that allows users to calculate the amount of tip they need to give after a service. The application should allow the user

- 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.

- Name the project TipCalculator
- Use `cmps312.tipcalculator` as the package.
- Select *Empty Compose Activity* as the project template.

2. Create TipCalculator app have the UI shown in Figure 1.


4:03

Tip Calculator

Bill Amount

( ) Okay (10%)
(●) Good (15%)
( ) Amazing (20%)

Round Up [off]

4:03

Tip Calculator

Bill Amount: 177

( ) Okay (10%)
(●) Good (15%)
( ) Amazing (20%)

Round Up [off]

The Total Tip = 26.55

4:03

Tip Calculator

Bill Amount: 177

( ) Okay (10%)
(●) Good (15%)
( ) Amazing (20%)

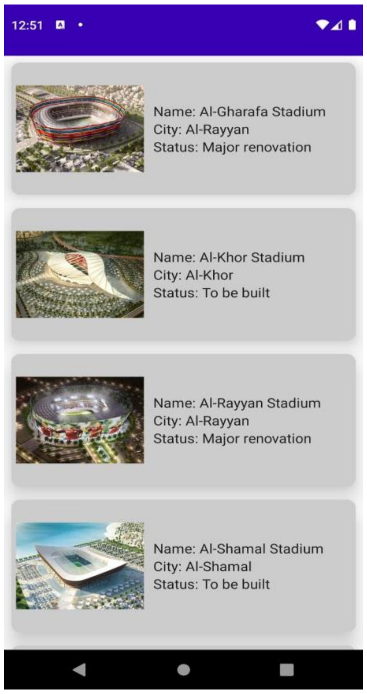
Round Up [on]

The Total Tip = 27

Figure 1. Tip Calculator App

<u>Part C – Stadium App</u>

In Part C, 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.


12:51

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

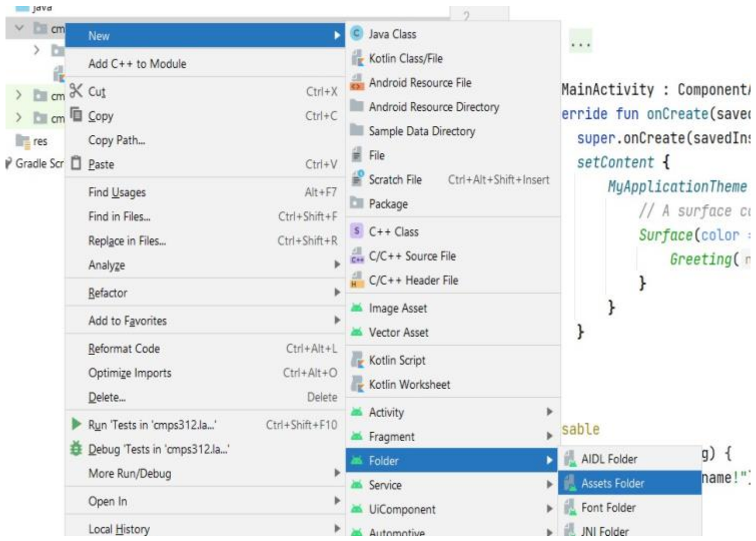
Figure 2. Stadiums App

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.

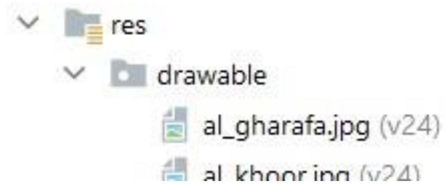
3. Add the following kotlin serialization dependency to the **module** build.gradle file.

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

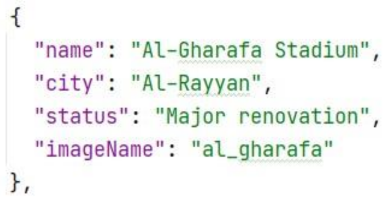


```
MainActivity : Component
erride fun onCreate(save
  super.onCreate(savedIn
  setContent {
      MyApplicationTheme
          // A surface c
          Surface(color
              Greeting(
          }
      }
  }
sable
g) {
name!"
```

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



6. 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`.



