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 use *Alert Dialog* and *Show Dialog* widgets for the app.
2. To use *TextEditController* in order to keep track the value of *Text* widget that has been changed for the app.
3. To use *Navigator* to perform navigation between UIs using *push* or *pop* methods.
4. Implementing *ListView* and *ListTile* widgets.
5. Implementing *ListView.builder*.

1 More on Flutter widgets used to develop mobile application

1.1 Implementing *showDialog*, *Alert Dialog* widgets and using *Navigator*

1. The *Alert Dialog* widget is used to prompt with a list of actions when there is an interaction between end user and the UI such as action to save or cancel the transaction.
2. Open your Visual Studio IDE.
3. Reuse the *main.dart* files from your previous lab session. [Note: Important to save your previous work..!].
4. Delete the existing code from *main.dart*.
5. Create a main program by running the *MyApp()*.
6. Then, create a *MyApp* widget by defining as a *StatelessWidget* widget.
7. Inside the *myApp* widget, assign a *build* method and return a *MaterialApp* widget with properties *title* and *home*. [**Note**: For *home* property, assign a *MyMainPage()* widget which later will be developed using *Stateful* widget.

```dart
/*
Author  : MNor
Date    : 15 Sept 2023
Purpose : Implementing Alert dialog box and Navigation.
*/

import 'package:flutter/material.dart';

Run | Debug | Profile
void main() => runApp(MyApp());

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Dialog Box',
      home: MyMainPage(),
    ); // MaterialApp
  }
}
```

8. Next, we need to create *MyMainPage()* widget as a *Stateful* widget because we need to retain the value key-in by the user via *TextField* later clicking the *TextButton Save* will rebuild the main widget. [**Hint**: In VS Code, to create a *Stateful* widget, type *st*, subsequently the snippet code will appear. Then, select the option for *Stateful* widget and change the name of *Stateful* widget] as *MyMainPage*.

9. The following is the requirements for constructing the *MyMainPage StatefulWidget*:

   (a) Define the variable and a controller for an editable text field.

   (b) Create a method known as *_showInputDialog()* to display InputDialog. In this method, use *showDialog*, *AlertDialog*, *Text*, *TextField* and *TextButton* widgets.

   (c) In *TextButton* widget for *Save* button, at the *onPressed* property, use *setState()* method in order to retain the current value that key-in by the user via *TextField* widget.

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

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

```dart
class _MyMainPageState extends State<MyMainPage> {
  //Initialise the variables..
  String _inputText = '';
  final myController = TextEditingController();

  //Create method to show Alert dialog box
  void _showInputDialog() {
    showDialog(
        context: context,
        builder: (BuildContext context) {
          return AlertDialog(
            title: const Text('Enter text..!'),
            content: TextField(
              controller: myController,
              decoration: const InputDecoration(hintText: 'Enter text'),
            ), // TextField
            actions: [
              TextButton(
                child: Text('Cancel'),
                onPressed: () {
                  Navigator.of(context).pop(); // Close the dialog box
                },
              ), // TextButton
```

10. Continue the coding by adding the *TextButton* widget for *Save* button.

11. Implement the *setState()* method inside *TextButton*.

12. For both *TextButton* widgets that representing *Cancel* and *Save*, when user click these buttons, you should hide the dialog box. This can be done via for *onPressed* property by assigning *Navigator* widget and calling the *pop()* method.

```
53                  TextButton(
54                    child: Text('Save'),
55                    onPressed: () {
56                      setState(() {
57                        _inputText = myController.text;
58                      });
59                      Navigator.of(context).pop(); // Close the dialog box
60                    },
61                  ), // TextButton
62                ],
63              ); // AlertDialog
64            });
65      }
66
```

13. Finally, construct the main UI for the *_MyMainPageState* class based on the following requirements:

    (a) Create *build()* method.

    (b) Return the UI as *Scaffold* widget.

    (c) Inside the *Scaffold* widget, add the *AppBar*, *Center*, *Text* and *FloatingActionButton* widgets.

    (d) Inside the *FloatingActionButton* widget, assigned the *_showInputDialog* method to *onPressed* property and add *Icon* widget tha representing the Edit mode.

14. Complete your coding.
15. Save and run your coding.
16. Your should get the sample of UI display after part (d).
17. Attached the both source codes and the outputs for program in the report.

