Practical No. 13

Title: Flutter program based on RestAPI

Aim: Create a Flutter application to demonstrate REST API

Introduction

Flutter is a popular open-source UI software development toolkit created by Google. It allows developers to build natively compiled applications for mobile, web, and desktop from a single codebase. When integrating Flutter with a RESTful API, developers can create dynamic and data-driven applications. Below is a brief introduction to Flutter programming using REST API

Creating a Flutter Project:

1. Create a New Project:

    a. Open your terminal and run flutter create my_flutter_app.

    b. Navigate into the project directory using cd my_flutter_app.

2. Dependencies:

    a. Open the pubspec.yaml file and add dependencies for HTTP requests.
    Example:

    dependencies:

    flutter:

    sdk: flutter

    http: ^0.13.3

    b. Run flutter pub get in the terminal to fetch the dependencies.

Making REST API Calls:

1. Import HTTP Package:

    a. Import the HTTP package in your Dart file where you want to make API calls.

```
import 'package:http/http.dart' as http;
```

2. Define API Endpoint:

    a. Set the API endpoint and any required headers.

```
final String apiUrl = 'https://example.com/api/data';
```

3. Making GET Request:

    a. Use the http.get method to make a GET request.

4. Making POST Request:

    a. For a POST request, use the http.post method.

This is a basic introduction to integrating Flutter with a REST API. As you progress, you may explore more advanced topics such as state management, error handling, and optimizing API calls for performance.

Exercise - Design flutter application using RestAPI

Implementation:

Program:

main.dart

```
import 'package:flutter/material.dart';
import 'package:practical_three_api/gender.dart';
import 'dart:async';
import 'dart:convert';
import 'package:http/http.dart' as http;

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

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
       debugShowCheckedModeBanner: false,
       title: 'Flutter Gender Prediction',
      theme: ThemeData(
       colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
       useMaterial3: true,
```

```
    ),
    home: const MyHomePage(title: 'Gender Prediction'),
   );
  }
}

class MyHomePage extends StatefulWidget {
  const MyHomePage({super.key, required this.title});

  final String title;

  @override
  State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  late Future<Gender> gender;
  final nameController = TextEditingController();
  bool _validate = false;
  String genderOutput = "";
  String probOutput = "";
  Future<Gender> getOutput(String name) async {
```

```
  final response = await
http.get(Uri.parse("https://api.genderize.io/?name=${name.toLowerCase().trim()}"));
  if (response.statusCode == 200) {
   // ignore_for_file: avoid_print
   print(jsonDecode(response.body));
   setState(() {
    genderOutput = "Gender: ${jsonDecode(response.body)["gender"].toString()}";
    probOutput = "Probability: ${jsonDecode(response.body)["probability"]}";
   });
   return Gender.fromJson(jsonDecode(response.body));
  }
  else {
   print("Failed to load data");
   throw Exception('Failed to load data');
  }
 }
 @override
 void initState() {
  super.initState();
 }

 @override
```

```
Widget build(BuildContext context) {
 return Scaffold(
  appBar: AppBar(
   //title: Text(widget.title),
  ),
  body: Center(
   // Center is a layout widget. It takes a single child and positions it
   // in the middle of the parent.
   child: Container(
    alignment: Alignment.center,
    color: Colors.white,
    margin: const EdgeInsets.only(left: 20.0, right: 20.0),
    child: Column(
     children: <Widget>[
      const Text(
       'Gender Prediction',
       style: TextStyle(fontSize: 28.0),
      ),
      TextField(
       controller: nameController,
       style: const TextStyle(fontSize: 20.0),
       decoration: InputDecoration(
```

```
    border: InputBorder.none,
    labelText: 'Enter Your Name',
    hintText: 'Enter Your Name',
    errorText: _validate? "Can't be empty" :null,
   ),
 ),
 ElevatedButton(
    style: ElevatedButton.styleFrom(
     backgroundColor: Colors.amber,
     textStyle: const TextStyle(
        color: Colors.black,
        fontSize: 20,
        fontStyle: FontStyle.normal),
    ),
    onPressed: () {
     setState(() {
      _validate = nameController.text.isEmpty;

      if(_validate == false)
       {
         gender = getOutput(nameController.text);
       }
```

