LAPORAN PRAKTIKUM

PEMROGRAMAN PERANGKAT BERGERAK

MODUL 14

DATA STORAGE API

Disusun Oleh :

Aji Prasetyo Nugroho / 2211104049

S1SE-06-2

Asisten Praktikum :

Muhammad Faza Zulian Gesit Al Barru

Aisyah Hasna Aulia

Dosen Pengampu :

Yudha Islami Sulistya

PROGRAM STUDI S1 SOFTWARE ENGINEERING

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

A. GUIDED

- home_screen.dart

  Source code :

```dart
import 'package:flutter/material.dart';
import 'package:pertemuan_14/services/api_service.dart';

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

  @override
  State<HomepageScreen> createState() => _HomepageScreenState();
}

class _HomepageScreenState extends State<HomepageScreen> {
  List<dynamic> _posts = []; // Menyimpan list posts
  bool _isLoading = false; // Untuk indikator loading
  final ApiService _apiService = ApiService(); // Instance ApiService

  // Fungsi untuk menampilkan SnackBar
/*************  ✨ Codeium Command ⭐  *************/
  /// Membuat dan menampilkan SnackBar dengan pesan yang diberikan.
/******  12753b93-c031-4ec5-b0ac-88f2831b5a13  *******/ 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; // Mengupdate posts setelah operasi berhasil
      });
      _showSnackBar(successMessage); // Menampilkan SnackBar sukses
    } catch (e) {
      _showSnackBar('Error: $e'); // Menampilkan SnackBar error
    } finally {
      setState(() {
        _isLoading = false;
      });
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
```

```
    appBar: AppBar(
      title: const Text('REST API - Praktikum 14'),
    ),
    body: Padding(
      padding: const EdgeInsets.all(12),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          // Indikator loading
          if (_isLoading)
            const Center(child: CircularProgressIndicator())
          // Pesan jika data kosong
          else if (_posts.isEmpty)
            const Text(
              "Tekan tombol GET untuk mengambil data",
              style: TextStyle(fontSize: 14),
            )
          // Menampilkan daftar data
          else
            Expanded(
              child: ListView.builder(
                itemCount: _posts.length,
                itemBuilder: (context, index) {
                  final post = _posts[index];
                  return Padding(
                    padding: const EdgeInsets.only(bottom: 12.0),
                    child: Card(
                      elevation: 4,
                      child: ListTile(
                        title: Text(
                          post['title'] ?? 'No Title',
                          style: const TextStyle(
                              fontWeight: FontWeight.bold, fontSize: 14),
                        ),
                        subtitle: Text(
                          post['body'] ?? 'No Body',
                          style: const TextStyle(fontSize: 12),
                        ),
                        trailing: IconButton(
                          icon: const Icon(Icons.delete, color: Colors.red),
                          onPressed: () => _handleApiOperation(
                            _apiService.deletePost(post['id']),
                            'Data berhasil dihapus!',
                          ),
                        ),
                      ),
                    ),
                  );
                },
              ),
            ),
          // Tombol GET
          ElevatedButton(
```

```
              onPressed: () => _handleApiOperation(
                  _apiService.fetchPosts(), 'Data berhasil diambil!'),
              style: ElevatedButton.styleFrom(backgroundColor: Colors.orange),
              child: const Text('GET'),
            ),
            // Tombol POST
            ElevatedButton(
              onPressed: () => _handleApiOperation(
                  _apiService.createPost(), 'Data berhasil ditambahkan!'),
              style: ElevatedButton.styleFrom(backgroundColor: Colors.green),
              child: const Text('POST'),
            ),
            // Tombol UPDATE
            ElevatedButton(
              onPressed: () => _handleApiOperation(
                _apiService.updatePost(1, 'Updated Title', 'Updated Body'),
                'Data berhasil diperbarui!',
              ),
              style: ElevatedButton.styleFrom(backgroundColor: Colors.blue),
              child: const Text('UPDATE'),
            ),
          ],
        ),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: () => _handleApiOperation(
          _apiService.fetchPosts(), // Contoh pemanggilan API
          'Posts fetched successfully!',
        ),
        child: const Icon(Icons.refresh),
      ),
    );
  }
}
```

- api_service.dart

  Source Code :

```
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 {
    try {
      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. Status Code: ${response.statusCode}');
    }
  } catch (e) {
    throw Exception('Error fetching posts: $e');
  }
}

// Fungsi untuk POST data
Future<void> createPost() async {
  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) {
      final newPost =
          json.decode(response.body); // Mengambil respons baru dari server
      posts.add({
        'id': newPost['id'] ?? (posts.length + 1),
        'title': newPost['title'] ?? 'Flutter Post',
        'body': newPost['body'] ?? 'Ini contoh POST.',
      });
    } else {
      throw Exception(
          'Failed to create post. Status Code: ${response.statusCode}');
    }
  } catch (e) {
    throw Exception('Error creating post: $e');
  }
}

// Fungsi untuk UPDATE data
Future<void> updatePost(int postId, String newTitle, String newBody) async {
  try {
    final response = await http.put(
      Uri.parse('$baseUrl/posts/$postId'),
      headers: {'Content-Type': 'application/json'},
      body: json.encode({
        'title': newTitle,
        'body': newBody,
        'userId': 1,
      }),
    );

    if (response.statusCode == 200) {
      // Update post dalam list lokal
```

```
        final index = posts.indexWhere((post) => post['id'] == postId);
        if (index != -1) {
          posts[index]['title'] = newTitle;
          posts[index]['body'] = newBody;
        }
      } else {
        throw Exception(
            'Failed to update post. Status Code: ${response.statusCode}');
      }
    } catch (e) {
      throw Exception('Error updating post: $e');
    }
  }

  // Fungsi untuk DELETE data
  Future<void> deletePost(int postId) async {
    try {
      final response = await http.delete(Uri.parse('$baseUrl/posts/$postId'));
      if (response.statusCode == 200) {
        // Menghapus post dari list lokal
        posts.removeWhere((post) => post['id'] == postId);
      } else {
        throw Exception(
            'Failed to delete post. Status Code: ${response.statusCode}');
      }
    } catch (e) {
      throw Exception('Error deleting post: $e');
    }
  }
}
```