1.2 Using *ListView* widget

1. Save your coding for the program written in part 1.1 in *main.dart* in Notepad for future reference.
2. Delete the existing code from *main.dart*.
3. Create a main program by running the *MyApp()*.
4. Then, create a *MyApp* widget by defining as a *StatelessWidget* widget.
5. In *MyApp* widget, create *build()* method.
6. In the *build()* method implementation, return a *MaterialApp* widget and follow by the *Scaffold* widget.

```
1  /*
2  Author  : MNor
3  Date    : 15 Sept 2023
4  Purpose : Implementing ListView Widget.
5  */
6
7  import 'package:flutter/material.dart';
8
   Run | Debug | Profile
9  void main() => runApp(MyApp());
10
```

7. In the *Scaffold* widget, implement the UI for the following requirements:
   (a) Add *AppBar* widget.
   (b) In the *body* property, add a *ListView* widget.
   (c) To display a sample of three (3) records, use three (3) *ListTile* widgets.
   (d) For each of *ListTile* widget, apply the properties that representing *leading*, *title*, *subtitle*, *trailing* and *onTap*.

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'MyApp',
      home: Scaffold(
        appBar: AppBar(
          title: Text('ListView Example'),
          centerTitle: true,
        ), // AppBar
```

```dart
        body: ListView(
          children: [
            ListTile(
              leading: Icon(Icons.person),
              title: Text('MNor'),
              subtitle: Text('mnor@gmail.com'),
              trailing: Icon(Icons.arrow_forward),
              onTap: () => print('Tapping the ListTile...!'),
            ), // ListTile
            ListTile(
              leading: Icon(Icons.person),
              title: Text('Aziz'),
              subtitle: Text('aziz@gmail.com'),
              trailing: Icon(Icons.arrow_forward),
              onTap: () => print('Tapping the ListTile...!'),
            ), // ListTile
            ListTile(
              leading: Icon(Icons.person),
              title: Text('Peter Ng'),
              subtitle: Text('peterng@gmail.com'),
              trailing: Icon(Icons.arrow_forward),
              onTap: () => print('Tapping the ListTile...!'),
            ), // ListTile
            ListTile(
              leading: Icon(Icons.person),
              title: Text('Krisnan'),
              subtitle: Text('krisnan@gmail.com'),
              trailing: Icon(Icons.arrow_forward),
              onTap: () => print('Tapping the ListTile...!'),
            ), // ListTile
          ],
        ), // ListView
      ), // Scaffold
    ); // MaterialApp
  }
}
```

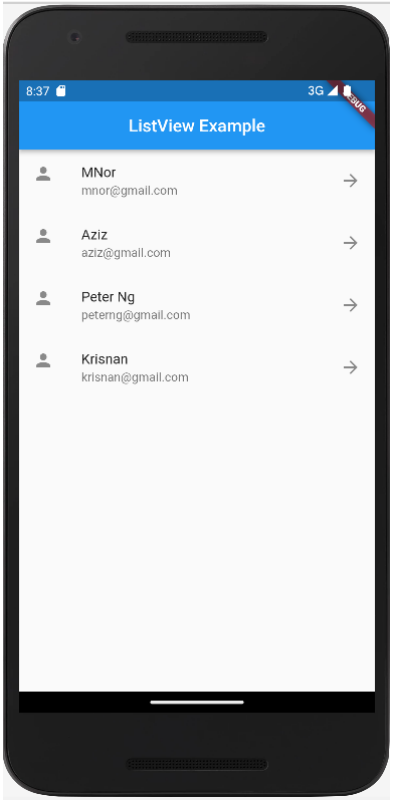
8. Review your code and ensure that are no syntax errors.

9. Open your Android virtual device or emulator.

10. Save and run you program.

11. Test your UI by tapping one or two records. Check your Debug Console. What is the output?

12. You should get the following sample UI.



13. Please attach the source and the output of UI in your report.

1.3 Exercise

1. Based on the source code written for *_MyMainPageState()* Stateful widget in part 1.1, extend the app by adding the second *FloatingActionButton* widget that purposely used to reset the value written in the text field.

2. You should create new method or function on how to implement the reset process by calling it via second *FloatingActionButton*.

3. Attached the portion of the source codes and the outputs for program in the report.

1.4 Exercise

1. Create a simple UI to display the a list of records that display only the title as below:

   (a) Java Programming

(b) Front-end development with React

(c) Next.js Framework

(d) Restful API using Python

(e) Cross-platform mobile apps with Google Flutter

2. Complete your coding and run the program.

3. Attach the source code and the sample of UI output in the report.

2 Populating Data Into *ListView*

2.1 Populate Simple Array of List into *ListView* widget

1. Save your coding for the program written in part 1.2 in *main.dart* in Notepad for future reference.

2. Delete the existing code from *main.dart*.

3. Create a main program by running the *MyApp()*.

