LAPORAN PRAKTIKUM
PEMROGRAMAN PERANGKAT BERGERAK

MODUL XIV
DATA STORAGE ‘API’

Disusun Oleh :

Muhammad Abdul Aziz / 2211104026

SE0601

Asisten Praktikum :

Muhammad Faza Zulian Gesit Al Barru

Aisyah Hasna Aulia

Dosen Pengampu :

Yudha Islami Sulistya, S.Kom., M.Cs.

PROGRAM STUDI S1 SOFTWARE ENGINEERING

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

GUIDED

File main.dart

```dart
import 'package:flutter/material.dart';
import 'package:prak_14/screens/home_screen.dart';

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(
      title: 'Flutter Demo',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: const HomeScreen(),
    );
  }
}
```

File screens/home_screen.dart

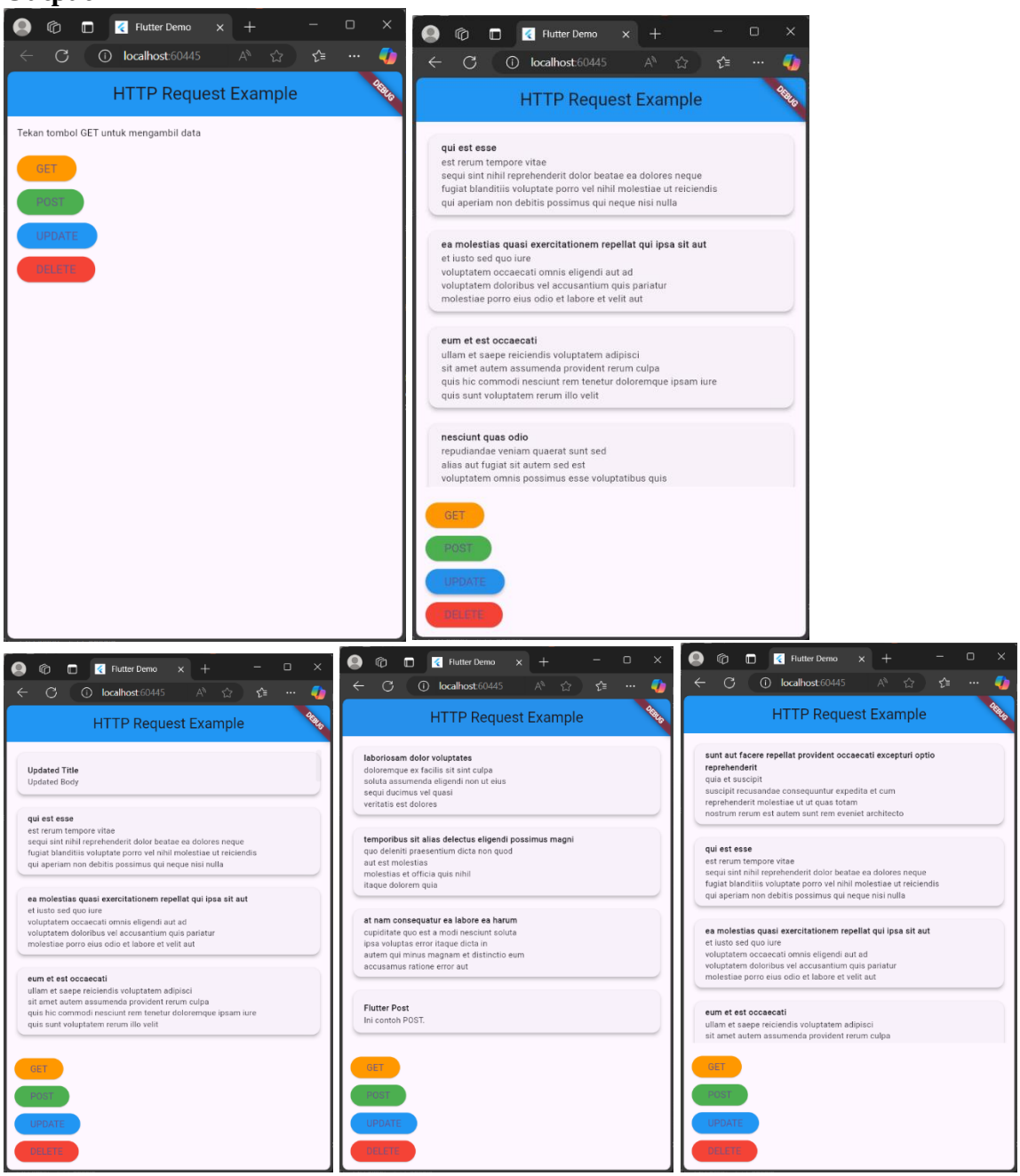
```dart
import 'package:flutter/material.dart';
import 'package:prak_14/services/api_service.dart';

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

  @override
  State<HomeScreen> createState() => _HomeScreenState();
}

class _HomeScreenState extends State<HomeScreen> {
  List<dynamic> _posts = []; // Menyimpan list posts
  bool _isLoading = false; // Untuk indikator loading
  final ApiService _apiService = ApiService(); // Instance ApiService
// Fungsi untuk menampilkan SnackBar
  void _showSnackBar(String message) {
    ScaffoldMessenger.of(context)
        .showSnackBar(SnackBar(content: Text(message)));
  }

// Fungsi untuk memanggil API dan menangani operasi
  Future<void> _handleApiOperation(
      Future<void> operation, String successMessage) async {
    setState(() {
      _isLoading = true;
    });
    try {
      await operation; // Menjalankan operasi API
      setState(() {
        _posts = _apiService.posts;
      });
      _showSnackBar(successMessage);
    } catch (e) {
      _showSnackBar('Error: $e');
    } finally {
      setState(() {
        _isLoading = false;
      });
    }
  }
```

```dart
@override
 Widget build(BuildContext context) {
   return Scaffold(
     appBar: AppBar(
       title: const Text('HTTP Request Example'),
       centerTitle: true,
       backgroundColor: Colors.blue,
     ),
     body: Padding(
       padding: const EdgeInsets.all(12.0),
       child: Column(
         crossAxisAlignment: CrossAxisAlignment.start,
         children: [
           _isLoading
               ? const Center(child: CircularProgressIndicator())
               : _posts.isEmpty
                   ? const Text(
                       "Tekan tombol GET untuk mengambil data",
                       style: TextStyle(fontSize: 12),
                     )
                   : Expanded(
                       child: ListView.builder(
                         itemCount: _posts.length,
                         itemBuilder: (context, index) {
                           return Padding(
