

**CET-222 Mobile Application Development**

**LAB # 20**

**LAB Title**

|  |
| --- |
| Implementing search functionality to filter API results. Fetch Data from an API. Displaying the fetched data in lists. |

**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 implementing search functionality to filter API results. Fetch Data from an API. Displaying the fetched data in lists.

**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.

**How to Fetch Data from an API and Display It in ListView in Flutter?**

Any modern app nowadays has a basic requirement of fetching data from some API. In Flutter, it is quite easy to do so. The basic idea to implement fetching and displaying data from a REST API goes like this:

1. Write a method to fetch data from the API
2. Calling that method gives us an API response (JSON)
3. Create a model class for the JSON response (we will need this to parse our JSON response to a Dart object)
4. Create a ListView builder wrapped inside a FutureBuilder (we will discuss this pattern further).
5. Pass Dart object details into the ListView builder

**Get Dependencies**

Start by adding the http dependency in your pubspec.yaml. We will need this to make the GET call.

dependencies:

http: ^0.13.3

Then import the following into your project.

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

import 'dart:convert';

**Our API for JSON Response**

For this example, we will be using JSONPlaceholder.

**API:**  
https://jsonplaceholder.typicode.com/albums/1/photos

The API response from this is a list of photos. Each entry contains albumId, id, title, url, and thumbnailUrl.

**Model Class**

We need a model class to convert this JSON response to a Dart object. Create a new Dart file, e.g., post.dart.

class Post {

final int albumId;

final int id;

final String title;

final String url;

final String thumbnailUrl;

Post({

required this.albumId,

required this.id,

required this.title,

required this.url,

required this.thumbnailUrl,

});

factory Post.fromJson(Map<String, dynamic> json) {

return Post(

albumId: json['albumId'],

id: json['id'],

title: json['title'],

url: json['url'],

thumbnailUrl: json['thumbnailUrl'],

);

}

}

In the model class, we mention all the fields (albumId, id, title, url, thumbnailUrl). The Post.fromJson method maps the JSON keys into our model class’s fields.

**Function to Call the API**

Now, let's write a function getPosts() to fetch data from our API.

Future<List<Post>> getPosts() async {

    final response = await http.get(

      Uri.parse('https://jsonplaceholder.typicode.com/albums/1/photos'),

    );

    if (response.statusCode == 200) {

      final List<dynamic> jsonResponse = json.decode(response.body);

      return jsonResponse.map((post) => Post.fromJson(post)).toList();

    } else {

      throw Exception('Failed to load posts');

    }

  }

This function returns Future<List<Post>>. Future in Dart represents data that will be available asynchronously. So, this function will return, in the future, a list where each element is of type Post.

Let's save this list of posts into a variable:

  late Future<List<Post>> postsFuture;

  @override

  void initState() {

    super.initState();

    postsFuture = getPosts();

  }

**FutureBuilder**

Now that we have a Future list of data (posts), let's display it on the screen. For this, we will use FutureBuilder and inside it, a ListView.builder.

FutureBuilder builds widgets based on asynchronous data. It requires two parameters:

* future (the Future we already have from our getPosts() function)
* builder (where we handle the API data).

Here is a basic structure of FutureBuilder:

FutureBuilder<List<Post>>(

            future: postsFuture,

            builder: (context, snapshot) {

              if (snapshot.connectionState == ConnectionState.waiting) {

                return CircularProgressIndicator();

              }

            else if (snapshot.hasError) {

                return Text('Error: ${snapshot.error}');

              }

            else if (snapshot.hasData) {

                return buildPosts(snapshot.data!);

              }

            else {

                return Text('No data available');

              }

            },

          )

**Function to Actually Display the Data**

If the snapshot has data, we call buildPosts(). Here is the buildPosts function:

