**Experiment No 04**

**Aim**: Create application that illustrate State management using bloc and provider

**Code**:

***Using Provider:***

***main.dart***

import 'dart:developer';

import 'package:flutter/material.dart';

import 'model.dart';

import "package:provider/provider.dart";

void main() {

runApp(ChangeNotifierProvider<Counter\_Model>(

create: ((context) {

return Counter\_Model();

}),

child: const DemoPage()));

}

class DemoPage extends StatelessWidget {

const DemoPage();

@override

Widget build(BuildContext context) {

return MaterialApp(

theme: ThemeData.from(colorScheme: const ColorScheme.light()),

home: const DemoPageChild());

}

}

class DemoPageChild extends StatelessWidget {

const DemoPageChild();

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(title: Text('Provider Counter-Aakash Dhotre')),

body: const DemoPageChildWidget());

}

}

class DemoPageChildWidget extends StatelessWidget {

const DemoPageChildWidget();

@override

Widget build(BuildContext context) {

return Padding(

padding: EdgeInsets.only(top: 15.0),

child: Row(children: [

Expanded(child:

Consumer<Counter\_Model>(builder: ((context, countermodel, child) {

return ElevatedButton(

onPressed: () {

countermodel.increament();

},

child: const Text(

"+",

textScaleFactor: 3,

));

}))),

Expanded(child: Center(child: const ShowCounter())),

Expanded(child:

Consumer<Counter\_Model>(builder: ((context, countermodel, child) {

return ElevatedButton(

onPressed: () {

countermodel.decrement();

},

child: const Text(

"-",

textScaleFactor: 3,

));

})))

]));

}

}

class ShowCounter extends StatelessWidget {

const ShowCounter();

@override

Widget build(BuildContext context) {

return Consumer<Counter\_Model>(builder: ((context, countermodel, child) {

return Text(countermodel.counter.toString(), textScaleFactor: 3);

}));

}

}

***model.dart***

import 'package:flutter/material.dart';

class Counter\_Model with ChangeNotifier {

int \_counter = 0;

int get counter => \_counter;

void increament() {

\_counter++;

notifyListeners();

}

void decrement() {

\_counter--;

notifyListeners();

}

}

***Using bloc:***

import 'package:flutter/material.dart';

import 'package:flutter\_bloc/flutter\_bloc.dart';

void main() {

runApp(BlocProvider<CounterBloc>(

create: (context) => CounterBloc(),

child: MaterialApp(

debugShowCheckedModeBanner: false,

title: "Bloc Counter",

home: Scaffold(

appBar: AppBar(title: Text("Bloc Counter-Aakash Dhotre")),

body: counter\_body(),

),

)));

}

class counter\_body extends StatelessWidget {

@override

Widget build(BuildContext context) {

return Center(

child: Row(children: <Widget>[

Expanded(

child: ElevatedButton(

onPressed: () {

BlocProvider.of<CounterBloc>(context).add(Increment());

//below line also works instead of above line

//context.read<CounterBloc>().add(Increment());

},

child: Text(

"+",

textScaleFactor: 4,

),

style: ElevatedButton.styleFrom(fixedSize: const Size(240, 80)))),

Expanded(child: Center(

child: BlocBuilder<CounterBloc, int>(builder: (context, count) {

return Text(

'$count',

textScaleFactor: 4,

style: TextStyle(fontSize: 24.0),

);

}))),

Expanded(

child: ElevatedButton(

onPressed: () {

BlocProvider.of<CounterBloc>(context).add(Decrement());

//below line also works instead of above line

// context.read<CounterBloc>().add(Decrement());

},

child: Text(

"-",

textScaleFactor: 4,

),

style: ElevatedButton.styleFrom(fixedSize: const Size(240, 80))))

]));

}

}

abstract class CounterEvent {}

class Increment extends CounterEvent {}

class Decrement extends CounterEvent {}

class CounterBloc extends Bloc<CounterEvent, int> {

void onChange(

Change<int> change) //overriding function executed when there is a change

{

debugPrint("$change");

super.onChange(change);

}

CounterBloc() : super(0) {

on<Increment>((event, emit) => emit(state + 1));

on<Decrement>((event, emit) => emit(state - 1));

}

}

**Output**:

Graphical user interface, waterfall chart

Description automatically generated Teams

Description automatically generated with medium confidence