

# CMPS 312 Mobile Application Development

## Lab 5 – Compose Components & Layouts

---

### Objective

Practice implementing mobile apps UI using compose components and layouts.

### Overview

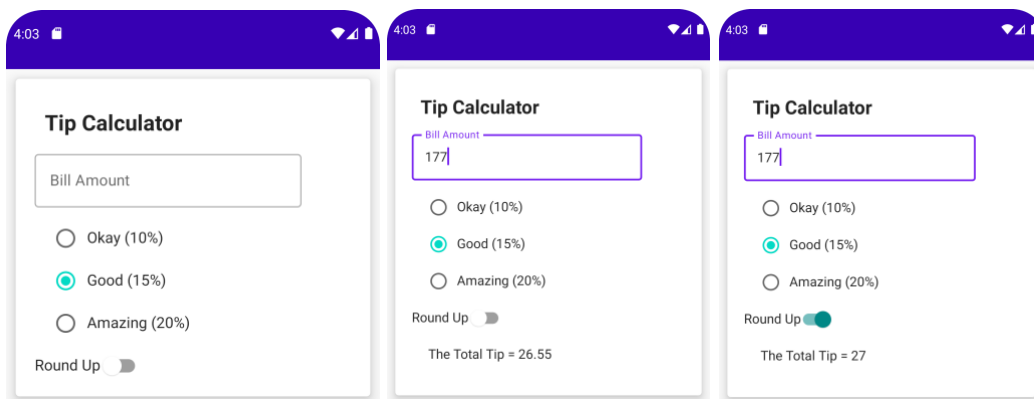
- **Part A** - Basic layouts in Compose CodeLab
- **Part B** - Implement Tip Calculator App
- **Part C** - Implement Stadiums App

### Part B - Tip Calculator App

Implement a Tip Calculator App, as shown in Figure 1, to allow the user to:

- Enter the total bill amount,
- Select one of three service quality options: Okay, Good, and Amazing.
- Depending on the service quality, the app should calculate and display the tip amount.
- The app should also allow the users to round up the tip.

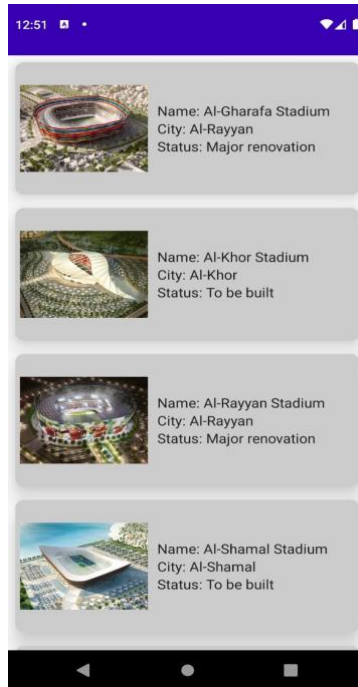
1. Create a new project named **TipCalculator** under **Lab5\TipCalculatorApp** folder on your GitHub repo.
  - Use `cmps312.tipcalculator` as the package.
  - Select *Empty Compose Activity* as the project template.
2. Implement the TipCalculator offering the UI and features shown in Figure 1.



**Figure 1. Tip Calculator App**

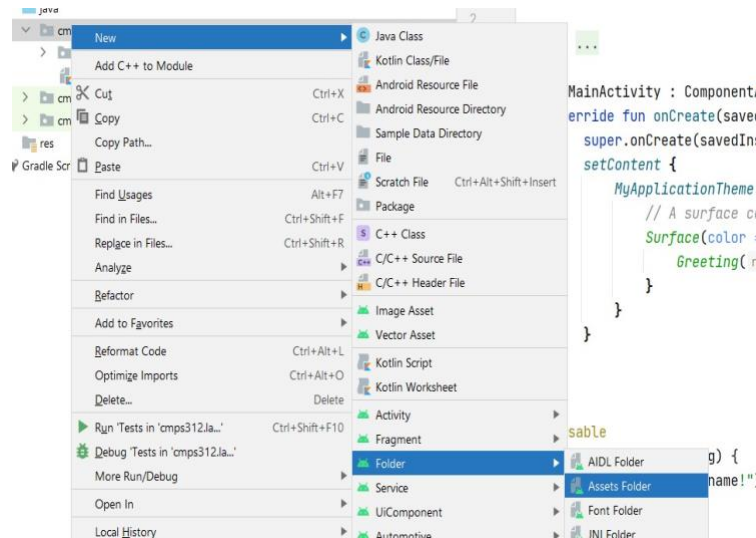
### Part C – Stadiums App

Implement the Stadiums App shown in Figure 2 by reading the stadiums details from a JSON file then displaying them in a LazyColumn.



**Figure 2. Stadiums App**

1. Sync Lab GitHub repo to get the resources needed for this project.
2. Create a new project named Qatar Stadiums place it under the **Lab5\StadiumApp** folder on your GitHub repo.  
Use `cmps312.stadiumapp` as the package. Select *Empty Compose Activity* as the project template.  
Place all Kotlin files related to the UI under `cmps312.stadiumapp.UI` package and the ones related to the app logic under `cmps312.stadiumapp.model` package.
3. Add the Kotlin serialization dependency to the **module** build.gradle file as explained in the [documentation](#).
4. Create an **assets** folder and add the *stadium.json* (available under Lab5 folder)



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



- Create a data class named **Stadium** (in a Kotlin file named Stadium) and annotate it with **@Serializable**. Derive Stadium properties from the JSON object shown below.

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

- 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.
- Test your StadiumRepository.**getStadiums()** in the MainActivity and display all the stadiums on the Logcat using Log.d( ...).
- Add a **StadiumCard** Kotlin file and implement **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.



**Figure 3. Stadium Card Composable**

10. Add **StadiumCardPreview** composable function to test the **StadiumCard** composable using AlGharafa Stadium details as shown in Figure 3.
11. Create **StadiumList** Kotlin file and implement **StadiumList** composable to display the stadiums returned by `StadiumRepository.getStadiums()` in a `LazyColumn`.
12. Add **StadiumListPreview** composable to test the **StadiumList** composable as shown in Figure 2.
13. Load the **StadiumList** in the MainActivity, then test the whole app.
14. 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.