

**SEB-312 Mobile Application Development**

**LAB # 11**

**LAB Title**

|  |
| --- |
| Implementing pagination to fetch data in batches. Adding infinite scrolling to a list. Open Ended Activity |

**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 pagination to fetch data in batches. Adding infinite scrolling to a list.

**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.

**Pagination implementation for infinite list**

In real-world Flutter applications, we often load dynamic data from servers or databases instead of using hardcode/dummy data as we often see in online examples.

For instance, you have an e-commerce app and need to load products from Rest API from the server side. If you only have 10 or 20 products, it will be easy to fetch them all with just one GET request. However, if your platform has thousands of products or even millions of products (e.g. a B2C platform), you have to implement pagination to reduce the burden on the server and reduce latency and improve the app’s performance as well.

In the beginning, we send a get request and load only a fixed number of results (choose a number that makes sense in your case) and display them in a ListView.

When the user scrolls to the bottom of the ListView, we send another GET request and fetch another fixed number of results, and so on.

To know when the user scrolls near the bottom of a ListView, we will create a ScrollController and attach it to the ListView. When ScrollController.position.extendAfter has a value less than a certain level (200 or 300 is good to go), we will call a function that sends a new GET request.

This app will load some dummy blog posts from a public Pest API endpoint provided by www.jsonplaceholder.typicode.com

When the app launches for the first time, we will fetch the first 20 posts. Next, every time we scroll near the bottom of the ListView, a function named \_loadMore will be called and this function will load 20 more posts.

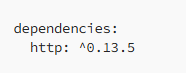
After all the posts from the API have been loaded, a message will appear letting us know that even if we scroll down, nothing will happen. Progress indicators will also appear while we are sending requests to the server.

**Get dependencies**

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

**flutter pub add http**

**flutter pub get**



**Implementation**

URL: <https://jsonplaceholder.typicode.com/posts>

**Step 1: Implement the main app structure**

Create a main.dart file with the following content:

import 'package:flutter/material.dart';

import 'package:flutter/foundation.dart';

import 'dart:convert';

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

void main() {

    runApp(MyApp());

}

class MyApp extends StatelessWidget {

    const MyApp({super.key});

    @override

    Widget build(BuildContext context) {

        return MaterialApp(

            debugShowCheckedModeBanner: false,

            title: 'Infinite List Pagination',

            theme: ThemeData(

                primarySwatch: Colors.pink,

            ),

            home: const HomePage(),

        );

    }

}

**Step 3: Create the HomePage stateful widget**

First, set up the basic HomePage widget.

class HomePage extends StatefulWidget {

const HomePage({super.key});

@override

State<HomePage> createState() => \_HomePageState();

}

**Step 4: Define State Variables**

Inside \_HomePageState, declare the required variables:

* **API URL** (\_baseUrl)
* **Pagination controls** (\_page, \_limit, \_hasNextPage)
* **Loading states** (\_isFirstLoadRunning, \_isLoadMoreRunning)
* **Data storage** (\_posts)
* **Scroll controller** (\_controller)

class \_HomePageState extends State<HomePage> {

final \_baseUrl = 'https://jsonplaceholder.typicode.com/posts';

int \_page = 0;

final int \_limit = 20;

bool \_hasNextPage = true;

bool \_isFirstLoadRunning = false;

bool \_isLoadMoreRunning = false;

List \_posts = [];

late ScrollController \_controller;

}

**Step 5: Initialize and Clean Up**

* initState(): Load initial data and set up scroll listener.
* dispose(): Remove the scroll listener to prevent memory leaks.

