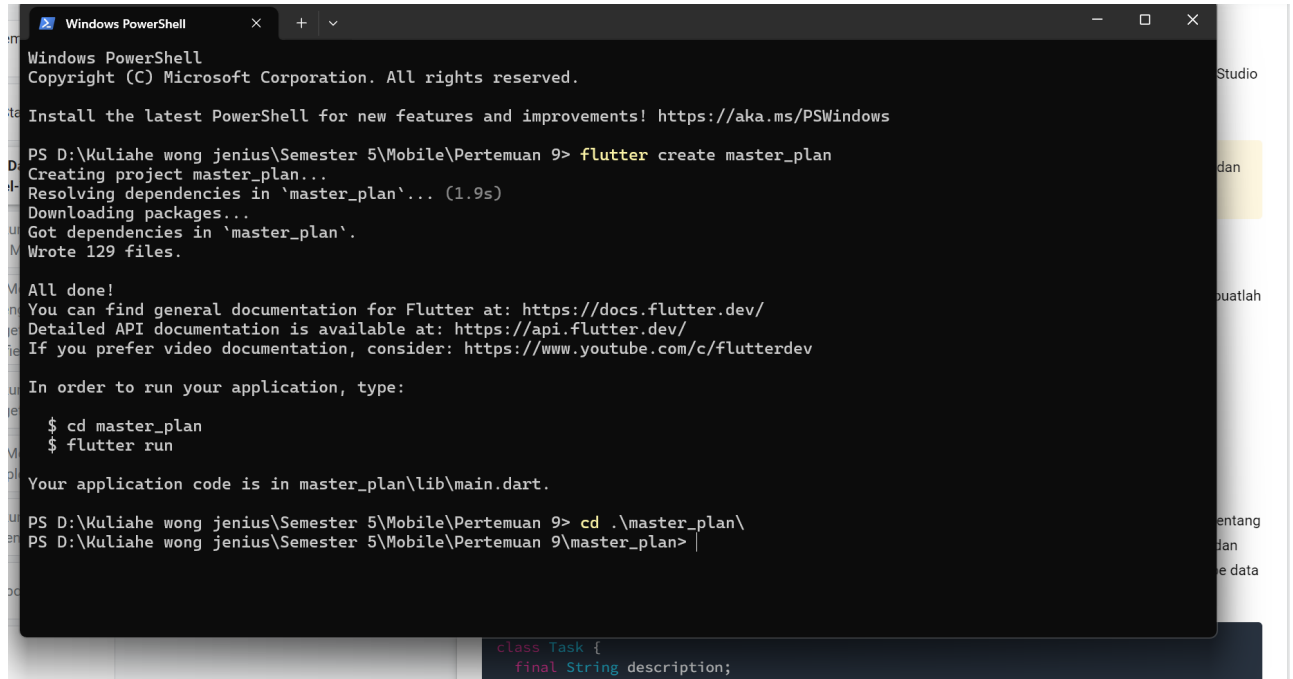


Praktikum 1

Soal 1

1. Buat project baru



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS D:\Kuliahe wong jenius\Semester 5\Mobile\Pertemuan 9> flutter create master_plan
Creating project master_plan...
Resolving dependencies in `master_plan`... (1.9s)
Downloading packages...
Got dependencies in `master_plan`.
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:

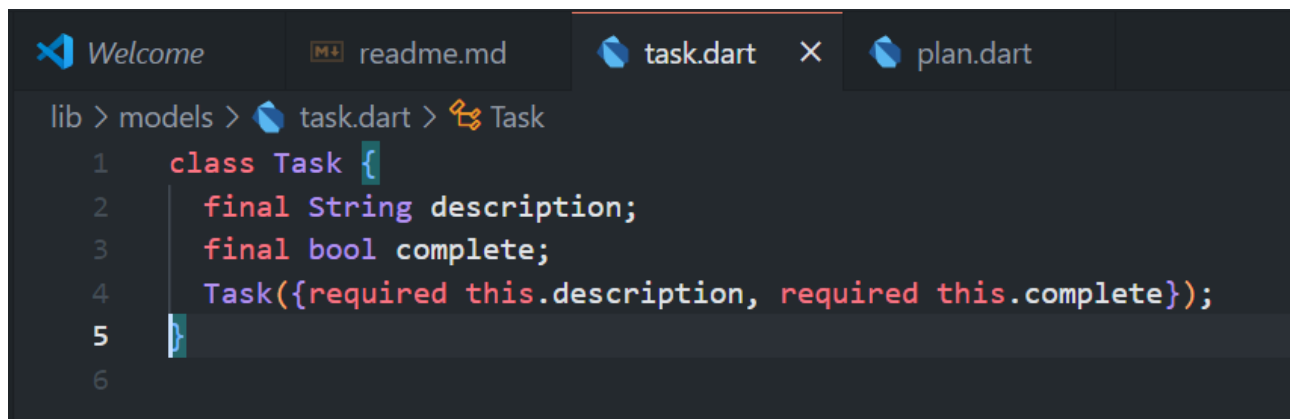
$ cd master_plan
$ flutter run

Your application code is in master_plan\lib\main.dart.

