Lab Manual for CSM3114 - Framework-Based Mobile Application Development

Prepared by Mohamad Nor Hassan*

October 2024

*Universiti Malaysia Terengganu

The Flutter framework let the students to develop cross-platform mobile applications which can run on Android or iOS platforms.

This lab session will introduce to the student on the development of basic mobile applications focusing on the Flutter and Dart fundamentals that emphasise on core Flutter and dart syntax when developing the solutions.

The learning outcomes of this lab session are:

1. To combine and use the existing Flutter's widgets that cover in in previous lab session in order to develop the simple mobile app.
2. To create a New Task by using *Dialog Box*.
3. To use *ListView.builder* to display a to-do list record.
4. Completing a simple *To-Do List* application.

1 Development of a simple *To-Do List* mobile application

1.1 Creating a Flutter project

1. Open your command prompt.

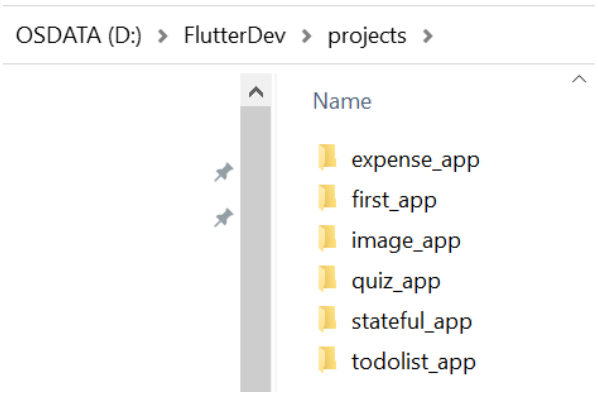
2. Go to your current Flutter's project folder. [**Note**: Your command to navigate to *FlutterDev*'s folder might be different if you use DOS command prompt..!]

```
MINGW64:/D/FlutterDev/projects

Owner@MNor-HPElite-G3 MINGW64 ~
$ cd /D

Owner@MNor-HPElite-G3 MINGW64 /D
$ cd FlutterDev

Owner@MNor-HPElite-G3 MINGW64 /D/FlutterDev
$ cd projects

Owner@MNor-HPElite-G3 MINGW64 /D/FlutterDev/projects
$ ls -l
total 20
drwxr-xr-x 1 Owner 197609 0 Nov 15 14:18 expense_app/
drwxr-xr-x 1 Owner 197609 0 Aug 22  2022 first_app/
drwxr-xr-x 1 Owner 197609 0 Oct 28 23:22 image_app/
drwxr-xr-x 1 Owner 197609 0 Oct 21 17:48 quiz_app/
drwxr-xr-x 1 Owner 197609 0 Nov  2 16:39 stateful_app/

Owner@MNor-HPElite-G3 MINGW64 /D/FlutterDev/projects
$ pwd
/D/FlutterDev/projects
```

