R2S

Resource Software Solution

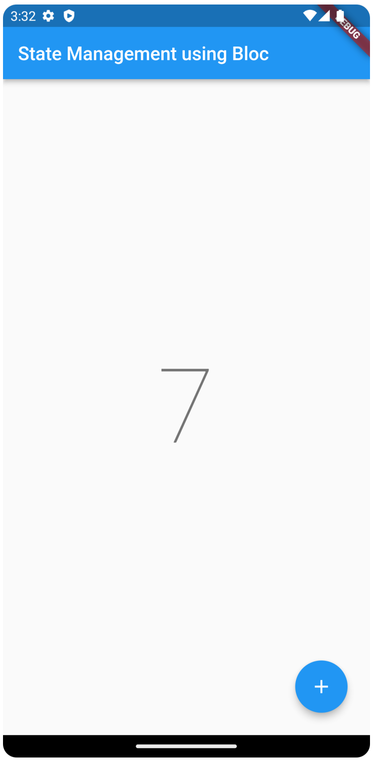
Flutter

Training Assignments

State management using Bloc

Overview

In the following tutorial, we're going to build a Counter in Flutter using the Bloc library.



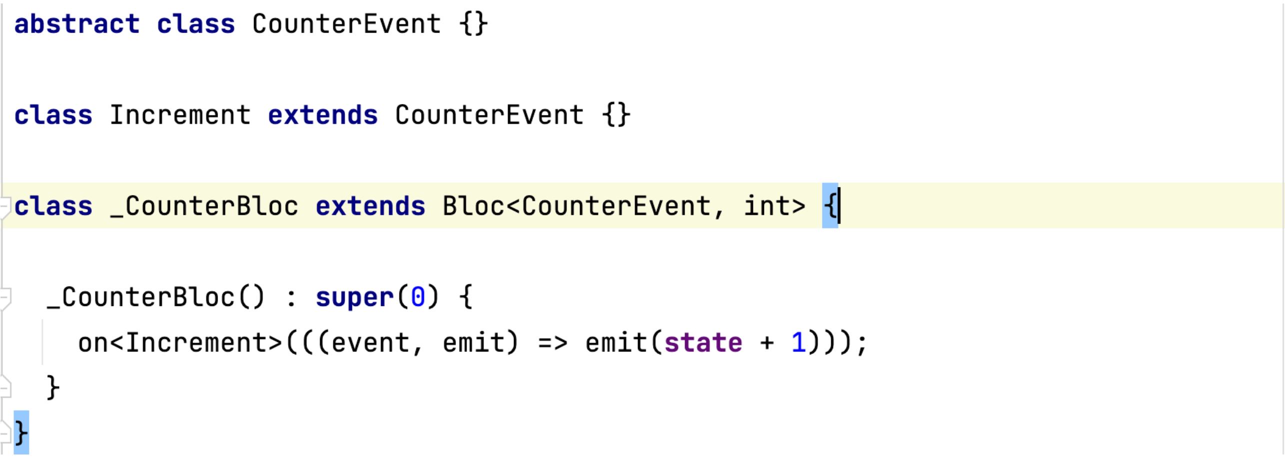
Tasks

1. Setup

We can then go ahead and replace the contents of pubspec.yaml with

```
flutter_bloc: ^8.1.1
```