PS D:\Kuliahe wong jenius\Semester 5\Mobile\Pertemuan 9> cd .\master_plan\
PS D:\Kuliahe wong jenius\Semester 5\Mobile\Pertemuan 9\master_plan>

class Task {
  final String description;
```

2. Membuat model task.dart



```
Welcome  readme.md  task.dart  plan.dart

lib > models > task.dart > Task

1  class Task {
2    final String description;
3    final bool complete;
4    Task({required this.description, required this.complete});
5
6
```

3. Membuat model plan.dart

```
lib > models > plan.dart > Plan > Plan
1  import 'package:master_plan/models/task.dart';
2
3  class Plan {
4    final String name;
5    final List<Task> tasks;
6    ⚡
7    Plan({this.name = '', this.tasks = const []});
8  }
9
```

```
lib > models > data_layer.dart
1  export 'plan.dart';
2  export 'task.dart';
3  ⚡
```

4. Buat file data_layer.dart

5. Pindah ke file main.dart

```
lib > main.dart > MasterPlanApp > build
1  import 'package:flutter/material.dart';
2  import 'package:master_plan/views/plan_screen.dart';
3
4  Run | Debug | Profile
5  void main() {
6    runApp(const MasterPlanApp());
7  }
8
9  class MasterPlanApp extends StatelessWidget {
10    const MasterPlanApp({super.key});
11
12    @override
13    Widget build(BuildContext context) {
14      return MaterialApp(
15        theme: ThemeData(primarySwatch: Colors.deepPurple),
16        home: PlanScreen(),
17      ); // MaterialApp
18    }
19  }
```

6. Buat plan_screen.dart

```

lib > views > plan_screen.dart > _PlanScreenState > build
12  class _PlanScreenState extends State<PlanScreen> {
13      void initState() {
14      }
15
16
17
18
19
20
21
22
23
24  }
25
26  @override
27  void dispose() {
28      scrollController.dispose();
29      super.dispose();
30  }
31
32  @override
33  Widget build(BuildContext context) {
34      return Scaffold(
35          appBar: AppBar(
36              backgroundColor: Theme.of(context).colorScheme.primary,
37              title: const Text('Master Plan Krisna Andika')), // AppBar
38          body: buildList(),
39          floatingActionButton: _buildAddTaskButton(); // Scaffold
40      }

```

7. buat method _buildAddTaskButton()

```

53
54  Widget _buildAddTaskButton() {
55      return FloatingActionButton(
56          onPressed: () {
57              setState(() {
58                  plan = Plan(
59                      name: plan.name, tasks: List<Task>.from(plan.tasks)..add(Task())); // Plan
60              });
61          },
62          child: Icon(Icons.add), // Use 'const' with the constructor to improve performance.
63      ); // FloatingActionButton
64  }
65

```

8. buat widget _buildList()

```

41
42  Widget buildList() {
43      return ListView.builder(
44          controller: ScrollController(),
45          keyboardDismissBehavior:
46              Theme.of(context).platform == TargetPlatform.iOS
47                  ? ScrollViewKeyboardDismissBehavior.onDrag
48                  : ScrollViewKeyboardDismissBehavior.manual,
49          itemCount: plan.tasks.length,
50          itemBuilder: (context, index) =>
51              _buildTaskTile(plan.tasks[index], index)); // ListView.builder
52  }
53

```

9. buat widget _buildTaskTile

```

65
66   widget _buildTaskTile(Task task, int index) {
67     return ListTile(
68       leading: Checkbox(
69         value: task.complete,
70         onChanged: (selected) {
71           setState(() {
72             plan = Plan(
73               name: plan.name,
74               tasks: List<Task>.from(plan.tasks)
75                 ..[index] = Task(
76                   description: task.description,
77                   complete: selected ?? false,
78                 ), // Task
79             ); // Plan
80           });
81         }), // Checkbox
82       title: TextFormField(
83         initialValue: task.description,
84         onChanged: (text) {
85           setState(() {
86             plan = Plan(
87               name: plan.name,
88               tasks: List<Task>.from(plan.tasks)
89                 ..[index] = Task(
90                   description: text,
91                   complete: task.complete,
92                 ), // Task
93             ); // Plan
94           });
95         },
96       ), // TextFormField
97     ); // ListTile
98   }
99 }

```

10. Tambah Scroll Controller