```dart
/*
Author  : MNor
Date    : 16 Sept 2023
Purpose : Populating array of list into ListView Widget.
*/

import 'package:flutter/material.dart';

Run | Debug | Profile
void main() => runApp(MyApp());
```

4. Then, create a *MyApp* widget by defining as a *StatelessWidget* widget.

5. Define a list of records that consists of 5 reference IT books.

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

  List<String> refBooks = [
    "Java Servlet Programming",
    "Front-end with Angular",
    "Linux Administration",
    "MongoDB for NoSQL",
    "Microsoft Azure Security"
  ];
```

6. In *MyApp* widget, create *build()* method.

7. In the *build()* method implementation, return a *MaterialApp* widget and follow by the *Scaffold* widget.

8. In the *Scaffold* widget, implement the UI for the following requirements:

   (a) Add *AppBar* widget.

(b) In the *body* property, add a *Container* widget.

(c) In the *Container* widget, add a *ListView* widget by referencing to *refBooks* list to the *Container* and *Text* widgets respectively. [**Note**: To fetch record from array of list, use *map* and *toList()* methods].

```
21
22    @override
23    Widget build(BuildContext context) {
24      return MaterialApp(
25          title: 'MyApp',
26          home: Scaffold(
27            appBar: AppBar(
28              title: const Text('Populate A List'),
29              centerTitle: true,
30            ), // AppBar
31            body: Container(
32              padding: EdgeInsets.all(60),
33              child: ListView(
34                children: refBooks.map((e) {
35                  return Container(
36                    child: Text(e),
37                    margin: EdgeInsets.all(5),
38                    padding: EdgeInsets.all(20),
39                    color: Colors.blue[200],
40                  ); // Container
41                }).toList(),
42              ), // ListView
43            ), // Container
44          )); // Scaffold // MaterialApp
45    }
46  }
```

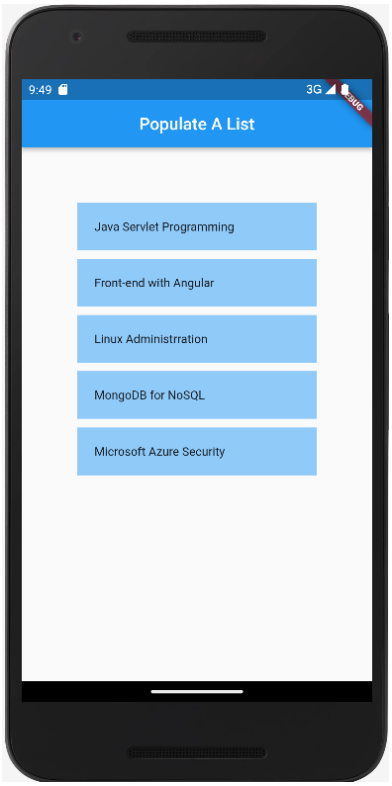
(d) Populate the records inside the *Container* widget.

(e) Finally, enhance the style for *Container* widget for properties such as *margin*, *padding* and *color*.

9. Complete your coding and run the program.

10. Attach the source code and the sample of UI output in the report.

11. You should get the following UI output.



2.2 Fetching records from a list into *ListView* and *ListTile* widgets

1. Save your coding for the program written in part 2.1 in *main.dart* in Notepad for future reference.

2. Delete the existing code from *main.dart*.

3. Create a main program by running the *MyApp()*.

```dart
/*
Author  : MNor
Date    : 16 Sept 2023
Purpose : Populating a list of record into ListView & ListTile Widgets.
*/

import 'package:flutter/material.dart';

Run | Debug | Profile
void main() => runApp(MyApp());