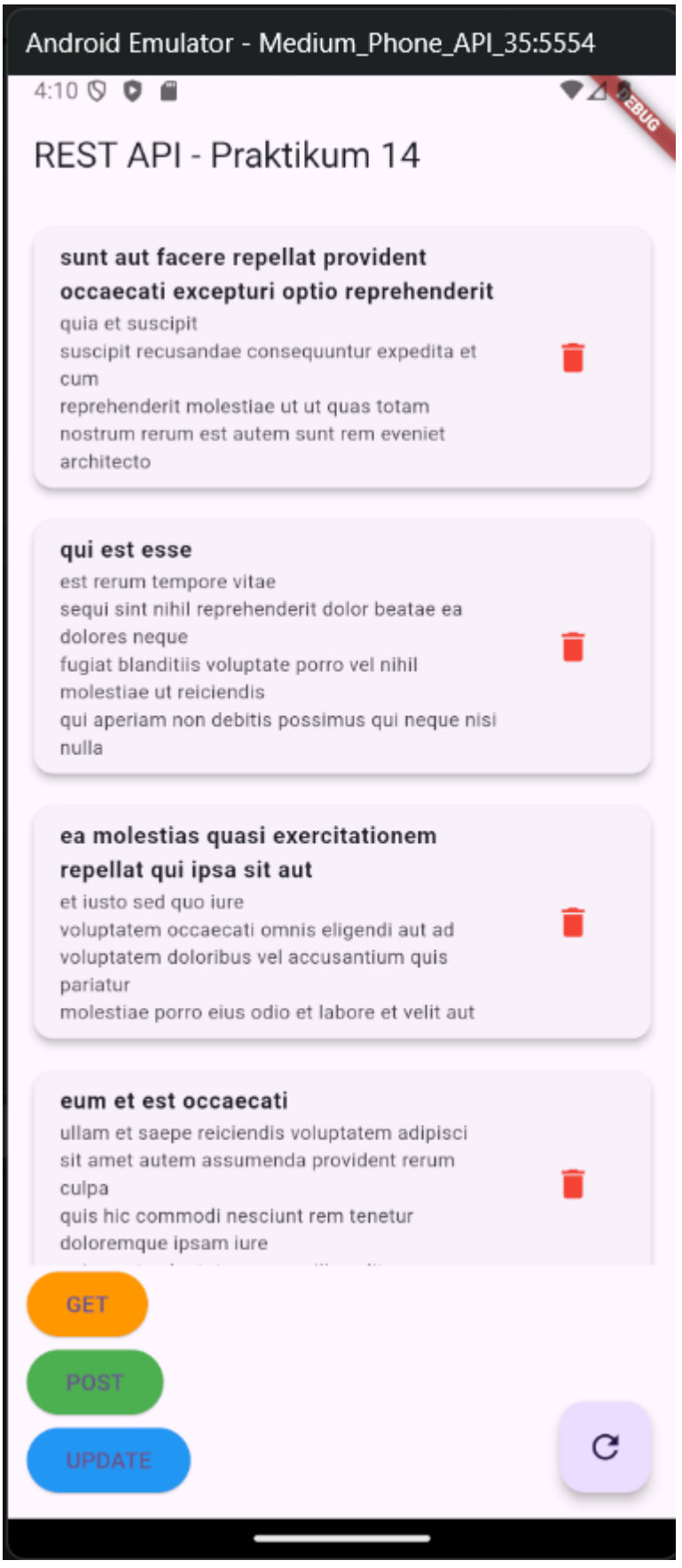
- main.dart

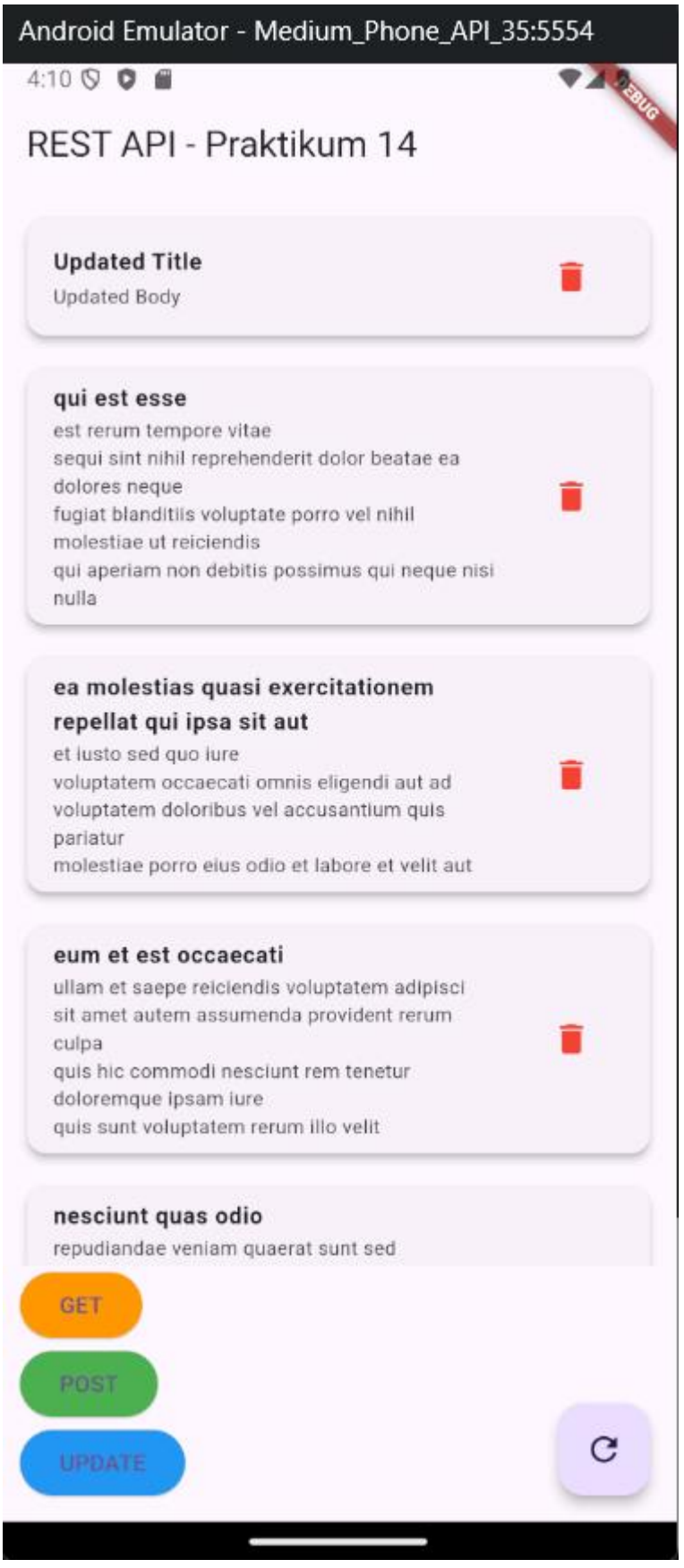
  Source Code :

```dart
import 'package:flutter/material.dart';
import 'package:pertemuan_14/screen/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: HomepageScreen(),
    );
  }
}
```

Output :


Android Emulator - Medium_Phone_API_35:5554
4:10
DEBUG
REST API - Praktikum 14
sunt aut facere repellat provident occaecati excepturi optio reprehenderit
quia et suscipit
suscipit recusandae consequuntur expedita et cum
reprehenderit molestiae ut ut quas totam nostrum rerum est autem sunt rem eveniet architecto
qui est esse
est rerum tempore vitae
sequi sint nihil reprehenderit dolor beatae ea dolores neque
fugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis
qui aperiam non debitis possimus qui neque nisi nulla
ea molestias quasi exercitationem repellat qui ipsa sit aut
et iusto sed quo iure
voluptatem occaecati omnis eligendi aut ad
voluptatem doloribus vel accusantium quis pariatur
molestiae porro eius odio et labore et velit aut
eum et est occaecati
ullam et saepe reiciendis voluptatem adipisci
sit amet autem assumenda provident rerum culpa
quis hic commodi nesciunt rem tenetur doloremque ipsam iure
GET
POST
UPDATE


Android Emulator - Medium_Phone_API_35:5554
4:10
DEBUG
REST API - Praktikum 14
sunt aut facere repellat provident occaecati excepturi optio reprehenderit
quia et suscipit
suscipit recusandae consequuntur expedita et cum
reprehenderit molestiae ut ut quas totam
nostrum rerum est autem sunt rem eveniet architecto
qui est esse
est rerum tempore vitae
sequi sint nihil reprehenderit dolor beatae ea dolores neque
fugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis
qui aperiam non debitis possimus qui neque nisi nulla
ea molestias quasi exercitationem repellat qui ipsa sit aut
et iusto sed quo iure
voluptatem occaecati omnis eligendi aut ad
voluptatem doloribus vel accusantium quis pariatur
molestiae porro eius odio et labore et velit aut
eum et est occaecati
ullam et saepe reiciendis voluptatem adipisci
sit amet autem assumenda provident rerum culpa
quis hic commodi nesciunt rem tenetur doloremque ipsam iure
GET
POST
UPDATE


