CMPS 312 Mobile Application Development
Lab 5: Flutter Widgets and Layouts Basics

Lab Overview

In this lab, you will practice User Interface (UI) development using Flutter. You will build a **Tip Calculator** and a **Banking** app, utilizing essential widgets and layout components to create visually appealing UIs.

Lab Objectives

By the end of this lab, you should be able to:

- **Set up and configure a Flutter app** using **MaterialApp** as the main entry point for theming and navigation.
- **Build a UI using key Flutter widgets** , including:
  - Scaffold to create the app structure that includes key elements such as a AppBar, and BottomNavigationBar.
  - AppBar for app title, navigation controls and actions.
  - BottomNavigationBar for navigation between major app screens.
  - Text, Icon, and Center for content display and alignment.
- **Implement layout widgets** such as:
  - Row and Column for responsive horizontal and vertical layouts.
  - Expanded to dynamically adjust widget sizes based on available space within a layout.
  - Card for displaying grouped content.
  - Switch to allow users to toggle between states.
  - Container and Padding for layout spacing and alignment.
- **Display images** using Image.asset for local assets and Image.network for online images.
- **Enhance UI appearance** with styling, theming (ThemeData), and Material Design components while ensuring a consistent look & feel of the app.
- Keeping your Flutter code organized and modular.

Lab Overview

This lab is divided into 3 main parts:

1. **Part A – Flutter warm-up by completing this code lab**
   https://codelabs.developers.google.com/codelabs/flutter-codelab-first
2. **Part B – Tip Calculator app:** a simple app to calculate tips for a waiter.
3. **Part C - Banking app:** design and implement the UI for a basic banking app.

Part B - Tip Calculator app

1) **Create a New Flutter Project**
   - Open your terminal or command prompt.
   - Run the command to create a new Flutter project named tip_app

flutter create tip_app .

Alternatively, you can use the IDE to create the project by using the following steps

I. Open VS Code.
II. Press **Ctrl + Shift + P** (or **Cmd + Shift + P** on Mac) to open the command palette.
III. In the command palette, type **Flutter: New Project** and select it.
IV. Choose Flutter Application.
V. Choose the location where you want to save your project.
VI. Enter a name for your project (e.g., tip_app), and press Enter.

2) **Open the Project**
   - Navigate to the project directory and open it in your preferred IDE (e.g., VS Code, Android Studio).
3) **Run the default app**
   - Ensure you have an emulator or a physical device connected.
   - Run the app to verify that everything is set up correctly.
4) **Explore the Project Structure**
   - Familiarize yourself with key files and directories:
     - lib/main.dart: Entry point of the application.
     - pubspec.yaml: Manages project dependencies and assets.
     - android, ios: Platform-specific code.
5) Create lib/tip_calculator.dart file and implement **`TipCalculator`** as a stateful widget having a Scaffold with an AppBar and a body section. The screen design is shown in figure 1.

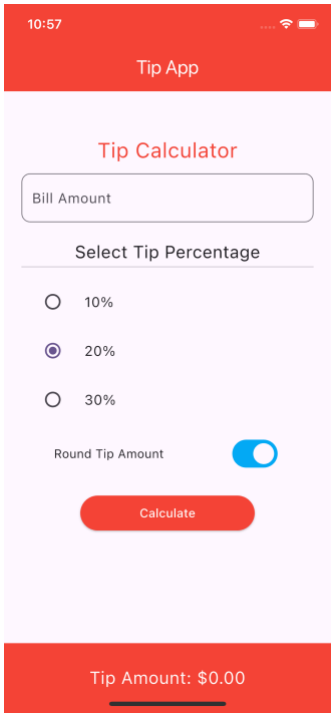

Figure 1 : Tip App

6) Run the app to test its functionality.

Part C - Banking app

You are tasked with designing and implementing a basic UI for a banking app using Flutter. The app displays the account card details and provides buttons for account transactions.

1) **Create a New Flutter Project**
   - Open your terminal or command prompt.
   - Run the command to create a new Flutter project named banking_app
2) **Open the Project** Navigate to the project directory and open it in your preferred IDE (e.g., VS Code, Android Studio).
3) **Run the default app**
4) Keep main.dart: Main application entry point. Add lib/home_page.dart file and design and implement the following HomePage:

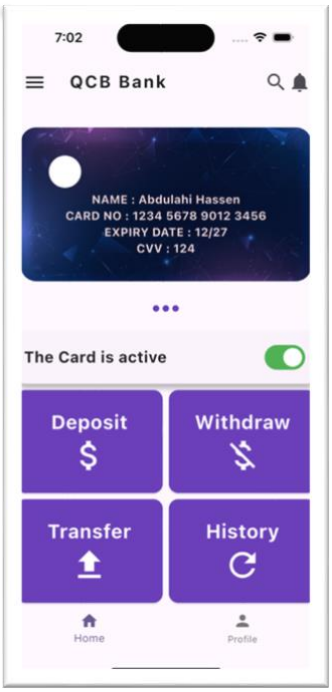

Figure 2. Banking App

To enhance the modularity and maintainability of your Flutter application, split HomePage code into several files, placed inside `widgets` folder, each representing a logical component of the app.

```
lib/
├── constants.dart              # Contains app-wide constants like text styles, colors, etc.
├── main.dart                   # Entry point of your Flutter application
├── widgets/                    # Folder to organize reusable widgets
│   ├── app_bar.dart            # Defines the AppBar widget
│   ├── credit_card.dart        # Defines the CreditCard widget
│   ├── bottom_nav.dart         # Defines the BottomNavigationBar widget
│   └── home_page.dart          # Defines the HomePage, which brings all components together
```

5) Run and test your code as you make progress.
6) As you make progress push your implementation to your GitHub repository.