National University of Modern Languages, Islamabad

LAB 4

Submitted To:	**Sir Shahid**
Submitted By:	**Tayyaba Ejaz**
Roll Number:	**2751**
Subject:	**MAD**
Due Date:	**20th October 2025**
Semester:	**8th**
Course:	**BSCS, Morning**

Faculty of Engineering and Computer Science

Table of contents

1. Lab Task Part 1

POST and GET API using HTTP

1.1 Description

In this task, a Flutter application was developed to demonstrate API integration using the http package. The app allows users to send and fetch data from the public testing API JSONPlaceholder.

- The app includes a feedback form where users can enter a message.
- When the user taps “Send Data”, the app performs a POST request to send the feedback to the API.
- When the user taps “Fetch Data”, the app performs a GET request to retrieve and display the first five posts from the API.
- The app includes real-time form validation, loading indicators, and SnackBar messages to display success or error feedback.

This project demonstrates the use of HTTP communication, JSON encoding/decoding, and basic state management in Flutter.

1.2 Code

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

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

import 'dart:convert';


void main() {

 runApp(const APIs());

}


class APIs extends StatelessWidget {

 const APIs({super.key});


 @override

 Widget build(BuildContext context) {
```

```
    return MaterialApp(
      home: const FeedbackForm(),
    );
  }
}

class FeedbackForm extends StatefulWidget {
  const FeedbackForm({super.key});

  @override
  State<FeedbackForm> createState() => _FeedbackFormState();
}

class _FeedbackFormState extends State<FeedbackForm> {
  final TextEditingController _messageController = TextEditingController();
  final _formKey = GlobalKey<FormState>();
  bool _isSending = false;
  bool _isFetching = false;
  List<Map<String, dynamic>> _fetchedData = [];

  final RegExp _inputRegEx = RegExp(r'^(?=.*\S).+$');

  // POST request
  Future<void> sendData(String message) async {
    final url = Uri.parse('https://jsonplaceholder.typicode.com/posts');
    setState(() {
      _isSending = true;
    });
```

```
  try {
    final response = await http.post(
      url,
      headers: {"Content-Type": "application/json"},
      body: jsonEncode({"title": message, "body": message, "userId": 1}),
    );

    final isOk = response.statusCode == 201;
    final msg = isOk ? "Data Sent!" : "Failed to send data";

    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text(msg)),
    );
  } catch (e) {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text("POST Exception: $e")),
    );
  } finally {
    setState(() {
      _isSending = false;
    });
  }
}

// GET request
Future<void> fetchData() async {
  final url = Uri.parse('https://jsonplaceholder.typicode.com/posts');
```

```
  setState(() {
    _isFetching = true;
  });

  try {
    final response = await http.get(url);
    if (response.statusCode == 200) {
      final List data = jsonDecode(response.body);
      setState(() {
        _fetchedData = List<Map<String, dynamic>>.from(data.take(5));
      });
    } else {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text("GET Error: ${response.statusCode}")),
      );
    }
  } catch (e) {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text("GET Exception: $e")),
    );
  } finally {
    setState(() {
      _isFetching = false;
    });
  }
}