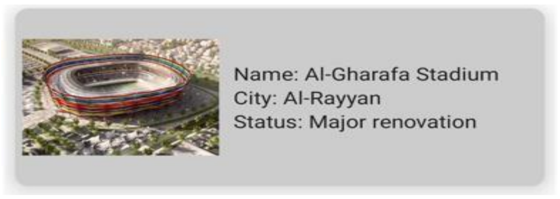
```
{
  "name": "Al-Gharafa Stadium",
  "city": "Al-Rayyan",
  "status": "Major renovation",
  "imageName": "al_gharafa"
},
```

7. 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

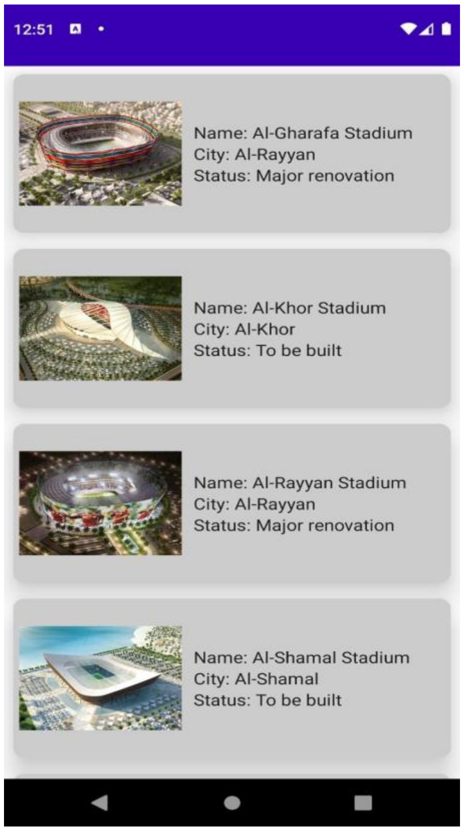


Name: Al-Gharafa Stadium
City: Al-Rayyan
Status: Major renovation

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.


12:51

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

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

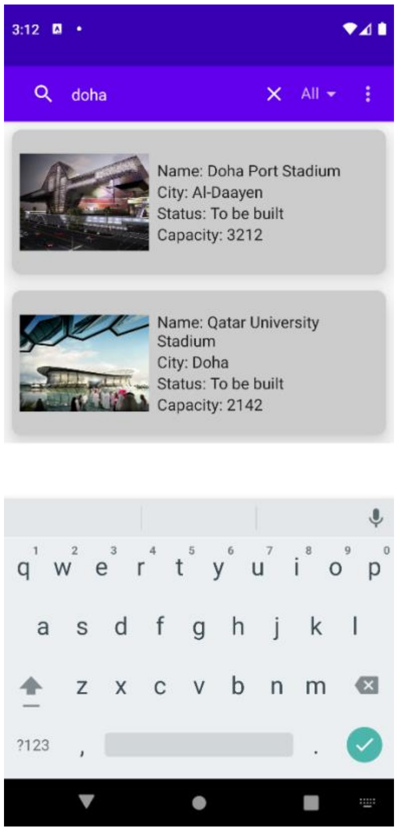
Name: Al-Rayyan Stadium
City: Al-Rayyan
Status: Major renovation

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

Figure 4. Stadiums List

Part D – List Interaction

1. Use the provided model solution of Qatar2022 app we created in the previous part C. Note that the json file and the app has been updated to add a new item seating capacity.
2. As shown in Figure 1, add a Top App Bar with Search Box to allow users to search stadiums.