                             padding: const EdgeInsets.only(bottom: 12.0),
                             child: Card(
                               elevation: 4,
                               child: ListTile(
                                 title: Text(
                                   _posts[index]['title'],
                                   style: const TextStyle(
                                       fontWeight: FontWeight.bold,
                                       fontSize: 12),
                                 ),
                                 subtitle: Text(
                                   _posts[index]['body'],
                                   style: const TextStyle(fontSize: 12),
                                 ),
                               ),
                             ),
                           );
                         },
                       ),
                     ),
           const SizedBox(height: 20),
           ElevatedButton(
             onPressed: () => _handleApiOperation(
                 _apiService.fetchPosts(), 'Data berhasil diambil!'),
             style: ElevatedButton.styleFrom(backgroundColor: Colors.orange),
             child: const Text('GET'),
           ),
           const SizedBox(height: 10),
           ElevatedButton(
             onPressed: () => _handleApiOperation(
                 _apiService.createPost(), 'Data berhasil ditambahkan!'),
             style: ElevatedButton.styleFrom(backgroundColor: Colors.green),
             child: const Text('POST'),
           ),
           const SizedBox(height: 10),
           ElevatedButton(
             onPressed: () => _handleApiOperation(
                 _apiService.updatePost(), 'Data berhasil diperbarui!'),
             style: ElevatedButton.styleFrom(backgroundColor: Colors.blue),
             child: const Text('UPDATE'),
           ),
           const SizedBox(height: 10),
           ElevatedButton(
             onPressed: () => _handleApiOperation(
                 _apiService.deletePost(), 'Data berhasil dihapus!'),
             style: ElevatedButton.styleFrom(backgroundColor: Colors.red),
             child: const Text('DELETE'),
           ),
         ],
       ),
     ),
   );
  }
}
```

File services/api_service.dart

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

class ApiService {
  final String baseUrl = "https://jsonplaceholder.typicode.com";
  List<dynamic> posts = []; // Menyimpan data post yang diterima
// Fungsi untuk GET data
  Future<void> fetchPosts() async {
    final response = await http.get(Uri.parse('$baseUrl/posts'));
    if (response.statusCode == 200) {
      posts = json.decode(response.body);
    } else {
      throw Exception('Failed to load posts');
    }
  }