Android Emulator - Medium_Phone_API_35:5554
4:10
DEBUG
REST API - Praktikum 14
Updated Title
Updated Body
qui est esse
est rerum tempore vitae
sequi sint nihil reprehenderit dolor beatae ea dolores neque
fugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis
qui aperiam non debitis possimus qui neque nisi nulla
ea molestias quasi exercitationem repellat qui ipsa sit aut
et iusto sed quo iure
voluptatem occaecati omnis eligendi aut ad
voluptatem doloribus vel accusantium quis pariatur
molestiae porro eius odio et labore et velit aut
eum et est occaecati
ullam et saepe reiciendis voluptatem adipisci
sit amet autem assumenda provident rerum culpa
quis hic commodi nesciunt rem tenetur doloremque ipsam iure
quis sunt voluptatem rerum illo velit
nesciunt quas odio
repudiandae veniam quaerat sunt sed
GET
POST
UPDATE

Deskripsi Program

Program ini adalah aplikasi berbasis Flutter yang menggunakan REST API untuk melakukan operasi data (GET, POST, PUT, dan DELETE) dengan memanfaatkan state management GetX. Aplikasi ini memuat data dari layanan API eksternal, menampilkan daftar data dalam format kartu yang informatif dan estetik, serta memberikan respon interaktif menggunakan Snackbar. Setiap operasi data seperti pengambilan, penambahan, pembaruan, atau penghapusan akan secara otomatis memperbarui tampilan daftar melalui mekanisme observasi yang disediakan oleh GetX, sehingga memudahkan pengguna melihat perubahan tanpa memuat ulang secara manual.

Antarmuka aplikasi dirancang agar mudah digunakan dengan elemen UI yang jelas dan tombol dengan ikon-ikon yang membantu. Operasi seperti mengambil data dari API atau memperbarui entri tertentu dilakukan dengan kontrol tombol yang terorganisir. Daftar data ditampilkan dalam kartu dengan desain modern, termasuk elemen visual seperti CircleAvatar untuk ID dan ikon hapus yang berwarna mencolok. Selain itu, fitur Snackbar memastikan bahwa setiap tindakan berhasil atau kesalahan akan diinformasikan secara real-time kepada pengguna, meningkatkan pengalaman pengguna secara keseluruhan. Program ini cocok sebagai contoh praktikum pengelolaan data API dengan integrasi state management yang efisien dan user-friendly.

B. Unguided

Modifikasi tampilan Guided dari praktikum di atas:

a. **Gunakan State Management dengan GetX:**

- Atur data menggunakan *state management* GetX agar lebih mudah dikelola.
- Implementasi GetX meliputi pembuatan controller untuk mengelola data dan penggunaan widget Obx untuk menampilkan data secara otomatis setiap kali ada perubahan.

b. **Tambahkan Snackbar untuk Memberikan Respon Berhasil:**

- Tampilkan snackbar setelah setiap operasi berhasil, seperti menambah atau memperbarui data.
- Gunakan Get.snackbar agar pesan sukses muncul di layar dan mudah dipahami oleh pengguna.

Source Code :

a. post_controller.dart

Source code :

```dart
import 'package:get/get.dart';
import 'package:pertemuan_14/services/api_service.dart';

class PostController extends GetxController {
  final ApiService _apiService = ApiService();
  var posts = <dynamic>[].obs; // Observable list untuk posts
  var isLoading = false.obs; // Observable untuk loading state

  // Fungsi untuk memuat data posts
  Future<void> fetchPosts() async {
    isLoading.value = true;
    try {
      await _apiService.fetchPosts();
      posts.assignAll(_apiService.posts);
      Get.snackbar('Success', 'Data berhasil diambil!');
    } catch (e) {
      Get.snackbar('Error', 'Gagal memuat data: $e');
    } finally {
      isLoading.value = false;
    }
  }

  // Fungsi untuk menambah data
  Future<void> createPost() async {
    isLoading.value = true;
    try {
      await _apiService.createPost();
      posts.assignAll(_apiService.posts);
      Get.snackbar('Success', 'Data berhasil ditambahkan!');
    } catch (e) {
      Get.snackbar('Error', 'Gagal menambah data: $e');
    } finally {
      isLoading.value = false;
```

```
    }
  }

  // Fungsi untuk memperbarui data
  Future<void> updatePost(int id, String title, String body) async {
    isLoading.value = true;
    try {
      await _apiService.updatePost(id, title, body);
      posts.assignAll(_apiService.posts);
      Get.snackbar('Success', 'Data berhasil diperbarui!');
    } catch (e) {
      Get.snackbar('Error', 'Gagal memperbarui data: $e');
    } finally {
      isLoading.value = false;
    }
  }

  // Fungsi untuk menghapus data
  Future<void> deletePost(int id) async {
    isLoading.value = true;
    try {
      await _apiService.deletePost(id);
      posts.assignAll(_apiService.posts);
      Get.snackbar('Success', 'Data berhasil dihapus!');
    } catch (e) {
      Get.snackbar('Error', 'Gagal menghapus data: $e');
    } finally {
      isLoading.value = false;
    }
  }
}
```

