# Experiment 2

**Name: Soham Hajare** **D15B/19**

**Aim**: To design Flutter UI by including common widgets.

**Introduction:**

Flutter, developed by Google, is a popular open-source UI toolkit for building natively compiled applications for mobile, web, and desktop from a single codebase. Flutter excels in creating visually appealing and responsive user interfaces. The key to this lies in the extensive use of common widgets that Flutter provides out of the box.

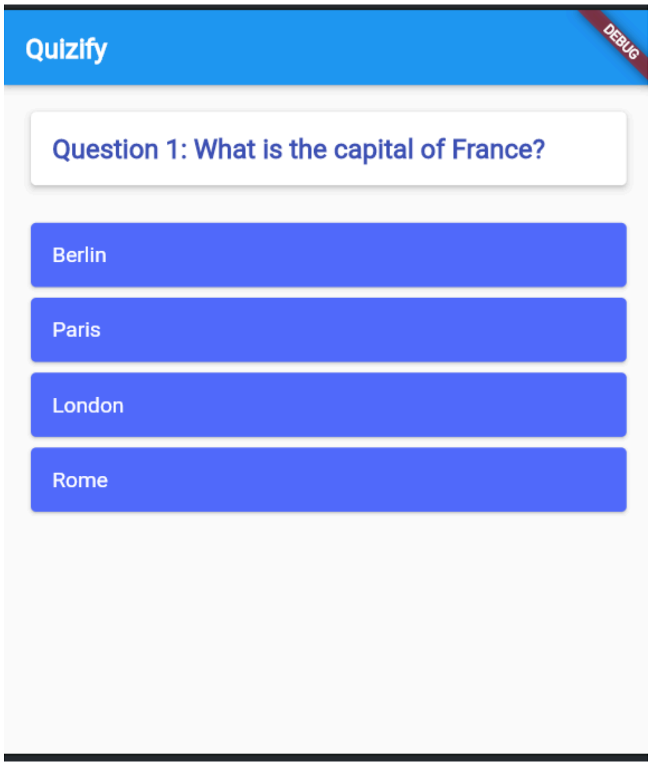
Widgets in Flutter are the building blocks of the user interface, representing everything from buttons and text to complex layouts and animations. These widgets are highly customizable and can be combined to create diverse and engaging user interfaces. In this context, a "widget" refers to both visual elements and structural components of the app.

**Code:**

**main.dart**

```
import 'package:flutter/material.dart';


void main() {
  runApp(MyApp());
}


class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      theme: ThemeData(
        primaryColor: Colors.indigo,
        hintColor: Colors.amber,
        fontFamily: 'Montserrat',
      ),
      home: Scaffold(
        appBar: AppBar(
          title: Text('Quizify', style: TextStyle(fontWeight: FontWeight.bold)),
        ),
        body: QuizPage(),
      ),
    );
  }
}


class QuizPage extends StatefulWidget {
  @override
  _QuizPageState createState() => _QuizPageState();
}


class _QuizPageState extends State<QuizPage> {
  List<Map<String, dynamic>> quizData = [
    {
      'question': 'What is the capital of France?',
      'options': ['Berlin', 'Paris', 'London', 'Rome'],
      'correctAnswer': 'Paris',
    },
    {
      'question': 'Which planet is known as the Red Planet?',
      'options': ['Venus', 'Mars', 'Jupiter', 'Saturn'],
      'correctAnswer': 'Mars',
    },
    // Add more questions as needed
  ];


  int currentQuestionIndex = 0;


  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: const EdgeInsets.all(16.0),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.stretch,
        children: [
          Card(
            elevation: 4,
            color: Colors.white,
            child: Padding(
              padding: const EdgeInsets.all(16.0),
              child: Text(
                'Question ${currentQuestionIndex + 1}:
${quizData[currentQuestionIndex]['question']}',
                style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold, color:
Colors.indigo),
              ),
            ),
          ),
          SizedBox(height: 20),
          Column(
            children: List.generate(
              quizData[currentQuestionIndex]['options'].length,
              (index) => Card(
                elevation: 2,
                color: Colors.indigoAccent,
                child: ListTile(
                  title: Text(
                    quizData[currentQuestionIndex]['options'][index],
                    style: TextStyle(color: Colors.white),
                  ),
                  onTap: () {
                    // Add your logic to handle option selection and move to the next question
                    print('Selected: ${quizData[currentQuestionIndex]['options'][index]}');
                    moveToNextQuestion();
                  },
                ),
              ),
            ),
          ),
        ],
      ),
    );
  }
  void moveToNextQuestion() {
    if (currentQuestionIndex < quizData.length - 1) {
      setState(() {
        currentQuestionIndex++;
      });
    } else {
      // Display quiz completion or navigate to the result screen
      print('Quiz completed!');
      // Add your logic for quiz completion
    }
  }
}
```

**Output:**



**Explanation:**

MyApp Class (StatelessWidget):

- The `MyApp` class represents the entry point of the application and extends the `StatelessWidget`.
- The `MaterialApp` widget is used to define app-wide visual styling. Key properties include:
    - `theme`: Sets the overall theme with primary color (indigo), hint color (amber), and font family ('Montserrat').
    - `home`: Specifies the home screen, which is a `Scaffold` widget.

Scaffold:

- The `Scaffold` widget provides the basic structure of the app, including an app bar and body content.
- The `appBar` property is set to an `AppBar` widget with the title "Quizify".
- The `body` property is set to an instance of the `QuizPage` class, defining the main content of the app.

AppBar:

- The `AppBar` widget is displayed at the top of the app.
- The title is set to a `Text` widget with the content "Quizify".

QuizPage Class (StatefulWidget):

- `QuizPage` is a stateful widget that manages the state of the quiz.
- The `build` method defines the layout of the widget.

Column:

- The `Column` widget arranges its children in a vertical column.

Padding:

- The `Padding` widget adds padding around its child widgets.

Card:

- The `Card` widget represents a material design card with a shadow.
- Key properties include `elevation` for shadow and `color` for background (set to white).

Text:

- The `Text` widget displays the current question using data from `quizData`.
- Styling includes font size, weight, and color.

SizedBox:

- The `SizedBox` widget adds a fixed-height empty space.

Column (nested):

- Another `Column` widget is nested to organize the list of options for the current question.

List.generate:

- The `List.generate` function generates a list of widgets based on the number of options for the current question.

Card (for each option):

- Another `Card` widget is used for each option, with elevation for a shadow and a background color set to `Colors.indigoAccent`.

ListTile:

- The `ListTile` widget displays the option text with styling, including color set to white.
- The `onTap` property is set to the `moveToNextQuestion` method, which handles the user's interaction.

moveToNextQuestion method:

- The `moveToNextQuestion` method checks if there are more questions using `currentQuestionIndex`.
- If more questions exist, it updates `currentQuestionIndex` with `setState` to trigger a rebuild.
- If no more questions, it prints "Quiz completed!"

**Conclusion:**

This experiment demonstrates the use of common Flutter widgets to create a basic login screen. Flutter provides a rich set of widgets that can be easily combined to build complex and beautiful user interfaces. Customization and theming can be applied to enhance the visual appeal of the app. This is just a simple example, and you can extend and modify it based on your specific requirements. Remember to handle the actual authentication logic securely in a real-world application.