@override
```

```
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(title: const Text("POST & GET using HTTP")),
    body: Padding(
      padding: const EdgeInsets.all(16.0),
      child: Column(
        children: [
          // Input field with live validation
          Form(
            key: _formKey,
            child: TextFormField(
              controller: _messageController,
              decoration: const InputDecoration(
                labelText: "Enter your feedback",
              ),
              autovalidateMode: AutovalidateMode.onUserInteraction,
              validator: (value) {
                if (value == null || !_inputRegEx.hasMatch(value)) {
                  return "Feedback cannot be empty";
                }
                return null;
              },
            ),
          ),
          const SizedBox(height: 20),
          // Buttons
          Row(
            mainAxisAlignment: MainAxisAlignment.spaceEvenly,
```

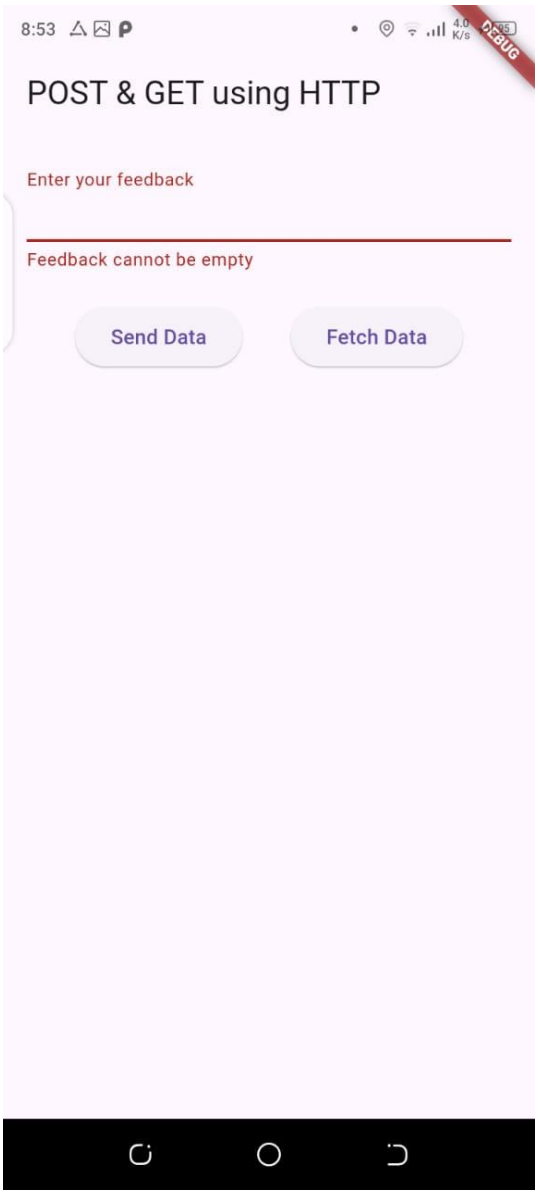
```
children: [
 ElevatedButton(
  onPressed: _isSending
     ? null
     : () {
   if (_formKey.currentState!.validate()) {
    sendData(_messageController.text);
   }
  },
  child: _isSending
     ? const SizedBox(
   height: 16,
   width: 16,
   child: CircularProgressIndicator(
      color: Colors.white, strokeWidth: 2),
  )
     : const Text("Send Data"),
 ),
 ElevatedButton(
  onPressed: _isFetching ? null : fetchData,
  child: _isFetching
     ? const SizedBox(
   height: 16,
   width: 16,
   child: CircularProgressIndicator(
      color: Colors.white, strokeWidth: 2),
  )
     : const Text("Fetch Data"),
```