b. home_screen.dart

Source code :

```
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:pertemuan_14/controller/post_controller.dart';

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

  @override
  Widget build(BuildContext context) {
    final PostController postController = Get.put(PostController());

    return Scaffold(
      appBar: AppBar(
        title: const Text('REST API - Praktikum 14'),
      ),
      body: Obx(
        () => Padding(
          padding: const EdgeInsets.all(12),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
```

```
          children: [
            if (postController.isLoading.value)
              const Center(child: CircularProgressIndicator())
            else if (postController.posts.isEmpty)
              const Text(
                "Tekan tombol GET untuk mengambil data",
                style: TextStyle(fontSize: 14),
              )
            else
              Expanded(
                child: ListView.builder(
                  itemCount: postController.posts.length,
                  itemBuilder: (context, index) {
                    final post = postController.posts[index];
                    return Padding(
                      padding: const EdgeInsets.only(bottom: 12.0),
                      child: Card(
                        color: Colors.deepPurple.shade50,
                        shape: RoundedRectangleBorder(
                          borderRadius: BorderRadius.circular(12),
                        ),
                        elevation: 4,
                        child: ListTile(
                          leading: CircleAvatar(
                            backgroundColor: Colors.deepPurple,
                            child: Text(
                              post['id'].toString(),
                              style: const TextStyle(color: Colors.white),
                            ),
                          ),
                          title: Text(
                            post['title'] ?? 'No Title',
                            style: const TextStyle(
                                fontWeight: FontWeight.bold, fontSize: 14),
                          ),
                          subtitle: Text(
                            post['body'] ?? 'No Body',
                            style: const TextStyle(fontSize: 12),
                          ),
                          trailing: IconButton(
                            icon: const Icon(Icons.delete, color: Colors.red),
                            onPressed: () =>
                                postController.deletePost(post['id']),
                          ),
                        ),
                      ),
                    );
                  },
                ),
              ),
            const Divider(height: 20),
            Row(
              mainAxisAlignment: MainAxisAlignment.spaceAround,
```

```
              children: [
                ElevatedButton.icon(
                  onPressed: postController.fetchPosts,
                  icon: const Icon(Icons.download),
                  style: ElevatedButton.styleFrom(
                    backgroundColor: Colors.orange,
                    padding: const EdgeInsets.symmetric(
                        horizontal: 20, vertical: 12),
                  ),
                  label: const Text('GET'),
                ),
                ElevatedButton.icon(
                  onPressed: postController.createPost,
                  icon: const Icon(Icons.add),
                  style: ElevatedButton.styleFrom(
                    backgroundColor: Colors.green,
                    padding: const EdgeInsets.symmetric(
                        horizontal: 20, vertical: 12),
                  ),
                  label: const Text('POST'),
                ),
                ElevatedButton.icon(
                  onPressed: () => postController.updatePost(
                      1, 'Updated Title', 'Updated Body'),
                  icon: const Icon(Icons.edit),
                  style: ElevatedButton.styleFrom(
                    backgroundColor: Colors.blue,
                    padding: const EdgeInsets.symmetric(
                        horizontal: 20, vertical: 12),
                  ),
                  label: const Text('UPDATE'),
                ),
              ],
            )
          ],
        ),
      ),
    ),
    floatingActionButton: FloatingActionButton(
      onPressed: postController.fetchPosts,
      backgroundColor: Colors.deepPurple,
      child: const Icon(Icons.refresh, color: Colors.white),
    ),
  );
 }
}
```

c. api_service.dart

Source code :

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

class ApiService {
  final String baseUrl = "https://jsonplaceholder.typicode.com";
```

```dart
List<dynamic> posts = []; // Menyimpan data post yang diterima

// Fungsi untuk GET data
Future<void> fetchPosts() async {
  try {
    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. Status Code: ${response.statusCode}');
    }
  } catch (e) {
    throw Exception('Error fetching posts: $e');
  }
}

// Fungsi untuk POST data
Future<void> createPost() async {
  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) {
      final newPost =
          json.decode(response.body); // Mengambil respons baru dari server
      posts.add({
        'id': newPost['id'] ?? (posts.length + 1),
        'title': newPost['title'] ?? 'Flutter Post',
        'body': newPost['body'] ?? 'Ini contoh POST.',
      });
    } else {
      throw Exception(
          'Failed to create post. Status Code: ${response.statusCode}');
    }
  } catch (e) {
    throw Exception('Error creating post: $e');
  }
}

// Fungsi untuk UPDATE data
Future<void> updatePost(int postId, String newTitle, String newBody) async {
  try {
    final response = await http.put(
      Uri.parse('$baseUrl/posts/$postId'),
      headers: {'Content-Type': 'application/json'},
```

```
        body: json.encode({
          'title': newTitle,
          'body': newBody,
          'userId': 1,
        }),
      );

      if (response.statusCode == 200) {
        // Update post dalam list lokal
        final index = posts.indexWhere((post) => post['id'] == postId);
        if (index != -1) {
          posts[index]['title'] = newTitle;
          posts[index]['body'] = newBody;
        }
      } else {
        throw Exception(
            'Failed to update post. Status Code: ${response.statusCode}');
      }
    } catch (e) {
      throw Exception('Error updating post: $e');
    }
  }