3. Creating the Flutter's project known as *todolist_app*.

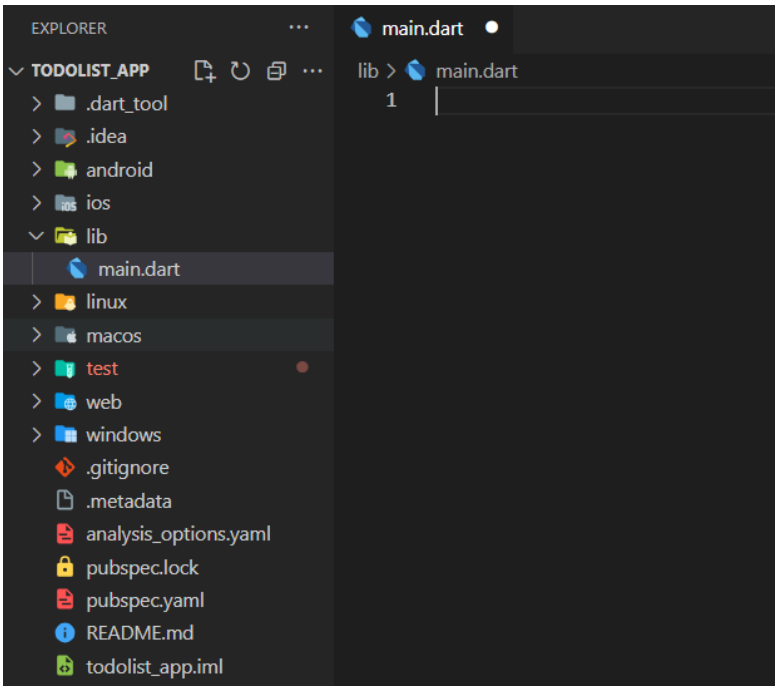
```
Owner@MNor-HPElite-G3 MINGW64 /D/FlutterDev/projects
$ flutter create todolist_app

  A new version of Flutter is available!

  To update to the latest version, run "flutter upgrade".

Creating project todolist_app...
Resolving dependencies in todolist_app...
Got dependencies in todolist_app.
Wrote 129 files.

All done!
You can find general documentation for Flutter at: https://docs.flutter.dev/
Detailed API documentation is available at: https://api.flutter.dev/
If you prefer video documentation, consider: https://www.youtube.com/c/flutterdev

In order to run your application, type:

  $ cd todolist_app
  $ flutter run

Your application code is in todolist_app\lib\main.dart.
```

4. Open your *FlutterDev*'s folder. You need to ensure the *todolist_app* created.



5. Go to VS Code IDE. Open *todolist_app*'s project folder.

6. Go to *main.dart* file. Delete the existing source code.



1.2 Construct the main body of code for *To-Do List* app

1. The purpose of this app are:

   (a) To remove an item from a list are

   (b) To view the task is marked as 'done' via a tap.

2. Create a *MyMainPage* as a Stateful widget.

3. Initialise the to-do list record.

```dart
/*Author: MNor
 Date: 18 Sept 2023
 Purpose: To create a simple To-Do List app by applying ShowDialogBox widget
          and ListView.builder
*/
import 'package:flutter/material.dart';

Run | Debug | Profile
void main() => runApp(MaterialApp(home: MyMainPage()));

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

  @override
  State<MyMainPage> createState() => _MyMainPageState();
}

class _MyMainPageState extends State<MyMainPage> {
  //Initiliase a to-do list records...
  List<String> _todoRecords = ['Task 1', 'Task 2', 'Task 3', 'Task 4'];

  @override
  Widget build(BuildContext context) {
    return const Placeholder();
  }
}
```

1.3 Construct pop up *Dialog Box and TexField for data entry*

1. Create a method or function known as *_addingTodo*.

2. Inside this method, implement the following requirements:

   (a) Create *ShowDialog* widget.

   (b) Inside the *builder* property, return a *AlertDialog* widget inside the *ShowDialog* widget.

   (c) Capture the value from *TextField* widget using *onChanged* property.

   (d) Then, assign a *Cancel* and *Submit* button in the *AlertDialog* widget.

   (e) In the *Submit* button, at the *onPressed* property, assign a *setState()* method and perform the addition on the current value key-in by the user via *TextField* widget.

   (f) In each of this button, remember to force the app to close the *AlertDialog* widget when user finishing perform the action. [**Hint:** Need to use *Navigator.of(context).pop()*]

3. Complete the coding for *_addingTodo* method.

4. Review your coding for *_addingTodo* method.

```dart
//Add _addingToDo() method..
void _addingToDO() {
  showDialog(
      context: context,
      builder: (BuildContext context) {
        String newToDo = '';

        return AlertDialog(
          title: Text('Enter New Task Below:'),
          content: TextField(
            onChanged: (value) {
              // Note: Can also use TextController
              newToDo = value;
            },
          ), // TextField
          actions: <Widget>[
            TextButton(
              onPressed: () {
                Navigator.of(context).pop();
              },
              child: Text('Cancel'),
            ), // TextButton
            TextButton(
              onPressed: () {
                //Call the setState to re-run the rebuilding of Widgets...
                //Then, add the current value from TextField into
                //_todoRecords list.
                setState(() {
                  _todoRecords.add(newToDo);
                });
                // Call navitor to automatically closed AlertDialog box..
                Navigator.of(context).pop();
              },
              child: Text('Submit'),
            ), // TextButton
          ], // <Widget>[]
        ); // AlertDialog
      });
}
```

1.4 Displaying *To_Do List* record using *ListView.builder* widget

1. Inside the *build* method, perform the following:

   (a) Return the *Scaffold* widget.