```
    ),
  ],
),
const SizedBox(height: 20),
// Display fetched data in simple containers
if (_fetchedData.isNotEmpty)
  Expanded(
    child: ListView.builder(
      itemCount: _fetchedData.length,
      itemBuilder: (context, index) {
        final item = _fetchedData[index];
        return Container(
          margin: const EdgeInsets.symmetric(vertical: 6),
          padding: const EdgeInsets.all(12),
          decoration: BoxDecoration(
            border: Border.all(color: Colors.grey),
            borderRadius: BorderRadius.circular(8),
            color: Colors.grey[200],
          ),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              Text(
                item['title'],
                style: const TextStyle(
                    fontWeight: FontWeight.bold, fontSize: 16),
              ),
              const SizedBox(height: 4),
```

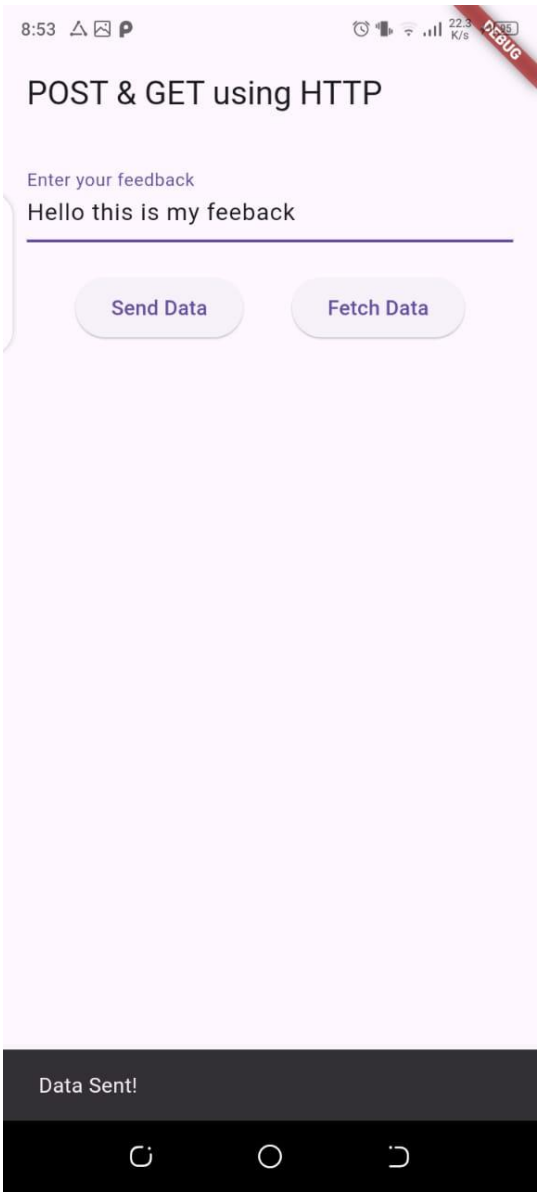
```
                    Text(item['body']),
                  ],
                ),
              );
            },
          ),
        ),
      ],
    ),
  ),
 );
}