  // Fungsi untuk DELETE data
  Future<void> deletePost(int postId) async {
    try {
      final response = await http.delete(Uri.parse('$baseUrl/posts/$postId'));
      if (response.statusCode == 200) {
        // Menghapus post dari list lokal
        posts.removeWhere((post) => post['id'] == postId);
      } else {
        throw Exception(
            'Failed to delete post. Status Code: ${response.statusCode}');
      }
    } catch (e) {
      throw Exception('Error deleting post: $e');
    }
  }
}
```

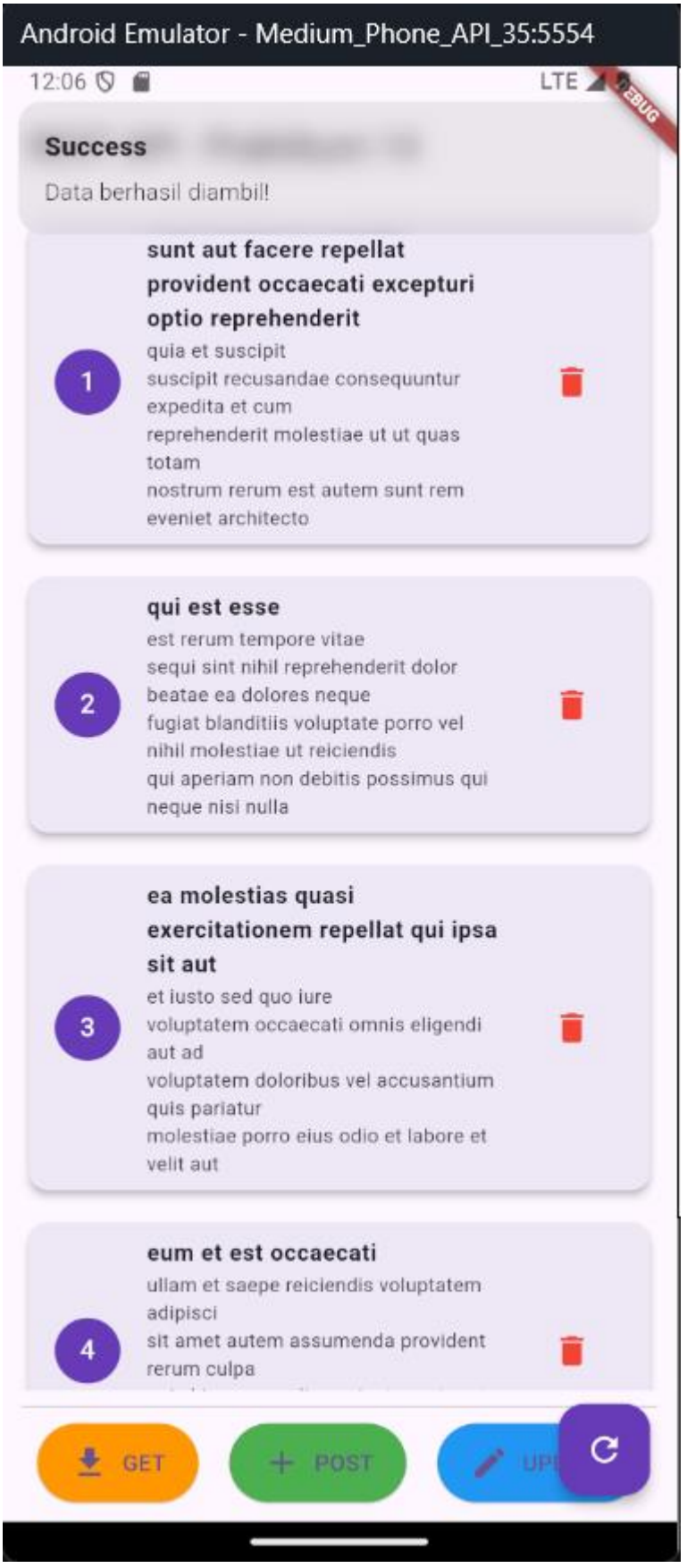
d. main.dart

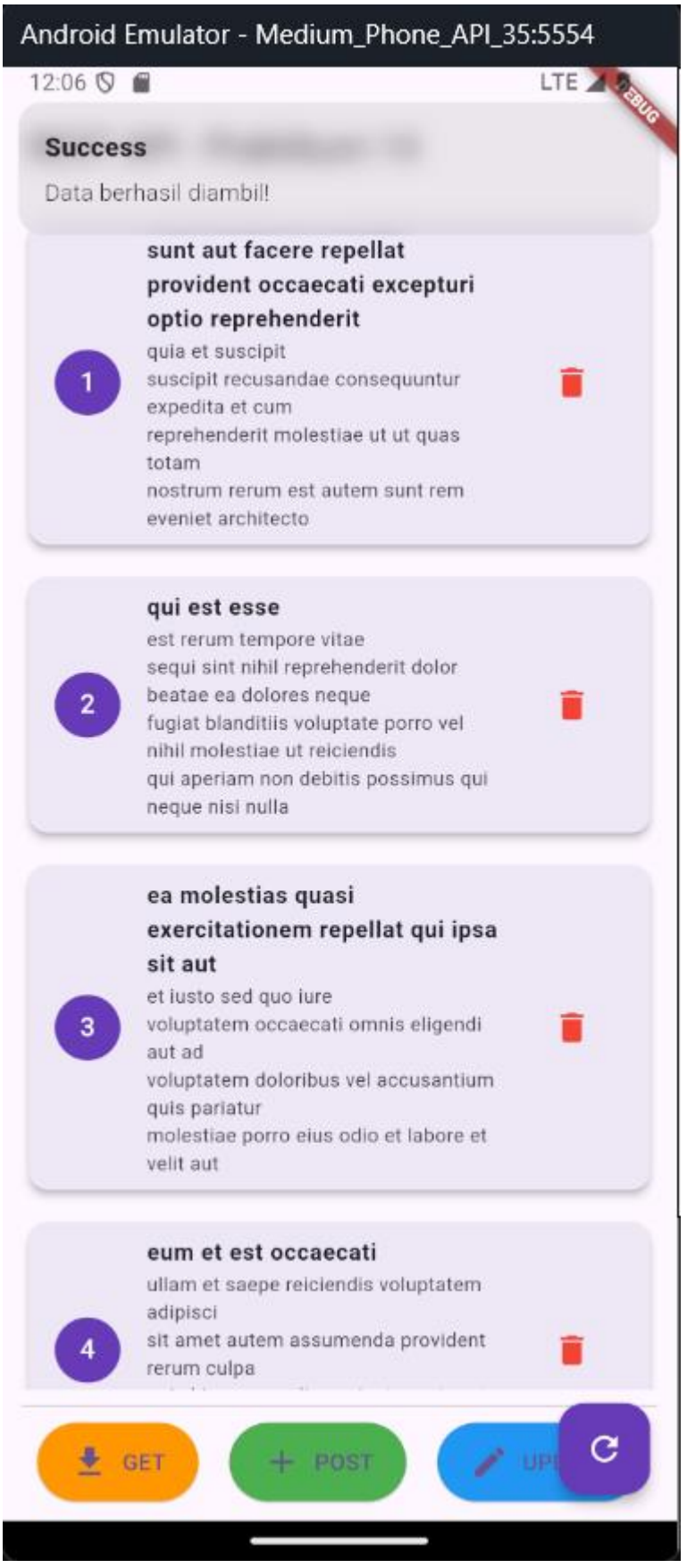
Source code :

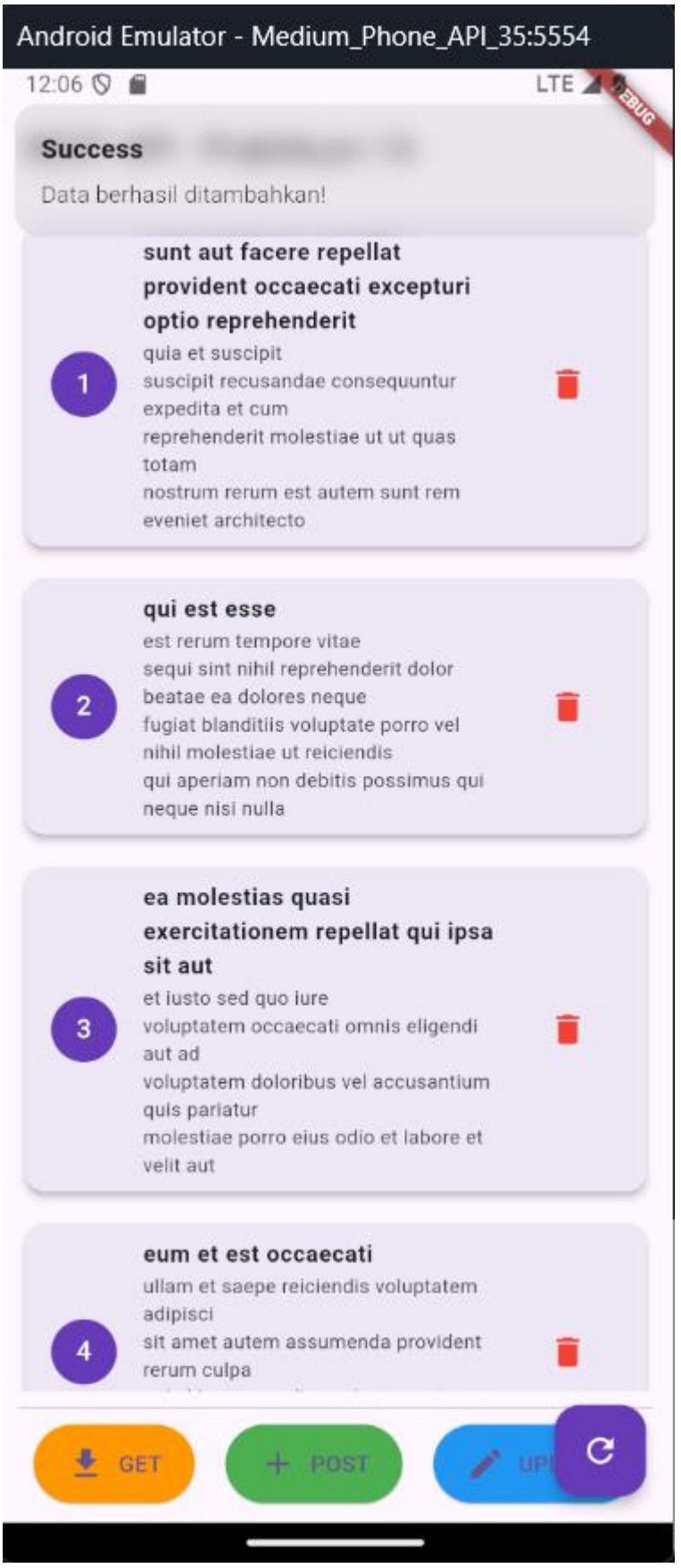
```
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:pertemuan_14/screen/home_screen.dart';