```

10   }
11
12   class _PlanScreenState extends State<PlanScreen> {
13     Plan plan = Plan();
14
15     late ScrollController scrollController;
16

```

11. Tambah Scroll Listener

```

16
17 @override
18 void initState() {
19     super.initState();
20     scrollController = ScrollController()
21         ..addListener(() {
22             FocusScope.of(context).requestFocus(FocusNode());
23         });
24 }

```

12. Tambah controller dan keyboard behavior

```
1  
2 Widget buildList() {  
3   return ListView.builder(  
4     controller: ScrollController(),  
5     keyboardDismissBehavior:  
6       Theme.of(context).platform == TargetPlatform.iOS  
7         ? ScrollViewKeyboardDismissBehavior.onDrag  
8         : ScrollViewKeyboardDismissBehavior.manual,  
9     itemCount: plan.tasks.length,  
10    itemBuilder: (context, index) =>  
11      _buildTaskTile(plan.tasks[index], index)); // ListView.builder  
12  }  
13 }
```

13. tambah metode dispose()

```

16
17 @override
18 void initState() {
19     super.initState();
20     scrollController = ScrollController()
21         ..addListener(() {
22             FocusScope.of(context).requestFocus(FocusNode());
23         });
24 }

```

Soal 2

Untuk melakukan eksport pada models, sehingga proses import hanya dilakukan sekali pada views

Soal 3

Agar objek tidak dapat diubah lagi setelah pendeklarasian objek, sehingga pada tahap2 selanjutnya proses menjadi duplikasi objek baru.

Soal 4



membentuk sebuah listile yang didasarkan pada jumlah task pada plan, pada set state checkbox apa bila terjadi perubahan checkbox maka variabel pada task boolean menjadi true atau false. semua perubahan akan terjadi pada variabel plan, sehinga terjadi perubahan secara terus menerus pada variabel plan

soal 5

ScrollListener digunakan apabila, tampilan melebihi batasan device maka akan dilakukan listener pada scroll seperti fokus.

Praktikum 2

soal 1

1. Buat file plan_provider.dart

```
readme.md  plan_provider.dart  plan_screen.dart  main.dart  D:\...\flutter_plugin_pubdev\...
b > provider > plan_provider.dart > PlanProvider > of
1  import 'package:flutter/material.dart';
2  import '../models/data_layer.dart';
3
4  class PlanProvider extends InheritedNotifier<ValueNotifier<Plan>> {
5    const PlanProvider( Convert 'child' to a super parameter.
6      {super.key, required Widget child, required ValueNotifier<Plan> notifier})
7      : super(child: child, notifier: notifier);
8
9    static ValueNotifier<Plan> of(BuildContext context) {
10      return context
11        .dependOnInheritedWidgetOfExactType<PlanProvider>()!
12        .notifier!;
13    }
14  }
```

2. Edit main.dart

```

lib > main.dart > MasterPlanApp > build
1  import 'package:flutter/material.dart';
2  import 'package:master_plan/models/plan.dart';
3  import 'package:master_plan/provider/plan_provider.dart';
4  import 'package:master_plan/views/plan_screen.dart';
5
6  Run | Debug | Profile
7  void main() {
8    runApp(const MasterPlanApp());
9  }
10
11 class MasterPlanApp extends StatelessWidget {
12   const MasterPlanApp({super.key});
13
14   @override
15   Widget build(BuildContext context) {
16     return MaterialApp(
17       theme: ThemeData(primarySwatch: Colors.deepPurple),
18       home: PlanProvider(
19         child: PlanScreen(), notifier: ValueNotifier<Plan>(const Plan()), // PL
20       ); // MaterialApp
21   }
22 }

```

3. Tambahkan method pada model plan.dart

```

lib > models > plan.dart > ...
1  import 'package:master_plan/models/task.dart';
2
3  class Plan {
4    final String name;
5    final List<Task> tasks;
6
7    const Plan({this.name = '', this.tasks = const []});
8
9    int get completedCount => tasks.where((task) => task.complete).length;
10
11    String get completenessMessage =>
12      '$completedCount out of ${tasks.length} tasks';
13  }
14

```

4. pindah ke plan screen

5. Edit method _buildAddTaskButton