@override

void initState() {

super.initState();

\_firstLoad(); *// Fetch initial data*

\_controller = ScrollController()..addListener(\_loadMore); *// Listen for scroll*

}

@override

void dispose() {

\_controller.removeListener(\_loadMore); *// Clean up*

super.dispose();

}

**Step 6: Fetch Initial Data (\_firstLoad)**

This function loads the first batch of posts.

void \_firstLoad() async {

setState(() {

\_isFirstLoadRunning = true; *// Show loading indicator*

});

try {

final res = await http.get(Uri.parse("$\_baseUrl?\_page=$\_page&\_limit=$\_limit"));

setState(() {

\_posts = json.decode(res.body); *// Store fetched data*

});

} catch (err) {

if (kDebugMode) {

print('Something went wrong');

}

}

setState(() {

\_isFirstLoadRunning = false; *// Hide loading indicator*

});

}

**Step 7: Load More Data on Scroll (\_loadMore)**

This function triggers when the user scrolls near the bottom.

void \_loadMore() async {

        if (\_hasNextPage == true &&

            \_isFirstLoadRunning == false &&

            \_isLoadMoreRunning == false &&

            \_controller.position.extentAfter < 300) {

            setState(() {

                \_isLoadMoreRunning = true;

            });

            \_page += 1;

            try {

                final res = await http.get(Uri.parse("$\_baseUrl?\_page=$\_page&\_limit=$\_limit"));

                final List fetchedPosts = json.decode(res.body);

                if (fetchedPosts.isNotEmpty) {

                    setState(() {

                        \_posts.addAll(fetchedPosts);

                    });

                } else {

                    setState(() {

                        \_hasNextPage = false;

                    });

                }

            } catch (err) {

                if (kDebugMode) {

                    print('Something went wrong!');

                }

            }

            setState(() {

                \_isLoadMoreRunning = false;

            });

        }

    }

**Step 6: Build the UI**

The UI consists of:

* **AppBar** (Title)
* **ListView** (Displays posts)
* **Loading indicators** (For initial load & pagination)
* **End-of-list message** (When no more data is available)

@override

    Widget build(BuildContext context) {

        return Scaffold(

            appBar: AppBar(

                title: const Text('Infinite List Pagination'),

            ),

            body: \_isFirstLoadRunning

                ? const Center(

                    child: CircularProgressIndicator(),

                )

                : Column(

                    children: [

                        Expanded(

                            child: ListView.builder(

                                controller: \_controller,

                                itemCount: \_posts.length,

                                itemBuilder: (\_, index) => Card(

                                    margin: const EdgeInsets.symmetric(

                                        vertical: 8, horizontal: 10),

                                    child: ListTile(

                                        title: Text(\_posts[index]['title']),

                                        subtitle: Text(\_posts[index]['body']),

                                    ),

                                ),

                            ),

                        ),

                        // Show loading indicator when loading more

                        if (\_isLoadMoreRunning)

                            const Padding(

                                padding: EdgeInsets.only(top: 10, bottom: 40),

                                child: Center(

                                    child: CircularProgressIndicator(),

                                ),

                            ),

                        // Show message when no more data

                        if (\_hasNextPage == false)

                            Container(

                                padding: const EdgeInsets.only(top: 30, bottom: 40),

                                color: Colors.amber,

                                child: const Center(

                                    child: Text('You have fetched all of the content'),

                                ),

                            ),

                    ],

                ),

        );

    }

**Complete Code:**

import 'package:flutter/material.dart';

import 'package:flutter/foundation.dart';

import 'dart:convert';

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

void main() {

    runApp(MyApp());

}

class MyApp extends StatelessWidget {

    const MyApp({super.key});

    @override

    Widget build(BuildContext context) {

        return MaterialApp(

            debugShowCheckedModeBanner: false,

            title: 'Infinite List Pagination',

            theme: ThemeData(

                primarySwatch: Colors.pink,

            ),

            home: const HomePage(),

        );

    }

}

class HomePage extends StatefulWidget {

    const HomePage({super.key});

    @override

    State<HomePage> createState() => \_HomePageState();

}

class \_HomePageState extends State<HomePage> {

    // API base URL

    final \_baseUrl = 'https://jsonplaceholder.typicode.com/posts';

    // Pagination variables

    int \_page = 0;

    final int \_limit = 20;

    bool \_hasNextPage = true;

    // Loading states

    bool \_isFirstLoadRunning = false;

    bool \_isLoadMoreRunning = false;

