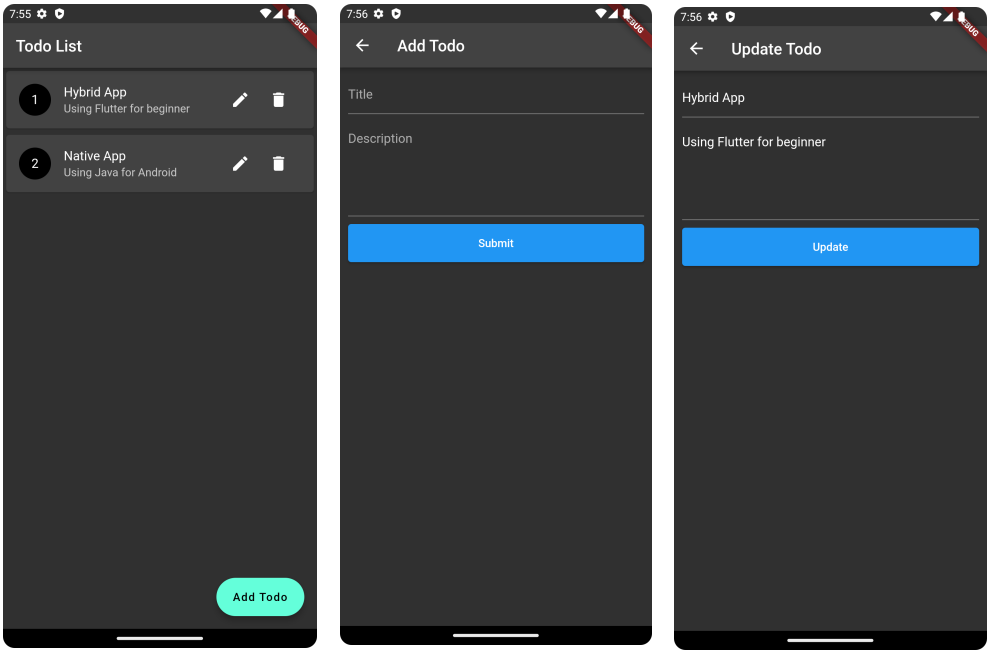
Flutter

Training Assignments

Working with REST APIs (cont)

Overview

Build Todo App and store the task on the real server. We have a flutter application which does all the four common operations like Create, Read, Update, Delete(CRUD).



Tasks

1. List All Todo

```dart
class TodoList extends StatelessWidget {
  const TodoList({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      theme: ThemeData.dark(),
      home: const _HomePage(),
    );  // MaterialApp
  }
}

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

  @override
  State<_HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<_HomePage> {
  late Future<List<Todo>> futureTodos;

```

```dart
@override
void initState() {
  super.initState();
  futureTodos = fetchTodos();
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(title: const Text('Todo List')),
    body: Center(
      child: RefreshIndicator(
        onRefresh: _refreshTodos,
        child: FutureBuilder(
          future: futureTodos,
          builder: (context, snapshot) {
            if (snapshot.hasError) {
              return Text('Retrieve Failed ${snapshot.error}');
            } else if (snapshot.hasData) {
              final List<Todo> todos = snapshot.data!;
              return ListView.builder(
                  itemCount: todos.length,
                  itemBuilder: (context, index) => Card(
                      child: (ListTile(
                          leading: CircleAvatar(child: Text('${index + 1}')),
                          title: Text('${todos[index].title}'),
                          subtitle: Text('${todos[index].description}'),
                          trailing: SizedBox(
                              width: 100,
                              child: Row(
                                children: [
                                  IconButton(
                                      onPressed: () =>
                                          navigateToAddPage(todos[index]),
                                      icon: const Icon(Icons.edit)),  // IconB
                                  IconButton(
                                      onPressed: () =>
                                          deleteById(todos[index].id!),
                                      icon: const Icon(Icons.delete))  // Icon
                                ],
                              )),  // Row, SizedBox
                      )),  // ListTile
                  ));  // Card, ListView.builder
            } else {
              return const CircularProgressIndicator();
            }
```

```
            },
          ),  // FutureBuilder
        ),  // RefreshIndicator
      ),  // Center
      floatingActionButton: FloatingActionButton.extended(
          onPressed: () => navigateToAddPage(null),
          label: const Text('Add Todo')),  // FloatingActionButton.extended
    );  // Scaffold
  }

  List<Todo> parseTodos(String response) {
    final parsed = jsonDecode(response).cast<Map<String, dynamic>>();
    return parsed.map<Todo>((json) => Todo.fromJson(json)).toList();
  }

  Future<List<Todo>> fetchTodos() async {
    const url = "https://60db1a79801dcb0017290e61.mockapi.io/todos";
    final uri = Uri.parse(url);
    final response = await http.get(uri);

    if (response.statusCode == 200) {
      return parseTodos(response.body);
    } else {
      throw Exception('Failed to load Todo');
    }
  }

  Future<void> _refreshTodos() async {
    final result = fetchTodos();
    setState(() {
      futureTodos = result;
    });
  }

  Future<void> navigateToAddPage(Todo? todo) async {
    final route = MaterialPageRoute(
        builder: (context) => AddTodoPage(
              todo: todo,
            ));  // AddTodoPage, MaterialPageRoute
    await Navigator.push(context, route);
    _refreshTodos();
  }

  Future<void> deleteById(String id) async {
    // Delete the item
    final url = "https://60db1a79801dcb0017290e61.mockapi.io/todos/$id";
```

```
123       final uri = Uri.parse(url);
124       final response = await http.delete(uri);
125
126       // Remove item from the list
127       if (response.statusCode == 200) {
128         _refreshTodos();
129         showMessage("Deletion Success");
130       } else {
131         showMessage('Deletion Failed');
132       }
133     }
134
135     void showMessage(String message) {
136       ScaffoldMessenger.of(context)
137           .showSnackBar(SnackBar(content: Text(message)));
138     }
139   }
```