  // Fungsi untuk POST data
Future<void> createPost() async {
  final response = await http.post(
    Uri.parse('$baseUrl/posts'),
    headers: {'Content-Type': 'application/json'},
    body: json.encode({
      'title': 'Flutter Post',
      'body': 'Ini contoh POST.',
      'userId': 1,
    }),
  );
  if (response.statusCode == 201) {
    posts.add({
      'title': 'Flutter Post',
      'body': 'Ini contoh POST.',
      'id': posts.length + 1,
    });
  } else {
    throw Exception('Failed to create post');
  }
}

// Fungsi untuk UPDATE data
Future<void> updatePost() async {
  final response = await http.put(
    Uri.parse('$baseUrl/posts/1'),
    body: json.encode({
      'title': 'Updated Title',
      'body': 'Updated Body',
      'userId': 1,
    }),
  );
  if (response.statusCode == 200) {
    final updatedPost = posts.firstWhere((post) => post['id'] == 1);
    updatedPost['title'] = 'Updated Title';
    updatedPost['body'] = 'Updated Body';
  } else {
    throw Exception('Failed to update post');
  }
}

// Fungsi untuk DELETE data
Future<void> deletePost() async {
  final response = await http.delete(
    Uri.parse('$baseUrl/posts/1'),
  );
  if (response.statusCode == 200) {
    posts.removeWhere((post) => post['id'] == 1);
  } else {
    throw Exception('Failed to delete post');
  }
}
}

```

Output



Penjelasan

Kode ini adalah implementasi HTTP Request di Flutter menggunakan REST API dari *JsonPlaceholder* dengan struktur terpisah antara logika dan tampilan. File utama main.dart menjalankan aplikasi dengan halaman utama HomeScreen. Kelas ApiService menangani logika HTTP Request yang terdiri dari empat operasi, yaitu fetchPosts() untuk mengambil data, createPost() untuk menambahkan data, updatePost() untuk memperbarui data, dan deletePost() untuk menghapus data. Data yang diperoleh dari API disimpan dalam variabel posts dan digunakan untuk ditampilkan di antarmuka pengguna. Pada HomeScreen, komponen UI dibangun menggunakan StatefulWidget, di mana status loading dikelola melalui variabel _isLoading dan data disimpan dalam _posts. Fungsi _handleApiOperation() digunakan untuk memanggil operasi API secara dinamis, menampilkan SnackBar sebagai notifikasi keberhasilan atau kegagalan operasi. Aplikasi ini memiliki empat tombol utama (GET, POST, UPDATE, DELETE) yang memicu fungsi terkait di ApiService dan menampilkan data dalam bentuk ListView berupa kartu dengan judul dan isi dari setiap post. Dengan pendekatan ini, logika pemrosesan data API dipisahkan dari tampilan, sehingga kode menjadi lebih rapi, modular, dan mudah dikelola.

UNGUIDED

File main.dart

```dart
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:prak_14/screens/home_screen.dart';