```

4. Then, create a *MyApp* widget by defining as a *StatelessWidget* widget.

5. Create a class *City* that consists of city code and city name.

6. Then, create a constructor for class *City* by passing both city code and city name as a parameters.

```dart
class City {
  //Define the attribute
  String code, cityName;

  //Define the constructor..
  City({required this.code, required this.cityName});
}
```

7. In *MyApp* widget, create *build()* method.

8. Create a records that represent a list of city code and city name and stored it as a List of City's object.

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

  List<City> city = [
    City(code: "KUL", cityName: "Kuala Lumpur"),
    City(code: "AKL", cityName: "Auckland"),
    City(code: "DTM", cityName: "Dortmund"),
    City(code: "LPL", cityName: "Liverpool"),
    City(code: "IBZ", cityName: "Ibiza"),
  ];
```

9. In the *build()* method implementation, return a *MaterialApp* widget and follow by the *Scaffold* widget.

```dart
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'MyApp',
      home: Scaffold(
        appBar: AppBar(
          title: Text('Fetch records into ListView'),
          centerTitle: true,
        ), // AppBar
```

10. In the *Scaffold* widget, implement the UI for the following requirements:

    (a) Add *AppBar* widget.

    (b) In the *body* property, add a *ListView* widget.

    (c) In *ListTile* widget, for *children* property, map a list of city and return as a *ListTile* widget and assign the value for city code and cite name to *title* and *subtitle* property.

(d) Enhance the style for *ListTile* widget.

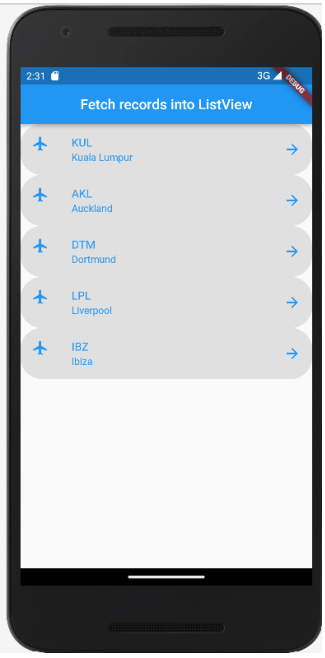
11. Complete the remaining of your coding.

```
37            centerTitle: true,
38          ), // AppBar
39          body: ListView(
40            children: city.map((e) {
41              return ListTile(
42                shape: RoundedRectangleBorder(
43                    borderRadius: BorderRadius.circular(30)),
44                selectedTileColor: Colors.grey[300],
45                selected: true,
46                leading: Icon(Icons.flight),
47                title: Text(e.code),
48                subtitle: Text(e.cityName),
49                trailing: Icon(Icons.arrow_forward),
50              ); // ListTile
51            }).toList(),
52          ), // ListView
53        ), // Scaffold
54      ); // MaterialApp
55    }
56  }
```

12. Save and run your program.

13. Attach the source code and the sample of UI output in the report.

14. You should get the following UI output.



2.3 Exercise

1. Create a list of records that contains student name and payment status..
2. Use *ListView* and *ListTile* widgets to populate a list of student that having payment status as 'Completed'.
3. Attach the source code and the sample of UI output in the report.

3 Implementing A *ListView.builder* Widget

3.1 Using a *ListView.builder* widget

1. The *ListView.builder* widget be able to assist developer to generate a ListView of records in more efficient ways.

2. It simplifies display of record using *ListTile* widget.

3. Save your coding for the program written in part 2.2 in *main.dart* in Notepad for future reference.

4. Delete the existing code from *main.dart*.

5. Create a main program by running the *MyApp()*.

```dart
/*
Author  : MNor
Date    : 16 Sept 2023
Purpose : Populating a list of record using ListView.builderWidget.
*/

import 'package:flutter/material.dart';

Run | Debug | Profile
void main() => runApp(MyApp());

```

6. Then, create a *MyApp* widget by defining as a *StatelessWidget* widget.

7. Create a class *City* that consists of city code and city name.

8. Then, create a constructor for class *City* by passing both city code and city name as a parameters.

```dart
class City {
  //Define the attribute
  String code, cityName;

  //Define the constructor..
  City({required this.code, required this.cityName});
}

```

9. In *MyApp* widget, create *build()* method.

10. Create a records that represent a list of city code and city name and stored it as a List of City's object.