2. Creating a Bloc (CounterBloc) and register event handlers (CounterEvent)



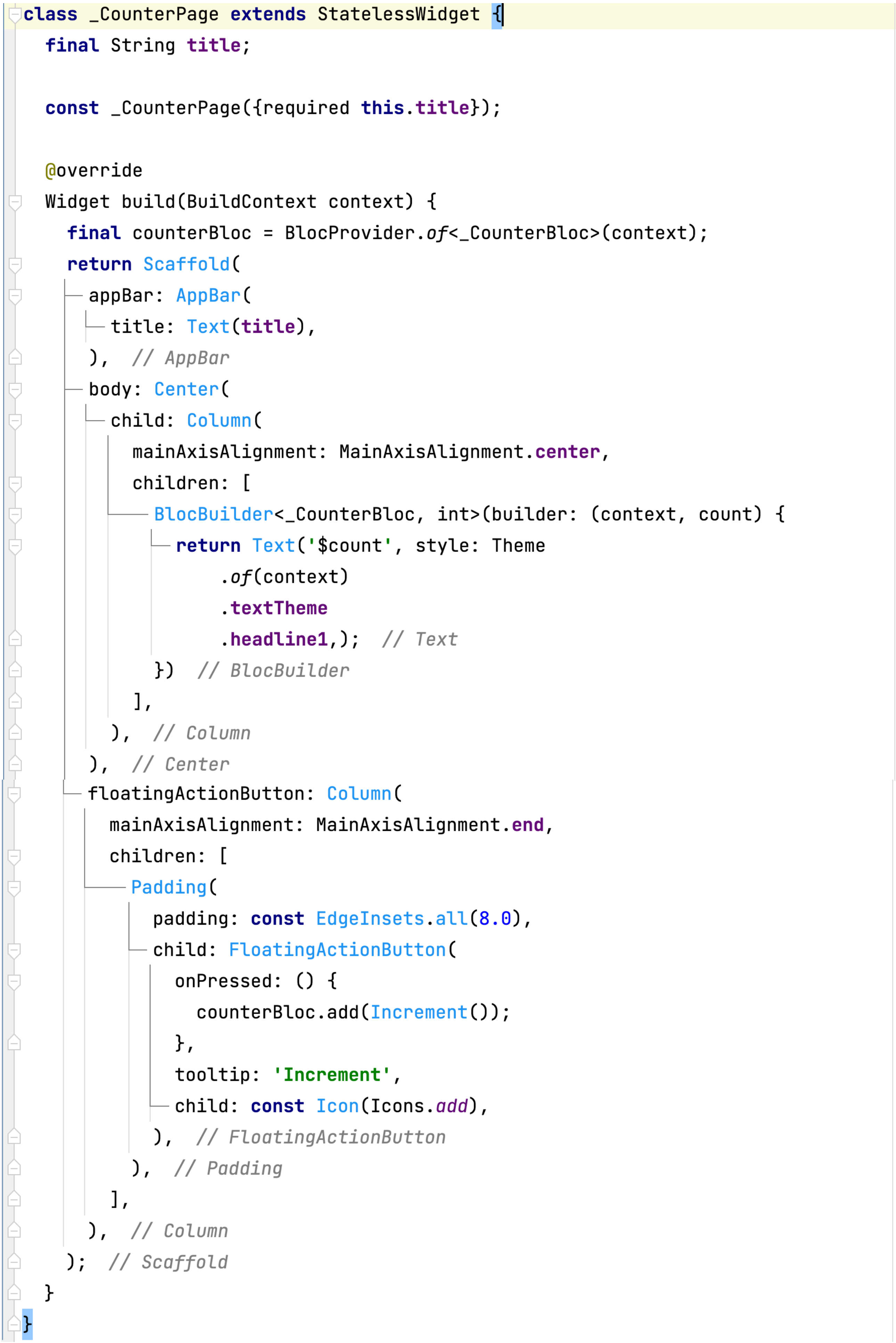
```
abstract class CounterEvent {}

class Increment extends CounterEvent {}

class _CounterBloc extends Bloc<CounterEvent, int> {

  _CounterBloc() : super(0) {
    on<Increment>(((event, emit) => emit(state + 1)));
  }
}
```

3. Counter View

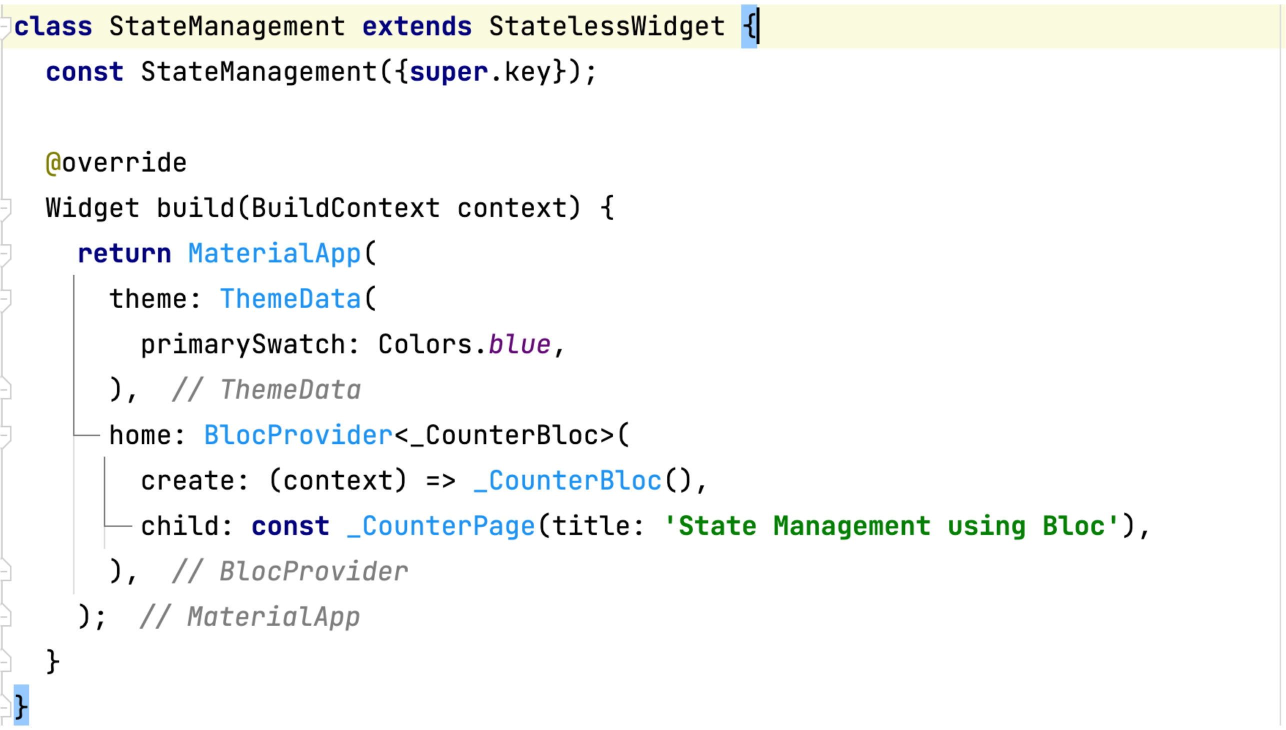


```
class _CounterPage extends StatelessWidget {
  final String title;

  const _CounterPage({required this.title});

  @override
  Widget build(BuildContext context) {
    final counterBloc = BlocProvider.of<_CounterBloc>(context);
    return Scaffold(
      appBar: AppBar(
        title: Text(title),
      ),  // AppBar
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            BlocBuilder<_CounterBloc, int>(builder: (context, count) {
              return Text('$count', style: Theme
                  .of(context)
                  .textTheme
                  .headline1,);  // Text
            })  // BlocBuilder
          ],
        ),  // Column
      ),  // Center
      floatingActionButton: Column(
        mainAxisAlignment: MainAxisAlignment.end,
        children: [
          Padding(
            padding: const EdgeInsets.all(8.0),
            child: FloatingActionButton(
              onPressed: () {
                counterBloc.add(Increment());
              },
              tooltip: 'Increment',
              child: const Icon(Icons.add),
            ),  // FloatingActionButton
          ),  // Padding
        ],
      ),  // Column
    );  // Scaffold
  }
}
```