@override
void dispose() {
 _messageController.dispose();
 super.dispose();
}
}.
```

1.3 Screenshots

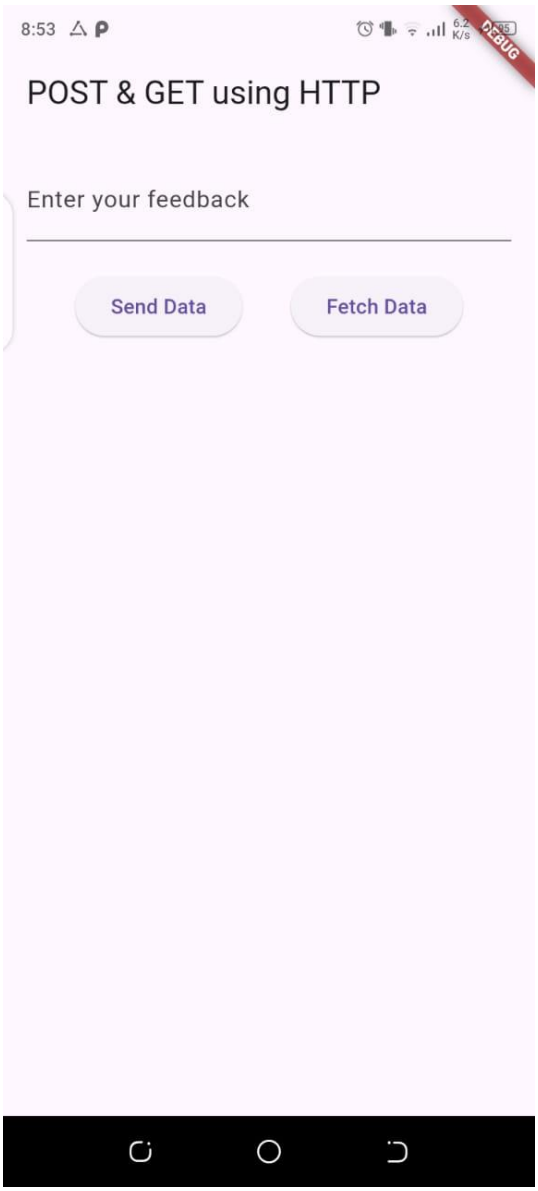


Input field validation



POST Request is sending the data

GET Request is fetching the data



Complete Screenshot of app

2. Lab Task Part 2

POST and GET API using DIO

2.1 Description

In this task, the same API functionality was implemented using the dio package, which provides advanced HTTP features compared to the basic http package.

- The app connects to JSONPlaceholder to perform both POST and GET requests.
- The “Send Data” button sends user feedback via a POST request using Dio, while the “Fetch Data” button retrieves the first five posts via a GET request.
- The app uses Dio’s built-in error handling, request options, and asynchronous state updates.
- It also features form validation, loading progress indicators, and SnackBar notifications for user interaction feedback.

This version highlights the benefits of Dio for HTTP operations, including simpler request configuration, better error management, and improved performance handling.

2.2 Code

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

import 'package:dio/dio.dart';


void main() {

 runApp(const APIsUsingDIO());

}


class APIsUsingDIO extends StatelessWidget {

 const APIsUsingDIO({super.key});


 @override
```

```
  Widget build(BuildContext context) {
    return const MaterialApp(
      home: FeedbackForm(),
    );
  }
}

class FeedbackForm extends StatefulWidget {
  const FeedbackForm({super.key});

  @override
  State<FeedbackForm> createState() => _FeedbackFormState();
}

class _FeedbackFormState extends State<FeedbackForm> {
  final TextEditingController _messageController = TextEditingController();
  final _formKey = GlobalKey<FormState>();
  bool _isSending = false;
  bool _isFetching = false;
  List<Map<String, dynamic>> _fetchedData = [];

  final Dio _dio = Dio();
  final RegExp _inputRegEx = RegExp(r'^(?=.*\S).+$');

  // POST request using Dio
  Future<void> sendData(String message) async {
    setState(() {
```

```
  _isSending = true;
});

try {
  final response = await _dio.post(
    'https://jsonplaceholder.typicode.com/posts',
    data: {
      "title": message,
      "body": message,
      "userId": 1,
    },
    options: Options(
      headers: {"Content-Type": "application/json"},
    ),
  );

  final isOk = response.statusCode == 201;
  final msg = isOk ? "✅ Data Sent Successfully!" : "❌ Failed to send data";

  ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(content: Text(msg)),
  );
} on DioError catch (e) {
  ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(content: Text("POST Error: ${e.message}")),
  );
} finally {
```

```
    setState(() {
      _isSending = false;
    });
  }
}

// GET request using Dio
Future<void> fetchData() async {
  setState(() {
    _isFetching = true;
  });

  try {
    final response =
    await _dio.get('https://jsonplaceholder.typicode.com/posts');

    if (response.statusCode == 200 && response.data is List) {
      final List<dynamic> data = response.data;
      setState(() {
        _fetchedData = data
            .take(5)
            .map((item) => Map<String, dynamic>.from(item))
            .toList();
      });
    } else {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text("GET Error: Unexpected data format")),
```

```
      );
    }
  } on DioError catch (e) {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text("GET Error: ${e.message}")),
    );
  } finally {
    setState(() {
      _isFetching = false;
    });
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(title: const Text("POST & GET using Dio")),
    body: Padding(
      padding: const EdgeInsets.all(16.0),
      child: Column(
        children: [
          // Input field with live validation
          Form(
            key: _formKey,
            child: TextFormField(
              controller: _messageController,
              decoration: const InputDecoration(
```

```
      labelText: "Enter your feedback",
    ),
    autovalidateMode: AutovalidateMode.onUserInteraction,
    validator: (value) {
      if (value == null || !_inputRegEx.hasMatch(value)) {
        return "Feedback cannot be empty";
      }
      return null;
    },
  ),
),
const SizedBox(height: 20),
// Buttons
Row(
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  children: [
    ElevatedButton(
      onPressed: _isSending
          ? null
          : () {
        if (_formKey.currentState!.validate()) {
          sendData(_messageController.text);
        }
      },
      child: _isSending
          ? const SizedBox(
        height: 16,
```