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

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

  @override
  Widget build(BuildContext context) {
    return GetMaterialApp(
      title: 'Flutter GetX Demo',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: const HomeScreen(),
    );
  }
}
```

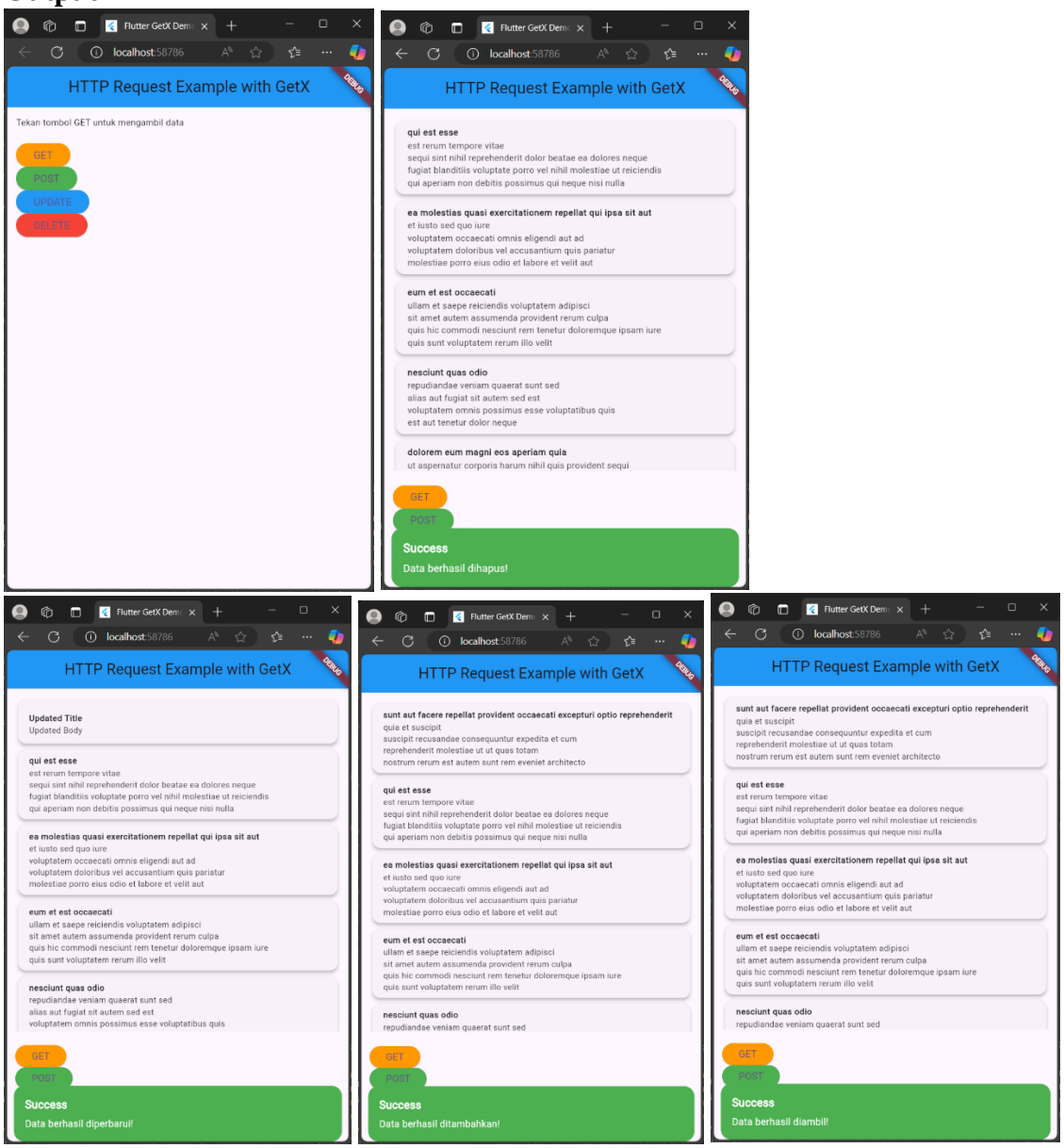
File screens/home_screen.dart

```dart
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:prak_14/controllers/post_controller.dart';

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

  @override
  Widget build(BuildContext context) {
    final ApiController controller = Get.put(ApiController());

    return Scaffold(
      appBar: AppBar(
        title: const Text('HTTP Request Example with GetX'),
        centerTitle: true,
        backgroundColor: Colors.blue,
      ),
      body: Padding(
        padding: const EdgeInsets.all(12.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Obx(() => controller.isLoading.value
                ? const Center(child: CircularProgressIndicator())
                : controller.posts.isEmpty
                    ? const Text(
                        "Tekan tombol GET untuk mengambil data",
                        style: TextStyle(fontSize: 12),
                      )
                    : Expanded(
                        child: ListView.builder(
                          itemCount: controller.posts.length,
                          itemBuilder: (context, index) {
                            return Card(
                              elevation: 4,
                              child: ListTile(
                                title: Text(
                                  controller.posts[index]['title'],
                                  style: const TextStyle(
                                      fontWeight: FontWeight.bold,
                                      fontSize: 12),
                                ),
                                subtitle: Text(
                                  controller.posts[index]['body'],
                                  style: const TextStyle(fontSize: 12),
                                ),
                              ),
                            );
                          },
                        ),
                      )),
```

File controllers/post_controller.dart

```dart
import 'dart:convert';
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:http/http.dart' as http;

class ApiController extends GetxController {
  final String baseUrl = "https://jsonplaceholder.typicode.com";

  var posts = <dynamic>[].obs;
  var isLoading = false.obs;