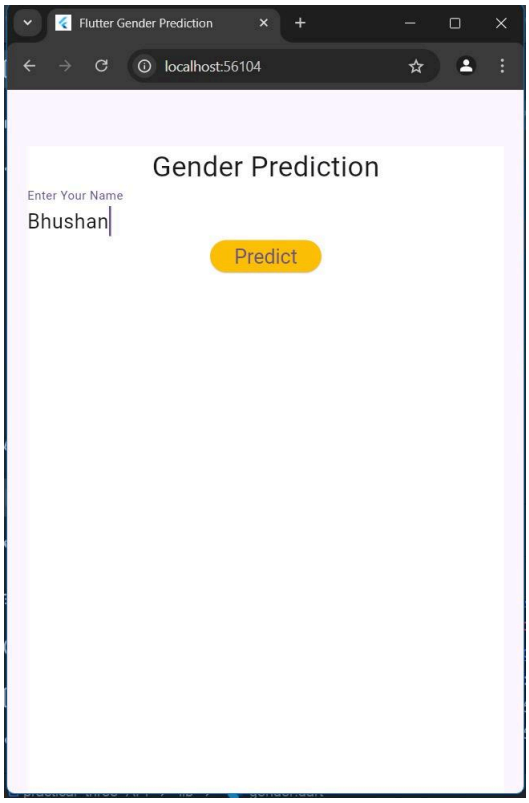
```
            });


          },
          child: const Text(
            'Predict')),
        Text(
          genderOutput,
          style: const TextStyle(fontSize: 20.0),
        ),
        Text(
          probOutput,
          style: const TextStyle(fontSize: 20.0),
        ),
      ],
    ),
  ),
 );
}
}
```

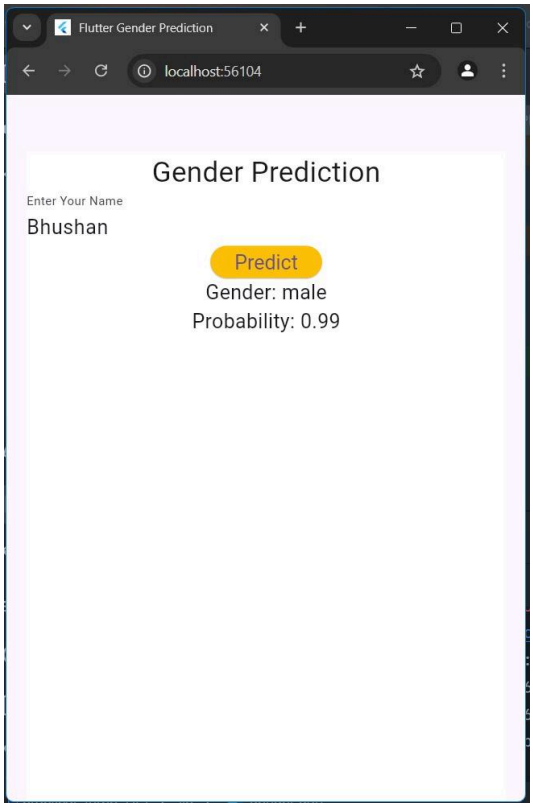
gender.dart

```
class Gender {
```

```
  final String? gender;
  final String? name;
  final double? probability;
  Gender(
     this.gender,
     this.name,
     this.probability,
     );
  factory Gender.fromMap(Map<String, dynamic> json) {
   return Gender(json['gender'], json['name'], json['probability']);
  }
  factory Gender.fromJson(Map<String, dynamic> json) {
   return Gender(json['gender'], json['name'], json['probability']);
  }
}
```

Output:





Conclusion -

Understanding integrating Flutter with a RESTful API too create a Flutter application to demonstrate REST API