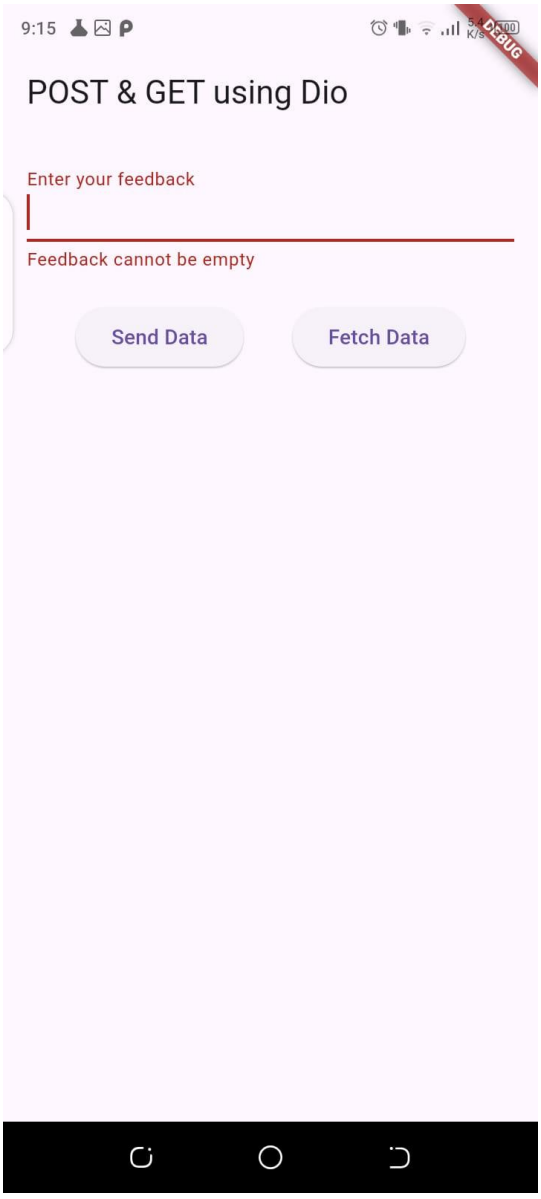
```
        width: 16,
        child: CircularProgressIndicator(
            color: Colors.white, strokeWidth: 2),
      )
          : const Text("Send Data"),
    ),
    ElevatedButton(
      onPressed: _isFetching ? null : fetchData,
      child: _isFetching
          ? const SizedBox(
        height: 16,
        width: 16,
        child: CircularProgressIndicator(
            color: Colors.white, strokeWidth: 2),
      )
          : const Text("Fetch Data"),
    ),
  ],
),
const SizedBox(height: 20),
// Display fetched data
if (_fetchedData.isNotEmpty)
  Expanded(
    child: ListView.builder(
      itemCount: _fetchedData.length,
      itemBuilder: (context, index) {
        final item = _fetchedData[index];
```

```
            return Container(
              margin: const EdgeInsets.symmetric(vertical: 6),
              padding: const EdgeInsets.all(12),
              decoration: BoxDecoration(
                border: Border.all(color: Colors.grey),
                borderRadius: BorderRadius.circular(8),
                color: Colors.grey[200],
              ),
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  Text(
                    item['title'] ?? "No Title",
                    style: const TextStyle(
                        fontWeight: FontWeight.bold, fontSize: 16),
                  ),
                  const SizedBox(height: 4),
                  Text(item['body'] ?? "No Content"),
                ],
              ),
            );
          },
        ),
      ),
    ],
  ),
),
```