```

61
62   Widget _buildAddTaskButton() {
63     ValueNotifier<Plan> planNotifier = PlanProvider.of(context);
64     return FloatingActionButton(
65       onPressed: () {
66         setState(() {
67           Plan currentPlan = planNotifier.value;
68           planNotifier.value = Plan(
69             name: currentPlan.name,
70             tasks: List<Task>.from(currentPlan.tasks)..add(const T
71         ));
72       },
73       child: Icon(Icons.add), // Use 'const' with the constructor to
74     ); // FloatingActionButton
75   }
76

```

6. Edit method _buildTaskTile

```

76
77   Widget _buildTaskTile(Task task, int index) {
78     ValueNotifier<Plan> planNotifier = PlanProvider.of(context);
79     return ListTile(
80       leading: Checkbox(
81         value: task.complete,
82         onChanged: (selected) {
83           Plan currentPlan = planNotifier.value;
84           planNotifier.value = Plan(
85             name: currentPlan.name,
86             tasks: List<Task>.from(currentPlan.tasks)
87               ..[index] = Task(
88                 description: task.description,
89                 complete: selected ?? false)); // Task // Plan
90         }, // Checkbox
91       title: TextFormField(
92         initialValue: task.description,
93         onChanged: (text) {
94           Plan currentPlan = planNotifier.value;
95           planNotifier.value = Plan(
96             name: currentPlan.name,
97             tasks: List<Task>.from(currentPlan.tasks)
98               ..[index] = Task(description: text, complete: task.complete)); // Plan
99         },
100     ), // TextFormField
101   ); // ListTile
102 }
103

```

7. Edit _buildList

```

49
50 Widget buildList(Plan plan) {
51   return ListView.builder(
52     controller: ScrollController(),
53     keyboardDismissBehavior:
54       Theme.of(context).platform == TargetPlatform.iOS
55       ? ScrollViewKeyboardDismissBehavior.onDrag
56       : ScrollViewKeyboardDismissBehavior.manual,
57     itemCount: plan.tasks.length,
58     itemBuilder: (context, index) =>
59       _buildTaskTile(plan.tasks[index], index)); // ListView.builder
60   }
61

```

8. Tetap di class PlanScreen

```

return Column(
  children: [
    Expanded(child: buildList(plan)),
    SafeArea(child: Text(plan.completenessMessage))
  ],
); // Column
}), // ValueListenableBuilder
floatingActionButton: _buildAddTaskButton()); // Scaffold

```

soal 2

- Inherited Widget :

```

8
9 static ValueNotifier<Plan> of(BuildContext context) {
10   return context
11     .dependOnInheritedWidgetOfExactType<PlanProvider>()!
12     .notifier!;
13 }

```

Digunakan untuk menyediakan data, sehingga data bisa di akses ke widget child.

- Inherited Notifier!

```

class PlanProvider extends InheritedNotifier<ValueNotifier<Plan>> {
  const PlanProvider({
    required ValueNotifier<Plan> notifier,
    super.child,
  }) : super(notifier: notifier, child: child);
}

```

Digunakan untuk me- listen sebuah objek yang ketika objek itu berubah maka, akan melakukan rebuild widget sehingga data sesuai.

Praktikum 3

1. Edit PlanProvider