   (b) Adding the *AppBar* widget.

   (c) Put the title as 'To Do List'.

   (d) In the *body* property, add the *ListView.builder* widget in order to fetch and iterate a list of To-Do list records.

   (e) Then, use a *ListTile* widget to display a record.

2. Implement your coding.

```dart
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('To Do List'),
      centerTitle: true,
    ), // AppBar
    body: ListView.builder(
        itemCount: _todoRecords.length,
        itemBuilder: (context, index) {
          final toDo = _todoRecords[index];

          return ListTile(
            title: Text(toDo),
          ); // ListTile
        }), // ListView.builder
  ); // Scaffold
}
}
```

3. Review and save your coding.

4. Open and your Android virtual device or emulator..

5. Run your To-Do List app.

6. You should get the following output.

1.5 Removing specific to-do list record when user tap the specific record

1. In order to implement this request, inside the *ListTile* widget, we need to use *onTap* property.

2. Implement the following logic at the *onTap* property.

```
73                  return ListTile(
74                    title: Text(toDo),
75                    onTap: () {
76                      setState(() {  // Calling setState() to re-build the widget..
77                        _todoRecords.removeAt(index);
78                      });
79                    },
80                  ); // ListTile
```

3. Review and save your coding.

4. Run your program.

5. Perform a testing by deleting Task 3.

6. Your should get the following output upon deleting Task 3.

1.6 Adding a *FloatingActionButton* in order to display *ShowDialogBox* for data entry

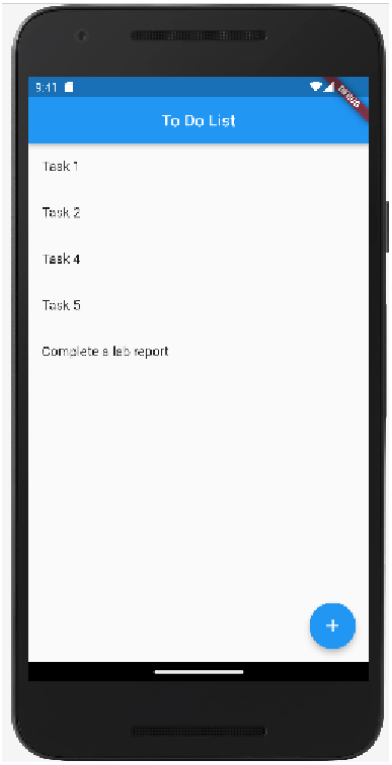
1. Next, to create *FloatingActionButton* widget. User will use this floating button to make a new task into the existing To-Do List app.

2. You should create *FloatingActionButton* widget after *ListView.builder* widget.

3. Add the *Icon* widget at the *child* property.

4. In order to display the new data entry via *ShowDialogBox* widget, at the *onPressed* property you should call the *_addingToDo()* method.

```
82            }), // ListView.builder
83          floatingActionButton: FloatingActionButton(
84            // For data new data entry
85            child: Icon(Icons.add),
86            onPressed: _addingToDo,
87          ), // FloatingActionButton
88        ); // Scaffold
89      }
90    }
```



5. Complete your coding, save and run the program.

6. Perform a testing by key-in new task as 'Task 5'.

7. Continue data entry by key-in new task as 'Complete a lab report'.

8. You should get the sample of output shown in the report manual.

9. Attach the source code from the beginning of your program and the output in the report.

1.7 Exercise

1. Based on the existing coding, enhance the UI appearance using the appropriate theme to increase the look and feel of UI such as the colour, increasing the font-size and others.