 Widget buildPosts(List<Post> posts) {

  return ListView.builder(

    itemCount: posts.length,

    itemBuilder: (context, index) {

      final post = posts[index];

      return Container(

        color: Colors.grey.shade300,

        margin: EdgeInsets.symmetric(vertical: 5, horizontal: 10),

        padding: EdgeInsets.symmetric(vertical: 5, horizontal: 5),

        height: 100,

        width: double.maxFinite,

        child: Row(

          children: [

            Expanded(

              flex: 1,

              child: Image.network(

                post.thumbnailUrl!,

                fit: BoxFit.cover,

                errorBuilder: (context, error, stackTrace) =>

                    Icon(Icons.broken\_image, size: 50),

              ),

            ),

            SizedBox(width: 10),

            Expanded(flex: 3, child: Text(post.title!)),

          ],

        ),

      );

    },

  );

}



This function takes the list of Post objects and displays them in a ListView, showing the thumbnail, title, and album ID for each post.

Complete Code:

import 'dart:convert';

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

import 'package:flutter/material.dart';

void main() {

  runApp(const MyApp());

}

class MyApp extends StatelessWidget {

  const MyApp({super.key});

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      home: const MyHomePage(),

      debugShowCheckedModeBanner: false,

    );

  }

}

class MyHomePage extends StatefulWidget {

  const MyHomePage({super.key});

  @override

  State<MyHomePage> createState() => \_MyHomePageState();

}

class \_MyHomePageState extends State<MyHomePage> {

  late Future<List<Post>> postsFuture;

  @override

  void initState() {

    super.initState();

    postsFuture = getPosts();

  }

  Future<List<Post>> getPosts() async {

    final response = await http.get(

      Uri.parse('https://jsonplaceholder.typicode.com/albums/1/photos'),

    );

    if (response.statusCode == 200) {

      final List<dynamic> jsonResponse = json.decode(response.body);

      return jsonResponse.map((post) => Post.fromJson(post)).toList();

    } else {

      throw Exception('Failed to load posts');

    }

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text("Fetch and Display in List"),

        centerTitle: true,

        backgroundColor: Colors.grey,),

      body: Center(

        child: FutureBuilder<List<Post>>(

            future: postsFuture,

            builder: (context, snapshot) {

              if (snapshot.connectionState == ConnectionState.waiting) {

                return CircularProgressIndicator();

              }

            else if (snapshot.hasError) {

                return Text('Error: ${snapshot.error}');

              }

            else if (snapshot.hasData) {

                return buildPosts(snapshot.data!);

              }

            else {

                return Text('No data available');

              }

            },

          ),

      ),

    );

  }

 Widget buildPosts(List<Post> posts) {

  return ListView.builder(

    itemCount: posts.length,

    itemBuilder: (context, index) {

      final post = posts[index];

      return Container(

        color: Colors.grey.shade300,

        margin: EdgeInsets.symmetric(vertical: 5, horizontal: 10),

        padding: EdgeInsets.symmetric(vertical: 5, horizontal: 5),

        height: 100,

        width: double.maxFinite,

        child: Row(

          children: [

            Expanded(

              flex: 1,

              child: Image.network(

                post.thumbnailUrl,

                fit: BoxFit.cover,

                errorBuilder: (context, error, stackTrace) =>

                    Icon(Icons.broken\_image, size: 50),

              ),

            ),

            SizedBox(width: 10),

            Expanded(flex: 3, child: Text(post.title)),

          ],

        ),

      );

    },

  );

}

}

class Post {

  final int albumId;

  final int id;

  final String title;

  final String url;

  final String thumbnailUrl;

  Post({

    required this.albumId,

    required this.id,

    required this.title,

    required this.url,

    required this.thumbnailUrl,

  });

  factory Post.fromJson(Map<String, dynamic> json) {

    return Post(

      albumId: json['albumId'],

      id: json['id'],

      title: json['title'],

      url: json['url'],

      thumbnailUrl: json['thumbnailUrl'],

    );

  }

}

**Assessment:**

**Task: User List App Using APIubmit a Contact Message**

Create a Flutter application that fetches user data from the public API

<https://jsonplaceholder.typicode.com/users>

and displays it in a scrollable list using ListView.builder. The app should use the http package to make a GET request, parse the JSON response into a Dart model class, and show each user’s name and email in the list. Implement the data loading using FutureBuilder, and handle possible loading and error states appropriately.