4. Counter Page



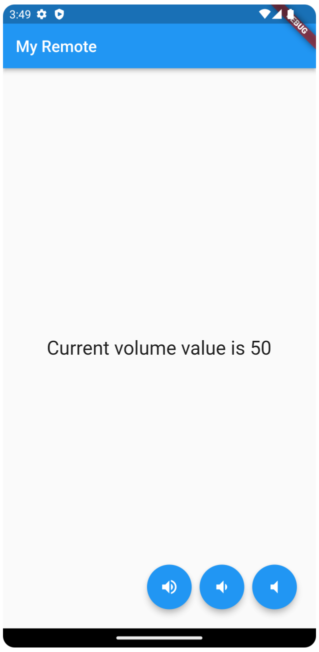
```
class StateManagement extends StatelessWidget {
  const StateManagement({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),  // ThemeData
      home: BlocProvider<_CounterBloc>(
        create: (context) => _CounterBloc(),
        child: const _CounterPage(title: 'State Management using Bloc'),
      ),  // BlocProvider
    );  // MaterialApp
  }
}
```

5. Run the App

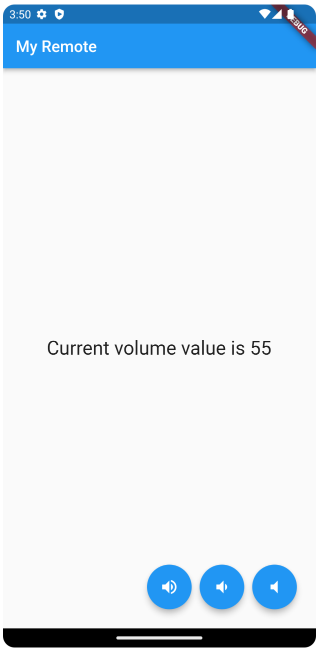
Extra tasks

This requirements here is to simulate a TV remote control with 3 simple buttons: volume up, volume down and mute.



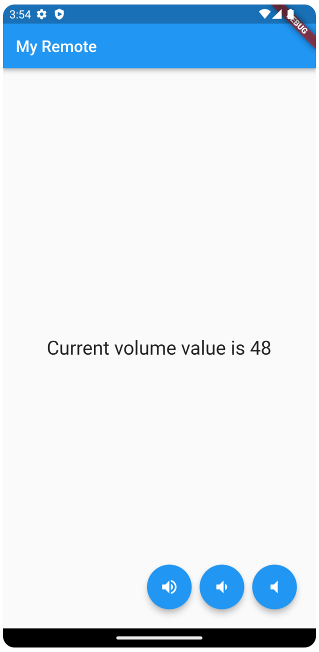
Current volume value is 50

Figure 1: Run



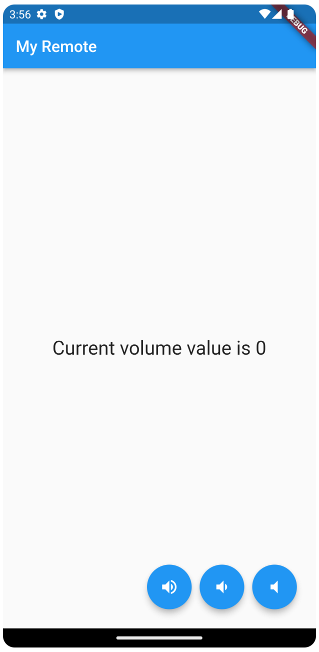
Current volume value is 55

Figure 2: Increment volume



Current volume value is 48

Figure 3: Decrement volume



Current volume value is 0

Figure 4: Mute volume

--THE END—