```

lib > provider > plan_provider.dart > PlanProvider > of

1  import 'package:flutter/material.dart';
2  import '../models/data_layer.dart';
3
4  class PlanProvider extends InheritedNotifier<ValueNotifier<List<Plan>>> {
5    const PlanProvider( Convert 'child' to a super parameter.
6      {super.key,
7        required Widget child,
8        required ValueNotifier<List<Plan>> notifier})
9      : super(child: child, notifier: notifier);
10
11    static ValueNotifier<List<Plan>> of(BuildContext context) {
12      return context
13        .dependOnInheritedWidgetOfExactType<PlanProvider>()!
14        .notifier!;
15    }
16  }
17

```

2. Edit main.dart

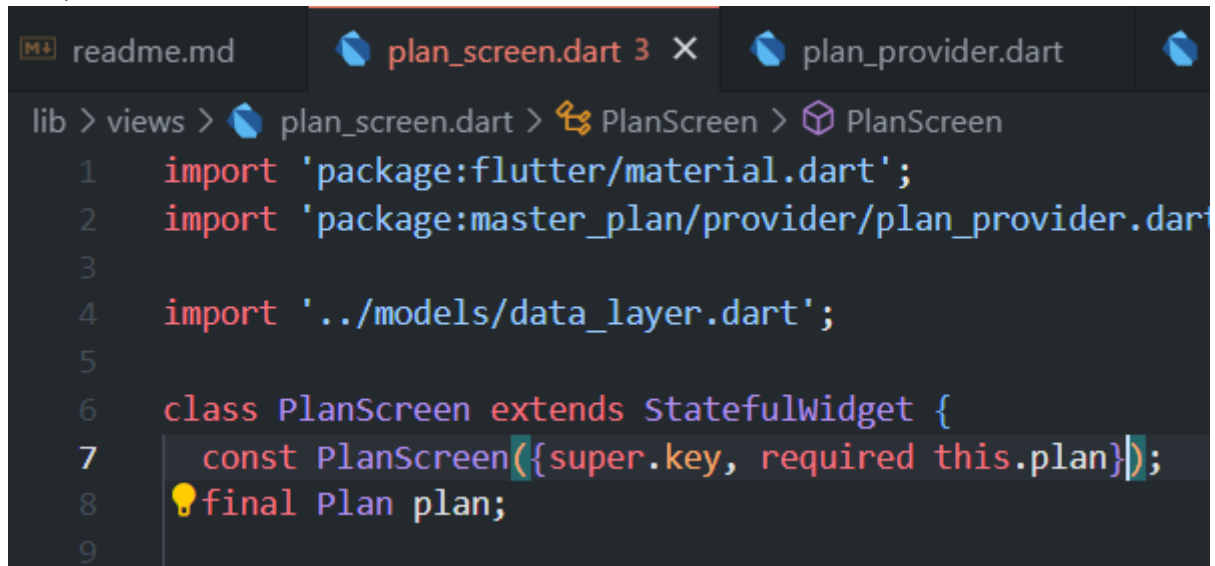
```

lib > main.dart > MasterPlanApp > build

1  import 'package:master_plan/models/plan.dart';
2  import 'package:master_plan/provider/plan_provider.dart';
3  import 'package:master_plan/views/plan_screen.dart';
4
5  Run | Debug | Profile
6  void main() {
7    runApp(const MasterPlanApp());
8  }
9
10 class MasterPlanApp extends StatelessWidget {
11   const MasterPlanApp({super.key});
12
13   @override
14   Widget build(BuildContext context) {
15     return PlanProvider(
16       child: MaterialApp( The 'child' argument should be last in widget constructor invocation
17         title: "State Management App",
18         theme: ThemeData(
19           primarySwatch: Colors.blue,
20         ), // ThemeData
21         home: const PlanScreen(),
22       ), // MaterialApp
23       notifier: ValueNotifier<List<Plan>>(const [])); // PlanProvider
24   }
25 }
26

```

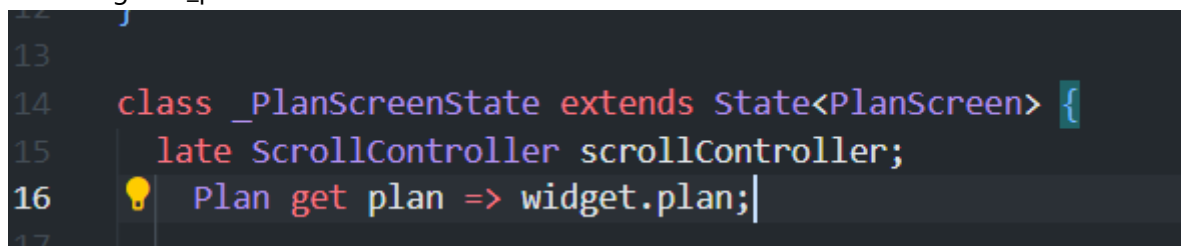
3. Edit plan_screen.dart