2. Create Todo Task & Edit Todo Task

```
6   class AddTodoPage extends StatefulWidget {
7     final Todo? todo;
8
9     const AddTodoPage({super.key, this.todo});
10
11    @override
12    State<AddTodoPage> createState() => _AddTodoPageState();
13  }
14
15  class _AddTodoPageState extends State<AddTodoPage> {
16    TextEditingController titleController = TextEditingController();
17    TextEditingController descriptionController = TextEditingController();
18    bool isUpdate = false;
19
20    @override
21    void initState() {
22      super.initState();
23
24      if (widget.todo != null) {
25        titleController.text = widget.todo!.title!;
26        descriptionController.text = widget.todo!.description!;
27        isUpdate = true;
28      }
```

```
29    }
30
31    @override
32    Widget build(BuildContext context) {
33      return Scaffold(
34        appBar: AppBar(
35          title: Text(isUpdate ? 'Update Todo' : 'Add Todo'),
36        ),  // AppBar
37        body: ListView(
38          padding: const EdgeInsets.all(10),
39          children: [
40            TextFormField(
41              controller: titleController,
42              decoration: const InputDecoration(hintText: 'Title'),
43            ),  // TextFormField
44            const SizedBox(
45              height: 10,
46            ),  // SizedBox
47            TextFormField(
48              controller: descriptionController,
49              decoration: const InputDecoration(hintText: 'Description'),
50              keyboardType: TextInputType.multiline,
51              minLines: 5,
52              maxLines: 8,
53            ),  // TextFormField
54            const SizedBox(
55              height: 10,
56            ),  // SizedBox
57            ElevatedButton(
58                onPressed: isUpdate ? update : submit,
59                child: Padding(
60                  padding: const EdgeInsets.all(16.0),
61                  child: Text(isUpdate ? 'Update' : 'Submit'),
62                ))  // Padding, ElevatedButton
63          ],
64        ),  // ListView
65      );  // Scaffold
66    }
67
68    Future<void> submit() async {
69      // Get the data from form
70      final title = titleController.text;
71      final description = descriptionController.text;
72      final body = {"title": title, "description": description};
73
74      // Submit data to the server
```

```dart
    const url = "https://60db1a79801dcb0017290e61.mockapi.io/todos";
    final uri = Uri.parse(url);
    final response = await http.post(uri, body: body);

    // Show success or fail message based on status
    // log('${response.statusCode}');
    // log(response.body);
    if (response.statusCode == 201 && mounted) {
      titleController.text = '';
      descriptionController.text = '';
      showMessage('Creation Success');
    } else {
      showMessage('Creation Failed');
    }
  }

  Future<void> update() async {
    final todo = widget.todo;
    final id = todo!.id!;
    final url = "https://60db1a79801dcb0017290e61.mockapi.io/todos/$id";
    final uri = Uri.parse(url);
    final body = {
      "title": titleController.text,
      "description": descriptionController.text
    };

    final response = await http.put(uri, body: body);

    if (response.statusCode == 200 && mounted) {
      showMessage('Edition Success');
    } else {
      showMessage('Edition Failed');
    }
  }

  void showMessage(String message) {
    ScaffoldMessenger.of(context)
        .showSnackBar(SnackBar(content: Text(message)));
  }
}
```

--THE END—