```
19  class MyApp extends StatelessWidget {
20    //const MyApp({super.key});
21    String cityDetail = '';
22
23    List<City> city = [
24      City(code: "KUL", cityName: "Kuala Lumpur"),
25      City(code: "AKL", cityName: "Auckland"),
26      City(code: "DTM", cityName: "Dortmund"),
27      City(code: "LPL", cityName: "Liverpool"),
28      City(code: "IBZ", cityName: "Ibiza"),
29    ];
30
```

11. In the *build()* method implementation, return a *MaterialApp* widget and follow by the *Scaffold* widget.

```
30
31    @override
32    Widget build(BuildContext context) {
33      return MaterialApp(
34        title: 'MyApp',
35        home: Scaffold(
36          appBar: AppBar(
37            title: Text('Fetch records into ListView.builder'),
38            centerTitle: true,
39          ), // AppBar
```

12. In the *Scaffold* widget, implement the UI for the following requirements:

    (a) Add *AppBar* widget.

    (b) In the *body* property, add a *ListView.builder* widget.

    (c) *ListView.builder* widget, add the *card* widget .

    (d) Populate a record from *List<City>* through the use of *ListTile* widget.

    (e) Enhance the style for *ListTile* widget.

13. Complete the remaining of your coding.

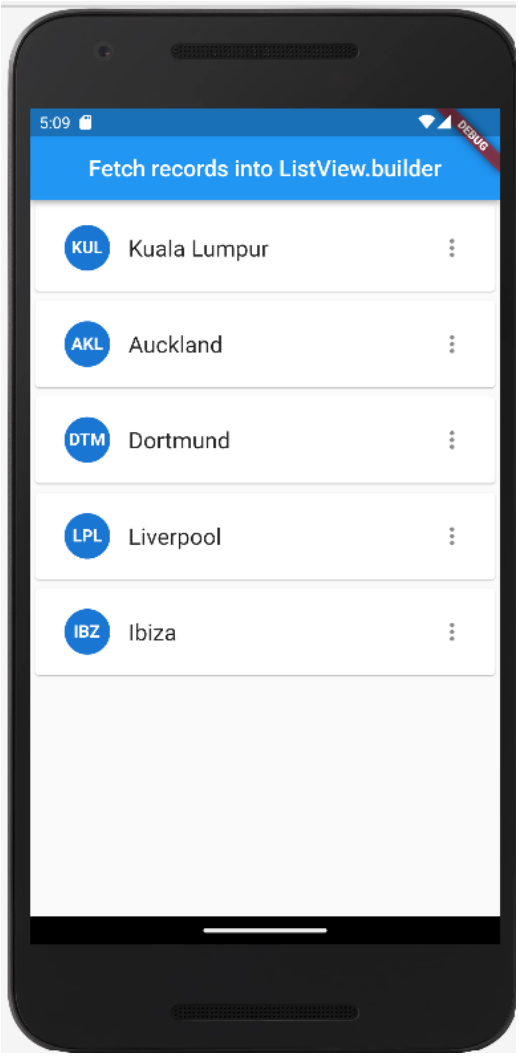
14. Save and run your program.

15. Attach the source code and the sample of UI output in the report.

```
40          body: ListView.builder(
41            itemCount: city.length,
42            itemBuilder: (context, index) {
43              return Card(
44                child: Padding(
45                  padding: EdgeInsets.all(10),
46                  child: ListTile(
47                    leading: CircleAvatar(
48                      child: Text(
49                        city[index].code,
50                        style: TextStyle(
51                          fontSize: 15,
52                          fontWeight: FontWeight.bold,
53                        ), // TextStyle
54                      ), // Text
55                    ), // CircleAvatar
56                    title: Text(
57                      city[index].cityName,
58                      style: TextStyle(
59                        fontSize: 20,
60                      ), // TextStyle
61                    ), // Text
62                    trailing: Icon(Icons.more_vert),
63                    onTap: () {
64                      cityDetail = city[index].cityName;
65                      print('You tapped on $cityDetail');
66                    },
67                  ), // ListTile
68                ), // Padding
69              ); // Card
70            },
71          ), // ListView.builder
72        ), // Scaffold
73      ); // MaterialApp
74    }
75  }
```

16. You should get the following UI output.

17. Attached the source code and the output in your report.


5:09
DEBUG
Fetch records into ListView.builder
KUL
Kuala Lumpur
AKL
Auckland
DTM
Dortmund
LPL
Liverpool
IBZ
Ibiza