```
lib > views > plan_screen.dart > PlanScreen > PlanScreen
1  import 'package:flutter/material.dart';
2  import 'package:master_plan/provider/plan_provider.dart';
3
4  import '../models/data_layer.dart';
5
6  class PlanScreen extends StatefulWidget {
7    const PlanScreen({super.key, required this.plan});
8    final Plan plan;
9
```

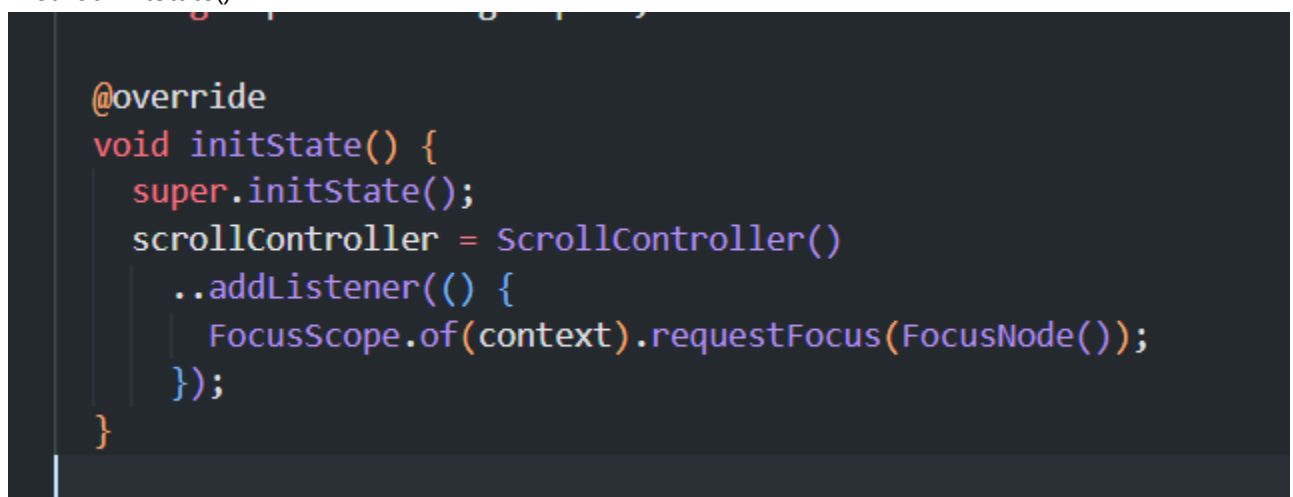
4. Error

5. Tambah getter_plan



```
13
14  class _PlanScreenState extends State<PlanScreen> {
15    late ScrollController scrollController;
16    Plan get plan => widget.plan;
17
```

6. Method initState()



```
@override
void initState() {
  super.initState();
  scrollController = ScrollController()
    ..addListener(() {
      FocusScope.of(context).requestFocus(FocusNode());
    });
}
```

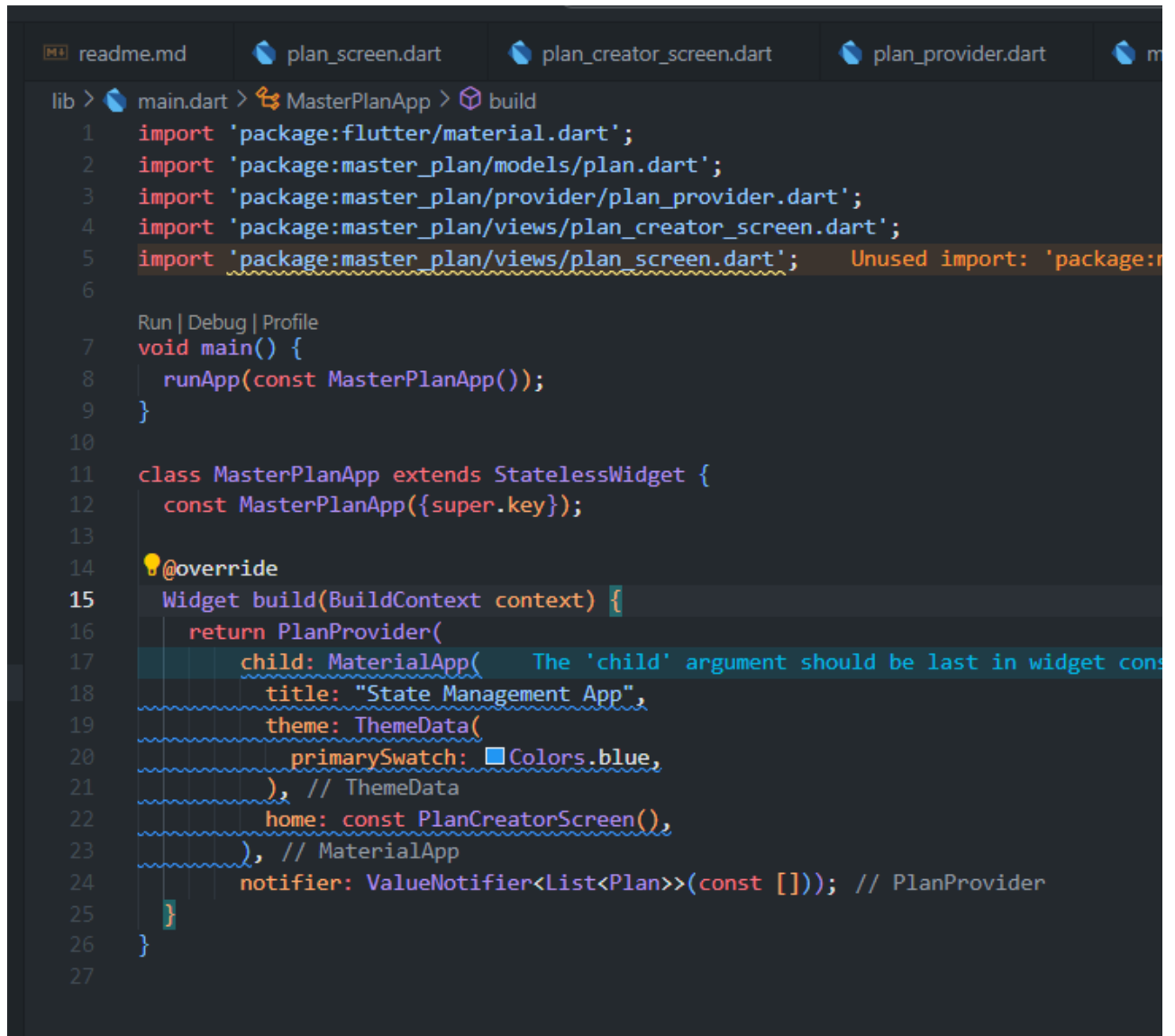
7. Widget Build