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

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

  @override
```

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

Output :


Android Emulator - Medium_Phone_API_35:5554
12:06
LTE
DEBUG
Success
Data berhasil diambil!
1
sunt aut facere repellat provident occaecati excepturi optio reprehenderit
quia et suscipit
suscipit recusandae consequuntur expedita et cum
reprehenderit molestiae ut ut quas totam
nostrum rerum est autem sunt rem eveniet architecto
2
qui est esse
est rerum tempore vitae
sequi sint nihil reprehenderit dolor beatae ea dolores neque
fugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis
qui aperiam non debitis possimus qui neque nisi nulla
3
ea molestias quasi exercitationem repellat qui ipsa sit aut
et iusto sed quo iure
voluptatem occaecati omnis eligendi aut ad
voluptatem doloribus vel accusantium quis pariatur
molestiae porro eius odio et labore et velit aut
4
eum et est occaecati
ullam et saepe reiciendis voluptatem adipisci
sit amet autem assumenda provident rerum culpa
GET
POST


Android Emulator - Medium_Phone_API_35:5554
12:06
LTE
DEBUG
Success
Data berhasil diambil!
1
sunt aut facere repellat provident occaecati excepturi optio reprehenderit
quia et suscipit
suscipit recusandae consequuntur expedita et cum
reprehenderit molestiae ut ut quas totam
nostrum rerum est autem sunt rem eveniet architecto
2
qui est esse
est rerum tempore vitae
sequi sint nihil reprehenderit dolor beatae ea dolores neque
fugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis
qui aperiam non debitis possimus qui neque nisi nulla
3
ea molestias quasi exercitationem repellat qui ipsa sit aut
et iusto sed quo iure
voluptatem occaecati omnis eligendi aut ad
voluptatem doloribus vel accusantium quis pariatur
molestiae porro eius odio et labore et velit aut
4
eum et est occaecati
ullam et saepe reiciendis voluptatem adipisci
sit amet autem assumenda provident rerum culpa
GET
POST


Android Emulator - Medium_Phone_API_35:5554
12:06
LTE
DEBUG
Success
Data berhasil ditambahkan!
1
sunt aut facere repellat provident occaecati excepturi optio reprehenderit
quia et suscipit
suscipit recusandae consequuntur expedita et cum
reprehenderit molestiae ut ut quas totam
nostrum rerum est autem sunt rem eveniet architecto
2
qui est esse
est rerum tempore vitae
sequi sint nihil reprehenderit dolor beatae ea dolores neque
fugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis
qui aperiam non debitis possimus qui neque nisi nulla
3
ea molestias quasi exercitationem repellat qui ipsa sit aut
et iusto sed quo iure
voluptatem occaecati omnis eligendi aut ad
voluptatem doloribus vel accusantium quis pariatur
molestiae porro eius odio et labore et velit aut
4
eum et est occaecati
ullam et saepe reiciendis voluptatem adipisci
sit amet autem assumenda provident rerum culpa
GET
POST


Android Emulator - Medium_Phone_API_35:5554
12:07
LTE
DEBUG
Success
Data berhasil diperbarui!
1
Updated Title
Updated Body
2
qui est esse
est rerum tempore vitae
sequi sint nihil reprehenderit dolor
beatae ea dolores neque
fugiat blanditiis voluptate porro vel
nihil molestiae ut reiciendis
qui aperiam non debitis possimus qui
neque nisi nulla
3
ea molestias quasi
exercitationem repellat qui ipsa
sit aut
et iusto sed quo iure
voluptatem occaecati omnis eligendi
aut ad
voluptatem doloribus vel accusantium
quis pariatur
molestiae porro eius odio et labore et
velit aut
4
eum et est occaecati
ullam et saepe reiciendis voluptatem
adipisci
sit amet autem assumenda provident
rerum culpa
quis hic commodi nesciunt rem tenetur
doloremque ipsam iure
quis sunt voluptatem rerum illo velit
nesciunt quas odio
repudiandae veniam quaerat sunt sed
GET
POST

Penjelasan Program

Program ini adalah aplikasi berbasis Flutter yang memanfaatkan REST API untuk melakukan operasi data seperti pengambilan (GET), penambahan (POST), pembaruan (PUT), dan penghapusan (DELETE) data dari sumber eksternal. Aplikasi ini dibangun dengan menggunakan framework Flutter dan didukung oleh state management GetX untuk efisiensi dalam mengelola data secara reaktif. Saat pengguna melakukan aksi tertentu, seperti mengambil data dari API, aplikasi secara otomatis memperbarui antarmuka pengguna melalui mekanisme observasi dari GetX. Selain itu, aplikasi memberikan umpan balik yang jelas kepada pengguna dengan menampilkan Snackbar setiap kali operasi berhasil atau gagal, sehingga meningkatkan pengalaman pengguna.

Antarmuka aplikasi dirancang agar intuitif dengan tampilan daftar data yang estetis menggunakan komponen Card yang modern dan informatif. Elemen-elemen visual seperti CircleAvatar untuk menampilkan ID data dan ikon tombol memberikan kejelasan tambahan. Pengguna juga dapat dengan mudah melakukan berbagai aksi melalui tombol yang telah diberi ikon dan warna yang konsisten untuk setiap operasi. Secara keseluruhan, aplikasi ini tidak hanya berfungsi sebagai alat pengelolaan data API tetapi juga sebagai demonstrasi yang kuat untuk mengintegrasikan GetX dengan Flutter dalam membangun aplikasi yang interaktif dan responsif.