

**SEB-312 Mobile Application Development**

**LAB # 10**

**LAB Title**

|  |
| --- |
| Making HTTP POST requests using the http package |

**Assessment of CLO: 04, PLO: 05**

|  |  |  |  |
| --- | --- | --- | --- |
| **Student Name:** |  | | |
| **Roll No.** |  | | |
| **Semester** |  | **Session** |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **S. No.** | **Perf. Level**  **Criteria** | **Excellent**  **(2.5)** | **Good**  **(2)** | **Satisfactory**  **(1.5)** | **Needs Improvement**  **(0 ~ 1)** | **Marks Obtained** |
| **1** | Project Execution & Implementation | Fully functional, optimized, and well-structured. | Minor errors, mostly functional. | Some errors, requires guidance. | Major errors, non-functional, or not Performed. |  |
| **2** | Results & Debugging  Or Troubleshooting | Accurate results with effective debugging  Or Troubleshooting. | Mostly correct, some debugging Or Troubleshooting needed. | Partial results, minimal debugging  Or Troubleshooting. | Incorrect results, no debugging Or Troubleshooting, or not attempted. |  |
| **3** | Problem-Solving & Adaptability  (VIVA) | Creative approach, efficiently solves challenges. | Adapts well, minor struggles. | Some adaptability, needs guidance. | Lacks innovation or no innovation, unable to solve problems. |  |
| **4** | Report Quality & Documentation | Clear, structured, with detailed visuals. | Mostly clear, minor gaps. | Some clarity issues, missing details. | Poorly structured, lacks clarity, or not submitted. |  |
| **Total Marks Obtained Out of 10** | | | | | |  |

**Experiment evaluated by**

|  |  |  |  |
| --- | --- | --- | --- |
| **Instructor’s Name** | **Sidra Khatoon** | | |
| **Date** |  | **Signature** |  |

**Objective**

The objective of lab is Making HTTP POST requests using the http package. Fetching and displaying data from a REST API.

**Instructions**

You have to perform the following tasks yourselves. Raise your hand if you face any difficulty in understanding and solving these tasks. **Plagiarism** is an abhorrent practice and you should not engage in it.

**Flutter - Make an HTTP POST Request:**

In app development, the ability to fetch data from external sources, such as REST APIs, is a fundamental requirement. In Flutter, whether you need to send some data to the RESTful API, access a database, or then Send content from the Flutter App, Flutter provides you with the tools and packages(HTTP) to do this kind of API calls easily. Here we will explore how to interact with external data sources, Send JSON data(Key-Value pairs), and integrate it into your Flutter application.

**About the API:**

This is a API link of HTTP POST Request

POST - https://jsonplaceholder.typicode.com/posts

**Step By Step Implementations**

**Step 1: Create a New Project in VS code**

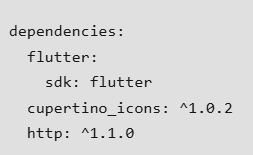
Create a new project in VS code name as per your choice.

**Step 2: Add the dependency to our project**

In the pubspec.yaml file we have to add the below dependency for the http package which provide various API call methods. Run these command in terminal.

**flutter pub add http**

**flutter pub add**



### **Step 3: Import the Package**

First of all import material.dart file, http,dart file and dart convert to encode text into JSON.

import 'dart:convert';

import 'package:flutter/material.dart';

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

**Step 4: Execute the main Method**

Here the execution of our app starts.

void main() {

  runApp(MyApp());

}

**Step 5: Create MyApp Class**

In this class we are going to implement the MaterialApp , here we are also set the Theme of our App.

class MyApp extends StatelessWidget {

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      debugShowCheckedModeBanner: false,

      theme: ThemeData(

        primarySwatch: Colors.green, // Set the app's primary theme color

      ),

      home: APICall(),

    );

  }

}

**Step 6: Create Class for APICall**

Now we are create class for APICall which is extend with stateless widget

class APICall extends StatefulWidget {

  const APICall({super.key});

  @override

  State<APICall> createState() => \_APICallState();

}

class \_APICallState extends State<APICall> {

  @override

  Widget build(BuildContext context) {

    return Scaffold();

  }

}

**Step 6: Build a basic UI**

Now we creating basic UI for our app. This UI consist on 2 TextField to take user input, 1 button to send data to API and text to show result of API call

class \_APICallState extends State<APICall> {

  TextEditingController nameController = TextEditingController();

  TextEditingController emailController = TextEditingController();

  String result = ''; // To store the result from the API call

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text('POST Request Example'),

      ),

      body: Padding(

        padding: EdgeInsets.all(16.0),

        child: Column(

          mainAxisAlignment: MainAxisAlignment.center,

          children: [

            TextField(

              controller: nameController,

              decoration: InputDecoration(labelText: 'Name'),

            ),

            TextField(

              controller: emailController,

              decoration: InputDecoration(labelText: 'Email'),

            ),

            ElevatedButton(

              onPressed: () {},

              child: Text('Submit'),

            ),

            SizedBox(height: 20.0),

            Text(

              result,

              style: TextStyle(fontSize: 16.0),

            ),

          ],

        ),

      ),

    );

  }

}

**Step 7: Clear TextField**

To clear TextField we called dispose method of stateful widget in our APICall class.

  @override

  void dispose() {

    nameController.dispose();

    emailController.dispose();

    super.dispose();

  }

**Step 8: Create a method of postData:**

Now we creating a method for post data. First we create variable of URL to store APIUrl

final String apiUrl = 'https://jsonplaceholder.typicode.com/posts';

Now write code for post data method

  Future<void> \_postData() async {

    try {

      final response = await http.post(

        Uri.parse(apiUrl),

        headers: <String, String>{

          'Content-Type': 'application/json; charset=UTF-8',

        },

        body: jsonEncode(<String, dynamic>{

          'name': nameController.text,

          'email': emailController.text,

          // Add any other data you want to send in the body

        }),

      );

**Output:**

      if (response.statusCode == 201) {

        // Successful POST request, handle the response here

        final responseData = jsonDecode(response.body);

        setState(() {

          result = 'ID: ${responseData['id']}\nName: ${responseData['name']}\nEmail: ${responseData['email']}';

        });

      } else {

        // If the server returns an error response, throw an exception

        throw Exception('Failed to post data');

      }

    } catch (e) {

      setState(() {

        result = 'Error: $e';

      });

    }

  }

**Step 9: Call the method postData:**

Now we call postData method in elevated button widget.

 ElevatedButton(

              onPressed: \_postData,

              child: Text('Submit'),

            ),

**Assessment:**

**Task: Submit a Contact Message**

**Objective:**

Create a Flutter app where the user submits a basic contact message using a POST request.

**Fields:**

* Name
* Message

**Instructions:**

1. Create a form with two TextField widgets for name and message.
2. On button press ("Send"), send the data to:

https://jsonplaceholder.typicode.com/posts

1. Use http.post() and send a JSON object:

{

"name": "Ali",

"message": "Please contact me about your services."

}

1. On success (status code 201), show:

Message sent successfully!

Name: Ali

Message: Please contact me about your services.

ID: 1