```
32  
33     @override  
34     Widget build(BuildContext context) {  
35         ValueNotifier<List<Plan>> plansNotifier = PlanProvider.of(context);  
36         return Scaffold(  
37             appBar: AppBar(  
38                 backgroundColor: Theme.of(context).colorScheme.primary,  
39                 title: const Text('Master Plan Krisna Andika')), // AppBar  
40             body: ValueListenableBuilder<List<Plan>>(  
41                 valueListenable: plansNotifier,  
42                 builder: (context, plans, child) {  
43                     Plan currentPlan = plans.firstWhere((p) => p.name == plan.name);  
44                     return Column(  
45                         children: [  
46                             Expanded(child: buildList(currentPlan)),  
47                             SafeArea(child: Text(currentPlan.completenessMessage))  
48                         ],  
49                     ); // Column  
50                 }), // ValueListenableBuilder  
51             floatingActionButton: _buildAddTaskButton(context)); // Scaffold  
52     }
```

8. Edit_buildTaskTile

```
84
85 Widget _buildTaskTile(Task task, int index, BuildContext context) {
86   ValueNotifier<List<Plan>> planNotifier = PlanProvider.of(context);
87
88   return ListTile(
89     leading: Checkbox(
90       value: task.complete,
91       onChanged: (selected) {
92         Plan currentPlan = plan;
93         int planIndex = planNotifier.value
94           .indexWhere((p) => p.name == currentPlan.name);
95         planNotifier.value = List<Plan>.from(planNotifier.value)
96           ..[planIndex] = Plan(
97             name: currentPlan.name,
98             tasks: List<Task>.from(currentPlan.tasks)
99               ..[index] = Task(
100                 description: task.description,
101                 complete: selected ?? false,
102               ), // Task
103           ); // Plan
104       }), // Checkbox
105     title: TextFormField(
106       initialValue: task.description,
107       onChanged: (text) {
108         Plan currentPlan = plan;
109         int planIndex =
110           planNotifier.value.indexWhere((p) => p.name == currentPlan.name);
111         planNotifier.value = List<Plan>.from(planNotifier.value)
112           ..[planIndex] = Plan(
113             name: currentPlan.name,
114             tasks: List<Task>.from(currentPlan.tasks)
115               ..[index] = Task(
116                 description: text,
117                 complete: task.complete,
118               ), // Task
119             ); // Plan
120       },
121     ), // TextFormField
122   ); // ListTile
123 }
124
125
```

9. Buat screen menu baru