    // List to hold fetched posts

    List \_posts = [];

    // Scroll controller

    late ScrollController \_controller;

    @override

    void initState() {

        super.initState();

        \_firstLoad();

        \_controller = ScrollController()..addListener(\_loadMore);

    }

    @override

    void dispose() {

        \_controller.removeListener(\_loadMore);

        super.dispose();

    }

    // First load function

    void \_firstLoad() async {

        setState(() {

            \_isFirstLoadRunning = true;

        });

        try {

            final res = await http.get(Uri.parse("$\_baseUrl?\_page=$\_page&\_limit=$\_limit"));

            setState(() {

                \_posts = json.decode(res.body);

            });

        } catch (err) {

            if (kDebugMode) {

                print('Something went wrong');

            }

        }

        setState(() {

            \_isFirstLoadRunning = false;

        });

    }

    // Load more function

    void \_loadMore() async {

        if (\_hasNextPage == true &&

            \_isFirstLoadRunning == false &&

            \_isLoadMoreRunning == false &&

            \_controller.position.extentAfter < 300) {

            setState(() {

                \_isLoadMoreRunning = true;

            });

            \_page += 1;

            try {

                final res = await http.get(Uri.parse("$\_baseUrl?\_page=$\_page&\_limit=$\_limit"));

                final List fetchedPosts = json.decode(res.body);

                if (fetchedPosts.isNotEmpty) {

                    setState(() {

                        \_posts.addAll(fetchedPosts);

                    });

                } else {

                    setState(() {

                        \_hasNextPage = false;

                    });

                }

            } catch (err) {

                if (kDebugMode) {

                    print('Something went wrong!');

                }

            }

            setState(() {

                \_isLoadMoreRunning = false;

            });

        }

    }

    @override

    Widget build(BuildContext context) {

        return Scaffold(

            appBar: AppBar(

                title: const Text('Infinite List Pagination'),

            ),

            body: \_isFirstLoadRunning

                ? const Center(

                    child: CircularProgressIndicator(),

                )

                : Column(

                    children: [

                        Expanded(

                            child: ListView.builder(

                                controller: \_controller,

                                itemCount: \_posts.length,

                                itemBuilder: (\_, index) => Card(

                                    margin: const EdgeInsets.symmetric(

                                        vertical: 8, horizontal: 10),

                                    child: ListTile(

                                        title: Text(\_posts[index]['title']),

                                        subtitle: Text(\_posts[index]['body']),

                                    ),

                                ),

                            ),

                        ),

                        // Show loading indicator when loading more

                        if (\_isLoadMoreRunning)

                            const Padding(

                                padding: EdgeInsets.only(top: 10, bottom: 40),

                                child: Center(

                                    child: CircularProgressIndicator(),

                                ),

                            ),

                        // Show message when no more data

                        if (\_hasNextPage == false)

                            Container(

                                padding: const EdgeInsets.only(top: 30, bottom: 40),

                                color: Colors.amber,

                                child: const Center(

                                    child: Text('You have fetched all of the content'),

                                ),

                            ),

                    ],

                ),

        );

    }

}

**Open Ended Activity:**

Build an application that fetches a list of posts from the JSONPlaceholder API (https://jsonplaceholder.typicode.com/) and displays them in a ListView. The application should:

1. **Fetch posts:** Make an API request to https://jsonplaceholder.typicode.com/posts to retrieve a list of posts.
2. **Handle data:** Parse the JSON response and map it to a Post data model (you can define this model as a Dart class).
3. **Manage state:** Use a state management approach (Provider) to manage the loading state, the list of posts, and potential errors.
4. **Display posts:** Display the list of posts in a ListView, showing the title and body of each post.
5. **Handle errors:** Display an error message if the API request fails.

**Assessment:**

1. What is pagination in the context of a REST API?
2. How do you implement pagination in a Flutter app with a REST API?
3. What is the role of query parameters like page and limit in pagination?
4. How do you handle pagination data in a REST API response in Flutter?
5. How do you display paginated data in a Flutter ListView?