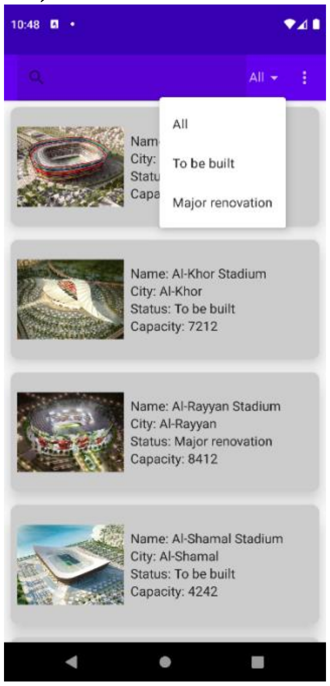


Name: Doha Port Stadium
City: Al-Daayen
Status: To be built
Capacity: 3212

Name: Qatar University Stadium
City: Doha
Status: To be built
Capacity: 2142

Figure 1.Search stadiums

3. As shown in Figure 2, add a Dropdown list to allow filtering the stadiums based on their status (To be built, Major Renovation or All).



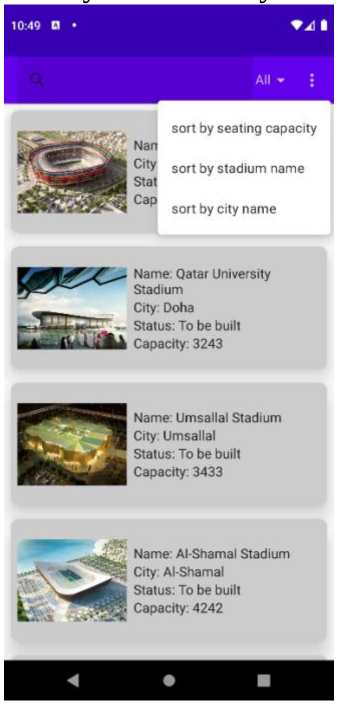
Name: Al-Khor Stadium
City: Al-Khor
Status: To be built
Capacity: 7212

Name: Al-Rayyan Stadium
City: Al-Rayyan
Status: Major renovation
Capacity: 8412

Name: Al-Shamal Stadium
City: Al-Shamal
Status: To be built
Capacity: 4242

Figure 2. Filter stadiums based on status

4. As shown in Figure 3, Create a sort menu that sorts the stadiums based on seating capacity, and alphabetically based on city and stadium name.



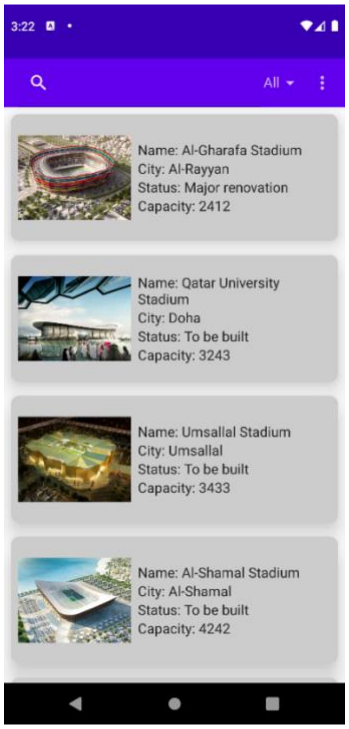
Nam
City
Stat
Cap

Name: Qatar University Stadium
City: Doha
Status: To be built
Capacity: 3243

Name: Umsallal Stadium
City: Umsallal
Status: To be built
Capacity: 3433

Name: Al-Shamal Stadium
City: Al-Shamal
Status: To be built
Capacity: 4242

Stadiums are sorted by seating capacity



Name: Al-Gharafa Stadium
City: Al-Rayyan
Status: Major renovation
Capacity: 2412

Name: Qatar University Stadium
City: Doha
Status: To be built
Capacity: 3243

Name: Umsallal Stadium
City: Umsallal
Status: To be built
Capacity: 3433

Name: Al-Shamal Stadium
City: Al-Shamal
Status: To be built
Capacity: 4242

Stadiums are sorted by name

Figure 3.Stadium sort options