```
lib > main.dart > MasterPlanApp > build
1 import 'package:flutter/material.dart';
2 import 'package:master_plan/models/plan.dart';
3 import 'package:master_plan/provider/plan_provider.dart';
4 import 'package:master_plan/views/plan_creator_screen.dart';
5 import 'package:master_plan/views/plan_screen.dart'; Unused import: 'package:
6
Run | Debug | Profile
7 void main() {
8   runApp(const MasterPlanApp());
9 }
10
11 class MasterPlanApp extends StatelessWidget {
12   const MasterPlanApp({super.key});
13
14   @override
15   Widget build(BuildContext context) {
16     return PlanProvider(
17       child: MaterialApp( The 'child' argument should be last in widget cons
18         title: "State Management App",
19         theme: ThemeData(
20           primarySwatch: Colors.blue,
21         ), // ThemeData
22         home: const PlanCreatorScreen(),
23       ), // MaterialApp
24       notifier: ValueNotifier<List<Plan>>(const [])); // PlanProvider
25   }
26 }
27
```

10. Pindah ke class `_PlanCreatorScreenState`

```
lib > views > plan_creator_screen.dart > _PlanCreatorScreenState > _buildListCreator
1  import 'package:flutter/material.dart';
2  import 'package:master_plan/models/data_layer.dart';
3  import 'package:master_plan/provider/plan_provider.dart';
4  import 'package:master_plan/views/plan_screen.dart';
5
6  class PlanCreatorScreen extends StatefulWidget {
7    const PlanCreatorScreen({super.key});
8
9    @override
10   State<PlanCreatorScreen> createState() => _PlanCreatorScreenState();
11 }
12
13 class _PlanCreatorScreenState extends State<PlanCreatorScreen> {
14   final textController = TextEditingController();
15
16   void addPlan() {
17     final text = textController.text;
18     if (text.isEmpty) {
19       return;
20     }
21
22     final plan = Plan(name: text, tasks: []);
23
24     ValueNotifier<List<Plan>> planNotifier = PlanProvider.of(context);
25
26     planNotifier.value = List<Plan>.from(planNotifier.value)..add(plan);
27
28     textController.clear();
29     FocusScope.of(context).requestFocus(FocusNode());
30
31     setState(() {});
32   }
33
34   @override
35   void dispose() {
36     textController.dispose();
37     super.dispose();
38   }
39 }
```


11. Pindah ke method build

```
39
40 @override
41 Widget build(BuildContext context) {
42   return Scaffold(
43     appBar: AppBar(
44       title: const Text('Master Plans Namaku'),
45     ), // AppBar
46     body: Column(
47       children: [_buildListCreator(), Expanded(child: _buildMasterPlans())],
48     ), // Column
49   ); // Scaffold
50 }
```

12. Buat widget _buildListCreator

```
Widget _buildListCreator() {
  return Padding(
    padding: const EdgeInsets.all(20.0),
    child: Material(
      color: Theme.of(context).cardColor,
      elevation: 10,
      child: TextField(
        controller: textController,
        decoration: const InputDecoration(
          labelText: 'Add a plan', contentPadding: EdgeInsets.all(20)), // InputDecoration
        onEditingComplete: addPlan, // TextField
      )); // Material // Padding
}
```

13. Buat void addPlan()

```
void addPlan() {
  final text = textController.text;
  if (text.isEmpty) {
    return;
  }

  final plan = Plan(name: text, tasks: []);

  ValueNotifier<List<Plan>> planNotifier = PlanProvider.of(context);

  planNotifier.value = List<Plan>.from(planNotifier.value)..add(plan);

  textController.clear();
  FocusScope.of(context).requestFocus(FocusNode());

  setState(() {});
}
```

14. Buat widget _buildMasterPlans

```
66 Widget _buildMasterPlans() {
67   ValueNotifier<List<Plan>> planNotifier = PlanProvider.of(context);
68   List<Plan> plans = planNotifier.value;
69
70   if (plans.isEmpty) {
71     return Column(
72       mainAxisAlignment: MainAxisAlignment.center,
73       children: <Widget>[
74         const Icon(Icons.note, size: 100, color: Colors.grey),
75         Text('Anda belum memiliki rencana apapun.',
76           style: Theme.of(context).textTheme.headlineSmall) // Text
77       ]); // <Widget>[] // Column
78   }
79   return ListView.builder(
80     itemCount: plans.length,
81     itemBuilder: (context, index) {
82       final plan = plans[index];
83       return ListTile(
84         title: Text(plan.name),
85         subtitle: Text(plan.completenessMessage),
86         onTap: () {
87           Navigator.of(context).push(MaterialPageRoute(
88             builder: (_) => PlanScreen(
89               plan: plan,
90             )); // PlanScreen // MaterialPageRoute
91         }); // ListTile
92     }); // ListView.builder
93   }
94 }
95
```

Hasil