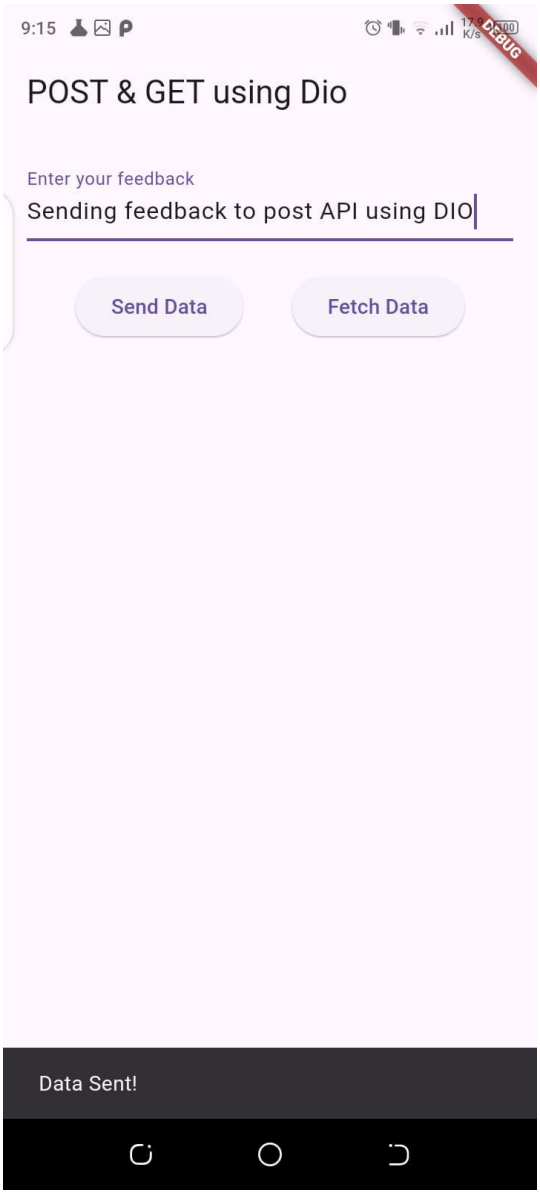
```
    );
  }

  @override
  void dispose() {
    _messageController.dispose();
    super.dispose();
  }
}
```

2.3 Screenshots

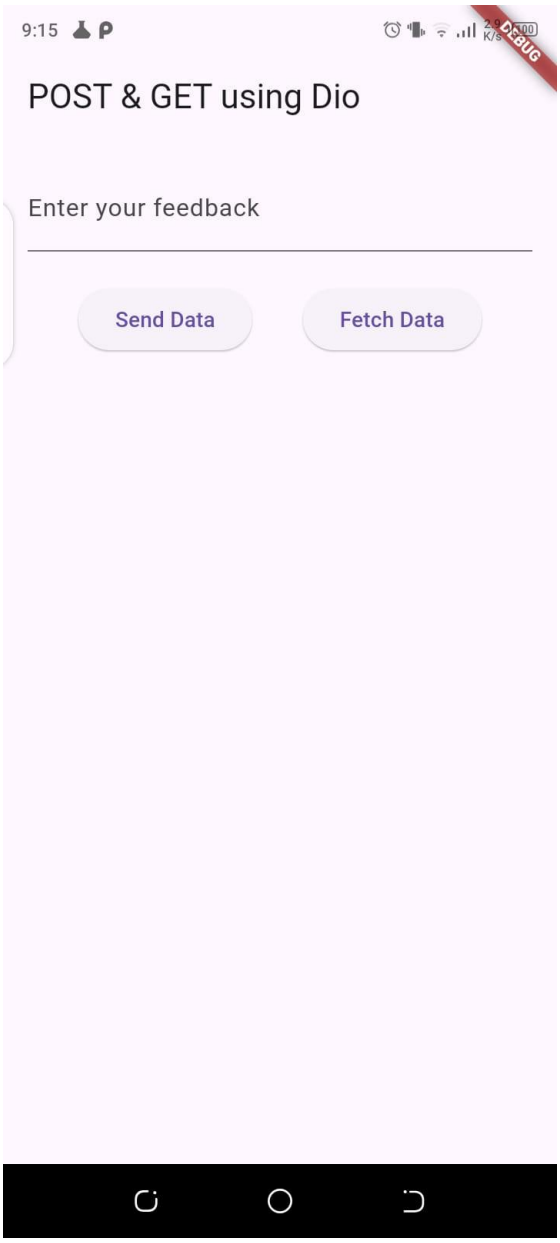


Input field validation



POST Request is sending the data

GET Request is fetching the data



Complete Screenshot of app