  // Snackbar helper
  void showSuccessSnackBar(String message) {
    Get.snackbar(
      'Success',
      message,
      backgroundColor: Colors.green,
      colorText: Colors.white,
      snackPosition: SnackPosition.BOTTOM,
      duration: const Duration(seconds: 2),
    );
  }

  void showErrorSnackBar(String message) {
    Get.snackbar(
      'Error',
      message,
      backgroundColor: Colors.red,
      colorText: Colors.white,
      snackPosition: SnackPosition.BOTTOM,
      duration: const Duration(seconds: 2),
    );
  }

  // GET Posts
  Future<void> fetchPosts() async {
    isLoading.value = true;
    try {
      final response = await http.get(Uri.parse('$baseUrl/posts'));
      if (response.statusCode == 200) {
        posts.value = json.decode(response.body);
        showSuccessSnackBar('Data berhasil diambil!');
      } else {
        throw Exception('Failed to load posts');
      }
    } catch (e) {
      showErrorSnackBar('Error: $e');
    } finally {
      isLoading.value = false;
    }
  }

  // POST Data
  Future<void> createPost() async {
    isLoading.value = true;
    try {
      final response = await http.post(
        Uri.parse('$baseUrl/posts'),
        headers: {'Content-Type': 'application/json'},
        body: json.encode({
          'title': 'Flutter Post',
          'body': 'Ini contoh POST.',
          'userId': 1,
        }),
      );
      if (response.statusCode == 201) {
        posts.add({
          'title': 'Flutter Post',
          'body': 'Ini contoh POST.',
          'id': posts.length + 1,
        });
        showSuccessSnackBar('Data berhasil ditambahkan!');
      } else {
        throw Exception('Failed to create post');
      }
    } catch (e) {
      showErrorSnackBar('Error: $e');
    } finally {
      isLoading.value = false;
    }
  }

  // UPDATE Data
  Future<void> updatePost() async {
    isLoading.value = true;
    try {
      final response = await http.put(
        Uri.parse('$baseUrl/posts/1'),
        body: json.encode({
          'title': 'Updated Title',
          'body': 'Updated Body',
          'userId': 1,
        }),
      );
      if (response.statusCode == 200) {
        var updatedPost = posts.firstWhere((post) => post['id'] == 1);
        updatedPost['title'] = 'Updated Title';
        updatedPost['body'] = 'Updated Body';
        showSuccessSnackBar('Data berhasil diperbarui!');
      } else {
        throw Exception('Failed to update post');
      }
    } catch (e) {
      showErrorSnackBar('Error: $e');
    } finally {
      isLoading.value = false;
    }
  }

  // DELETE Data
  Future<void> deletePost() async {
    isLoading.value = true;
    try {
      final response = await http.delete(Uri.parse('$baseUrl/posts/1'));
      if (response.statusCode == 200) {
        posts.removeWhere((post) => post['id'] == 1);
        showSuccessSnackBar('Data berhasil dihapus!');
      } else {
        throw Exception('Failed to delete post');
      }
    } catch (e) {
      showErrorSnackBar('Error: $e');
    } finally {
      isLoading.value = false;
    }
  }
}
```

Output



Penjelasan

Kode ini adalah implementasi HTTP Request di Flutter menggunakan REST API dari *JsonPlaceholder* dengan struktur terpisah antara logika dan tampilan. File utama main.dart menjalankan aplikasi dengan halaman utama HomeScreen. Kelas ApiService menangani logika HTTP Request yang terdiri dari empat operasi, yaitu fetchPosts() untuk mengambil data, createPost() untuk menambahkan data, updatePost() untuk memperbarui data, dan deletePost() untuk menghapus data. Data yang diperoleh dari API disimpan dalam variabel posts dan digunakan untuk ditampilkan di antarmuka pengguna. Pada HomeScreen, komponen UI dibangun menggunakan StatefulWidget, di mana status loading dikelola melalui variabel _isLoading dan data disimpan dalam _posts. Fungsi _handleApiOperation() digunakan untuk memanggil operasi API secara dinamis, menampilkan SnackBar sebagai notifikasi keberhasilan atau kegagalan operasi. Aplikasi ini memiliki empat tombol utama (GET, POST, UPDATE, DELETE) yang memicu fungsi terkait di ApiService dan menampilkan data dalam bentuk ListView berupa kartu dengan judul dan isi dari setiap post. Dengan pendekatan ini, logika pemrosesan data API dipisahkan dari tampilan, sehingga kode menjadi lebih rapi, modular, dan mudah dikelola.